Beispiellösung der Klausur zu Objektorientierte Programmierung mit Java 09. Juli 2004 (SS 2004)

Bearbeitungszeit: 120 Minuten
NICHT MIT BLEISTIFT SCHREIBEN!

Name:
Vorname:
Matrikelnummer:

Nr	Max	Bewertung	
1	**8**	xxxxx	
(a)	6		xxxxx
(b)	2		xxxxx
2	**9**	xxxxx	
(a)	3		xxxxx
(b)	3		xxxxx
(c)	2		xxxxx
(d)	1		xxxxx
3	**9**	xxxxx	

Nr	Max	Bewertung	
4	**14**	xxxxx	
(a)	5		xxxxx
(b)	4		xxxxx
(c)	1		xxxxx
(d)	4		xxxxx
5	**12**	xxxxx	
6	**18**	xxxxx	
7	**10**	xxxxx	
(a)	2		xxxxx
(b)	8		xxxxx
Summe	**80**		

Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe oder die Rückseite unter Angabe der Aufgabe. Nennen Sie möglichst alle Annahmen, die Sie für die Lösung einer Aufgabe treffen! Bei den vorgegebenen Programmen wurden die `include`-Zeilen der Übersicht halber weggelassen.

Viel Erfolg!

Aufgabe 1 **(8 Punkte)**

Der Einstiegspunkt in ein Java-Programm ist üblicherweise die folgende Methode:

```
public static void main( String[] args )
```

Diese Methode befindet sich in einer Klasse, die dem Java-Interpreter als Argument übergeben wird.

(a) 6 Punkte

Erklären Sie kurz die Bestandteile *public, static* und *void* des Methodenkopfes.

public *zeigt an, daß die Methode nach außen hin bekannt ist.*

static *zeigt an, daß es sich um eine Klassenmethode und nicht um eine Instanzmethode handelt; ihre Verwendung ist also nicht an ein Objekt der Klasse gebunden.*

void *zeigt an, daß die Methode keinen Rückgabewert besitzt.*

(b) 2 Punkte

Das Array `args` enthält die Kommandozeilenparameter (falls welche vorliegen). Angenommen, wir schreiben ein Java-Programm, das *ausschließlich ohne* Argumente aufgerufen werden wird. Ist dann anstatt des oben angegebenen Methodenkopfes auch der folgende zulässig?

```
public static void main()
```

Begründen Sie Ihre Antwort!

Zulässig ist er insoweit zwar schon, als daß der Compiler keine Fehlermeldung produzieren wird. Ein Problem bekommen wir jedoch zur Laufzeit: Da in Java zur Signatur einer Methode neben dem Methodennamen und dem Rückgabewert auch die Parameterliste gehört, sind die obigen Methoden (mit bzw. ohne Parameterliste) aus Sicht des Java-Interpreters unterschiedlich. Er benötigt zur Ausführung eines Programms aber unbedingt eine Methode mit der Signatur

```
public static void main( String[] args )
```

Kann er diese nicht finden, so gibt er eine entsprechende Fehlermeldung aus.
Also: Auch wenn die Kommandozeilenparamter nicht gebraucht werden, müssen sie dennoch in der Signatur aufgeführt werden!

Aufgabe 2 **(9 Punkte)**

Gegeben sei die Java-Klasse `Obscure.java` (siehe Programmtext 2.1 auf Seite 4):

(a) 3 Punkte

Beschreiben Sie kurz, welche Ausgabe sich der Programmierer dieser Klasse wohl gewünscht hat (entweder mit eigenen Worten oder als Bildschirmausgabe)!

```
zero    entspricht      0
one     entspricht      1
two     entspricht      2
three   entspricht      3
```

Programm 2.1 Obscure.java

```
import java.util.*;

public class Obscure {

   public static HashMap hm = new HashMap();

   public static String[] NumString = new String[4];

   public static void fillHashMap() {
      for ( int i = 0; i < 4; i++ ) {
         hm.put( NumString[i], i );
      }
   }

   public static void printHashMap() {
      for ( int i = 0; i < 4; i++ ) {
         String temp = NumString[i];
         System.out.println( temp + "\tentspricht\t" +
            hm.get( temp ) );
      }
   }

   public static void main ( String[] args ) {
      NumString[0] = "zero";
      NumString[1] = "one";
      NumString[2] = "two";
      NumString[3] = "three";

      fillHashMap();
      printHashMap();
   }

}
```

(b) 3 Punkte

Der Dokumentation der Klasse `HashMap` entnehmen Sie, dass diese Klasse u. A. folgende Methoden besitzt:

```
public Object remove(Object key)
public boolean containsKey(Object key)
public Object put(Object key, Object value)
public Object get(Object key)
```

Beim Compilieren stellen Sie fest, daß der Compiler folgende Fehlermeldung produziert:

```
theseus$ javac Obscure.java
Obscure.java:10: cannot resolve symbol
symbol  : method put  (java.lang.String,int)
location: class java.util.HashMap
       hm.put( NumString[i], i );
          ^
1 error
```

Diese hat ihre Ursache in der Methode `fillHashMap()`. Was führt in dieser Methode zur einer Fehlermeldung?

Lösung: *Dieses Programm ist nicht lauffähig, weil die Methode* `put()` *der Klasse* `HashMap` *die folgende Signatur besitzt:*

```
public Object put(Object key, Object value)
```

Das heißt insbesondere, daß als Wert Objekte übergeben werden müssen, im obigen Programm wird aber ein primitiver Datentyp übergeben. Der Compiler findet also keine passende Methode und bricht den Übersetzungsvorgang mit der zitierten Fehlermeldung ab.

(c) 2 Punkte

Geben Sie nun an, welche Änderung vorgenommen werden muß, damit das Programm lauffähig wird!

In der Methode `fillHashMap()` *muß die HashMap unter Verwendung einer sogenannten Wrapperklasse gefüllt werden. Wrapperklassen transformieren primitive Datentypen in Objekte.*
Eine HashMap erwartet sowohl für den Schlüssel als auch für den Wert Objekte, in der Impementierung aus der Aufgabenstellung wird aber versucht, als Wert eine Integer-Variable (und damit also kein Objekt*) zuzuweisen.*
Zur Lösung dieses Problems muß die relevante Zeile der Methode in

```
hm.put( NumString[i], new Integer(i) );
```

abgeändert werden.

(d) 1 Punkte
Wie ist der Fachbegriff in Java, auf den es bei dieser Änderung ankommt?

Wrapperklasse

Aufgabe 3 **(9 Punkte)**

Gegeben seien die **Klasse A**,

```
package a;
public class A {
    private int x;
    protected int y;
    public int z;
    public void print() {
        System.out.println(x); // (1)
        System.out.println(y); // (2)
        System.out.println(z); // (3)
    }
}
```

die **Klasse B**

```
package b;
import a.A;
public class B extends A {
    public void print() {
        System.out.println(x); // (4)
        System.out.println(y); // (5)
        System.out.println(z); // (6)
    }
}
```

und die **Klasse C**.

```
package c;
import b.B;
public class C {
    public static void main(String[] args) {
        B b = new B();
        System.out.println(b.x); // (7)
        System.out.println(b.y); // (8)
        System.out.println(b.z); // (9)
    }
}
```

Kreuzen Sie in der folgenden Tabelle an, ob der Compiler die Anweisung mit der jeweiligen Nummer akzeptiert oder eine Fehlermeldung wegen dieser Anweisung ausgibt.

Anweisung	akzeptiert	meldet Fehler
(1)	X	
(2)	X	
(3)	X	
(4)		X
(5)	X	
(6)	X	
(7)		X
(8)		X
(9)	X	

Richtige Kreuze bringen einen Punkt, falsche Kreuze führen zu Punktabzug!

Aufgabe 4 **(14 Punkte)**

Gegeben sei die folgende Klasse `Quadrat`:

Programm 4.2 Quadrat.java

```
public class Quadrat {

   protected int len;   // Seitenlaenge

   Quadrat( int len ) {
      this.len = len;
   }

   public int getArea() {
      return len * len;
   }
}
```

(a) 5 Punkte

Erweitern Sie diese Klasse so, daß eine neue Klasse `Rechteck` entsteht (die neben der Seitenlänge auch eine Seitenhöhe besitzt). Schreiben Sie den dazu notwendigen **Konstruktor** und überschreiben Sie die Methode `getArea()`.

```
public class Rechteck extends Quadrat {

   protected int height;   /* Seitenhoehe,
                              die Seitenlaenge haben wir
                              bereits in der Oberklasse */

   Rechteck ( int len, int height ) {
      super(len);
      this.height = height;
   }

   public int getArea() {
      return len * height;
   }

}
```

(b) 4 Punkte

Gegeben sei die folgende `main()`-Methode:

Programm 4.3 Eine `main()`-Methode

```
public static void main ( String[] args ) {

   Rechteck q1 = new Rechteck(2,4);
   System.out.println(q1.getArea());

   Quadrat q2 = q1;
   System.out.println(q2.getArea());

}
```

Geben Sie an, welche Methoden-Implementierungen bei `q1.getArea()` und bei `q2.getArea()` aufgerufen werden und wie die Ausgabe aussieht. Begründen Sie Ihre Antwort. (Sie können selbstverständlich davon ausgehen, daß die `main()`-Methode in eine öffentlich zugängliche Klasse geschrieben wurde und die Klassen `Quadrat` und `Rechteck` zur korrekten Verwendung vorliegen.)

```
8
8
```

(c) 1 Punkte

Wie heißt der objektorientierte „Mechanismus“, der bei `q2.getArea()` zum Tragen kommt?

Polymorphie

(d) 4 Punkte

Um die Gleichheit zweier Quadrat-Instanzen zu überprüfen verwendet man die Methode `equals()` (der Klasse `Quadrat`). Diese bekommt als Parameter ein Objekt der Klasse `Object` übergeben, das mit Hilfe des Operators `instanceof` dahingehend überprüft wird, zu welcher Klasse es gehört.
Implementieren Sie in der nachfolgenden Methode `equals` den Teil, der zum Vergleich zweier Objekte der Klasse `Quadrat` relevant ist:

Programm 4.4 Die `equals()`-Methode

```
public boolean equals ( Object obj ) {

   if ( obj instanceof Quadrat ) {
     /*
        Das hier ist nun Ihr Teil!!!
     */
   } else {
      return super.equals(obj);
   }

}
```

```
    public boolean equals ( Object obj ) {

       if ( obj instanceof Quadrat ) {
          Quadrat q = (Quadrat) obj;
          return (len == q.len);
       } else {
          return super.equals(obj);
       }

    }
```

Aufgabe 5 **(12 Punkte)**

Gegeben sei die folgende Klasse `UseComplex.java` (siehe Programm 5.5 auf Seite 12.

Programm 5.5 `UseComplex.java`

```
public class UseComplex {

   public static void main ( String[] args ) {

      // Zwei Objekte der komplexen Zahlen anlegen
      Complex z1 = new Complex(5, 7);
      Complex z2 = new Complex(7, 2);

      /* Komplexe Zahl ausgeben */
      System.out.println( "Ausgabe von z1:   "+z1 );
      System.out.println( "Ausgabe von z2:   "+z2 );

      /* Eine komplexe Zahl mit einem Skalar (Integer)
         multiplizieren und das Ergebnis ausgeben */
      Complex z3 = z1.multiplyBy( 5 );
      System.out.println( "5 * z1 ergibt:   "+z3 );

      /* Zwei komplexe Zahlen multiplizieren
         und das Ergebnis ausgeben */
      Complex z4 = z1.multiplyBy( z2 );
      System.out.println( "z1 * z2 ergibt:   "+z4 );

   }
}
```

Dieses Programm verwendet die Klasse `Complex`, welche die komplexen Zahlen darstelle. Es produziert die folgende Ausgabe:

```
theseus$ java UseComplex
Ausgabe von z1:   5 + i*7
Ausgabe von z2:   7 + i*2
5 * z1 ergibt:   25 + i*35
z1 * z2 ergibt:   21 + i*59
```

Schreiben Sie die Klasse `Complex.java` mit allen notwendigen Variablen und Methoden, die dazu nötig sind, um das gegebene Programm 5.5 lauffähig zu machen.

Erläuterungen zu komplexen Zahlen:

- Eine komplexe Zahl (x, y) besteht stets aus einem *Realteil* x und einem *Imaginärteil* y, die beide jeweils $\in \mathbb{R}$ sind.
- Eine gebräuchliche Darstellung einer komplexen Zahl z ist:

$$z = x + i \cdot y$$

- Die Addition zweier komplexer Zahlen ist komponentenweise definiert, das heißt:

$$(a, b) + (c, d) = (a + c, b + d)$$

- Die Multiplikation einer komplexen Zahl (x, y) mit einem Skalar $\lambda \in \mathbb{R}$ ist ebenfalls komponentenweise definiert:

$$\lambda \cdot (x, y) = (\lambda \cdot x, \lambda \cdot y)$$

- Die Multiplikation zweier komplexer Zahlen $(a_1, b_1), (a_2, b_2)$ ist wie folgt definiert:

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1 a_2 - b_1 b_2 + i \cdot (a_1 b_2 + b_1 a_2))$$

Zur Vereinfachung können Sie für diese Aufgabe annehmen, daß für eine komplexe Zahl (x, y) gilt: $x, y \in \mathbb{N}$!

Programm 5.6 Complex.java

```
public class Complex {

   private int real, im;

   public Complex(int a, int b) {
      real = a;
      im   = b;
   }

   public Complex add( Complex b ) {
      return new Complex( this.real+b.real, this.im+b.im );
   }

   public Complex multiplyBy( int b ) {
      return new Complex( this.real*b, this.im*b );
   }

   public Complex multiplyBy( Complex b ) {
      return new Complex(
         this.real*b.real - this.im*b.im,
            this.real*b.im + this.im*b.real
      );
   }

   public String toString() {
      return new String ( "" + this.real + " + i*" + this.im );
   }
}
```

Aufgabe 6 **(18 Punkte)**

Implementieren Sie unter Verwendung von `javax.swing` die folgende grafische Applikation `XTrans` (mit **einem** Fenster, das die folgenden beiden Gestalten haben kann):

$\Longrightarrow$

Erklärung:

- Die Anwendung soll aus zwei grafischen Widgets (einem Kombinationsfeld und einer Schaltfläche) bestehen.
- Das Kombinationsfeld enthält eine Auflistung von Strings (hier: ”Hund”, ”Katze”, ”Maus” bzw. ”Dog”, ”Cat”, ”Mouse”)
- Bei jedem Anklicken der Schaltfläche soll die Sprache für
  1. den Inhalt des Kombinationsfeldes sowie
  2. die Aufschrift der Schaltfläche

  von Deutsch nach Englisch bzw. umgekehrt geändert werden.
- Desweiteren soll Ihre Anwendung beim Anklicken des entsprechenden Symbols in der Titelzeile des Fensters geschlossen werden.

Eventuell könnte Ihnen die folgende Methode für JComboBox-Objekte hilfreich sein:

public void setModel(ComboBoxModel aModel)

```
Sets the data model that the JComboBox uses to obtain
the list of items.

Parameters:
    aModel - the ComboBoxModel that provides the displayed
             list of items
```

aus: Java-Dokumentation, http://java.sun.com

Lösung:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class XTrans {

   public static void main ( String[] args ) {

      /* Zwei Stringarrays vorbereiten
         (fuer deutschen bzw. englischen Inhalt) */
      final String[] animalsEnglish = { "Dog", "Cat", "Mouse" };
      final String[] animalsGerman = { "Hund", "Katze", "Maus" };
      final DefaultComboBoxModel animalsModelGerman =
         new DefaultComboBoxModel( animalsGerman );
      final DefaultComboBoxModel animalsModelEnglish =
         new DefaultComboBoxModel( animalsEnglish );

      // Das Fenster (besser: den Frame) vorbereiten
      JFrame frame = new JFrame();
      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      frame.getContentPane().setLayout( new GridLayout( 0,1 ) );

      /* Die beiden grafischen Widgets erzeugen
         und dem Frame hinzufuegen /*
      final JComboBox cmbBox = new JComboBox( animalsModelGerman );
      final JButton button = new JButton( "Switch to English" );
      frame.getContentPane().add( cmbBox );
      frame.getContentPane().add( button );

      // Aktion fuer Anklicken der Schaltflaeche waehlen
      ActionListener al = new ActionListener() {
         public void actionPerformed ( ActionEvent e ) {
            if (e.getActionCommand().indexOf( "English" ) != -1) {
               button.setText( "In Deutsch umschalten" );
               cmbBox.setModel( animalsModelEnglish );
            }
            if (e.getActionCommand().indexOf( "Deutsch" ) != -1) {
               button.setText( "Switch to English" );
               cmbBox.setModel( animalsModelGerman );
            }
         }
```

```
        };

        button.addActionListener( al );

        // Das Uebliche zum Anzeigen
        frame.pack();
        frame.show();
    }
}
```

Aufgabe 7 **(10 Punkte)**

Gegeben ist die Schnittstelle der folgenden Java-Klasse:

```
public class StringOps {
    public static String concat(String s1, String s2);
}
```

Die Methode `concat()` hängt einfach die beiden Strings aneinander und gibt das Ergebnis zurück.

(a) 2 Punkte

Geben Sie *zwei* verschiedenartige Testfälle für die Methode `concat()` an (und noch kein Java-Programm!).

Lösungsvorschlag:

s1	s2	erwartete Ausgabe
„“	„“	„“
„“	„a“	„a“

(b) 8 Punkte

Implementieren Sie diese *zwei* Testfälle als JUnit-Tests in der eigenen Klasse *StringOpsTest*. Geben Sie die vollständige Implementierung (inkl. import-Anweisung) an. Gestalten Sie die Implementierung so, dass ggf. auch (ohne Änderung der Klasse) nur einer der beiden Testfälle ausgeführt werden könnte.

```
import junit.framework.*;

public class StringOpsTest {
    public StringOpsTest(String s) {
        super(s);
    }
    public void testZeroZero() {
        String result = StringOps.concat("", "");
        String expected = "";
        assertEquals(expected, result);
    }
    public void testZeroOne() {
        String result = StringOps.concat("", "a");
        String expected = "a";
        assertEquals(expected, result);
    }
}
```