University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Programmierung für Mathematik (HS13)

Übung 12

Diese Übung beinhaltet die Aufgaben aus der letztjährigen Abschlussprüfung (HS12).

Aufgabe 1: Codeverständnis

Notieren Sie, was der folgende Code ausgeben würde, wenn er so in einer `main`-Methode stehen würde:

```
1 String x = "A";
2 String y = "a".toUpperCase();
3 System.out.println(x == y);
```

Ausgabe:

```
1 double x = 10;
2 float y = 10.9f;
3 System.out.println(x == (int) y);
```

Ausgabe:

```
1 int n = 6;
2 while(n < 10) {
3   System.out.println(n--);
4   n = n + 3;
5 }
```

Ausgabe:

```
1  int sum = 0;
2  int[] a = new int[]{1,2,3,4,5,6,7,8,9};
3  for (int i = 0; i < a.length; i++) {
4    if (a[i] % 2 == 0) {
5      sum += a[i];
6    }
7  }
8  System.out.println(sum);
```

Ausgabe:

```
1  int[] a = new int[]{1,2,3,4,5};
2  System.out.println(a.length - a[a[2]]);
```

Ausgabe:

```
1  boolean a = false;
2  boolean b = !a;
3  if (a || b) {
4    if (a && a) {
5      System.out.println("a");
6    } else if (a || true) {
7      System.out.println("b");
8    }
9  } else {
10   System.out.println("c");
11 }
```

Ausgabe:

Aufgabe 2: Algorithmen

1. *Abundante Zahlen* sind Zahlen deren echte Teilersumme grösser ist als die Zahl selbst. 12 ist beispielsweise eine abundante Zahl, weil die Summer ihrer echten Teiler grösser als 12 ist:

   $$1 + 2 + 3 + 4 + 6 = 16$$

   Schreiben Sie eine Methode die alle abundanten Zahlen bis zu einer oberen Grenzen *(upperBound)* auf die Kommandozeile ausgibt.

   Ihre Lösung:

```
public void printAbundantNumbers(int upperBound){
```

2. Schreiben Sie eine **rekursive** Methode, die die Summe aller natürlichen Zahlen von 1 bis zu einem Parameter n zurückliefert.

   Ihre Lösung:

3. Schreiben Sie eine Methode die das kleinste Element in einem Array zurückliefert:

   Ihre Lösung:

```
public int getSmallestValue(int[] array){
```

Aufgabe 3: Design

1. Schreiben Sie eine Klasse `Point`, welche einen Punkt in einem 2-dimensional-en Koordinatensystem darstellen soll. Die Koordinaten des Punktes sollen mit einem Konstruktor definiert werden können. Implementieren Sie auch einen Konstruktor der keine Parameter erwartet und einen Punkt im Ursprung des Koordinatensystems instanziert. Fügen Sie Ihrer Klasse eine Methode hinzu, die überprüft ob ein Punkt der als Parameter übergeben wird, sich an der selben Position im Koordinatensystem befindet.
   Ihre Lösung:

2. Fügen Sie Ihrer Klasse eine Methode hinzu, die überprüft ob der als Parameter übergebene Punkt sich im selben Quadranten befindet.
   Ihre Lösung:

3. Implementieren Sie eine Klasse `Line` die eine Gerade darstellen soll. Die Gerade wird durch 2 Punkte definiert. Verwenden Sie dafür ihre Klasse aus Aufgabe a). Fügen Sie Ihrer Klasse eine Methode hinzu, die als Parameter eine andere Gerade erwartet und überprüft, ob die beiden Geraden sich schneiden:
   Ihre Lösung:

```
public boolean hasIntersection(Line otherLine){
```

Aufgabe 4: Adressverwaltung

1. Bilden Sie folgenden Sachverhalt auf Klassen, Attribute und Methoden ab. Verwenden Sie wenn möglich für Klassen-, Attribut-, und Methodennamen die Begriffe, die in Klammern vorgegeben sind.

   Ein Komilitone hat sie gebeten eine Applikation zu entwickeln, die ihm dabei hilft seine Kontakte zu verwalten. Jeder Kontakt *(Contact)* hat einen Namen *(name)*, ein Alter *(age)* und eine Telefonnummer *(number)*. Zusätzlich soll für jeden Kontakt auch gespeichert werden, ob es sich um einen Mitarbeiter *(coWorker)* handelt. Fügen Sie der *Contact*-Klasse eine Methode *printContact* hinzu, die den Namen, das Alter, die Telefonnummer, sowie eine Information ob es sich bei diesem Kontakt um einen Mitarbeiter handelt, auf die Konsole ausgibt.
   Die Kontakte werden in einem Adressbuch *(Directory)* verwaltet. Es soll möglich sein, dem Adressbuch Kontakte hinzuzufügen und auch wieder zu entfernen.

2. Fügen Sie Ihrem Adressbuch eine Methode hinzu, die den Kontakt mit dem höchsten Alter zurückliefert. Sie können davon ausgehen, dass jeder Kontakt im Adressbuch ein anderes Alter hat.

   Ihre Lösung:

Aufgabe 5: Vererbung (Bonusaufgabe)

Bilden Sie folgenden Sachverhalt auf ein Interface, sowie auf Klassen, Attribute und Methoden ab. Verwenden Sie für Klassen-, Attribut- und Methodennamen die Begriffe, die in Klammern vorgegeben sind:

Alle Angestellen *(Employee)* einer Firma haben einen Vornamen *(firstName)* und einen Nachnamen *(lastName)*. Kundenbetreuer *(AccountManager)* sind spezielle Angestellte, die zusätzlich noch eine bestimmte Anzahl von Kunden *(clients)* haben. Ausserdem gibt es in der Firma noch Vorgesetzte *(Boss)*, die auch Angestellte sind.

Einmal im Monat erhalten alle Angestellten eine Lohnabrechnung. Wenn sie eine Lohnabrechnung erhalten, geben die Angestellten eine Meldung der folgenden Art auf dem Bildschirm aus: "[VORNAME] [NACHNAME] hat den Lohn in der Höhe von [LOHNSUMME] Franken erhalten." Um die Lohnsumme zu berechnen, werden folgende Regeln angewandt:

- Angestellte: Basislohn
- Kundenbetreuer: Basislohn + 500 pro betreutem Kunde
- Vorgesetze: Basislohn + 15% Bonus

Schreiben Sie einen *TestDriver* und instanzieren Sie je einen Angestellten, einen Kundenbetreuer und einen Vorgesetzten. Lassen Sie allen Personen einen Basislohn von 10'000 ausbezahlen.

Überschreiben Sie die *toString*-Methode in der Klasse *Employee*, so dass der Vorname und Nachname zurückgegeben wird.

Ihre Lösung:

(Fortsetzung Ihrer Lösung zur Aufgabe 5)