This guide explains how to write solutions to CEOI 2019 problems in Java.

About your program:

- The program must be a console application (i.e., no GUI).
- The program must read stdin and write to stdout, unless instructed otherwise in the statement.
- The grader discards whatever you print to stderr.
  Printing to stderr is allowed, but it may slow your solution down.

About classes, packages and such:

- Your solution **must not** be declared as a part of a package, it must stand alone.
- Even if your solution contains multiple classes, its entire implementation must be in the single `filename.java` file you submit.
- Each task has a short name that consists of letters only.
  This name can be found in the task statement.
  For example, the example task “Count squares” has a short name `countsquares`.
- When compiling your solution, we rename the submitted file to `shortname.java`.

- The **recommended** approach is to have a `public class shortname` that contains the `main()` function. Your source file can also contain additional classes. (Obviously, these cannot be public.)

- If you want to use different file names locally, another **supported** approach is to have only one outer class in your solution. This class **must not be public** (as its name will not match the filename during the compilation). If you need to use more than one class, the additional classes should be nested inside the outer class. See Example 3 below.

About the input:

- Input files have Linux newlines (only LF, ASCII 10, not CR+LF).

About the output:

- Each line of your output **must be terminated by a newline**.
- Follow the exact output specification from the statement.
  Do not print extra whitespace you were not instructed to print.
- It is recommended to use a buffered writer if the output is large.

About large I/O:

- Whenever the input and/or output is large, it is recommended to use a buffered reader and/or writer.

Example 1:

Suppose we have a task with the short name `add`.
The task is to input two small integers and output their sum.
Here is a minimalistic correct implementation.
Note the use of `println` to also print a newline after the answer.

```
import java.util.Scanner;

public class add {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        System.out.println(a+b);
    }
}
```

Example 2:

The following program works as well. As long as you only have one class and it is not public, its name may be arbitrary.

```
import java.util.Scanner;

class IReallyNeedMoreSleep {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a = in.nextInt();
        int b = in.nextInt();
        System.out.println(a+b);
    }
}
```

Example 3:

Here is a solution for the same task that uses two classes. Note that only the class named after the task is public, and that class contains the `main()` function that will be executed.

```
import java.util.Scanner;

class Adder {
    int total;
    public Adder() { total = 0; }
    public void add(int x) { total += x; }
    public int getSum() { return total; }
}

public class add {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        Adder a = new Adder();
        a.add( in.nextInt() );
        a.add( in.nextInt() );
        System.out.println( a.getSum() );
    }
}
```

Example 4:

Finally, here is a second solution for the same task that uses two classes. This time the classes both use custom names. All classes other than the one whose `main()` method you want to execute must be nested inside that class.

```
import java.util.Scanner;

class SolveTheAddTask {
    class Adder {
        int total;
        public Adder() { total = 0; }
        public void add(int x) { total += x; }
        public int getSum() { return total; }
    }

    int solve() {
        Scanner in = new Scanner(System.in);
        Adder a = new Adder();
        a.add( in.nextInt() );
        a.add( in.nextInt() );
        return a.getSum();
    }

    public static void main(String[] args) {
        System.out.println( new SolveTheAddTask().solve() );
    }
}
```

Using and printing floating-point numbers:

Use the data type `double` to store floating-point numbers.
Use `System.out.printf("%.15f", x);` to print the value of `double x` to 15 decimal places.

In the **scissors** task, it is recommended to print all numbers to at least 10 decimal places.