Winter Semester 17/18

Software Engineering Design & Construction

Dr. Michael Eichberg
Fachgebiet Softwaretechnik
Technische Universität Darmstadt

Template Method Pattern

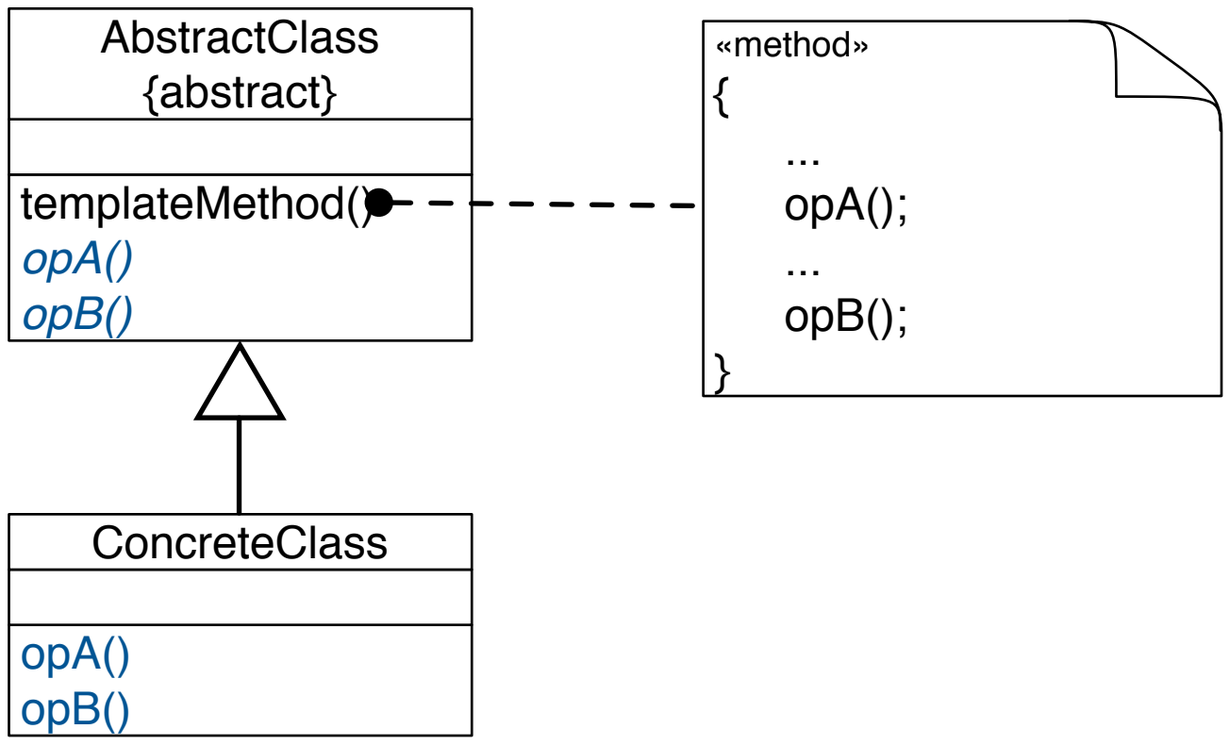
The Template-Method Pattern in a Nutshell

Intent:

- Separate high-level policies from detailed low-level mechanisms.
- Separate invariant from variant parts.

Solution Idea:
Use abstract classes to:

- Define interfaces to detailed mechanisms and variant parts.
- Implement high-level policies and invariant parts to these interfaces.
- Control sub-class extensions.
- Avoid code duplication.



Example Application of Template Method

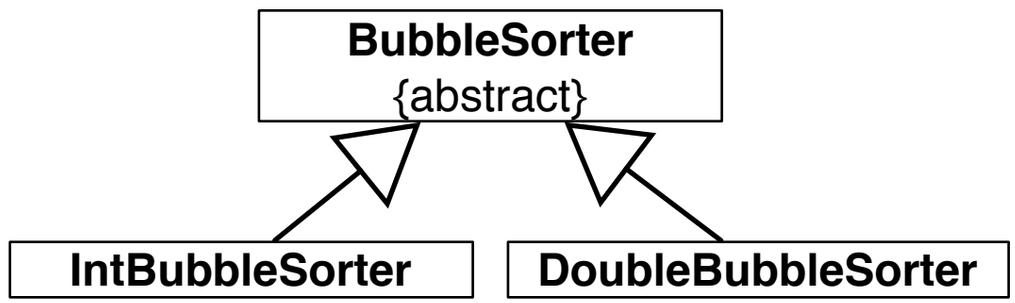
Functional requirements:

- Need a family of sorting algorithms …(bubble sort, quick sort, etc.)
- for different kinds of data (int, double, etc.)
- Clients that use sorting algorithms should be reusable with a variety of specific algorithms.

Non-functional requirements on the design:

- Need to separate the high-level „sorting“ policies from low-level mechanisms.
- Low-level mechanisms are responsible for:
  - deciding when an element is out of order,
  - swapping out-of-order elements.

Separating the Policy of Sorting




```
public abstract class BubbleSorter {
  protected int length = 0;
```

Policy

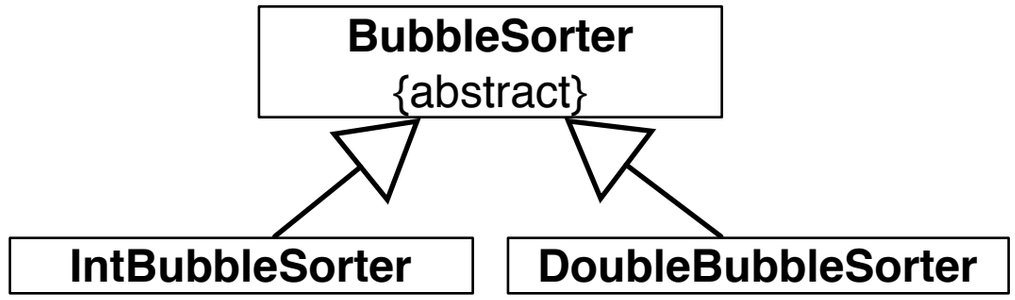
```
 protected void sort() {
    if (length <= 1) return;
      for (int nextToLast = length - 2; nextToLast >= 0; nextToLast--)
        for (int index = 0; index <= nextToLast; index++)
          if (outOfOrder(index)) swap(index);
  }
```

Mechanism

```
  protected abstract void swap(int index);
  protected abstract boolean outOfOrder(int index);

}
```

Filling the Template for Specific Algorithms

```
public class IntBubbleSorter
  extends BubbleSorter {

  private int[] array = null;

  public void sort(int[] theArray) {
    array = theArray;
    length = array.length;
    /*"super"*/sort();
  }
  protected void swap(int index) {
    int temp = array[index];
    array[index] = array[index + 1];
    array[index + 1] = temp;
  }
  protected boolean outOfOrder(int index) {
    return (array[index] > array[index + 1]);
  }
}
```



What are the advantages and deficiencies of the Template-Method Pattern?

Task: Identify the *Template Method Pattern* in *Log4J*

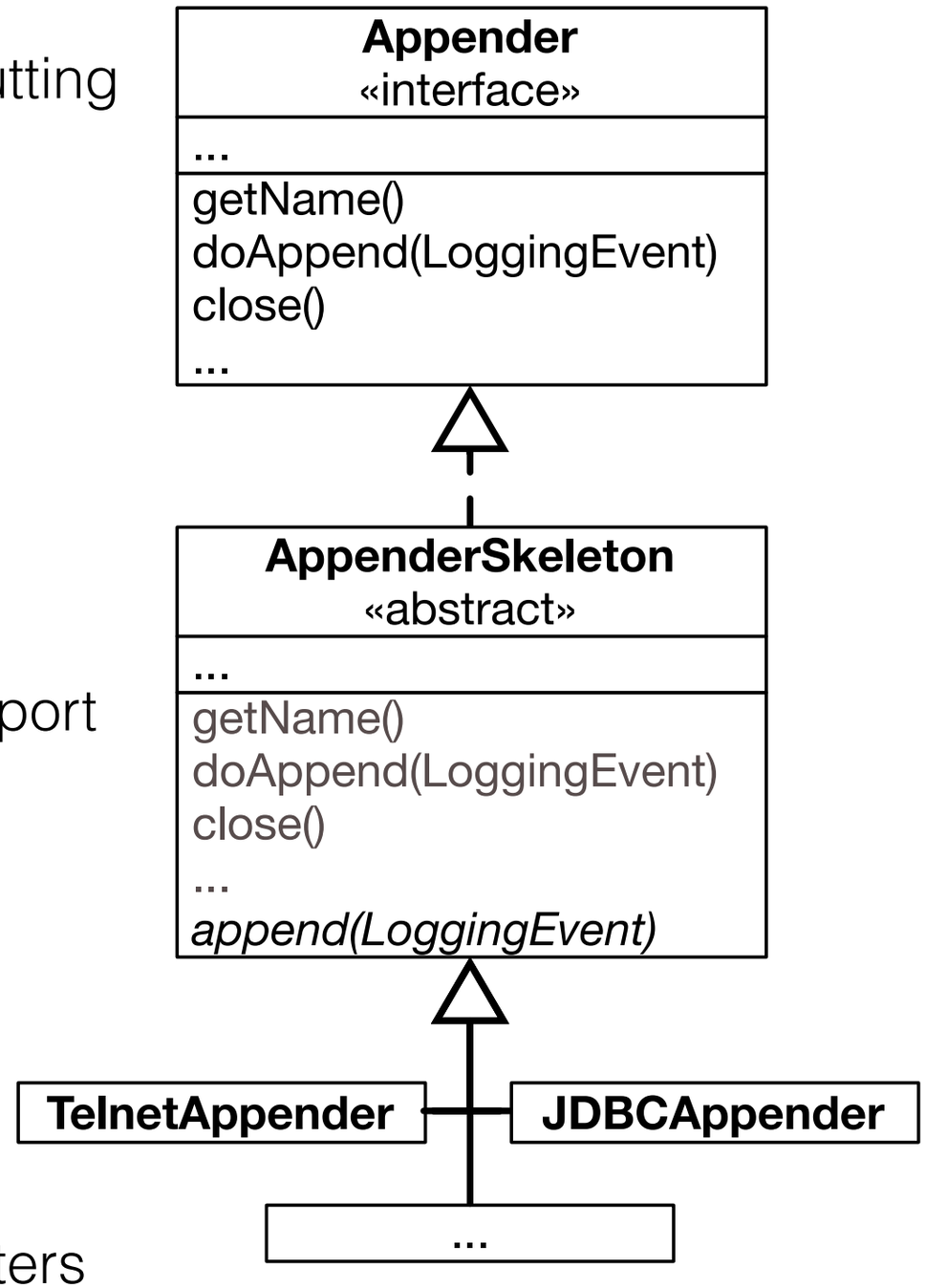
`interface Appender`
Implement this interface for your own strategies for outputting log statements. […]

`public void doAppend(LoggingEvent event)`
Log in Appender specific way.

`abstract class AppenderSkeleton implements Appender`
Abstract superclass of the other appenders. This class provides the code for common functionality, such as support for threshold filtering and support for general filters. […]

`abstract void append(LoggingEvent event)`
Subclasses should implement this method to perform actual logging.

`void doAppend(LoggingEvent event)`
This method performs threshold checks and invokes filters before delegating actual logging to the append(LoggingEvent) method.



Functional Counterpart of Template

One can look at the Template-Method Pattern as a style for emulating higher-order functions available in programming languages that support functional-style programming.

Alternative design for Log4J in Scala?

```
class AppenderSkeleton(
  private val append : (LoggingEvent) => Unit // Function1[LoggingEvent,Unit]
) {
  def doAppend(loggingEvent : LoggingEvent) {
    // filtering, threshold checks,…
    append(loggingEvent)
  }
}
```

Scala (2.11.x) HashTable

```
/**
[…]
There are mainly two parameters that affect the performance of a hashtable: the initial size
and the load factor. The size refers to the number of buckets in the hashtable, and the load
factor is a measure of how full the hashtable is allowed to get before its size is
automatically doubled. Both parameters may be changed by overriding the corresponding
values in class HashTable.
*/
trait HashTable[A, Entry >: Null <: HashEntry[A, Entry]] extends HashTable.HashUtils[A] {
  […]
  /** The actual hash table.*/
  @transient protected var table: Array[HashEntry[A, Entry]] = new Array(initialCapacity)

  /** The initial size of the hash table.*/
  protected def initialSize: Int = 16

  private def initialCapacity = capacity(initialSize)


  […]

}
```