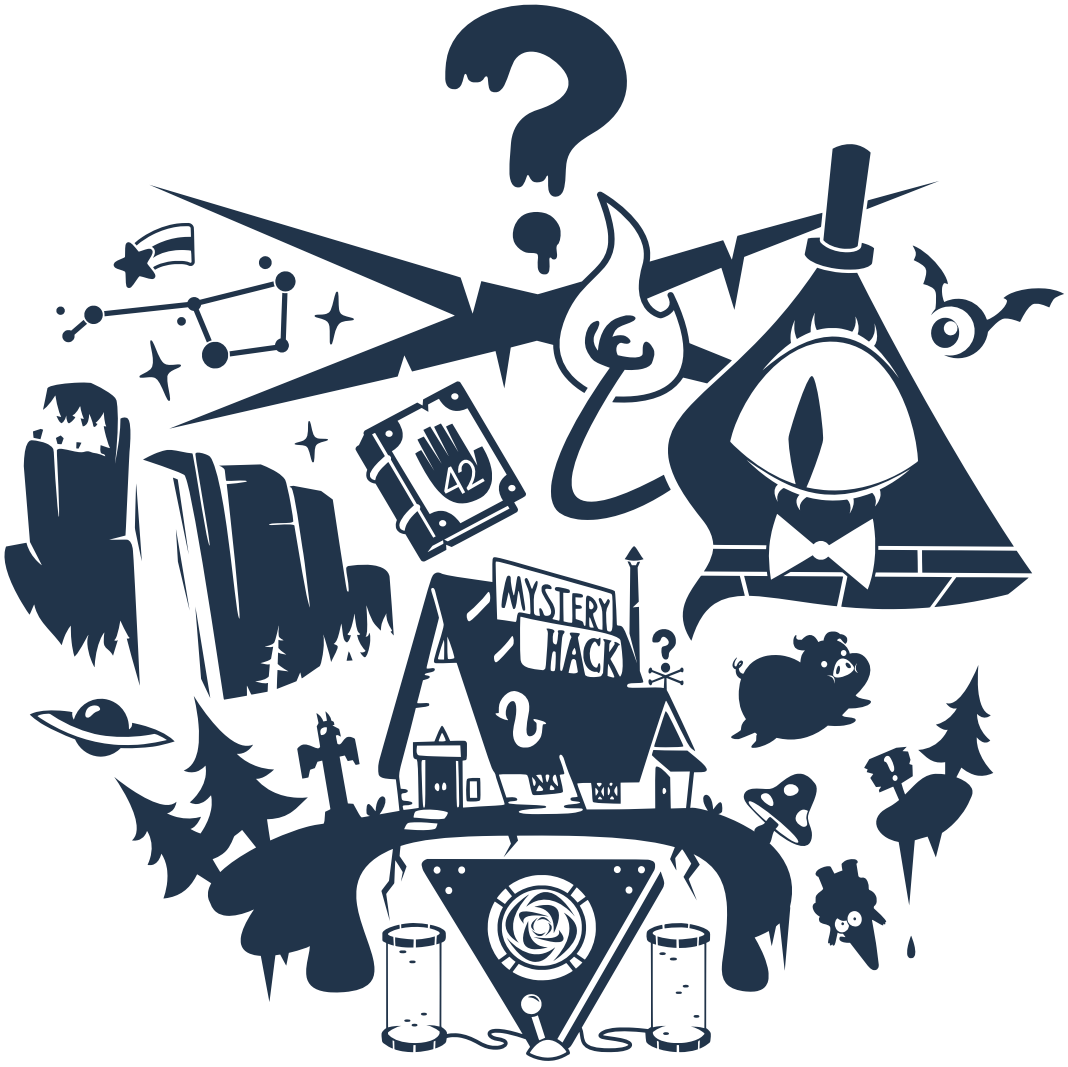
Exercise — Classics

version **#1.0.0**



Yet Another Kind of Assistants 2026
<assistants@tickets.assistants.epita.fr>

Copyright

Contents

*https://intra.forge.epita.fr

File Tree

```
classics/
├── pom.xml
└── src/
    └── main/
        └── java/
            └── fr/
                └── epita/
                    └── assistants/
                        ├── Main.java
                        └── classics/
                            └── Classics.java  (to submit)
```

Authorized imports

- java.lang.*

1 Objectives

In this exercise, you will learn the basics of *Java* syntax and programming by writing classic algorithms.

2 Specifications

The `Classics` file, which already contains all prototypes, is provided alongside a **Main** class for testing purposes. You simply need to replace every `FIXME` with your own code. You can run the `Main` file to test your code.

Be careful!

Do not change the prototypes of the methods in the `Classics` file, as this would prevent your code from being tested. You can still write your own auxiliary methods.

2.1 Factorial

Write a *method* that returns the factorial of a given number.

```
public static long factorial(int n);
```

2.2 Tribonacci

Write a *method* that computes the tribonacci of a given number. A negative number should return `-1`.

$$T_n = T_{n-1} + T_{n-2} + T_{n-3}$$
$$T_0 = 0, T_1 = T_2 = 1$$

Going further...

There are several definitions available for this sequence. You ***must*** use the one provided in the subject.

```
public static long tribonacci(int n);
```

Be careful!

The naive recursive implementation will timeout on our testsuite. You must solve the exercise with an optimized method.

2.3 Palindrome

Write a *method* that checks if a given string is a palindrome. An empty string is a valid palindrome. A `null` string is not a palindrome. The method is not case sensitive and should ignore spaces.

```
public static boolean isPalindrome(String s);
```

2.4 Insertion sort

Write a *method* that sorts an array of integers using the insertion sort algorithm. You do not have to handle the case where the input array is `null`.

```
public static void insertionSort(int[] array);
```

2.5 Combine strings

Write a method that uses a `StringBuilder` to combine two strings by alternating their characters. If one string is longer than the other, the remaining characters are appended at the end. You do not have to handle the case where one or multiple inputs are `null`.

```
public static String combine(String s1, String s2);
```

3 Testing

After writing your code, you can run the `Main` file to test it. Don't forget to add more tests to make sure everything works properly.

Being a hero means fighting back even when it seems impossible.