

Coding Standards for Java

An Introduction

Why Coding Standards are Important?

- Coding Standards lead to **greater consistency** within your code and the code of your teammates.
- Easier to understand
- Easier to develop
- Easier to maintain
- Reduces overall cost of application

Components of Java Source File

```
/*  
 * Copyright notice  
 */
```

Beginning Comments

```
package java.awt;
```

```
import java.awt.peer.CanvasPeer;
```

Package and Import Statements

```
/**  
 * class description  
 *  
 * @version 1.10 04 Oct 1996  
 *  
 * @author First name Last name  
 */
```

Class/interface documentation comment (/**...*/)

```
public class ColorPickerPanel {
```

class or interface statement

```
/* A class implementation comment can go here. */
```

Class/interface implementation comment (/ *...*/), if necessary

```
/**  
 *class variables – doc comment  
 */
```

```
public static Integer colorVariant;
```

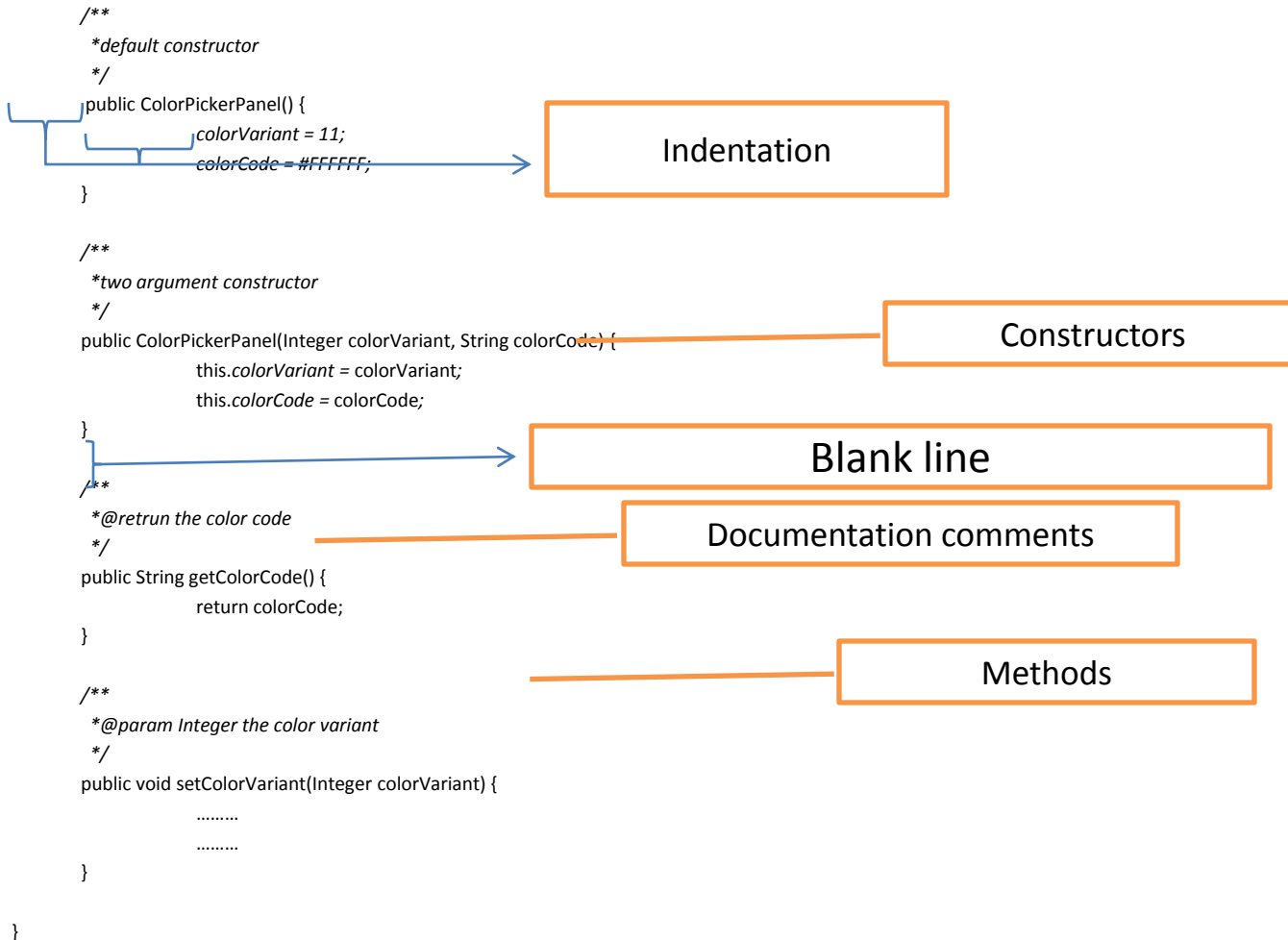
Class (static) variables

```
/**  
 *instance variables – doc comment  
 */
```

```
private String colorCode;
```

Instance variables

Components of Java Source File continued...



Comments

```
/**
 * class description
 * @version 1.10 04 Oct 1996
 * @author First name Last name
 */
```

Documentation Comments

```
public class ColorPickerPanel {
```

```
    /* A class implementation comment can go here. */
```

Single Line Comments

```
    /*
    private static final String DEFAULT_COLOR = "#FFFFFF"
    private static final int DEFAULT_VARIANT = 0;
    */
```

Block Comments

```
    /**
     *class variables – doc comment
     */
    public static Integer colorVariant;
```

```
    /**
     *instance variables – doc comment
     */
    private String colorCode;
```

Implementation Comments

```
    /**
     *@param Integer the color variant
     */
```

```
    public void setColorVariant(Integer colorVariant) {
        if (colorVariant == 0) {
        } else {
        }
    }
```

```
        colorCode = "#000000"; /* set the color to black*/
```

Trailing Comments

```
        colorCode = "#FFFFFF"; // set the color to white
```

End-of-Line Comments

```
}
```

Naming Conventions

What Makes Up a Good Name?

Use full English descriptors

For example, use names like **firstName**, **grandTotal**, or **CorporateCustomer**

Use terminology applicable to the domain

*Banking domain - **Customer**, Software services domain - **Client***

Use mixed case to make names readable

Avoid long names (< 15 characters is a good idea)

User abbreviations sparingly

Capitalize the first letter of standard acronym

Naming Conventions Continued...

Classes / Interfaces –

Class names should be nouns, in mixed case with the first letter of each internal word capitalized.

Methods –

Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

Variables –

Variables should be nouns, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

Class Constants –


Class Constants should be all uppercase with words separated by underscores (“_”).

Blank Spaces

Before / After Parenthesis –

A keyword followed by a parenthesis should be separated by a space.

```
while_(true){  
    ...  
}
```



blank spaces

A blank space should appear after commas in argument lists.

All binary operators except . should be separated from their operands by spaces.

```
a = (a + b) / (c * d);
```

The expressions in a for statement should be separated by blank spaces.

```
for (expr1; expr2; expr3)
```

Casts should be followed by a blank.

```
User x = (User) anObject;
```


Returning Values

Try to make the structure of your program match the intent.

Example:

```
if (booleanExpression) {  
    return TRUE;  
} else {  
    return FALSE;  
}
```

should instead be written as

```
return booleanExpression;
```

Ternary Operator (?:)

Use Ternary Operator for conditional assignment

Example:

```
Int x;  
If (expression) {  
    x = 9;  
} else {  
    x = 0;  
}
```

can be written as

```
x = (expression) ? 9 : 0;
```

much more to learn ...

References

Google...

Ambler's Law of Standards

Industry standards > organizational standards > project standards > personal standards > no standards

so, what is the lesson to learn?

Whenever possible, reuse standards and guidelines,
don't reinvent them

Static Code Analysis

Static code analysis is the analysis of computer software that is performed without actually executing programs built from that software.
(analysis performed on executing programs is known as dynamic analysis)

In most cases the analysis is performed on some version of the source code and in the other cases some form of the object code.

The term is usually applied to the analysis performed by an automated tool, with human analysis being called program understanding, or code review.

Some of automated SCA Tools –

PMD, AppPerfect, FindBugs, IntelliJ IDEA etc.

PMD

<http://pmd.sourceforge.net/>

PMD scans Java source code and looks for potential problems like:

Possible bugs - empty try/catch/finally/switch statements

Dead code - unused local variables, parameters and private methods

Suboptimal code - wasteful String/StringBuffer usage

Overcomplicated expressions - unnecessary if statements, for loops that could be while loops

Duplicate code - copied/pasted code means copied/pasted bugs

PMD – Rule Set for SCA

<http://pmd.sourceforge.net/rules/index.html>

Basic JSP rules

NoLongScripts: Scripts should be part of Tag Libraries, rather than part of JSP pages.

NoScriptlets: Scriptlets should be factored into Tag Libraries or JSP declarations, rather than being part of JSP pages.

NoInlineStyleInformation: Style information should be put in CSS files, not in JSPs. Therefore, don't use or tags, or attributes like "align='center'".

NoClassAttribute: Do not use an attribute called 'class'. Use "styleclass" for CSS styles.

NoJspForward: Do not do a forward from within a JSP file.

AppPerfect Java Code Test

AppPerfect Java Code Test is a static Java code analysis software designed to perform the following two key tasks: *Automate Java code review* and *Enforce Good Java Coding Practices*.

AppPerfect Code Test analysis your Java and Java Server Pages (JSP) source code and applies *over 750 Java coding rules* to apply the collective knowledge of leading experts in the Java programming field to your code.

Some Rules –

- Avoid method calls in loop

- Declare methods not using instance variables static

- User equals method instead of equality operator

- Etc.

AppPerfect Java Code Test

a screen shot

The screenshot displays the AppPerfect Java Code Test interface for a project named 'PetstoreProject'. The left sidebar shows a tree view of the project structure with 48 violations. The main window shows the source code of 'CustomerList.java' with 5 violations highlighted. The bottom panel shows a table of violations.

Project Structure (Left Sidebar):

- PetstoreProject (48 violations)
 - Server Pages (5 violations)
 - Source Files (43 violations)
 - com.appperfect.pe
 - CheckForm.java
 - Constants.java
 - CreateAction.java
 - CreateCheckAction.java
 - CreateLoginAction.java
 - CreditCardInfo.java
 - Customer.java
 - CustomerForm.java
 - CustomerInfo.java
 - CustomerList.java (5 violations)**
 - CustomerMode.java
 - EditAction.java
 - EditForm.java
 - ForwardAction.java
 - Login.java (0 violations)
 - LoginForm.java
 - LoginVariable.java
 - Node.java (0 violations)
 - PersonalInfo.java

Source Code (Main Window):

```
Node temp = head;
while (temp != null)
{
    if (temp.getLogin().getUserName() == userName
        && temp.getLogin().getPassword() == passWord)
    {
        // ...
    }
    return 0;
}
```

Violations Table (Bottom Panel):

Source	Violations	Severity	Category	Line Number
CreateAction.java	0			
CreateCheckAction.java	0			
CreateLoginAction.java	0			
CreditCardInfo.java	0			
Customer.java	0			
CustomerForm.java	0			
CustomerInfo.java	0			
CustomerList.java	5			
Avoid_method_calls_in_loop		High	Optimization	42
Declare_methods_not_using_instance_m...		Low	OOPS	13
Declare_methods_not_using_instance_m...		Low	OOPS	18
Use_Equals_Instead_Equality_Operator		Medium	PossibleErrors	56
Use_Equals_Instead_Equality_Operator		Medium	PossibleErrors	57

Status Bar: Professional | Finished Analyzing | 00:00:12 | Aug 2 | 40M of 63M

Queries!

Google...

Thank You
Mahesh Babu M