How to code correctly (short version)

- Read the problem and find the solution; find the **algorithm** (the set of steps) that you need to perform to solve it. Don't think about coding at this point.
- Rewrite the algorithm as a **pseudocode**.
- **Translate** it into Java (or any other programming language).

How to practice efficiently

Solve the PSS or book tasks **on paper** by **timing** yourself. Then, look at your paper and see if you can find any errors. Once you are sure you don't have any errors, write the same code on a computer and **run it**. In case there are still some errors, take note of them and move forward.
While doing this, **prioritize accuracy over speed**. Once you are comfortable with writing correct codes, try to work on timing.

How to code correctly (long version)

First, start by **outlining the idea of the solution**. Don't rush and start writing the actual code. Define all the logical steps that you must carry out in order for your solution to be correct. Test your algorithm on several test cases by hand. You have sufficient math background to solve most of the problems (and you can google the formulas/concepts/algorithms).

It might be helpful to convert your solution into **pseudocode**. Create a draft of future code mainly using natural language rather than strict programming syntax. This will help to define the visual structure of the future code.

After you have outlined the solution, start writing your code. Implement your solution **unit by unit**: don't write all the code at once. Write a logical unit, test it, see that it behaves the way you want, and move on to the next unit. Don't hesitate to use a lot of ***System.out.print*** statements.

If you don't remember something from syntax/definitions- **refer to the slides**; they have everything you need. Don't want to spend time looking through slides? **Google** and **chatGPT** are there to help you.

When you finish writing the code- **read it**, translate each logical unit, and see if it makes sense. **Test** on the test cases you used while outlining the solution. Did you pass them? Yeeeey! Now create new ones and test again until you are 100% sure your solution covers all possible inputs.

And if your program doesn't give an expected solution- start **debugging**. Take your test case, and start applying your code on it line by line. The odds are high that you will find the error. Use ***sout***, it is truly your best friend. Alternatively, use **IDE debuggers**.

Another great debugging method is the **rubber duck debugging**. Put a rubber duck/toy/pet/family member beside you and force yourself to explain the code line by line to the "duck" (don't skip lines, don't assume that ducky knows anything about programming!).

Following all the steps discussed will help you come up with correct/working codes. Don't think that this approach is a waste of time; this is the right way of gaining experience. Doing this several times will help you to work much faster in the future.

Example

Problem: calculate the average of n values.

Algorithm:

Get all the values, sum them up, and divide by n.

Pseudocode:

```
read value n
double sum = 0
for i from 1 to n
      read value and add it to sum
return sum/n
```

Code:

```
public static double average(){
      int n = scanner.nextInt();
      double sum = 0;
      for(int i=0; i<n; i++)
            sum+= scanner.nextInt();
      return sum/n;
}
```