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 2

Date of Issue: 15.04.10
Deadline: 21.04.10
(until 10 a.m. as PDF via E-Mail)

Exercise 1 Happy Birthday

In a software project, the classes `Person`, `AgePerson` und `AgeManager` were implemented; their source is given in Figure 1.

a) For testing purposes, the following code is used.

```
...
AgeManager ageManager=new AgeManager();
AgePerson  agePerson =new AgePerson("Jane","Doe",28);

ageManager.add("02.02.1980",agePerson);

while(true) {
  System.out.println("Is␣"+agePerson+"␣managed:␣"+ageManager.isManaged(agePerson));

  Thread.sleep(5000);
}
...
```

Explain the following output:

```
Is Jane Doe managed: true
Jane Doe: Age updated.
Is Jane Doe managed: false
Jane Doe: Age updated.
Is Jane Doe managed: false
Jane Doe: Age updated.
Is Jane Doe managed: false
...
```

Hint: Look into the documentation for `Object.hashCode` *and* `Map`

b) Give a general programming guideline that helps to avoid theses problems.

c) Explain why the parameter *birthday* in `AgeManager.add` has to be *final*.

Exercise 2 Source Compatibility of Java-Packages

The packages of a software implementation usually evolves is different ways. A package may be exchanged by another one, classes, fields and methods can be added or removed, and so on. For the developer of the package it is important to know, wether his changes are source compatible or not. A change breaks source compatibility, if a program exists that compiles with the unchange package version but does not compile with the changed one.

Look at the following packages. Is it safe to make the given modifications? If not, give counter examples.

```
// unmodified version
package util;

public interface List {
  public abstract List n() ;
}

// modified version
package util;

public interface List {
  public abstract List n() ;
  public abstract List m() ;
}
```

```
// unmodified version
package p;

public final class C {
  protected C m(Object v) {}
}}

// modified version
package p;

public class C {
  public C m(C v) {}
}}
```

```
public class Person {
  protected String firstname;
  protected String lastname;

  public Person(String firstname,
                String lastname) {
    this.firstname=firstname;
    this.lastname =lastname;
  }

  public boolean equals(Object second) {
    if (second.getClass()==getClass()) {
      Person person=(Person)second;

      return(person.firstname.equals(firstname) &&
             person.lastname.equals(lastname));
    }

    return(false);
  }

  public int hashCode() {
    return(firstname.hashCode() ^
           lastname.hashCode());
  }

  public String toString() {
    return(firstname+" "+lastname);
  }
}

public class AgePerson extends Person {
  private int age;

  public AgePerson(String firstname,
                   String lastname,
                   int age) {
    super(firstname,lastname);

    this.age=age;
  }

  public void updateAge() {
    age++;

    System.out.println(this+": Age updated.");
  }

  public boolean equals(Object second) {
    if (second.getClass()==getClass()) {
      AgePerson agePerson=(AgePerson)second;

      return(agePerson.firstname.equals(firstname) &&
             agePerson.lastname.equals(lastname) &&
             agePerson.age==age);
    }

    return(false);
  }

  public int hashCode() {
    return(super.hashCode()^age);
  }
}
```

```
import java.util.*;
import java.util.Map.*;

import java.text.*;

public class AgeManager {
  private Map<AgePerson,Calendar>
    persons=new HashMap<AgePerson,Calendar>();

  private Calendar
    currentDate=new GregorianCalendar();

  public AgeManager() {
    new Timer().schedule(new TimerTask() {
      public void run() {
        currentDate.add(Calendar.DAY_OF_MONTH,1);

        updateAges(currentDate);
      }
    },new Date(),10);
  }

  protected synchronized void updateAges(Calendar date) {
    int month=date.get(Calendar.MONTH);
    int day  =date.get(Calendar.DAY_OF_MONTH);

    for(Entry<AgePerson,Calendar> entry : persons.entrySet())
      if ((entry.getValue().get(Calendar.MONTH)==month) &&
          (entry.getValue().get(Calendar.DAY_OF_MONTH)==day))
        entry.getKey().updateAge();
  }

  public synchronized void add(final String birthday,
                               AgePerson person)
                               throws ParseException {
    persons.put(person,new GregorianCalendar() {{
       setTime(DateFormat.getDateInstance()
                         .parse(birthday));
    }});
  }

  public synchronized boolean isManaged(AgePerson person) {
    return(persons.containsKey(person));
  }
}
```

Figure 1: Sources for the classes `Person`, `AgePerson` and `AgeManager`.