Software Engineering and Programming Basics - WS2021/22
Assignment 1

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professorship of Software Engineering

11| 2022

Organisational

Deadline

27.11.2022 - 23:59 CET

Submission

Please read these instructions thoroughly! Submissions that do not adhere to the instructions **will not be graded**! Do also take into account the 'Assignment Guidelines' that have been added in Opal!

Submit your files in **Opal** in the designated course node named **Assignment 1**. Your answers need to be submitted as a .java file. If you use several classes, please create a separate .java file for each class.

Make sure your files are correctly named: Package, class and method names should be **exactly** as given on the Task Sheet!
Use the package name **assignment1**.

At the beginning of each file, include the full name and matriculation number of each member of your group in a comment. You can work with **maximum three people** per group.

Important Note: One submission per group is enough. However, **you need to make absolutely sure that you have included the names and matriculation numbers of all group members correctly**! People who cannot be unambiguously identified cannot receive any points!
You can take a look at the Sample Solution provided in Opal for a suitable format.

Allowed Classes

You are allowed to use the following classes and their methods from the Java Standard library:
Object, String, StringBuilder, Math, Arrays

All these classes are introduced in the extra lecture "Important Classes in Java".

You are **not** allowed to use any other classes, except those you write on your own.
You are **not** allowed to use any third party libraries.

Plagiarism Notice

You are **allowed** to search for and use solutions on the internet, provided they do not make use of any other classes than the ones mentioned above. **However**, if you use any code that has not been created by yourself, you need to state the URL of its source in a comment! See the 'Assignment Guidelines' presentation for an example.

You are **not** allowed to use a solution by another student outside of your group! All submissions will be checked for plagiarism. If any copies are found, none of the involved parties will receive any points!

Questions

It is important that all students are able to access all necessary information. Therefore, if you have any questions, please ask them in the course forum in the thread 'Assignment 1: Questions'. Make sure to ask your questions early enough. We will not answer questions on the weekend of the deadline!

Assignment 1: Weather Forecast

Your client is a weather forecast that needs data to be prepared in specific ways. You are asked to engineer a class in java that prepares data on rainfall that has been received from different weather stations across the region. You need to adhere to certain specifications, so the data provided by your class can be successfully utilized.

The following specifications have been agreed upon:

The name of the class is **Rainfall**.

It needs to store the data received from the weather stations. The data is provided in the following format:

The rainfall data measured by the weather stations in the region is given as a **two dimensional int array**. Each row of the array corresponds to a certain weather station, while each column of the array corresponds to a certain day.

Weather Stations	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Station 1	-10	22	33	19	45	75	20
Station 2	35	-6	57	8	10	-100	10
Station 3	15	20	29	39	30	75	20

Table 1: Rainfall data from 3 weather stations over 7 days

Additionally, a (one dimensional) String array with weather descriptors is provided. The Strings can have one of these three values: ”sunny”, ”rainy” or ”thunderstorm”.

Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
sunny	rainy	thunderstorm	sunny	sunny	thunderstorm	sunny

Table 2: Weather descriptors for the corresponding 7 days

Constructor

Create a constructor for the class **Rainfall** which can receive a two-dimensional int array *and* a String array as parameters (in this order).

Method 1 - dataPreparation

The class **Rainfall** needs a method called **dataPreparation** which processes the data that has been stored in the class. It returns a two-dimensional int array with the correctly processed data.

The rainfall data received from the weather stations can be faulty: Sometimes, they malfunction and report negative numbers. This method needs to handle these cases in certain ways, depending on the day's weather descriptor:

- If the weather descriptor is "sunny", replace the negative value with 0.
- If the weather descriptor is "rainy", replace the negative value with the average[1] of the values reported from the other weather stations of that day. Make sure that you do not include any other negative values in the calculation of the average, in case more than one weather station has malfunctioned!
- If the weather descriptor is "thunderstorm", replace the negative value by its corresponding positive value.

Example: Here we have modified the negative entries from the original table to the correct values according to rules defined above.

Weather Stations	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Station 1	0	22	33	19	45	75	20
Station 2	35	21	57	8	10	100	10
Station 3	15	20	29	39	30	75	20

Table 3: Rainfall data after data preparation

[1] Round down averages (example: 4.99 → 4)

Method 2 - totalRainfall

The class **Rainfall** needs another method called **totalRainfall**.
This method calculates the total rainfall measured by all weather stations over all days and returns the result as an int value.

Make sure that you only work with data that has been sufficiently prepared using the **dataPreparation** method!

Example: Here we have calculated the rainfall total for all stations per day, as well as total rainfall per station. The overall total that the method in this example should return is **683**.

Weather Stations	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Total
Station 1	0	22	33	19	45	75	20	214
Station 2	35	21	57	8	10	100	10	241
Station 3	15	20	29	39	30	75	20	228
Total	50	63	119	66	85	250	50	**683**

Table 4: Rainfall data with calculated total rainfall

Method 3 - trend

The final method of the class **Rainfall** that you need to create is called **trend**.
It receives an int value *n* and returns a weather descriptor as a String.
The method's purpose is to forecast what the weather on the next day will be like, depending on the average[2] rainfall per station and day of the past *n* days.

- If the average rainfall of the past *n* days was below 50, return "sunny".
- If the average rainfall of the past *n* days was 50 or higher, return "rainy".
- Special case: If the average rainfall of the past *n* days was exactly 75, return "thunderstorm".

Make sure that you only work with data that has been sufficiently prepared using the **dataPreparation** method!

Example:

```
n = 3
average of the past 3 days per station and day = (45+75+20+10+100+10+30+75+20)/9 = 42
→ the method returns "sunny"

n = 2
average of the past 2 days per station and day = (75+20+100+10+75+20)/6 = 50
→ the method returns "rainy"
```

[2]Round down averages (example 4.99 $\to$ 4)