Lucerne University of Applied Sciences and Arts

Programming and Algorithms

Personal Documentation

Ervin Mazlagić

Horw, autumn 2013

Contents

1 Preface

This is a personal documentation and notebook for the first course in programming at the Lucerne University of Applied Sciences and Arts. The goal of this document is to collect useful informations and nice snippets of code out of the course.

This document shall not be provided as a official or unofficial cheatsheet for the course exam or similar. instance

2 Objects and classes

2.1 Summary exercises

Exercise 1.31

What are the types of the following values?

0	short, char, byte, int, long
"hello"	String
101	short, char, byte, int, long
-1	int, char, byte, int, long
true	boolean
"33"	String
3.1415	float, double

Exercise 1.32

What would you have to do to add a new filed, for example one called name, to a circle object?

```
private String name;
```

Exercise 1.33

Write the signature of a method named send that has one parameter of type String, and does not return a value.

```
public void send(String foo)
```

Exercise 1.34

Write a signature for a method named average that has two parameters, both of type int, and returns an int value.

```
public int average(int foo, int bar)
```

Exercise 1.35

Look at the book you are reading right now. Is it an object or class? If it is a class, name some objects. If it is an object, name its class.

The book is definitely an object, because it's a specific thing and in no way generic. The class could have a name like `SchoolBook`, `CodingBook` or just `Book`.

Exercise 1.36

Can an object have several different classes? Discuss.

No it can't.

3 Understanding class definitions

3.1 Start with Eclipse

In the first chapter we've worked with the BlueJ IDE but now I want to check Java-Coding with a common and popular Java-IDE like Eclipse To get the BlueJ-Projects work with Eclipse there are some things that have to be done.

1. Create a new project in Eclipse.
2. Import the source (BlueJ example-code).
3. Add a package-name to the source.
4. Create a main (replaces all interaction which were invoked by hand).

Listing 1: TicketMachine

```
1
2 package foobar;
3
4 public class TicketMachine
5 {
6     // The price of a ticket from this machine.
7     private int price;
8     // The amount of money entered by a customer so far.
9     private int balance;
```

Listing 2: Main (TicketMachine)

```
1  package foobar;
2
3  public class Main
4  {
5      public static void main(String[] args)
6      {
7          TicketMachine tm1;
8          tm1 = new TicketMachine(300);
9
10         tm1.insertMoney(200);
11
12         System.out.println("Balance: "+tm1.getBalance());
13
14         tm1.insertMoney(100);
15
16         tm1.printTicket();
17     }
18 }
```

3.2 Chapter Exercises

Exercise 2.21

Suppose that the class Pet has a field called name that is of type String. Write an assignment statement in the body of the following constructor so that the name field will be initialized with the value of the constructor's parameter.

```
1 public Pet(String petsName)
2 {
3     name = petsName;
4 }
```

Exercise 2.22 (challenge)

The following object creation will result in the constructor of the Date class being called. Can you write the constructor's header?

```
new Date("March", 23, 1861)
```

Try to give meaningful names to the paramenters.

```
1 public Date(String month, int day, int year)
2 {
3     ...
4 }
```

3.3 Selfstudy-Questions OOP2

Exercise 4

A class is build by three essential components. What are they?

- Instance variables (member variables, attributes)
- constructor
- methods

Exercise 5

What is the order of the three components?

The order doesn't matter technically but there is a common convention:

1. instance variables
2. constructor
3. methods

Exercise 6

What's their purpose?

instance variables are holding data of an object. All of this data together builds the object's state.

constructor is a special method that initializes objects.

methods are sequences which are defining the object's behaviour and characteristics.

Exercise 8

What is a variable?

A variable (or field) is a data storage inside an object that can be used for persistent data storage (limited by the lifetime of the object).

Exercise 9

What are the synonyms to instance variables?

- member variable
- attribute
- filed
- variable

Exercise 10

What do you think where the term instance variable comes from?

An instance is a realisation of an class by an object. The expression variable is well defined an known in computer science and if a variable explicitly belongs to an object, so it's clear that this is a variable of an instance or instance variable.

Exercise 11

How can you put comments into a Java-Code?

There are different ways to add comments in a Java source file without having trouble with the compiler.

- Use the single line comment by double slash.

```
// this method return the speed
private void getSpeed()
```

- Use the multiline comment by slash-dot

```
/**
 * This is a method that will return the
 * actual speed of the monstetruck that
 * is driven by the crazy clown IT .
 */
private void getSpeed()
```

Exercise 12 (important)

With which access-modification do you declare instance variables usually? Is it **`private`** *or* **`public`***? Do you have a reason for your answer?*

Usually we declare instance variables as private. The reason for this is a common pattern that is used to get or set these data form outside the objects by so called accessor and mutator methods (getSpeed, setSpeed, changeSpeed).

Exercise 13

Explain the relation between a constructor and the state of an onject.

The constructor is creating (initializing) an object and has nothing to do with the state of the object once it's set up.

Exercise 14

How do we name constructors?

Constructors are usually named after the class their used for.

Exercise 15

What's the lifetyme of instance variables, how long are they reachable/accessable?

The lifetime of variables is coupled to the lifetime of their objects. As long as the object is alive the variables are also alive.

Exercise 16

Why sould you (if possible) initialise instance variables explicit?

If we don't initialize variables explicit the compiler will use default values for the initialization. By explicit initialisation we don't have any disadwantage and it serves well to document what is sctually happening.

Exercise 17

What's the defualt value which is given to a `int` *variavle by its initialisation?*

The default value for an `int` is zero.

Exercise 19

What's the use of parameters?

Parameters provide additional information to a method or object. This is useful in many ways.

Exercise 20

What's the difference between a formal and a actual parameter?

A formal parameter is a parameter that is defined as parameter but has no actual value corresponding. A actual parameter is a parameter with a specific value.

Exercise 21

Is the following statement correct; "formal parameters are special variables"?

Parameters are temporary and restricted variables because their space is allocated by a call to the method or object and as soon as a value is transmitted to it. Once that call has completed its task, the formal parameter disappears and the values in it are lost.

Exercise 22

What's about the accessability of formal parameters?

The accessability of parameters are limited to the lifetime of the task which is creating them (method). Also parameter are only reachable from inside the box that they are used in (like a local variable).

Exercise 23

In which way this differs from instance variables?

Instance variables have a lifetime that is identical with the lifetime of their objects. Also parameters are only reachable from inside the block, instance variables are reachable from everywhere inside the class.

Exercise 24

How do the lifecycles of formal parameters and instance variables differ?

Instance variables are persistent (limited by lifetime of the object) and the lifetime of formal parameters is not really defined in runtime.

Exercise 26

How would you translate the expressions "assignment" and "expression" in german?

- assignment = Zuweisung
- expression = Ausdruck

Exercise 27

How does an assignment-instruction work exactly? What's about to be aware of in relation to data types?

An assignment can be done with the operator "=". For example:

```
// create a instance variable for speed
private int speed;

// set the speed
public void setSpeed(int newSpeed)
{
    speed = newSpeed;
}
```

By assigning data you have to be aware of data types. For example you can't assign a **int** to a **float** and so on. There are some strategies to "cast" or "parse" data between different data types but that's not our topic now.

3.4 Team Exercise 1-4

Create a Balloon-Class and create some objects and interact with them.

../workspace/balloon/src/flight/Balloon.java

```java
package flight;

/**
 * Balloon models a simple abstraction of a physical balloon.
 */

public class Balloon
{
    // size of the balloon. The balloon is abstracted a perfect
    // bowl defined by its diameter.
    private float diameter;

    // horizontal position of the balloon
    private int posHorizontal;

    // altitude (vertical position) of the balloon
    private int posVertical;

    // color of the balloon
    private String color;

    // number of the ballon
    private int number;

    // simple constructor
    public Balloon()
    {
        diameter = 300f;
        posHorizontal = 300;
        posVertical = 300;
        color = "red";
    }

    // more detailed constructor
    public Balloon(String newColor)
    {
        color = newColor;
    }

    public void setPosition(int newHorizontal, int newVertical)
    {
        posHorizontal = newHorizontal;
        posVertical = newVertical;
    }

    public void setDiameter(float newDiameter)
    {
        diameter = newDiameter;
    }

```

```
51      public void setColor(String newColor)
52      {
53          color = newColor;
54      }
55
56      public void setNumber(int newNumber)
57      {
58          number = newNumber;
59      }
60
61      public int getHorizontal()
62      {
63          return posHorizontal;
64      }
65
66      public int getVertical()
67      {
68          return posVertical;
69      }
70
71      public float getDiameter()
72      {
73          return diameter;
74      }
75
76      public String getColor()
77      {
78          return color;
79      }
80
81      public int getNumber()
82      {
83          return number;
84      }
85  }
```

../workspace/balloon/src/flight/Main.java

```
1   package flight;
2
3   public class Main
4   {
5       public static void main(String[] args)
6       {
7           // create a new balloon (with the simple constructor)
8           Balloon b1 = new Balloon();
9           // get the current horizontal position
10          System.out.println("Horizontal: " + b1.getHorizontal());
11          // set a new horizontal position
12          b1.setPosition(400, 400);
13          // get the current horizontal position
14          System.out.println("Horizontal: " + b1.getHorizontal());
15
16          // create a new ballon with the detailed constructor
17          Balloon b2 = new Balloon("yellow");
```

```
18          // get the color of the new ballon
19          System.out.println("Color: " + b2.getColor());
20
21
22      }
23  }
```

3.5 Team Exercise 5

You want to write records, so you have to write a class Book for this. This class shall have the following four attributes:

- Title (String)
- Author (String)
- Price (float)
- Year on buy (int)

The class shall also have two constructors.

- Title and author are parameters. The books is not bought yet and this is why the price is 0.0 and the "year of buy" is -1.
- All attributes are initialized by parameters.

The class shall have the following methods.

- Two methods to get the title and author.
- A method to get and to set the year of buy.
- A method to get and to set the price.

../workspace/Book/src/library/Book.java

```
1  package library;
2
3  public class Book
4  {
5      // title of the book
6      private String title;
7
8      // author of the book
9      private String author;
10
11     // price of the book
12     private float price;
13
14     // year of buy
15     private int year;
16
17     /**
18      * Create a new book with all attributes.
19      */
20     public Book(String newTitle, String newAuthor, float newPrice, int
           newYear)
21     {
22         title = newTitle;
23         author = newAuthor;
24         price = newPrice;
25         year = newYear;
26     }
27
28     public Book(String newTitle, String newAuthor)
```

```
29      {
30          title = newTitle;
31          author = newAuthor;
32          price = 0.0f;
33          year = -1;
34      }
35
36      public String getTitle()
37      {
38          System.out.println("Title: " + title);
39          return title;
40      }
41
42      public String getAuthor()
43      {
44          System.out.println("Author: " + author);
45          return author;
46      }
47
48      public void setYear(int newYear)
49      {
50          year = newYear;
51      }
52
53      public void setPrice(float newPrice)
54      {
55          price = newPrice;
56      }
57
58      public int getYear()
59      {
60          System.out.println("Year: " + year);
61          return year;
62      }
63
64      public float getPrice()
65      {
66          System.out.println("Price: " + price + " USD");
67          return price;
68      }
69
70  }
```

../workspace/Book/src/library/Main.java

```
1   package library;
2
3   public class Main
4   {
5       public static void main(String[] args)
6       {
7           Book b1 = new Book("Objects First with Java", "David Barnes",
                70.0f, 2013);
8           b1.getTitle();
9           b1.getAuthor();
```

```
10          b1.getYear();
11          b1.getPrice();
12      }
13  }
```

3.6 Team Exercise 5 - Optional

Think about banc accounts, their behaviour and attributes. Implement a class Account. To avoid round sum problems work with integer values. Play around with your class and get you some money!

../workspace/Account/src/money/Account.java

```
1  package money;
2
3  public class Account
4  {
5      private String ownerFirstName;
6      private String ownerLastName;
7      private String ownerAddress;
8      private String ownerEMail;
9      private int yearOfBirth;
10     private int yearOfAccount;
11     private int accountNumber;
12     private long accountBalance;
13     private long accountDebit;
14     private long accountCredit;
15     private boolean accountActive;
16
17     /**
18      * Create a new inactive account with default values.
19      */
20     public Account()
21     {
22         ownerFirstName = "Default";
23         ownerLastName = "Default";
24         ownerAddress = "Default";
25         ownerEMail = "Deafult";
26         yearOfBirth = -1;
27         yearOfAccount = -1;
28         accountNumber = -1;
29         accountBalance = 0;
30         accountDebit = 0;
31         accountCredit = 0;
32         accountActive = false;
33     }
34
35     /**
36      * Create a new active account.
37      */
38
39     public Account( String newFirstName,
40                     String newLastName,
41                     String newAddress,
42                     String newEMail,
43                     int newYearOfBirth,
44                     int newYearOfAccount,
45                     int newAccountNumber,
46                     long newAccountBalance)
47     {
48         ownerFirstName = newFirstName;
49         ownerLastName = newLastName;
```

```
50          ownerAddress = newAddress;
51          ownerEMail = newEMail;
52          yearOfBirth = newYearOfBirth;
53          yearOfAccount = newYearOfAccount;
54          accountNumber = newAccountNumber;
55          accountBalance = newAccountBalance;
56          accountDebit = 0;
57          accountCredit = 0;
58          accountActive = true;
59      }
60
61      public String getOwnerFirstName()
62      {
63          System.out.println("First name: " + ownerFirstName);
64          return ownerFirstName;
65      }
66
67      public void setOwnerFirstName(String newOwnerFirstName)
68      {
69          ownerFirstName = newOwnerFirstName;
70      }
71  }
```

../workspace/Account/src/money/Main.java

```
1   package money;
2
3   public class Main
4   {
5       public static void main(String[] args)
6       {
7           Account acc1 = new Account();
8           acc1.getOwnerFirstName();
9           acc1.setOwnerFirstName("David Barnes");
10          acc1.getOwnerFirstName();
11      }
12  }
```

3.7 Selfstudy-Questions OOP3

Exercise 1

What is a header? What is a body?

A header is a part of a method. For example `public int getSpeed()` is the header of the method

```
1   public int getSpeed()
2   {
3       return speed;
4   }
```

Exercise 2

Write down the signatures of the methods form class TicketMachine.

- getPrice(...)
- getBalance(...)
- insertMoney(int ...)
- printTicket(...)

Exercise 3

Where can you place expressions and definitions?

I don't understand that question.

Exercise 4

What is a block?

A block ist the part of a method which is between the curly braces.

Exercise 5

How many `return` *expressions do you find in Code 2.1?*

Exercise 7

What's the meaning of the return-type `void`*?*

The return type `void` indicates that the method has no return value.

Exercise 8

Fill out the table.

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

Exercise 9

In the code of the TicketMachine, there are two places where you can place a compound assignment operator. Find those two places.

Exercise 12

Describe the conditional operator of the pseudo-code on page 42 in german. Try to translate the code in german (except for the keywords `if` *and* `else`*.*

Exercise 16

At pitfall on page 48 is a very important information. Translate the first sentence in german.

Exercise 17

Fill out the following table.

	Field	**formal parameter**	**local variable**
can store values?	Yes	No	limited
Where is/are they defined?	class	class	method
How long do they exist?	permanent	imaginary	limited
From where can you access them?	global	nowhere	localy

3.8 Team Exercise 1

```
2.5 * (2+3) = 12.5
(int) 2.5 * 2 + 3 = 7
(int) 2.5 * (2+3) = 10
(int) (2.5 * 2 + 3) = 8
(int) (2.5 * (2+3)) = 12
(int) 2.5 * 2 + (float) 3 = 7.0
```

3.9 Team Exercises 2

- *What are konventions?*
  Konventions are rules that are not strict.
- *For whatr are they good for?*
  They give the programmer a good orientation if every coder used the same conventions or if a coder used a convention consistantly. This improves the portability and make the code easy to maintain.
- *Give the signatures for the following attributes by konventions:*
  - `String secondName`
  - `float hours`
  - `int personalNumber`
  - `Object myObject`

  Answer

  - `public String getSecondName()`
  - `public float getHours()`
  - `public int getPersonalNumber()`
  - `public Object getMyObject()`
- *Can you define a konvention for mutator-methods?*
  Of course you can, just as for getter-methods.

3.10 Team Exercise 3

At the exercises 2.27 and 2.59 you had to note error messages.

- *Compare your Notes*
  "Missing return statement" and "unreachable statment".
- *Try to define rules out of these error messages.*
  If you use a return declaration in the header you have to use a return expression and the return expression has to be at the end of the block.

3.11 Team Exercise 4

Now you'll get into the **switch** expression on your own. You can use the appendix D of your book and the file `Selection.jar` from ILIAS

3.11.1 Team Exercise 4.1

Look at the following snippet.

```
1  public void output(int value)
2  {
3      System.out.println();
4      System.out.println("actual parameter: " + value);
5      switch(value)
6      {
7          case 1:
8              System.out.println("one");
9              break;
10         case 2:
11             System.out.println("two");
12             break;
13         case 3:
14             System.out.println("three");
15             break;
16         default:
17             System.out.println("other value");
18             break;
19     }
20 }
```

3.11.2 Team Exercise 4.2

../workspace/Selection/src/choose/Selection.java

```
1  /* Copyright 2012 Hochschule Luzern - Technik & Architektur */
2
3  package choose;
4
5  /**
6   * Klasse Selection für die Lernaufgabe zu switch.
7   * @author Peter Sollberger
8   */
9  public class Selection
```

```
10 {
11
12     /**
13      * Der Konstruktor von Selection ist "leer".
14      */
15     public Selection()
16     {
17     }
18
19     /**
20      * In Abhängigkeit des übergebenen Wertes erfolgt die
21      * Ausgabe eines Textes.
22      */
23     public void output(int value)
24     {
25         System.out.println("aktueller Parameter: " + value);
26
27         switch (value)
28         {
29             case 1:
30                 System.out.println("eins");
31             case 2:
32                 System.out.println("zwei");
33             case 3:
34                 System.out.println("drei");
35             default:
36                 System.out.println("anderer Wert");
37         }
38     }
39 }
```

../workspace/Selection/src/choose/Main.java

```
1  package choose;
2
3  public class Main
4  {
5      public static void main(String[] args)
6      {
7          Selection mySel = new Selection();
8          mySel.output(5);
9          System.out.println("END OF PROGRAM");
10     }
11 }
```

3.11.3 Team Exercise 4.3

3.11.4 Team Exercise 4.4

3.11.5 Team Exercise 4.5

3.12 Summary exercises

Glossary

A

accessor

A accessor or accessor method is a method that provides access to information about an object's state (get-methods). 22

C

class

A class describes the kind of an object. This is done by giving instance variables and methods. The objects represents individual instatioations of the class. 22

constructor

A constructor is a special method in a class which is responsible to initialize objects properly. In difference to usual methods it has no return value and is only used once. 22

F

field

Fields store data for an object to use. Fields are also known as instance variables.. 22

H

header

A header is a part of a method. It is the part that is not only including the signature but the whole definition. Example: `public int getAge(String name)` is the header whereas `getAge(String )` is the signature. 22

I

instance

An instance is a realisation of a class to a real object, so instance is a synonym to object. 2, 22

M

method

A method is a action (function) of a specific class that can be invoked on an object of the given class. Objects usually do something when a method is invoked, so a good keyword to it would be *what*, as most methods are named by a verb. The methods give the objects their own particular and characteristic behavior. 22

O

object

An object is a instance of a class. 22

P

parameter

Addition information (data) given to a method or object is called parameter. 22

S

signature

The signature of a method is the part that identifies it to the compiler. For example the signature of `public setSpeed(int newSpeed, int newTolerance)` is not the whole head of the method but the name `setSpeed` and the list of parameter-types `int ..., int ...`. 22

state

A object or its status is represented by his state. The state is represented by the values in the fields (instance variables). 22

T

type

The type defines the kind of data or value (for example to a parameter, return value (see data types) or a variable. 22