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 2011)

Practice Sheet 3

Date of Issue: 26.04.11
Deadline: 03.05.11
(until 10 a.m. as PDF via E-Mail)

Exercise 1 Introspection and Reflection

With the techniques of introspection and reflection, it is possible to store objects or whole object graphs in a file.

a) Implement the class `PersistencyMangager` that supports writing and reading of arbitrary objects to and from files. Your class should implement the following Interface

```
interface PersistencyManager {
   public void writeObject(Object o, String toFileName);
   public Object readObject(String fromFileName);
}
```

`writeObject` writes the state of the object `o`, the states of the objects in the object graph reachable by `o` and the reference structure to a newly created file with the given filename. `readObject` reads an object graph from the file and reconstructs it, the return value is the object, which acted as entry in the graph during the storage.

Hint: Give each object a persistent identifier and use the identifier for storing references.

b) Look at the following code snippet:

```
IPersistencyManager pm;
Object o, o2;
...
pm.writeObject(o, ''File'');
o2 = pm.readObject(''File'');
```

Is the object graph referenced to by `o` equal to the one referenced by `o2`? If not, what is the difference?

c) Look into the Java-Documentation, which possibilities are offered for storing and retrieving objects? What are the advantages and disadvantages of the different solutions? Compare them with your solution.

Exercise 2 Required and Provided Interfaces

Download the source for the class `ObjectOutputStream` from the website of the lecture.

a) Which different *provided interfaces* does the class have? Give examples for each of them.

b) What is the *required interface* of an `ObjectOutputStream`-object?

Exercise 3 Structural vs. Nominal Typing

Type systems for programming languages can be based on the structure or on the names of types. In nominal type systems types are considered to be equal if they have the same name, whereas in structural type systems types are equal if the have the same structure. In a structural type systems usually a type T is a subtype of S if it offers at least all “features” of S, where features are e.g. attributes and methods.

The following assignment would be legal in structural typed languages, because B has all the attributes A has plus an additional one. It is of course but illegal in nominal type systems.

```
class A {
    int x;
    int y;

};
```

```
class B {
     int x;
     int y;
       int z;
};
```

```
A a  = new B();
```

Most widely used languages use a nominal type system. An example for a structural typed OO-programming language is Go (see `http://golang.org`). C++, Scala and others use a hybrid type system.

a) What is the advantage of structural typed languages?

b) The University Administration System is now implemented in a language with a structural type system and it is extended with two new modules. One for handling the exams and one for the handling of employes.

```
class Person {
    String name;
    void print(){...}
}
```

```
class Student {
    String name;
    int reg_num;
    void print() {...}
}
```

```
class Professor {
    string name;
    string room;
    void print(){...}
}
```

```
class Employee {
    string name;
    int reg_num;
    int salary;
    void print() {...}
}
```

```
class ....
    public void registerForExam(Student s) {...}
```

Which kind of errors can occur now?