- **AUFGABE 1 (6 Punkte) Theorie.**

1. (2 Punkte) Welchen Aufwand haben das *Suchen* sowie das *Einfügen* eines Elementes bei
   - einem sortierten, statischen Array
   - einer sortierten, dynamisch verketteten Liste
   - einem sortierten, balanzierten Baum?

2. (2 Punkte) Was versteht man unter *dynamischem Binden* und wie hängt es mit *Vererbung* zusammen?

3. (2 Punkte) Erläutern Sie das *Model–View–Control* Paradigma.

• **AUFGABE 2 (8 Punkte) JAVA.**

1. (2 Punkte) Was ist eine Referenz in *JAVA*? Erläutern Sie die beiden verschiedenen Arten von *Gleichheit*, die für Referenzen definiert sind.

2. (1 Punkt) Erläutern Sie, wie durch die Schlüsselworte *protected*, *public* und *private* der Zugriff auf Klasen-Interna geregelt wird.

3. (2 Punkte) Ersetzen Sie folgendes Code-Fragment durch äquivalente Fallunterscheidung(en) ohne Verwendung der `switch`-Anweisung. (Dabei seien `foo` und `bar` bereits definiert.)

```
int i;
switch (i) {
    case 0,1 : foo(0); break;
    case 2   : bar(2);
    default  : foo(-1); break;
}
```

4. (3 Punkte) Wo sind die Fehler im folgenden *JAVA*-Code? Beschreiben Sie die Fehler und geben Sie die Zeilenzahlen des Auftretens an.

```
 1 class Foo {
 2     private int count;
 3     Foo() {
 4         count = 0;
 5     }
 6     void foo() {
 7         super();
 8         count++;
 9     }
10 }
11
12 class Bar extends Foo {
13     NewBar(int count) {
14         this.count = count;
15     }
16 }
```

• AUFGABE 3 (7 Punkte) Numerik.

Für $\alpha \in \mathcal{R}$ und $|x| < 1$ berechnet sich die *binomischeReihe* wie folgt:

$$(1+x)^{\alpha} = \sum_{n=0}^{\infty} \binom{\alpha}{n} x^n$$

Dabei ist der *Binomialkoeffizient* definiert als:

$$\binom{\alpha}{n} = \prod_{k=1}^{n} \frac{\alpha - k + 1}{k}$$

1. (2 Punkte) Schreiben Sie eine *JAVA*-Methode `double binkoef( double alpha, double n )`, welche den Binomialkoeffizienten $\binom{\alpha}{n}$ berechnet.

2. (4 Punkte) Schreiben Sie eine *JAVA*-Methode `double binom( double alpha, double x, int stel )`, die den Wert der binomischen Reihe für gegebenes α und `x` auf `stel` Stellen hinter dem Komma per Approximation berechnet.

   Beenden Sie den Approximationsprozeß, wenn eine Genauigkeit von $10^{-\text{stel}}$ erreicht ist. Formulieren Sie diese Bedingung mit Hilfe einer zu erstellenden Hilfsfunktion `boolean close( double x, double y, int stel )`.

   Hinweis: Für die Potenzfunktion können Sie die Methode `Math.pow` verwenden, zur Berechnung der Summe in jedem Schritt die Gleichung

   $$S_n = S_{n-1} + \binom{\alpha}{n} x^n$$

   Bevor Sie mit dem Schreiben beginnen, lesen Sie bitte die nächste Teilaufgabe ...

3. (1 Punkt) Sehen Sie für Ihre Methode `binom` – bei unveränderten Argument- und Resultattypen – auch eine Fehlerbehandlung vor, wenn das Argument `x` nicht im Bereich $-1 < x < 1$ liegt. Verwenden Sie hierzu die Ausnahme `IllegalArgumentException(String message)`.

• AUFGABE 4 (9 Punkte) Abstrakte Datentypen.

Bei der Bearbeitung dieser Aufgabe können Sie voraussetzen, daß Ihnen die Klasse `List` mit den folgenden Methoden zur Verfügung steht:

- `List()` (leere Liste)
- `void insert(Object o)` (Einfügen vor aktueller Position)
- `Object currentValue()` (aktuelles Element)
- `void remove()` (aktuelles Element entfernen)
- `void reset()` (zum Listenanfang)
- `void advance()` (zum nächsten Element)
- `boolean finished()` (Listenende erreicht, d.h. hinter letztem Element?)
- `boolean isempty()` (Liste leer?)

Mit Hilfe obiger Listenklasse sollen Sie eine Klasse `SortedList` implementieren, welche eine sortierte Liste mit Zugriffsoperationen auf das kleinste sowie das größte Element realisiert. In der `SortedList` sollen Element vom Typ `Object` gespeichert werden, die Ordnung der Elemente erfolgt über einen Schlüssel `int key`.

1. (2 Punkte) Schreiben Sie eine Klasse `SortedList`, die eine sortierte Liste von beliebigen Objekten verwaltet, deren Reihenfolge jeweils durch einen Schlüssel vom Typ `int` gegeben ist. Dabei sollen Sie zur Listenverwaltung die Operationen aus der Klasse `List` verwenden.

   Hinweis: Da Sie in der Klasse `List` nur Elemente vom Typ `Object` speichern können, müssen Sie entweder

   - eine Wrapper-Klasse schreiben, um die Datenelemente und den Schlüssel zusammen verwalten zu können, oder
   - für die Schlüssel eine zweite Liste anlegen, wobei Sie den Schlüsselwert mittels `Integer( int i )` zu einem Objekt der Klasse `Integer` machen müssen, dessen Wert Sie mit `int Integer.intValue()` auspacken können.

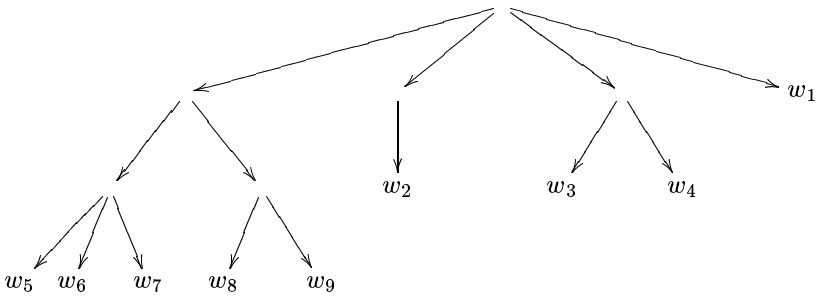
2. (3 Punkte) Ergänzen Sie die Klasse um die Methode `void insert(Object o, int key)`, welche ein Element `o` an der gemäß `key` richtigen Stelle in der Liste einfügt.

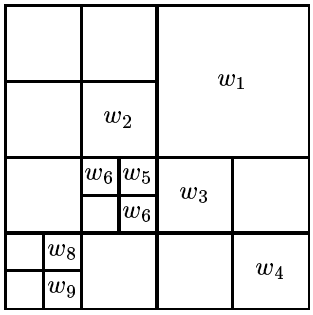
3. (4 Punkte) Ergänzen Sie die Klasse um die beiden Methoden `Object min()` und `Object max()`, welche das kleinste bzw. größte Element (gemäß Schlüssel) zurückliefern und jeweils aus der Liste entfernen. Wenn die Operationen auf die leere Liste angewendet werden, soll `null` zurückgegeben werden.

• AUFGABE 5 (9 Punkte) QuadTrees.

Die Lage verschiedener Gewichte w_i in der Ebene läßt sich durch Aufteilung der Ebene in Teilfelder beschreiben, wobei die Gewichte jeweils in der Mitte der Felder liegen.

Dies kann mit Hilfe eines QuadTrees realisiert werden. Ein *QuadTree* ist ein Baum, bei dem jeder innere Knoten bis zu vier Unterbäume (NW, NO, SO, SW) für die einzelnen Teilfelder besitzt, und die Daten in den Blättern gespeichert sind.





1. (2 Punkte) Schreiben Sie eine Klasse `QuadTree` mit den Konstruktoren `QuadTree(double weight)` und `QuadTree(QuadTree nw, QuadTree no, QuadTree so, QuadTree sw)`.

2. (3 Punkte) Ergänzen Sie die Klasse um eine Methode `double totalWeight()`, welche das Gesamtgewicht $\sum_{i=1}^{N} w_i$ aller Elemente im Baum liefert.

3. (4 Punkte) Ergänzen Sie die Klasse um eine Methode `double xMedium()`, welche die x-Koordinate des Schwerpunktes liefert. Diese berechnet sich (bei N Blättern im Baum) als Summe der gewichteten x-Koordinaten, geteilt durch das Gesamtgewicht aller Elemente:

   $$x_s = \frac{\sum_{i=1}^{N} x_i * w_i}{\sum_{i=1}^{N} w_i}$$

   Dabei habe das Feld insgesamt die Ausdehung des Einheitsquadrates, d.h. $[0,1] \times [0,1]$. Für die linken Teilfelder NW und SW gilt somit $x \in [0, 0.5]$, für die rechten Teilfelder NO und SO $x \in [0.5, 1]$.

   Hinweis: Zur Lösung dieser Aufgabe benötigen Sie eine Hilfsmethode `double xMedium(double left, double right)`, die als zusätzliche Parameter die linke und rechte x-Koordinate des durch den Baum repräsentierten Feldes erhalten.

   Beispiele: Das Gewicht w_1 liegt somit an Position $x_1 = 0.5 + \frac{1}{2} * (1 - 0.5) = 0.75$, da das Feld NO sich im Bereich $x \in [0.5, 1]$ bewegt und das Gewicht in der Mitte liegt. Das Gewicht w_2 liegt an Position $x_2 = 0.25 + \frac{1}{2} * (0.5 - 0.25) = 0.375$, das es im rechten unteren Unterfeld des Feldes NW liegt.

• AUFGABE 6 (5 Punkte) Artikel-Preise.

Ein Händler bietet neben einfachen Waren zu Werbezwecken auch kompliziertere, zusammengesetzte Artikel an. So bekommt beispielsweise jeder Kunde, der 10 Waschmaschinen kauft, eine weitere geschenkt. Oder wer einen Fernseher gleichzeitig mit einem Videorekorder kauft, erhält 3 Prozent Rabatt, das entspricht einem Rabattfaktor von 0,03.

Für die Preisberechnung an der elektronischen Kasse ist es notwendig, den Preis der Artikel zu kennen. Dabei ergibt sich der Preis von zusammengesetzten Artikeln aus der Summe der einzelnen (einfachen oder wiederum zusammengesetzten) Artikel abzüglich eines artikelspezifischen Rabattfaktors.

1. (1 Punkt) Schreiben Sie eine abstrakte Klasse `Artikel`, welche die Methode `abstract double getPreis()` zur Bestimmung des Preises eines Artikels enthält.
2. (4 Punkte) Schreiben Sie zwei Klassen `EinArtikel` und `ZusArtikel`, die beide von der Klasse `Artikel` abgeleitet sind. Definieren Sie in der Klasse `EinArtikel` ein Attribut `double preis`, welches den Preis dieses Artikels verwaltet. Die Klasse `ZusArtikel` soll dagegen ein Array von Artikeln sowie den Rabattfaktor `double rabatt` enthalten.

   Versehen Sie beide Klassen mit geeigneten Konstruktoren zur Initialisierung der Attribute. Implementieren Sie nun in beiden Klassen die Methode `double getPreis()`, so daß der Preis wie oben beschrieben berechnet wird.