A Comparison of C++, C#, Java, and PHP in the context of e-learning

MIKAEL OLSSON

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2009

TRITA-ICT-EX-2009:8

A Comparison of C++, C#, Java, and PHP in the context of e-learning

Mikael Olsson

April 30, 2009

Master's Thesis in Computer Science

Royal Institute of Technology

Examiner: Prof. Gerald Q. Maguire Jr.

Abstract

The first part of this master thesis presents an effective method for producing video tutorials. This method was used during this thesis project to create tutorials on the e-learning site PVT (http://www.programmingvideotutorials.com). Part one also discloses how the production method was developed and how tutorials produced using this method compare to professional video tutorials. Finally, it evaluates the result of this thesis work and the efficiency of the production method.

The second part of this thesis compares the syntactical similarities and differences between four of the languages taught via video tutorials for PVT. These languages are: C++, C#, Java, and PHP. The purpose of this comparison is to provide a bridge for programmers knowing one of these languages to rapidly learn one or more of the other languages. The reason why this would be necessary is because there is no single language suited for every area of software development. Knowing a multitude of languages gives a programmer a wider range of job opportunities and more choices in how to solve their problems. Part two of the thesis also includes a comparison of Java and C# in the context of a video tutorial series that shows how to build a basic text editor.

Sammanfattning

Den första delen av denna examensredovisning beskriver en effektiv metod för att producera videokurser. Denna metod har använts under detta examensarbete för att skapa kurser på utbildningssajten PVT (http://www.programmingvideotutorials.com). Del ett berättar också hur produktionsmetoden utvecklades och hur kurser producerade enligt denna metod står sig mot professionellt skapade videokurser. Slutligen utvärderas resultatet av examensarbetet och effektiviteten av produktionsmetoden.

Den andra delen av denna redovisning framför de syntaktiska likheterna och olikheterna mellan fyra av de språk som har lärts ut via videokurser på PVT. Dessa språk är: C++, C#, Java, och PHP. Meningen med denna jämförelse är att underlätta för programmerare som kan ett av dessa språk och som snabbt vill lära sig ett eller flera av de andra språken. Anledningen till varför detta skulle vara nödvändigt är för att det inte finns något enstaka språk anpassat till alla områden av programutveckling. Att kunna ett flertal språk ger en programmerare ett bredare utbud av jobbmöjligheter och mer val i hur han eller hon kan lösa sina problem. Del två av redovisningen inkluderar också en jämförelse av Java och C# i sammanhanget av en videokurs som visar hur man bygger en enkel text editor.

Table of Contents

List of Tables

List of Examples

List of Acronyms and Abbreviations

AVI	Audio Video Interleave – A common multimedia container format.
e-learning	Education through the use of computer technology.
ASP.NET	A programming framework for building web-based applications.
BASIC	A family of high-level programming languages.
CBT	Computer-based Training – Education through the use of training programs.
Codec	Coder-decoder – A program used to encode and decode data.
CLR	Common Language Runtime – The virtual machine of Microsoft .NET.
DVD	Digital Versatile Disc – A popular optical disc storage media format.
Flash	A popular technology for adding animation and interactivity to web sites.
GUI	Graphical User Interface – A type of human-computer interface.
HTML	Hypertext Markup Language – The standard language for creating web pages.
IDE	Integrated Development Environment – A software development environment.
iPhone	An internet-connected multimedia smartphone created by Apple Inc.
kbit/s	Kilobits per second – A unit of data transfer rate equal to 1,000 bits per second.
kHz	Kilohertz – A unit of frequency equal to 1,000 periods per second.
LAME	An open source program used to encode audio into MP3.
MP3	MPEG-1 Audio Layer 3 – A common digital audio encoding format.
MSDN	Microsoft Developer Network – An information service from Microsoft for software developers.
MVT	Motionless Video Tutorial – The video tutorial production method developed for this thesis project.
PHP	PHP Hypertext Preprocessor – A scripting language primarily used for producing dynamic web pages.
PNG	Portable Network Graphics – A lossless bitmap image format.
PPT	PowerPoint Presentation – A file format used to store presentations in Microsoft PowerPoint.
PVT	Programming Video Tutorials – The e-learning site where the video tutorials produced in this thesis project can be found.
SCORM	Sharable Content Object Reference Model – A collection of standards for web-based e-learning.
SWF	Shockwave Flash – The dominant format for displaying animated and interactive vector graphics on the Internet.
VLE	Virtual Learning Environment – A program facilitating e-learning.
XML	Extensible Markup Language – A language used to structure and store data.
ZIP	A popular data compression and archival format.

1. Introduction

A video tutorial is a type of e-learning in which the student learns about a topic by watching a video. It is a particularly effective medium for teaching skills, because the student not only observes as the skill is demonstrated and explained, but he or she can also follow along with the teacher by performing the skill themselves. Essentially, video tutorials provide many of the same benefits as having a real live teacher, but with the added advantage that the video tutorial can be paused, rewound, and re-watched – however many times the student wants. From the teacher's standpoint the video tutorial medium also has a lot of leverage in that once a video tutorial has been produced it can be distributed *worldwide* using the Internet at almost no cost.

The objectives of this thesis project were to produce video tutorials in the area of computer programming and to upload these tutorials to the e-learning site PVT[1]. The main tutorials to be created were intended to teach the programming languages C++, C#, Java, PHP, and ASP.NET; as well as to demonstrate how to build a basic text editor in each of these languages. The uploaded tutorials were to be made available in three different formats: a downloadable AVI version, a streamable Adobe Flash version, and a SCORM version that could be used with a Virtual Learning Environment (VLE). The audio scripts and code examples used in the tutorials were also to be made available on the e-learning site (in this case as a wiki)[2].

Chapter two will explain how the production method used in this project was developed. Typically, the standard method of producing video tutorials involves recording the screen content using screen capture software, such as TechSmith Corporation's Camtasia Recorder[3]. As shown in chapter two this method has certain weaknesses, mainly because both the audio and the video have to be recorded live. Therefore, a new method was developed based upon studying the approaches used by several large video tutorial companies and through trial and error. In this new method the video consists of a series of snapshots that are synchronized to the separately recorded audio using TechSmith Corporation's Camtasia Studio. This method is similar to the non-linear editing (NLE) method used for film postproduction, which allows video frames to be moved around in time without any loss in quality.

In chapter three all the steps in the fully developed production method are explained in detail – in the context of how they were used during the project. This method contains six steps. The first step is to outline what video sections the tutorial will contain and in which order the sections will appear. The rest of the steps are repeated in a cycle for each of the outlined video sections. The second step is to compile an audio script for the section. The third step is to create practical examples based on this script. The fourth step is to use the examples to create snapshot images. Finally, the fifth step is to record and edit the audio. And the sixth step is to synchronize the snapshots with the video and produce the finished product.

In chapter four the results of this thesis project are evaluated. This evaluation includes what kinds of video sections were completed during the project. This chapter also assesses the efficiency of each step in the production method as well as the time needed to update a tutorial section.

Chapters five to eight compare the syntactic similarities and differences between four of the languages taught via video tutorials on PVT, namely: C++, C#, Java, and PHP. These programming languages were chosen because they are among the most widely used today[4] and all share similar syntax. However, although their syntax may be similar the languages have numerous important differences in both syntax and semantics. This comparison is based not only on the finished tutorials uploaded to PVT during the project, but also on scripts for unfinished tutorials that have yet to be produced. While effort has been taken to make this comparison as complete and accurate as possible, its intention is not to address every possible syntax variation in the four languages; as the subject area is simply too broad. Instead, the main objective

is to cover the most commonly used syntax – thus enabling a programmer to leverage their existing knowledge of one of these languages in order to quickly begin to program in another of these languages.

Chapter nine compares the software development tutorials that were finished in this project, which show how to design and implement a basic text editor. The solutions used for making this text editor are compared section by section using the two languages for which the tutorials were finished: C# and Java. This chapter also explains how creating these tutorials were different from producing the programming syntax tutorials.

Chapter ten presents some overall conclusions from this thesis project and suggests some future work that should be undertaken.

2. Developing a Production Method

In order to determine a suitable method of video tutorial production the tutorials of several large e-learning companies were studied. These companies include: 3dbuzz[5], AppDev[6], CBT Nuggets[7], Keystone[8], Learn Visual Studio .NET[9], Learnkey[10], Lynda[11], Total Training[12], and Virtual Training Company[13]. Through this research and my own trials a unique six-step method of producing tutorials was developed (See chapter 3). This new method focuses on producing tutorials that are superior to those I have studied in four different ways – creating tutorials that are updatable, more convenient to use, quicker to produce, and more relevant to their subject.

2.1 Updatable

One of the main goals of the production method was that each tutorial would be created in a way that allowed the tutorial to be easily updated and changed. This idea that a tutorial would continue to be updated and improved after its initial production was unique among the tutorials production methods I studied. Therefore, a radical new approach to video tutorial production was required.

Making tutorials that are easy to update requires three things. First, each video section has to be stand-alone, so that it can be changed independently of any other section. If the video sections in a tutorial were not standalone, then editing one section would require that all sections that depended on the edited section would have to be redone.

The second condition for making updatable tutorials is that the material needed to reproduce a tutorial must be stored in a lossless fashion. Otherwise a video section would not be considered updatable, because whenever the snapshots or the audio that make up a section would be edited the quality of the reproduced video would deteriorate.

The third condition is that it must be possible to edit the video and audio separately from each other. If the audio and video were recorded at the same time they could not be edited independently and the whole section would generally have to be re-recorded in order to make an update. The solution for this dilemma was to ensure that the video consists of a series of distinct snapshots. These snapshots could easily be produced and edited separately from the audio, then they could both be synchronized together.

As a result of the tutorials being updatable it was possible to employ mass collaboration[14] in improving the video tutorial scripts through the use of a wiki[2]. This means that anyone visiting the site can edit the audio scripts and code examples used to produce the tutorials. Thus, visitors can help to improve sections, fix errors, or even contribute with entirely new sections to the tutorials. This form of mass collaboration has not been used in the context of video tutorials in any of the e-learning companies that I studied and may prove to be an important change in how e-learning material is produced.

2.2 Convenient

Another goal for the production method was that the tutorials would be convenient to view and accessible for a wide range of users. One way in which this is achieved is by providing multiple formats for watching the videos. The tutorials are first of all available in the Adobe Flash format (.swf). This format is streamed online from the site[1] which guarantees that the student will always see the latest version of the tutorials. Flash also allows for interactivity which allowed this version to include a side menu for easy navigation between the video sections of a tutorial. Another advantage of Flash is that it can be viewed by 99%[15] of the internet users.

The second format available is the downloadable AVI version (.avi). This version has a slightly higher quality than the Flash version, because it is encoded with a

lossless video codec optimized for video tutorials (See section 3.6). This codec[24] has the minor inconvenience that it must first be installed on the student's computer before he or she can view this format. However, using a common video codec such as Windows Media Video (.wmv) or QuickTime (.mov) would be more inconvenient, because of the large file size that these codecs produce for this kind of content. For example, the video section covering programming loops in my C# tutorial has a file size of 972 Kb (without the audio) using the lossless codec. Encoded using Camtasia Studio's recommended settings the same section is 1,641 Kb in the WMV format and 1,246 Kb in the MOV format.

The third format that the tutorials exist in is the SCORM[16] conforming version which can be downloaded in the ZIP format (.zip). This version of the tutorials is to be used with a Virtual Learning Environment (VLE), such as Moodle[17]. The format provides a convenient way to distribute tutorials to a large number of students, for example in universities or corporations.

In addition to these three versions – Flash, AVI, and SCROM – a fourth text only version of the tutorials can also be read through the site's wiki[2]. This version includes both the audio script and the code examples in clear text, which offers certain accessibility advantages over the other three video versions. For example, the text/code can be copied, enlarged, or read using a screen reader.

Another convenience feature is that the tutorials are recorded using a large font size and low resolution (640x480). This makes it easier for the student to follow along with the examples in the tutorials as they are viewing them – since the window showing the tutorial need not take up the user's entire desktop screen. The large font size even makes it possible to watch the tutorials on handheld devices. This was tested to work fine on an iPhone, where the text was clearly readable on the phone's 3.5 inches screen with 480x320 pixels resolution[18]. However, because the iPhone does not have a Flash player the tutorials tested had to be streamed from the site's YouTube channel[19] instead of from the main site.

One more feature that can be deemed as convenient is that the structure of the tutorials allows them to be used as a quick reference. Because all the tutorials are grouped into short named sections a student can quickly search through the site's library of tutorials to find the material that they need for the moment. Most sections also start with a picture summarizing what the section is about, thus facilitating the student's decision of whether this section is likely to be relevant or not.

The reason why convenience was deemed as an important goal was due to some bad examples that I encountered in my preliminary study. In these cases an e-learning company made it more difficult than it had to be for their users to view their tutorials. For example, CBT nuggets and Keystone only allow their tutorials to be ordered on DVD and not viewed or downloaded online – thus they are inconvenient to access. Another example of poor accessibility is AppDev which forces the user to install a VLE in order to view their tutorials. This VLE not only takes up the whole screen, but does not allow the video to be resized.

2.2.1 Choosing Video over Text

The video format itself can be seen as a convenience feature. Since the video part of the tutorials consists of a series of still images, an easier option than producing video would have been to simply combine the snapshots and script into an HTML tutorial. However, video tutorials have several advantages over HTML tutorials. For starters, because video tutorials include audio narrations instead of text the students are freed from the tediousness of reading and scrolling through HTML pages. The students can simply sit back and relax without having to do anything but learn. Another advantage is that in a video tutorial I can use highlights and progressive disclosure* to direct the

* Progressive disclosure means to gradually add more information so as to direct the student's attention and not to overwhelm him or her.

students focus. These techniques make it easy for the student to know what I am referring to. The script in a video tutorial can therefore be shorter than the equivalent HTML tutorial, because I do not have to be as specific to what part of the code examples I am referring to. The HTML tutorials does have accessibility advantages over video tutorials, such as allowing the text/code to be resized, copied, and read using a screen reader. However, all of these advantages have been preserved through the use of a wiki. My final reason for choosing the video format was that video tutorial learning in my personal opinion is a more fun and interesting way to learn compared to reading.

2.3 Rapid Production

A third goal for the new method was that it should allow rapid tutorial production. In order for this to be accomplished a highly streamlined workflow was needed that avoided any unnecessarily time-consuming steps. In my experience the longest step in producing a tutorial was the live recording of the audio and video. This was because a single mistake in either the audio or the video would generally ruin the recording, thus a lot of re-takes were required for each section in order to reach an acceptable level of quality. I also had to know the subject by heart, since I could not look at any script at the same time as I was recording. Although this is the standard method used by all e-learning companies that I studied; it was a very unforgiving approach to producing video tutorials that need to be updated and changed at a later point in time.

The solution adopted was to allow non-linear editing of the tutorials by creating the video and audio separately, which proved to be an immense timesaver. Instead of using live recordings the video was created as a series of snapshots that were later synchronized to the audio. By creating both the audio and the examples/snapshots from the same script they could easily be made to match each other without any inconsistencies. This method made it possible to produce nearly flawless tutorials in a single take, because in contrast to live video the snapshots could be designed and organized without any time constraints (i.e., the re-synchronization allows the video to be cut or expanded as necessary to suit the audio narration). The snapshots could also be reviewed and edited to make sure that there were no mistakes or typos, giving a more professional feel than live video recordings, which in general are never flawless. Best of all, once the snapshots had been created they did not have to be re-recorded in order to reproduce the desired video. If either the snapshots or the audio needed to be modified new video could be reproduced and the appropriate audio added in a fraction of the time needed to redo a live recording.

As a result of the audio being recorded separately from the video there was no requirement (hence no pressure) that the recording had to be flawless, in contrast to live video recordings. This was because when recording only the audio it is possible to take pauses, reiterate, and rephrase the script without any problem, since this content could be edited out or re-ordered later. Because of the updatable attribute of the tutorials the production of the script also required less time, since there was no need to try to make the script “perfect” by double checking everything. Instead, the tutorials could be produced knowing that if any mistakes were later discovered they could easily be fixed.

2.4 Relevant

The fourth and last goal was for each video section to only cover what was relevant to that section. Having read one too many programming books where the author was paid by the number of pages he or she wrote and not by the content, this was a major issue to be addressed. A common theme in most tutorials I studied was that the pace was very slow and that the teachers had trouble separating the relevant from the irrelevant. The teachers did not use a script and thus had to improvise, oftentimes

making the tutorials several times longer than they needed to be. Because of how much time was spent explaining everything – even simple concepts could be made to seem difficult.

In the new method that was being developed each video section was designed to be short and to the point, explaining only the practical “how to” and “why to” of each section without any repetition or needless theory. A primary concern for the new method was to keep the student’s attention and as such I would not bore them by repeating every piece of information. Instead, the students would be responsible for the repetition they need, since they can easily rewind or re-watch each video section until they have fully understood the content. Once the repetition and irrelevant parts of a tutorial have been stripped away, it is surprising how short and simple each section became. As an example, one of my tutorials teaches most of the commonly used elements of PHP in less than 30 minutes.

3. Production Method

This chapter describes the method adopted to produce the video tutorials for this thesis project. The method (named the MVT method*) contains six steps: outline, script, examples, snapshots, audio, and production. The first step is done once for every tutorial and the five following steps are repeated in order to produce each video section.

3.1 Work Environment

Before attempting to produce a video tutorial it is important to first have the correct work environment setup. In terms of software the following kinds of applications are required: a text editor, a presentation program, a video editing program, a graphics editing program, and a digital audio editor. It is also recommended that the producer has a screenshot program, such as Techsmith Corporation's SnagIT[20]. The applications I prefer to use in the listed categories are: Microsoft Visual Studio, Microsoft PowerPoint, Techsmith Corporation's Camtasia Studio, Paint.NET, Audacity, and SnagIT.

When it comes to hardware the producer should have at least two and preferably three computer monitors installed. Using multiple monitors significantly speeds up the video production since the producer must often work with several applications in parallel. One of the monitors should always be set to 640x480 pixels, which is the resolution used for the video tutorials. This is the "recording" monitor where the producer sets up and captures the video portion of a tutorial.

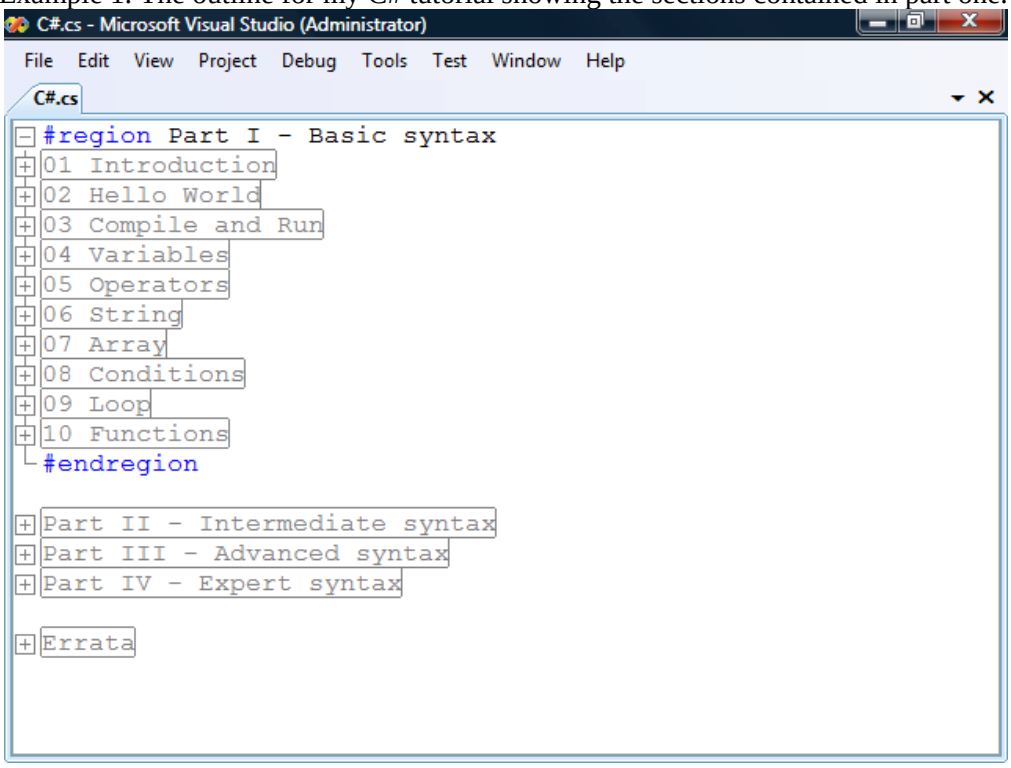
Another piece of hardware that is required is a microphone in order to record the audio. This does not have to be an expensive studio microphone, but it should not be the cheapest one either. I currently record using a Sennheiser PC-151 Headset[21]. When recording audio it is important to first remove as much background noise as possible. Although a noise reduction filter can be applied to the audio after the recording is done too much filtering will significantly reduce the audio quality.

3.2 Outline

The first step in producing a tutorial is to make an outline for it as a text file. I prefer to do this in Visual Studio because it allows me to collapse the text using the `#region` directive. What I do in this step is simply to name and order every section that the tutorial will contain. Each section name should describe a very specific area of the tutorial, because this will make it easier for the students to search through the tutorial. It is also important that the sections are placed in an order such that they do not have to refer to later sections to explain something. To organize the large number of sections that a tutorial can contain I prefer to visually group every ten or so sections into a region called a part. At the bottom of the file I keep another region containing mistakes and updates to sections that I am already done producing. Whenever enough of these fixes are found, either by me or by people writing in the wiki, then that section will be scheduled for an update. A period of time every month will be dedicated to performing these updates, unless an update is deemed critical, in which case that section will be fixed as soon as possible.

* MVT method – Motionless video tutorial method.

Example 1. The outline for my C# tutorial showing the sections contained in part one.



3.3 Script

Once the outline is complete I start working on the sections one at a time. The first step in this cycle is to write the audio script. This script is compiled from sources all over the web that are summarized in the outlined text file. The script is then rewritten extensively to make it readable and easier to understand. My goal is to cover everything important about the section in less than 500 words. The reason for this limit is twofold. First, it forces me to come up with simpler ways to explain everything. And second, it only allows me to keep what is important to the section, in accordance with the goal of relevance. I chose 500 words because I have found that it is very rare that a section cannot be explained within this limit. If despite my best efforts I am unable to explain a section in 500 words, then I will split that section into two parts rather than go past the word limit, because in my experience this tends to be faster than producing a single longer section.

3.4 Examples

The third step is to produce programming examples for the concepts I talk about in the audio script. These examples are created and tried out in the development program used for that tutorial, then copied into the script where they can be organized. Storing the code in the script file also makes it easier to reproduce the snapshots if any changes needed to be made later. If the section contains elements that would be useful to have as a quick reference a Microsoft PowerPoint slide of those elements is created as well. This slide will be used as the first frame of the video section. If needed, other slides will also be created to present tables and other information rich content in the section. I chose to use PowerPoint for the creation of these slides because this program makes it easy to create, edit, and design these slides. Note that only an image of each slide is used in the video tutorial, thus the student does not need to have a PowerPoint viewer.

3.5 Snapshots

When the code examples for a section are done they are used to create the snapshots. The key here is to only change what the student should focus on from one frame to the next. It is also important that the snapshots closely match what is talked about in the audio script (or vice versa, adjusting the script to match the snapshots). The snapshots used are 640x480 pixels. For capturing the snapshots I use SnagIT, because it is the only program I have found that is able to capture and save a fixed region of the screen with the press of a single button. The pictures are losslessly stored in the PNG format (.png). When the snapshots are complete they are renamed with a number and a letter, where the numbers correspond to paragraphs in the audio script. This way new snapshots can be added without having to rename every single file to maintain the order. Finally, when the pictures are named and placed in that section's folder a graphical editor is used to fix any small mistakes or distractions in the images. I personally use Paint.NET[22] to correct the images, because it is a free, easy to use program that is more than adequate for this task.

3.6 Audio

Before recording the audio it is important to first read the script out loud a couple of times to make it sound more natural and not as if a script is being read. While it is possible to record the audio without any preparation this form of rehearsal does in my personal experience significantly improve the quality of the final result. The rehearsal also allows me to make any final adjustments to the script in order to improve its readability. These improvements can be very difficult to discover before the script is read out loud. Once I can read the script smoothly I record the audio with a 44kHz sampling rate. The program I use to record is Audacity[23], because it is a free audio editor that is both easy to use and feature-rich. If I make a mistake when recording the audio I will just speak that sentence again until the entire script has been recorded. Mistakes are then edited out and the pauses are adjusted as needed. Any pause in the beginning of the audio is removed so that all videos will start immediately when they begin playing. A one second pause is also added at the end of the audio to mark the end of the section. This pause provides a much needed break before the next section begins. When the editing is done the file is saved in the lossless wave format (.wav). The audio is then finalized by running a script that does noise reduction, normalization, and downsampling the file to 22kHz. This final version is saved under another filename in case a different set of filters are to be used in the future.

3.7 Production

With both the snapshots and audio now completed it is time to produce the video. For this step the files are added to the timeline in a Camtasia Studio project and a one second fade-in transition is applied between all clips. This slow transition is important in order to direct the students focus and to make it easier for them to distinguish what changes are taking place. The snapshots are then carefully synchronized to the audio. This is mainly achieved by moving the beginning of each video fade to the start of the audio that corresponds to the snapshot following that fade. When all snapshots are synchronized the last frame is extended so that it ends at the same time as the audio ends (note that the audio was extended by one second of silence). Once this is done the video is ready to be produced as an AVI file. The video is compressed with the Techsmith Screen Capture Codec (TSCC). This codec was chosen because it is a freely available lossless video codec optimized for compressing screen recordings[24]. As for the audio it is compressed as 22kHz LAME encoded MP3 at 32kbit/s. The MP3 format was chosen because it is a popular compressed audio format that most video players support without having to install additional codecs. When the AVI file is completed it is reviewed a final time to make sure that everything is correct. The file is then added to another Camtasia project containing all the completed video

sections for that tutorial. This project is used to produce the streamable flash version (.swf) and the SCORM version (.zip). Finally, everything is uploaded to the site and the script is added to the wiki.

4. Evaluation

4.1 Production Results

During the course of the thesis project 85 video tutorial sections were completed and uploaded to PVT (See Appendix 1). Although the original plan was to produce two video sections per work day for a total of 200 sections this goal proved to be too ambitious. The actual production rate was only slightly above one video per day and some time had to be devoted to writing this thesis. Out of the 85 completed sections 75 of them cover programming syntax and 10 cover software development.

Among the 75 programming syntax videos: 15 of them were for Java, 22 for C#, 11 for C++, 16 for PHP, and 11 for ASP.NET. Based on the outlines made for the tutorials the released Java sections constitute 75% of the language syntax. C# was covered to about 80%, although it had several more sections than Java. This is because the C# language is substantially larger than Java. Only about 1/3 of C++ was covered because it was the most difficult language to teach. PHP was the smallest language taught and was essentially completely covered in the released sections. The last 11 sections cover the fundamentals of how to produce ASP.NET pages with C#.

The 10 tutorial sections covering software development demonstrate how to build a basic text editor in Java and C#. Equivalent tutorials were also planned for the other three languages, but these were not finished in time to be evaluated and included in this thesis.

Table 1. Produced video sections.

Topic	Videos	Complete	Topic	Videos
C++	11	33%		
C#	22	80%	C# Example	5
Java	15	75%	Java Example	5
PHP	16	100%		
ASP.NET	11			

4.2 Production Method Efficiency

Using the MVT method a typical tutorial section takes me on average four hours to complete; with between three and five hours being common. About one to two hours of this time is spent creating the audio script. Another hour is needed to create the examples and the snapshots. The audio is recorded and edited in 30 to 60 minutes time and the production step takes another half an hour. These numbers have been confirmed repeatedly, because as a way of trying to improve my own efficiency I always time how long it takes to complete each video section. Although the amount of time needed for each production step has decreased significantly since I first started producing tutorials; there is still a lot of room for improvement, particularly in the script and audio steps.

Even though four hours may seem like a long time to produce such short videos (sections are generally 2-3 minutes long) a lot of that time is actually spend making the video so concise. It is interesting to note how much time these few minutes corresponds to in a professionally made commercial tutorial. As an example, in my C++ tutorial I cover loops (See section 6.2 on page 23) in 2 minutes and 1 second. In 3dBuzz's C++ tutorial their section covering the exact same area is 18 minutes and 47 seconds long. Simply by following the MVT method's goal of keeping only what is

relevant (See section 2.4) I was able to teach the same information in almost one tenth the time. Unfortunately, I do not know how much time was required to produce the 3dBuzz tutorial; thus I cannot compare their time efficiency for preparation with mine.

Table 2. Average time spent at each production step.

Step	Description	Duration	%
1. Outline	Creating an outline	5m	2,1 %
2. Script	Gathering information Rewriting script	30m 60m	37,5 %
3. Examples	Creating examples Organizing examples	30m 15m	18,8 %
4. Snapshots	Taking snapshots Editing snapshots	10m 10m	8,3 %
5. Audio	Recording Audio Editing Audio	30m 15m	18,8 %
6. Production	Synchronizing Producing/Uploading	15m 10m	10,4 %

4.2.1 Optimizations to the Production Method

The method for producing the tutorials was adjusted significantly during the project in order to speed up production. In the beginning the outline step included compiling the material that would be contained in all sections. This compilation part proved to take a lot more time than doing the compilation for individual sections – so this part was delegated to the script step. Another optimization made was moving the script step before the examples and snapshot steps. This came as a result of noticing that it was generally easier to produce the examples based on the script rather than to rewrite the script based on the examples.

The original method also included a step where all the PowerPoint slides covering the theory part of the tutorial were compiled. This step was moved into the examples step because the actual content of every section has to be known in order for the slides to be produced. The new method of creating the slides only when they were needed was not only faster, but also made a lot more sense. In the old method some of the slides produced were never used and those that could be used often had to be remade in the examples step.

These seemingly small changes actually reduced the production time needed to complete a tutorial by a third. Using the old method the first two steps would take nearly a week and resulted in no completed videos (i.e., there was no video to show for all this effort). During the second week when the sections were worked on individually, both the script and slides had to be modified extensively before they could be produced, which took at least half a week. With the new method only an hour or so is needed to plan and outline the tutorial and the rest of the time is spent focused on producing individual sections. This workflow not only proved to be much more effective, but also provided better motivation since it was possible to see the resulting videos on a day to day basis.

4.2.2 Time Needed to Update Sections

The time needed to update a video section various significantly depending on what changes need to be made. For example, if there is a small change in the audio script I only need to edit that part of the audio and resynchronize any clips that have been offset by the change. All in all this type of change can be done under ten minutes. Another common change is to edit one or more of the snapshots that make up a video section. Such an easy change does not require any resynchronization to be made and can generally be done in a few minutes. If more extensive changes need to be made to both the audio script and the snapshots, then it may take up to half an hour to reproduce the video.

The site's wiki[2] is meant to make the process of updating tutorial sections significantly easier. Because wikis are self-correcting the audio scripts should gradually improve over time and become more accurate[25]. Therefore, I would not have to personally spend time looking for improvements or errors in the video sections. All I would need to do is to make the necessary changes to the snapshots and audio in accordance with the suggested changes on the wiki.

Unfortunately, this wiki project has as of yet not received much attention. Despite several marketing attempts I have been unable to get even a single person other than myself to make any contributions – thus no updates have been made as a result of the wiki. Although I initially found this lack of participation from the visitors to be quite vexing, I believe the answer is quite simple. The reality of any wiki project is that only a small minority of the visitors ever make any contributions[26]. For example, out of the 684 million visitors Wikipedia received in 2008 only 75,000 of them are active contributors (people who contribute at least 5 times per month)[27]. As of yet my e-learning wiki has not attracted more than a few thousand visitors – thus it is understandable why it has not received any contributions.

4.3 Production Method Comparison

As mentioned in the introduction to this thesis the standard method of producing video tutorials involves recording the screen content using screen capture software. This method, which I call the SR method*, is used by all e-learning companies that I have studied. The MVT method has both advantages and disadvantages when compared with this SR method.

4.3.1 Advantages

The first advantage is that it is easier to produce professional looking tutorials using the MVT method. Since this method allows the audio and snapshots to be edited separately any flaws or mistakes can easily be removed. The producer can also take as much time as he or she needs to design and organize the snapshots before the video is produced – thus making it easy to create a high quality professional looking video tutorial.

In the SR method the audio and video are recorded at the same time. Therefore, tutorials produced using this method typically cannot be edited after they have been recorded without the changes being noticeable. At the same time it is difficult to avoid making mistakes when recording a video tutorial live, and these mistakes tend to reduce the perceived quality of the tutorial. Therefore, making a professional looking video tutorial using the SR method generally requires a lot of retakes, which in my experience can be very frustrating. In the MVT method there are no retakes necessary. Each step in the production cycle is performed only once for every section.

Another important advantage of the MVT method is that the tutorials can be updated quickly. Using the SR method a tutorial section cannot be updated without re-record the whole section again from the beginning.

* SR method – Screen recording Method.

4.3.2 Disadvantages

The MVT method has two drawbacks compared with the SR method. First, in terms of production speed the SR method is faster (the first time a video section is produced). The production speed can vary a lot depending on the producer's experience, the number of retakes necessary, and the required quality of the tutorial. However, because the SR method does not use a word-for-word script and does not require the tutorial to be updatable it is significantly faster than the MVT method, provided that the producer knows the subject area well enough.

The second drawback is that the MVT method cannot be used in all areas where the SR method can be used. Because the video is made up of snapshots the MVT method is not suitable for tutorials that require a lot of motion. For example, the MVT method would not be suitable for teaching 3D animation or any form of computer drawing.

5. Basic Syntax and Semantics

5.1 Data Types

The fundamental data types built into the four programming languages can be grouped into four categories: integer, floating-point, character, and boolean types. As can be seen in the table below, C++, Java, C#, and PHP all share the same fundamental types, with some differences. An important difference is illustrated by Java and PHP which does not provide any unsigned integer types, whereas C++ and C# do. C# has distinct data types for signed and unsigned integers, while C++ uses the keywords `signed` and `unsigned` to specify these as distinct sub-types.

Another difference is that C++, Java and C# provides different data types depending on how large or precise a value the integer or floating-point type must be able to store, whereas PHP do not. PHP in contrast to the other three languages is a loosely typed language, meaning the programmer does not need to explicitly specify the data type of the variable(s). Instead, PHP automatically converts variables to the correct data type, depending on the context in which they are used. The keywords shown in the PHP column below are not used for declaring variables, only for type casting[28].

Table 3. The fundamental (native) data types.

C++	Java	C#	PHP	Description
short, unsigned short int, unsigned int long, unsigned long	byte short int long	sbyte, byte short, ushort int, uint long, ulong	int	Integers
float double	float double	float double decimal	float	Floating-point numbers
char wchar_t	char	char	string	Character Wide-character
bool	boolean	bool	bool	Boolean value

5.2 Variables

Variables are declared in the same way in C++, C#, and Java, with the data type specified first followed by an identifier. In contrast, PHP variables are declared without specifying the type of the identifier, since variables in PHP are automatically declared and initialized the first time they are used. Another difference between PHP and the other three languages is that PHP variables must be prefixed with a dollar sign "$", which indicates to the parser that they are variables.

Assigning values to variables is done with the same syntax in all four languages, using the assignment operator "=". C++ also supports an alternative way of initializing variables called constructor initialization. This is done by enclosing the initial value in parentheses and only works at the time the variable is declared. If more than one variable needs to be declared and optionally initialized there is a shorthand way of doing this using the comma operator "`,`" (See Example 1. Variable

declaration and initialization.Example 1). This operator works for all languages except for PHP.

In C# and Java the use of global variables are forbidden. All variables must be contained within a type or a function. This is not the case in C++ and PHP, where variables may be declared globally as well as within containing types or functions. Variables in PHP are always initialized to their default values whether they are declared globally, locally, or in a class. In C++ only global varables are automatically initialized, but not local or class variables. Local variables are also not automatically initialized in C# and Java, only class variables (fields) are. In contrast to C++, C# and Java do not allow local variables to be used unless they are initialized. Consequently, using uninitialized variables is a common programming mistake in C++[29].

Example 2. Variable declaration and initialization.

Java	C#
`int a = 50, b = 10;`	`int a = 50, b = 10;`
C++	**PHP**
`int a = 50, b = 10;` `int b(50);`	`$a = 50;`

5.3 Constants

Constants can be divided into two categories: compile-time and run-time constants. Compile-time constants are replaced by the compiler and must therefore be initialized to a constant value at the same time as they are declared. Run-time constants on the other hand are not set until the program runs. They can therefore be initialized to a dynamic value, and may be set in a constructor if they are declared in a class.

Both kinds of constants exists in all languages except for C++, which only has compile-time constants. The C++ developer can use either the `const` modifier or the `#define` directive to declare compiler-time constants. With `#define` the preprocessor (which is run before the compiler) replaces any occurrences of its first argument with whatever follows it to the end of the line. Const on the other hand has the benefit of type checking[30].

In PHP, constants can be set either using the `define()` function for a run-time constant, or using the `const` modifier for a compiler-time constant. Note that these constants are used in the script without the "`$`" parser token, which is required when using variables[31]. C# uses `const` for compile-time and `readonly` for runtime constants. The Java language declares compile-time constants with `static final`, and run-time constants with only `final`[32].

The `final` modifier in Java along with the `const` keyword in C++ can also be applied to method parameters to make them unchangeable. This functionality does not exist in C# or PHP. In C++, the `const` modifier can even be applied to a function's return type in order to return a constant, or after a class method's parameter list, to hint that the method does not change the interal state of a class[33].

Example 3. Creating constants.

Java

```
static final int PI = 3.14;
final int PI;
```

C#

```
const int PI = 3.14;
readonly int PI;
```

C++

```
#define PI 3.14
const int PI = 3.14;
```

PHP

```
define("PI", 3.14);
const PI = 3.14;
```

5.4 Comments

Comments are used to increase the readability of the source code and have no effect on the execution of a program. As can be seen in the table below, C#, PHP, and Java all inherit the standard C++ multiline (`/* */`) and single-line (`//`) comment notations. In addition to these PHP also has the Perl-style single-line comment (#). For writing documentation both C# and Java provide their own notations, specifically the Javadoc multiline comment (`/** */`) for Java and the single line XML comment (`///`) for C#.

Table 4. Comment notations.

C++	Java	PHP	C#	Description
`/* */`	`/* */`	`/* */`	`/* */`	multiline comment
`//`	`//`	`// #`	`//`	single-line comment
	`/** */`		`///`	documentation comment

5.5 Operators

The operators in the four programming languages are very similar, both in terms of the symbols used and their functionality. In fact, C# and C++ share the same operators, while Java and PHP have made some small additions. The operators have all been included in a table below. For comparison purposes they have been grouped into five categories: arithmetic, assignment, comparison, logical, and bitwise operators.

Table 5. Operators

All languages	PHP	Java	Operator type
`+ - * / %`			arithmetic
`= += -= *= /= %= ++ -- &= |= ^= <<= >>=`	`=&`	`>>>=`	assignment
`== != > < >= <=`	`=== !== <>`		comparison
`&& || !`	`and or xor`		logical
`& | ^ << >> ~`		`>>>`	bitwise

Java includes the zero-fill right shift operator “>>>” and its assignment counterpart “>>>=”. The zero-fill right shift operator moves all bits to the right filling with zeroes on the left, whereas the right shift operator “>>” ignores the sign bit. The need

for this operator comes as a result of Java not having unsigned data types. There is no need for it in C++ and C# since the right shift operator "`>>`" works differently on unsigned variable types and signed variables, i.e., for signed variables it fills with the sign bit.

PHP 4 introduced the identical "`===`" and not identical "`!==`" operators for comparing both value and data type[34]. The language also includes the BASIC-style not equal operator "`<>`" along with three extra logical operators "`and`", "`or`", and "`xor`". The first two of these "`and`" and "`or`" have the same meaning as "`&&`" and "`||`", but all three have a lower level of precedence than the other logical operators.

5.6 Strings

String variables are used to store values that contain string literals. They are built-into every language except for C++, where the `string` header from the standard library needs to be included in order to work with strings. To create strings in Java the `String` class is used, which is included by default with the `java.lang` package. Similarly, C# uses the string keyword which is an alias for the `System.String` class.

One difference between the languages is that string concatenation is performed using the "`+`" and "`+=`" operators in C++, C#, and Java, while PHP uses the "`.`" and "`.=`" operators. Furthermore, strings in C++, C#, and PHP can be accessed as arrays, while Java strings cannot. Another difference is that C++ and PHP strings can be modified, whereas Java and C# strings are immutable. Methods in Java and C# that appear to modify a string's content actually simply return a new string.

In all languages except for PHP string constants are always delimited by double quotes (`""`). PHP strings on the other hand can be delimited in four different ways. There are two singleline notations – double-quote and single-quote – and two multiline notations: heredoc and nowdoc. Variables and escape characters are not parsed in single-quote and nowdoc strings, whereas they are parsed in double-quote and heredoc strings.

In contrast to PHP, Java and C++ do not have a way to cause escape characters to be ignored. However, C# strings can be set to ignore both escape characters and newlines by pre-appending the string constant with the "`@`" symbol. Creating multiline strings is also possible in C++ by using the backslash character to break the line – without including the new line or white space in the string.

Example 4. Creating strings.

Java

```
String a = "Hello";
String b =
new String(" World");
String c = a + b;
```

C#

```
string a = "Hello";
string b = a + " World";

string e = @"verbatim
             string";
```

C++

```
#include <string>
using namespace std;

string a = "Hello";
string b(" World");
string c = a + b;

// C-style string
char *c = "Hello \
           World";
```

PHP

```
$a = "Hello";
$b = $a . " World";

$a = 'Hello $b'; // no parsing

$s = <<<LABEL
Heredoc (with parsing)
LABEL;

$s = <<<'LABEL'
Nowdoc (without parsing)
LABEL;
```

5.7 Arrays

Arrays are data structures used for storing a collection of values. In C++, C#, and Java arrays have fixed sizes and all values in the array must in general be of the same data type. In contrast to this PHP arrays are dynamically sized and they can contain a mixture of data types.

Java and C# arrays are objects that are bounds-checked, making them safer than the unconstrained C++ arrays. Because they are objects they also have methods for retreiving their own length, which C++ arrays do not.

The syntax for declaring C# arrays is different from C++ arrays in that the square brackets follows the array type in C# and not the identifier (e.g., `int[] foo;`). Java arrays on the other hand can be declared with either the square brackets after the data type or after the identifier. In PHP, arrays are declared automatically when they are used, just as with variables.

Arrays are initialized in the same way in C# and Java, using the `new` keyword followed by the data type and the length in square brackets. This notation is also used in C++ to create arrays in the heap, which in contrast to the other garbage collected languages these arrays must be deleted manually. In addition to heap based arrays, C++ also allows stack-based arrays, which none of the other languages do. These can be declared by placing the desired length of the array inside the square brackets. Unlike heap arrays however, the length of stack-based arrays must be a constant value. Lastly, in PHP an array does not have to be initialized and can be used as if it were an array of unbounded length.

At the same time as an array is declared it can be initialized with multiple values using a curly bracket notation in C++, C#, and Java. Note that this notation can only be used for stack-based arrays in C++, not for arrays in the heap. In PHP the array constructor can be used to initialize an array with content.

When assigning individual elements all languages access the array in the same way: by referencing the element with its index in square brackets. Only PHP has the option of leaving out the index when appending a value to the end of the array.

In addition to numeric arrays PHP also has built-in support for associative arrays. With these arrays the key is given as a string name instead of a number. The double arrow operator "`=>`" is used to tell which key refers to what value. This operator can also be used in the numerical array's constructor to indicate where elements will be placed.

Example 5. Single-dimensional arrays.

Java

```
int x[] = new int[3];
int[] y = {1,2,3};
```

C#

```
int[] x = new int[3];
int[] y = {1,2,3};
```

C++

```
int x[3];
int y[] = {1,2,3};

int z[] = new int[3];
delete z[]
```

PHP

```
$a = array(1,2,3);
$a[] = 4;

$map = array("one" => "a", "two" => 2);
$map["one"] = "a";
```

5.7.1 Multidimensional Arrays

Arrays can be made multidimensional by adding additional pairs of square brackets. As with single-dimensional arrays, these arrays can either be filled in one at a time, or all at once at the time of the declaration. All languages except for C++ support creating arrays-of-arrays, or jagged arrays. Java has a shortened notation for creating

these, while C# must use the new initializaiton syntax for the subarrays. Multidimensional arrays in PHP can also be assigned quickly by nesting array constructors.

In addition to jagged arrays, C# includes true multidimensional arrays, or rectangular arrays. This kind of array is also the only type of multidimensional array that C++ has built-in. They differ from jagged arrays in that each subarray must have the same dimensions.

C++ creates rectangular arrays using the same syntax as Java's jagged arrays. However, in C++ the dimensions must be specified, which is optional in Java. Rectangular arrays in C# are declared by separating the dimensions with commas instead of using multiple square brackets. This array type also supports the short initialization syntax without the use of new, just like Java and C++.

Example 6. Jagged and rectangular multidimensional arrays.

Java

```
String[][] j =
  {{ "00" },
   { "10", "11" }};
```

C#

```
string[,] r =
  {{ "00", "01" },
   { "10", "11" }};

string[][] j =
  { new string[] { "00" },
    new string[] { "10", "11" }};
```

C++

```
#include <string>
std::string r[2][2] =
  {{ “00”, “01” },
   { “10”, “11” }};
```

PHP

```
$j = array(array( "00", "01" ),
           array( "10", "11" ));
```

5.8 Pointers

A pointer is a variable that contains a memory address. This is a powerful low-level feature commonly used in C++. It provides the ability to directly manipulate specific memory locations. Because pointers are so powerful they are also one of the main sources of software bugs[35]. Consequently, Java and PHP do not have pointers, and C# only allows restricted use of them. Instead, these languages use only references. References are restricted pointers that cannot be modified with pointer-arithmetic, making them much safer and simpler to use than pointers.

C++ also has a form of reference data types created by adding the address-of operator "&" to the data type. These types are more restricted than pointers, in that they must be initialized when they are declared (or in a constructor if they belong to a class) and once assigned they can never be changed[36].

Just like in C++, a variable in PHP can be assigned the reference of another variable by placing the address-of operator "&" before it. This allows PHP to create references to value types in a way that is not possible in C# or Java.

Core C# does not have any pointers, however code blocks and methods can be marked with the `unsafe` keyword to enable the use of pointer types and operators. Such code can deal directly with pointers just as in C++.

A major use of pointers in C++ is to point to memory that has been dynamically allocated on the heap using the `new` keyword. Such memory must be explicitly deleted using the `delete` keyword when it is no longer used in order to avoid memory leaks. C#, Java, and PHP do not have a `delete` keyword since they all have garbage collectors that handle the cleanup. This simplifies the syntax of these languages a bit compared to C++.

Example 7. Using pointers.

C++

```
int i = 1;
int *p;
p = &i;
*p = 2;

int& ref = i;
ref = 3;
```

C#

```
unsafe
{
  int i = 1;
  int *p;
  p = &i;
 *p = 2;
}
```

Java

```
// not supported
```

PHP

```
$i = 1;
$ref = &$i;
$ref = 3;
```

6. Intermediate Syntax and Semantics

6.1 Conditions

The conditional statements are used to execute different code blocks based on the stated conditions. All four programming languages use the same two conditional statements – `if` and `switch` – with almost the same syntax. In addition to the normal syntax PHP also provides an alternative "colon-syntax" for the conditional statements. The opening brace is then replaced with a colon, the closing braces are removed and the last closing brace is replaced with either the `endif` or `endswitch` keyword (See Example 7).

6.1.1 If

The if-statement will execute only if the conditional expression inside the parethessis evaluates to true. The curly braces that make up the body of the if-statement can be left out if there is only one statement in the codeblock.

The only differences between the languages is that in Java and C# the expression must evaluate to a boolean value, whereas in C++ and PHP it can be anything that evaluates to a number. If the number is zero it will be interpreted as false and any other value will be interpreted as true. Conditional expressions in Java and C# are therefore easier to understand at the cost of being a bit longer. The additional type safety also helps to find some forms of errors at compile time, such as the common programming error of mistakenly writing assignment "=" instead of comparison "==" in a condition[37].

Example 8. Using the if-statement.

Java

```
if(bool) {}
else if(bool) {}
else {}
```

C#

```
if(bool) {}
else if(bool) {}
else {}
```

C++

```
if(expression) {}
else if(expression) {}
else {}
```

PHP

```
if(expression) {}
else if(expression) {}
else {}

if (expression):
else if (expression):
else:
endif;
```

6.1.2 Switch

The switch statement executes the case whose label matches the value of the expression given in parenthesis. In C++ and Java this expression must evaluate to an integer, while in C# it may also be a string. In addition to integers and strings, PHP also allows the expression to be a floating-point value.

Another difference is that the switch case statements in C# must end with a jump statement (such as `break`) unless the case is empty. This is enforced because unintentional fall-throughs by leaving out the `break` keyword is a common programming mistake in other programming languages. To explicitly perform a fall-through in C# the `goto` jump statement can be used, while the `break` keyword can just be left out to achieve the same effect in the other languages.

Example 9. Performing fall-throughs in switch statements.

Java

```
switch (integer)
{
  case 0:
  case 1: break;
  default: break;
}
```

C#

```
switch (integer, string)
{
  case 0: … goto case 1;
  case 1: break;
  default: break;
}
```

C++

```
switch (integer)
{
  case 0:
  case 1: break;
  default: break;
}
```

PHP

```
switch (integer, string, float)
{
  case 0:
  case 1: break;
  default: break;
}
```

6.1.3 Ternary

In addition to the if and switch conditional statements, the ternery conditional operator "`?:`" can also be used. This operator replaces a single if-else clause that assigns one of two values to a specific variable based on a condition.

The operator takes three expressions. If the first expression evaluates to true, then the value of the second expression is returned, otherwise the value of the third expression is returned. In C++ and PHP this operator can be used as a stand-alone statement as well as an expression, whereas in Java and C# it can only be used as an expression.

Example 10. Using the ternary conditional operator.

Java

```
x = (bool) ? 0 : 1;
```

C#

```
x = (bool) ? 0 : 1;
```

C++

```
x = (expression) ? 0 : 1;
(expression) ? (x = 0) : (x = 1);
```

PHP

```
$x = (expression) ? 0 : 1;
(expression) ? ($x = 0) : ($x = 1);
```

6.2 Loops

Loop statements are used to execute a specific codeblock several times. There are three of them in C++: while, do-while, and for. In addition to these three Java, PHP, and C# also include the foreach loop. Although the foreach loop is missing from C++ it can easilly be replaced with the for loop.

The syntax of the while, do-while, and for loops are identical in all four languages, except for the boolean/numeric difference in the conditional expression's value as meantioned earlier (See section 6.1.1). The for-each loop however has a slightly different syntax in each language, which can be seen in the examples below.

As with the conditional statements in PHP the while, for, and foreach loops can also use the alternative "colon syntax". That is, the braces can be rewritten into a colon and the `endwhile`, `endfor`, or `endforeach` keyword used. PHP also features an extension of the foreach loop – to be used with associative arrays to get the key's name as well as its value.

Example 11. Loop statements.

Java

```
while (i < 10) {}
do {} while (j < 10);
for (int k = 0; k < 10; k++) {}
for (int value : array) {}
```

C#

```
while (i < 10) {}
do {} while (j < 10);
for (int k = 0; k < 10; k++) {}
foreach (int value in array) {}
```

C++

```
while (i < 10) {}
do {} while (j < 10);
for (int k = 0; k < 10; k++) {}
```

PHP

```
while ($i < 10) {}
do {} while ($j < 10);
for ($k = 0; $k < 10; $k++) {}
for (int $arr as $val) {}
for (int $arr as $key => $val) {}
```

6.3 Jump Statements

Jump statements redirect the program's execution from one program location to another. The four languages share the following four jump statements: `break`, `continue`, `return`, and `throw`. C++ and C# also support the `goto` statement, although this keyword is restricted in C#; where it may only be used to jump within the current scope[38], but not into another code block as is allowed in C++. The `goto` keyword is also reserved in Java, even though it is not implemented.

The `break` and `continue` keywords can be used inside of loops to end a loop or skip an iteration. In PHP, both `break` and `continue` can be given a numeric argument in order to apply to an outer level loop. Java has a feature similar to this where a label is added before the outer loop and then that label is given as an argument to break or continue. To achieve the same result in C# and C++ the `goto` statement has to be used.

The `return` keyword has the same usage in all four languages. A return exits a method and returns a value or reference to the caller as determined by the data type specified in the function's declaration. In PHP, since data types are not specified, a function may return a value of any type. The `throw` keyword will be looked at later (See section 8.3).

Table 6. Jump statements.

C++	PHP	Java	C#	Description
break	break	break	break	ends current loop
continue	continue	continue	continue	skips current iteration
return	return	return	return	returns to caller
goto			goto	unconditional jump
throw	throw	throw	throw	throws an exception

Example 12. Breaking out of nested loops.

Java

```
MyLabel: while (true)
{
  while (true)
  { break MyLabel; }
}
```

C#

```
while (true)
{
  while (true)
  { goto MyLabel; }
}
MyLabel:
```

C++

```
while (true)
{
  while (true)
  { goto MyLabel; }
}
MyLabel:
```

PHP

```
while (true)
{
  while (true)
  { break 2; }
}
```

6.4 Functions

A function is a reusable code-block that will execute only when called. Functions are defined in the same way in C++, C#, and Java, with the return type followed by the function's name, parameter list, and body. In PHP the return type is replaced by the `function` keyword, since data types are not explictly specified. Another difference is that functions as well as classes in PHP are case insensitive.

In C# and Java functions must always belong to a class, whereas in C++ and PHP functions may also be declared globally. PHP functions can even be defined inside of other functions, allowing a run-time decision as to whether or not a function should be defined.

Functions as well as classes in C++ need to be declared before they can be called. If the function is to be used before it is implemented, then a prototype must be specified. In contrast, C#, Java, and PHP requires no forward declarations.

Example 13. Defining a method.

Java

```
class MyClass
{
  void MyMethod(int x) {}
}
```

C#

```
class MyClass
{
  void MyMethod(int x) {}
}
```

C++

```
void MyMethod(int x); // prototype
void MyMethod(int x) {}
```

PHP

```
function MyMethod($x) {}
```

6.4.1 Passing Arguments

Arguments to a method can be passed either by value or by reference. When passing arguments by reference both value and reference data types can be changed or replaced and the changes will affect the original. On the other hand, with pass-by-value only a local copy of the variable or reference is accessible from within the method. Therefore, changing the variable or replacing the object will not change the original. However, the state of an object can still be modified, since the copy points to the original's memory location.

For passing parameters C++, C#, and PHP support both pass-by-reference and pass-by-value, while Java supports only pass-by-value. In order to pass a value type by reference in Java the variable must first be encapsulated within a class. Objects are

always passed by reference in C# and PHP, while value types are by default passed by value. This is different from C++, where all variables and objects are passed by value unless they are explicitly passed as a pointer or a reference.

C# can force a value type to be passed by reference by using either the `ref` or `out` keyword in both the caller and the method declaration. The `ref` keyword simply passes the variable by reference, while `out` allows the developer to pass an unassigned variable which must then be initialized in the method.

In C++ and PHP, passing a value type by reference is done by adding an ampersand "&" before the arguments name in the function definition. To pass a variable by pointer instead in C++ the dereference operator "*" is added after the argument's data type and the address is passed when calling the method.

Example 14. Passing parameters by value and/or reference.

Java

```
void ByValue(int a) {}

int x = 1;
ByValue(x);
```

C#

```
void ByValue(int a) {}
void ByRef(ref int a) {}

int x = 1;
ByValue(x);
ByRef(ref x);
```

C++

```
int ByValue(int a) {}
int ByRef(int &a) {}
int ByPointer(int* a) {}

int x = 1;
ByValue(x);
ByRef(x);
ByPointer(&x);
```

PHP

```
function ByValue($a) {}
function ByRef(&$a) {}

$x = 1;
ByValue($x);
ByRef($x);
```

6.4.2 Method Overloading

Method overloading means that a function can be defined multiple times with different arguments. This is a powerful feature which allows a method to handle a variety of parameters transparent to the user of the class.

Methods can be overloaded in C++, C#, and Java, but not in PHP. Since PHP does not have strong typing, a function parameter can already accept any data type by default. In combination with this PHP allows default parameter values – thus enabling the effect of method overloading. Default parameters are also avaliable in C++, but not in C# or Java. They are defined in the same way in PHP and C++, simply by assigning values to the arguments, starting with the last argument of the method.

Example 15. Method overloading and default parameters.

Java

```
void F(int a) {}
void F(float a) {}
```

C#

```
void F(int a) {}
void F(float a) {}
```

C++

```
void F(int a = 3) {}
void F(float a = 3.14) {}
```

PHP

```
function F($a = 3.14) {}
```

6.4.3 Variable Parameter Lists

In all four languages it is possible to make a function take a variable number of arguments. This is done by using the `params` keyword in C# as a modifier to the last argument of the method which must be an array[39]. Similarly, the same effect is achieved in Java by appending an ellipsis "..." to the type name of the last argument of the method[40]. Just as in C# the argument becomes available as an array in the called method.

A variable parameter in C++ is also created using an ellipsis, but unlike in Java it is added as an argument to the end of the list. The optional arguments can then be retrieved by using the `va_list`, `va_start`, `va_arg`, and `va_end` macros[41]. Unlike the other languages, PHP always allow more arguments to be passed then are specified in the method declaration. These additional arguments can be accessed using the built-in functions `func_get_arg()`, `func_get_args()`, and `func_num_args()`.

Example 16. Methods that accept a variable number of parameters.

Java

```
void F(int... args)
{
  for(int i : args) {}
}
```

C#

```
void F(params int[] args)
{
  foreach (int i in args) {}
}
```

C++

```
#include <stdargs.h>
void  F(int first, ...)
{
  int i = first;
  va_list marker;
  va_start(marker, first);

  while(i != -1)
    i = va_arg(marker, int);

  va_end(marker);
}
```

PHP

```
function F()
{
  $args = func_get_args();
  foreach ($args as $arg) {}
}
```

6.4.4 Main Method

The entry point of C++, C#, and Java programs is the main method. PHP does not have a specific method from which execution starts. Instead, the scripts are simply executed from top to bottom.

The syntax of the main method varies somewhat between the languages. In Java this method must have the `public static void` modifiers and accept the command-line arguments as a string array. The main method in C++ needs to have the `int` return type, but it is optional whether the method actually returns an integer or not. The parameter list for main may be empty or contain the number and values of the command-line arguments.

The main method in C# must be capitalized as all methods are by convention in .NET. It can return a value of either `int` or `void` type and can be declared with or without the command-line string array parameter. If the `int` type is specified, then the return statement is not optional as it is in C++. The `static` modifier must be applied to main just as in Java, but the method does not have to be marked as `public`. The main method will default to private access so that only the CLR can invoke the method[42].

Example 17. Valid main function signatures[43].

Java

```
public static void main(String[] a)
public static void main(String a[])
public static void main(String... a)
```

C#

```
static void Main()
static void Main(string[] a)
static int  Main()
static int  Main(string[] a)
```

C++

```
int main(int argc, const char* a[])
int main()
```

PHP

```
// has no main method
```

7. Advanced Syntax and Semantics

7.1 Class

A class is a template used to create objects. Each class is made up of class members, with the main two members being fields and methods. Fields are variables that hold the state of the object, while the methods define what can be done to the instance of the object.

In Java and C# everything belongs to a class and there are no global functions or data. Although classes are important in C++ and PHP, they are not required. The equivalent of global functions and fields in C++ and PHP would be using static methods and fields within a class in Java or C#.

Class methods in C++ can be defined either inside or outside of the class, while in the other three languages class methods are only allowed inside of the class. A method in C++ that is implemented inside the class becomes an inline method, but if it is defined outside of the class then it is outline method. An outline class method has only a prototype defined inside the class and the actual implementation follows the class definition. The method's name outside the class must be prefixed with the class name and the scope resolution operator "`::`" to indicate which class it belongs to[44]. A class definition in C++ is also different from the other three languages in that it always ends with a semicolon.

Objects in C++ can be created on either the stack or the heap. An object in the heap is created with the `new` keyword and stack-based objects are created without the use of `new`. The initialization part used when creating stack-based objects can optionally be left out, in which case the no parameter constructor will be used to create the object. Java and PHP have no way of creating objects on the stack, however C# can create stack-based types using the `struct` keyword as will be shown later (See section 8.5).

Fields in Java, C#, and PHP can be given initial values at the same time as they are declared. These values cannot refer to non-static fields or methods of the currant instance, but can otherwise be set as in the constructor. Direct assignments to fields cannot be done in C++; thus all field initialization must be performed using the constructor.

Another difference with regard to fields is that PHP has to use the `$this` pseudo variable to access instance fields from inside a class. In the other languages `this` also exists as a keyword referencing the current instance, but it is not necessary to use it in order to access instance fields.

All members of a class are accessed using the dot operator in Java and C#. This operator is also used in C++ for accessing instance members of stack-based objects. PHP has the arrow operator "`->`" to access instance members, this same operator is used in C++ when accessing objects allocated in the heap.

Example 18. Creating and using classes and class methods.

Java

```
class MyClass
{
  public void MyMethod() {}

  public static void
  main(String[] a)
  {
    MyClass c = new MyClass();
    c.MyMethod();
  }
}
```

C#

```
class MyClass
{
  public void MyMethod() {}

  static void Main()
  {
    MyClass c = new MyClass();
    c.MyMethod();
  }
}
```

C++

```
class MyClass
{
  public:
   void MyMethod();
   void MyInlineMethod() {}
};
void MyClass::MyMethod() {};

int main()
{
  MyClass stack;
  stack.MyMethod();

  MyClass* heap = new MyClass();
  heap->MyMethod();
  delete heap;
}
```

PHP

```
class MyClass
{
  function MyMethod() {}
}

$c = new MyClass;
$c->MyMethod();
```

7.1.1 Static

The static keyword is used to declare fields and methods that can be accessed without having to create an instance of the class. These static members exist as a single copy which belongs to the class itself, whereas instance members are created as new copies for each new object.

In C++ and PHP variables can also be declared static inside a function to enable the function to remember the state of the variable. This variable can still only be used inside the function's scope, but it retains its value until the program ends. A class in C# can also be marked as static, but only if it only contains static members and constant fields. These static classes then become restricted, so that they cannot be inherited or instanciated into objects.

To access a static member the class name is used instead of an object's name. Following this is the dot operator in C# and Java, just as when accessing instance members; while PHP and C++ uses the scope resolution operator "`::`".

Example 19. Creating and using static methods.

Java

```
class MyClass
{
  public static void F() {}

  public static void
  main(String[] a)
  { MyClass.F(); }
}
```

C#

```
class MyClass
{
  public static void F() {}

  static void Main()
  { MyClass.F(); }
}
```

C++

```
class MyClass
{
  public:
   static void F();
};
void MyClass::F() {}

int main()
{ MyClass::F(); }
```

PHP

```
class MyClass
{
  static function F() {}
}

MyClass::F();
```

7.1.2 Constructors and Destructors

A constructor is a special kind of class method used to initialize an object. This method is called whenever a new instance of a class is created. In C++, C#, and Java the constructor has the same name as the class and does not have a return type, since it implicitly returns an new instance of the class. The constructor in PHP 5 has the special name `__construct()`, although using the class name as in the other languages also works for backwards compatibility reasons[45]. Another difference in PHP is that the parenthesis is optional when calling a constructor that takes no parameters.

C++ member variables can be assigned values using the constructor's initialization list, as well as in the constructor itself like in the other languages. The initialization list starts with a colon after the parameters to the constructor and is followed by calls to the field constructors using the constructor initialization syntax (See section 5.2).

In addition to constructors, classes can also have a destructor. The destructor is called before an object is destroyed. Thus the destructor can release any unmanaged resources allocated by the object. A class may only have one destructor and it never takes any parameters or return anything.

In C++ and C# the destructor has the same name as the class, but unlike the constructor it is prefixed by a tilde "~". Java uses the `finalize()` method to define its destructor and PHP has the `__destruct()` method.

Destructors are called automatically in C#, Java, and PHP by the garbage collectors when objects are no longer used. However, in the case of C++ the destructor is only invoked for stack-based objects when the object goes out of scope. The destructor for objects in the heap must be called explicitly with `delete`, as otherwise the system would have no way of knowing when an object could be gotten rid of.

Example 20. Defining constructors and destructors.

Java

```
class MyClass
{
  public MyClass() {}
  public void finalize() {}
}
```

C#

```
class MyClass
{
  public MyClass() {}
  ~MyClass() {}
}
```

C++

```
class MyClass
{
  public:
   MyClass();
  ~MyClass();
};

MyClass::MyClass()  {}
MyClass::~MyClass() {}
```

PHP

```
class MyClass
{
  function __construct() {}
  function __destruct()  {}
}
```

7.1.3 Class Members

Classes in all four languages can contain fields and methods. In the case of PHP, these are the only two class members that are allowed. C++ classes have a little bit more variety due to the addition of the enum type and overloaded operators[46].

The enum type is also available in Java and C#. These languages are also the only two that allow interfaces and other classes to be contained in a class. Although C++ does not allow a class to directly enclose another class, it is legal to define classes inside of functions. This is also allowed in Java, but not in C# or PHP.

C# has by far the largest variety of class members. In addition to those already mentioned, classes may also contain delegates, events, properties, indexers, operator overloading methods, and implicit/explicit type conversion methods.

Table 7. Allowed class members.

C++	PHP	Java	C#	Description
fields	fields	fields	fields	class variables
methods	methods	methods	methods	class functions
		interface	interface	class contract
		class	class	nested class
enum		enum	enum	constant container
operators			operators	overloaded operators
			delegates, events	function pointers
			properties, indexers	accessor methods
			implicit, explicit	type conversion

7.2 Access Modifiers

Each top-level* and class level member has an accessibility level which determines where that member will be visible. As can be seen in the table below, all four languages have the three access modifiers: `public`, `protected`, and `private`. Members in Java can also have package private access, which cannot be declared explicitly using a keyword. Package private corresponds to `internal` access in C# and grants access within the entire assembly or package.

The `protected` modifier in Java has a different meaning from the other three languages in that it grants access within the package as well as in derived classes, which is the same as `protected internal` in C#. In contrast, the `protected` access level used in the other three languages only grants access inside the defining class and in deriving classes. Java does not have an equivalent to this.

Table 8. Class level access modifiers.

C++	PHP	Java	C#	Accessible from
public	public	public	public	Anywhere
protected	protected		protected	Inside defining or derived class
private	private	private	private	Inside defining class
		default	internal	Own assembly/package
		protected	protected internal	Own assembly and derived classes

Java has package private access by default for all class and top-level members. Class members in C++ and C# have `private` access by default, while PHP class members become `public` without an access modifier. Namespace members have `internal` access by default in C#, while such members cannot have any access modifiers in C++ or PHP. In addition to `internal` in C# and package private in Java, top-level elements can also be granted `public` access in both languages, but nothing else.

The syntax for using access modifiers is the same in all languages, except for C++. In the case of C++, the access modifier is followed by a colon and becomes the default modifier for all subsequent members. In the other languages an access modifier only applies to a single member. Fields in PHP must have an explicit access modifier or be declared using the deprecated `var` keyword which gives them `public` access. Fields cannot be declared by only using the field's name as is done when declaring local variables.

Example 21. Using access modifiers.

Java

```
class MyClass {
  int y; // package private
  public int x;
}
```

C#

```
class MyClass {
  int y; // private
  public int x;
}
```

C++

```
class MyClass {
  int x; // private
 public: int y;
};
```

PHP

```
class MyClass {
  var $y; // public
  public $x;
}
```

* A top-level member is declared directly in a namespace (See section 8.1)

7.3 Inheritance

Inheritance enables a class to reuse accessible members from another class. The child class also gains the ability to be used instead of the parent class. C#, Java, and PHP only support a single-inheritance of classes, while C++ supports multiple inheritance. The reason for this is because “multiple inheritance causes more problems and confusion than it solves”[47]. Most notably, the ambiguity that arises when two parents defines the same method, known as the diamond problem[48].

Java and C# uses a single-rooted class hierarchy where all objects ultimately inherit from the root class `Object`. Therefore, if no base class is specified then the class will implicitly inherit from `Object`. This does not apply to the fundamental types in Java which are disjoint from the object model, while C# provides a completely unified type system where all types are derived from `Object`. In C#, the fundamental types are merely aliases for objects, for example `int` is actually an alias for `System.Int32` (a `struct`). C++ and PHP have no single root class, although the object data type exists in PHP.

As can be seen in the examples below, Java and PHP uses the `extends` keyword when utilizing inheritance, while C++ and C# uses the colon operator. Another difference is that when a class is inherited in C++ it is possible to change the access level of the inherited members. For the other languages `public` inheritance is the default and is the only inheritance level which let all members keep their original access[49]. However, the default inheritance level in C++ is `private`, which gives all inherited members `private` access.

Example 22. Inheriting a class.

Java

```
class Apple extends Fruit {}
```

C#

```
class Apple : Fruit {}
```

C++

```
class Apple : public Fruit {};
```

PHP

```
class Apple extends Fruit {}
```

7.3.1 Overriding

In Java, class methods are always virtual. Therefore, redefining an inherited class method in Java will *override* the parent’s implementation. This means that the method will be redefined both upwards and downwards in the class hierarchy, so even if the class is upcast the redefined method will still get called. In contrast, redefining an instance method will only *hide* the parent’s method. If the child class is then upcast to the parent’s type, then the parent’s definition will be used.

In C# and C++, methods are non-virtual by default and can only be made `virtual` by explicitly specifying this keyword. This means that both the hide and override mechanisms can be applied to any methods in C++ and C#. An inherited method in C# can be hidden using the `new` keyword; additionally if the parent method is marked as virtual it can be overridden with the override keyword. In contrast, C++ will implicitly override the parent’s method if it is declared with the virtual modifier, and implicitly hide it if it is non-virtual.

Redefining inherited methods is also allowed in PHP, but because of the loose typing concepts such as hiding and overriding are not relevant. There is no way to encapsulate a child class in a parent container, because attempting to do so would automatically convert the container to the child’s type.

Example 23. Overriding and hiding parent methods.

Java

```
class A
{
   public void F() {}
   public static void G() {}
}
class B extends A
{
   // overrides
   public void F() {}

   // hides
   public static void G() {}
}
```

C#

```
class A
{
   public void F() {}
   public virtual void G() {}
}
class B extends A
{
   // hides
   public new void F() {}

   // overrides
   public override void G() {}
}
```

C++

```
class A
{
  public:
   void F() {}
   virtual void G) {}
};
class B : public A
{
   // hides
   public void F() {}

   // overrides
   public void G() {}
};
```

PHP

```
class A
{
  function F() {}
}
class B extends A
{
  // redefines
  function F() {}
}
```

7.3.2 Final

Java, PHP, and C# all provide class modifiers to prevent a class from being inherited. The `final` modifier is used in Java and PHP, while C# has the `sealed` keyword. In addition to this functionality, the `final` keyword can also be used as a method modifier, both in Java and PHP. It prevents the method from being overridden by derived classes. To achieve the same result in C# and C++ the methods just have to not be marked as `virtual`.

Example 24. Non-virtual methods and classes that cannot be inherited.

Java

```
final class A
{
   final public void F() {}
}
```

C#

```
sealed class A
{
   public void F() {}
}
```

C++

```
// not supported
```

PHP

```
final class A
{
  final function F() {}
}
```

7.3.3 Calling Constructors

All four languages provide a way to call a base class constructor with specific parameters. C# and Java also allow calling one constructor from another constructor within the same class[50]. However, this constructor chaining cannot be done in C++. Additionally, constructor chaining is not supported in PHP since this language has no method overloading and therefore there is only one constructor per class.

In C# and Java the first line of a constructor must either be a call to another constructor or a call to the base class constructor. If the first line is neither of these, then the compiler automatically inserts a call to the base class's default constructor. Invoking another constructor is done using the `this` keyword in both languages, while calling the base class constructor is done using `super` in Java and `base` in C#. Unlike Java, the `this()` call in C# is done by placing the call before the constructor's body, similar to the constructor's initialize list in C++.

The default base class constructor is not implicitly called in C++. Instead, the constructor must be explicitly invoked in the beginning of the constructor's initialization list using the parent's class name. PHP is the only language where the parent's constructor is inherited. Therefore the base class constructor only needs to be called if the child class also defined a constructor. In order to invoke a parent's constructor its class name is followed by the scope resolution operator "`::`" and a call to the constructor. This call does not have to be the first line of the constructor, as in all the other three languages.

A minor difference between the four languages is that the `super` and `base` keywords in Java and C# only allow access to methods in the immediate parent class, i.e., a single level up in the hierarchy. However, base-class scoping in C++ and PHP allows access to methods that are deeper in the hierarchy.

Similar to C++, if a class constructor is not defined in C#, Java, or PHP, a default constructor with no parameters is automatically generated. The default constructor in C++ only allocates memory for the object, but does not initialize the fields, while the other three language's default constructor initializes all fields to their default values.

Example 25. Calling base class constructors and constructor chaining.

Java

```
class Shape
{
  public String name;
  public Shape(String a)
  { name = a; }
}
class Square extends Shape
{
  int x;
  public Rectangle()
  { this(10); }
  public Rectangle(int a)
  {
    super("Box");
    x = a;
  }
}
```

C#

```
class Shape
{
  public string name;
  public Shape(string a)
  { name = a; }
}
class Square : Shape
{
  int x;
  public Rectangle() : this(10) {}
  public Rectangle(int a)
  {
    base("Box");
    x = a;
  }
}
```

C++

```
#include <string>
using namespace std;
class Shape
{
 public: string name;
  Shape(string a)
  { name = a; }
}
class Square : Shape
{
  public: int x;
   Rectangle() :
    Shape("Box"), x(10) {}
   Rectangle(int a) :
    Shape("Box"), x(a) {}
}
```

PHP

```
class Shape
{
  public $name;
  function __construct($a)
  { $this->name = $a; }
}
class Square extends Shape
{
  var $x;
  function __construct($a = 10)
  {
    Shape::__construct("Box");
    $this->x = $a;
  }
}
```

7.4 Interface

An interface is used to specify members that deriving classes must implement. This is a powerful language feature avaliable in Java, PHP, and C#. Although classes in these languages may only inherit from a single base class, they can implement any number of interfaces. This enables multiple inheritance of types, while avoiding the problems associated with multiple inheritance of implementation, i.e., there can be only one applicable implementation.

In Java and PHP an interface can contain method signatures* and constants, while in C# they can contain signatures of methods, properties, indexers, and events. The signatures does not have any implementations, instead their bodies are replaced by semi-colons. All interface members are implicitly marked `public`. Explicitly declaring members as public is optional in Java and PHP, but illegal in C#. Fields in Java interfaces are also implicitly marked with `static final`, forcing them to become compiler-time constants.

To specify what interfaces a class must implement the interfaces are placed in a comma-separated list after the class name. In C#, the interfaces are mixed in with the base class, while in PHP and Java the interfaces are listed after the `implements` keyword. By convention, the interfaces are placed after the base class.

* A method signature includes the method's name, the number and type of its parameters, and its return type.

Example 26. Using and declaring interfaces.

Java

```
interface MyInterface
{
  int MyMethod();
  int PI = 3.14;
}

class C implements MyInterface {…}
```

C#

```
interface MyInterface
{
  int MyMethod();
  int MyProperty { get; set; }
  int this[int index] { get; set; }
  event System.EventHandler MyEvt;
}

class C : MyInterface {…}
```

C++

```
// not supported
```

PHP

```
interface MyInterface
{
    public function MyMethod();
    const PI = 3.14;
}

class C implements MyInterface {…}
```

7.4.1 Explicit Interface Implementation

The ambiguity problem with multiple inheritance still remains with interfaces. If two interfaces are implemented in the same class and have the same method signature there will be a conflict. In Java and PHP, there is no solution to this problem and a single method will be implemented for both interfaces. C# solves this problem through the use of explicit interface methods which allows an implemented method to be bound to a specific interface. Explicit interface methods will always be private and therefore hide the method from the primary class interface.

Example 27. Explicit interface implementation.

C#

```
public interface INetwork { void Close(); }
public interface IFile    { void Close(); }

public class Node : INetwork, IFile
{
    void INetwork.Close() {}
    void IFile.Close() {}
}

// Usage
IFile f = new Node();
f.Close();
```

7.5 Abstract Class

A class declared as abstract can contain abstract class members. These classes are similar to interfaces in that they both define member signatures that deriving classes must implement and neither one of them can be instanciated. The key differences are that the abstract class can contain non-abstract members while the interface cannot, and that a class can implement any number of interfaces, but can only inherit from one class, abstract or not.

Abstract classes and methods exists in C#, Java, and PHP with the same syntax and functionality. The class is modified with the abstract keyword which permits the use of the abstract function modifier that allows a method to be left undefined. In

addition to methods – which is the only class member in Java and PHP that can be declared as abstract – C# also allows abstract properties, indexers, events, and classes.

C++ can also implement abstract classes using pure virtual functions. To create a pure virtual function the function's prototype is assigned the value zero (=0). This will force deriving classes to provide the implementation. Since C++ has multiple inheritance these functions can also be used to simulate interfaces.

Example 28. Abstract classes and methods.

Java

```
abstract class MyAbstract
{
  public abstract int F();
}
```

C#

```
abstract class MyAbstract
{
  public abstract int F();
}
```

C++

```
class MyAbstract
{
  public: virtual int F() = 0;
};
```

PHP

```
abstract class MyAbstract
{
  public abstract function F();
}
```

8. Expert Syntax and Semantics

8.1 Namespace

Namespaces are used to avoid naming conflicts and to organize types into groups in a hierarchy. They are defined in the same way in both C++ and C#, using the namespace keyword followed by an identifier and a code block containing the grouped members. In both of these languages multiple namespaces can be defined within a single source file and the same namespace may exist in several files.

Java uses packages to organize types. Unlike namespaces, a package by convention specifies the folder where the source file is located. Therefore, there can only be one package statement in each source file and it must be the first line of code in the file. This means there is no way to declare members belonging to different packages within a single sourcefile as can be done with namespaces.

As of version 5.3 PHP also includes namespaces[51]. They can be declared either as in C++ with a code block, or with a statement as in Java where all code following that statement belongs to the namespace. If namespace declarations are used in a script, then all code in that file must belong to a namespace. The first namespace therefore needs to be declared in the first line of the script. Although any PHP code can be contained within a namespace, only classes, functions, and constants are actually affected by them[51].

To access a namespace member from outside of that namespace, the dot operator is used in both C# and Java. While C++ uses the scope resolution operator "`::`" and PHP uses the backslash character "\" for this purpose. Like C++, C# also has the "`::`" operator, but calls it the namespace alias qualifier operator. C# uses this operator to qualify the global namespace (`global::`) if there are any ambiguities[52].

Example 29. Namespace declarations.

Java	C#
`package Hello.World;`	`namespace Hello.World { … }`
C++	**PHP**
`namespace Hello::World { … }`	`namespace Hello\World;` `namespace Hello\World { … }`

8.1.1 Namespace Members

PHP allows any code to be contained directly within a namespace. This not only includes classes, interfaces, functions and data, but also executable statements. In the other three languages statements may only be used within functions. C# and Java only allow certain container types to be declared as top-level members. Java has three of these: `class`, `interface`, and `enum`. Adding to that list C# has `struct` and `delegate`, as well as nested namespaces. The top-level container types allowed in C++ include: `class`, `struct`, `union`, `enum`, and nested namespaces. C++ also permits functions and data to be declared globally.

Table 9. Allowed top-level members.

C++	PHP	Java	C#	Description
class	class	class	class	object template
	interface	interface	interface	class contract
enum		enum	enum	constant container
			delegate	function pointer
namespace			namespace	nested namespace
struct/union			struct	lightweight class
functions	functions			global function
variables	variables			global data

8.1.2 Using Namespaces

Accessible namespace members can always be referenced using their fully qualified name*. However, it is generally easier to import the namespace so that the members can be referenced using only their name. Importing namespaces in C# is done with the `using` keyword, Java has `import,` and C++ has `using namespace`. The C# and Java keywords must both be defined globally at the top of the source file, while the C++ keywords may be used anywhere, i.e., globally or even inside of code blocks.

Since C#, Java, and PHP are all run in managed environments their standard library functions are always readily availiable. However, in C++ programs merely importing a namespace does not provide access to the members included in that namespace. Due to the requirement that prototypes have to be available the programmer also needs to include either the relevant prototypes or an include file which has these declarations – the later can be done using the `#include` directive.

PHP does not have any means for importing namespaces, as the standard library functions are all part of the global namespace there is no great need for explicitly importing namespaces. However, to import user-defined functions and types PHP does provide a built-in `include()` method, which similarly to `#include` in C++ it inserts the content of the indicated file. An important difference between these two includes is that in PHP the include is done during compilation and in C++ it is done before compilation.

Java allows importing of packages as well as specific classes. In contrast, C# and C++ only allow importing of whole namespaces. Java is also the only one of the four languages that can perform a static import – in order to import all static members of a class so that they can be used without having to specify the class name.

* A fully qualified name specifies the absolute path to a member by including the namespace hierarchy where that member is located. For example, the fully qualified name for the C# `WriteLine()` function is `System.Console.WriteLine()`.

Example 30. Importing code and/or namespaces.

Java

```
import java.util.List;
import java.util.*;
import static java.awt.Color.*;
```

C#

```
using System;
```

C++

```
#include <iostream>
using namespace std;
```

PHP

```
include("MyClass.php");
```

8.1.3 Alias

Aliasing is used to create alternative names for existing namespaces and types. This is typically done to shorten fully qualified names or to avoid namespace collisions. Type aliasing also allow aliased types used throughout a program to be changed easily from a single location in the code. C++, C#, and PHP all allow types and namespaces to be aliased, while Java does not. C++ uses `typedef` to alias a type and `namespace` to alias a namespace. The `typedef` keyword is followed by the type and then the alias, while `namespace` is followed by the alias which is assigned a namespace.

C# has the `using` keyword for creating both type and namespace aliases, which both are declared in the same way as the C++ `namespace` alias. PHP also uses the same keyword for both type and namespace alias, namely `use`. This keyword is followed by the path to the type or namespace separated by backslashes and then the `as` keyword along with the alias. If the `as` part is left out, then the type or namespace name itself will be used as the alias.

Example 31. Type and namespace aliasing.

Java

```
// not supported
```

C#

```
using MyType = My.Name.MyClass;
using MyAlias = My.Name;
```

C++

```
typedef My::Name::MyClass MyType;
namespace MyAlias = My::Name;
```

PHP

```
use My\Name\MyClass as MyType;
use My\Name as MyAlias;
```

8.2 Preprocessor

The C++ preprocessor directives tells the preprocessor to perform specific actions before the actual compilation takes place. Two commonly used directives are `#define` to replace tokens in the code and `#include` to insert the contents of other files into the source file[53]. There is also directives for conditional compilation using `#if`, `#elif`, `#else`, and `#endif`, which can be useful for debugging and cross-platform portability[53].

C# does not have a preprocessor, but it does include a smaller set of the C++ preprocessor directives, mainly used for conditional compilation[54]. Specifically, there is no `#include` directive and the `#define` constants can only be used to specify compilation conditions. Java does not have any preprocessor directives and neither does PHP. Even though PHP stands for “PHP hypertext preprocessor” this simply refers to the HTML being dynamically generated and not that PHP would have a preprocessor similar to C++. PHP does include some functions similar to

preprocessor directives, such as the `declare()` and `include()` functions meantioned earlier (See sections 5.3 and 8.1.2)[55]. However, since PHP is normally not compiled until runtime[56] these functions do not give the performance benefit of preprocessor directives.

Example 32. Using preprocessor directives.

C++

```
#define DEBUG
#if DEBUG
  // …
#else
  // …
#endif
```

C#

```
#define DEBUG
#if DEBUG
  // …
#else
  // …
#endif
```

Java

```
// not supported
```

PHP

```
// not supported
```

8.3 Exception Handling

Exception handling is used to deal with unexpected situations that may occur in a program. To handle exceptions all four of the languages use the try-catch statement. This statement consists of a `try` block containing the code that may cause the exception, and one or more `catch` blocks for handling exceptions. Java and C# also have the `finally` block to clean up resources allocated in the try block.

Exceptions are thrown using the `throw` keyword in all four languages. Only classes inheriting from `Throwable` can be thrown in Java, while classes deriving from `Exception` may be thrown in C# and PHP. In C++ any data type can be given as an argument to `throw`. However, there are also standard exceptions which are derived from the class `exception`.

Example 33. Generating and handling exceptions.

Java

```
try { throw new IOException(); }
catch (IOException e) { }
catch (Exception e) { }
finally { }
```

C#

```
try { throw new IOException(); }
catch (IOException e) { }
catch (Exception e) { }
finally { }
```

C++

```
try { throw "Failed"; }
catch (string e) {}
catch (...) {}
```

PHP

```
try { throw new Exception(); }
catch (Exception $e) {}
```

8.3.1 Exception Specification

An exception specification is a guarantee that a function will only throw the specified exceptions. This mechanism exists in Java and C++, but not in C# or PHP. Java specifies exceptions after the function parameter list using the `throws` keyword followed by the exception types. Exceptions in Java are grouped into two categories; checked and unchecked, depending on whether or not they need to be specified. Methods that throw checked exceptions will not compile unless these exceptions are specified and the calling method catches them. Unchecked exceptions on the other hand do not have to be caught or declared with the `throws` clause.

Exceptions in C++ are specified using the `throw` keyword similarly to Java. However, Java's exception specification is superior to that in C++, because Java's specifications are checked and enforced at compile time. C++ compilers do not force exceptions to be specified nor will they complain if the wrong exceptions are thrown. The main reason to specify exceptions is to inform the programmer of what exceptions a function may throw[57]. The exception specification in C++ is therefore only marginally better than in C# and PHP, as in the later two languages there is no way to programmatically specify what exceptions a function may throw. Instead, in these languages it is up to the developer of the methods to document this.

Example 34. Specifying exceptions.

Java	C++
`void F() throws IOException {…}`	`void F() throw (int) {…}`
C#	**PHP**
`// not supported`	`// not supported`

8.4 Enumerator

An enumerator is a special type containing a fixed list of named constants. Compared to using ordinary constants, the enum type forces the programmer to clearly specify what constant values are allowed. This makes enumerators more readable and easier to use than constants. The enum data type exists in C++, Java, and C#. The type is not supported in PHP, but associative arrays can be used to achieve the same result. However, associative arrays can be updated at runtime – while enumerators are static.

Enums in C++ and C# are very similar and can be seen as syntacic sugar around the integer types. They both allow the data type of the constants to be defined and the default constant values to be overriden. Java enums in contrast to C# and C++ are much more powerful. Introduced in Java 1.5, the `enum` type is a special class that extends `java.lang.enum` and can have fields and methods just as any other class[58].

Example 35. Creating an enumerator.

Java	C#
`enum State { Run, Stop }`	`enum State : byte` `{ Run = 0, Stop = Run + 5 }`
C++	**PHP**
`enum State : short` `{ Run = 0, Stop = Run + 5 };`	`// workaround` `$State = array("RUN"=>0, "STOP"=>5);`

8.5 Struct

The `struct` data type exists in both C# and C++, but are significantly different. In C++ a `struct` is exactly like a class, except that the default access level is public rather than private[59]. In contrast, the `struct` keyword in C# is used to create a stack-based value type, which can be seen as a lightweight class. C#'s struct does not

allow inheritance nor can they be inherited by classes, but they can implement interfaces. Structs do not exist in PHP or Java.

Example 36. Creating a struct.

C++

```
struct Point
{
  int x, y;
};
```

C#

```
struct Point
{
  public int x, y;
}
```

Java

```
// not supported
```

PHP

```
// not supported
```

8.6 Operator Overloading

Operator overloading allows operators to be redefined and used where one or both of the operands are of the user-defined class. When done correctly, this can simplify the code and make user-defined types as easy to use as the fundamental types. This feature is supported in both C++ and C#, but not in Java or PHP. However, the `String` class in Java does have the "`+`" and "`+=`" operators overloaded, but they exist as a special built-in case.

C++ supports overloading for all operators shown in the operator table earlier (See Table 5 on page 17), along with a couple of others (`new`, `delete`, `[]`, `->`, `*`, `->`, `()`, `&`, and comma)[60]. C# allows overloading of almost all operators in the same table (minus `&&`, `||`, and adding: `true`, `false`)[61]. The combined assignment operators cannot be overloaded explicitly in C#, but their semantics are changed as their corresponding arithmetic or bitwise operators are overloaded. Also, certain C# operators must be overloaded in pairs (`== !=`, `< >`, `<= >=`, and `true false`).

The syntax for performing operator overloading is similar in C++ and C#. Both languages use the `operator` keyword followed by the operator symbol as the function name. However, in C# the function is static, thus one of the parameters must be of the containing type. This is different from C++ where the operator is implicitly applied to the current instance of the class. The operator overloading function therefore requires one less parameter in C++ than in C#.

Example 37. Operator overloading.

C++

```
class MyNum
{
 public:
  int val;
  MyNum(int i) : val(i) {}

  MyNum operator+(MyNum &a)
  { return MyNum( val + a.val ); }

  MyNum operator++()
  { return MyNum( val + 1 ); }
};
```

Java

```
// not supported
```

C#

```
class MyNum
{
  public int val;
  public MyNum(int i) { val = i; }

  public static MyNum operator +(MyNum a, MyNum b)
  { return new MyNum(a.val + b.val); }

  public static MyNum operator ++(MyNum a)
  { return new MyNum(a.val + 1); }
}
```

PHP

```
// not supported
```

8.7 Implicit and Explicit Conversions

C# allows the programmer to explicitly define custom type conversions for an object. The function signature looks similar to that used in unary operator overloading, but instead of an operator symbol the return type is specified and the parameter is the type that needs to be converted. The function must also be defined as either explicit or implicit. The `explicit` keyword declares a conversion that must be invoked with an explicit cast and the `implicit` keyword declares an implicit conversion.

Custom defined implicit and explicit conversions do not exist in PHP or Java, but they can be defined in C++. However unlike C#, conversion from the object type cannot be implemented in C++, only conversions to the object type are allowed. Implicit conversions are defined by overloading a constructor to take a single parameter of the desired type. When that type is assigned to an instance of the class, then the constructor will implicitly be called to perform the conversion. To define an explicit conversion the `explicit` constructor modifier is used, which specifies that the constructor may only be used when an explicit cast is present.

Example 38. Custom type conversions.

C++

```
#include <string>
using namespace std;

class MyType
{
  int i; string s;
  public:
   MyType(string a) { s = a; }
   explicit MyType(int a) { i = a; }
};

void test()
{
  MyType A = "Hi"; // implicit
  MyType B(10);    // explicit
  B = (MyType)10;  // explicit
  B = 10;          // illegal
}
```

Java

```
// not supported
```

C#

```
class MyType
{
    private int i;
    public MyType(int a) { i = a; }

    public static implicit operator
    int(MyType t) { return t.i; }

    public static explicit operator
    MyType(int i) { return new MyType(i); }

    void test()
    {
        MyType x = (MyType)5; // explicit
        int i = x;            // implicit
    }
}
```

PHP

```
// not supported
```

8.8 Properties

Accessor methods (getters and setters) are commonly used to provide safe access to fields. To make this easier, C# has special built-in methods called properties. The first part of a property looks like a field declaration, but is followed by a code block which includes a `get` and/or a `set` accessor. These accessors in turn have their own code blocks which return and assign the corresponding property's value. C# properties have the advantage over normal methods that they are implemented as methods, but used as though they are fields. This allows them to hide implementation or verification code; while at the same time giving easy access to the fields[62].

Example 39. Using accessor methods.

Java

```
class Time
{
  private int seconds;

  public int getSec()
  { return seconds; }

  public void setSec(int s)
  { seconds = s; }
}
```

C#

```
class Time
{
  private int seconds;

  public int sec
  {
    get { return seconds; }
    set { seconds = value; }
  }
}
```

C++

```
class Time
{
  int seconds;
 public:
  int getSec()
  { return seconds; }

  void setSec(int s);
  { seconds = s; }
};
```

PHP

```
class Time
{
  var $seconds;

  function getSec()
  { return $seconds; }

  function setSec($a)
  { $seconds = $a; }
}
```

8.8.1 Indexers

An indexer is a special kind of accessor method available in C# that allows an object to be accessed like an array. This can be useful for example when a class is created to hold a collection type. Indexers are declared in the same way as properties, except that the `this` keyword is used instead of a name and their accessors takes one or more parameters.

The effect of the indexer's get accessor can be duplicated in C++ by overloading the square brackets. However, Java and PHP have no similar feature, thus the programmer would have to use normal accessor methods.

Example 40. Using indexers.

C++

```
class MyArray
{
  int a[10];
 public:
  int operator[](int i)
  { return a[i]; }
};
```

C#

```
class MyArray
{
  object[] data
  = new object[10];

  public object this[int i]
  {
    get { return data[i]; }
    set { data[i] = value; }
  }
}
```

Java

```
// not supported
```

PHP

```
// not supported
```

8.9 Generics

Generics provide a way to make a class or function operate with any type. This is done by adding one or more special type parameters to the class or method. This allows greater code reuse, type safety, and increased performance[63].

Generics are a relatively recent feature that were not added until C# version 2.0 and Java version 1.5[63][64]. PHP does not support generics, but C++ has templates which provide the same functionality. The syntax for generics is very similar in Java and C# and only slightly different than templates in C++. To use generics the type parameter (typically called "`T`") is surrounded by angle brackets after the class or method identifier. All uses of this generic type inside the body will be replaced when the class is instantiated or the method is called. In C++ the format for class and function templates is "`template <class type>`" followed by the class or function declaration[65].

Generic programming is implemented differently in all three languages. One difference is that in C++ separate copies of the class or function are generated for each type parameter when compiled, which is not the case in Java or C#[66]. Another difference is that Java does not allow generics with fundamental types, while any types are allowed in C# and C++. In C#, generics can be specified for interfaces, events and delegates, in addition to classes and methods.

Example 41. Using generics.

Java

```
class MyGeneric<T>
{
  void Add(T t) { }
}

MyGeneric<Integer> g
 = new MyGeneric<Integer>();
```

C#

```
class MyGeneric<T>
{
  void Add(T t) { }
}

MyGeneric<int> g
 = new MyGeneric<int>();
```

Java

```
template <class T>
class MyGeneric
{
  void Add(T t) { }
};

MyGeneric<int> g;
```

PHP

```


// not supported
```

9. Software Development

In addition to the programming syntax tutorials there were also two software development tutorials produced in this project; one for Java and another for C# (See Appendix 1). Both of these software development tutorials demonstrate how to build a basic text editor and their solutions will be compared in this chapter. The focus of this tutorial series is to teach practical programming techniques, rather than programming basics. Therefore, the tutorials do not include any lengthy explanations of the GUI controls or standard library functions that are used. Instead, those areas will be covered in future tutorial series'.

One of the challenges in making these software development tutorials was to keep the sections independent of each other. As mentioned in the beginning of this thesis (See section 2.1 on page 3) this was one of the criteria for making a tutorial easy to update. The reason why this criteria was difficult to implement is because each section in an example application tutorial literary builds upon the previous ones. In order to solve this I had to structure the tutorials so that each section only showed the code that was relevant to that section as much as possible, without displaying more than collapsed code blocks of methods implemented in previous sections. This technique of not showing previously implemented code was not always feasible to use, but it did allow large parts of the sections to be independent of each other.

9.1 Section I – Multipad

The first section in this tutorial series provides a short introduction and shows the student how to set up the project. The name I chose for the project – Multipad – comes from a Sourceforge project[67] I had created a long time ago by the same name. As in the programming syntax tutorials the development environments used for these practical tutorials are Visual Studio for C# and Netbeans for Java. Microsoft Visual Studio was the obvious choice for the tutorial that would demonstrate real life C# development, because it is by far the most widely used IDE among software developers[68]. Since Visual Studio is also available in free lightweight versions, such as Visual C#, there was also no reason to use a free alternative, such as SharpDevelop[69]. My choice of using Netbeans for the practical Java tutorial was not as easy to make since both Eclipse[70] and IntelliJ IDEA[71] are great alternatives. Still, I decided to keep using Netbeans because in contrast to IDEA Netbeans is free, and in contrast to Eclipse it includes a built-in visual GUI builder that would make designing the text editor's interface much easier.

9.2 Section II – Interface

The second section creates the interface for the text editor, which consists of a textbox and a menu. In both the Java and C# tutorials these interfaces are created using visual builders. The textbox element corresponds to the `JTextArea` component* in Java and to the `TextBox` control† in C#. The menu element is added using the `JMenuBar` component in Java and the `MenuStrip` control in C#. To this menu a couple of standard menu items are then added together with events for when the items are clicked. In the C# version this type of event is called `click` and in Java it is called `actionPerformed`.

To simplify the code the same event handler is used for all menu items. The C# version of the event handler determines which menu item is clicked using a switch statement. The expression for the switch statement is the name of the object that triggered the event (passed through the `sender` argument) and the labels in the switch are the names of all the menu items. This solution could not be used in Java

* A Java class inheriting from javax.swing.JComponent is called a component.
† A C# class inheriting from System.Windows.Forms.Control is called a control.

since the switch statement in this language cannot take a string as its expression. Instead, a series of if-statements are used, which compare the menu items to the source of the event.

At the end of the second section the edit menu items are implemented using methods of the `JTextArea` and `TextBox` classes. The `JTextArea` does not include an `undo()` method as the `TextBox` class does, but otherwise both classes have methods for cut, copy, paste, and select all. The difference in the casing used for the method names in the code shown below is intentional, because Java and C# have different naming conventions.

Example 42. Implementing a menu event handler.

Java

```
private void menuAction(
ActionEvent evt)
{
 Object s = evt.getSource();
 if (s == mNew)
  newFile();
 else if (s == mOpen)
  openFile();
 else if (s == mSave)
  saveFile();
 else if (s == mSaveAs)
  saveFileAs();
 else if (s == mExit)
  System.exit(0);
 else if (s == mCut)
  txt.cut();
 else if (s == mCopy)
  txt.copy();
 else if (s == mPaste)
  txt.paste();
 else if (s == mSelectAll)
  txt.selectAll();
}
```

C#

```
private void Menu_Click(object
sender, EventArgs e) {
 switch (((ToolStripMenuItem)
           sender).Name) {
  case "mNew":
   NewFile(); break;
  case "mOpen":
   OpenFile(); break;
  case "mSave":
   SaveFile(); break;
  case "mSaveAs":
   SaveFileAs(); break;
  case "mPrint":
   PrintFile(); break;
  case "mExit":
   Application.Exit(); break;
  case "mUndo":
   txt.Undo(); break;
  case "mCut":
   txt.Cut(); break;
  case "mCopy":
   txt.Copy(); break;
  case "mPaste":
   txt.Paste(); break;
  case "mSelectAll":
   txt.SelectAll(); break;
  }
```

9.3 Section III – New Open

The third section implements the new-file and open-file methods. The first of these methods clears the textbox after giving the user the option of saving his or her changes (See Example 42). In order to determine if any changes has been made the C# version uses the `Modified` field of the `TextBox` control, while the Java version only checks if the `JTextArea` is empty. This is because `JTextArea` does not include a modified field, so this feature was originally left out of the Java version. However, to make the Java text editor more similar to the C# editor the fifth section of the Java tutorial adds a modified field.

If the textbox has been modified a confirm dialog box is displayed to the user. This dialog box is created using the `JOptionPane` class in Java and the `MessageBox` class in C#. If the user answers yes in the dialog box the still unimplemented save-file method is called. Whatever answer the user gives the textbox is then cleared. To do this the Java version sets the text in the textbox to an empty string using the `setText` method of the `JTextArea`. The C# version does the same using the `Text` property of the `TextBox`.

The Java version introduces the `filepath` string field in this section. This field will hold the path to the currently opened file if any, and should therefore be cleared when the new-file method is called. The C# version does not add this field until the fourth section.

Example 43. Creating a new-file method.

Java

```
private void newFile()
{
 if (!txt.getText().isEmpty())
 {
  int r = JOptionPane.
  showConfirmDialog(this,
  "Save current document?",
  "Question",
  JOptionPane.YES_NO_OPTION);

  if(r == JOptionPane.YES_OPTION)
   saveFile();
 }
 txt.setText("");
 filepath = null;
}
```

C#

```
private void NewFile()
{
 if (txt.Modified)
 {
  DialogResult r =
  MessageBox.Show(this,
  "Save current document?",
  "Save",
  MessageBoxButtons.YesNo,
  MessageBoxIcon.Question,
  MessageBoxDefaultButton.
  Button1);

  if(r == DialogResult.Yes)
   SaveFile();
 }
 txt.Text = "";
}
```

To implement the open-file method an open file dialog box is needed. In C# this functionality is given by the `OpenFileDialog` class and in Java through the `JFileChooser` class. If the user cancels the dialog box the method returns, otherwise the new-file method is called in order to clear the textbox. The file selected in the dialog box is then read into the textbox using the `ReadToEnd()` method of the `StreamReader` class in the C# version. The code needed to read a file in Java is a bit longer and so this functionality is delegated to another method called `open()`.

In this `open()` method a `FileReader` object is constructed using the path to the selected file. This `FileReader` is then used to construct a `BufferedReader` in order to be able to read more than one character at a time from the file. Unlike the `StreamReader` class in C#, the `BufferedReader` does not have a method for reading all characters at once. Instead, the buffer is read one line at a time using the `readline()` method and the returned string is added to the `JTextArea` using the `append()` method. This is repeated until the `ready()` method returns false with the help of a while loop. Note that the `readline()` method does not return the newline character(s). Therefore, the Java version, which has to be able to run on more systems than Windows, needs to append the current systems line separator to each line that is read. Finally, the `open()` method updates the `filepath` string to the location of the opened file.

The C# version of the open-file method uses a finally block to close the file reader. This solution is not used in the Java version, because the `close()` method of the `BufferedReader` can thrown an `IOException` that must be caught. Therefore, in order to avoid having another try-catch statement in the finally block the buffer is closed at the end of the try block instead. In the case where a stream is successfully opened, but still throws an exception when it is read, the garbage collection will be responsible for closing the `BufferedReader`.

Example 44. Creating an open-file method.

Java

```
private void openFile()
{
 if(dFile.showOpenDialog(this) !=
  JFileChooser.APPROVE_OPTION)
  return;
 newFile();
 try {
  open(dFile.getSelectedFile().
   getPath());
 }
 catch (Exception e) {
  JOptionPane.showMessageDialog(
   this, e.getMessage());
 }
}

private void open(String file)
throws Exception
{
 BufferedReader in =
  new BufferedReader(
  new FileReader(file));
 String sep = System.
  getProperty("line.separator");
 while (in.ready())
  txt.append(in.readLine()+sep);
 in.close();
 filepath = file;
}
```

C#

```
private void OpenFile()
{
 if (dOpen.ShowDialog() !=
  DialogResult.OK)
  return;
 NewFile();

 StreamReader sr = null;
 try {
  StreamReader = new
   StreamReader(
   dOpen.FileName);
  txt.Text =
   sr.ReadToEnd();
  txt.Modified = false;
 }
 catch (Exception e) {
  MessageBox.Show(
  "Failed to open file.\n"
  + e.Message);
 }
 finally {
  if(sr!=null) sr.Close();
 }
}
```

9.4 Section IV – Save SaveAs

The fourth section shows how to implement the save-file and save-file-as methods for the text editor. The first of these methods saves the current document to the location specified in the `filepath` string. If this string is not set then the save-file-as method is called instead. The C# version saves the file using the `WriteLine()` method of a `StreamWriter` object that is constructed using the `filepath` string. Before the Java version can save the file it first has to construct a `FileWriter` object from the `filepath`, and then use that `FileWriter` to construct a `BufferedWriter`. The text can then be saved using the `write()` method of the `BufferedWriter` class. Just as in the open-file method the Java version closes the stream at the end of the try statement instead of from a finally clause as is done in the C# version.

Example 45. Creating a save-file method.

Java

```
private void saveFile()
{
 if (filepath == null)
 { saveFileAs(); return; }

 try {
  BufferedWriter out =
   new BufferedWriter(
   new FileWriter(filepath));
  out.write(txt.getText());
  out.close();
 }
 catch (Exception e) {
  JOptionPane.
  showMessageDialog(
   this, e.getMessage());
  }
}
```

C#

```
private void SaveFile()
{
 if (filepath == null)
 { SaveFileAs(); return; }

 StreamWriter sw = null;
 try {
  sw = new
   StreamWriter(filepath);
  sw.WriteLine(txt.Text);
  txt.Modified = false;
 }
 catch (Exception e)
 {
  MessageBox.Show(
  "Failed to save file.\n"
  + e.Message);
 }
 finally
 { if(sw!=null) sw.Close(); }
}
```

The save-file-as method asks the user for a location with the help of a save file dialog box. In the C# tutorial, a `SaveFileDialog` class is created to display this dialog box. The Java tutorial on the other hand can reuse the `JFileChooser` object created for the open-file method to show a save file dialog box. If the dialog box is canceled the method returns, otherwise the `filepath` string is updated to the selected location and the save-file method is called.

Example 46. Creating a save-file-as method.

Java

```
private void saveFileAs()
{
 if(dFile.showSaveDialog(this) !=
  JFileChooser.APPROVE_OPTION)
  return;
 filepath = dFile.
  getSelectedFile().getPath();
 saveFile();
}
```

C#

```
private void SaveFileAs()
{
 if (dSave.ShowDialog() !=
  DialogResult.OK)
  return;
 filepath = dSave.FileName;
 SaveFile();
}
```

9.5 Section V

The fifth section was the last section produced for this tutorial series during the project and it implements different features for the Java and C# versions. The C# section is labeled “Print” and goes a bit further than the Java tutorial by showing the student how to implement a `print()` method. The Java section is labeled “Modified” and adds a modified field to the text editor. This addition unfortunately makes section five dependent upon changes in sections two, three, and four, since section five needs to show those section’s code in order to implement the modified flag. In retrospective, this field should have been added in the second section as was done in the C# version.

10. Conclusions

This thesis project has shown that the MVT method is a viable alternative to the traditional screen recording method, at least in the area computer programming. The MVT method makes it easy to produce high quality tutorials using a streamlined workflow. The produced tutorials can be updated quickly and are convenient to use. The audio script makes it easy for the producer to keep the tutorials relevant and the tutorial production rate is fairly quick, although not as fast as producing tutorials using the screen recording method.

As for the e-learning wiki it did not receive any contributions during the thesis project, but on the other hand it did not cost anything to maintain it. At the very least the wiki makes it easier for visitors to report errors that they find in the tutorials. Given more time and better marketing I am fairly certain that the e-learning wiki idea would become successful.

10.1 Future Work

Although a substantial number of video tutorials have been added to PVT a lot of work still remains. For example, once the syntax tutorials have been completed tutorials covering programming libraries, user interfaces, and application development will also need to be made. In order for PVT to grow at a faster rate I will need to assemble a team to assist me with tutorial production, tutorial updating, administration, marketing, and so on. By bringing more people onto this project the MVT method will also likely come to evolve further. As I see it now there are still some areas of the method that can be improved – mainly by improving the audio quality and by automating the production of the various video formats.

Appendix A.

Table 10. List of video sections uploaded to PVT during the project.

Programming syntax sections	
CPP - 01 - Introduction CPP - 02 - Hello World CPP - 03 - Compile and Run CPP - 04 - Variables I CPP - 05 - Variables II CPP - 06 - Operators CPP - 07 - Pointers CPP - 08 - Arrays CPP - 09 - String CPP - 10 - Conditions CPP - 11 - Loops	ASP.NET - 01 - Introduction ASP.NET - 02 - Using ASP.NET ASP.NET - 03 - Hello World ASP.NET - 04 - Form Control ASP.NET - 05 - HTML Controls ASP.NET - 06 - Control Members ASP.NET - 07 - Web Controls ASP.NET - 08 - Events ASP.NET - 09 - Events II ASP.NET - 10 - User Controls ASP.NET - 11 - Validation Controls
Java - 01 - Introduction Java - 02 - Hello World Java - 03 - Compile and Run Java - 04 - Variables Java - 05 - Operators Java - 06 - String Java - 07 - Arrays Java - 08 - Conditions Java - 09 - Loops Java - 10 - Functions Java - 11 - Class Java - 12 - Static Java - 13 - Inheritance Java - 15 - Package and Import Java - 16 - Access levels	PHP - 01 - Introduction PHP - 02 - Using PHP PHP - 03 - Variables PHP - 04 - Operators PHP - 05 - Strings PHP - 06 - Arrays PHP - 07 - Conditions PHP - 08 - Loops PHP - 09 - Functions PHP - 10 - Class PHP - 11 - Inheritance PHP - 12 - Access Levels PHP - 13 - Static PHP - 14 - User Input PHP - 15 - Cookie PHP - 16 - Session
CSharp - 01 - Introduction CSharp - 02 - Hello World CSharp - 03 - Compile and Run CSharp - 04 - Variables CSharp - 05 - Operators CSharp - 06 - String CSharp - 07 - Arrays CSharp - 08 - Conditions CSharp - 09 - Loops CSharp - 10 - Functions CSharp - 11 - Class	CSharp - 12 - Inheritance CSharp - 13 - Overriding CSharp - 14 - Access Levels CSharp - 15 - Static CSharp - 16 - Properties CSharp - 17 - Indexers CSharp - 18 - Interface CSharp - 19 - Abstract CSharp - 20 - Namespaces CSharp - 21 - Enumerations CSharp - 22 - Exception Handling
Software development sections	
Java Example - 01 - Multipad Java Example - 02 - Interface Java Example - 03 - New Open Java Example - 04 - Save SaveAs Java Example - 05 - Modified	CSharp Example - 01 - Multipad CSharp Example - 02 - Interface CSharp Example - 03 - New Open CSharp Example - 04 - Save SaveAs CSharp Example - 05 - Print

References

[1] Mikael Olsson
Programming Video Tutorials (website)
http://www.programmingvideotutorials.com
Last accessed: 18 March 2009

[2] Mikael Olsson
PVT wiki (website)
http://www.programmingvideotutorials.com/wiki
Last accessed: 18 March 2009

[3] TechSmith Corporation, © 1995-2009
Camtasia Studio (software)
http://www.techsmith.com/camtasia.asp
Last accessed: 18 March 2009

[4] TIOBE Software BV, © 2009
TIOBE Programming Community Index for February 2009 (website)
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
Last accessed: 2 Mars 2009

[5] 3D Buzz, © 2008
3D Buzz (website)
http://www.3dbuzz.com
Last accessed: 18 March 2009

[6] AppDev Products, LLC, © 2008
Microsoft Training at AppDev: IT Training, Developer Training, Microsoft Certification Training, Microsoft .NET Training (website)
http://www.appdev.com
Last accessed: 18 March 2009

[7] CBT Nuggets, LLC, © 1999-2009
CBT Nuggets: Training for Cisco CCNA SQL MCSE VB.NET A+ Linux PMP & Many More IT Certification Exams! (website)
http://www.cbtnuggets.com
Last accessed: 18 March 2009

[8] KeyStone Learning Systems, LLC, © 2008
Online Video Training, Computer Based Training, eLearning, Certification, Classroom, Webinar, and Enterprise Training Solutions - KeyStone Learning Systems (website)
http://www.keystonelearning.com
Last accessed: 18 March 2009

[9] LearnVisualStudio.NET, © 2002-2009
ASP.NET, C#, Visual Basic Tutorials and Training on LearnVisualStudio.NET (website)
http://www.learnvisualstudio.net
Last accessed: 18 March 2009

[10] Learnkey, Inc., © 2009
LearnKey State of the Art Training (website)
http://www.learnkey.com
Last accessed: 18 March 2009

[11] Lynda.com, Inc., © 1995-2009

Tutorials – Online Training – Lynda.com (website)
http://www.lynda.com
Last accessed: 18 March 2009

[12] Total Training, Inc., © 2009
Total Training: Online Video Training, Adobe Photoshop, Flash, Dreamweaver and Microsoft Online Software Training (website)
http://www.totaltraining.com
Last accessed: 18 March 2009

[13] The Virtual Training Company
Online software tutorials, training CDs, Photoshop Tutorials, Dreamweaver Tutorials, Apple Tutorials from vtc.com (website)
http://www.vtc.com
Last accessed: 18 March 2009

[14] Don Tapscott and Anthony D. Williams
Wikinomics: How mass Collaboration Changes Everything (book)
Published: 28 Dec 2006, Penguin Group (USA)
ISBN-10: 1591841380

[15] Adobe Systems, Inc., © 2009
Adobe - Flash Player Statistics (website)
http://www.adobe.com/products/player_census/flashplayer/
Last updated: 7 Aug 2008

[16] One-Minute SCORM Overview for Anyone (website)
http://www.scorm.com/resources/oneminuteoverview/ OneMinuteOverview.htm
Last accessed: 18 March 2009

[17] Moodle.org: open-source community-based tools for learning (software)
http://moodle.org
Last accessed: 18 March 2009

[18] Will Park, IntoMobile, © 2005-2009
Apple iPhone - Disappointing LCD Screen? (article)
http://www.intomobile.com/2007/03/11/apple-iphone-disappointing-lcd-screen.html
Published: 11 March 2007

[19] Mikael Olsson
YouTube - ProgrammingVideos's Channel (website)
http://www.youtube.com/user/ProgrammingVideos
Last accessed: 18 March 2009

[20] TechSmith Corporation, © 1995-2009
Screen Capture | Snagit | Screen capture software by TechSmith (website)
http://www.techsmith.com/screen-capture.asp
Last accessed: 18 March 2009

[21] Sennheiser Communications
Sennheiser: PC 151 (hardware)
http://www.sennheiser.com/comm/icm_eng.nsf/root/05351
Last accessed: 18 March 2009

[22] Rick Brewster, dotPDN LLC, © 2008
Paint.NET - Free Software for Digital Photo Editing (software)
http://www.getpaint.net
Last accessed: 18 March 2009

[23] Audacity (software)
http://audacity.sourceforge.net/
Last accessed: 18 March 2009

[24] 1) TechSmith Corporation, © 1995-2009
TechSmith Screen Capture Codec (TSCC) (website)
http://www.techsmith.com/codecs/tscc/default.asp
Last accessed: 3 Mars 2009

2) TechSmith Corporation, © 1995-2009
Download Video Codecs (website)
http://www.techsmith.com/download/codecs.asp
Last accessed: 3 Mars 2009

[25] Time Inc., © 2009
It's a Wiki, Wiki World – TIME (article)
http://www.time.com/time/magazine/article/0,9171,1066904,00.html
Published: 29 May 2005

[26] Jakob Nielsen, © 2006
Participation Inequality: Lurkers vs. Contributors in Internet Communities (Jakob Nielsen 's Alertbox)
http://www.useit.com/alertbox/participation_inequality.html
Last accessed: 21 April 2009

[27] Wikipedia:About - Wikipedia, the free encyclopedia
http://en.wikipedia.org/w/index.php?title=Wikipedia:About&oldid=285003875
Last accessed: 21 April 2009

[28] The PHP Group, © 2001-2009
PHP Manual – Type Juggling (website)
http://www.php.net/language.types.type-juggling
Last updated: 27 Feb 2009

[29] Cprogramming.com, © 1997-2005
Common Programming Mistakes (website)
http://www.cprogramming.com/tutorial/common.html
Last accessed: 2 Mars 2009

[30] Microsoft Corporation, © 2009
Constant Values (website)
http://msdn.microsoft.com/en-us/library/357syhfh.aspx
Last accessed: 2 Mars 2009

[31] The PHP Group, © 2001-2009
PHP Manual – Constant (website)
http://www.php.net/constant
Last updated: 27 Feb 2009

[32] Roedy Green, © 1995-2009 Canadian Mind Products
Java Glossary – Constant (website)
http://mindprod.com/jgloss/constant.html
Last updated: 3 Jan 2008

[33] Andrew Hardwick
The C++ 'const' Declaration: Why & How (website)
http://duramecho.com/ComputerInformation/WhyHowCppConst.html
Last updated: 22 May 2006

[34] The PHP Group, © 2001-2009
PHP Manual – Comparison Operators (website)
http://www.php.net/operators.comparison
Last updated: 27 Feb 2009

[35] James Gosling and Henry McGilton, Sun Microsystems, Inc., © 1997
The Java Language Environment (white paper)
http://java.sun.com/docs/white/langenv/Simple.doc2.html#4107
Published: May 1996

[36] Reference (C++) – Relationship to pointers (website)
http://en.wikipedia.org/w/index.php?title=Reference_(C%2B%2B)&oldid=273921693#Relationship_to_pointers
Last updated: 28 Feb 2009

[37] Alex, LearnCpp.com
Eight C++ programming mistakes the compiler won't catch (website)
http://www.learncpp.com/cpp-programming/eight-c-programming-mistakes-the-compiler-wont-catch/
Posted: 2 July 2007

[38] Microsoft Corporation, © 2009
Flow Control (C# vs. Java) (website)
http://msdn.microsoft.com/en-us/library/ms228393.aspx
Last accessed: 2 Mars 2009

[39] Microsoft Corporation, © 2009
Params (C# Reference) (website)
http://msdn.microsoft.com/en-us/library/w5zay9db.aspx
Last accessed: 2 Mars 2009

[40] Sun Microsystems, © 2004
Varargs (website)
http://java.sun.com/j2se/1.5.0/docs/guide/language/varargs.html
Last accessed: 2 Mars 2009

[41] Wes Haggard
Variable number of parameters... C/C++ vs C# - Wes' Puzzling Blog (blog post)
http://weblogs.asp.net/whaggard/archive/2004/07/03/172616.aspx
Last updated: 3 July 2004

[42] MSDN Forums – why is "Main" in C# not a "public" static (forum post)
http://social.msdn.microsoft.com/Forums/en-US/csharpgeneral/thread/9184c55b-4629-4fbf-ad77-2e96eadc4d62
Last accessed: 2 Mars 2009

[43] Main function (programming) (website)
http://en.wikipedia.org/w/index.php?title=Main_function_(programming)&oldid=273951281
Last updated: 28 Feb 2009

[44] IBM Corporation, © 1994, 2009
Inline member functions (C++ only) (website)
http://publib.boulder.ibm.com/infocenter/lnxpcomp/v8v101/index.jsp?topic=/com.ibm.xlcpp8l.doc/language/ref/inline_member.htm
Last accessed: 2 Mars 2009

[45] The PHP Group, © 2001-2009
PHP Manual – Constructors and Destructors (website)

http://www.php.net/manual/en/language.oop5.decon.php
Last updated: 27 Feb 2009

[46] C++ structures and classes – Declaration and usage (website)
http://en.wikipedia.org/w/index.php?title=C%2B%2B_structures_and_classes&oldid=272658002#Declaration_and_usage
Last updated: 23 Feb 2009

[47] Tony Sintes, Network World, Inc., © 2006-2008
JavaWorld – Why not multiple inheritance? (article)
http://www.javaworld.com/javaqa/2002-07/02-qa-0719-multinheritance.html
Published: 19 July 2002 in JavaWorld

[48] 1) Bill Venners, Network World, Inc., © 2006-2008
Designing with Interfaces (article)

http://www.javaworld.com/javaworld/jw-12-1998/jw-12-techniques.html
Published: Dec 1998 in JavaWorld

2) Tony Sintes, Network World, Inc., © 2006-2008
Java Diamonds Are Forever (article)
http://www.javaworld.com/javaworld/javaqa/2001-03/02-qa-0323-diamond.html
Published: 23 March 2001 in JavaWorld

3) Eddy Truyen, Wouter Joosen, Bo Nørregaard Jørgensen, and Pierre Verbaeten,
Research Institute for Computer science (RIACS)
A Generalization and Solution to the Common Ancestor Dilemma Problem in Delegation-Based Object Systems (Tech. Report)
Pages 103-119 of the report labeled: Proceedings of the 2004 Dynamic Aspects Workshop
http://aosd.net/2004/workshops/daw/Proc-2004-Dynamic-Aspects.pdf
Published: March 2004, Lancaster, England

[49] Alex, LearnCpp.com
Inheritance and access specifiers (website)
http://www.learncpp.com/cpp-tutorial/115-inheritance-and-access-specifiers/
Posted: 14 Jan 2008

[50] Alex, LearnCpp.com
Constructors (Part II) (website)
http://www.learncpp.com/cpp-tutorial/88-constructors-part-ii/
Posted: 7 Sep 2007

[51] The PHP Group, © 2001-2009
PHP Manual – Namespaces overview (website)
http://www.php.net/manual/en/language.namespaces.rationale.php
Last updated: 27 Feb 2009

[52] Microsoft Corporation, © 2009
How to: Use the Namespace Alias Qualifier (C# Programming Guide) (website)
http://msdn.microsoft.com/en-us/library/c3ay4x3d.aspx
Last accessed: 2 Mars 2009

[53] Microsoft Corporation, © 2009
Preprocessor Directives (website)
http://msdn.microsoft.com/en-us/library/3sxhs2ty.aspx
Last accessed: 2 Mars 2009

[54] Microsoft Corporation, © 2009

C# Preprocessor Directives (website)
http://msdn.microsoft.com/en-us/library/ed8yd1ha.aspx
Last accessed: 2 Mars 2009

[55] Jupitermedia Corporation, © 2009
List of Reserved Words (website)
http://www.phpbuilder.com/manual/en/reserved.php
Last updated: 26 June 2008

[56] PHP (website)
http://en.wikipedia.org/w/index.php?title=PHP&oldid=274191239#Speed_optimization
Last updated: 1 March 2009

[57] Herb Sutter, © 2009
A Pragmatic Look at Exception Specifications (website)
http://www.gotw.ca/publications/mill22.htm
Last accessed: 2 Mars 2009

[58] Sun Microsystems, © 2004
Enums (website)
http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html
Last accessed: 2 Mars 2009

[59] C++ structures and classes – Differences between struct in C and classes in C++ (website)
http://en.wikipedia.org/w/index.php?title=C%2B%2B_structures_and_classes&oldid=272658002#Differences_between_struct_in_C_and_classes_in_C.2B.2B
Last updated: 23 Feb 2009

[60] Microsoft Corporation, © 2009
Operator Overloading (website)
http://msdn.microsoft.com/en-us/library/5tk49fh2.aspx
Last accessed: 2 Mars 2009

[61] Microsoft Corporation, © 2009
Operator Overloading (C# vs Java) (website)
http://msdn.microsoft.com/en-us/library/ms228498.aspx
Last accessed: 2 Mars 2009

[62] Microsoft Corporation, © 2009
Properties (C# Programming Guide) (website)
http://msdn.microsoft.com/en-us/library/x9fsa0sw(VS.80).aspx
Last accessed: 2 Mars 2009

[63] Microsoft Corporation, © 2009
Generics (C# Programming Guide) (website)
http://msdn.microsoft.com/en-us/library/512aeb7t(VS.80).aspx
Last accessed: 2 Mars 2009

[64] Gilad Bracha, Sun Microsystems, Inc.
Generics in the Java Programming Language – Introduction (PDF, pp. 2)
http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf
Published: 5 July 2004

[65] Juan Soulie, cplusplus.com, © 2000-2009
Templates (website)
http://www.cplusplus.com/doc/tutorial/templates.html
Last updated: 16 Nov 2007

[66] Comparison of Java and C++ – Templates vs. Generics (website)
http://en.wikipedia.org/w/index.php?title=Comparison_of_Java_and_C%2B%2B&oldid=273495583#Templates_vs._Generics
Last updated: 26 Feb 2009

[67] Mikael Olsson
Multipad (freeware)
http://sourceforge.net/projects/multipad
Last accessed: 18 March 2009

[68] Ziff Davis Enterprise Holdings Inc., © 1999-2009
Visual Studio most widely used IDE, survey says (article)
http://www.windowsfordevices.com/news/NS2484248296.html
Published: 14 June 2006

[69] IC#Code, © 2000-2009
SharpDevelop (freeware)
http://www.icsharpcode.net/OpenSource/SD/Default.aspx
Last accessed: 18 March 2009

[70] The Eclipse Foundation
Eclipse IDE (freeware)
http://www.eclipse.org/
Last accessed: 18 March 2009

[71] JetBrains, © 2000-2009
IntelliJ IDEA (software)
http://www.jetbrains.com/idea/
Last accessed: 18 March 2009

TRITA-ICT-EX-2009:8

www.kth.se