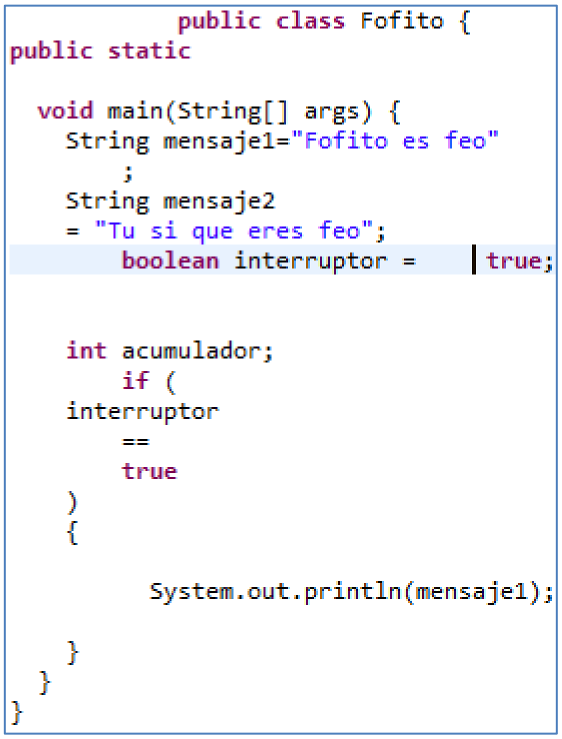
Java coding guidelines

This document is a summary of the guidelines to create elegant Java code.

These guidelines are not part of the Java specification: a given piece of Java code can break all of these guidelines and still be a complete correct program.

However, experience teaches that following these guidelines leads in the long-term to better programs. Code that is correctly formatted severely increases the readability, making much easier to find problems.

For example, these two pieces of code are exactly the same code… which one do you think is easier to read and understand?



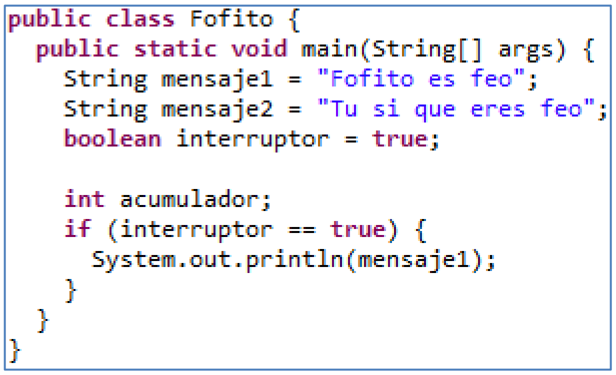
```
      public class Fofito {
public static

 void main(String[] args) {
  String mensaje1="Fofito es feo"
    ;
  String mensaje2
  = "Tu si que eres feo";
    boolean interruptor =   true;


  int acumulador;
    if (
  interruptor
    ==
    true
  )
  {

     System.out.println(mensaje1);

  }
 }
}
```



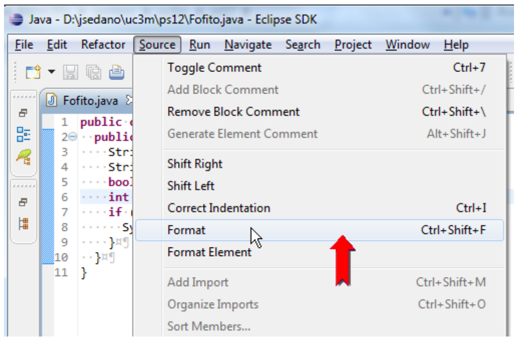
```
public class Fofito {
 public static void main(String[] args) {
  String mensaje1 = "Fofito es feo";
  String mensaje2 = "Tu si que eres feo";
  boolean interruptor = true;

  int acumulador;
  if (interruptor == true) {
   System.out.println(mensaje1);
  }
 }
}
```

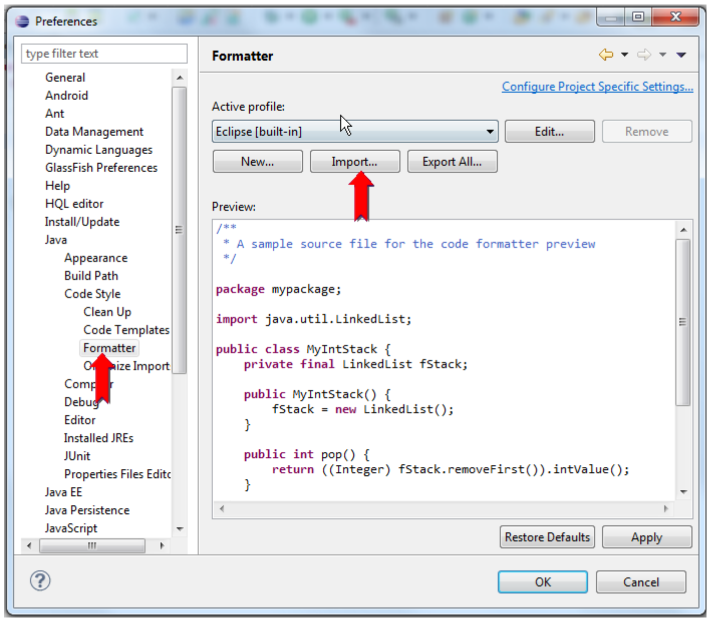
Oracle publishes a guide called *Code Conventions for the Java programming Language* (http://www.oracle.com/technetwork/java/codeconv-138413.html) with several conventions that are widely spread among the Java community of programmers. Such guide will provide advice on how to name the classes, methods and properties, how to indent code, where to add carriage-returns, how to write the braces,…

Do not read on until you have read the guide from Oracle. It is a bit long but it will be worth the time.

Eclipse can help you to correctly format the code. With a .java file open in the editor, use *Source →Format* to format the code with the built-in formatting rules.


Java - D:\jsedano\uc3m\ps12\Fofito.java - Eclipse SDK
File Edit Refactor Source Run Navigate Search Project Window Help
Fofito.java
Toggle Comment Ctrl+7
Add Block Comment Ctrl+Shift+/
Remove Block Comment Ctrl+Shift+\
Generate Element Comment Alt+Shift+J
Shift Right
Shift Left
Correct Indentation Ctrl+I
Format Ctrl+Shift+F
Format Element
Add Import Ctrl+Shift+M
Organize Imports Ctrl+Shift+O
Sort Members...

The default built-in formatting rules mostly comply with the rules defined by Oracle. However, the teachers of the course use a slightly different set of rules (actually, the only difference is that lines are indented using 2 spaces, instead of 1 TAB). If you want to use such slightly different formatting rules, you can download the rules from this link (http://www.it.uc3m.es/java/2012-13/resources/j_java.xml) and import them into Eclipse through its preferences:



```
/**
 * 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();
    }
```

Additionally to those rules, experience teaches that some other rules should be followed:

- Try to avoid characters that are not US-ASCII. US-ASCII allows standard letters (a, b, c,…), numbers (1, 2, 3,…) and standard punctuation marks (?, :, !, $, %,…) but not accentuated letters (á, é, ñ, ï, ô, ù,…) nor “Spanish” punctuation marks (¿ and ¡). Exception to this norm may be the literals to be shown to the user.
- Use meaningful names for classes, methods, properties and variables. You may look cool naming your variables with funny names, or using exercise1() for the methods… but that

will complicate your life later. And also the life of the teacher that will evaluate your code, by the way.

- Use the comments to explain what the code does. Read it again: WHAT the code does, not HOW the code does it. An exception may be specially complicated algorithms. Use meaningful names for classes, methods, proper… hold a second! Didn't we wrote that above? Sure. Many times, if you use meaningful names, you will not need to add a comment. For example, if a method is called setHeight() you know that it obviously sets the height; however, if you call the method setH(), you need to add a comment to say so.
- Try to use short methods (not method names, but short methods themselves). If the method fits your screen, you will be able to see it, read it and understand it at a glance. However, if it does not fit, being continuously scrolling up and down will confuse you.