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.

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).

../workspace/naive–ticket–machine/src/foobar/TicketMachine.java

```
/**
 * TicketMachine models a naive ticket machine that issues
 * flat-fare tickets.
 * The price of a ticket is specified via the constructor.
 * It is a naive machine in the sense that it trusts its users
 * to insert enough money before trying to print a ticket.
 * It also assumes that users enter sensible amounts.
 *
 * @author David J. Barnes and Michael Kölling
 * @version 2011.07.31
 */

package foobar;

public class TicketMachine
{
    // The price of a ticket from this machine.
    private int price;
    // The amount of money entered by a customer so far.
    private int balance;
    // The total amount of money collected by this machine.
    private int total;

    /**
     * Create a machine that issues tickets of the given price.
     * Note that the price must be greater than zero, and there
     * are no checks to ensure this.
     */
    public TicketMachine(int cost)
    {
        price = cost;
        balance = 0;
        total = 0;
    }

    /**
     * Return the price of a ticket.
     */
```

```
39      public int getPrice()
40      {
41          return price;
42      }
43
44      /**
45       * Return the amount of money already inserted for the
46       * next ticket.
47       */
48      public int getBalance()
49      {
50          return balance;
51
52      }
53
54      /**
55       * Receive an amount of money from a customer.
56       */
57      public void insertMoney(int amount)
58      {
59          balance = balance + amount;
60      }
61
62      /**
63       * Print a ticket.
64       * Update the total collected and
65       * reduce the balance to zero.
66       */
67      public void printTicket()
68      {
69          // Simulate the printing of a ticket.
70          System.out.println("##################");
71          System.out.println("# The BlueJ Line");
72          System.out.println("# Ticket");
73          System.out.println("# " + price + " cents.");
74          System.out.println("##################");
75          System.out.println();
76
77          // Update the total collected with the balance.
78          total = total + balance;
79          // Clear the balance.
80          balance = 0;
81      }
82  }
```

../workspace/naive–ticket–machine/src/foobar/Main.java

```
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 Selfstudy-Questions OOP2

Exercise 4

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

Exercise 5

What is the order of the three components?

Exercise 6

What's their purpose?

Exercise 8

What is a variable?

Exercise 9

What are the synonyms to instance variables?

Exercise 10

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

Exercise 11

How can you put comments into a Java-Code?

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?*

Exercise 13

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

Exercise 14

How do we name constructors?

Exercise 15

How long are the variables of an object alive (reachable)?

Exercise 16

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

Exercise 17

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

Exercise 19

What's the use of parameters?

Exercise 20

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

3.3 Summary exercises

Glossary

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. 8

F

field

Fields store data for an object to use. Fields are also known as instance variables.. 8

I

instance

An instance is a realisation of a class to a real object, so instance is a synonym to object. 8

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. 8

O

object

An object is a instance of a class. 8

P

parameter

Addition information (data) given to a method or object is called parameter. 8

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 ...`. 8

state

A object or its status is represented by his state. The state is represented by the values in the fields (instance variables). 8

T

type

The type defines the kind of data or value (for example to a parameter, return value (see data types) or a variable. 8