Übung 2 - Objektorientierte Programmierung

1. Java Applet

Programmieren Sie ein kleines Java Applet. Dazu muss die Klasse java.applet.Applet erweitert werden und die Methode

```
void paint(java.awt.Graphics g)
```

überschrieben werden.

Das Rahmenprogramm zu ihrem Applet sieht damit folgendermassen aus:

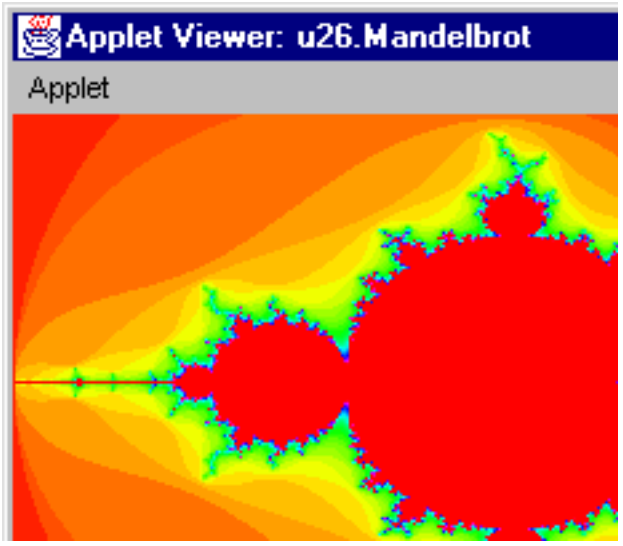
```
import java.awt.*
import java.applet.Applet;

public class ArtApplet extends Applet {
    public void paint (Graphics g){
        // draw a picture on g
    }
}
```

In der Methode paint soll z.B. eine Linie oder ein Rechteck gezeichnet werden. Dazu können die Methoden der Klasse Graphics verwendet werden. Diese Methoden erlauben es, auch eine etwas ansprechendere Grafik zu zeichnen (siehe nebenstehendes Bild).

Die Groesse des Applets kann mit der Methode getSize (geerbte Methode des Applets) abgefragt werden. Resultat dieser Methode ist vom Typ java.awt.Dimension mit den Feldern width und height.

Um Ihr Applet anschauen zu können müssen Sie eine HTML Seite Definieren in welcher das Applet mit dem APPLET Tag enthalten ist. Diese HTML Seite kann dann entweder mit dem appletviewer (Tool aus dem Java jdk) oder mit einem Browser, der Java (noch) unterstützt, betrachtet werden.

Sie können Ihr Applet erweitern, so dass eine Mandelbrotfigur gezeichnet wird. Diese Figur stellt eine Funktion in der komplexen Ebene dar (in der Figur im Bereich -2-1.2i bis 1+1.2i).



Der Funktionswert für das kompexe Argument c berechnet man, indem man für c die Iteration

zn+1 := zn*zn + c, zn = 0

ausführt bis |z| >= 2.0 oder n >= 40. Die Anzahl n der benötigten Iterationen bis zum Abbruch ist der Funktionswert für c. Dieser wird als Farbwert dargestellt. Um ein Pixel zu setzen soll ein Rechteck mit Breite und Höhe 1 gezeichnet werden.

Beim Berechnen der Figur soll die Klasse für komplexe Zahlen aus Übung 1 verwendet werden.

2. Polymorphismus
Was wird ausgegeben wenn folgendes Programm *Test* ausgeführt wird? Geben Sie jeweils an welche Methode aus welcher Methode aufgerufen wird und ob dieser Aufruf statisch oder dynamisch gebunden ist.

```
public class Test {
    public static void main(String[] args){
        C c = new D();
        c.f();
    }
}
class B {
    void f(){System.out.println("B::f"); h(); }
    void h(){System.out.println("B::h");}
}
class C extends B {
    void f(){System.out.println("C::f"); g(); }
    void g(){System.out.println("C::g"); super.f();}
}
class D extends C {
    void f(){System.out.println("D::f"); super.f();}
    void g(){System.out.println("D::g"); super.g();}
    void h(){System.out.println("D::h"); }
}
```

3. Überladen und Überschreiben von Methoden
Was wird ausgegeben, wenn folgendes Programm *XYtest* ausgeführt wird? Sie können das Beispiel mit einem Compiler zu übersetzen (Übersetzung ist fehlerfrei möglich), ausführen und dann versuchen, aus dem Resultat eine Regel abzuleiten. Wie wird entschieden, welche Methode ausgeführt wird?

```
class A {}
class B extends A {}

class X {
   int get(A x){return 0;}
}

class Y extends X {
   int get(A x){return 1;}
   int get(B y){return 2;}
}

public class XYtest {
   public static void main(String[] args){
      A a = new A();
      B b = new B();

      X x = new X();
      Y y = new Y();
      X xy = new Y();

      System.out.println(x.get(a));
      System.out.println(x.get(b));
      System.out.println(y.get(a));
      System.out.println(y.get(b));
      System.out.println(xy.get(a));
      System.out.println(xy.get(b));
   }
}
```

Abgabe: 12. März 2012, 08:00