**Code Review on**

***Kids Learning Software***

a desktop-based application for

effortless learning by children

|  |  |
| --- | --- |
| Submitted to:  Dr. Amit Kumar Mondal  Associate Professor  Computer Science  and Engineering Discipline  Khulna University  Khulna | Submitted by:  M.M. Emon Hossain  Student ID: 210234  &  Sandesh Sapkota  Student ID: 210242  3rd Year, 1st Term  Computer Science  and Engineering Discipline  Khulna University  Khulna |

**Course No:** CSE 3106

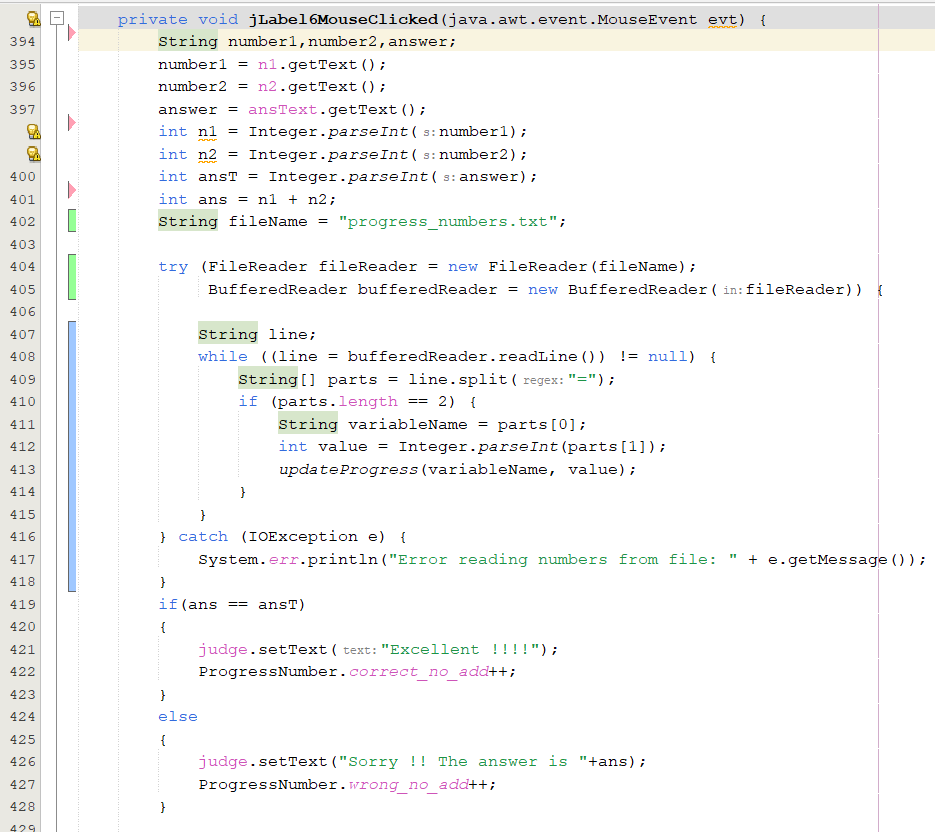
**Course Title:** Software Development Project



**Date of Submission:** March 18, 2024

**Code smell**

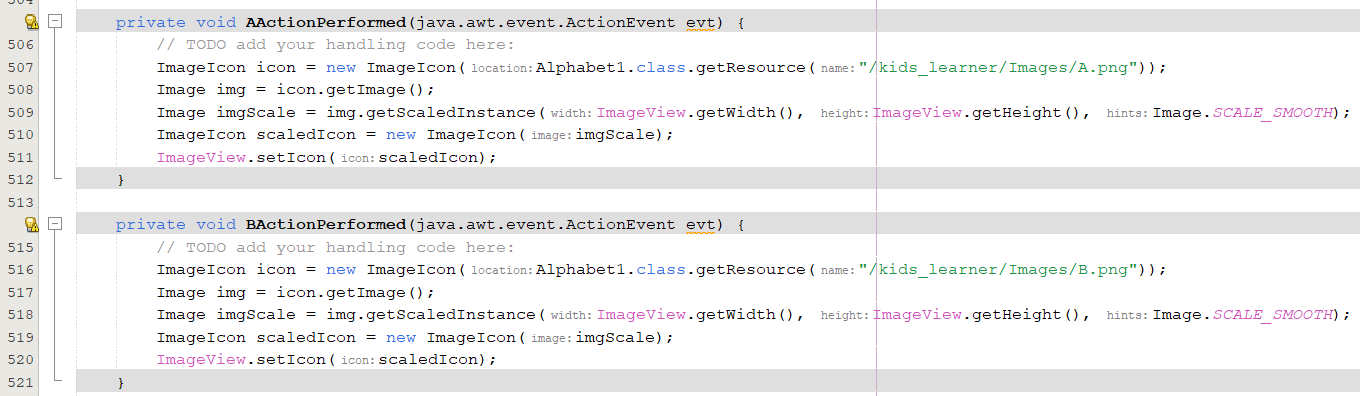
1. **Large and complex method**



It is a example of P\_addition.java. This method that is too long or complex can be difficult to understand and maintain. It can also make the code more error-prone and difficult to test. Breaking the method into smaller like fileReader method can make it easier to understand and maintain.

**2. Duplicate code:**

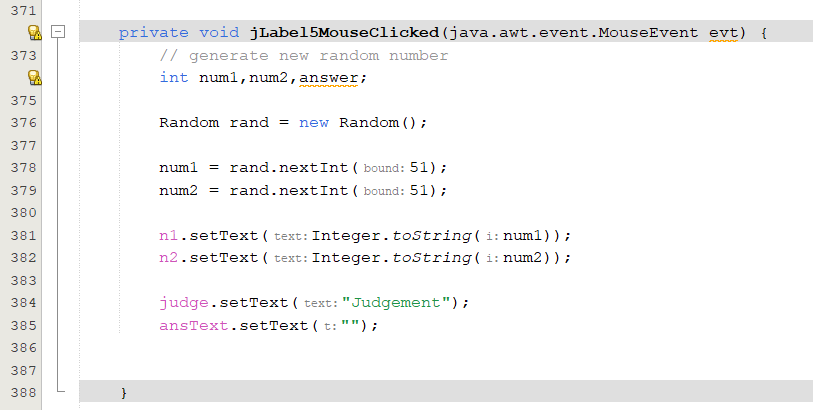
Here a example of Alphabet.java code,



There is a problem that there are same code in this two method. An image resize method can replace this code.

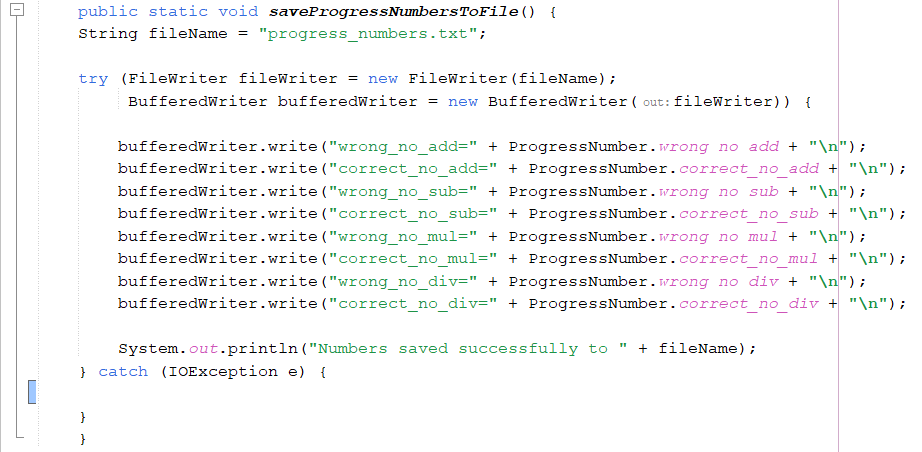
3. **Inconsistent naming conventions:**

Here a example of P\_addition.java code,



In this code we cannot understand for which section the code working. Here jLabel5MouseClicked() method name can be replaced with a suitable name.

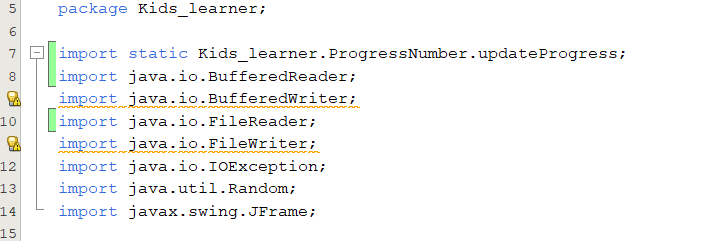
**4. Incomplete error handeling:**

****

The problem with this code is that it does not handle the error properly. If there is an error while making the API call, the function simply logs the error to the console and returns nothing.

**5. Unnecessary dependencies:**

There are various unnecessary external libraries and frameworks imported in various modules in this project which are not being used. For example, In P\_addition.java

****

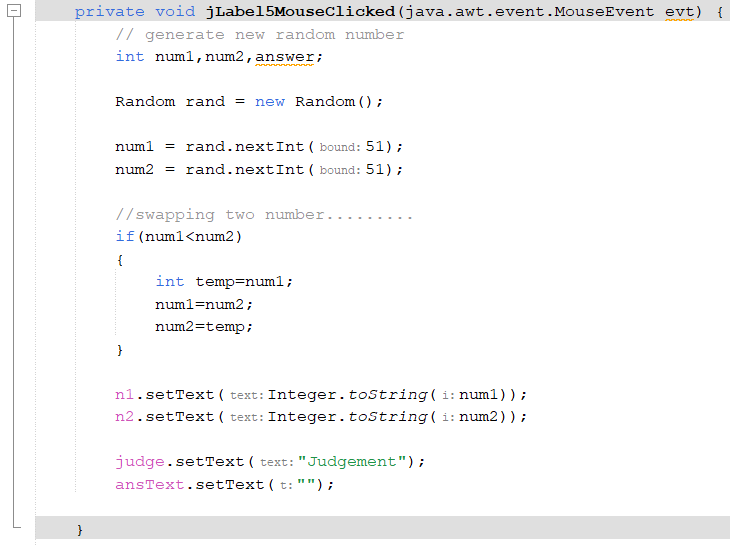
This some package are not in use but still they are decleared.

**6. Too many if/else statements:**

In the Practice segment, there are some excessive use of if/else statements. But in other modules, the usage of if/else statements are moderated.

**7. Magic numbers or hard-coded values**

A example of P\_addition.java



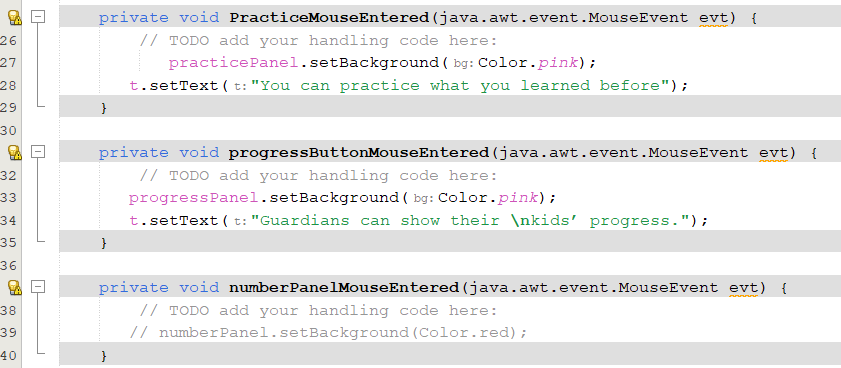
Here they directly use 51 in num1= read.nextInt(); it is a magic number. They also apply this magic number in several code. To improve code readability and maintainability, it's recommended to replace these magic numbers values with meaningful constants or variables with descriptive name

**8. Poor use of inheritance:**

There is no poor use of inheritance in this project which is particularly causing any problem.

**9. Excessive comments:**

In some modules there are excessive comments and some doesnot have any comment at all. In the Home.java there are excessive comments and Progress\_Number.java module there is no comment at all. So, there are inconsistencies in using comments. Here a example of home.java of excessive comments.



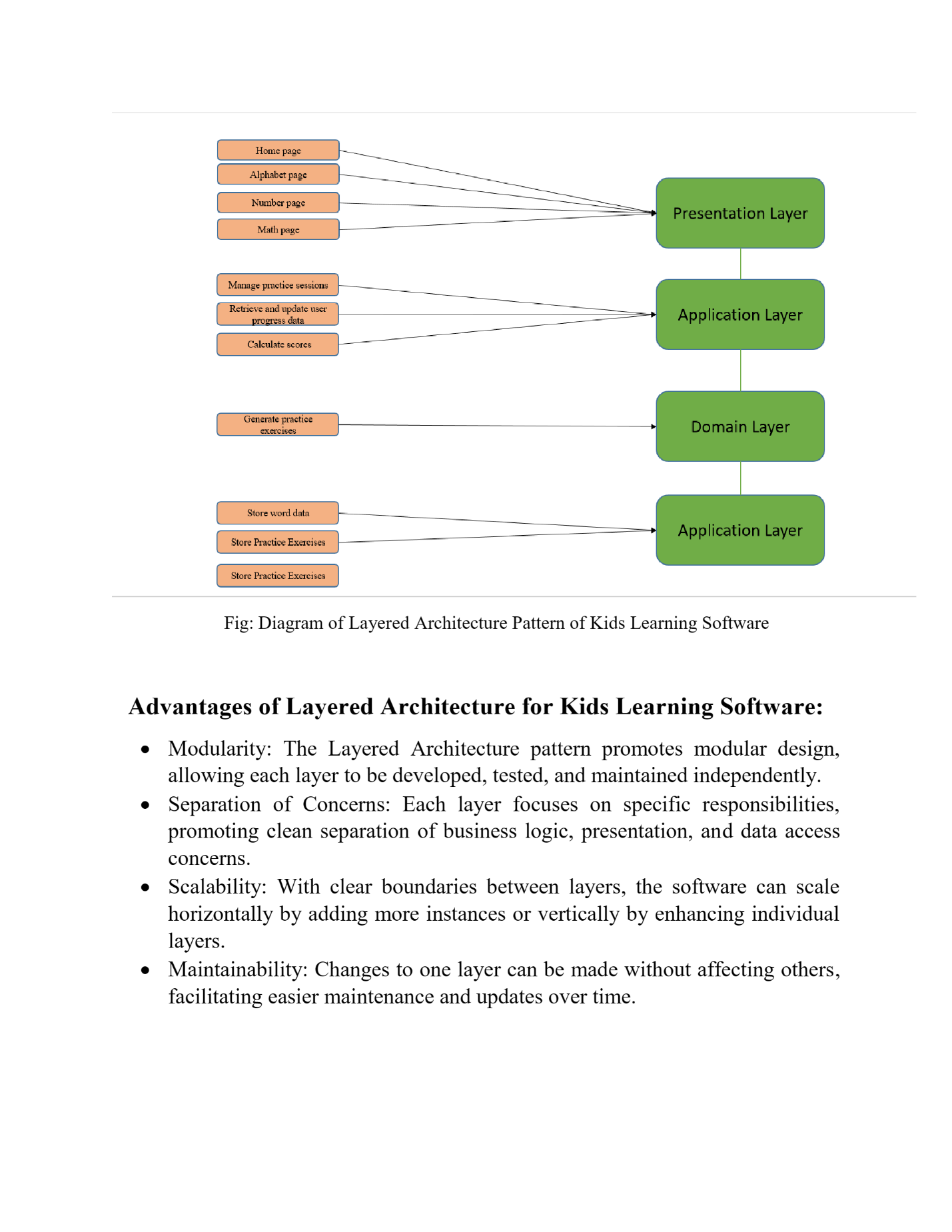
**10. Long parameter lists:**

There is no method with long parameter lists in the project.

**Proposed Architecture Evaluation:**

The proposed architecture of the project is **“Layered Architecture”**. The envisioned project architecture follows the elegant design of "Layered Architecture," manifesting its principles in the project's very essence.

**Layers Representation of “Kids Learning Software”:**



1. **Presentation Layer:**

This layer contains Home page, Alphabet page, Number page and Math page. All these pages are responsible for handling user interfaces and interactions.

1. **Application Layer:**

It interacts with Generate practice sessions that is in the Domain Layer to fetch or modify data in the existing modules of this layer which are Manage practice sessions, Retrieve and update user progress data and calculate scores.

1. **Domain Layer:**

This layer can be referred to as the heart of the layer encapsulating various entities, behaviors and their relationships. It has generate practice session module that interacts with application and data access layers for data persistence.

1. **Data Access Layer:**

Two modules (Store word data and Store practice exercises) in this layer handles data storage and retrieval. It interacts with file system and shields the rest of the application from data storage details.

So, by analyzing the architecture, it can be stated that the project is not consistent with the domain layer that is proposed in the architecture pattern. The domain layer cannot be distinguished from the other layers. Therefore, the proposed architecture pattern is not completely followed throughout the project.

**Modularity Check**

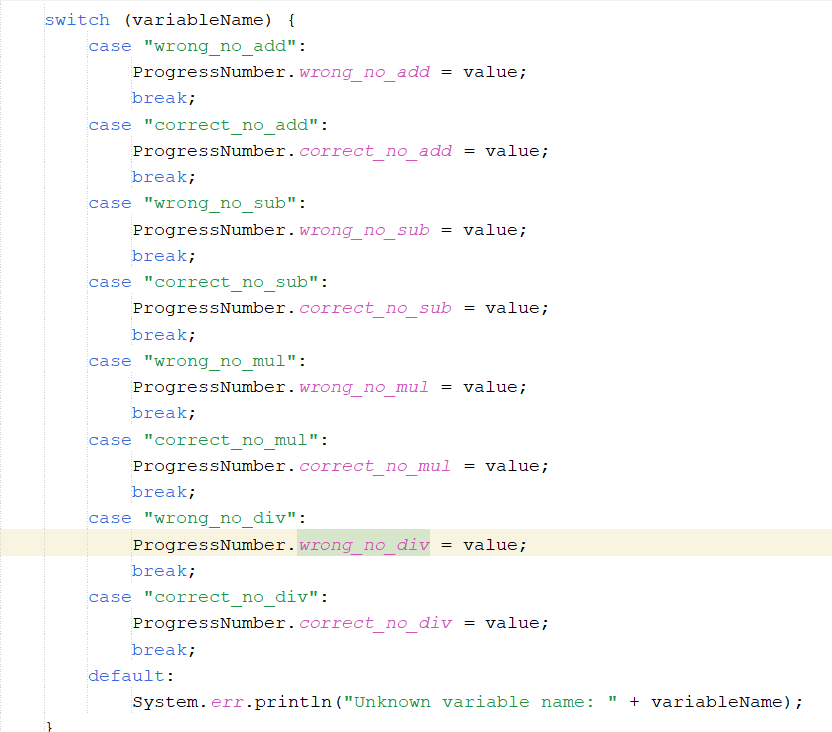
The project is divided into modules which are:

* **Home.java:** responsible for displaying all kinds system components that users can interact with
* **Alphabet.java:** contains features for kids to learn alphabets
* **Math.java:** user interface displaying all kinds of numeric operations that children can practice.
* **Number.java:** contains features to learn about numbers.
* **(Addition, Subtraction, Multiplication, Division).java:** these modules contains all kinds of necessary features for numeric arithmetic operations
* **Practice.java:** Children can practice different kinds of exercises in this module
* **Progress.java:** responsible for generating progress on each practice exercises
* **(p\_addition, p\_subtraction, p\_multiplication, p\_division).java:** these modules contains practice exercises for each of the arithmetic operations.
* **ProgressNumber.java:** contains progress from number exercises.
* **Screen.java:** welcoming interface module.

All these modules specify that the functionalities of the project are divided into suitable modules that perform

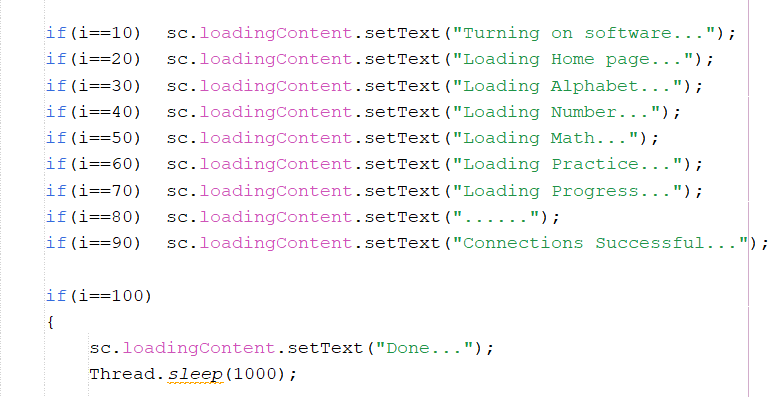
**If/else Conditions to Switch Statement**

There is one switch case statements implemented in the project.

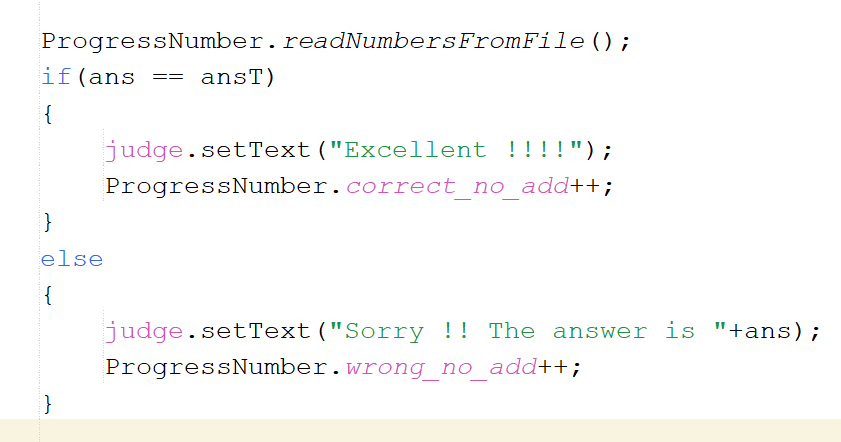


And below are the two if else statements:

1.



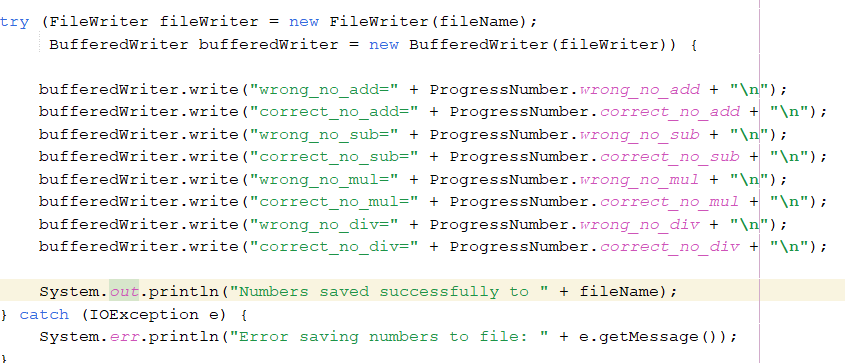
2.



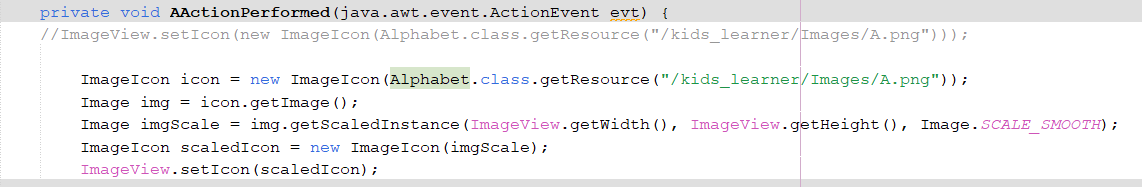
The first figure doesn’t contain any else statements indicating that this part could have been implemented using switch statements. And the second figure contains only one if-else block which have been implement across p\_addition, p\_subtraction, p\_multiplication, p\_division modules. This is a simple if-else block and is not required to be converted to switch statement.

**Divergent Change**

1.



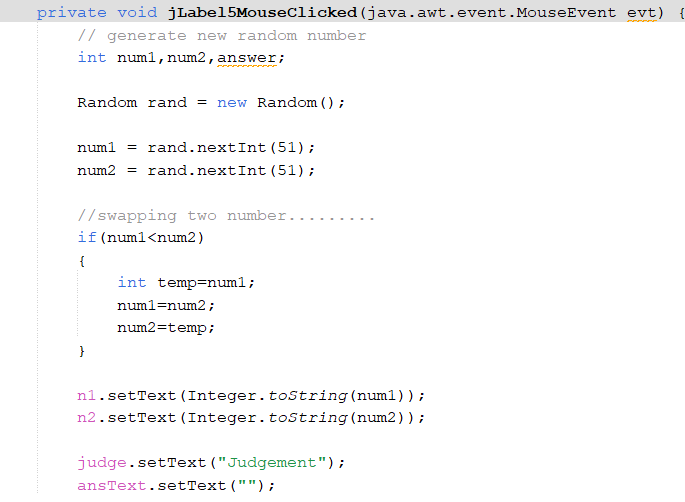
This part of the code in all kinds of progress modules mentioned in Modularity Check section needs to be changed every time a new data storage system is required to be implemented.

2. 

There are total of 26 methods like this in the alphabet.java module. Whenever a new alphabet system is introduced, all the 26 methods will be required to be modified which is a really extensive process.

In this case, it’s likely to have a situation in which two objects are better than one.

**Shotgun Surgery**



There are similar methods for practice exercise mentioned in the Modularity Check section. Here random numbers are being generated for practicing different operations. If users wishes to input values according to on which numbers they wish to evaluate then developers would need to change this method in the all related modules indicating that changes are scattered all over the place.