Programming 1

Ali Osman TOPAL (aliosman.topal@uni.lu)

Object-Oriented Programming

Submission (Preliminary: 31/10, Final: 7/11)
The mandatory exercises for **Code Submission** are **1, 2, 5** (marked with an ❗). All other exercises on this sheet are optional but still highly recommended! The **Explainer Video** 🎥 for this sheet must be realized on **Exercise 2 ("Student List")**. A Flipgrid invitation link will be posted on Moodle.

1 Object-Oriented Programming

Exercise 1 – ❗ Quadratic Equation #Basic Class #Method Declaration #Constructor #Parameter #Return
Design a class named `Quadratic` for a quadratic equation $ax^2 + bx + c = 0$

1° Add a constructor to initialize the variables a, b, and c, which represent three coefficients of the equation.
2° Add getter methods for a, b, and c.
3° Add a method `getDiscriminant()` that returns the discrimanant, which is $b^2 - 4ac$
4° Add methods named `getRoot1()` and `getRoot2()` for returning two roots of the equation:
$root_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$, $root_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$
5° Write a test program that asks the user to enter values for a, b, and c, and then displays the result based on the discriminant:
If the discriminant positive, display two roots.
If the discriminant is 0, display the one root.
If the discriminant is less than 0, display "The equation has no roots".

Sample Outputs:

Enter the coefficients a b c (separate them with a space) : **1 4 6**
The equation has no roots!

Enter the coefficients a b c (separate them with a space) : **1 5 6**
There are two roots:
Root1: -2.0
Root2: -3.0

Enter the coefficients a b c (separate them with a space) : **1 4 4**
There is one root: -2.0

Exercise 2 – ❗ 🎥 Student List

Due to Corana restrictions, a maximum of 15 students can participate in the labs. As a TA of the Programming 1 course, it is your job to prepare the list of students for the on-campus sessions. However, attendance at the on-campus sessions is voluntary and the list changes every minute. So, you have decided to create an application to keep the list. You know that it is best to use a class and add all the necessary methods to it. This way the new TA can easily change the application if circumstances change.

Now you need to create a class and use as attributes an array to store the names (first and last name) and a counter for the number of registered students. The moment the object is created the counter should be initialized with 0 and the array with the size provided in input by the user. Then several methods: to **`add`** a student, to **`remove`** a student, **`search`** for a student, **`report`** the registered students and **`remove all`** the students from the list for a new session. Then, write a main program to test the implementation of these methods.

ℹ **add a student:** if there is sufficient space in the array we add the student. Use the stack approach (add the new student at the end of the list). If there is no space show a warning on the console.

ℹ **remove a student:** Follow these steps to remove a student from the list.

Step 1: Search for the student
Step 2: Remove the student by replacing it with "null "
Step 3: Shift the following names (if there is any)
Step 4: Remove the last element
Step 5: Decrement the counter.

Exercise 3 – Bank Account

1° Create a class **`Person`** with attributes `id` and `name`, a getter to retrieve their values and a constructor which sets their values.

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. Report on the console, the amount of money transferred and the transfer time/date.

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

Exercise 4 – 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 5 – ! 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).