The purpose of this assignment is to practice the functional programming approach in Java and learn how to use the Optional class.

Exercise 1 (acquire the expertise) – Worker Salaries

Open the *WorkerSalaries.java* file in your IDE. The program creates a list of workers and computes some statistics by **simulating** salary increases for selected workers. It focuses on identifying salaries below a particular threshold, applying salary increases based on specified factors, and ultimately calculating the average of the adjusted salaries.
The objective here is to transition the current implementation towards a functional programming style using streams, by following these steps:

1. Replace all for-loops in the *avgIncreasedSalaries* with a single stream operation.
2. Introduce two more variants of salary increase, the first one of 5% and the second one of 1%, by implementing two additional stream operations.
   Important: ensure to maintain the precise order of these additional variants of salary increase!
3. Compare the results of the three variants of salary increase (3%, 5%, and 1%) and consider whether side effects might be causing any issues. As a hint, consider printing the list of workers.
4. Remove any side-effects, if present.
5. Refactor the *Payroll* class into an immutable class, adjusting the necessary logic within the stream operations, and then compare the results with the previous version of the program.

Exercise 2 (problem solving) – Text analyzer

Write a program that reads a text file and outputs the following information, without using any for / while loops but instead using only streams in combination with terminal operations on *Optional*:

- the longest word
- the shortest word with at least 5 characters
- the word that contains the highest number of vowels
- the first word with at least 10 characters

Each information must be obtained using a single stream operation.
For each stream operation implement the following four *Optional* terminal operation variants:

- *ifPresent()*: the consumer function should print the result to console.
- *orElse()*: should provide a default string, such as “<not found>” and the result should be printed to console.
- *ifPresentOrElse()*: if the result is present the consumer function should print the result to console, otherwise an error message should be written to standard error (e.g. “Could not find shortest word.”).
- *orElseThrow()*: the stream operation result must be printed to console, otherwise the thrown exception must be handled by writing an error message to standard error (e.g. “Could not find shortest word.”).

HINTS:

- The following snippet may be used to read the contents of a text file into a List of “words”:

```
Arrays.asList(new String(Files.readAllBytes(Paths.get("/path/to/file.txt"))).split("\\s+"))
```

Exercise 3 (optional deepening) – Product catalog

Open the *ProductCatalog.java* file in your IDE. The program simulates a store that provides a range of product types, such as headphones and smartphones. It incorporates a loyalty-based reward system, applying discounts to the standard product prices based on customer loyalty levels. Customers can utilize the program

to search for specific product types, which will return a list of available products along with their respective prices. The program further includes sample searches performed by different customers.
The goal of the exercise is to identify and resolve the unintended side-effects.