

Java Coding Guidelines

Style

- ✓ Comment member variables
- ✓ Obvious/obfuscated comments are useless)

General Coding

- ✓ Eliminate code replication
- ✓ Do not ignore compiler warnings. Do not suppress compiler warnings unless **well** justified.
- ✓ No spurious object creation.

```
String firstName = "";
String lastName = new String("");
// Assignments above wasted assignment/allocation since just replacing values
firstName = in.nextLine();
lastName = in.nextLine();
```

- ✓ Import only necessary classes. Do not use wildcard imports (e.g., java.util.*) unless there are 4 or more classes from that package.
- ✓ For simple boolean methods, return directly from expression instead of using if.

```
boolean empty() { // Yuck
    if (length == 0) {
        return true;
    }else {
        return false;
    }
}
```

```
boolean empty() { // Yep
    return (length == 0);
}
```

This avoids potential errors such as getting true/false returns backwards.

- ✓ Do not use C-style array declarations.

```
int x[]; // No!!!
int[] x; // Yep
```

- ✓ Do not call toString() if it is implicitly called.

```
System.out.println(blah.toString()); // NO!!!!
System.out.println(blah); // Yep
```

- ✓ Do not use deprecated methods.
- ✓ Use "Mom".equals(s) instead of s.equals("Mom") to handle the case of s == null.
- ✓ Always specify access (or comment why package is appropriate). Use correct access.

- ✓ No junk member variables. Member variables are for state related to object, not for variables used by several methods.
- ✓ Don't write useless code
 - a. Empty/autogenerated constructors - Why maintain the code?
 - b. `String blah = thing;`
`return blah;`
 - c. Unnecessarily calling `toString()`.
 - d. `if (done == true)`

Collections

- ✓ Use collection interface references instead of concrete type references

```
ArrayList l = new ArrayList(); // NO!  
List l = new ArrayList(); // Yep!
```

- ✓ Don't use the older collection classes such as `Vector` and `Hashtable`. Instead use `ArrayList` and `HashMap`. The main difference is that the new collection classes are not synchronized so their performance should be better. `Vector` and `ArrayList` differ slightly in their expansion algorithms. Unlike `Hashtable`, `HashMap` permits null values and a null key. If you need synchronization, use the `Collections` class synchronization wrapper.

```
List<String> l = Collections.synchronizedList<String>(new ArrayList<String>());
```

Note that Java concurrent package contains `ConcurrentHashMap` and `CopyOnWriteArrayList` for better performing synchronized maps and lists.

- ✓ Catch the most specific exception type.

JUnit

- ✓ Use specific JUnit asserts.

```
assertTrue(x.equals(y)); // No!  
assertEquals(x, y); // Yep!
```