GOOD PROGRAMMING STYLE

3RD WEEK LECTURE

엄현상(Eom, Hyeonsang)
School of Computer Science and Engineering
Seoul National University

Outline

- Good Programming Style Java
- Q&A

Good Programming Style JAVA
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.

```
double tax1; // sales tax rate (example of poor variable name)
double tax2; // income tax rate (example of poor variable name)
double salesTaxRate; //no comments required due to
double incomeTaxRate; //self-documenting variable names
```

Good Programming Style JAVA
Name Convention Cont'd

- **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.

```
for (int i = 0; i < MAX_TEMPERATURE; i++)
{
        boilingPoint = boilingPoint + 1;
}
```

Good Programming Style JAVA
Name Convention Cont'd

- **Variable Naming Conventions Cont'd**
  - **Declaring and commenting local variables**
    - Declare each local variable on its own line of code. Do not group variables, separated by commas as a short-cut to declaring each on its own line.
    - If the meaning and use of the variable is not clear, add an endline comment (//) stating what the variable is used for and why. However, a better solution is to choose a meaningful name to avoid the need for the endline comment.
    - Whenever possible, initialize the variable with its starting value in the declaration statement.

Good Programming Style JAVA

- **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.

```
day = (3 + numberOfDays) % 7; //NO! uses magic numbers
final int WEDNESDAY = 3;
final int DAYS_IN_WEEK = 7;
day = (WEDNESDAY + numberOfDays) % DAYS_IN_WEEK;
//Yes, self-documenting
```

Good Programming Style JAVA
Name Convention Cont'd

- **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** (for example, deposit). If the verb is not descriptive enough by itself, include a noun (for example, addInterest). Add adjectives if necessary to clarify the noun (for example, convertToEuroDollars).

Good Programming Style JAVA

Name Convention Cont’d

- **Method Naming Conventions Cont’d**
  - Use the prefixes get and set for ***getter*** and ***setter*** methods. Getter methods merely return the value of a instance variable; setter methods change the value of a instance variable. For example, use the method names getBalance and setBalance to access or change the instance variable balance.
  - If the method **returns a boolean value**, use is or has as the prefix for the method name. For example, use isOverdrawn or hasCreditLeft for methods that return true or false values. Avoid the use of the word ***not*** in the boolean method name, use the ! operator instead.
    For example, use !isOverdrawn instead of isNotOverdrawn.

Good Programming Style JAVA

Name Convention Cont'd

- **Parameter Naming Conventions**
  - With **formal parameter names**, follow the same naming conventions as with variables, i.e. use mixed case, begin with a lower case letter, and begin each subsequent word with an upper-case letter
  - Consider using the prefix a or an with parameter names. This helps make the parameter distinguishable from local and instance variables.
  - Occasionally, with very general purpose methods, the names chosen may be rather generic (for example, aNumber). However, most of the time the parameter names should succinctly describe the type of value being passed into the method.

```
public void deposit(long anAccountNumber, double aDepositAmount) { … }
```

Good Programming Style JAVA
Commenting Convention

- Use comments to provide overviews or summaries of chunks of code and to provide additional information that is not readily available in the code itself.
- Comment the details of nontrivial or non obvious design decisions; avoid comments that merely duplicate information that is present in and clear from reading the code.

Good Programming Style JAVA
Commenting Convention Cont'd

- **File Header Comments**
  - **File header comments** provide readers and graders with the information they need to find your project on the network, identify the program's author, and to collect statistics on the project's difficulty based on the hours required to complete the program.

```
//***************************************************************
// Assignment: Program 2
// Account: (Enter your snu  account number here)
//
// Author: (Enter your full name here)
//
// Completion time: (Enter the total number of hours you
//                    spent on the assignment)
//
// Honor Code: I pledge that this program represents my own
//   program code. I received help from (enter the names of
//   others that helped with the assignment, write no one if
//   you received no help) in designing and debugging my program.
//**************************************************************
```

Good Programming Style JAVA
Commenting Convention Cont'd

- **Single-Line Comments**
  - Use single-line comments, also called inline comments, to provide **brief summary comments** for chunks of code.
  - Proceed single-line comments with a blank line and align the comment with the code it summarizes. Do not feel the need to comment every single line of code, rather summarize chunks of code between 3 to 7 lines in length.
  - 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.

```
// Compute the exam average score for the midterm exam
   sumOfScores = 0;
   for (int i = 0; i < scores.length; i++)
      sumOfScores = sumOfScores + scores[i];
   average = float(sumOfScores) / scores.length;
```

Good Programming Style JAVA
Commenting Convention Cont'd

- **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.
  - Trailing comments are used to explain tricky code, specify what abbreviated variable names refer to, or otherwise clarify unclear lines of code.
  - In general, **avoid the use of trailing comments**. Instead rewrite tricky or unclear code, use meaningful variable names, and strive for self-documenting code.

```
ss = s1 + s2 + s3;     //add the three scores into the sum
a = float(ss) / x;     //calculate the mean of the scores
```

Note:
This is an example of the WRONG WAY to comment a program

Good Programming Style JAVA
Formatting

- Formatting refers to the indentation, alignment, and use of white space to lay out your program to increase its **readability** by others.
- **Consistency** is the key to producing readable code. While many can argue to merits of 3 versus 4 spaces of indentation, placement of curly braces, etc., the real key is to adopt a formatting style and keep to it.

Good Programming Style JAVA
Formatting

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

```
public class HelloWorld
{
...public void greetUser(int currentHour)
...{
......System.out.print("Good");
......if (currentHour < AFTERNOON)
......{
.........System.out.println(" Morning");
......}
......else if (currentHour < EVENING)
......{
.........System.out.println( "Afternoon");
......}
......else
......{
.........System.out.println("Evening");
......}
...}
}
```

Note: The period char (.) is used to show indentation

Good Programming Style JAVA
Formatting

- **White Space**
  - Use blank lines and blank spaces to improve the **readability** of your code.
  - Use **blank lines to separate chunks of program code**. Chunks are logical groups of program statements (generally 3 to 7 lines in length) and usually proceeded with a single-line summary comment. Use one blank line before every program chunk. Use two blank lines before the start of each new method within a class.
  - Use one blank space on both sides of operator symbols, after commas in argument lists, and after semicolons in for statements.

Good Programming Style JAVA
Formatting

- **Line Length**
  - **Avoid lines longer than 80 characters** When an expression will not fit on a single line of 80 characters, break it according to these general principles:
    - 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.

```
someMethod(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

longName1 = longName2 * (longName3 + longName4 - longName5)
         + 4 * longname6;
```