Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Jens-Peter Redlich
M. Kabierski, A. Heuts, Dr. W. Müller, M. Sänger

Berlin, den 01.02.2021

Übungen zur Vorlesung
Grundlagen der Programmierung

Übungsblatt 6

Abgabe: Die Abgabe Ihrer Lösungen erfolgt über den Moodle-Kurs der Vorlesung (https://moodle.hu-berlin.de/course/view.php?id=98193). Nutzen Sie die dort im Abschnitt Übung angegebene **Aufgabe 6.** Die Abgabe ist bis zum **15.02.2021** um **09:15 Uhr** möglich. **Die Abgabe erfolgt in Gruppen.** Bitte bilden Sie im Moodle Gruppen! Bitte verwenden Sie **keine** (noch nicht in der Vorlesung eingeführten) **Java-Bibliotheken**.

Hinweise:

- Achten Sie darauf, dass Ihre Java-Programmdateien mittels des UTF-8-Formats (ohne byte order marker (BOM)) codiert sind und keinerlei Formatierungen enthalten.
- Verzichten Sie auf eigene Packages bei der Erstellung Ihrer Lösungen, d. h. **kein** `package`-Statement am Beginn Ihrer Java-Dateien.
- Quelltextkommentare können die Korrektoren beim Verständnis Ihrer Lösung (insbesondere bei inkorrekten Abgaben) unterstützen; sind aber nicht verpflichtend.
- Testen Sie Ihre Lösung bitte auf einem Rechner aus dem Informatik Computer-Pool z. B. gruenau6.informatik.hu-berlin.de (siehe Praktikumsordnung: Referenzplattform)

Aufgabe 1 (Zahlensysteme) 4 Punkte

Wandeln Sie die folgenden Zahlen jeweils in die vorgegebene Zahlendarstellungen um:

1) Basis 10 (Dezimal) ➔ Basis 17

2021_{10}

2) Basis 9 ➔ Basis 3

2021_{9}

3) Basis 10 (Dezimal) mit Vorzeichen ➔ **14-Bit Binärzahl im Zweierkomplement**

-2021_{10}

4) 3-Byte hexadezimalen Zweierkomplement ➔ **Dezimalzahl mit Vorzeichen**

$FFF81E_{16}$

Geben Sie Ihre Lösungen als **UTF8-Textdatei A1.txt** ab. Ihre Antworten sollen dabei wie im folgenden Beispiel formatiert sein:

```
1: 42_17
2: 12_3
3: 0111001010101_2
4: +42
```

Verwenden Sie gern auch die beigefügte Vorlage *A1.txt* für die Erstellung Ihrer Lösungsdatei.

Aufgabe 2 (Bindung) 6 Punkte

Vollziehen Sie den folgenden Java-Quelltext nach. Überlegen Sie, welche Ausgaben ein Aufruf der `PolyTest`-Main-Methode generiert und warum. Nehmen Sie dabei an, dass jede Zeile der Main-Methode ausgeführt wird, auch wenn zuvor ein Fehler aufgetreten sein sollte.

```
01 class Basis {
02   protected String s = "basis";
03   protected String t = "basisT";

04   public String s() { return s; }
05   public String t() { return t; }
06 }

07 class Ableitung extends Basis {
08   protected String s = "ableitung";
09   protected String t = "ableitungT";

10   public Ableitung() { }
11   public String s() { return s; }
12 }

13 public class PolyTest {

14   static void out(Object o) { System.out.println(o); }

15   public static void main(String[] args) {
16      Ableitung a = new Ableitung();
17      out( a.s );
18      Basis b = a;
19      out( b.s );
20      out( a.s() );
21      out( b.s() );
22      out( a.t() );
23      out( b.t() );
24   }
25 }
```

Wählen Sie aus den folgenden Multiple-Choice-Fragen zu den Zeilen 17, 19, 20, 21, 22, 23 die zutreffenden Antworten aus. Es können pro Frage mehrere Antworten richtig sein. Für jede korrekte Teilfrage gibt es einen Punkt. Für jede falsche Antwort wird ein halber Minuspunkt berechnet. Die Beantwortung einer Teilfrage kann jedoch nicht negativ werden.

Zeile 17:

a. Ein Zugriff auf eine protected-Variable ist aus diesem Kontext nicht zulässig.
b. Der Zugriff auf die String-Variable geschieht durch statische Bindung.
c. Der Zugriff auf die String-Variable geschieht durch dynamische Bindung.
d. Die Konsolen-Ausgabe dieser Zeile ist „ableitung“.
e. Die Konsolen-Ausgabe dieser Zeile ist „basis“.

Zeile 19:

a. Ein Zugriff auf eine protected-Variable ist aus diesem Kontext nicht zulässig.
b. Der Zugriff auf die String-Variable geschieht durch statische Bindung.
c. Der Zugriff auf die String-Variable geschieht durch dynamische Bindung.
d. Die Konsolen-Ausgabe dieser Zeile ist „ableitung“.
e. Die Konsolen-Ausgabe dieser Zeile ist „basis“.

Zeile 20:

a. Der Zugriff auf die `s()`-Methode geschieht ausschließlich durch statische Bindung.
b. Der Zugriff auf die `s()`-Methode geschieht durch dynamische Bindung.
c. Die Konsolen-Ausgabe dieser Zeile ist „ableitung“.
d. Die Konsolen-Ausgabe dieser Zeile ist „basis“.

Zeile 21:

a. Der Zugriff auf die `s()`-Methode geschieht ausschließlich durch statische Bindung.
b. Der Zugriff auf die `s()`-Methode geschieht durch dynamische Bindung.
c. Die Konsolen-Ausgabe dieser Zeile ist „ableitung“.
d. Die Konsolen-Ausgabe dieser Zeile ist „basis“.

Zeile 22:

a. Der Zugriff auf die `t()`-Methode geschieht ausschließlich durch statische Bindung.
b. Der Zugriff auf die `t()`-Methode geschieht durch dynamische Bindung.
c. Die Konsolen-Ausgabe dieser Zeile ist „ableitungT“.
d. Die Konsolen-Ausgabe dieser Zeile ist „basisT“.

Zeile 23:

a. Der Zugriff auf die `t()`-Methode geschieht ausschließlich durch statische Bindung.
b. Der Zugriff auf die `t()`-Methode geschieht durch dynamische Bindung.
c. Die Konsolen-Ausgabe dieser Zeile ist „ableitungT“.
d. Die Konsolen-Ausgabe dieser Zeile ist „basisT“.

Nutzen Sie zur Angabe Ihrer gewählten Antwortmöglichkeiten bitte das Programm `Aufgabe2.java`, welches via Moodle bereitgestellt wird. Dieses enthält sechs Methoden für jede Teilaufgabe. Jede Methode wiederum enthält vier bzw. fünf Variablen `a`, `b`, `c`, `d / a`, `b`, `c`, `d`, `e` vom Typ `boolean`, welche initial auf `true` gesetzt wurden. Ändern Sie jeweils die Werte dieser Variablen, sodass diese mit Ihren gewählten Antwortmöglichkeiten für die Teilaufgabe übereinstimmen. Testen und überprüfen Sie Ihre Abgabe mit der `main`-Methode, in welcher ihre gewählten Antworten noch einmal ausgegeben werden. Geben Sie in Moodle die Datei `Aufgabe2.java` mit Ihren gewählten Antworten ab.

Aufgabe 3 (Kühlschrank) 10 Punkte

Gegeben sei die folgende abstrakte Basisklasse `Lebensmittel`:

```
public abstract class Lebensmittel {

	protected String name;
	protected int menge; // in Milliliter bzw. Gramm

	public Lebensmittel(String name, int menge) {
		this.name = name;
		this.menge = menge;
	}

	public abstract boolean essen();

	public abstract boolean trinken();

	public String status() {
		return "Lebensmittel";
	}
}
```

Schreiben Sie die sechs zusätzlichen Klassen `Getraenk`, `Speise`, `Wasser`, `Mate`, `Brot` und `Kaese` für die gilt:

- Jedes `Getraenk` und jede `Speise` ist ein Lebensmittel.
- Jedes `Wasser` und jede `Mate` ist ein Getraenk.
- Jedes `Brot` und jede `Kaese` ist eine Speise.
- Für jede `Speise` soll eine Methode `essen(int menge)` und für jedes Getraenk eine Methode `trinken(int menge)` implementiert werden.
- Die Klassen `Wasser` und `Mate` sollen beim Aufruf der Methode `essen()` stets `false` zurückgeben. Beim Aufruf von `trinken()` wird die Menge von Wasser um 200ml und die von Mate um 100ml reduziert.
- Die Klassen `Kaese` und `Brot` sollen beim Aufruf der Methode `trinken()` stets `false` zurückgeben. Beim Aufruf von `essen()` wird die Menge von Käse um 20g und die von Brot um 50g reduziert.
- Kein `Lebensmittel` kann eine Menge von weniger als 0ml oder 0g haben. Sollte die zu konsumierende Menge beim Aufruf von `essen()` oder `trinken()` größer sein als die verfügbare Menge, wird 0 als neue Menge festgelegt und `false` zurückgegeben. Dementsprechend kann es kein `Lebensmittel` mit einer Menge weniger als 0 geben.
- Der einzige explizite Konstruktor der Klasse `Mate` soll nur den Namen als Parameter erhalten, die Menge ist stets 500ml.

- Der einzige explizite Konstruktor der Klasse `Brot` soll statt des Namens eine Nummer (Integer) als Parameter erhalten mit folgender Bedeutung:
    - 0 – Weißbrot,
    - 1 – Schwarzbrot,
    - 2 – Mischbrot,
    - sonst: Spezialbrot.

  Nach dem Aufruf des Konstruktors hat jedes Brot den entsprechenden Namen.
- Die Methode `status()` soll stets den Namen der Klasse und die Instanzvariable `name` und `menge` zurückgeben. Die Ausgabe der Menge soll dabei jeweils mit der entsprechenden Einheit (g oder ml) erfolgen. Beispielsweise soll eine Instanz der Klasse `Mate` mit dem Namen „Club Mate“ und einer Menge von 500ml folgenden String zurückgeben:

  `"Mate (Club Mate, 500 ml)"`

Versuchen Sie die Klassendefinitionen möglichst klein zu halten: Benutzen Sie so oft wie möglich Funktionalität übergeordneter Klassen bzw. implementieren Sie Funktionalität in der Klassenhierarchie so weit oben wie möglich. Beachten Sie darüber hinaus folgende Hinweis zur Abgabe:

- Kommentieren sie ihre Klassen und Methoden wie gewohnt.
- Achten Sie bei der Abgabe darauf eventuelle Package-Namen zu entfernen.
- Achten Sie auf die korrekte Kodierung ihrer Dateien im UTF-8 Format.
- Sie dürfen die Basisklasse `Lebensmittel` editieren und bei Bedarf zusätzliche Funktionalitäten hinzufügen.

Benutzen Sie zum Testen Ihrer Implementation die folgende Klasse `Kuehlschrank`. Die Ausgabe von `java Kuehlschrank` sollte wie folgt aussehen:

```
Mate (Club Mate, 500 ml)
Brot (Schwarzbrot, 750 g)
Wasser (Sprudel, 250 ml)
Mate (Flora Power, 500 ml)
Brot (Spezialbrot, 1000 g)
Kaese (Parmesan, 200 g)
Wasser (Still, 1500 ml)
Trinken: Mate (Club Mate, 400 ml)
Essen: Brot (Schwarzbrot, 700 g)
Trinken: Wasser (Sprudel, 50 ml)
Trinken: Mate (Flora Power, 400 ml)
Essen: Brot (Spezialbrot, 950 g)
Essen: Kaese (Parmesan, 180 g)
Trinken: Wasser (Still, 1300 ml)
```

Packen Sie ihre sieben Implementierungen `Lebensmittel.java`, `Speise.java`, `Getraenk.java`, `Wasser.java`, `Mate.java`, `Kaese.java` und `Brot.java` in ein ZIP-Archiv mit dem Namen **A3.zip**, welches ausschließlich diese sieben Dateien enthält. Achten Sie darauf, dass das Archiv **keine Unterverzeichnisse** enthält.

Die Datei muss sich mit dem Kommandozeilenprogramm `unzip` auf dem Institutsrechner `gruenau6.informatik.hu-berlin.de` entpacken lassen. Ansonsten wird Ihre Abgabe mit 0 Punkten bewertet!

Viel Spaß und Erfolg beim Lösen der Aufgaben.