Prof. Dr. J. Giesl
D. Cloerkes, S. Dollase, N. Lommen, D. Meier, F. Meyer

Tutoraufgabe 1 (Überblickswissen):

a) Was versteht man unter *Casting* und wie ist die Syntax dafür in Java?

b) Welche großen Vorteile bietet das Konzept der Vererbung in Java?

c) Viele Vorteile der Vererbung zwischen zwei Klassen `A` und `B1` lassen sich auch erzielen, ohne dass diese Klassen tatsächlich voneinander erben. So ist es beispielsweise möglich, die Klasse `B1` in eine Klasse `B2` zu überführen, welche sich genau wie `B1` verhält, ohne jedoch von `A` zu erben.

Wir passen die Klasse `B1` an, indem wir die `extends`-Klausel weglassen und stattdessen ein neues Attribut `a` vom Typ `A` in die Klasse `B2` einfügen und diesem Attribut ein neues Objekt der Klasse `A` zuweisen. Wenn in `B1` Attribute oder Methoden der Oberklasse `A` verwendet werden, müssen diese sich in `B2` nun auf das Attribut `a` beziehen.

Wir haben also die *Vererbungsbeziehung* zwischen `A` und `B1` durch eine *Nutzungsbeziehung* zwischen `A` und `B2` ersetzt. Was sind die Vor- und Nachteile dieser beiden Varianten?

```
class A {
    int i;
    void m() {}
}
```

```
class B1 extends A {
    void foo() {
        m();
        i += 1;
    }
}
```

```
class B2 {
    A a = new A();
    void foo() {
        a.m();
        a.i += 1;
    }
}
```

Tutoraufgabe 2 (Rekursive Datenstrukturen):

In dieser Aufgabe geht es um einfach verkettete Listen als Beispiel für eine dynamische Datenstruktur. Wir legen hier besonderen Wert darauf, dass eine einmal erzeugte Liste nicht mehr verändert werden kann. Achten Sie also in der Implementierung darauf, dass die Attribute der einzelnen Listen-Elemente `nur` im Konstruktor geschrieben werden.
Für diese Aufgabe benötigen Sie die Klasse `ListExercise.java`, welche Sie aus dem Moodle-Lernraum herunterladen können.
In der gesamten Aufgabe dürfen Sie **keine Schleifen** verwenden (die Verwendung von Rekursion ist hingegen erlaubt). Ergänzen Sie in Ihrer Lösung für alle öffentlichen Methoden außer Konstruktoren und Selektoren geeignete javadoc-Kommentare.

a) Erstellen Sie eine Klasse `List`, die eine einfach verkettete unveränderliche Liste als rekursive Datenstruktur realisiert. Die Klasse `List` muss dabei mindestens die folgenden öffentlichen Methoden und Attribute enthalten:

- `static final List EMPTY` ist die einzige `List`-Instanz, die die *leere* Liste repräsentiert
- `List(List n, int v)` erzeugt eine neue Liste, die mit dem Wert `v` beginnt, gefolgt von allen Elementen der Liste `n`
- `List getNext()` liefert die von `this` referenzierte Liste ohne ihr erstes Element zurück
- `int getValue()` liefert das erste Element der Liste zurück

b) Implementieren Sie in der Klasse `List` die öffentlichen Methoden `int length()` und `String toString()`. Die Methode `length` soll die Länge der Liste zurück liefern. Die Methode `toString` soll eine textuelle Repräsentation der Liste zurück liefern, wobei die Elemente der Liste durch Kommata separiert hintereinander stehen. Beispielsweise ist die textuelle Repräsentation der Liste mit den Elementen 2, 3 und 1 der `String` `"2, 3, 1"`.

c) Implementieren Sie in der Klasse `ListExercise` die öffentliche Methode `int skipSum`, welche eine Liste als Argument erhält. Die Methode `skipSum` soll dabei, beginnend mit dem ersten Element als Startelement, jedes zweite Element der Liste aufsummieren. Beispielsweise sollte für die Liste mit den Elementen 2, 3, 1 und 5 die Summe $2 + 1 = 3$ berechnet werden.

d) Ergänzen Sie die Klasse `List` darüber hinaus noch um eine öffentliche Methode `getSublist`, welche ein Argument i vom Typ `int` erhält und eine unveränderliche Liste zurückliefert, welche die ersten i Elemente der aktuellen Liste enthält. Sollte die aktuelle Liste nicht genügend Elemente besitzen, wird einfach eine Liste mit allen Elementen der aktuellen Liste zurückgegeben.

e) **Video**
Vervollständigen Sie die Methode `merge` in der Klasse `ListExercise.java`. Diese Methode erhält zwei Listen als Eingabe, von denen wir annehmen, dass diese bereits aufsteigend sortiert sind. Sie soll eine Liste zurückliefern, die alle Elemente der beiden übergebenen Listen in aufsteigender Reihenfolge enthält.

Hinweise:

- Verwenden Sie zwei Zeiger, die jeweils auf das kleinste noch nicht in die Ergebnisliste eingefügte Element in den Argumentlisten zeigen. Vergleichen Sie die beiden Elemente und fügen Sie das kleinere ein, wobei Sie den entsprechenden Zeiger ein Element weiter rücken. Sobald eine der Argumentlisten vollständig eingefügt ist, können die Elemente der anderen Liste ohne weitere Vergleiche hintereinander eingefügt werden.

f) **Video**
Vervollständigen Sie die Methode `mergesort` in der Klasse `ListExercise.java`. Diese Methode erhält eine unveränderliche Liste als Eingabe und soll eine Liste mit den gleichen Elementen in aufsteigender Reihenfolge zurückliefern. Falls die übergebene Liste weniger als zwei Elemente enthält, soll sie unverändert zurück geliefert werden. Ansonsten soll die übergebene Liste mit der vorgegebenen Methode `divide` in zwei kleinere Listen aufgespalten werden, welche dann mit `mergesort` sortiert und mit `merge` danach wieder zusammengefügt werden.

Hinweise:

- Sie können die ausführbare `main`-Methode verwenden, um das Verhalten Ihrer Implementierung zu überprüfen. Um beispielsweise die unveränderliche Liste mit den Elementen 2, 4 und 3 sortieren zu lassen, rufen Sie die `main`-Methode durch `java ListExercise 2 4 3` auf.

Tutoraufgabe 4 (Entwurf einer Klassenhierarchie):

In dieser Aufgabe sollen Sie einen Teil der Tierwelt modellieren.

- Ein Tier kann ein Säugetier, ein Wurm oder ein Insekt sein. Jedes Tier hat ein Alter.
- Ein spezielles Merkmal der Säugetiere ist ihr Fell. Für jedes Säugetier ist somit die Anzahl der Haare pro Quadratzentimeter Haut bekannt.
- Verschiedene Wurmarten haben im Allgemeinen wenig gemeinsam. Jeder Wurm hat jedoch eine bekannte Länge in Zentimetern.
- Alle Insekten haben einen Chitinpanzer. Bekannt ist, wie viel Druck in Pascal der Chitinpanzer eines Insektes aushalten kann.
- Menschen sind Säugetiere. Sie sind der Meinung, dass Intelligenz eines ihrer besonderen Merkmale sei. Deswegen ist der IQ jedes Menschen bekannt.
- Bandwürmer sind Würmer. Sie haben die Angewohnheit, Menschen zu befallen. Ihr vielleicht wichtigstes Merkmal ist ihr Wirt, ein Mensch, ohne den sie nicht lange überleben können.
- Bienen und Ohrwürmer sind Insekten. Das heißt insbesondere, dass Ohrwürmer keine Würmer sind.
- Bienen stechen Säugetiere, wenn sie sich bedroht fühlen.
- Ohrwürmer verfügen über Zangen, deren Größe in Millimeter in der Welt der Ohrwürmer von großer Bedeutung ist. Folglich ist die Zangengröße jedes Ohrwurms bekannt. Außerdem verwend(et)en Menschen Ohrwürmer als Medizin zur Behandlung von Erkrankungen der Ohren eines Menschen.

Entwerfen Sie unter Berücksichtigung der Prinzipien der Datenkapselung eine geeignete Klassenhierarchie für die oben aufgelisteten Arten von Tieren. Notieren Sie keine Konstruktoren. Um Schreibarbeit zu sparen, brauchen Sie keine Selektoren anzugeben. Achten Sie darauf, dass gemeinsame Merkmale in Oberklassen zusammengefasst werden. Ergänzen Sie außerdem geeignete Methoden, um die Behandlung von Ohrenerkrankungen, den Bandwurm-Befall und den Bienenstich abzubilden.

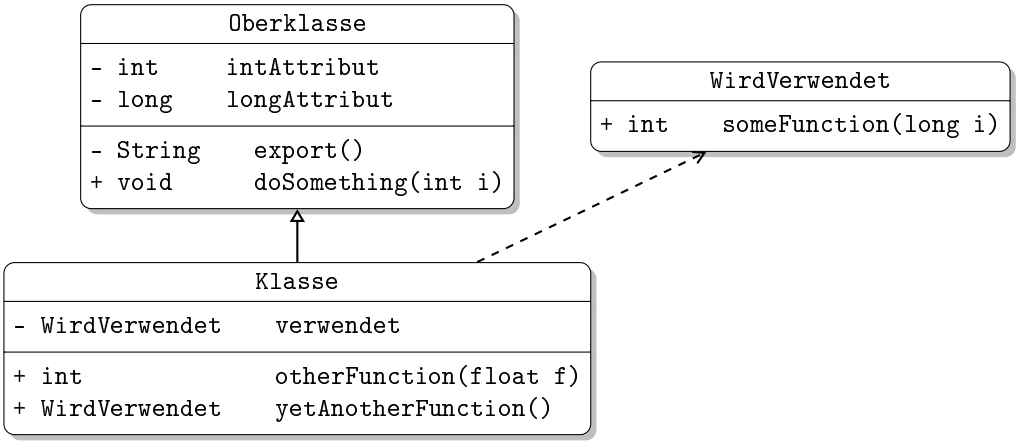

Abbildung 1: Graphische Notation zur Darstellung von Klassen.

Verwenden Sie hierbei die Notation aus Abb. 1. Eine Klasse wird hier durch einen Kasten beschrieben, in dem der Name der Klasse sowie Attribute und Methoden in einzelnen Abschnitten beschrieben werden. Weiterhin bedeutet der Pfeil $B \twoheadrightarrow A$, dass A die Oberklasse von B ist (also `class B extends A`) und A `->` B, dass A den Typ B in den Typen seiner Attribute oder in den Ein- oder Ausgabeparametern seiner Methoden verwendet. Benutzen Sie ein `-` um `private` und ein `+` um `public` abzukürzen.
Tragen Sie keine vordefinierten Klassen (`String`, etc.) oder Pfeile dorthin in Ihr Diagramm ein.