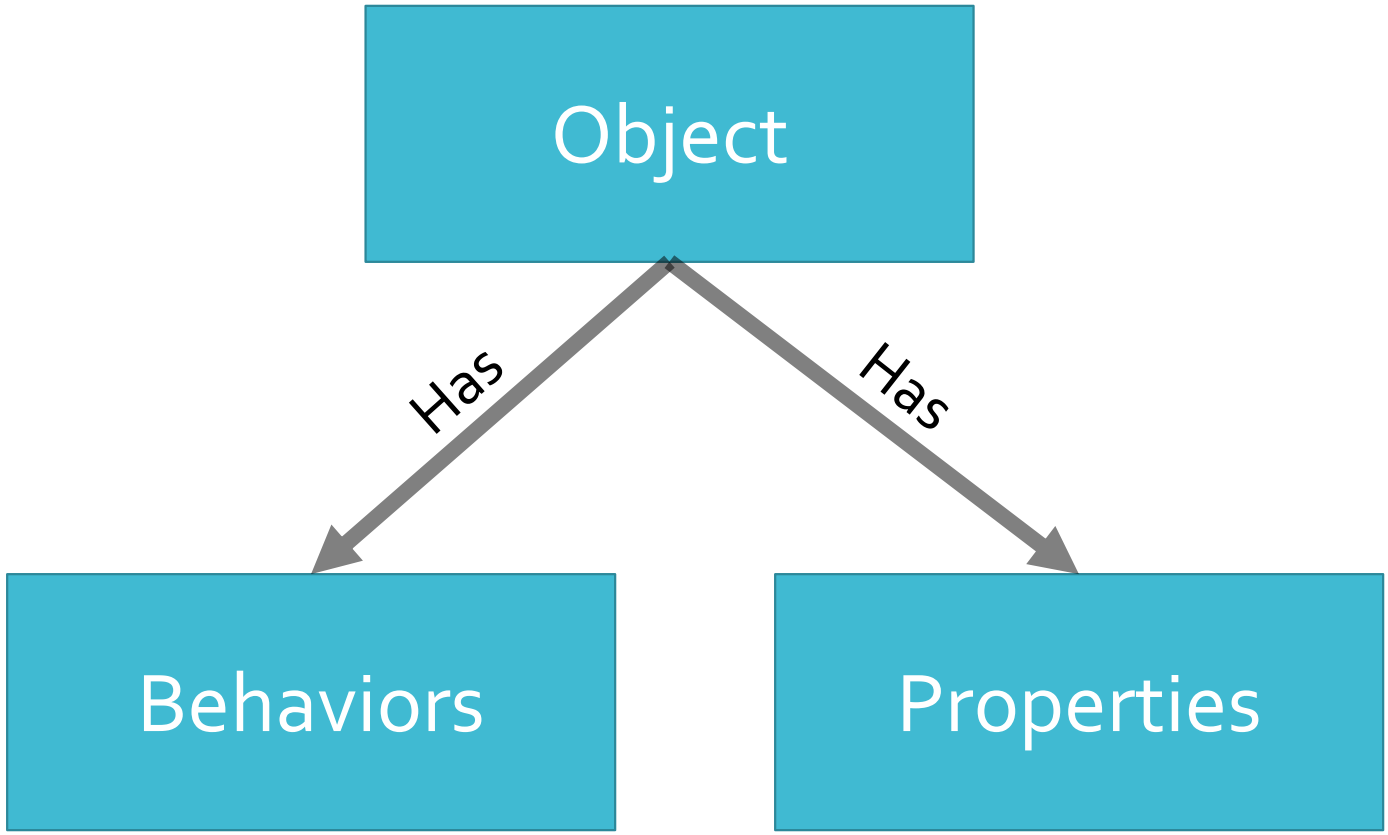
Class in Java & Coding Convention

Goals

- Learn what is coding convention
- Begin coding with class,
  - fields
  - methods
  - constructors
- Write a program with coding convention

Review

Objects



Declaring and using a class

```
public class ClassName {
    // fields  Properties
    fieldType fieldName;

    // methods Behaviors
    public returnType methodName() {
        // statements;
        return returnType;
    }
}

public void someMethod(){
    ClassName object = new ClassName();
    returnType ret = object.methodName();
}
```

Coding convention

Coding Convention (What & Why)

- Coding Convention is the rule that led to greater consistency within your code and the code of your teammates.
  - makes maintenance of your code a lot easier
  - improve the readability
  - reduce training management and effort
  - avoid junior mistakes.
  - result in a correct entered JavaDoc output
- Different places where the Conventions can be applied
  - Naming Conventions
  - Comments Conventions

Any code is 20% of its time is written and 80% time is read, so write it well

Naming Conventions

- WRONG
  - `public class _HelloWorld{ }`
  - `void PRINT(){`
- RIGHT
  - `public class HelloWorld { }`
  - `void printName(){`

- **Class names**
  - should be nouns,
  - in mixed case with the first letter of each internal word capitalized. Also known as the CamelNotation.
- **Method name**
  - should be verb
  - in mixed case with the first letter lowercase, with the first letter of each internal word capitalized

Naming Conventions

- WRONG
  - `int AMOUNT = 100;`
  - `public static final int heightX = 100;`
  - `package learning.com.java.algorithms._functions;`
- RIGHT
  - `int amount = 100;`
  - `public static final int HEIGHT_X = 100;`
  - `package learning.com.programs.algorithms.functions;`

- **Variables**
  - should be short yet meaningful.
  - Non final-name start with a lower-case letter and internal words start with capital letters.
- **Constant**
  - Constant of should contain only upper-case letters and underscores.

Naming Conventions

- WRONG
  - ```
    void foo(double d) {
      char s;
      Object f;
      String k;
      Object UK;
    }
    ```
- RIGHT
  - ```
    void foo(double d) {
      char cType;
      Object oVehicle;
      String sName;
    }
    ```

- One-character local variable

Naming Conventions

- WRONG
  - ```
    Class AmountTransfer{
      private int amount1,amount2,amount3;
      //...
    }
    ```
- RIGHT
  - ```
    Class AmountTransfer{
      private int amount1;
      private int amount2;
      private int amount3;
      //...
    }
    ```

- Do not declare multiple variables in one statement

Comment Conventions

```
/*
 * Copyright notice
 */
package lab3;
/** * class description
 * @version 1.10 04 March 2014
 * @author First name Last name
 */
public class Student {
 /* A class implementation comment can go here. */
 /**
  * class variables – doc comment
  */
 private int stdId;
 /**
  * instance variables – doc comment
  */
 public String stdName;
```

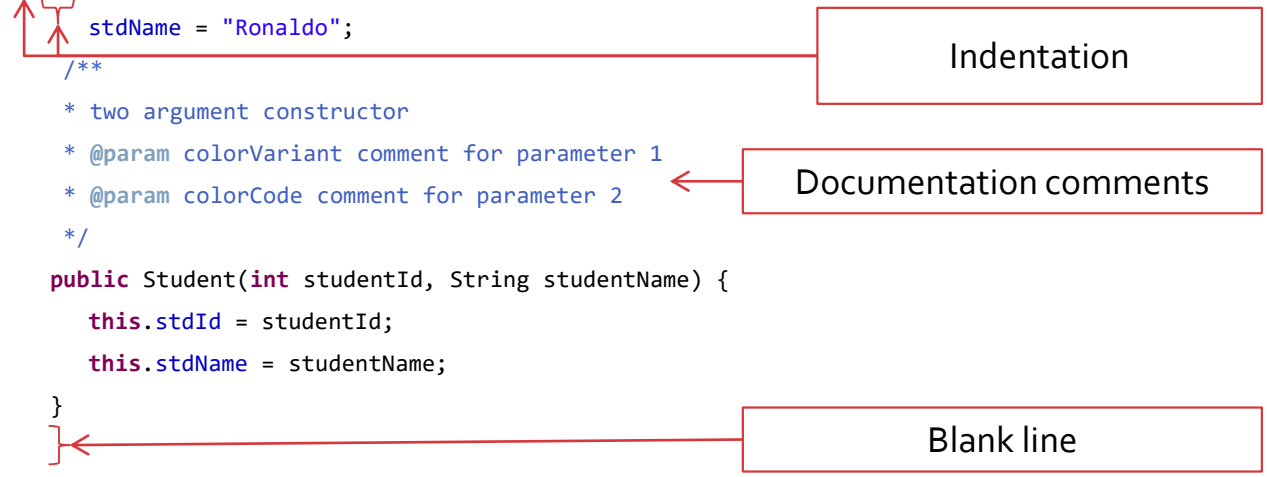
Beginning Comments

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

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

Comment Conventions

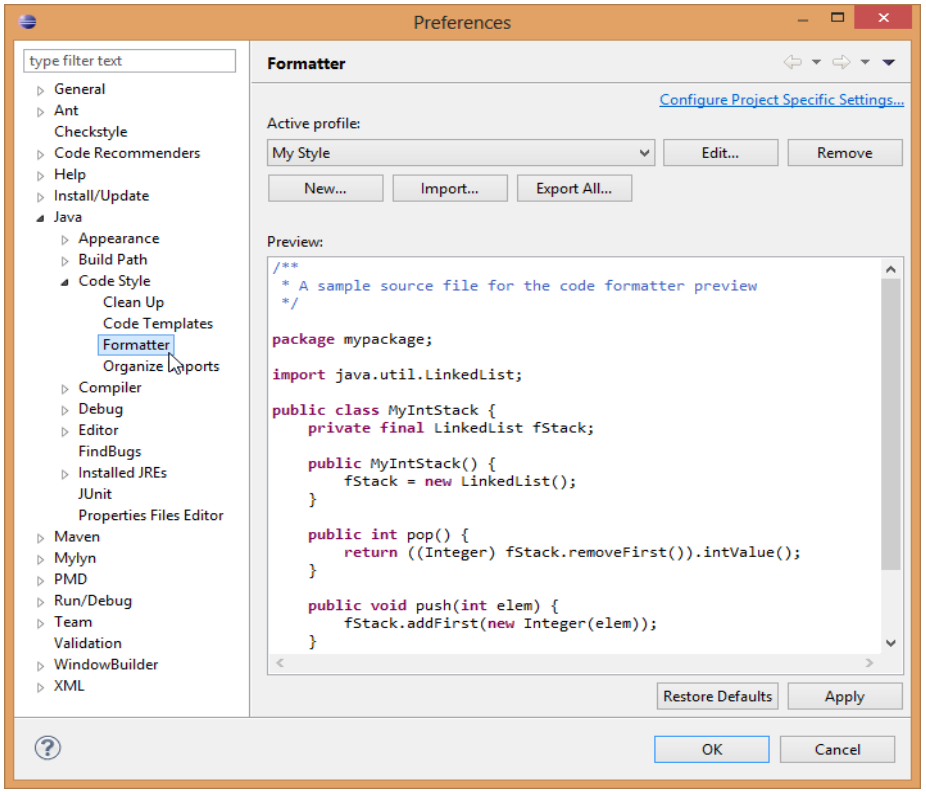
```java
/**
 * default constructor
 */
public Student() {
   stdId = 7;
   stdName = "Ronaldo";
}
/**
 * two argument constructor
 * @param colorVariant comment for parameter 1
 * @param colorCode comment for parameter 2
 */
public Student(int studentId, String studentName) {
   this.stdId = studentId;
   this.stdName = studentName;
}

/**
 * @return the student identity
 */
public int getStudentId() {
   return stdId;
}
/**
 * @param studentId student identity
 */
public void setStudentId(int studentId) {
   stdId = studentId; //inline comment here
}
```



Coding convention in Eclipse

Eclipse built-in Formatter

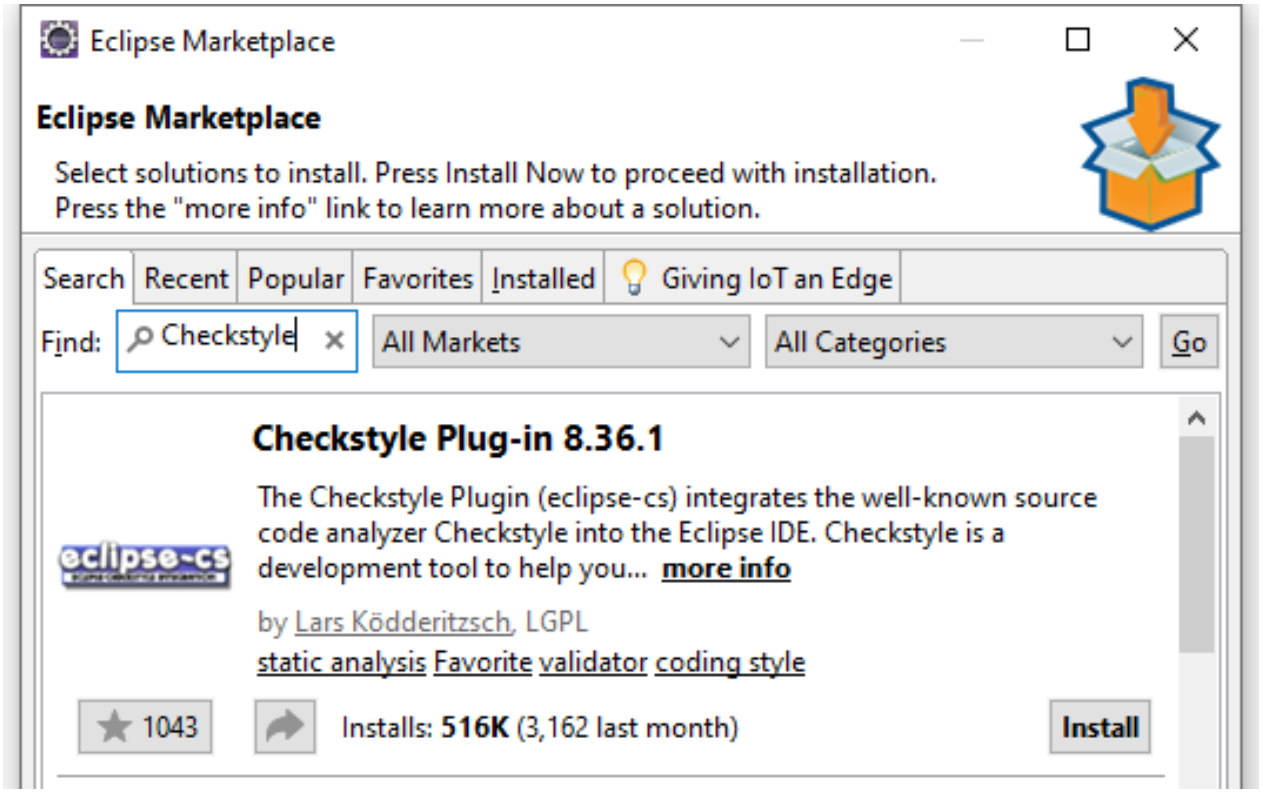
- Setting
  - Window -> Preference -> Java -> Code style -> Formatter
- Format your code:
  - Source -> Format
  - **Ctrl + Shift + F**



Eclipse plugin: Checkstyle

1. Installation:

- Search and install from Eclipse market (Help -> Eclipse Market)



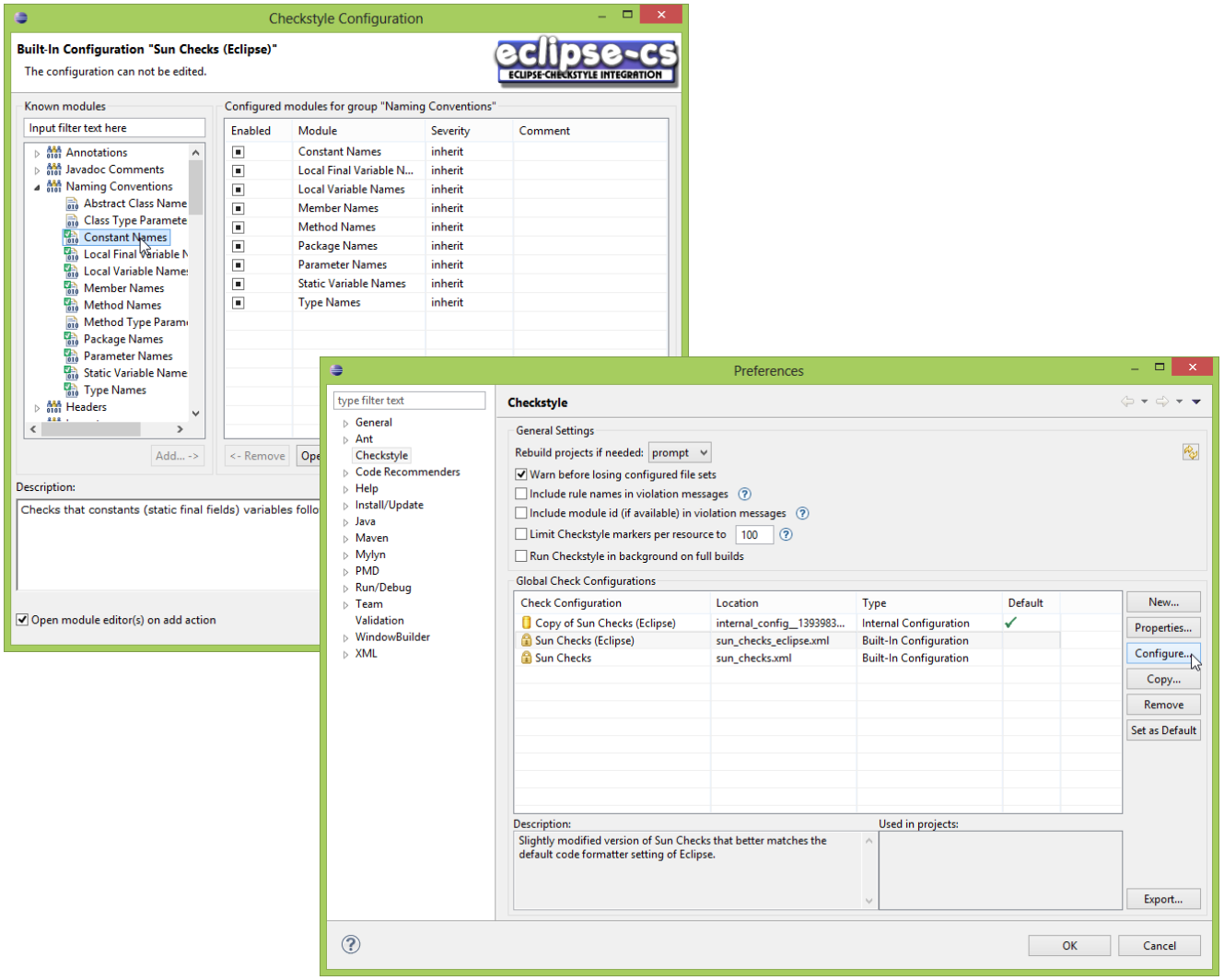
Checkstyle

2. Configure

To configure the style: Window-> Preferences -> Checkstyle

Click Configure if you want to change rule.

Click “Set as Default” after selecting right entry

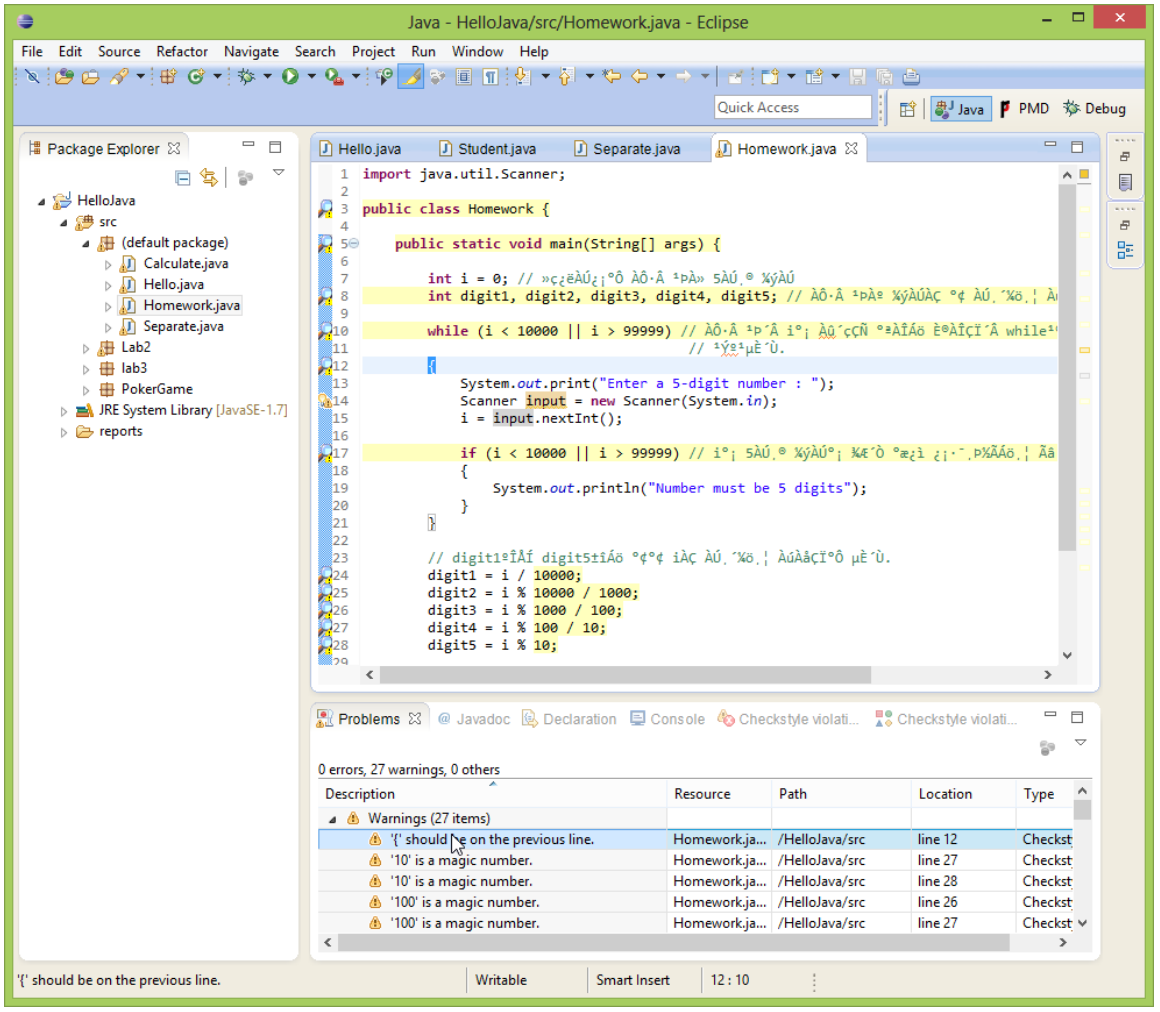


Checkstyle

3. Using Checkstyle in your project

Right click on project or file -> Checkstyle -> Check code with Checkstyle

The violation display on [Problems] window



Exercise

Exercise

- Create the Rectangle class include
  - 2 members: height, width
  - 2 constructors: no input parameters & input the height and width
  - 2 methods which set the height and width of the rectangle. It should verify in the range (from 0.0 to 20.0)
  - 2 methods which calculate the perimeter and area of the rectangle
  - 1 method to print all the value of the rectangle.
- Write the program to test class Rectangle

```
Rectangle created
Hight =1.0 Width =1.0
1. Set Length
2. Set Width
3. Exit
Choice: 1
Enter height: 10
Length: 10.000000
Width: 1.000000
Perimeter: 22.000000
Area: 10.000000
---------------------------
1. Set Length
2. Set Width
3. Exit
Choice: 2
Enter width: 10
Length: 10.000000
Width: 10.000000
Perimeter: 40.000000
Area: 100.000000
---------------------------
1. Set Length
2. Set Width
3. Exit
Choice: 3
```

Code optimization

Code optimization

String abc = "abc"; String def = "def";

- BAD

```
if ( (abc + def) == "abcdef" ) {
    ......
}
```

- GOOD:

```
if ( (abc + def).equals("abcdef") ){
   .....
}
```

Code optimization

- BAD:

```
if (m ==  1) System.out.println("Jan");
else if (m ==  2) System.out.println("Feb");
else if (m ==  3) System.out.println("Mar");
else if (m ==  4) System.out.println("Apr");
else if (m ==  5) System.out.println("May");
else if (m ==  6) System.out.println("Jun");
else if (m ==  7) System.out.println("Jul");
else if (m ==  8) System.out.println("Aug");
else if (m ==  9) System.out.println("Sep");
else if (m == 10) System.out.println("Oct");
else if (m == 11) System.out.println("Nov");
else if (m == 12) System.out.println("Dec");
```

- GOOD:

```
String[] months = { "", "Jan", "Feb", "Mar",
"Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec" }; ...
System.out.println(months[m]);
```

Code optimization

- BAD:

```
final int  COLOR_RED   = 1;
final int  COLOR_GREEN = 2;
final int  COLOR_BLUE  = 3;
```

- GOOD:

```
interface Color {
  final int RED   = 1;
  final int GREEN = 2;
  final int BLUE  = 3;
}
```

Code optimization

- BAD:

```
for (int i = 0; i <= 1000; i++) {
   double ret = 0;
   // do something ...
}
```

- GOOD:

```
double ret;
for (int i = 0; i <= 1000; i++) {
   ret = 0;
   // do something ...
}
```