Selbststudium OOP3

Auftrag

Kapitel 2.7

1. Was ist ein Header? Was ist ein Body?

   Header:

   Der Header ist das "WAS" – Zugriffsmodifizierer und die Signatur

   ```
   public Account(String newOwner, int newPin)
   ```

   Body:

   Der Body ist das „WIE" – beinhaltet Deklarationen und die eigentliche implementation.

   Alles im Block darunter:

   ```
   {
      [...]
   }
   ```

2. Geben Sie die Methodensignaturen der TicketMachine (Code 2.1) (gemäss der Definition in SW 3) an.

   ```
   TicketMachine(int ticketCost)
   getPrice()
   getBalance()
   insertMoney(int amount)
   printTicket()
   ```

3. Wo können Anweisungen und Deklaration stehen?

   In der Klasse und den einzelnen Methoden darin.

4. Was ist ein Block?

   ```
   { […] }
   ```

5. Wie viele return Anweisungen finden Sie in Code 2.1?

   Zwei (bei getPrice() und getBalance())

6. zu bearbeitende Aufgaben: 2.23 bis 2.29

   2.23:

   getPrice() liefert den Wert von price zurück.

   getBalance() liefert den Wert von balance zurück.

   2.24:

   Wie viel wurde bereits eingeworfen?

   2.25:

   Nein, bei „return" wird der Rückgabewert zurück gegeben, egal ob der Methodenname geändert wurde.

   2.26:

   ```
   public int getTotal()
   {
      return total;
   }
   ```

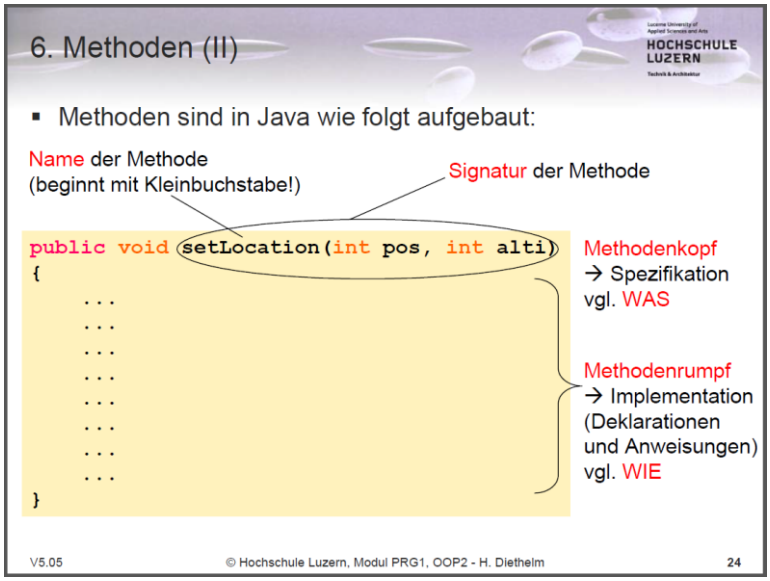
   2.27:

   ```
   missing return statement
   ```

   2.28:

   Abgesehen vom Namen der Methode sind beide Signaturen gleich.

   getBalance() hätte jedoch einen Rückgabewert (int), printTicket() besitzt keinen Rückgabewert (void).



2.29:

Nein, sie haben keinen Rückgabewert.

Gemäss Beschreibung liefern sie ebenfalls keinen Rückgabewert und ändern nur die Instanz-Variablen.

Kapitel 2.8

7. Was bedeutet der return-Typ void?

Es gibt keinen Rückgabewert.

8. Auf S. 35 ist in der Fussnote der compound assignment operator beschrieben.
Es gilt: a += b entspricht a = a+b.
Füllen Sie analog dazu folgende Tabelle aus:

compound assignment	assignment
a += b	a = a +b
a -= b	*a = a - b*
a *= b	*a = a * b*
a /= b	*a = a / b*

9. Im Code der TicketMachine (Code 2.1) gibt es zwei Stellen, an denen Sie den „compound assignment operator“ verwenden können. Finden Sie diese beiden Stellen.

```
insertMoney(int amount)
{
   balance += amount;
}


public void printTicket()
{
    // Update the total collected with the balance.
    total += balance;
}
```

10. zu bearbeitende Aufgaben: 2.31 bis 2.35

2.31:

Anhand des Rückgabewertes, in diesem Fall keiner (void)

2.32:

```
public void setPrice(int cost)
{
   Price = cost;
}
```

2.33:

```
public void increase(int points)
{
   score += points;
}
```

2.34:

Ja, der „score“ wird durch die Methode verändert.

2.35:

```
public void discount(int amount)
{
   price -= amount;
}
```

Kapitel 2.9

11. zu bearbeitende Aufgaben: 2.38 bis 2.42

2.38:

Es wird der String „price“ ausgegeben und nicht der Wert welcher in der Variable „price“ gespeichert ist.

```
# price cents.
```

2.39:

Das gleiche wie bei 2.38

```
# price cents.
```

2.40:

Nein, es wird immer der String „price“ ausgegeben und nicht der Wert der Variable.

2.41:

```
    /**
     * Print ticket price
     */
    public void showPrice()
    {
        System.out.println("The price of a ticket is " + price + " cents.");
    }
```

2.42:

```
The price of a ticket is 5 cents.
The price of a ticket is 15 cents.
```

Der Preis ist in einer Instanzvariable gespeichert, die können je nach Instanz unterschiedlich sein.

Kapitel 2.13

12. Beschreiben Sie das „conditional statement“ in pseudo-code auf S. 42 unten in Deutsch. Übersetzen Sie hierzu alle Beschreibungen ausser den zwei Worten „if“ und „else“ in Deutsch.

```
if (führe Tests aus, welche true oder false als Resultat haben) {
  falls das Resultat true war, führe diese Statements aus
}
else {
  falls das Resultat false war, führe diese Statements aus
 }
```

13. zu bearbeitende Aufgaben: 2.46 und 2.47

2.46:

```
Use a positive amount: -5
Use a positive amount: 0
```

Die Balance wird nur geändert wenn ein Wert > 0 eingegeben wird.

2.47:

Bei 0 wird nun keinen Error mehr ausgegeben. Die Balance ändert sich natürlich dadurch nicht. (x + 0 = x)

Kapitel 2.14

14. zu bearbeitende Aufgaben: 2.50.

```
public void printTicket()
{
    if(balance >= price) {
        // Simulate the printing of a ticket.
        System.out.println("##################");
        System.out.println("# The BlueJ Line");
        System.out.println("# Ticket");
        System.out.println("# " + price + " cents.");
        System.out.println("##################");
        System.out.println();

        // Update the total collected with the price.
        total = total + price;
        // Reduce the balance by the prince.
        balance = balance - price;
    }
    else {
        System.out.println("You must insert at least: " +
                           (price - balance) + " more cents.");

    }
}
```

Es wird nur ein Ticket gedruckt wenn genügend Geld eingeworfen wurde. Andernfalls wird eine Meldung ausgegeben wie viel noch fehlt.

Der total sowie balance Wert wird nun richtig gerechnet und nicht einfach auf 0 gesetzt.

Kapitel 2.16

15. zu bearbeitende Aufgaben: 2.58 und 2.59

2.58:

Die Variable balance wird auf 0 gesetzt und danach als Returnwert verwendet, dieser ist somit immer 0.

2.59:

```
Compiler Error: unreachable statement
```

Bei einem Return wird die Methode sofort beendet, allfälig folgende Befehle würden nie ausgeführt.

16. Unter Pitfall auf S. 48 steht eine sehr wichtige Information. Übersetzen Sie den ersten Satz als Merksatz ins Deutsche.

Wenn eine lokale Variblae den gleichen Namen hat, wie eine Instanzvariable, dann kann nicht auf die Instanzvariable zugegriffen werden.

Kapitel 2.17

17. Füllen Sie die Tabelle auf der folgenden Seite als Zusammenfassung aus

	Field / Attribut	**formaler Parameter**	**lokale Variable**
kann Werte speichern? (ja/nein)	*Ja*	*Nein*	*Ja*
Wo wird er/sie/es definiert?	*Class*	*Signatur*	*Methode*
Wie lange existiert er/sie/es? Wie ist die Lebensdauer?	*Solange es die Klasse gibt.*	*Während des Aufrufes der Methode*	*Während des Aufrufes der Methode*
Von wo kann auf ihn/sie/es zugegriffen werden? Wie ist die Sichtbarkeit?	*Je nach AccessModifier, überall*	*Innerhalb der Methode*	*Innerhalb der Methode, nachdem sie deklariert wurde*

Programmierübung

Schreiben Sie eine Klasse `FormalAddress`, welche je nach Alter und Geschlecht die richtige Anrede erzeugen kann.

Die Klasse besitzt Attribute für Vor- und Nachname als Zeichenkette, das Geburtsjahr als ganze Zahl (z.B. 1990) und das Geschlecht (männlich oder weiblich) als Boolean.

Diese Attribute werden im Konstruktor mit den übergebenen Werten initialisiert.

Die Klasse stellt die Methode `createFormalFormOfAddress(…)` zur Verfügung, welcher als aktueller Parameter das heutige Jahr übergeben wird.

Je nach Alter (muss berechnet werden), gibt diese Methode drei unterschiedliche Zeichenketten zurück:

- Alter unter oder gleich 16 Jahre:
  `Hello «Vorname»`
- Alter grösser 16 Jahre und männlich:
  `Dear Mr. «Nachname»`
- Alter grösser 16 Jahre und weiblich:
  `Dear Ms. «Nachname»`

Testen Sie Ihre Klasse in BlueJ, indem Sie Objekte erstellen, die Methode aufrufen und den Rückgabewert überprüfen.

```
/**
 * PRG1 OOP3 FormalAddress Programmieruebung
 *
 * @author Felix Rohrer (felix.rohrer@stud.hslu.ch)
 * @version 2013-03-04
 */
public class FormalAddress
{
    // instance variables
    private String preName;
    private String surName;
    private int birthYear;
    private boolean isFemale;

    /**
     * Constructor for objects of class FormalAddress.
     *
     * @param preName Vorname
     * @param surname Nachname
     * @param birthYear Geburtsjahr
     * @param isFemale Weiblich
     */
    public FormalAddress(String preName, String surName, int birthYear, boolean isFemale)
    {
        // initialise instance variables
        this.preName = preName;
        this.surName = surName;
        this.birthYear = birthYear;
        this.isFemale = isFemale;
    }

    /**
     * creates the corresponding greetings
     *
     * @param  currentYear   aktuelles Jahr
     * @return Begruessungstext
     */
    public String createFormalFormOfAddress(int currentYear)
    {
        // calculate age - we don't check if the age is valid
        int age = currentYear - this.birthYear;
        // enum greeting
        if (age <= 16) {
            return ("Hello " + this.preName);
        } else {
            if (this.isFemale) {
                return ("Dear Ms. " + this.surName);
            } else {
                return ("Dear Mr. " + this.surName);
            }
        }
    }
}
```