Styling Guidelines

General indentation & appearance

Every line is indented the same as the line above it, except the cases described explicitly below.

Good	Bad
`System.out.println("Line 1");`	`System.out.println("Line 1");`
`System.out.println("Line 2");`	`        System.out.println("Line 2");`

Block statements:

1. The opening brace comes at the end of the same line of the block statement (separated with a single space).
2. Indent the entire code block one level deeper than the statement line.
3. The closing brace should be on a separate line right after the last non-blank line of the code block,indented the same as the statement line.

Good	Bad
`public static int foo(int i) {`	`Public static int foo(int i) {`
`        System.out.println(“bar”);`	`                System.out.println(“bar”);`
`        while (i > 0) {`	`while (i > 0) {`
`                System.out.print(“and  again”);`	`                System.out.print(“and  again”);`
`                System.out.print(“and  again”);`	`        System.out.print(“and  again”);`
`        }`	`}`
`}`	`}`

Line separation:

1. Do not write multiple statements in the same line.
2. Concatenating statements is allowed.

Good
`String me = “Shimon”;`
`String you = EasyInput.getString();`
`String i_love_u = me + “ love “ + you;`
Bad
`String me = “Shimon”; String i_love_u = me + “ love “ + EasyInput.getString();`

Line wrapping:

1. Keep lines no longer than 80 characters (assuming a tab of 4 spaces long). If a statement is longer than 80 characters, break it according to these rules:
   - Prefer breaking a higher level statement than a lower level.
   - Before an operator.
   - After an opening parenthesis.
   - After a comma.
   - Within a long String and concatenate.
   - Preserve logics; if several breaks needed for a single statement, prefer keeping logical groups in each line.
2. After a line break, indent the second line one level deeper than the statement (not the line) that was broken. If the second line requires more breaking, indent the next lines at the same level as the second line.

Good	Bad

double x = 3;
double y = 10;
double z = (int)(x * x * x)
                * ((x + y) / 2)
                / (x – y) * y;
	double x = 3;
	double y = 10;
	double z = (int)(x * x * x) * ((x + y) / 2) / (x – y)
	* y;

Files:

1. Open a file at indentation level 0.
2. End a file with a closing brace (‘ } ’) at indentation level 0, followed by a single blank line.

Good	Bad

public class JavaWorkshop1 {
        …
}
	public class JavaWorkshop1 {
	        …
	}

Documentation

Any documentation should be at the same level of indentation as the documented code.

Blocks

Each type (class, method, etc.) is preceded by a block comment.

Good	Bad

public class EasyInput {
/*
* This method receives an integer input from
* the user via console.
* The method returns the value the user
* entered or 0 if the input is illegal.
*/
public static int getInt () {
	public class EasyInput {
	    // Get int
	    public static int getInt () {
	        …
	    }
	    …
	}

Code

Precede each logical piece of code with a one line comment. Logical pieces of code are:

1. Sections within a method (local variables, parameter validation and algorithm)
2. A short logical section, usually 2-5 lines of code, rarely a single line.
3. Control structure statements.

Good	Bad

public static void foo (int bar) {
    // validate parameters
    if (bar < 0) {
        return;
    }
 	
    // handle input values
    if (bar == 5) {
	public static void foo (int bar) {
	    if (bar < 0) {
	        return;
	    } else if (bar == 5) {
	        …
	    }
	    …
	}

Some statements may require a one line comment on the same line. This may occur with variable declarations, case statements, else statements and others.

Good	Bad

public static void foo (int bar) {
    if (bar == 0) { // handle this
        doSomething1();
    } else if (bar == 1) { // handle that
        doSomething2();
    } else if (bar == 2) { // handle other
        doSomething3();
    }
}
	public static void foo (int bar) {
	    if (bar == 0) {
	        doSomething1();
	    } else if (bar == 1) {
	        doSomething2();
	    } else if (bar == 2) {
	        doSomething3();
	    }
	}

If several consecutive lines require an one line comments, prefer indenting them the same. Otherwise separate the comments from the statement with a single space or tab.

Good	Bad

public static void foo (int bar) {
  int var1; // temporary variable
  double var2; // algorithm result
  String c; // error message
	public static void foo (int bar) {
	  int var1; // temporary variable
	  // algorithm result
	  double var2;
	  String c; // error message

Naming

Choose a name that describes the role of the subject, and not the value it holds.

Good	Bad
public static void foo (int bar) { double average = 0;	public static void foo (int bar) { double zero = 0;

Names should not be too short and not too long. Avoid abbreviations, prefer initials.

Good	Bad
int sum;	int t;
double average;	double resultToBeReturnedAtTheEnd;
int numberOfStudents ;	int noStdnts;
String pageUrl;	String pageUniformResourceLocator;

Expressions

Operators

1. Separate binary and ternary operators from operands with a single space on both sides.
2. Avoid using == true and == false in boolean expressions.

Good	Bad
x + y / z > 0	x+y/z>0
int x = 7;	int x=7;
if (x > 0) {	if (x > 0 == true) {

Parentheses

1. Separate parentheses from the outside with a single space.
2. Do not separate parentheses from the inside.
3. If a parenthesized expression begins / ends with parenthesis, attach them (no space).
4. Use parentheses only if it's required by a statement or it makes the code clearer.

Good	Bad
((x + y) / z) > 0	((x + y) / z) > 0
if ((x + 3 > 0) || (y < (x – 4) / 2) && !foo()) {	if ((x + 3) > 0 || y < x – 4 / 2 && (!foo()) {