

# COMP 3350 Group 3 Programming Standards

## 1 All files shall begin with a explanatory header

```
/**  
 * File name  
 *  
 * COMP3350 SECTION A02  
 *  
 * @author  
 * @date  
 *  
 * PURPOSE:  
 *  
 **/
```

## 2 If their purpose is not obvious, methods will have be preceded by an explanatory header

```
/**  
 * Function name  
 *  
 * brief description of the function  
 * @param {datatype} parameterName1 - parameter description  
 * @param {datatype} parameterName2 - parameter description  
 * @returns {datatype} Description of the returns value  
 */
```

## 3 Non-obvious code chunks will be commentated

#### **4 Contents of classes will be grouped consistently**

The following order is recommended:

```
class Something {  
  // Instance variables  
  // Static variables  
  // Class-level constants  
  // Constructors  
  // Other instance methods  
  // Static methods: main comes first, if it exists  
}
```

#### **5 Classes will be in the package scheme that most corresponds to their purpose**

**6 All handlers and objects in logic / database layer will have interfaces that have names that begin with “I”**

#### **7 Consistent indentation to clarify control structures**

#### **8 No magic numbers, all constants will be assigned a variable**

#### **9 Variables have as small a scope as possible**

#### **10 float and strings not used with == or !=**

#### **11 for loop values not changed inside the loop**

**12 Best possible loop construct used**

**13 Avoid unnecessary duplication of code**

**14 Classes named in the form “ClassName”, constants are “ALL\_UPPER\_CASE” and all other identifiers are camelcase “otherNames”**

**15 meaningful variable names**

**16 Proper separation of responsibilities between layers**

**17 Access specifies as specific as possible**

## Punishment

Any group member caught violating these standards will have their name and code in question added to the wall of shame

### Wall of Shame

**Name:** Jillian

**Violation:** Getting called out by Rob for having ugly declarations in the UI (if you're gonna have ugly code at least hide it better smh).

**Code:**

```
comp3350-winter2024 / ThreeQuartersCS-A02-3 / Repository

18 import java.util.ArrayList;
19 import java.util.List;
20
21 public interface ITransactionHandler {
22     boolean addTransaction(Transaction t);
23
24     boolean modifyTransaction(Transaction t);
25
26     boolean deleteTransaction(Transaction t);
27
28     Transaction getTransactionByID(int id);
29
30     List<Transaction> getAllTransactions();
31
32     ArrayList<Transaction> getAllByNewestFirst();
33
34     ArrayList<Transaction> getAllByOldestFirst();
35
36     ArrayList<Transaction> getTransactionByCategoryID(int categoryID);
37
38     ArrayList<Transaction> getTransactionByName(String name);
39
40     ArrayList<Transaction> getTransactionByPlace(String place);
41
42     /*
43
44     getTransactionByAmount
45
46     Returns all transactions with amount values equal to the specified amount
47
48     */
49     ArrayList<Transaction> getTransactionByAmount(double amount);
50
51     ArrayList<Transaction> getTransactionByAmountBetween(double minAmount, double maxAmount);
}
```