Java OOP: Children Arithmetic Teaching

# Introduction

I am Mac Hai Long ID: 11060, a student of VGU, currently applying Computer Science. In this project, I’m a Leader/Developer of a Children Arithmetic Teaching program.

# Program overview

This Children Arithmetic Teaching is an application to teach children to perform Arithmetic calculation.

The program contains following variable:

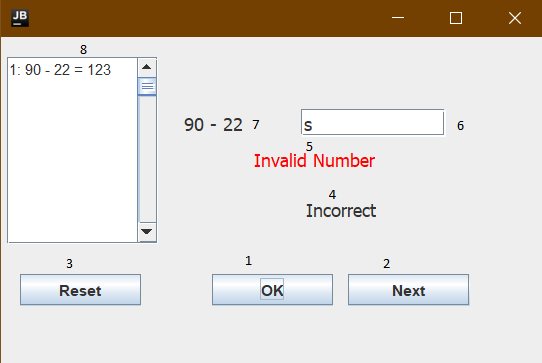
### firstNumber: This is the first number to be randomize

### secondNumber: This is the second number to be randomize

### result: this is the result (true/false) of the Arithmetic calculation that will be calculate and compare to user answer.

### questionToDisplay: this take the randomized firstNumber and secoundNumber and add it to a string to be display to user.

This application has following functionalities and can be accessed via the main UI:



### 1. “OK” button: When user click this button, it will trigger the program to check for user input in (6)

2. “Next” button: When user click this button, it will show user next question and clear the previous answer from user.

3. “Reset” button: When user click this button, it will reset user answer history and reset the user tries counter.

4. This will show if user answer is correct or not after user click OK (1) or press enter key in the input field (6).

5. User Error Field: This notify the user when user has input a wrong value type (not Integer) or has not enter any value. This trigger after user click Ok (1) or press Enter in input field (6).

6. This is user input field. User will enter their answer of the question currently displayed at question field (7)

7. This is a question field. This will show user the question and will refresh after use click “Next” (2). This question remains still if user click “Reset” (3) and “OK” (1).

8. This is a place where the program will display the user’s previous attempt to answer the question and their answer is wrong.

# Analyze & Design

## Program tasks

The programing task is assigned to me only. As well as the testing and design phase.

The programing task is divided in to 2 main phase backend and frontend:

Backend:

1.Number generator

2. Question to display

3. Verify the result

4. Display the result

5. Store the data of user previous attempt.

Frontend:

1. Create the UI

2. Mapping the function of backend to frontend, also add logic to each field:

1. The next button. This will generate new question and delete everything (user input, question, user attempts)

2. Ok button: This button will check if user input is valid, then will compare the user answer with calculated answer.

3. Reset button: this will reset user previous incorrect attempts.

4. Question label: this will take the question from backend and display it to frontend.

5. User input check: this will check for the use input, which in this case is int only.

6. User answer check: This will check for user answer and compare it with calculated answer in the backend.

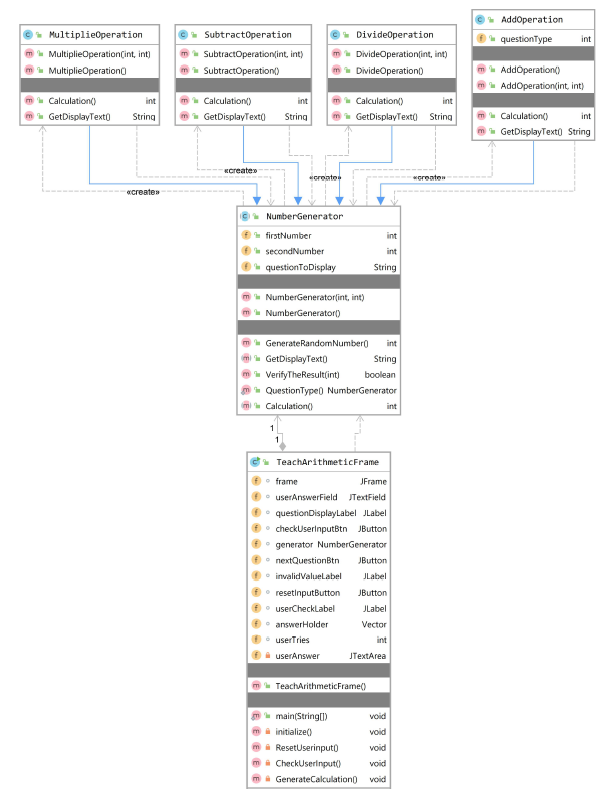
7. User answer field: This text field will take user input and pass it to be process.

8. User attempts field: this will display the user incorrect attempts.

## Design

I want to create a Hieratical system of class representing the mathematic operation. The design will consist of a NumberGenerator class. This class is the abstract class of the program. Then other child class represent the mathematical operation will be created as a child of this class for example AddOperation() class, … The base Number Generator class will handle most of the heavy work like generating random number and verifying user result to the expected result. Then all of this will be pass to frontend. In the frontend, we will call for those function and process logic display

## Program diagram



## Class descriptions

TeachArithmeticFrame(): This class is the main execute class, which hold logic of the User Interface elements.

NumberGenerator(): This is the abstract class of the Arithmetic calculation. It’s consisted of some important method which will be explain below:

1. Constructor: will generate 2 Random numbers using GenerateRandomNumber function.

2. Method:

a. GenerateRandomNumber() : This will generate a random number and return it as int.

b. GetDisplayText(): This will check for the type of the operation and return a string to be use on UI.

c. VerifyTheResult(): This take user input and compare with program calculated one. This return a Boolean.

d. GenerateQuestion(): A static method will randomly pick the type of operator class.

Operation class: There are 4 Operation classes in the program (AddOperation, DivideOperation, MultipleOperation, SubtractOperation). They are children on NumberGenerator. Their methods and properties are like their parent class. Except for DivideOperation() which has a condition for divider.

## Development steps

First, we will create the generator function, this function will create an Operation object, it will automatically create the in the constructor of the base class (NumberGenerator class). The Verify Function is created to verify user result with the input is int. GetDisplayText() and Calculation() is defined here. It will be later override in the Operator class.

GenerateQuestion() is create, it will use to random the question type and switch the type of Arithmetic question base on that.

After that we define Calculation() method to calculate the result of the operation. Finally, we define GetDisplayTest() Function to return a String to be use on front end.

Move to frontend, Swing is used in this program. We first implement the frame, all the button (OK, Next, reset) as well as all the text field and label.

Then we add logic in the UI. Firstly, is to create an object for the UI and the question label. After that is the user answer box.

The answer box also gets the action listener to check for the Enter key to be press. Next is the Next button. It will have to clear all the error messages, the user input and reset the user tries.

Then come the OK button. This is the most function rich in the UI, since it checks the answer for correctness and the input value. When this button is click, it will trigger CheckUserInput() function. This handle most the logic of the program.

The CheckUserInput() will first get the user answer and convert it to int then pass the value to the Verify result function. This function will return the Boolean if user answer is correct or not. If it’s not correct, it will save user attempts into the Vector.

All the label about miss type data or answer check display is being implement in the OK button.

Next, we implement the reset button and user answer history Text field. User answer Data is being store in a vector, as well as user attempts. The vector store data will be clear wen user hit the reset button. All the remaining data should not be affected.

# Error handling

## User Answer validation:

Answer must be an Integer. Otherwise to program will throw an error back to user in the User Error Field (5).

# Possible improvement (or alternatives)

1. Operator limiter (allow user to select which kind of operator to be generate)

2. Real number calculation.

3. Advance operation (Sqrt, …)

4. Point System.

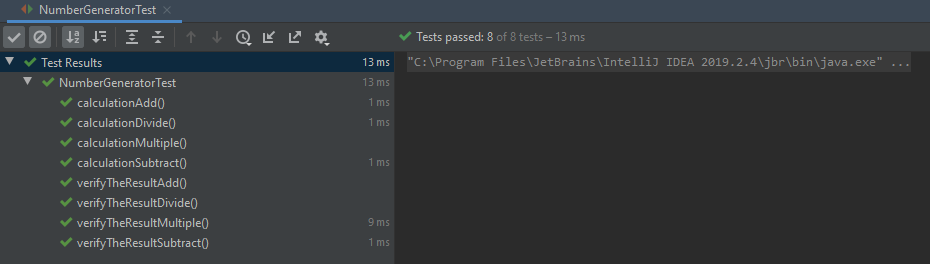
5. Intro screen for how to use.

# Test plan used

We use Junit unit Test. All the input and output will be used here, to verify the functionality of the calculation.

package Test;  
  
import Calculator.\*;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class NumberGeneratorTest {  
 NumberGenerator generator;  
  
  
 @org.junit.jupiter.api.Test  
 void verifyTheResultAdd() {  
 generator = new AddOperation(3,4);  
 *assertTrue*(generator.VerifyTheResult(7));  
 }  
 @org.junit.jupiter.api.Test  
 void verifyTheResultSubtract() {  
 generator = new SubtractOperation(8,4);  
 *assertTrue*(generator.VerifyTheResult(4));  
 }  
 @org.junit.jupiter.api.Test  
 void verifyTheResultMultiple() {  
 generator = new MultipleOperation(8,4);  
 *assertTrue*(generator.VerifyTheResult(32));  
 }  
 @org.junit.jupiter.api.Test  
 void verifyTheResultDivide() {  
 generator = new DivideOperation(8,4);  
 *assertTrue*(generator.VerifyTheResult(2));  
 }  
  
 @org.junit.jupiter.api.Test  
 void calculationAdd() {  
 generator = new AddOperation(2,3);  
 *assertEquals*(5, generator.Calculation());  
 }  
 @org.junit.jupiter.api.Test  
 void calculationSubtract() {  
 generator = new SubtractOperation(2,3);  
 *assertEquals*(-1, generator.Calculation());  
 }  
 @org.junit.jupiter.api.Test  
 void calculationDivide() {  
 generator = new DivideOperation(9,3);  
 *assertEquals*(3, generator.Calculation());  
 }  
 @org.junit.jupiter.api.Test  
 void calculationMultiple() {  
 generator = new MultipleOperation(2,3);  
 *assertEquals*(6, generator.Calculation());  
 }  
}

Result:



# Appendix – Source code

Main Program :

AddOperation.java

DivideOperation.java

MultipleOperation.java

NumberGenerator.java

SubtractOperation.java

TeachArithmeticFrame.java

Test class:

NumberGeneratorTest.java

The above files are attached in the zip file under /scr and /Test folder.