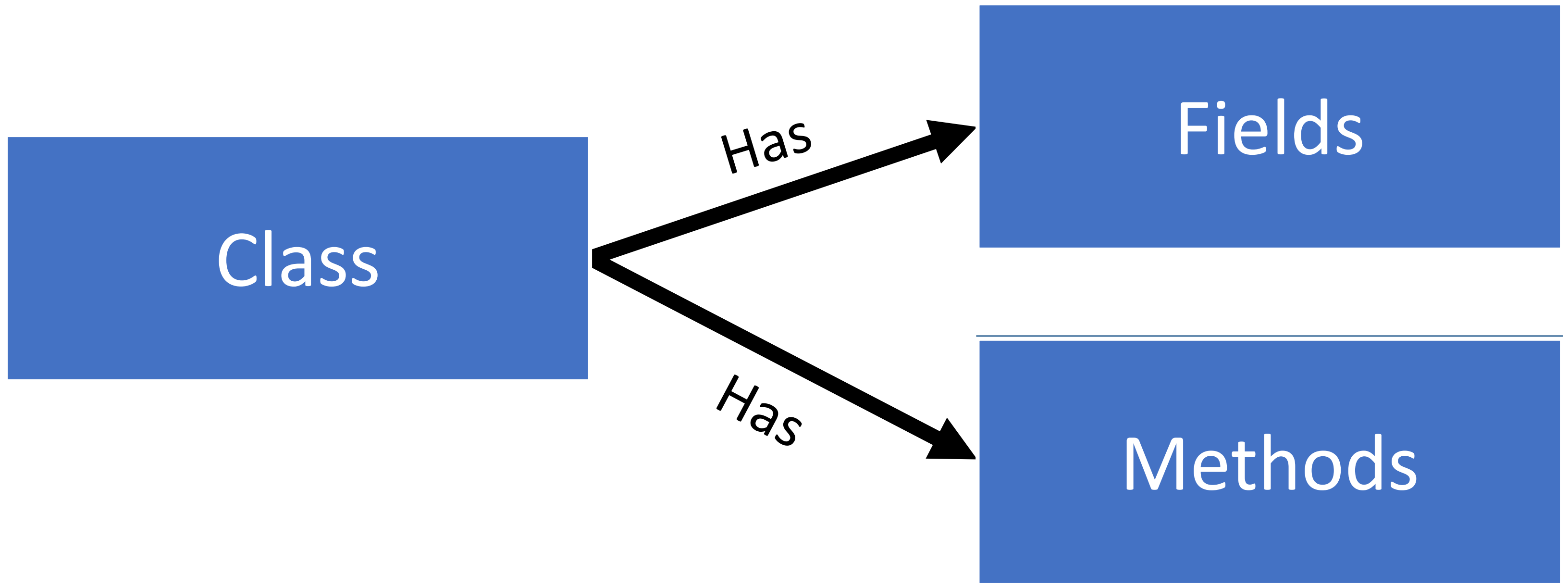
Lab 3:
Class & Coding Convention

Goals

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

I. CLASS

Objects & Classes



Declaring and using a class

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

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

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

II. CODING CONVENTION

Coding Convention (What & Why)

- Coding Convention is the rule lead 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 (2)

- 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.

Assignment Conventions (3)

- WRONG
  - `fooBar.fChar = barFoo.lchar = 'c’;`
  - `d = (a = b + c) + r;`
- RIGHT
  - `fooBar.fChar = 'c';`
  - `barFoo.lchar = 'c’;`
  - `a = b + c;`
  - `d = a + r;`

- Avoid assigning several variables to the same value in a single statement. It is hard to read.

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 (2)

```
 /**
  * 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
}
```

Indentation

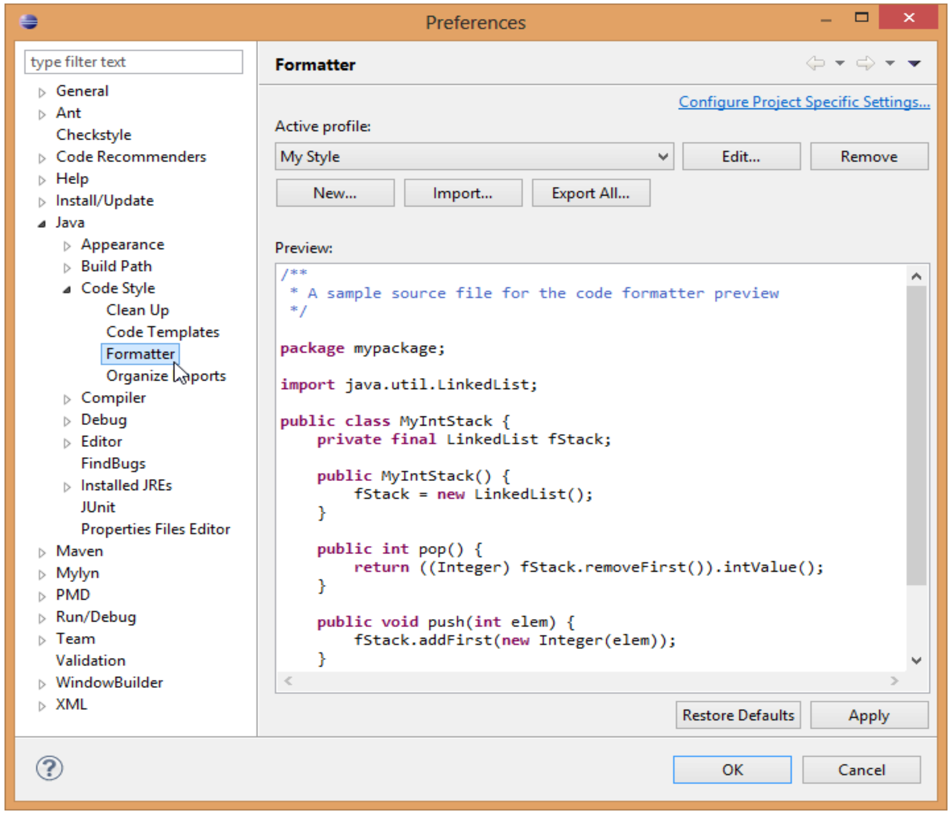
Documentation comments

Blank line

III. CODING CONVENTION IN ECLIPSE

Eclipse built-in Formatter

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



Active profile: My Style

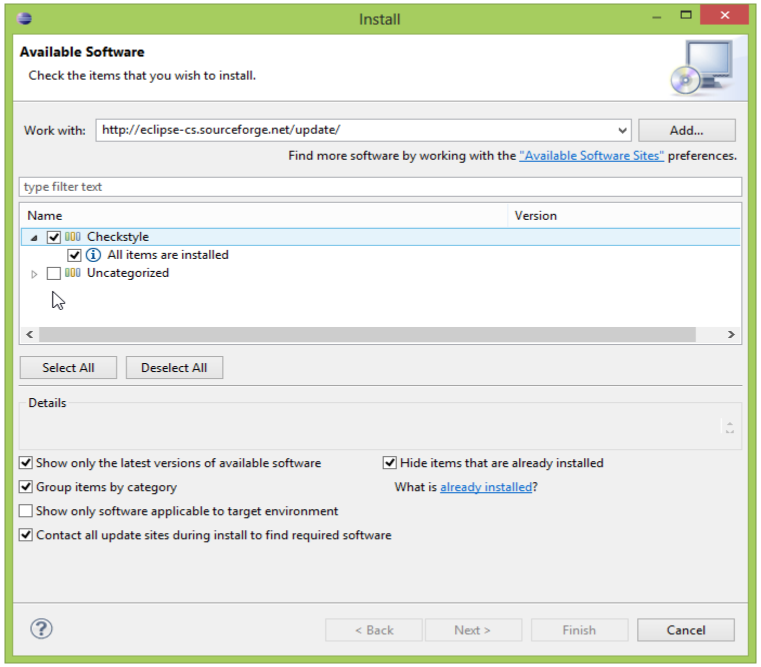
Preview:

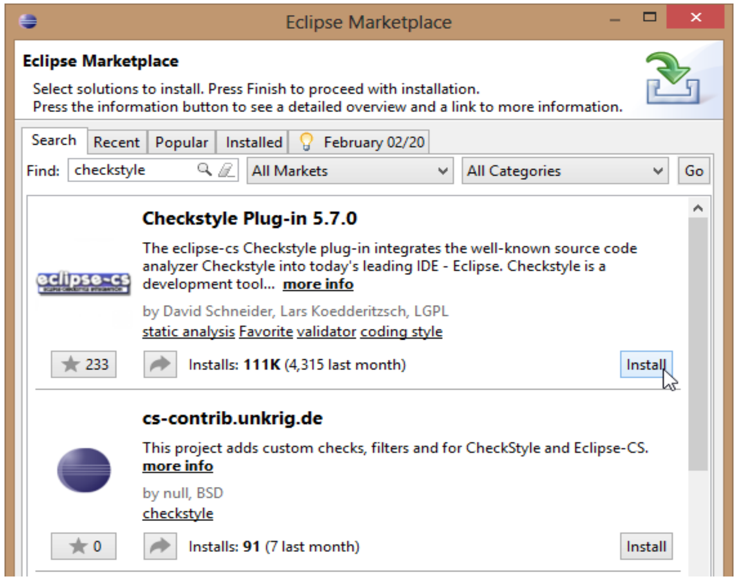
```
/**
 * A sample source file for the code formatter preview
 */

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
    private final LinkedList fStack;

    public MyIntStack() {
        fStack = new LinkedList();
    }

    public int pop() {
        return ((Integer) fStack.removeFirst()).intValue();
    }

    public void push(int elem) {
        fStack.addFirst(new Integer(elem));
    }
```

Eclipse plugin: Checkstyle

1. Installation:

- http://eclipse-cs.sourceforge.net/update/ update site (Help -> Install Sofware) or
- Search and install from Eclipse market (Help -> Eclipse Market)

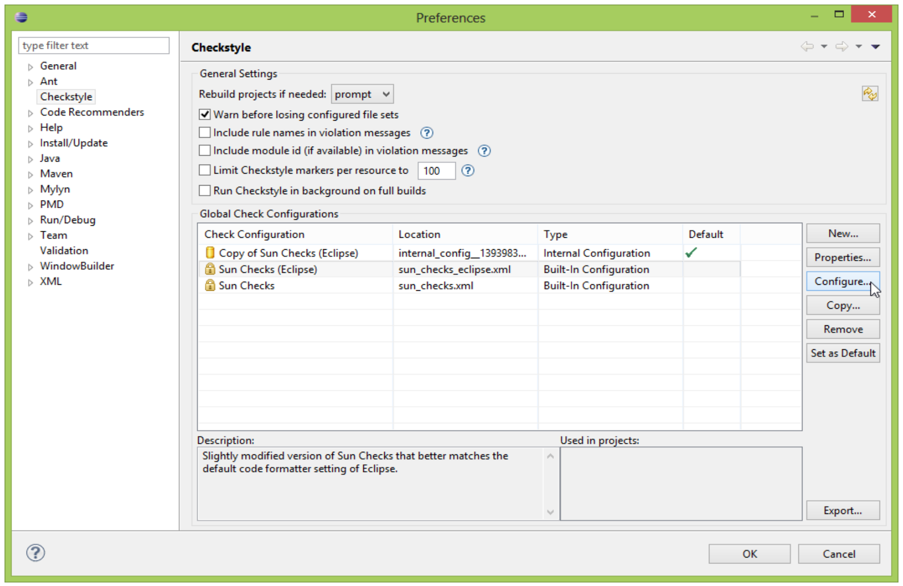

Install
Available Software
Check the items that you wish to install.
Work with: http://eclipse-cs.sourceforge.net/update/
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
Checkstyle
All items are installed
Uncategorized
Select All
Deselect All
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel


Eclipse Marketplace
Eclipse Marketplace
Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.
Search
Recent
Popular
Installed
February 02/20
Find: checkstyle
All Markets
All Categories
Go
Checkstyle Plug-in 5.7.0
The eclipse-cs Checkstyle plug-in integrates the well-known source code analyzer Checkstyle into today's leading IDE - Eclipse. Checkstyle is a development tool... more info
by David Schneider, Lars Koedderitzsch, LGPL
static analysis Favorite validator coding style
233
Installs: 111K (4,315 last month)
Install
cs-contrib.unkrig.de
This project adds custom checks, filters and for CheckStyle and Eclipse-CS. more info
by null, BSD
checkstyle
0
Installs: 91 (7 last month)
Install

Checkstyle (2)

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

General Settings

Rebuild projects if needed: prompt

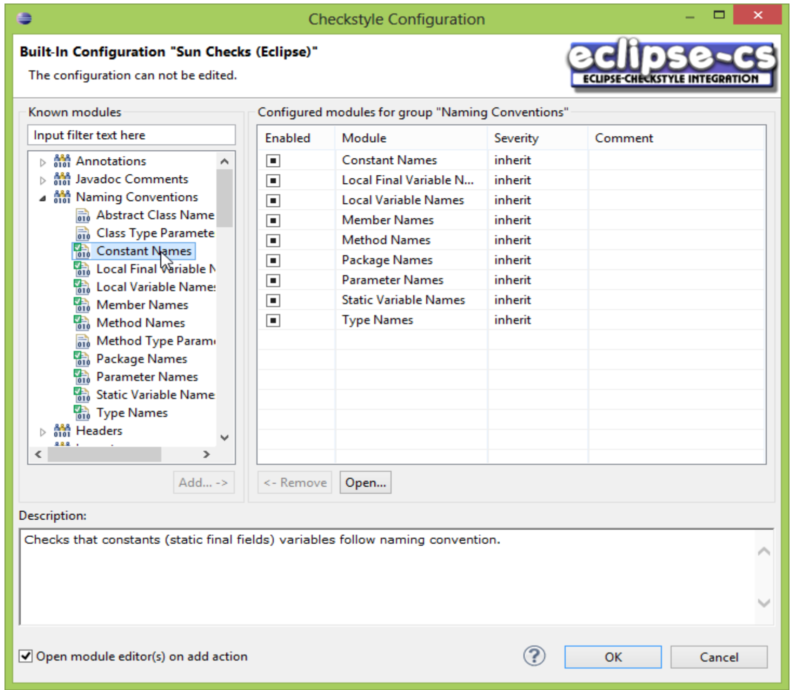
[✓] Warn before losing configured file sets
[ ] Include rule names in violation messages
[ ] Include module id (if available) in violation messages
[ ] Limit Checkstyle markers per resource to 100
[ ] Run Checkstyle in background on full builds

Global Check Configurations

Check Configuration	Location	Type	Default
Copy of Sun Checks (Eclipse)	internal_config__1393983...	Internal Configuration	✓
Sun Checks (Eclipse)	sun_checks_eclipse.xml	Built-In Configuration	
Sun Checks	sun_checks.xml	Built-In Configuration	

Description:
Slightly modified version of Sun Checks that better matches the default code formatter setting of Eclipse.

Used in projects:



Built-In Configuration "Sun Checks (Eclipse)"
The configuration can not be edited.

Configured modules for group "Naming Conventions"

Enabled	Module	Severity	Comment
■	Constant Names	inherit	
■	Local Final Variable N...	inherit	
■	Local Variable Names	inherit	
■	Member Names	inherit	
■	Method Names	inherit	
■	Package Names	inherit	
■	Parameter Names	inherit	
■	Static Variable Names	inherit	
■	Type Names	inherit	

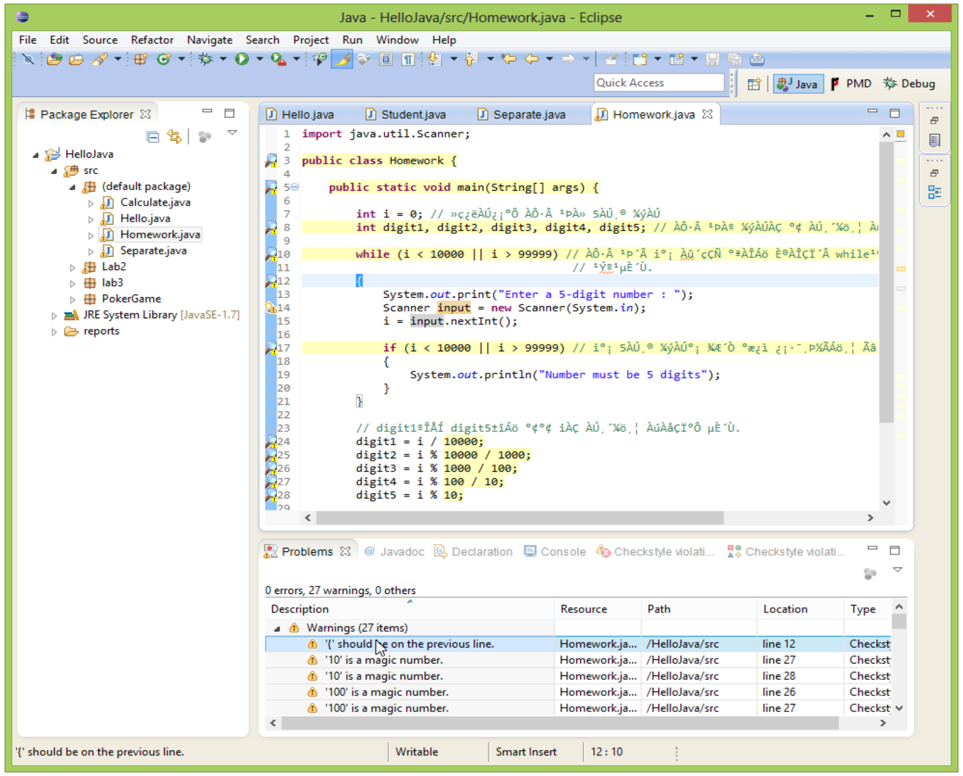
Description:
Checks that constants (static final fields) variables follow naming convention.

[✓] Open module editor(s) on add action

Checkstyle (3)

3. Using Checkstyle in your project

- Right click on project or file -> Checkstyle -> Check code with Checkstyle
- The violation display on [Problems] window



```
 1 import java.util.Scanner;
 2
 3 public class Homework {
 4
 5     public static void main(String[] args) {
 6
 7         int i = 0; // »ç¿ëÀÚ¿¡°Ô ÀÔ·Â ¹ÞÀ» 5ÀÚ¸® ¼ýÀÚ
 8         int digit1, digit2, digit3, digit4, digit5; // ÀÔ·Â ¹ÞÀº ¼ýÀÚÀÇ °¢ ÀÚ¸´¼ö¸¦ À
 9
10         while (i < 10000 || i > 99999) // ÀÔ·Â ¹Þ´Â i°¡ Àû´çÇÑ °ªÀÎÁö È®ÀÎÇÏ´Â while¹
11                                        // ¹Ýº¹µË´Ù.
12         {
13             System.out.print("Enter a 5-digit number : ");
14             Scanner input = new Scanner(System.in);
15             i = input.nextInt();
16
17             if (i < 10000 || i > 99999) // i°¡ 5ÀÚ¸® ¼ýÀÚ°¡ ¾Æ´Ò °æ¿ì ¿¡·¯¸Þ½ÃÁö¸¦ Ãâ
18             {
19                 System.out.println("Number must be 5 digits");
20             }
21         }
22
23         // digit1ºÎÅÍ digit5±îÁö °¢°¢ iÀÇ ÀÚ¸´¼ö¸¦ ÀúÀåÇÏ°Ô µË´Ù.
24         digit1 = i / 10000;
25         digit2 = i % 10000 / 1000;
26         digit3 = i % 1000 / 100;
27         digit4 = i % 100 / 10;
28         digit5 = i % 10;
```

0 errors, 27 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (27 items)				
'{' should be on the previous line.	Homework.ja...	/HelloJava/src	line 12	Checkst
'10' is a magic number.	Homework.ja...	/HelloJava/src	line 27	Checkst
'10' is a magic number.	Homework.ja...	/HelloJava/src	line 28	Checkst
'100' is a magic number.	Homework.ja...	/HelloJava/src	line 26	Checkst
'100' is a magic number.	Homework.ja...	/HelloJava/src	line 27	Checkst

'{' should be on the previous line. | Writable | Smart Insert | 12 : 10

IV. PRACTICE

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

Exercise 1: Extend

- Something you can add more to your program
  - Add new Point class
    - which contain 2 members (int x, int y)
  - Add a members:
    - Point pTopLeft
  - Add new methods:
    - Boolean Contains(Point)
    - Rectangle Intersect(Rectangle)
    - Rectangle Union(Rectangle)