CPSC 501

Assignment 2

Name: Zicheng Huang

UCID: 30009205

Refactoring:

1. Extract method (Make the code and structure clearer for the user to read)

   Before:

```
out.println("================ Basic ClassInfo ================");

out.println("Class: " + cls.getName());
out.println("Immediate superclass: " + cls.getSuperclass());
out.println("Is Interface: " + cls.isInterface());
out.print("Interfaces: ");
Class[] interArray = cls.getInterfaces();
arrayInfo(interArray);
out.println("============== Basic ClassInfo End ================");
out.println();
out.println("================ Method Info ====================");

Method[] mArray = cls.getDeclaredMethods();

for(Method m : mArray) {
    out.println("----Method name: " + m.getName());

    out.print("----Exceptions throw: ");
    Class[] excArray = m.getExceptionTypes();
    arrayInfo(excArray);

    out.print("----Parameter types: ");
    Class[] parArray = m.getParameterTypes();
    arrayInfo(parArray);

    out.println("----Return type: " + m.getReturnType());
    out.println("----Modifier: " + Modifier.toString(m.getModifiers()));
    out.println();
}
out.println("============== Method Info End ==================");
out.println();
out.println("================ Constructor Info ================");
Constructor[] conArray = cls.getConstructors();
for(Constructor c : conArray) {
    out.println("----Parameter types: ");
    Class[] parArray = c.getParameterTypes();
    arrayInfo(parArray);

    out.println("----Modifier: " + Modifier.toString(c.getModifiers()));
}
out.println("============== Constructor Info End ==============");
```

   After:

```
printClassInfo(ObjClass);
printMethodInfo(ObjClass);
printConstructorInfo(ObjClass);
```

```
// Class Information
public void printClassInfo(Class cls) {
    out.println("================ Basic ClassInfo ================");

    out.println("Class: " + cls.getName());
    out.println("Immediate superclass: " + cls.getSuperclass());
    out.println("Is Interface: " + cls.isInterface());
    out.print("Interfaces: ");
    Class[] interArray = cls.getInterfaces();
    arrayInfo(interArray);
    out.println("============== Basic ClassInfo End ================");
}

// Method Information
public void printMethodInfo(Class cls) {
    out.println("================ Method Info ====================");

    Method[] mArray = cls.getDeclaredMethods();

    for(Method m : mArray) {
        out.println("----Method name: " + m.getName());

        out.print("----Exceptions throw: ");
        Class[] excArray = m.getExceptionTypes();
        arrayInfo(excArray);

        out.print("----Parameter types: ");
        Class[] parArray = m.getParameterTypes();
        arrayInfo(parArray);

        out.println("----Return type: " + m.getReturnType());
        out.println("----Modifier: " + Modifier.toString(m.getModifiers()));
        out.println();
    }
    out.println("============== Method Info End ==================");
}
```

```
// Constructor Information
public void printConstructorInfo(Class cls) {
    out.println("================ Constructor Info ================");
    Constructor[] conArray = cls.getConstructors();
    for(Constructor c : conArray) {
        out.println("----Parameter types: ");
        Class[] parArray = c.getParameterTypes();
        arrayInfo(parArray);

        out.println("----Modifier: " + Modifier.toString(c.getModifiers()));
    }
    out.println("============== Constructor Info End ==============");
}
```

2. Rename function (rename the method to make it sense and it is easy to be understood by both users and programmers)

Before:

```
// Array recursive helper function
public void recursiveMethod(Object obj) {
    out.println("----Array name: " + obj.getClass().getName());
    out.println("----Array Component type: " + obj.getClass().getComponentType());
    out.println("----Array length: " + Array.getLength(obj));
    if(obj.getClass().getComponentType().isArray()) {
        for(int i = 0; i < Array.getLength(obj); i++) {
            out.println("No." + i + "= ");
            recursiveMethod(Array.get(obj, i));
            out.println();
        }
    }else {
        for(int i = 0; i < Array.getLength(obj); i++) {
            out.println("No." + i + "= " + Array.get(obj, i));
        }
    }
}
```

After:

```
// Array recursive helper function
public void arrayRecursiveMethod(Object obj) {
    out.println("----Array name: " + obj.getClass().getName());
    out.println("----Array Component type: " + obj.getClass().getComponentType());
    out.println("----Array length: " + Array.getLength(obj));
    if(obj.getClass().getComponentType().isArray()) {
        for(int i = 0; i < Array.getLength(obj); i++) {
            out.println("No." + i + "= ");
            arrayRecursiveMethod(Array.get(obj, i));
            out.println();
        }
    }else {
        for(int i = 0; i < Array.getLength(obj); i++) {
            out.println("No." + i + "= " + Array.get(obj, i));
        }
    }
}
```