

Coding Standard

Why?!

1. A standard makes it easier for the instructor and TAs as well as classmates to read your code.
2. A class standard makes it easier for a grader to recognize when a program does not use a consistent standard. Often when each student is allowed to define his or her own standard, students switch standards multiple times in a single project. It is tedious for a grader to deduce each person's standard and then check for self-consistency.
3. Learning to adhere consistently to a coding standard is a good practice.
4. This standard does not necessarily represent the best nor the only good way to write Java code.
5. Consistency trumps all rules.

General Practices

- All variable names (fields) shall begin with a lower case letter.
- All variables that do not ever change value shall be all uppercase.
- All class variables (non-local variables) will be given descriptive names.
- All methods will be given descriptive names.
- All class names shall begin with an uppercase letter

Indenting

- Code blocks will be indented to show the block structure with exactly two spaces per level.
- Tab characters shall not be used for indenting (Processing turns every tab-keypress into two spaces).
- All statements within a block must be indented to the same level.

Brackets

Open brackets must be the **LAST** non-space character on a line.

```
public class Hello
{
    public static void main(String[] a)
    {
        System.out.println("Hello World");
    }
}
```

Good!

```
public class Hello {
    public static void main(String[] a) {
        System.out.println("Hello World");
    }
}
```

Good!

```
public class Hello
{ public static void main(String[] a)
  { System.out.println("Hello World");
  }
}
```

Bad!

Closing brackets will be indented on a line with no other commands. The only exception being comments placed on the line with a closing bracket.

```
public class Hello {
    public static void main(String[] a) {
        System.out.println("Hello World");
    }
} // end class Hello
```

Good!

```
public class Hello
{
    public static void main(String[] a)
    {
        System.out.println("Hello World");
    } System.out.println("Bye");
}
```

Bad!

Whenever a structure (for, while, if, etc.) spans more than one line, brackets must be used. For example:

Good! `if (x == 5) y=y+1;`

Good! `if (x == 5)
{
 y=y+1;
}`

Bad! `if (x == 5)
 y=y+1;`

Class Comments

At the top of every class file, there must be a comment block with the following information. Format the information as you think best.

```
//*****  
// Your first and last name  
//  
// Description of what the class  
// is used for and how to use it.  
//*****
```

Method Comments

At the top of **every method**, there must be a comment block with the following information. Format the information as you think best.

```
//*****  
// Each parameter's (arguments) type and name:  
//   input and/or output,  
//   its meaning,  
//   its range of values.  
// Method's return value.  
// Description of what the method does.  
// Method's Algorithm  
//*****
```

Class Setup

```
public class MyClass {  
  
    Class Variable(s)    int myClassIntVariable1;  
                        int myClassIntVariable2;  
  
                        public MyClass()  
                        {  
    Constructor(s)    // Executable Code  
                        }  
  
                        public int myMethod1(int x, int y)  
                        {  
    Method(s)        // Executable Code  
                        }  
  
                        public void main(String[] args)  
                        {  
                        // Executable Code  
                        }  
  
                        }  
}
```