ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Dr. Jochen Hoenicke
Alexander Nutz

26.5.2014
Abgabe: Montag 2.6.2014
via Daphne/Subversion

5. Übungsblatt zur Vorlesung
Einführung in die Informatik

Aufgabe 1: Polymorphie

Schreiben Sie ein Programm, das Funktionen verschiedener Arten verwalten, ausgeben und berechnen kann.

- Auf jeder Funktion soll eine Methode `double computeY(double x)` aufgerufen werden können, die für einen gegebenen x-Wert den y-Wert der Funktion bestimmt, sowie eine Methode `String toString()`, die die Funktion in möglichst einfacher aber lesbarer Form ausgibt (also etwa `5 x^3 - 8` oder `(x^2) o (4^x)`). Nutzen Sie ein Interface.

- Erstellen Sie Klassen für Polynome, Affine Funktionen (d.h. Funktionen von der Form $f(x) = mx+b$ für zwei Konstanten m, b) und einfache Exponentialfunktionen von der Form $f(x) = c^x$ für eine Konstante c. Hinweis: Die statische Methode `double Math.pow(double x, double y)` berechnet x^y für Sie.

- Erstellen Sie weiterhin Klassen, die Operationen auf den Funktionen erlauben, nämlich Funktionskomposition und Funktionssummierung.

- Probieren Sie, ob alles funktioniert, in dem Sie folgende `main` Methode verwenden: (Sie können natürlich sinnerhaltende Anpassungen vornehmen)

```
public class FunctionTest {
  public static void main(String[] args) {
    double x = Double.parseDouble(args[0]);
    Function f1 = new Polynomial(new double[] { 0.0, 2.0, -4.0 });
    Function f2 = new AffineFunction(-5.5, 2.0);
    Function f3 = new SimpleExponentialFunction(2.0);
    Function f4 = new CompositeFunction(f1, f2);
    Function f5 = new FunctionSum(f3, f4);
    Function[] functions = new Function[] {f1, f2, f3, f4, f5};
    for (Function f : functions) {
      System.out.println("for x = " + x + ", and f(x) = " + f
        + ", the result is f(x) = " + f.computeY(x));
    }
  }
}
```

Aufgabe 2: Call By Reference
Was ist die Ausgabe von folgendem Programm, und warum?

```
public class CallByReference {
  public static void main(String[] args) {
    int[] arr = new int[5];
    for (int i = 0; i < arr.length; i++) {
     arr[i] = i;
    }
    printIntArray(arr);
    m1(arr);
    printIntArray(arr);
    m2(arr);
    printIntArray(arr);
  }

  static void m1(int[] ia) {
    for (int i = 0; i < ia.length; i++) {
      ia[i] = i + 1;
    }
  }

  static void m2(int[] ia) {
    int[] ib = new int[ia.length];
    for (int i = 0; i < ib.length; i++) {
      ib[i] = i + 2;
    }
    ia = ib;
  }

  /**
   * Takes an in array as argument and prints it nicely
   * to standard out (the console typically).
   */
  static void printIntArray(int[] a) {
    String result = "[";
    String comma = "";
    for (int i = 0; i < a.length; i++) {
      result += comma + a[i];
      comma = ", ";
    }
    result += "]";
    System.out.println(result);
  }
}
```

Aufgabe 3: Konstruktoren
Vervollständigen Sie in folgendem Programm die Konstruktoren. Benutzen Sie dabei `super` und `this` um andere Konstruktoren aufzurufen.

```
public class Address {
  String name;
  String street;
  int nr;
  int plz;
  String city;

  public Address(String name, String street, int nr, int plz,
      String city) {
    ...
  }
  public Address(name) {
    ...
  }
}

public class FaxAddress extends Address {
  int phone;
  int fax;
  public FaxAddress(String name, String street, int nr, int plz,
      String city, int phone, int fax) {
    ...
  }
  public FaxAddress(String name, int phone, int fax) {
    ...
  }
}
```