COS5005-B - **SOFTWARE ENGINEERING WITH GROUP PROJECT**

Coursework Assignment - 2

Coding Convention Document

Project :	**Web Browser**
Group Mentor:	**Dr. Noman Javed**
Group Members:	

1. Badar Shahzad Khan,
2. Mohammad Hassan Iqbal Khan,
3. Naeem Rashid,
4. Ramzan Shahid Khan,
5. Shoaib khan,
6. Sana ullah khan.

Introduction

This document explains the coding convention of java language which we are using in our project. If we want to maintain the project reliable and understandable then we have to follow java coding convention. For, that purpose we are following below mention tradition. It helps developer to understand what is happening in the class and method and what is actually purpose ot them.

Scope

This document explains the basic coding techniques and standards. These standards adopt all developers to code in a efficient way. These techniques help all future developers to collaborate and understand easily. This document helps developer to develop code which is easier to maintain and increase production.

File Organization

Java source files are saved as *.java and to compiled java byte code file is saved as *.class file. Each java source file contains one class in it and each class must be placed in separate file. Files longer than 2000 lines should be avoided.

Java source files should follow the following order:

1. Package.
2. Import statements
3. Beginning comments
4. Class and Interface Declarations.

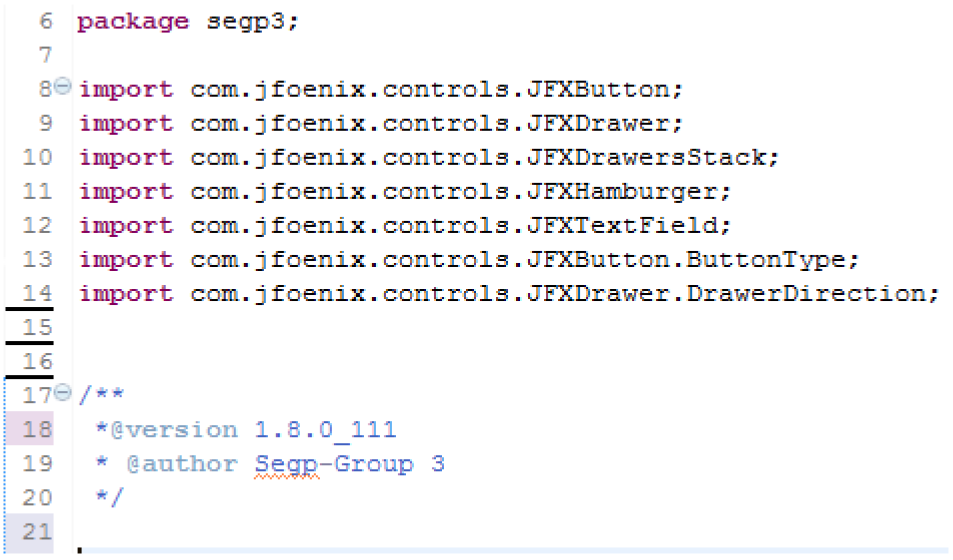
In java source code there should be no duplicate import statement. The code should not be hard coded and maximum number of parameters in any class should be *12.*

Source Code Style Guidelines:

The following steps should be followed while writing source code of any application.

1. Beginning Comments:

   The header comments should follow the name of package, then Import statements and then the documentation comments.



```java
package segp3;

import com.jfoenix.controls.JFXButton;
import com.jfoenix.controls.JFXDrawer;
import com.jfoenix.controls.JFXDrawersStack;
import com.jfoenix.controls.JFXHamburger;
import com.jfoenix.controls.JFXTextField;
import com.jfoenix.controls.JFXButton.ButtonType;
import com.jfoenix.controls.JFXDrawer.DrawerDirection;


/**
 *@version 1.8.0_111
 * @author Segp-Group 3
 */
```

2. Package and Import Statements:

   The first non-commit line of most of the java source file is a package statement and after that import statement follows. For example:

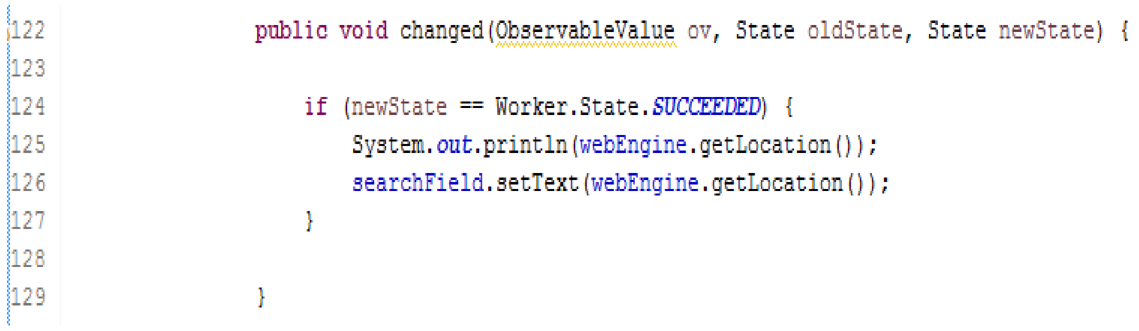
```java
package downloader;

import java.io.BufferedInputStream;
import java.io.File;
```

3. Indentation:

   Four spaces should be used as the unit of indentation and this pattern should be followed throughout the document.

4. Left and Right braces:

   The starting brace should be at the end of the conditional statement and the ending brace should be at the new line aligned with the conditional statement.



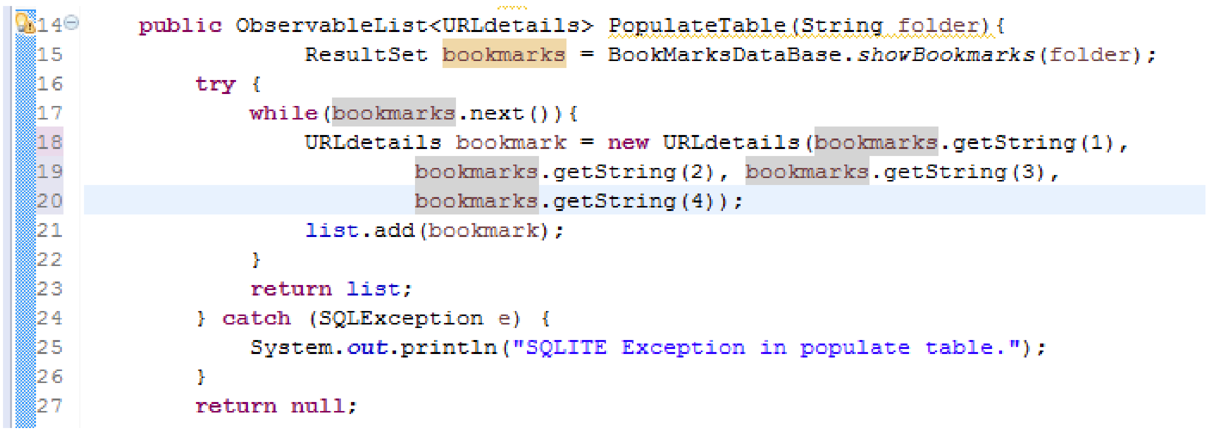
```
            public void changed(ObservableValue ov, State oldState, State newState) {

                if (newState == Worker.State.SUCCEEDED) {
                    System.out.println(webEngine.getLocation());
                    searchField.setText(webEngine.getLocation());
                }

            }
```

5. Wrapping lines:

   Lines longer than 80 character should be avoided. If an expression does not fit on same line it should be broken according to following principles:

   - Break after a comma.
   - Break after an operator.
   - Prefer higher level breaks to lower level breaks.

   Following picture shows the break point on “,”.

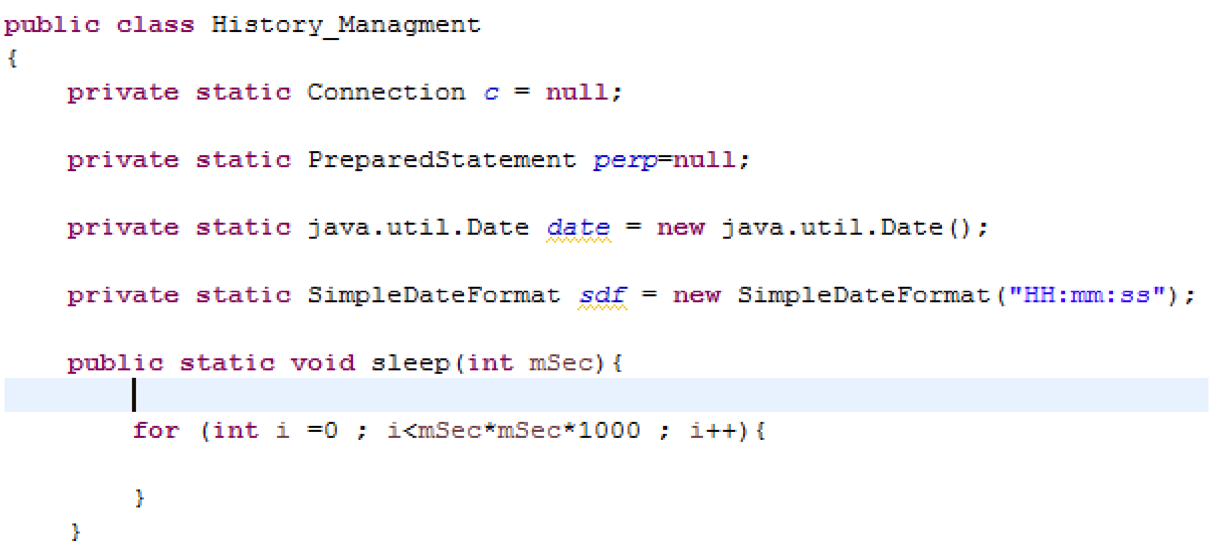


```
    public ObservableList<URLdetails> PopulateTable(String folder){
                ResultSet bookmarks = BookMarksDataBase.showBookmarks(folder);
        try {
            while(bookmarks.next()){
                URLdetails bookmark = new URLdetails(bookmarks.getString(1),
                        bookmarks.getString(2), bookmarks.getString(3),
                        bookmarks.getString(4));
                list.add(bookmark);
            }
            return list;
        } catch (SQLException e) {
            System.out.println("SQLITE Exception in populate table.");
        }
        return null;
```

6. White space:

   White spaces or blank lines improve the readability of the code.
   One blank line should be used in following cases:

   1. Between methods.
   2. Between local variables in a method.
   3. Before a block or single line comment.

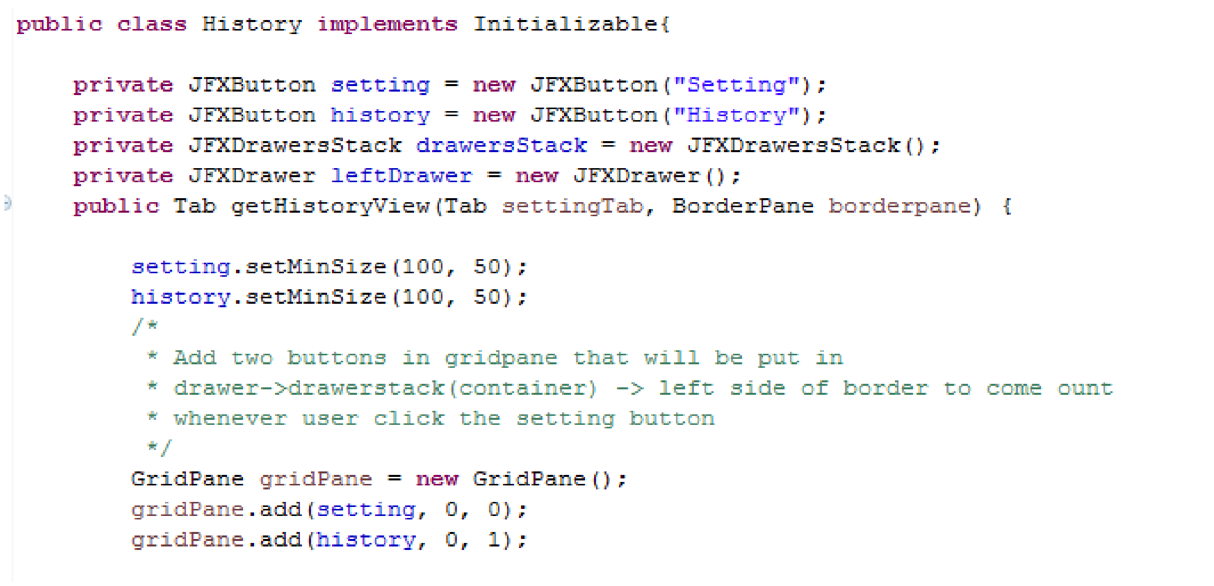

```
public class History_Managment
{
    private static Connection c = null;

    private static PreparedStatement perp=null;

    private static java.util.Date date = new java.util.Date();

    private static SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");

    public static void sleep(int mSec){

        for (int i =0 ; i<mSec*mSec*1000 ; i++){

        }
    }
```

7. Implementation comments:

   Comments should be written in following ways:

   1. Block Comments:

      Block comments are used to describe files, methods, data structures and algorithms. Block comments should be used at start of file and when you are going to describe any code. Block comment can be written in method or before method. Block comment should have blank line at start and at end. Block comments are written as /*.....*/. Below the figure is an example of block comments.

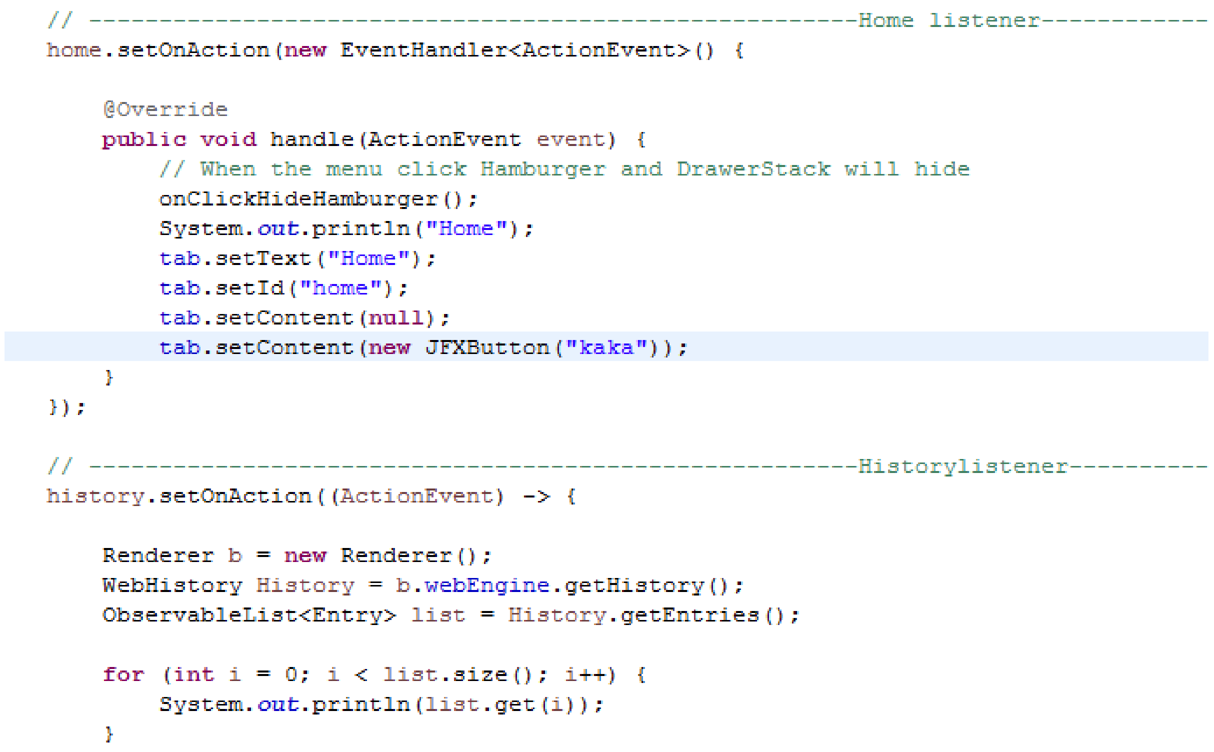

```
public class History implements Initializable{

    private JFXButton setting = new JFXButton("Setting");
    private JFXButton history = new JFXButton("History");
    private JFXDrawersStack drawersStack = new JFXDrawersStack();
    private JFXDrawer leftDrawer = new JFXDrawer();
    public Tab getHistoryView(Tab settingTab, BorderPane borderpane) {

        setting.setMinSize(100, 50);
        history.setMinSize(100, 50);
        /*
         * Add two buttons in gridpane that will be put in
         * drawer->drawerstack(container) -> left side of border to come ount
         * whenever user click the setting button
         */
        GridPane gridPane = new GridPane();
        gridPane.add(setting, 0, 0);
        gridPane.add(history, 0, 1);
```

2. Single Line Comment:

There are some comments which can be explained in single line. We can also use /*....*/ to write single line comments or we can use “//” too.

3. Commenting Code:

The // delimiter is used to comment a complete line . It can not be used on multiple lines for text comments. But it can be used on multiple lines for commenting sections of codes.



```
// -------------------------------------------------------Home listener------------
home.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {
        // When the menu click Hamburger and DrawerStack will hide
        onClickHideHamburger();
        System.out.println("Home");
        tab.setText("Home");
        tab.setId("home");
        tab.setContent(null);
        tab.setContent(new JFXButton("kaka"));
    }
});

// -------------------------------------------------------Historylistener----------
history.setOnAction((ActionEvent) -> {

    Renderer b = new Renderer();
    WebHistory History = b.webEngine.getHistory();
    ObservableList<Entry> list = History.getEntries();

    for (int i = 0; i < list.size(); i++) {
        System.out.println(list.get(i));
    }
```

8. Variables:

First the static variables should be declared in the sequence. Static variables should be defined before the methods in the classes. Then protected variables, then package level variables and then private variables followed by instance variables in the same sequence.

9. Methods:

Each method should follow javadoc tags. Each item line starts with asterisk. In multiple line comment each line is intended so that line up vertically with previous line.

10. Declarations:

One declaration per line is recommended. Following steps should be followed to declare the code.

1. Instance variables should be declared in following sequence.
   - Public instance variables.
   - Protected variables.
   - Package level variables.
   - Private variables.

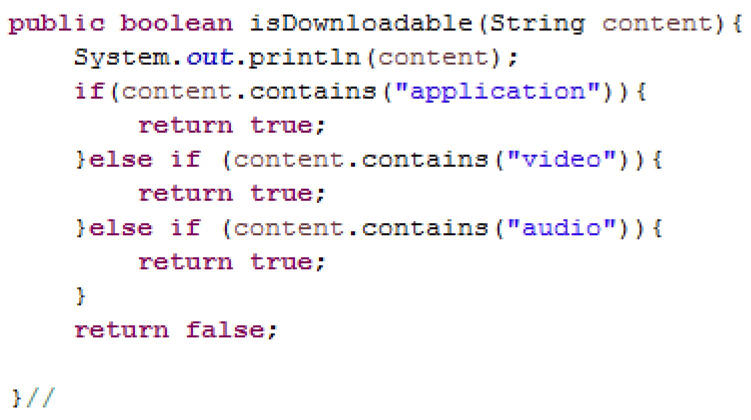
2. Next the class constructor should be declared.
3. Now if inner classes are available, declare them.
4. Method should be declared in grouped with functionality rather than scope or accessibility.
5. Declaration for local variables should be only in the beginning of blocks .Like as in try catch statements we declare the variable outside the try catch.
6. Numerical constants should not be declared directly.

11. Standards for statements:

Each line should contain maximum one statement. If their are many statements than they should appear in braces. Braces should be used around all statements especially when they are part of the for condition or if condition.

1. If,if-else:

   The ‘if’ keyword and conditional expression must be placed on the same line. The ‘if’ statement must use braces{}.



```
public boolean isDownloadable(String content){
    System.out.println(content);
    if(content.contains("application")){
        return true;
    }else if (content.contains("video")){
        return true;
    }else if (content.contains("audio")){
        return true;
    }
    return false;

}//
```

2. For statement:

   When you are using comma between initialization and the update clause for statement, avoid using many variables. The key word for and the parenthesis should be separated by one space. The block statement is written on new line and closing braces should start in new line intended to match its corresponding opening statement.

```
Renderer b = new Renderer();
WebHistory History = b.webEngine.getHistory();
ObservableList<Entry> list = History.getEntries();

for (int i = 0; i < list.size(); i++) {
    System.out.println(list.get(i));
}

// When the menu click Hamburger and DrawerStack will hide
onClickHideHamburger();
```

3. While statement:

   The while condition uses the same layout as ‘if’ statement. The key word while and parenthesis should be separated by one space. Statement block is placed on new line same as ‘for’ statement block.

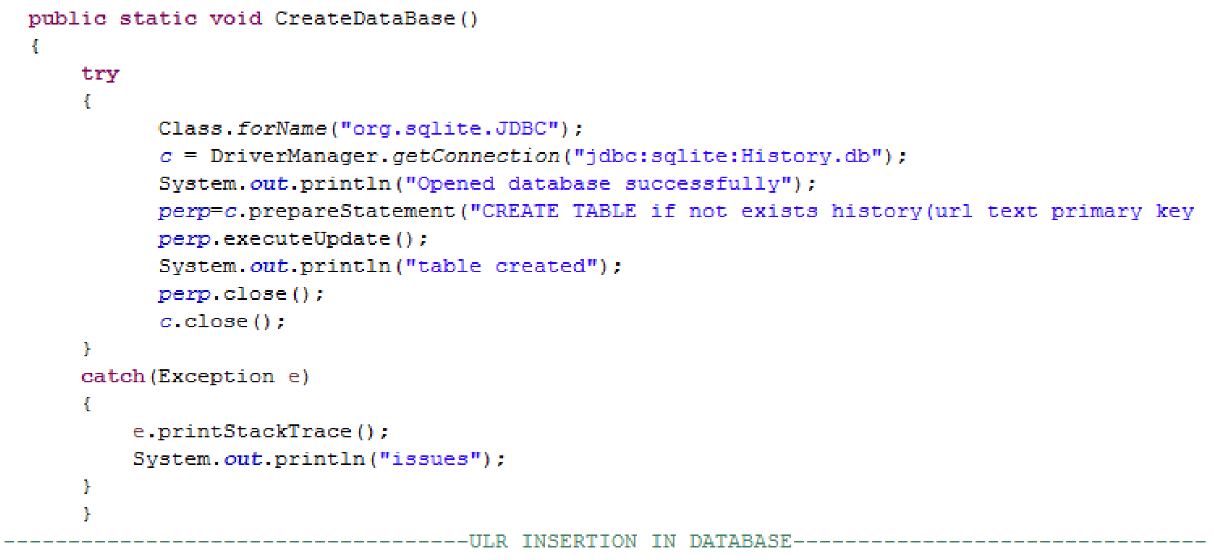
4. Do-while statement:

   Do-while should have the following form:

```
public static void main(String[] args){
    int count = 1;
    do {
        System.out.println("Count is: " + count);
        count++;
    } while (count < 11);
}
```

5. Try-catch statement:

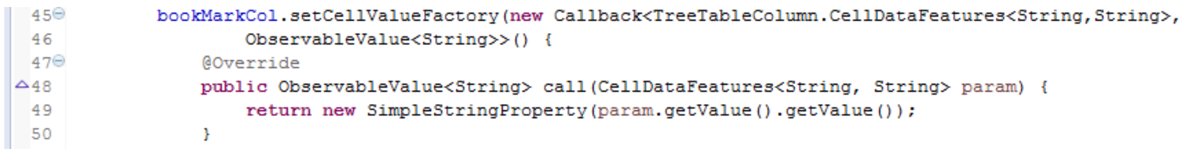
   Try catch statement should be written in following format.



```
public static void CreateDataBase()
{
    try
    {
            Class.forName("org.sqlite.JDBC");
            c = DriverManager.getConnection("jdbc:sqlite:History.db");
            System.out.println("Opened database successfully");
            perp=c.prepareStatement("CREATE TABLE if not exists history(url text primary key
            perp.executeUpdate();
            System.out.println("table created");
            perp.close();
            c.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
        System.out.println("issues");
    }
    }
------------------------------------ULR INSERTION IN DATABASE--------------------------------
```

6. Return statement:

This statement should not use any braces unless the return statement is larger.



```
        bookMarkCol.setCellValueFactory(new Callback<TreeTableColumn.CellDataFeatures<String,String>,
                ObservableValue<String>>() {
            @Override
            public ObservableValue<String> call(CellDataFeatures<String, String> param) {
                return new SimpleStringProperty(param.getValue().getValue());
            }
```

References

- http://www.oracle.com/technetwork/java/codeconventions-135099.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/returnvalue.html