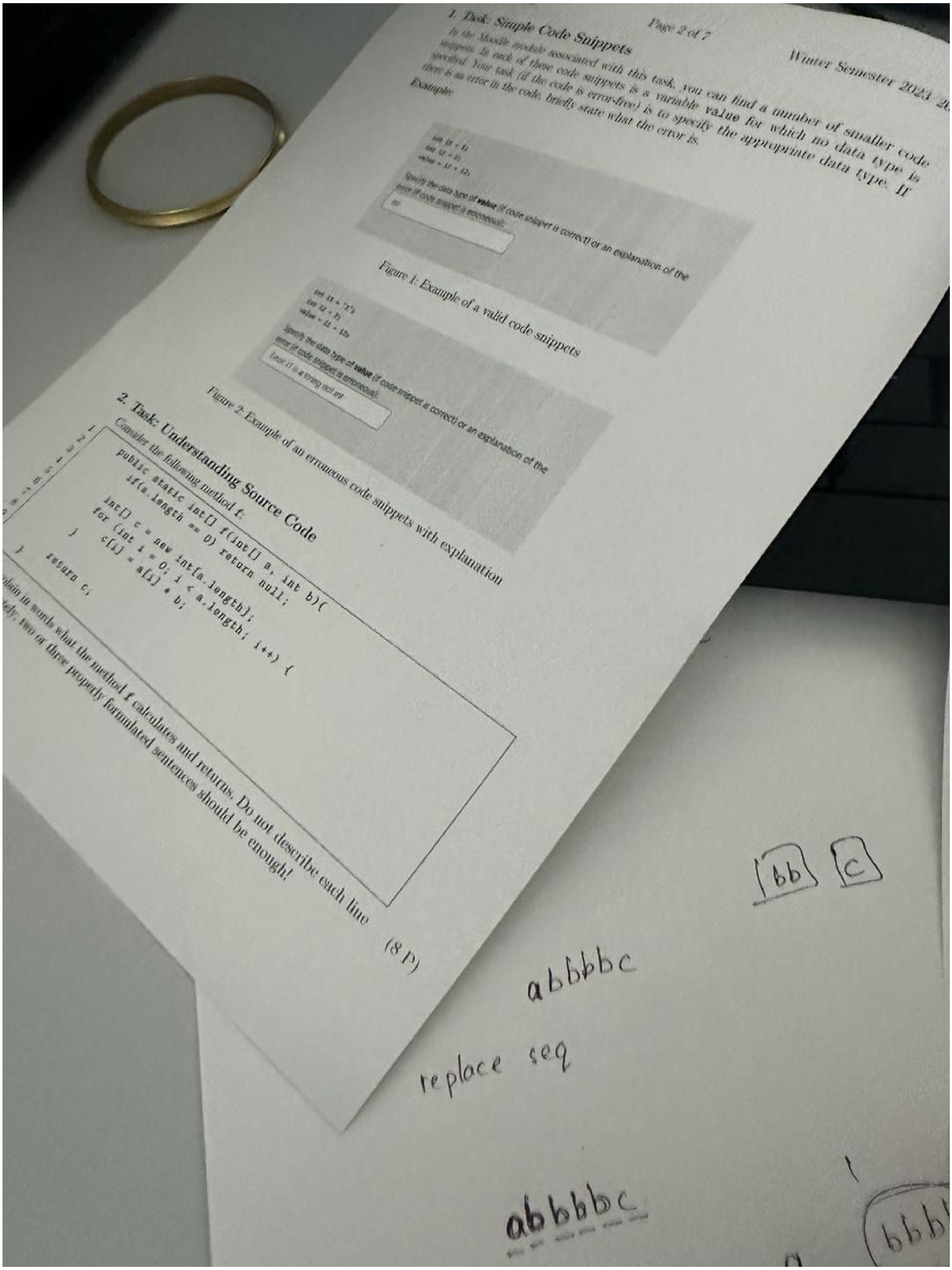
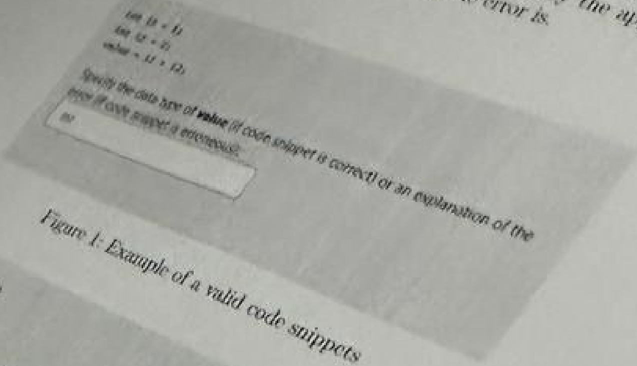


## 1. Task: Simple Code Snippets

In the Moodle module associated with this task, you can find a number of smaller code snippets. In each of these code snippets is a variable `value` for which no data type is specified. Your task (if the code is error-free) is to specify the appropriate data type. If there is an error in the code, briefly state what the error is.

Example:

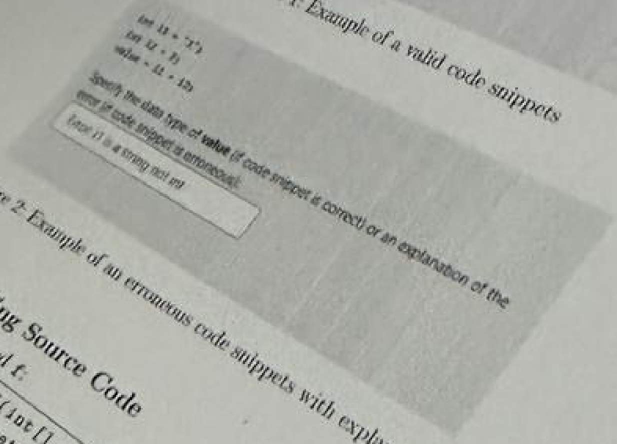


```
int i1 = 1;
int i2 = 2;
value = i1 + i2;
```

Specify the data type of **value** (if code snippet is correct) or an explanation of the error (if code snippet is erroneous):

int

Figure 1: Example of a valid code snippets



```
int i1 = "1";
int i2 = 2;
value = i1 + i2;
```

Specify the data type of **value** (if code snippet is correct) or an explanation of the error (if code snippet is erroneous):

Error: i1 is a string not int

Figure 2: Example of an erroneous code snippets with explanation

## 2. Task: Understanding Source Code

Consider the following method f:

```
public static int[] f(int[] a, int b){
    if(a.length == 0) return null;

    int[] c = new int[a.length];
    for (int i = 0; i < a.length; i++) {
        c[i] = a[i] * b;
    }
    return c;
}
```

plain in words what the method f calculates and returns. Do not describe each line
tely, two or three properly formulated sentences should be enough! (8 P)

bb c

abbbc

replace seq

abbbc

bbb

(b) Implement the algorithm from `f` in a method `g`. This new method must provide the same results as `f` for all parameter values. However, `g` is *not allowed* to use a `for` statement. Of course, calling `f` is *not allowed* either. (7 P)

The source code for (a) and a suitable `main` method are available in Moodle under `Task 02b`. Your solution to (a) should be written in the corresponding Moodle module (`Task 02a`), the solution to (b) should be stored in the file `Task.java` at `Task 02b` in Moodle; an evaluation with simple examples is provided there.

## 3. Task: Strings I

**(10 P)**

A prefix is a sequence of leading letters of a word. For example, "ab" is a prefix of the string "abcde". The longest common prefix of two strings (e.g. "abbbcd" and "abbcde") therefore describes the longest sequence of leading letters of two strings (here "abb").

Implement the method

```
String longestCommonPrefix(String s1, String s2)
```

which receives two strings `s1` and `s2` as input. The aim of the method is to determine the longest common prefix of two strings and return it. If two strings do not have a common prefix, return the "empty" string (`""`). **Attention:** For this task, you can assume that `s1` and `s2` have the same length.

**Examples:**

```
longestCommonPrefix("abbbcd","abbcde") = "abb"
longestCommonPrefix("abcd","acbd") = "a"
longestCommonPrefix("abcd","efgh") = ""
```

A general scheme for the solution can be found in Moodle under `Task 03` in the file `Task.java`, with the possibility of a test evaluation by `Evaluation.java`. The solution must be saved in the file `Task.java`.

## 4. Task: Recursion

**(13 P)**

In this task, you should copy your solution from task 3 and modify it so that it solves the task from task 3 recursively.

Implement the method

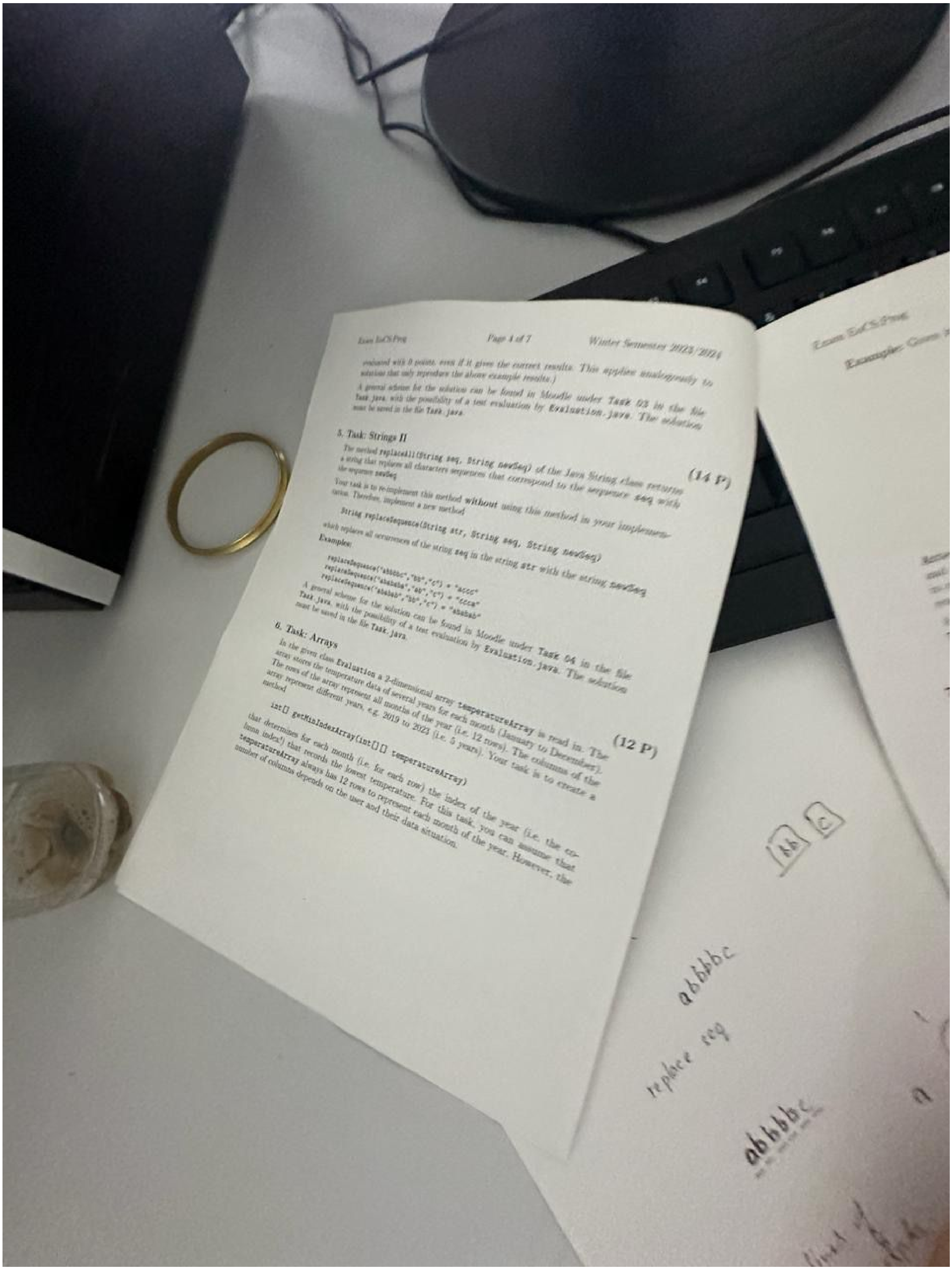
```
String longestCommonPrefixRecursive(String s1, String s2)
```

which receives two strings `s1` and `s2` as input. The aim of the method is to determine the longest common prefix of two strings recursively and return it. If two strings do not have a common prefix, return the "empty" string (`""`). **Attention:** For this task, you can assume that `s1` and `s2` have the same length.

**Examples:**

```
longestCommonPrefixRecursive("abbbcd","abbcde") = "abb"
longestCommonPrefixRecursive("abcd","acbd") = "a"
longestCommonPrefixRecursive("abcd","efgh") = ""
```

[illegible] not allowed to use any variables other than the parameters [illegible] loops [illegible] also not allowed. (Otherwise your solution will be


bb
c
abbbc
replace seq
abbbc

evaluated with 0 points, even if it gives the correct results. This applies analogously to solutions that only reproduce the above example results.)

A general scheme for the solution can be found in Moodle under `Task 03` in the file `Task.java`, with the possibility of a test evaluation by `Evaluation.java`. The solution must be saved in the file `Task.java`.

## 5. Task: Strings II

**(14 P)**

The method `replaceAll(String seq, String newSeq)` of the Java String class returns a string that replaces all characters sequences that correspond to the sequence `seq` with the sequence `newSeq`.

Your task is to re-implement this method **without** using this method in your implementation. Therefore, implement a new method

```
String replaceSequence(String str, String seq, String newSeq)
```

which replaces all occurrences of the string `seq` in the string `str` with the string `newSeq`.

**Examples:**

```
replaceSequence("abbbbc","bb","c") = "accc"
replaceSequence("abababa","ab","c") = "ccca"
replaceSequence("ababab","bb","c") = "ababab"
```

A general scheme for the solution can be found in Moodle under `Task 04` in the file `Task.java`, with the possibility of a test evaluation by `Evaluation.java`. The solution must be saved in the file `Task.java`.

## 6. Task: Arrays

**(12 P)**

In the given class `Evaluation` a 2-dimensional array `temperatureArray` is read in. The array stores the temperature data of several years for each month (January to December). The rows of the array represent all months of the year (i.e. 12 rows). The columns of the array represent different years, e.g. 2019 to 2023 (i.e. 5 years). Your task is to create a method

```
int[] getMinIndexArray(int[][] temperatureArray)
```

that determines for each month (i.e. for each row) the index of the year (i.e. the column index!) that records the lowest temperature. For this task, you can assume that `temperatureArray` always has 12 rows to represent each month of the year. However, the number of columns depends on the user and their data situation.

**Example:** Given is a two-dimensional array with 5 columns:

$$getMinIndexArray\left(\begin{bmatrix} 2 & \mathbf{1} & 2 & 3 & 2 \\ 4 & \mathbf{3} & 5 & 4 & 4 \\ \mathbf{3} & 5 & 5 & 5 & 6 \\ 7 & 8 & 8 & \mathbf{6} & 8 \\ 14 & 15 & 14 & \mathbf{13} & 15 \\ 18 & 17 & \mathbf{16} & 19 & 18 \\ 19 & 19 & 18 & 19 & \mathbf{17} \\ 21 & 20 & 21 & \mathbf{19} & 20 \\ 13 & 14 & 13 & \mathbf{12} & 14 \\ 6 & 7 & 6 & 7 & \mathbf{5} \\ \mathbf{1} & 2 & 3 & 2 & 3 \\ 3 & 4 & 3 & 3 & \mathbf{2} \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \\ 3 \\ 2 \\ 4 \\ 3 \\ 3 \\ 4 \\ 0 \\ 4 \end{bmatrix}$$

**Attention:** You can assume that `temperatureArray` is not the `null` reference when called and that there is only *one* year with minimal temperature for each month. However, bear in mind that, unlike the number of rows (always 12!), the number of columns in the array read is dynamic.

A general scheme for the solution can be found in Moodle under `Task 06` in the file `Task.java`, with the possibility of test evaluation by `Evaluation.java`. The solution must be saved in the file `Task.java`.

## 7. Task: Object Oriented Programming Concepts

The following task consists of two subtasks to assess basic knowledge in the area of object-oriented programming. Given is a class `Evaluation` which can be used to test your solution. **Attention:** The class and instance variables of the classes you implement should all be externally accessible. Implement the following classes:

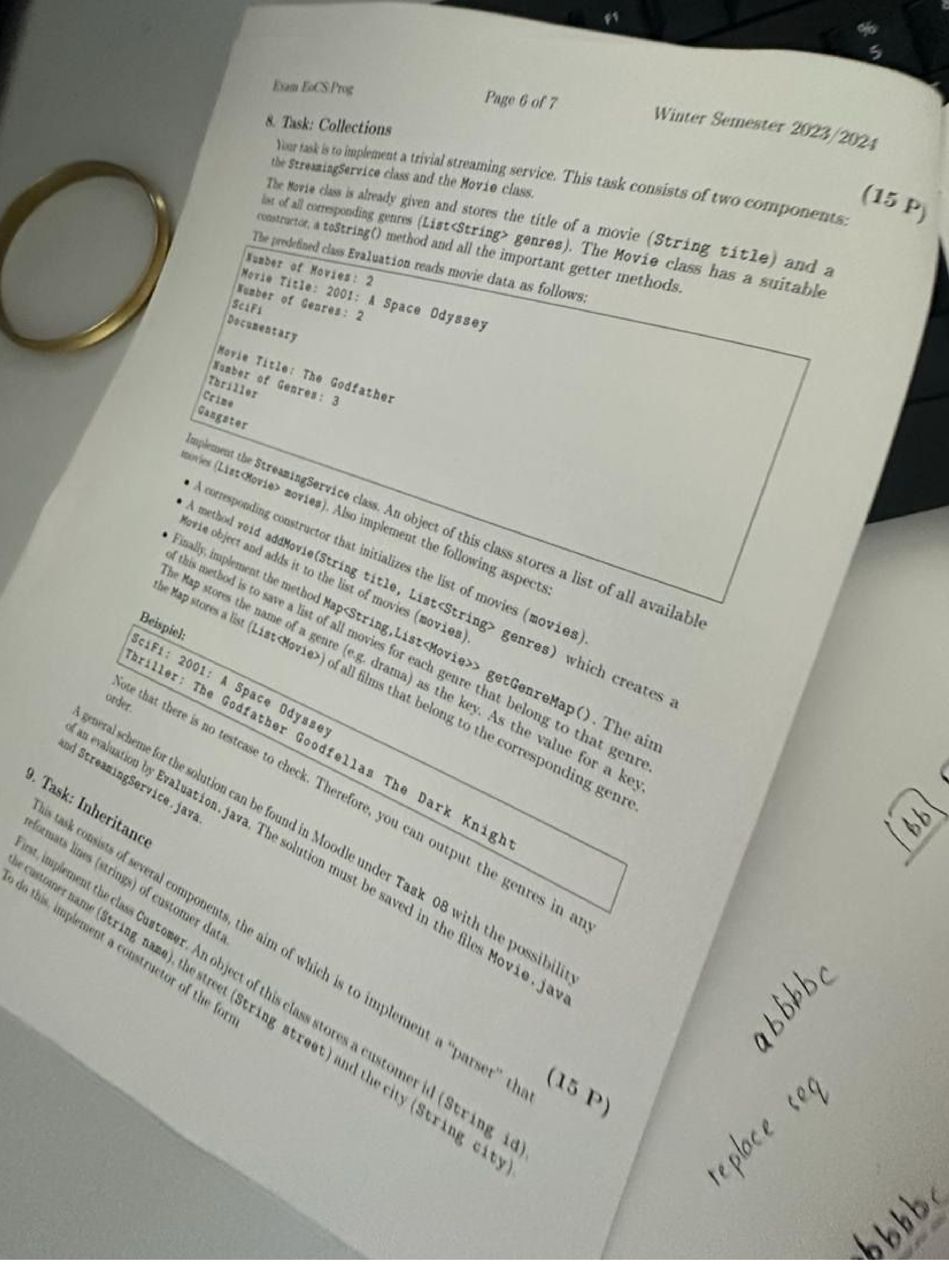
(a) Implement the class `Tuple`, which stores a pair of values with the names `left` and `right`. The variables `left` and `right` should be able to store any data type available in Java (i.e. `String`, `int`, `double`, etc.). In addition, each `Tuple` object has an automatically incrementing ID (`int id`). The ID should start at 0. Also implement a corresponding constructor for the class, which is passed values for `left` and `right`. (6 P)

(b) In the class `Evaluation`, implement the method (7 P)

```
double squareDiv(double a, double b).
```

The method is supposed to calculate $\frac{a}{b^2}$ in the case that $b \neq 0$. If $b == 0$, throw a `ArithmeticException` with the message "b is not allowed to be zero".
Next, extend the `main` method so that the `squareDiv(a,b)` method is used by it. Catch the `ArithmeticException` and continue to request new values for a and b via the `Scanner` until no more exceptions are thrown and `squareDiv(a,b)` can be [illegible]lated.

[illegible]l scheme for the solution can be found in Moodle under `Task 07a` and `Task 07b`, [illegible]e possibility of an evaluation by `Evaluation.java`.



## 8. Task: Collections

**(15 P)**

Your task is to implement a trivial streaming service. This task consists of two components: the `StreamingService` class and the `Movie` class.

The `Movie` class is already given and stores the title of a movie (`String title`) and a list of all corresponding genres (`List<String> genres`). The `Movie` class has a suitable constructor, a `toString()` method and all the important getter methods.

The predefined class `Evaluation` reads movie data as follows:

```
Number of Movies: 2
Movie Title: 2001: A Space Odyssey
Number of Genres: 2
SciFi
Documentary

Movie Title: The Godfather
Number of Genres: 3
Thriller
Crime
Gangster
```

Implement the `StreamingService` class. An object of this class stores a list of all available movies (`List<Movie> movies`). Also implement the following aspects:

- A corresponding constructor that initializes the list of movies (`movies`).
- A method `void addMovie(String title, List<String> genres)` which creates a `Movie` object and adds it to the list of movies (`movies`).
- Finally, implement the method `Map<String,List<Movie>> getGenreMap()`. The aim of this method is to save a list of all movies for each genre that belong to that genre. The `Map` stores the name of a genre (e.g. drama) as the key. As the value for a key, the `Map` stores a list (`List<Movie>`) of all films that belong to the corresponding genre.

**Beispiel:**

```
SciFi: 2001: A Space Odyssey
Thriller: The Godfather Goodfellas The Dark Knight
```

Note that there is no testcase to check. Therefore, you can output the genres in any order.

A general scheme for the solution can be found in Moodle under **Task 08** with the possibility of an evaluation by `Evaluation.java`. The solution must be saved in the files `Movie.java` and `StreamingService.java`.

## 9. Task: Inheritance

**(15 P)**

This task consists of several components, the aim of which is to implement a "parser" that reformats lines (strings) of customer data.

First, implement the class `Customer`. An object of this class stores a customer id (`String id`), the customer name (`String name`), the street (`String street`) and the city (`String city`).

To do this, implement a constructor of the form

bb c

abbbc

replace seq

abbbbc