Programming 1

Christian Grévisse (christian.grevisse@uni.lu)
Gilles Neyens (gilles.neyens@uni.lu)

Lab 4 – Subprograms in Java

Part I – Practical Exercises on Object-Oriented Programming

Exercise 1 – Calculator

- Create a class `Calculator` with an attribute `value`.
- Add a constructor that allows you to set the initial value of the calculator through a parameter.
- Add another constructor, without parameter, that calls the first constructor to set the initial value to 0.
- Add a getter method `getValue` that returns the current value.
- Add the methods `add`, `subtract`, `multiply` and `divide`. Each one of them takes a parameter. The current value will be adapted appropriately. The `subtract` method shall reuse the implementation of the `add` method, while the `divide` method shall reuse the implementation of the `multiply` method. By *reuse*, we do not mean copying and pasting the former code, rather than calling the other method with an adapted argument. However, pay attention to illegal arguments!
- Write a main program that consists of a REPL (read-eval-print loop), i.e. the user is shown a menu of options (for the different operations of the calculator) he may choose from. He may also indicate the argument for the chosen operation. At each time, the new value of the calculator gets shown, its initial value being 0. Finally, the user shall also have the possibility to stop the execution of the program from the REPL.

Exercise 2 – Bank Account

1° Create a class `Person` with a single attribute `name`, a getter to retrieve its value and a constructor which sets its value.

2° Create a class `BankAccount` with attributes `holder` of type `Person` and `balance` of type `double`.

- Add a constructor to initialize their values as well as getters. The balance must at least be 0.
- Add a method `deposit` which adds a given amount to the balance. Make sure the amount to deposit is positive.
- Add a method `withdraw` which deducts a given amount from the balance, if funds are sufficient. Again, the amount to be deducted must be positive. The method returns a `boolean` value indicating the success or failure of the operation. In case of insufficient funds, an error message is shown.
- Add a method `printBalance` which shows the current balance on the console.
- Add a method `transfer` which takes as parameters an object of type `BankAccount` and an amount. The transfer to the beneficiary will only succeed if the amount can be withdrawn from the giver's account.

3° Write a main program where you create two accounts and test depositing, withdrawing and transferring money.

Exercise 3 – Date & Time

1° Create a class `Date` with attributes `day`, `month` and `year`.

- Add a method `isLeapYear` that returns whether the `year` is a leap year or not.
- Add a method `daysInMonth` that returns the number of days in the current `month`.
- The constructor of the class sets the initial values of the attributes based on parameters, which are checked upon their

validity and adapted if necessary.
  - Add a method `advance` that advances the current date by a day.
  - Add a method `format(boolean us, String delimiter)` which returns the formatted date as a `String`. Use leading zeros for days or months less than 10. The delimiter string delimits the 3 parts. If `us` is `true`, the month precedes the day.

2° Create a class `Time` with attributes `hours`, `minutes`, `seconds`.
  - The constructor of the class sets the initial values of the attributes based on parameters, which are checked upon their validity and adapted if necessary.
  - Add a method `tick` that advances the current time by a second (cf. Lab 2 - Exercise 7). The method returns a `boolean` value indicating whether a new day has begun. Note that hours here are always *stored* as values between 0 and 23.
  - Add a method `format(boolean us)` which returns the formatted time as a `String`. Use leading zeros for hours, minutes or seconds less than 10. The format is `hh:mm:ss`. If `us` is `true`, hours are *formatted* as values between 1 and 12 and a suffix (AM/PM) is added at the end.
  - Add the methods `secondsSinceMidnight` and `secondsUntilMidnight`, whose returned values should be self-explaining.

3° Create a class `DateTime` that has an attribute of type `Date` and another of type `Time`.
  - Add a method `tick` that advances the time by one second and advances, if necessary, the date by one day.
  - Add a method `print(boolean us, String delimiter)` that uses the `format` methods of `Date` and `Time` to print the current date and time on the console.

4° Write a main program to test the implementation of these classes.

Exercise 4 – Giveaways

Terri Aki owns 3 shops in Luxembourg. To celebrate the 50 years of existence of his business he decides to award his customers giveaways based on the price of the items they purchase. As his shops are spread out all over Luxembourg, he wants to ensure that customers from every region have a chance to win a giveaway. Write a program that helps Terri Aki to manage the giveaways in the different shops.

1° Create a class `Item` with the attribute `price` and a constructor which sets the initial value of the price. Also write a getter for this attribute.

2° Create a class `Shop` with the attribute `localNumberOfGiveaways` and the class attribute `maxNumberOfGiveaways`.
  - Add a constructor which sets the initial value of `localNumberOfGiveaways`.
  - Add a method `buy(Item item)` which:
    - prints the price of the purchased item
    - shows an appropriate message in the console if the shop itself has no more giveaways left, or if there are no more giveaways left at national level
    - randomly awards a giveaway based on the price of the item purchased if there are still giveaways left (in total and in the shop itself). There will be a 2% chance for items cheaper than 20 €, 5% for items between 20 € and 100 €, and 10 % for items which cost more than 100 €. It prints a message in the console if the customer won a giveaway. Don't forget to update the number of remaining giveaways

3° Write a main program that reads the total number of giveaways and creates 3 shops among which this number is equally distributed. It then simulates customers buying items between 0 € and 120 € (price randomly set) from the 3 shops as long as there are still giveaways left. Finally, it prints the number of giveaways left (per shop and at national level).

Programming 1

Christian Grévisse (christian.grevisse@uni.lu)
Gilles Neyens (gilles.neyens@uni.lu)

Lab 4 – Subprograms in Java

Part II – Theoretical Questions on Object-Oriented Programming

Name: ..... **Student Number:** .....

Exercise 5 – It's a kind of magic ...

In Lab 1 - Exercise 4, you first encountered the `Scanner` class that allows to read user input from the console in a Java program.

What may have seemed like a kind of magic back then were the following lines:

```
1 Scanner scanner = new Scanner(System.in);
2 int n = scanner.nextInt();
```

By now, this should be no more magic:

1° What is `Scanner`? .....

2° What is `scanner`? .....

3° What is the keyword `new` good for? .....

4° What is `System.in` in this context? .....

5° What is `nextInt()`? .....

6° How can you be sure that `scanner.nextInt()` returns an `int`? .....

Exercise 6 – Quick Quiz Questions

Fill in the missing words:

1° The .....statement in a method can be used to pass the value of an expression back to the calling method.

2° The keyword ..... indicates that a method does not return a value.

3° The keyword ..... requests memory from the system to store an object, then calls the corresponding ..... to initialize the object.

4° Each parameter must specify both a ..... and a .....

5° A variable known only within the method in which it is declared is a ..... variable.

6° A ..... variable represents class-wide information that is shared by all the instances of that class.

Exercise 7 – True or false?

State whether the following statements are true or false. If false, explain why.

1° Empty parentheses following a method name in a method declaration indicate that the method does not require any parameters to perform its task. .....

2° A method can be invoked on a primitive-type variable. .....

3° Variables declared in the body of a particular method are known as *instance variables* and can be used in all methods of the class. .....

4° Any class containing `public static void main(String[] args)` can be used to execute a Java application. .....

Exercise 8 – Variables in the wild

Consider the following piece of code:

```
1  public class Employee {
2
3    static int countEmployees = 0;
4
5    int employeeID;
6    String name;
7    double salary;
8
9    public Employee(String name, double salary) {
10     this.employeeID = ++countEmployees;
11     this.name = name;
12     this.salary = salary;
13   }
14
15   public double calculateSalaryGrowth(int years) {
16     double annualGrowth = 0.005;
17     double futureSalary = salary;
18
19     for(int i = 0; i < years; i++) {
20       futureSalary *= (1 + annualGrowth);
21     }
22
23     return futureSalary;
24   }
25 }
```

Answer the following questions:

1° What is the scope of:

a) `employeeID`, `name`, `salary` (lines 5-7) .....

b) `years` (line 15) .....

c) `annualGrowth`, `futureSalary` (lines 16-17) .....

d) `i` (line 19) .....

2° If 3 instances of type `Employee` are created, what will be the value of their attributes `employeeID` and `countEmployees` (after all 3 have been created)? Explain. .....