Institut für Informatik
Praktische Informatik und Bioinformatik
Prof. Dr. Ralf Zimmer

Wintersemester 2016/17

Übungen zum Bioinformatik-Tutorium

Blatt 10

Termin: Dienstag, 08.01.2018, 11 Uhr

1. Vererbung - Person, Student, Professor

Implementiere die Klasse Person. Sie hat die `private` Felder `name` und `age`, die über getter und setter zugänglich sein sollen. Überschreibe die `String toString()` Methode so dass die Informationen zu der Person ausgegeben werden sowie die Methode `boolean equals(Person p)` die überprüft ob die beiden Personen anhand Name und Alter identisch sind.

Implementiere die Klasse `Student`, die von `Person` erbt. Ein Student hat zusätzlich noch eine Matrikelnummer anhand der überprüft werden kann ob zwei Personen gleich sind. Überschreibe also die `equals` Methode von `Student`. Implementiere die Klasse `Professor` die genauso wie `Student` aufgebaut ist aber statt einer Matrikelnummer ein Set von Strings hat, in dem die Namen der aktuell von ihm gehaltenen Veranstaltungen enthalten sein sollen. `toString` soll dann für Professoren nicht nur die `Person`-Informationen liefern, sondern auch die Namen der gehaltenen Veranstaltungen.

Teste deine Klassen mit Hilfe einer main-Methode, in der du die Professoren des Lehrstuhls (samt ihrer Veranstaltungen, siehe Instituts-Website), dich selbst, drei deiner Kommilitonen und drei deiner Lieblings-Prominenten anlegst. Benutze Collections und Schleifen, um alle 100 paarweisen `equals`-Vergleiche dieser 10 Personen durchzuführen. Für jeden Vergleich sollen beide Personen und das Vergleichsergbnis ausgegeben werden.

```
public class Person{
    private String name;
    private int age;
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return name;
    }
    public void setAge(int age){
        this.age = age;
    }
    public int getAge(){
        return age;
    }
    public String toString(){
        return "Name: "+this.name+" | Alter: "+this.age;
    }
    public boolean equals(Object obj){
        if(!(obj instanceof Person))
            return false;
        return ((Person)obj).name.equals(this.name) &&
                        ((Person)obj).age == this.age;
    }
    public int compareTo(Person o){
        return this.getName().compareTo(o.getName());
    }
}
```

```java
import java.util.*;
public class Professor extends Person
{
    Set<String> veranstaltungen;

    public Professor(String name, int alter){
        this(name,alter,new HashSet<>());
    }
    public Professor(String name, int alter, Set<String> veranstaltungen){
        this.setName(name);
        this.setAge(alter);
        this.veranstaltungen = veranstaltungen;
    }
    public void addVeranstaltung(String veranstaltung){
        this.veranstaltungen.add(veranstaltung);
    }
    public Set<String> getVeranstaltungen(){
        return veranstaltungen;
    }
    public boolean equals(Object obj){
        if(!(obj instanceof Professor)) return false;
        Professor other = (Professor)obj;
        if(other.veranstaltungen.size()!=veranstaltungen.size())
            return false;
        return veranstaltungen.containsAll(other.veranstaltungen);
    }
}
```

```java
public class Student extends Person{
    private int matrikelnummer;

    public Student(String name, int alter, int matrikelnummer){
        this.setName(name);
        this.setAge(alter);
        this.matrikelnummer = matrikelnummer;
    }
    public void setMatrikelnummer(int matrikelnummer){
        this.matrikelnummer = matrikelnummer;
    }
    public int getMatrikelnummer(){
        return matrikelnummer;
    }
    public boolean equals(Object obj){
        if(!(obj instanceof Student)) return false;
        return ((Student)obj).matrikelnummer == this.matrikelnummer;
    }
}
```

```java
public static void main(String[] args){
        Person[] personen = new Person[10]();
        personen[0]=new Person("my name",18);
        //weitere Personen anlegen (nicht gezeigt)
        for(int i=0;i<personen.length;i++){
                for(int j=0; j<personen.length;j++){
                        System.out.println(personen[i]+" vs "+personen[j]+
                                ": "+personen[i].equals(personen[j]));
                }
        }
}
```

2. Vererbung - Variablen und Methoden

Lege zwei Klassen `Superclass` und `Subclass extends Superclass` an. Definiere in `Superclass` die Variablen `public String test_public`, `private String test_private` und `protected String test_protected` und initialisiere sie mit beliebigen Werten. Lege in `Subclass` eine `main` Methode an und erzeuge dort eine Variable `Subclass sub = new Subclass()`.

Überlege dir für alle folgenden Aufgaben zunächst das Resultat und überprüfe dann deine Ergebnisse. Falls eine Aufgabe einen Compilerfehler erzeugt, kommentiere den entsprechenden Code aus.

(a) Gib in `Subclass.main()` über direkten Zugriff (`sub.test_public`, etc.) die Werte der drei Variablen in separaten print-Statements aus `Superclass` aus.

(b) Definiere die selben Variablen aus `Superclass` auch in `Subclass`, initialisiere sie mit anderen Werten und lass dein Programm laufen.

(c) Gib in der `main` Methode nun die Werte der Variablen aus beiden Klassen aus.

(d) Überschreibe in `Superclass` die Methode `public String toString()` und konkateniere in dieser alle Variablen. Gib `sub.toString()` aus.

(e) Implementiere in `Superclass` eine Methode `void modify()`, die an alle Variablen den String "_super" anhängt. Rufe in der `toString` Methode in `Superclass` zunächst `modify()` auf.

(f) Überschreibe nun in `Subclass` die Methode `void modify()`. In dieser soll nun der String "_sub" angehängt werden. Lass dein Programm erneut laufen.

```
public class Superclass {
    public String test_public="public";
    protected String test_protected="protected";
    private String test_private="private";

    public String toString(){
        modify();
        return test_public+"␣"+test_protected+"␣"+test_private;
    }
    public void modify(){
        test_public+="_super";
        test_private+="_super";
        test_protected+="_super";
    }
}
```

```
public class Subclass extends Superclass{
    public String test_public = "sub public";
    public String test_protected = "sub protected";
    public String test_private = "sub private";

    public void modify(){
        test_public+="_sub";
        test_protected+="_sub";
        test_private+="_sub";
    }
    public static void main(String[] args)
    {
        Subclass sub = new Subclass();
        System.out.println(sub);
    }
}
```