Java Codding Guidelines for OOP Course

This document describes the coding style guidelines that will be used in OOP course for better readability and maintainability of the code.

# Naming Conventions

## Casing and Word Separation

### Packages

Names representing packages should be in all lower case.

Example: mypackage

### Classes and Types

Names representing classes and types should be written in mixed case starting with upper case (except for primitive types).

Examples: Coin, BookShelf

Note: If type or class name contains an acronym, it should still obey the rule of mixed case.

Example: IdfSoldier

### Variables

Variable names should be in mixed case starting with lower case.

Examples: book, student, guitarString

Note: If a variable name contains an acronym, it should still obey the rule of mixed case.

Example: oopCourse

### Methods

Variable names should be in mixed case starting with lower case.

Examples: addBook(), getValue(), playGame()

## Comprehension of Names

The names of java constructs should be understandable. The names like “tmp”, “temp”, “var”, “str” should be avoided.

# Formatting

## Length of Lines

The length of lines should not exceed 120 characters.

## Indentation

The indentation size will be 4 spaces.

## Blocks

All the nested blocks should be indented.

The blocks will begin from a header (if applicable) and an opening parenthesis.

The blocks will end with a closing parenthesis on a separate line.

Example:

**public int** getWalletSize() {  
 **return this**.**coinList**.size();  
}

If block has a tail, it should follow the closing parenthesis:

do {

...

} while(condition);

The “if…else” block should be formatted as in the following example:

**if**(**this**.**validCoinValues**.contains(value)){  
 **this**.**value** = value;  
} **else** {  
 System.***out***.println(**"Error: value not valid"**);  
}

All the body of the loops and of if…else statements will be surrounded by parentheses, even if the body contains only one expression.

Example:

**if**(**this**.**validCoinValues**.contains(value)){  
 **this**.**value** = value;  
} **else** {  
 System.***out***.println(**"Error: value not valid"**);  
}

Wrong Example:

**if**(**this**.**validCoinValues**.contains(value))  
 **this**.**value** = value;  
**else**  
 System.***out***.println(**"Error: value not valid"**);

## Spaces

All the binary and ternary operators should be surrounded by spaces.

Examples:

**if** (foundSyllable□&&□word.charAt(word.length()□-□1)□!=□**'e'**) {  
 numSyllables++;  
}  
  
**if** (numSyllables□==□0) {  
 numSyllables□=□1;  
}

## Empty Lines

There should be at least one empty line between the methods and the classes.

# Comments

Each method should contain a leading comment describing its purpose, behaviour and validity ranges of the parameters. The comments before the method should be formatted to fit the specification format of Javadoc. The specification will contain @effects, @modifies and @requires statements, unless specified differently in the exercise, or the started file uses different statements.

The comments inside the code should add additional information about the reasoning of the coded behaviour, but not the translation of the code to English.

Wrong example:

**private int** getSumOfDigits(**int** number) {  
 **int** sumOfDidgits = 0;  
 **while** (number > 0) {  
 *// Add to sumOfDidgits the remainder of the division of number by 10* sumOfDidgits += number % 10;  
 *// Divide number by 10* number /= 10;  
 }  
  
 **return** sumOfDidgits;  
}

Correct example:

**private int** getSumOfDigits(**int** number) {  
 **int** sumOfDidgits = 0;  
 **while** (number > 0) {  
 *// Add the least significant digit to sumOfDidgits* sumOfDidgits += number % 10;  
 *// Remove the least significant digit* number /= 10;  
 }  
  
 **return** sumOfDidgits;  
}