

Advanced Aspects of Object-Oriented Programming (SS 2009)

Practice Sheet 5

Date of Issue: 19.05.09
Deadline: 25.05.09
(until 10 a.m. as PDF via E-Mail)

Exercise 1 Abstract Factory Pattern

Inform yourself about the Abstract Factory Pattern.

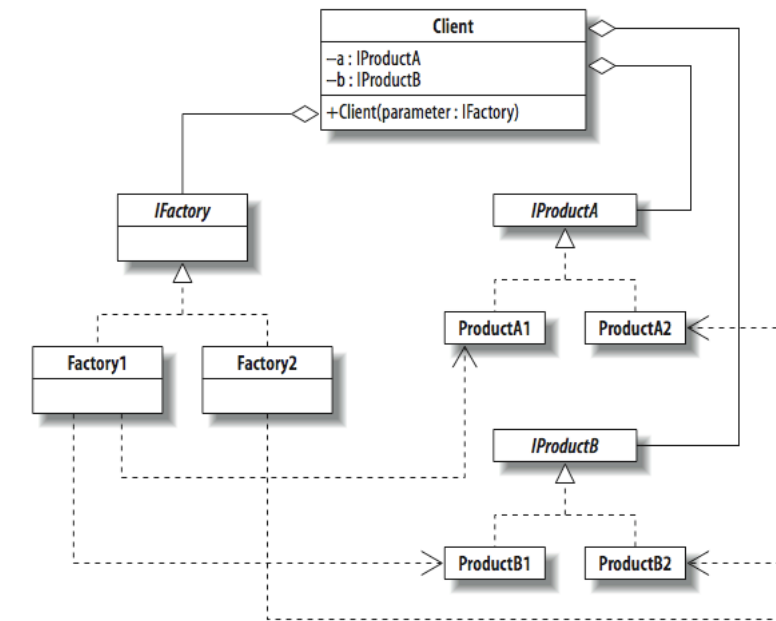


Figure 1: Abstract Factory Pattern

- What is the Abstract Factory Pattern good for?
- Implement the following scenario using the pattern: *In today's society, the products of many famous brands are copied and presented for sale, often as the real thing. Sometimes, the customer do not care from which factory the goods are actually originating. As an example, consider a family of Gucci handbags and their corresponding look-a-likes.*

Exercise 2 Generics

- Write a generic method `flip` which takes an object of class `Pair` and flips the elements of the given `Pair` object. *Hint: In order to flip the elements, both need to be of same type.*
- Write a method `equals` which only allows to compare two objects using the `compareTo` Method of the `Comparable` interface.
- Implement a generic method `toArray` having the following signature:

```
public static <T> T[] toArray(List<T> l) { ... }
```
- What is the difference between a `Collection<?>` and a `Collection<Object>` ?
- Explain the output of the following program:

```

public final class GenericClass <T> {
    private void overloadedMethod(Collection <?> o) {
        System.out.println("overloadedMethod_(Collection <?>");
    }
    private void overloadedMethod(List <Number> s) {
        System.out.println("overloadedMethod_(List <Number>");
    }
    private void overloadedMethod(ArrayList <Integer> i) {
        System.out.println("overloadedMethod_(ArrayList <Integer >");
    }

    private void method(List <T> t) {
        overloadedMethod(t); // which method is called?
    }

    public static void main(String [] args) {
        GenericClass <Integer> test = new GenericClass <Integer >();
        test.method(new ArrayList <Integer >());
    }
}

```

Exercise 3 Type Erasure

```

public class ComparableTest <A,B> implements Comparable <A>,
    Comparable <B> {
    public int compareTo(A a) { return -1; }
    public int compareTo(B b) { return 1; }

    public static void main(String ... args) {
        ComparableTest <String , Integer > C=new ComparableTest <String , Integer >();
    }
}

```

- a) Transform the given source code using the type erasure algorithm utilised by Java 5.
- b) Explain why proper compilation of the source code is impossible.