CS2103T Code Standard

Name

- **Names** representing packages should be in all **lower case**.
- **Class/enum** names must be nouns and written in **PascalCase**.
- **Variable** names must be in **camelCase**.
- **Constant** names must be all **uppercase** using underscore to separate words (aka SCREAMING_SNAKE_CASE).
- Names representing **methods** must be verbs and written in **camelCas**e.
- **Abbreviations and acronyms** should **not be uppercase** when used as a part of a name.

Good

```
exportHtmlSource();
openDvdPlayer();
```

Bad

```
exportHTMLSource();
openDVDPlayer();
```

- Variables with **a large scope** should have **long names**, variables with **a small scope** can have **short names**.
- Boolean variables/methods should be named to sound like Booleans (use a prefix such as is, has, was, etc.)
- Plural form should be used on **names** representing a collection of objects.
- Iterator variables can be called i, j, k etc.
- Associated constants should have a common prefix.

Layout

- Basic indentation should be **4 spaces (not tabs)**
- Line length should be **no longer than 120 chars (hard limit)**. Try to keep line length shorter than **110 chars (soft limit)**.
- Indentation for wrapped lines should be **8 spaces [NOT 2 TABS]** (i.e. twice the normal indentation of 4 spaces) more than the parent line.
- Place line break to **improve readability**
  - Break after a comma.
  - Break before an operator.

```
totalSum = a + b + c
        + d + e;
setText("Long line split"
        + "into two parts.");
method(param1,
        object.method()
                .method2(),
        param3);
```

  - A method or constructor name **stays attached to the open parenthesis (** that follows it.
  - Prefer higher-level breaks to lower-level breaks. In the example below, the first is preferred, since the break occurs outside the parenthesized expression, which is at a higher level.

Good

```
someMethodWithVeryVeryVeryVeryVeryVeryVeryVeryVeryVeryLongName(
        int anArg, Object anotherArg);
```

Bad

```
someMethodWithVeryVeryVeryVeryVeryVeryVeryVeryVeryVeryLongName
        (int anArg, Object anotherArg);
```

  - Prefer higher-level breaks to lower-level breaks.

Good

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

Bad

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

- Use **K&R style** brackets (aka [Egyptian style](#)).

Good

```
while (!done) {
    doSomething();
    done = moreToDo();
}
```

Bad

```
while (!done)
{
    doSomething();
    done = moreToDo();
}
```

- The while and the do-while statements should have the following form

```
while (condition) {
    statements;
}
```

```
do {
    statements;
} while (condition);
```

- The switch statement: No indentation for case clauses.
- The explicit //Fallthrough comment should be included whenever there is a case statement **without a break** statement.

```
switch (condition) {
case ABC:
    statements;
    // Fallthrough
```

- A try-catch statement should have the following form:

```
try {
    statements;
} catch (Exception exception) {
    statements;
} finally {
    statements;
}
```

- **White Space**
  - Operators should be surrounded by a space character.
  - Java reserved words should be followed by a white space.
  - Commas should be followed by a white space.
  - Colons should be surrounded by white space when used as a binary/ternary operator.
  - Does not apply to switch x:. Semicolons in for statements should be followed by a space character.

Good	Bad
`a = (b + c) * d;`	`a=(b+c)*d;`
`while (true) {`	`while(true){`
`doSomething(a, b, c, d);`	`doSomething(a,b,c,d);`
`for (i = 0; i < 10; i++) {`	`for(i=0;i<10;i++){`

- Logical units within a block should be separated by one blank line.

```
// Create a new identity matrix
Matrix4x4 matrix = new Matrix4x4();

// Precompute angles for efficiency
double cosAngle = Math.cos(angle);
double sinAngle = Math.sin(angle);

// Specify matrix as a rotation transformation
matrix.setElement(1, 1,  cosAngle);
matrix.setElement(1, 2,  sinAngle);
matrix.setElement(2, 1, -sinAngle);
matrix.setElement(2, 2,  cosAngle);

// Apply rotation
transformation.multiply(matrix);
```

Package and Import Statements

- Put every class in a package. Every class should be part of some package.
- The **ordering** of import statements must be consistent.
- Imported classes should always be **listed explicitly.**

Good

```
import java.util.List;
import java.util.ArrayList;
import java.util.HashSet;
```

Bad

```
import java.util.*;
```

Types: Array specifiers must be attached to the type not the variable.

Good

```
int[] a = new int[20];
```

Bad

```
int a[] = new int[20];
```

Variables

- Variables should be initialized where they are declared and they should be declared in the smallest scope possible.

Good

```
int sum = 0;
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        sum += i * j;
    }
}
```

Bad

```
int i, j, sum;
sum = 0;
for (i = 0; i < 10; i++) {
    for (j = 0; j < 10; j++) {
        sum += i * j;
    }
}
```

- Class variables should **never be declared public** unless the class is a data class with no behavior. This rule does not apply to constants.

Bad

```
public class Foo{

   public int bar;

}
```

Loops: The loop body should be **wrapped by curly brackets** irrespective of how many lines there are in the body.

Conditionals: The conditional should be put on a separate line.

Comments

- **MUST** write descriptive header comments for **all** public classes / methods. But they can be omitted for the following cases:
  - Getters/setters
  - When overriding methods (**provided the parent method's Javadoc** applies exactly as is to the overridden method)

```
/**
 * Returns lateral location of the specified position.
 * If the position is unset, NaN is returned.
 *
 * @param x X coordinate of position.
 * @param y Y coordinate of position.
 * @param zone Zone of position.
 * @return Lateral location.
 * @throws IllegalArgumentException If zone is <= 0.
 */
public double computeLocation(double x, double y, int zone)
        throws IllegalArgumentException {
    //...
}
```

- The opening /** on a separate line.
- Write the first sentence as a short summary [Javadoc automatically places it in the method summary table (and index)]
- In method header comments, the first sentence should start in the form **Returns ..., Sends ..., Adds** ... etc. (**not Return** or Returning etc.)
- Subsequent * is aligned with the first one.
- **Space** after each *.
- **Empty line** between description and parameter section.
- **Punctuation** behind each parameter description.
- **No blank line** between the documentation block and the method/class.
- **@return can be omitted** if the method does not return anything or the return value is obvious from the rest of the comment.
- **@params can be omitted** if all parameters of a method have **self-explanatory names** or already explained in the main part of the comment.
- When writing Javadocs for overridden methods, the **@inheritDoc tag** can be used to reuse the header comment from the parent method but with further modifications.