Exam
DIT948: Programming

Date:	2016-12-22
Time	14:00–18:00
Place	Lindholmen
Teacher	Musard Balliu
Examiner	Ivica Crnkovic
Questions	0729744942 first visit around 15:00
Results	will be posted within 15 working days
Grades	Pass (G) 50p, Pass with Honors (VG) 80p
Allowed aids	Any book on Java programming is allowed Any English dictionary No electronic translators are allowed
Reviewing:	2017-01-13 13:00–15:00, Lindholmen Second floor room in Patricia (Note: subject to change, changes will be posted on the course website).

Please observe carefully the following:

- Write in legible English (illegible translates to "no points"!).
- Motivate your answers, and clearly state any assumptions made.
- Code or text copied from the book will not be rewarded. Your own contribution is required.
- Start each task on a new sheet!
- Write on only one side of the paper!
- Before handing in your exam, number and sort the sheets in task order!
- Write your anonymous code and page number on every page!

Not following these instructions will result in the deduction of points!

1. [10pts] Consider the following Java program:

```
public class Game {
  private static int board = 10;
  private int score = 0;

  public Game(int score){
    this.score=score;
  }

  public void addBoard(){
    Game.board+=this.score;
  }

  public int readBoard(){
    return Game.board;
  }

  public static void messBoard(Game g1, Game g2){
    g1.addBoard();
    System.out.println("Board score: " + Game.board);
    g1=g2;
    System.out.println("User score: " + g1.score);
  }

  public static void main(String[] args) {
    Game p1 = new Game(10);
    System.out.println("Board score: " + p1.readBoard());
    p1.addBoard();
    System.out.println("Board score: " + p1.readBoard());
    Game p2 = new Game(20);
    p2.addBoard();
    System.out.println("Board score: " +p2.readBoard());
    messBoard(p1, p2);
    System.out.println("User1 score: " +p1.score);
    System.out.println("User2 score: " +p2.score);
    System.out.println("Board score: " + Game.board);
    p2=p1;
    messBoard(p1, p2);
  }
}
```

What will be printed to the console when running the program Game?

2. [10pts]

   Your boss wrote a program that was supposed to compute the "weighted sum" of an array of integers `arr` and an array of reals (of type double) `coeff`. The method should be static and return 0 if the two arrays have different lengths or are empty. Otherwise, the method should return the weighted sum, which is defined as the sum of the products (multiplications) of corresponding elements (namely elements at the same array positions), of the two input arrays. For instance, if `arr=[1, 2, 3]` and `coeff=[0.1, 0.2, 0.3]` then the weighted sum is calculated as $1 * 0.1 + 2 * 0.2 + 3 * 0.3$ and thus the returned result is 1.4.

   Unfortunately, things didn't really work out as expected: in fact, the program doesn't even compile! Still, it's your boss, you can't just throw it away and write a new program. Make the changes necessary for the program to compile and return the correct result.

```
public static void weightedSum(int[] arr, double[] coeff){
  double sum;
  if (arr.length != coeff.length || arr.length == 0 ){
    return 0;
  }
  for(int i=0; i < arr.length; i++){
    sum=arr[i] * coeff[i+1];
  }
}
return sum;
```

3. [10pts] Consider the following classes:

```
public class C1 {
  public C1(){
    System.out.println("Calling constructor of C1");
  }
}

public class C2 extends C1 {
  public C2(){
    this("Calling constructor of C2");
```

```
    System.out.println("Default constructor of C2");
  }
  public C2(String s){
    System.out.println(s);
  }
}
```

Which of the lines 3–5 in class `Test`, if any, will cause a compilation error and why? Comment out such lines (if any) and write what will be printed out to the console when running the class `Test`.

```
1 public class Test {
2   public static void main(String[] args){
3      C2 c2 = new C1();
4      C1 c1 = new C1();
5      C1 c = new C2();
6   }
7 }
```

4. [30pts]

   For many institutions, including universities, it becomes difficult to schedule meetings during the month of January as several employees might be on vacation. The ultimate goal of this exercise is to implement Java programs that help the user perform simple queries about university employees and their vacation days.

   You will implement three classes to represent university employees, their vacation information, and meeting schedules during the month of January. The classes should provide different variables and methods to perform operations like printing the list of employees and their vacation dates, computing the number of employees at work for a given date, and so on.

   Concretely, you will implement the following classes:

   An `Employee` is specified by the name and the position. For instance, *Professor John Smith* can be represented by an object of class `Employee`, where name="John Smith" and position="Professor".

   A `Vacation` extends an `Employee` by (additionally) specifying the initial vacation day *from* and the final vacation day *until*. You can assume that the days refer to the month of January, namely they range from

1 to 31. You are NOT required to do any error checking on vacation days or distinguish between week days and weekends.

A `Schedule` is specified by an array of `Vacation` objects, containing the employees' data and their vacation days. You can assume that a given employee occurs at most once in the array.

You don't need to do any error checking or write comments.

Please do not write "between the lines"! On your paper, clearly mark out which code is supposed to go where. For example, in the code supposed to fill in the block `A`, the instance variable of the class `Employee`, should appear as

```
+----------A-----------+
|  private String name;|
+----------------------+
```

and not squeezed on the page with the exam subjects!

Remarks:

- Since block A has been filled in for you above, there are twelve blocks left for you to fill in for this exercise. Blocks B, C and M are worth one point each, blocks D, E, F and G are worth two points each, blocks H, I, J, K and L are worth 3 points each, and block N is worth 4 points.

End of remarks.

```
/**
 * This is a class representing basic data of an employee.
 * An Employee consists of a name and a position.
 */
public class Employee {
  // private instance variable declaration

  // Block A
  // name, a String representing name of the employee

  //Block B
  // position, a String reprerenting the position of the employee
```

```
  /**
  * Constructor with parameters
  * It initializes this object with formal parameters below
  * @param name: the String representing the name
  * @param position: the String representing the position
  */
  public Employee(String name, String position){
  // Block C
  }

  /**
  * Constructor with parameters
  * It initializes this object with formal parameter below
  * @param e, an Employee object
  */
  public Employee(Employee e){
  // Block D
  }

  /**
  * Checks if this Employee is the same as Employee e, that is,
  * whether or not the name and the position are the same.
  * @param e: an Employee object
  * @return true or false
  */
  public boolean equals(Employee e){
  // Block E
  }

  /**
  * Returns the string representation  of this object.
  * Example: If name is "John Smith" and position is "Professor",
  * it returns the string "Professor John Smith"
  */
  public String toString(){
  // Block F
  }
}
```

```
/**
 * This is a subclass of Employee, extending an employee with
 * vacation days for the month of January
 */
public class Vacation extends Employee {
  // private instance variable declaration
  // from, an integer representing the first vacation day
  // until, an integer representing the last vacation day
  // Block G

  /**
  * Constructor with parameters
  * It initializes this object with formal parameters below
  * @param e:  an Employee object
  * @param from: first vacation day
  * @param until: last vacation day
  */
  public Vacation(Employee e, int from, int until){
  // Block H
  }

  /**
  * Default constructor of the class
  * It initializes: name with "Bob Brown", position with "Doctor",
  * from with 2 and until with 11
  */
  public Vacation(){
  // Block I
  }

  /**
  * Calculates the number of vacation days for this object
  * @return the number of vacation days
  */
  public int getVacationDays(){
  // Block J
  }

  /**
  * Checks if this employee is on vacation on a given day
  * @param day: the day
```

```
  * @return true or false
  */
  public boolean isOnVacation(int day){
  // Block K
  }

  /**
  * Returns the string representation  of this object.
  * Example: If name="Bob Brown", position="Doctor",
  * from=2 and until=11 it returns the String
  * "Doctor Bob Brown is on vacation from 2 until 11 of January"
  */
  public String toString(){
  // Block L
  }
}


/**
 * This class implements different methods related to employees
 *  and their vacations at a given university
 */
public class Schedule {
  // private array of Vacation objects
  private Vacation[] vacationList;

  /**
  * Constructor with parameters
  * It initializes this object with formal parameter below
  * @param vacationList:  an array of Vacation objects
  */
  public Schedule(Vacation[] vacationList){
  // Block M
  }

  /**
  * Returns the string representation of the array of Vacation objects.
  * Example: If the array has 2 elements, the string should look like:
  * "Doctor Bob Brown is on vacation from 2 until 11 of January"
  * "Professor John Smith is on vacation from 12 until 21 of January"
  */
```

```
  public String toString(){
  // Block N
  }
}
```

5. [20pts] In this exercise, we refer to the classes from above, but do not require you to have implemented them (only that you understand how to create and use instances of them). Note that you are only allowed to use methods from the classes in Exercise 3, and not implement your own methods.

   (a) In class `Schedule`, implement a method with the signature

   ```
   /**
   * Returns the number of employees that are working on a given day.
   * Assume that the array contains at least one element
   * @return the number of employees working that day
   */
   public int numEmployeesAtWork(int day)
   ```

   that returns number of (how many) employees that are working on a given day. Namely, it returns the number of employees that are not on vacation on that day.

   (b) In class `Schedule`, implement a method with the signature

   ```
   /**
   * Returns the first day where the maximum number of employees
   * can participate to a meeting. Namely, the day of month January
   * where the maximum number of employees are at work (not on vacation)
   * @return the best day for scheduling a meeting
   */
   public int bestMeetingDay()
   ```

   that best day for scheduling a meeting (namely the first day where the maximum number of employees are at work).

   Each method is worth 10 points.

6. [20pts] (From *CodingBat*)

   Write a subroutine/method that returns the "centered" average of an array of integers. The centered average is the mean average of the array

values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use integer division to produce the final average. You may assume that the array is length 3 or more. Recall that the mean average of an array of values is the sum of the array values divided by the number of such values in the array (namely, the length of the array).

The method should have the following signature:

```
public int centeredAverage(int[] nums)
```

Examples:

```
centeredAverage([1, 2, 3, 4, 100]) -> 3
centeredAverage([1, 1, 5, 5, 10, 8, 7]) -> 5
centeredAverage([-10, -4, -2, -4, -2, 0]) -> -3
centeredAverage([5, 3, 4, 6, 2]) -> 4
centeredAverage([5, 3, 4, 0, 100]) -> 4
centeredAverage([100, 0, 5, 3, 4]) -> 4
```