Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. Kathrin Geilmann

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2010)

Practice Sheet 6

Date of Issue: 14.05.10
Deadline: 19.05.10
(until 10 a.m. as PDF via E-Mail)

Exercise 1 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.

Exercise 2 Generics

a) Java does not support the creation of generic arrays, i.e. `new List<E>[]`, `new List<String>[]`, `new E[]`, where E is a declared type variable, are illegal statements. In order to find out the reasons for it, assume that the statement `List<String>[] stringLists = new List<String>[1];` is legal.

Write a code snippet that leads to a ClassCastException at a cast, that has been inserted by the type erasure. Your code should be legal Java except for the given statement. *Hint: Take advantage of the covariance of arrays and try to assign a list of a different type to* `stringLists[0]`.

b) Implement a generic static method `flatten` having the following signature

```
public static <...> List<...> flatten (List<...> l)
```

The method takes a list of lists of ... and flattens it by one level. Replace each ellipsis so your implementation provides maximum reusability without violating type correctness.

Exercise 3 Virtual Types

Look at the implementation of Subject and Observer and their specialization to WindowObserver and WindowSubject (slide 3.34 of the lecture)

a) Is the following code type correct? Justify your answer.

```
WindowSubject s1 = new WindowSubject();
s1.notifyObservers(new Object());
s1.notifyObservers(new WindowEvent());
Subject s2 = new Subject();
s2.notifyObservers(new Object());
s2.notifyObservers(new WindowEvent());
```

b) Transform the implementation of slide 3.34 and the source from exercise 3.a to standard Java using the transformation given in the lecture.

c) The given implementation ensures that WindowSubject and WindowObserver are used together, i.e. not allowing WindowObserverS to notify subjects that are not a subtype of WindowSubject or vice versa.

Write an implementation for these classes and interfaces, that takes advantage of generics and tries to achieve the same guarantees as the virtual types based one. If a property cannot be ensured statically, introduce runtime checks and throw appropriate exceptions.