Performance impact of using polymorphism

Malin Källén

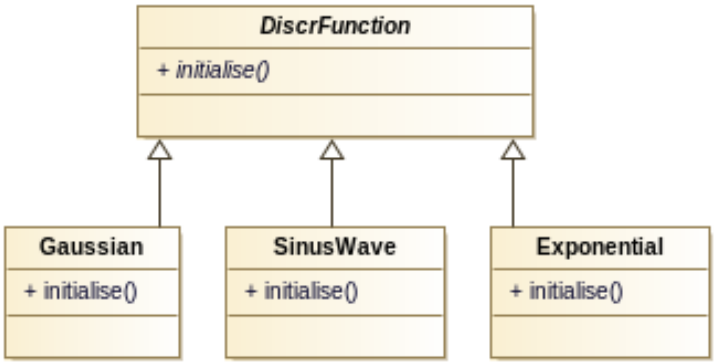
Division of Scientific Computing
Department of Information Technology
Uppsala University

May 28 2014

Object orientation

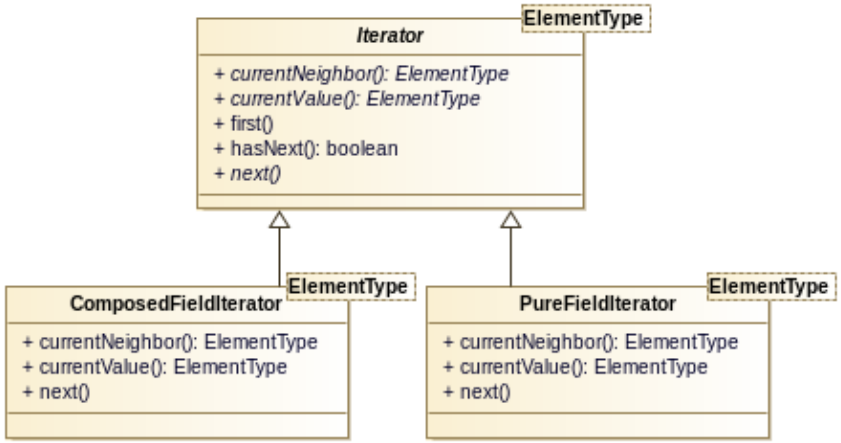
- Encapsulation
  - "Mind your own business"
- Inheritance
  - Extension and reuse of more general code
- Polymorphism
  - Provide several implementations for one interface

Example of polymorphism:




```
void setUp(DiscrFunction f) {
    ...
    f.initialise();
    ...
}
```

Polymorphism and dynamic binding




```
Iterator *iterator;
...
for each element:
    ...
    iterator->currentValue();
    ...
    for each neighbor:
        ...
        iterator->currentNeighbor();
    ...
```

Lots of calls to dynamically bound methods hamper compiler optimisations.

Main reason in this case: Impossibility to inline dynamically bound methods.

Static polymorphism

Keep the class hierarchy, but provide compile-time information about the concrete type. (Curiously Recurring Template Pattern, J. Coplien 1995)

- Considerably better performance
- Much (but not all!) of the flexibility brought by the dynamic binding is lost
- Harder to understand how to use and extend the code
- Some other technical limitations

Is it a reasonable compromise?