Week 2-2 : Good Programming Style

Java Programming style

- Name Convention
  - Use full English descriptions for names. Avoid using abbreviations.
    - For example, use names like firstName, lastName, and middleInitial rather than the shorter versions fName, lName, and mi.
  - Avoid overly long names (greater than 15 characters).
    - For example, setTheLengthField should be shortened to setLength.
  - Avoid names that are very similar or differ only in case.
    - For example, avoid using the names product, products, and Products in the same program for fear of mixing them up.

  - **Variable Naming Conventions**
  - **Avoid generic names** like number or temp whose purpose is unclear.
  - Compose variable names using **mixed case letters** starting with a lower case letter.
    - For example, use salesOrder rather than SalesOrder or sales_order.
  - Use plural names for arrays.
  - For example, use testScores instead of testScore.
    - **Exception**: for loop counter variables are often named simply i, j, or k, and declared local to the for loop whenever possible.

  - **Constant Naming Conventions**
  - Use **ALL_UPPER_CASE** for your named constants, separating words with the underscore character. For example, use TAX_RATE rather than taxRate or TAXRATE.
  - **Avoid using magic numbers** in the code. Magic numbers are actual numbers like 27 that appear in the code that require the reader to figure out what 27 is being used for. Consider using named constants for any number other than 0 and 1.

- Method Naming Conventions
- Try to come up with **meaningful method names** that succinctly describe the purpose of the method, making your code self-documenting and reducing the need for additional comments.
- Compose method names using **mixed case letters**, beginning with a lower case letter and starting each subsequent word with an upper case letter.
- Begin method names with a strong action verb
- Use the prefixes get and set for ***getter*** and ***setter*** methods.
- If the method **returns a boolean value**, use is or has as the prefix for the method name.

- Parameter Naming Conventions
- With **formal parameter names**, follow the same naming conventions as with variables,
-

- Commenting Convention
  - Single-Line Comments
  - Begin single-line comments with a **double slash** (//) that tells the compiler to ignore the rest of the line. Note: do not place any characters between the two slashes
  - Trailing Comments
  - Trailing comments are used to provide an explanation for a **single line of code**. Begin trailing comments with a double slash (//) and place them to the right of the line of code they reference.

- Formatting
  - Indentation
  - Use **three spaces** for indentation to indicate nesting of control structures.

  - White Space
  - Use blank lines and blank spaces to improve the **readability** of your code.

  - Line Length
  - Break after a comma.
  - Break before an operator.
  - Align the new line with the beginning of the expression at the same level

on the previous line.

Reference : Java Good Programming Style - http://www.cwu.edu/

Basic of Java

- Generate Class

  Public class Java {

  ...

  }

- Method

  public int sum(int n, int m)

  return n+m;

  }

- Print
- System.out.println("print");

[Exercise]

1. 키보드로 입력된 성적에 대해 학점을 부여하는 코드를 작성

   Get a score and assign a grade about that score.

   A : 90<=score, B : 80<=score<90, C : 70<=score<80, D : 60<=score<70, F : score<60

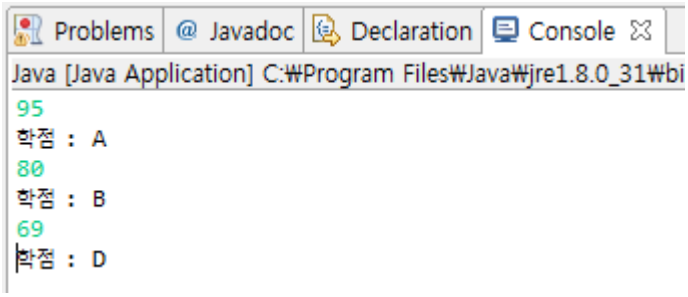
```
import java.util.Scanner;

public class Grading {
    public static void main(String[] args) {
        char grade;
        Scanner a = new Scanner(System.in);

        /* Blank */


    }

}
```



2. 입력된 수 중 가장 큰 수를 구하시오.
   Print out a max number.

```
import java.util.Scanner;

public class Array {
        public static void main(String[] args) {
                Scanner in = new Scanner(System.in);
                int intArray[] = new int[5];



                /* Blank */


        }
}
```