Practical 3

0Example :

ADT Counter
A counter enables the client to keep count of things.

reset()
Description: Resets the counter to 0.
Postcondition: The count has been reset to 0

increment()
Description: Increase the count value by 1.
Postcondition: The count value has been increased by 1.

integer getCurrentCount()
Description: Retrieve the current count value.
Postcondition: The counter remains unchanged.|

Question 1

a)
//Yeu Yang
Rational fraction: a fraction of which both numerator and denominator are rational numbers or are polynomials

setNumerator(integer)
Description: provide data to store in numerator member variable
Parameters: an integer value
Preconditions: None
Postconditions: the value of the integer will be stored in the numerator member variable

setDenomitator(integer)
Description: provide data to store in denominator member variable
Parameters: an integer value
Preconditions: None
Postconditions: the value of the integer will be stored in the denominator member variable

getNumerator()
Description: access data stored in numerator member variable
Parameters: None
Preconditions: None
Postconditions: None
Returns: integer data stored in numerator member variable

getDenominator()
Description: access data stored in denominator member variable
Parameters: None
Preconditions: None

Postconditions: None
Returns: integer data stored in denominator member variable

//Yin Lam - addition
fraction sum(fraction)
Description: Sum up the two fractions.
Parameters: two fraction values
Precondition: Both fractions must contain valid values. (denominator != 0)
Postcondition: None
Return: a fraction value

//Yong Chen - subtraction
fraction subtraction(fraction)
Description: Subtract the current fraction with the parameter fraction
Precondition:Both fractions contain valid values (denominator != 0)
Postcondition: None
Return: Resulting fraction

//Yong Kang = multiplication
multiplication(fraction)
Description: Multiple the first fraction with second fraction
Parameter: two fraction values
Precondition: Both fractions are in simplest form and contain valid values.(denominator != 0)
Postcondition: None
Return: fraction

//Yong Kit = division
division(fraction)
Description: Divide the first fraction by second fraction
Precondition: Both fractions must contain valid values and the second fraction must be not 0.
Postcondition: None.
Return: fraction value.

b) Fraction Interface

```
public interface FractionInterface{
	//Choon Peng = setters
	public void setNumerator(int numerator);
	public void setDenomitator(int denominator);

	//Dih Yong = getters
	int getNumerator();
	int getDenomitator();

	//Han Yao = sum
	FractionInterface sum(FractionInterface fraction);
	//Hao Han = subtraction
	FractionInterface subtraction(FractionInterface fraction);
	//Joan = division
	FractionInterface division(FractionInterface fraction);
	//Hui Shuang = multiplication
	FractionInterface multiplication(FractionInterface fraction);

}
```

c) Fraction Class

```
public class Fraction implements FractionInterface{
	//Lee Yong = attributes
	int numerator, denominator;

	//Choon Peng = setters
	public void setNumerator(int numerator)
	{
		this.numerator=numerator;
	}
	public void setDenomitator(int denominator)
	{
		this.denominator =denominator;
	}

	//Dih Yong = getters
	public int getNumerator(){
		return numerator;
	}
	public int getDenominator(){
		return denominator;
	}
```

```
//Han Yao = sum
public FractionInterface sum(FractionInterface fraction){
      return ( numerator * fraction.getDenominator() + denominator *
      fraction.getNumerator() ) / denominator *
      fraction.getDenominator();
}

//Hao Han = subtraction
public FractionInterface subtraction(FractionInterface fraction){
      return (numerator * fraction.getDenominator - denominator *
      fraction.getNumerator) / numerator * fraction.getDenominator;
}

//Joan = division
public FractionInterface divide(FractionInterface fraction) {
      return((numerator * fraction.getDenominator()) / (denominator
      * fraction.getNumerator()));
}

//Hui Shuang = multiplication
public FractionInterface multiply(FractionInterface fraction){
      return((numerator * fraction.getDenominator()) / (denominator
      * fraction.getNumerator()));
}

//JiaJian
@Overide
public String toString(){
      Return numerator +"/"+ denominator;
}
```

Note: Show GUI (button + event handler)

Question 2

a. Write the SquareMatrix ADT

//Jun Yan

SquareMatrix : A matrix which represents a two-dimensional array of integers with *n* rows and *n* columns.

makeEmpty(m)

Description : which sets the first m rows and columns to zero

Parameter : an integer value , m .

Postcondition : All the entry m rows and columns will be reset to 0.

storeValue(i, j, value)

Description : which stores value into the position at row i, column j.

Parameter : Three integer values , i , j and value .

Postcondition : The row i ,and j column array will be assigned to **value**.
Return: return true if the matrix value is successfully added.

Add(matrix)
Description : Adds two matrices together
Postcondition ：The current **matrix** value will be added.
Copy(matrix)
Description : Copies one matrix into another
Postcondition : The current **matrix** value will be copied.

b. Translate the ADT specification from part (a) into a Java interface.

```
//Kah Yee
public interface SquareMatrixInterface {

    //sets the first m rows and columns to zero
    public void makeEmpty(int m);

    //stores value into the position at row i, column j.
    public boolean storeValue(int i, int j, int value);

    //adds two matrices together
    public void add(int[][] matrix);

    //copies one matrix into another
    public void copy(int[][] matrix);
}
```

c. Create a Java class that implements the interface. Override the toString()

```
//Kean Min - Create header of the Java class + declare data member
public class Square implements SquareMatrixInterface{
      private int[][] matrix;

//Kuan Xian - constructor
public Square(int rowCol){
      matrix = new int [rowCol][rowCol];
}

//Lee Ling - makeEmpty
public void makeEmpty(int m){
      if(m > matrix.length)
            m = matrix.length;

      for(int i= 0; i < m; i++){
            for(int j = 0; j < m; j++)
                  matrix[i][j] = 0;
      }
}
```

```
//Ming Yeu - storeValue
public boolean storeValue(int i, int j, int value)
{
	if(i <= matrix.length && i > 0 && j <= matrix.length && j >
0){
		matrix[i - 1][j - 1] = value;
		return true;
	}
	return false;
}
```

	0	1	2
0	5		
1			
2			7

```
matrixA.storeValue(1,1, 5);
matrixA.storeValue(3,3,7);
matrixA.storeValue(4,4,8); no response
matrixA.storeValue(0,1, 10); no response

//Wai Kian - add
@Override
public void add (int [][] matrix){
	for(int i=0;i<matrix.length;i++){
		for(int j=0;j<matrix.length;j++){
			this.matrix[i][j] += matrix[i][j];
		}
	}
}


//Raphael - Copy
public void copy(int [][] matrix){
	for(int i = 0;i<matrix.length;i++){
		for(int j = 0;j<matrix.length;j++){
			this.matrix[i][j] = matrix[i][j];
		}
	}
}

//Yann Tang - toString
@Override
public String toString(){
 String str = “”;
```

```
    for (int i = 0; i< matrix.length; i++){
     for (int j = 0; j < matrix.length; j ++){
      str += String.format("%2d", this.matrix[i][j])+ " ";
      }

     str += "\n";
     }
   return str;
   }

   }//end of class
```

d. Create a simple driver program that tests the ADT.