Brandenburgische
Technische Universität
Cottbus - Senftenberg

Brandenburgische Technische Universität Cottbus-Senftenberg
Institut für Informatik, Informations- und Medientechnik
PD Dr.-Ing. habil. T. Hinze, Prof. Dr. phil. habil. G. Wagner

Sommersemester 2013

# Einführung in die Programmierung
# Wiederholung der Endklausur am 24. September 2013

| Name: | Vorname: |
|---|---|
| Matrikelnr.: | Studiengang: |

## Hinweise

- Die Bearbeitungszeit beträgt **120 Minuten**.
- Außer einem dokumentenechten Stift (schwarz oder blau) sind keinerlei Hilfsmittel zugelassen. Ebenso darf kein selbst mitgebrachtes Papier verwendet werden.
- Notieren Sie bitte auf dem Deckblatt Ihren Namen, Vornamen, Studiengang und Ihre Matrikelnr.
- Schreiben Sie Ihre Lösungen zu den Aufgaben in die freigelassenen Stellen zwischen den Aufgaben. Falls der Platz nicht ausreicht, werden Ihnen auf Anforderung weitere Blätter zur Verfügung gestellt.
- Nehmen Sie sich kurz Zeit, um zunächst alle Aufgaben durchzulesen, bevor Sie mit der Bearbeitung beginnen.

| Aufgabe | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Σ |
|---|---|---|---|---|---|---|---|---|
| Max. Punkte | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 70 |
| Erreichte Punkte | | | | | | | | |

## Viel Erfolg!

### Aufgabe 1 (Ausgewähltes Grundlagen- und Faktenwissen) — 10 Punkte

1. Nennen Sie zwei verschiedene Sprachkonstrukte in Java, mit denen sich *Fallunterscheidungen* ausdrücken lassen. (2 Punkte)

   -
   -

2. Welche der folgenden drei Aussagen (a) bis (c) sind richtig und welche falsch? (3 Punkte)

   (a) Java ist eine *objektorientierte*, *plattformunabhängige* und *statisch getypte* Programmiersprache.

   .....

   (b) Jede Java-Klasse muss eine `main`-Methode enthalten.

   .....

   (c) Vom Java-Compiler erzeugte `.class`-Dateien können mit der *Java Virtual Machine* ausgeführt werden.

   .....

3. Wo sind Attribute und Methoden sichtbar, die als `private` gekennzeichnet sind? (1 Punkt)

4. Wieviel Bit Speicher belegt ein Wert vom Typ `double` in Java? (1 Punkt)

5. Welchen Dezimalwert hat die als Hexadezimalzahl in Java gegebene Konstante `0xA` ? (1 Punkt)

6. Gegeben seien folgende Variablenvereinbarungen in Java:

```
double x = 2.0e+2;
long y;
```

   Welche Rechenergebnisse liefern die folgenden zwei Ausdrücke jeweils für `y`? (2 Punkte)

```
y = (long) x; .....
```

```
y = (long) x % (long) x; .....
```

**Aufgabe 2 (Programmtransformation von JavaScript nach Java)** **10 Punkte**

Überführen Sie folgendes JavaScript-Programm zur Berechnung aller Primzahlen bis 100 mit dem Sieb des Erathostenes in ein semantisch äquivalentes Java-Programm, in welchem *keine* `for`-Schleifen verwendet werden. Semantisch äquivalent bedeutet hier, dass in der JavaScript- und in der Java-Version gleiche Ausgaben entstehen.

```
<script type="text/javascript">
 var n = 100;
 var gestrichen = new Array();

 for(var i = 2; i < n; i = i+1) {
   gestrichen[i] = false;
 }
 for(var i = 2; i < n; i = i+1) {
   if (!gestrichen[i]) {
     //i ist prim, gib i aus
     document.writeln(i);
     //Streiche seine Vielfachen i*i, i*i+i, i*i+2*i, ..., n
     for(var k = i*i; k < n; k = k + i) {
       gestrichen[k] = true;
     }
   }
 }
</script>
```

Verwenden Sie folgendes Java-Programmgerüst:

```
public class PrimzahlenBis100 {
  public static void main(String[] args) {
    int n = 100;
    boolean gestrichen[] = new boolean[n+1];









  } //main
} //class
```

**Aufgabe 3 (Verstehen eines gegebenen Java-Programmes)** **10 Punkte**

Gegeben sei folgendes Java-Programm:

```
public class MagischeFunktion {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    int i = 1;

    if (n < 0) {
      System.out.println("Fehlerhafte Eingabe.");
    } else {
      if (n > 1) {
        i = n;
        while (n > 1) {
          n = n - 1;
          i = i * n;
        }
      }
      System.out.println(i);
    }
  }
}
```

1. Wieviele Argumente werden bei Aufruf von `MagischeFunktion` erwartet und in welchen Datentyp werden sie umgewandelt? (2 Punkte)

2. Welche Ausgabe wird durch den Aufruf `java MagischeFunktion 1` erzeugt? (1,5 Punkte)

3. Welche Ausgabe wird durch den Aufruf `java MagischeFunktion 2` erzeugt? (1,5 Punkte)

4. Welche Ausgabe wird durch den Aufruf `java MagischeFunktion 3` erzeugt? (1,5 Punkte)

5. Welche Ausgabe wird durch den Aufruf `java MagischeFunktion 4` erzeugt? (1,5 Punkte)

6. Wie heißt die durch das Programm berechnete Funktion auf natürlichen Zahlen in der Mathematik? (2 Punkte)

**Aufgabe 4 (Fehler in gegebenem Java-Quelltext finden und beseitigen)** **10 Punkte**

Das folgende Java-Programm soll den Flächeninhalt und den Inkreisradius eines regelmäßigen Fünfecks bestimmen, dessen Seitenlänge $a$ vom Nutzer eingegeben wird. Damit berechnet sich der Flächeninhalt durch $\frac{a^2}{4} \cdot p$ und der Inkreisradius durch $\frac{a}{10} \cdot p$, wobei $p = \sqrt{25 + 10 \cdot \sqrt{5}}$.
Das gegebene Java-Programm enthält *zehn* Fehler, die eine erfolgreiche Compilierung verhindern. Kennzeichnen Sie die fehlerhaften Stellen im Quelltext, und beseitigen Sie die Fehler.

```
import java.util.*;

clas Pentagon {
  public static void main(String[] a) {
    double a = 0.0
    String erg;

    System.out.print("Regelmaessiges Fuenfeck - Eingabe der Seitenlaenge: ");
    Scanner sc = new Scan(System.in);
    a = Double.parseDouble(sc.next());
    if (a < 0) {
      System.out.printIn("Seitenlaenge ungueltig."); return;

    if (a = 0) {
      erg = "Flaecheninhalt: 0, Inkreisradius: 0";
    } else {
      erg = "Flaecheninhalt: " + Flaecheninhalt(a) +
            ", Inkreisradius: " + Inkreisradius(a);
    }
    System.out.println(erg);
  }

  public static double Inkreisradius(double s)
    double p = Math.sprt(25.0 + 10.0 * Math.sqrt(5.0));
    return s * p / 10;
  }

  public double Flaecheninhalt(double s) {
    double p = Inkreisradius(s) * 10 / s;
    return s * s * p / 4;
  }
}
```

**Aufgabe 5 (Java-Lückenquelltext zur Verwaltung von 2D-Punktkoordinaten) 10 Punkte**

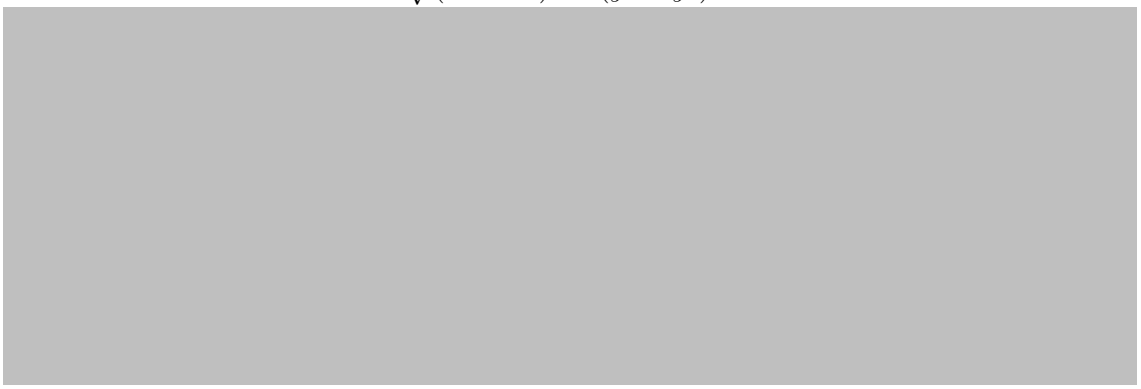
Gegeben ist das folgende lückenhafte Java-Programm zur Verwaltung von 2D-Punktkoordinaten, das aus den zwei Klassen `MeinePunkte2D` sowie `Punkt2D` besteht. Wir nehmen dazu ein zweidimensionales kartesisches Koordinatensystem an, dessen x-Achse wie in der Mathematik üblich horizontal von links nach rechts verläuft und dessen y-Achse vertikal von unten nach oben gerichtet ist. In der Klasse `Punkt2D` werden x-Wert und y-Wert eines Punktes als private Attribute geführt, die über die Methoden `GibXWert` und `GibYWert` von außerhalb der Klasse ausgelesen werden können.

- Vervollständigen Sie die Lücken im gegebenen Programm, indem Sie an den grau hinterlegten Stellen entsprechenden Programmcode ergänzen. Der resultierende Programmcode soll die jeweils als Kommentar beschriebene Funktionalität umsetzen und ohne weitere Ergänzungen compilierbar und fehlerfrei ausführbar sein. An den vorgegebenen Quelltextteilen dürfen keine Veränderungen vorgenommen werden.

```
public class MeinePunkte2D {
  public static void main(String[] args) {
    //Punkt (2, 4) anlegen als Objekt a der Klasse Punkt2D
    
    //Punkt a ausgeben mittels der Methode Ausgeben
    
    //Punkt (7, 12) anlegen als Objekt b der Klasse Punkt2D
    
    //Punkt b verschieben um 2 Einheiten nach links
    //und 4 Einheiten nach unten mittels der Methode Verschiebe
    
    //Bestimme den Abstand zwischen den Punkten a und b mittels der
    //Methode BerechneAbstandZu und gib den berechneten Abstand aus
    
  }
}

class Punkt2D {
  private double x;
  private double y;

  public Punkt2D(double xwert, double ywert) {
    this.x = xwert;
    this.y = ywert;
  }

  public double GibXWert() {return x;}

  public double GibYWert() {return y;}

  public String Ausgeben() {return "(" + x + ", " + y + ")";}
```

```
  public void Verschiebe(double diffx, double diffy) {
    //Verschiebt den als Attributwerte gespeicherten Punkt um
    //den Wert diffx nach links sowie um den Wert diffy nach unten
```

```
  }

  public double BerechneAbstandZu(Punkt2D p) {
    //Berechnet den Abstand zwischen dem als Attributwerte
    //gespeicherten Punkt und dem uebergebenen Punkt p.
    //Der Abstand zwischen zwei Punkten (x1, y1) und (x2, y2)
    //ist durch die Formel sqrt((x1 - x2)^2 + (y1 - y2)^2) definiert.
```



```
  }
}
```

**Aufgabe 6 (Lineare Liste)** **10 Punkte**

Mit der generischen Klasse `LinkedList` steht in Java ein vorgefertigtes Instrument zur Verwaltung dynamischer linearer Listen zur Verfügung. Als Elemente einer solchen Liste können Objekte einer beliebigen, frei wählbaren Klasse dienen, deren Klassenname (z.B. `Druckjob`) hinter `LinkedList` in spitzen Klammern angegeben wird. In der Klasse `LinkedList` gibt es u.a. die vordefinierten öffentlichen Methoden `add` zum Hinzufügen eines neuen Elementes am Listenende, `get` zur Bereitstellung des Elementes an der `i`-ten Position (Zählung beginnt mit 0) und `remove` zum Löschen des Elementes an der `i`-ten Position (Zählung beginnt mit 0). Die Methode `size` liefert die Anzahl Elemente in der Liste.

Implementieren Sie unter Nutzung der generischen Klasse `LinkedList` eine Warteschlange für Druckjobs. Der dazu benötigte Quellcode besteht aus drei Klassen:

- Schreiben Sie eine Klasse `Druckjob`, die als privates Attribut eine Jobnummer (integer-Zahl) enthält. Darüber hinaus besitzt die Klasse einen Konstruktor, dem eine Jobnummer übergeben wird. Er legt ein neues Objekt an und übernimmt die Jobnummer als Attributwert. Zusätzlich soll die Klasse eine öffentliche Methode `getJobnummer` haben, die die als Attribut hinterlegte Jobnummer zurückgibt.

```
class Druckjob {









} //class
```

- Vervollständigen Sie die Klasse `Warteschlange`, die mittels `LinkedList` eine lineare Liste von Druckjobs verwaltet. Als privates Attribut führt die Klasse `Warteschlange` diese lineare Liste als Objekt `jobs`.
  - Die Methode `IstLeer` liefert den booleschen Wert `true` zurück, wenn die Warteschlange mindestens ein Element besitzt, ansonsten `false`.
  - Die Methode `Anhaengen` übergibt einen neuen Druckjob an die Warteschlange. Dazu fügt sie den Druckjob am Ende der Warteschlange ein.
  - Die Methode `Drucken` entfernt den Druckjob am Anfang der Warteschlange aus der Liste, sofern die Warteschlange nicht leer ist.

```
class Warteschlange {
  private LinkedList<Druckjob> jobs;
```

```
  public Warteschlange() {jobs = new LinkedList<Druckjob>();}
```

```
} //class
```

- Schreiben Sie eine Klasse `MeineDruckerwarteschlange`, die lediglich die `main`-Methode enthält. Darin wird ein Objekt `tintenspritz` der Klasse `Warteschlange` angelegt, anschließend werden hintereinander zwei Druckjobs mit den Jobnummern `314` und `815` erzeugt und in die Warteschlange mittels `Anhaengen` eingestellt, danach erfolgt einmal der Aufruf zum Drucken, im Anschluss wird ein neuer Job mit der Nummer `4712` erzeugt und in die Warteschlange platziert und letztendlich erfolgen abschließend noch zwei Aufrufe zum Drucken.

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

```
  } //main
} //class
```

**Aufgabe 7 (Algorithmische Programmierübung)** **10 Punkte**

Schreiben Sie ein Java-Programm, das den Nutzer zunächst auffordert, 5 Zahlenwerte $a_0$ bis $a_4$ vom Typ `int` einzugeben. Anschließend soll das Programm den kleinsten der eingegebenen Zahlenwerte (Minimum) und den größten der eingegebenen Zahlenwerte (Maximum) bestimmen sowie das arithmetische Mittel $m = \frac{1}{5} \sum_{i=0}^{4} a_i$ und die Streuung $v = \frac{1}{5} \sum_{i=0}^{4} (a_i - m)^2$ berechnen. Die erhaltenen Ergebnisse für Minimum, Maximum, $m$ und $v$ sollen vom Programm geeignet ausgegeben werden. Beachten Sie, dass mögliche Nachkommaanteile von $m$ und $v$ berücksichtigt werden. Ihr Java-Programm soll ohne weitere Ergänzungen compilierbar und ausführbar sein.