Grundlagen

Aufgabe 1.1 (5 Punkte)
Kopieren Sie in der vorgegebenen Methode alle Werte des übergebenen Arrays `ary` in ein neues Array und liefern Sie diese Kopie zurück.

```
public class ArrayKopieren {

  public static int[] kopiereArray(int[] ary) {









  }

}
```

Aufgabe 1.2 (3 Punkte)
Kodieren Sie die Dezimalzahl 0.125 als *Float* in der binären Exponent-Mantisse-Darstellung, wie sie in der Vorlesung eingeführt wurde.

Aufgabe 1.3 (4 Punkte)
Der folgende logische Ausdruck soll auf seinen Wahrheitswert untersucht werden:

```
((A && !B) || (B && !A)) && (true || false)
```

Tragen Sie das Ergebnis der Auswertung des logischen Ausdrucks für die jeweilige Belegung von A und B ein.

	A ist `true`	A ist `false`
B ist `true`		
B ist `false`		

Aufgabe 1.4 (6 Punkte)
Betrachten Sie die Java-Klasse `Sichtbar` und geben Sie die Werte der Variablen `a` und `b` zu den Zeitpunkten `1 - 6` an. Falls eine oder mehrere der Variablen zu einem Zeitpunkt nicht definiert sind, so setzen Sie an der entsprechenden Stelle ein Minus (`-`) in die untenstehende Tabelle.

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

      /* Zeitpunkt 1 */

      int[] b = {1,1,2};

      /* Zeitpunkt 2 */

      int a = 1;

      a = a + methode1(b);

      /* Zeitpunkt 6 */
    }

    public static int methode1(int[] b) {

      /* Zeitpunkt 3 */

      int a = b[1];
      b[1] = 0;

      /* Zeitpunkt 4 */

      b[1] = ++a;
      b = new int[2];

      /* Zeitpunkt 5 */

      return 7;
    }
}
```

Zeitpunkt	1	2	3	4	5	6
`int a`						
`int[] b`						

Aufgabe 1.5 (6 Punkte)
Im Folgenden soll das Newton-Verfahren für die Funktionstypen $f(x) = x^2 - a$ implementiert werden. Schreiben Sie dazu eine *rekursive* Methode, welche bei Übergabe der Parametern `double a` und `int n` die folgende, rekursiv definierte Formel auswertet:

$$x_0 = a$$
$$x_n = x_{n-1} - \frac{1 - \frac{a}{(x_{n-1})^2}}{\frac{2 \cdot a}{(x_{n-1})^3}}$$

Wird die Methode mit den Werten a und n aufgerufen, soll sie x_n zurückliefern. Verzichten Sie auf eine Klassendefinition und Fehlerbehandlung.

Laufzeiten, Verifikation, Terminierung

Aufgabe 2.1 (3 Punkte)
Geben Sie auf die folgenden Theoriefragen eine kurze und prägnante Antwort:

1. Ein Algorithmus liege in der Laufzeitklasse $O(n^2)$. Nennen Sie eine Laufzeitklasse in der er ebenfalls liegt.

2. Ein Algorithmus löse ein Problem in konstanter Laufzeit, wenn das Problem die Größe 1 oder 0 hat. Ansonsten *halbiert* er die Problemgröße in *konstanter Laufzeit* und löst das kleinere Problem rekursiv. In welcher Laufzeitklasse liegt der Algorithmus? (2 Punkte)

Aufgabe 2.2 (6 Punkte)
Geben Sie zu den folgenden drei Methoden in O-Notation an, in welchen Laufzeitklassen sie in Abhängigkeit des Parameters $n > 0$ liegen. Geben Sie hierbei immer die *kleinste* Laufzeitklasse an.

```
public static int a(int n) {
  int j = n;
  int i = 0;

  while(j > 0){
    j = j/2;
    i++;
  }

  return i;
}
```

Laufzeitklasse (2 Punkte):

```
public static int b(int n) {
  int j = 0;

  for(int i = 0; i < n; i++) {
    i = n;
    j++;
  }

  return j * n;
}
```

Laufzeitklasse (2 Punkte):

```
public static int[] c(int n) {
  int[] a = new int[n];

  for(int i = 0; i < n; i++){
    a[i] = i;
  }

  QuickSort.sort(a);

  return a;
}
```

Laufzeitklasse (2 Punkte):

Aufgabe 2.3 (5 Punkte)
Begründen Sie warum der durch die Methode `konv()` gegebene Algorithmus *terminiert*.

```
public static double konv() {
  double EPSILON = 1.0 / 1024.0;
  double delta = 1.0;
  double sum = 0.0;
  int k = 1;

  while(delta > EPSILON) {
    double value = 1.0;
    for(int i = 0; i < k; i++) {
      value *= (1.0 / 2.0);
    }

    k++;
    delta = (sum + value) - sum;
    sum += value;
  }

  return sum;
}
```

Suchen und Sortieren

Aufgabe 3.1 (2 Punkte)

1. Wie groß ist die Laufzeit im *Best Case*, um ein Element in einer Hashtabelle zu finden?

2. Wie groß ist die Laufzeit im *Worst Case*, um ein Element in einem `SuchBaum` zu finden?

Aufgabe 3.2 (5 Punkte)

Schreiben Sie eine Methode `numberCount(Baum baum, int number)`, die in dem übergebenen Baum `baum` *rekursiv* die Anzahl der Knoten mit Wert `number` zählt und diese Anzahl zurückgibt. Gehen Sie davon aus, dass der Baum mit *Integer*-Objekten gefüllt ist.

```
public static int numberCount(Baum baum, int number) {



}
```

Aufgabe 3.3 (6 Punkte)
Sortieren Sie die Zahlenfolge

```
7   10   8   1   4   6
```

nach der Methode des in der Vorlesung behandelten **HeapSort**. Zeichnen Sie dazu zunächst den initialen binären Baum und diesen anschließend jeweils nach dem Ende eines *sift*-Vorgangs bzw. nach dem Ende der `sift`-Methode.

Aufgabe 3.4 (4 Punkte)
Wie groß sind die asymptotischen Laufzeiten der folgenden Sortieralgorithmen in den jeweiligen *Cases*?

	Best Case	*Average Case*	*Worst Case*
SelectionSort			
MergeSort			
QuickSort			

Objektorientierung

Aufgabe 4.1 (11 Punkte)
Gegeben seien die folgenden Java-Klassen (die `AlgoTools` seien bereits importiert). Beantworten Sie hierzu die Fragen auf nächsten Seiten.

```
public class Alpha {

  public String s;
  private int a = 0;

  public Alpha(String s){
    this.s = s;                    // Stelle 1
  }

  public String toString(){
    return "Alpha: " + s;
  }
}
```

```
public class Beta extends Alpha {

  public Beta(String s){
    super(s);                      // Stelle 2
  }

  public String getS(){
    return s;
  }

  public String toString(){
    return "Beta: " + s;
  }
}
```

```
public class Test{

  public static void main(String[] argv){

    Alpha a1 = new Alpha("Alpha 1");          // Stelle 3
    Alpha a3 = new Beta("Beta 1");            // Stelle 4

    IO.println(a1.toString());                // Stelle 5
    IO.println(a3.toString());                // Stelle 6
    IO.println(a3.getS());                    // Stelle 7

    a1.a = 2                                  // Stelle 8
  }
}
```

Die folgenden Teilaufgaben beziehen sich auf die drei Klassen `Alpha`, `Beta` und `Test`. Beantworten Sie jede Frage mit **maximal drei** prägnanten Sätzen.

a) Erläutern Sie Funktion und Notwendigkeit des Schlüsselwortes `this` an Stelle 1. (2 Punkte)

b) Erläutern Sie die Funktion des Schlüsselwortes `super` an Stelle 2. (2 Punkte)

c) Erläutern Sie das Verhalten von Java an den Stellen 3 und 4. Unterscheiden Sie zwischen den Begriffen Referenz und Instanz. Warum funktioniert die Instanziierung an Stelle 4? (2 Punkte)

d) Welche Konsolenausgabe liefern die Methodenaufrufe an den Stellen 5 und 6? (2 Punkte)

e) Erläutern Sie, warum der Aufruf an Stelle 7 nicht funktioniert. Wie kann man Java dazu zwingen, den Code zu übersetzen *ohne* die Instanziierung der Objekte zu verändern? (2 Punkte)

f) Warum funktioniert die Anweisung an Stelle 8 nicht? (1 Punkt)

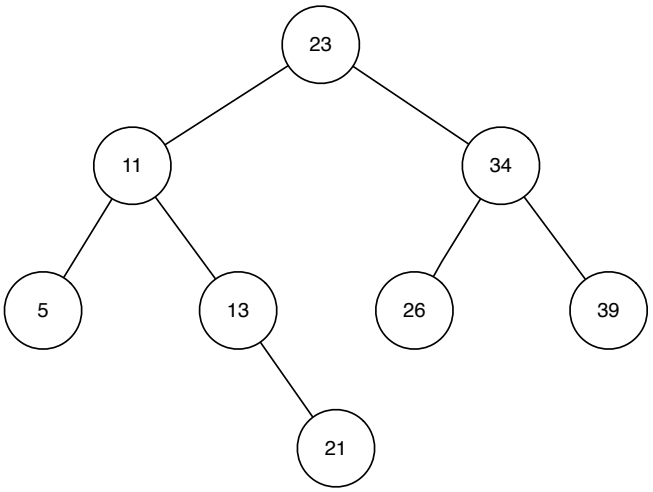
Abstrakte Datentypen

Aufgabe 5.1 (3 Punkte)
Gegeben seien die folgenden Postorder- und Inorder-Traversierungen:

```
Postorder: 18 14 32 42 21 9 12 3 16
Inorder:   32 14 18 16 42 3 21 12 9
```

Prüfen Sie, ob sich mit diesen Informationen *ein eindeutiger* binärer Baum wiederherstellen lässt. Falls ja, zeichnen Sie den binären Baum, falls nein *begründen* Sie kurz, warum kein *eindeutiger* Baum zugrunde liegt.

Aufgabe 5.2 (6 Punkte)
Gegeben sei folgender AVL-Baum:



a) Fügen Sie den Schlüssel `17` ein, indem Sie den Knoten direkt in den obigen AVL-Baum einzeichnen. Versehen Sie den kompletten Baum mit Balancewerten. (1 Punkt)

Führen Sie eine eventuell notwendige Rotation aus und zeichnen Sie den reorganisierten Baum erneut. Vermerken Sie, welche Rotation Sie verwendet haben. (2 Punkte)

b) Fügen Sie in den neu gezeichneten Baum aus Aufgabenteil a) (in einer neuen Zeichnung unten) den Schlüssel 22 ein und versehen Sie den kompletten Baum mit Balancewerten. (1 Punkt)

Führen Sie eine eventuell notwendige Rotation aus und zeichnen Sie den reorganisierten Baum erneut. Vermerken Sie die Art der Rotation, die Sie verwendet haben. (2 Punkte)

Aufgabe 5.3 (9 Punkte)
Implementieren Sie den abstrakten Datentypen `Schlange` mit Hilfe der `VerweisListe` aus der Vorlesung. Eine `ListenSchlange` speichert alle Elemente in einer `VerweisListe` als unterliegende Datenstruktur.

Aufgabe 5.4 (3 Punkte)
a) Wie heißt das Prinzip, nach dem der ADT `Keller` arbeitet?

b) Welche Traversierung führt in einem `SuchBaum` zu einer Ausgabe der Knoten in aufsteigender Reihenfolge?

c) Durch welchen Knoten wird ein gelöschter Vater mit zwei Söhnen in einem `SuchBaum` ersetzt?

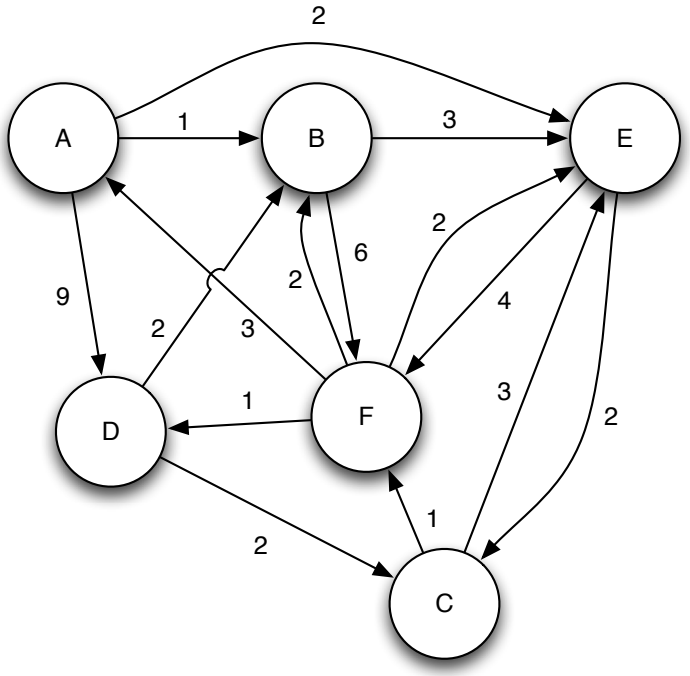
Graphenalgorithmen

Aufgabe 6.1 (2 Punkte)
Geben Sie auf die folgenden Theoriefragen eine kurze und prägnante Antwort:

1. Mit welchem Algorithmus lässt sich ein *minimaler Spannbaum* berechnen?

2. Was ist ein Matching in einem Graphen?

Aufgabe 6.2 (8 Punkte)
Gegeben sei der folgende gerichtete, gewichtete Graph $G = (V, E)$:



a) Stellen Sie G mit Hilfe einer *Adjazenz-Matrix* dar. Benutzen Sie die folgende Vorlage (2 Punkte):

	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

b) Berechnen Sie mit dem *Algorithmus von Dijkstra* kürzeste Wege von Knoten A zu allen anderen Knoten in G. Notieren Sie deutlich die errechneten Gesamtkosten für den jeweils günstigsten Weg zu einem Knoten und machen Sie beispielsweise durch Markierung der Kanten klar, welche Kanten zu den gefundenen kürzesten Wegen gehören. (Hinweis: Sollten Sie dabei einen Fehler machen, weisen Sie deutlich darauf hin und wählen Sie für Ihre Korrektur eine Darstellung der Form `H: I -> J -> K -> H (42)`.) (6 Punkte)

Aufgabe 6.3 (3 Punkte)
Geben Sie für den folgenden Graphen eine *topologische Sortierung* an.