Assignment 1: Command line arguments

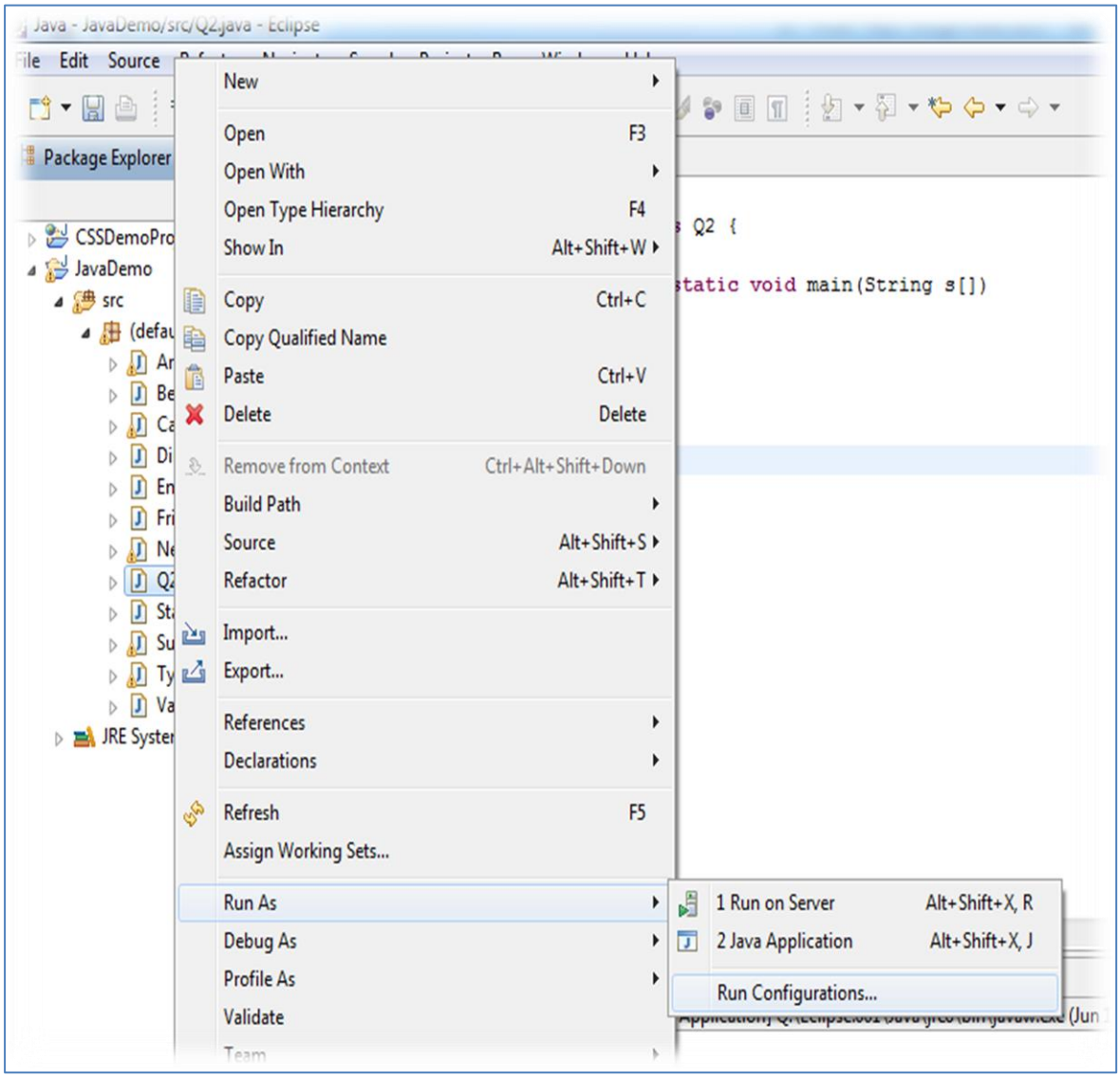
Problem description:

Write a program which accepts set of integers at command prompt and displays second largest and second smallest numbers as output.

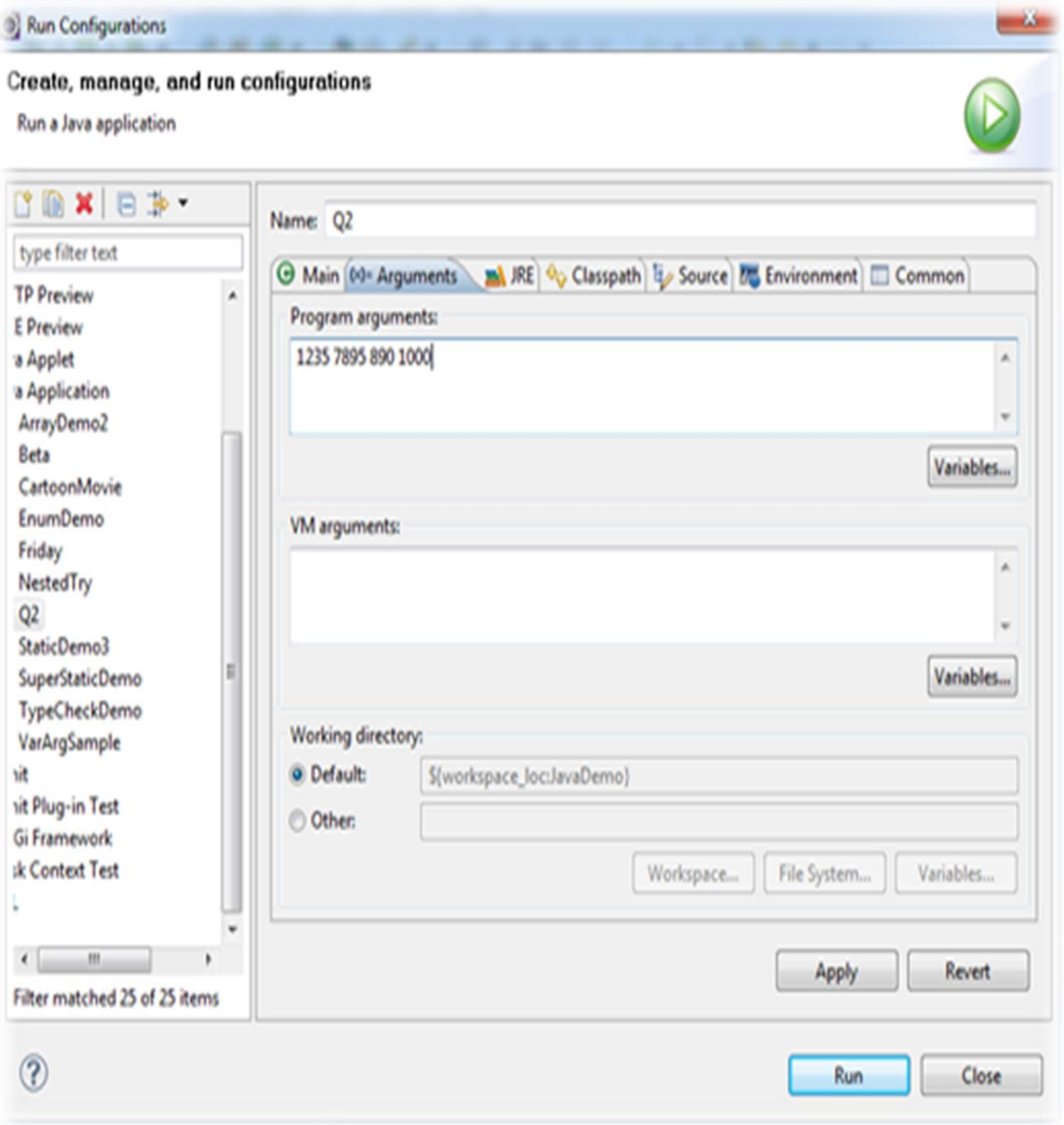
Example:

Step1:

Command line arguments are passed as shown below (select run configuration):



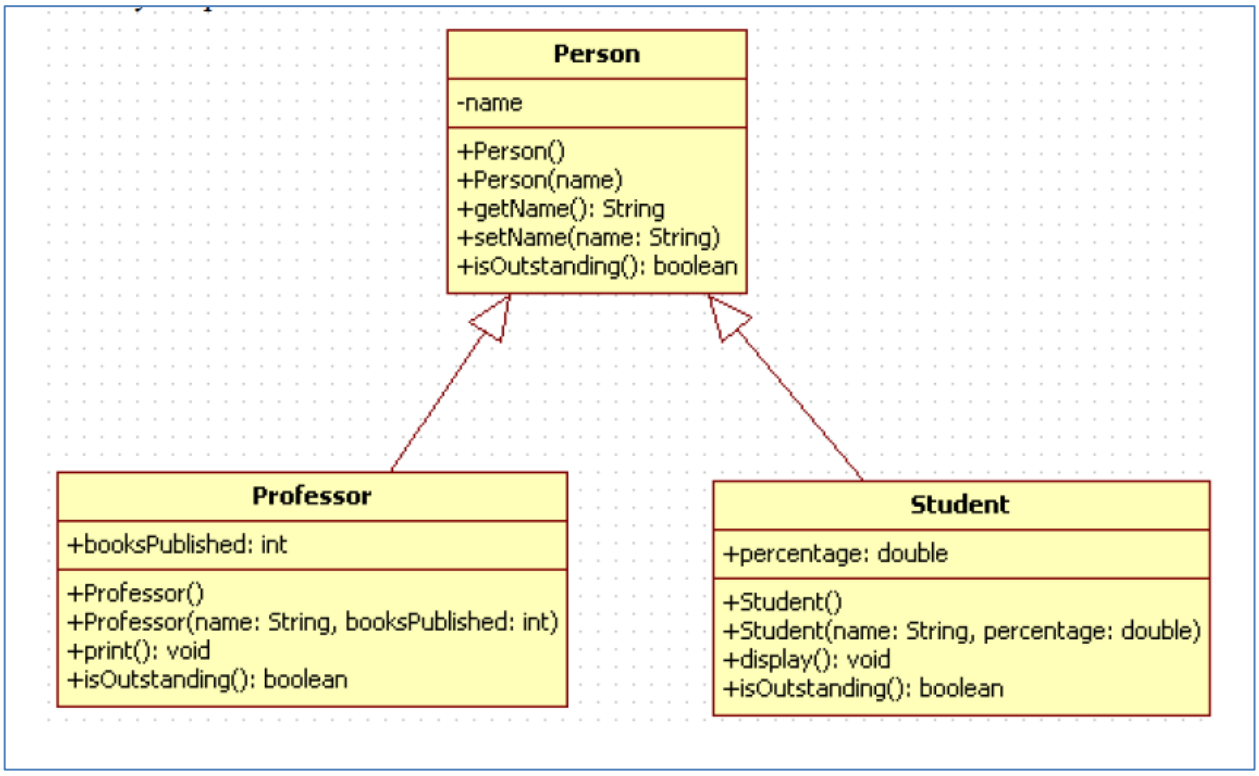
Step2:
Select second tab and pass the arguments, delimited by space, as shown below:



Assignment 2: Method Overriding

Problem description:

Write an application which implements below UML diagram:



Following operations must be implemented:

1) Complete the entity classes shown in Class diagram

2) Implement isOutstanding( ) based on the following rules:

- Professor is outstanding if he has published more than 4 books
- Student is outstanding if his percentage is greater than or equal to 85.

3) The print ( ) method of Professor displays the name and books published by professor.

4) The display( ) method of Student displays the name and percentage of student

5) Complete the main ( ) method by creating PersonExample.java to implement the following:

- Store 5 persons in a single collection of array type (Person[ ]) , this can be a combination of students and professors
- Using Run-Time Type Identification print Professor and Student details.
- Identify all outstanding persons

Assignment 3: Method overloading

Problem description: Below code generates an error. Analyze the code and debug it. Read the method overloading guidelines:

```
public class AddDemo {

	void add(double num1, int num2) {
		double sum = num1 +num2;
		System.out.println(sum);
	}

	void add(double num1, double num2) {
		double sum = num1 +num2;
		System.out.println(sum);
	}

	void add(int num1, double num2) {
		double sum = num1 +num2;
		System.out.println(sum);
	}

	public static void main(String[] demo) {
		AddDemo obj = new AddDemo();

		obj.add(10, 20);
		obj.add(15.5, 6);
	}
}
```

Assignment 4: Variable parameter lists

Problem description:

A client wants to automate the process of displaying passengers list who is onboarding to flight. The list could vary from time to time. Develop an application which uses single method to display names of variable length of arguments.

Example: The method can be invoked by passing individual strings like:
"Rama", "Laxmana", "Bharatha", "Shatrughna" Or
the method can also be invoked by passing array of strings: String passengers [];
Here is sample client code, which invokes method which accepts variable length of arguments:

```
public static void main(String[] args) {
    displayPassengers(1, "Rama", "Laxmana", "Bharatha", "Shatrughna");

    String[] passengers = {"Ravana", "Vibhishana", "Kumbhakarna", "Shoorpanakha"};

    displayPassengers(1, passengers);
}
```

Assignment 5: Comparing Objects

Problem description:

A college admin would like to compare students of two batches based on their semester total score.

Create a class StudentComparison.java to implement

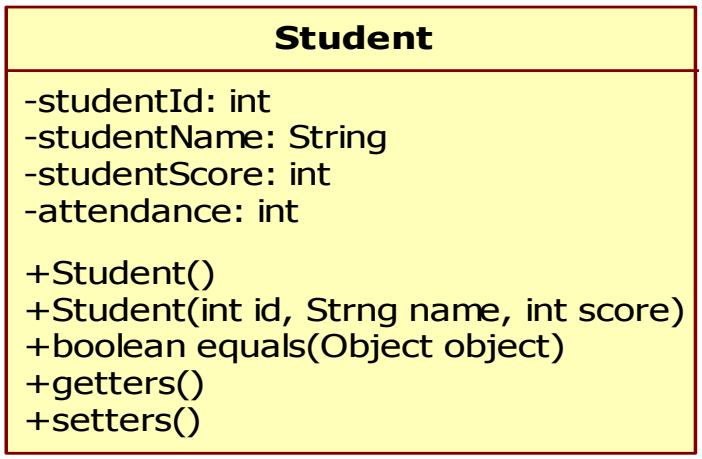
- Student compareStudents(Student batch1[], Student batch2[])

  This will return student object who has secured highest score from array of student objects from batch1 and batch2. If the student scores from both batches are same scores than return the student who has attended maximum number of classes. (If the attendance is also same, than return the oldest student between them)

Create a TestMain.java to implement

- main() method which creates StudentComparison class and invoke compareStudents() method.

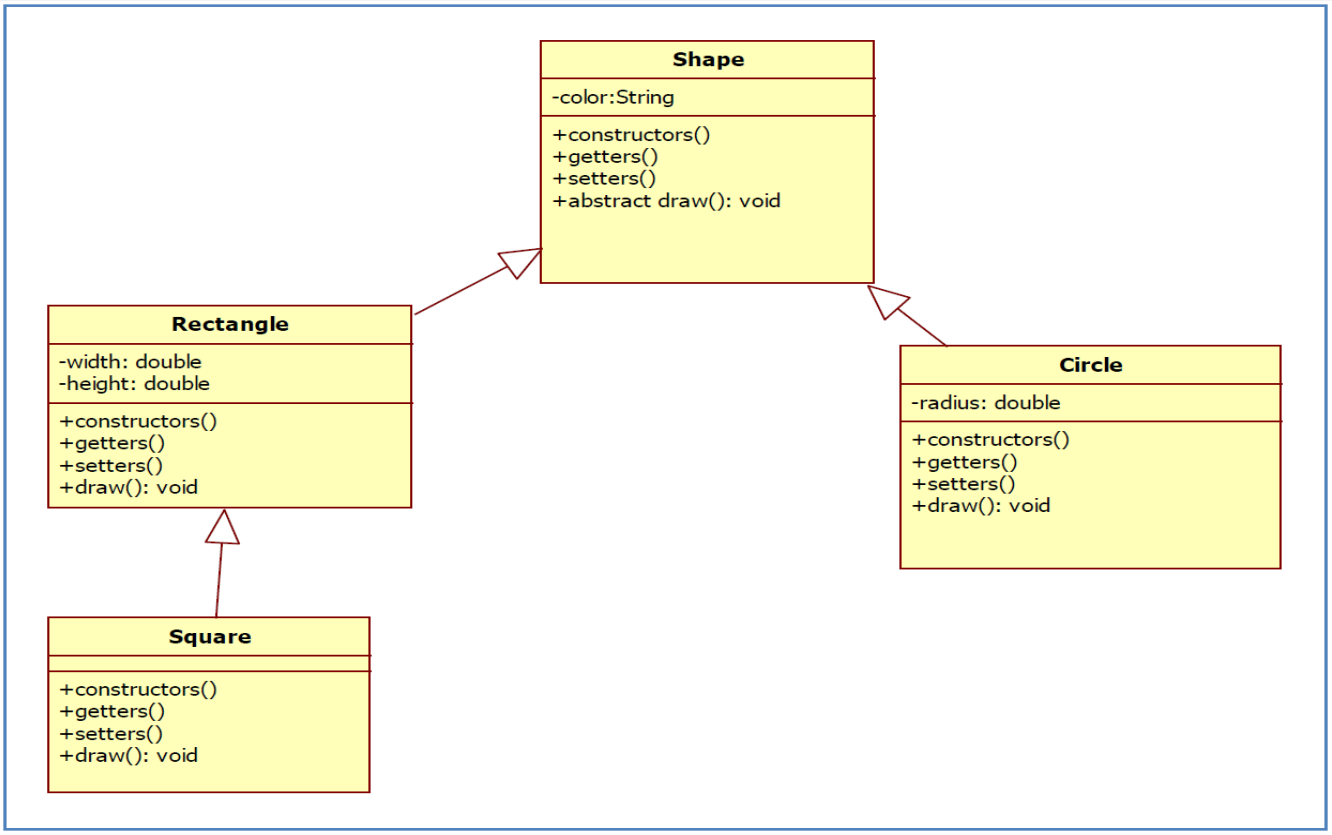
Below is the Student class diagram. Override the equals() method to compare two objects.



Assignment 6: Abstract Class

Problem description:

Write an application which implements below UML diagram:



Here draw() method used to set color of a shape and to calculate area of respective shape. Below is the sample code for shape "Rectangle":

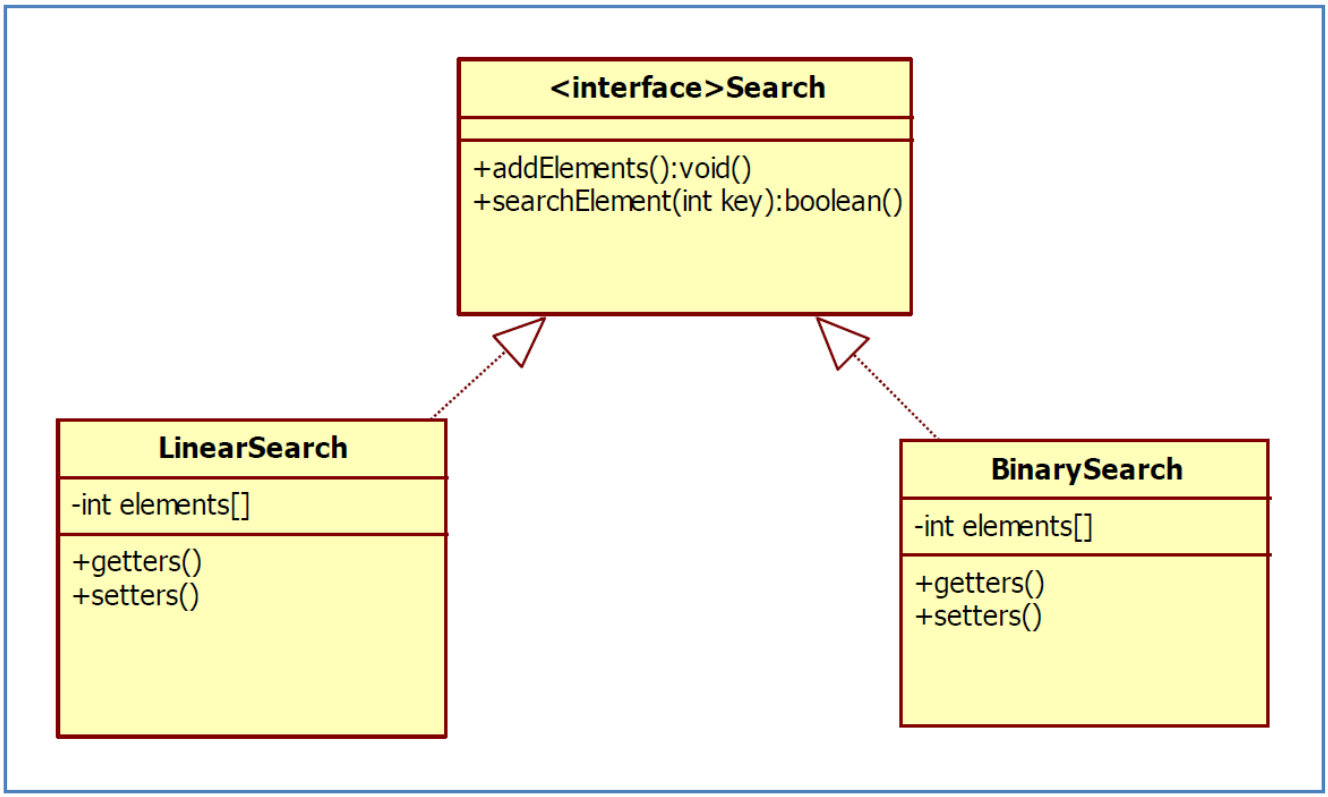
```
public void draw() {
    setHeight(10);
    setWidth(20);
    setColor("Green");
    double area = getHeight() * getWidth();
    System.out.println("Rectangle filled with " + getColor() + " and Area = " + area);
}
```

Write separate class "TestShape" which test above application using main method.

Assignment 7: Interface

Problem description:

Write a program which implements following UML class diagram:



1. addElements() – Should implement adding elements to array in both 'LinearSearch' and 'BinarySearch' classes.
2. searchElement(int key) – Should search array for given 'key' element. If found then it must return 'true' otherwise return 'false'.

Use separate class called "TestSearch" to test above code by creating appropriate object in main method. Print following message based on return value of 'searchElement(int key) method

Assignment 8: Unchecked Exceptions

Problem description:

Observe the below code and will it always get executed?

```
public class MathDivision {
    public static void main(String args[]) {
        int num1 = 10;
        int num2 = 0;
        double d = (double) (num1 / num2);
        System.out.println(d);
    }
}
```

If this code does not give guarantee of executing always, how can we use try catch block to ensure its execution? Rewrite the code using try catch block.

Assignment 9: Checked Exceptions

Problem description:

class Class

Instances of the class Class represent classes and interfaces in a running Java application.
Below is brief description about two of its methods.

- **forName(String className)**
  Returns the Class object associated with the class or interface with the given string name.

And it throws: **ClassNotFoundException**, if the class cannot be located by the specified class loader

- **newInstance()**
  Creates a new instance of the class represented by Class object. The class is instantiated as if by a new expression with an empty argument list. The class is initialized if it has not already been initialized.

And it throws: **IllegalAccessException** - if the class or its constructor is not accessible.

Here is the sample code of using these methods:

```
package bank;

public class Account {
    public Account() {
        System.out.println("Account class");
    }
}
```

```
package bank;

public class TestAccount {
    public static void main(String[] args) throws Exception {
        Class class1 = Class.forName("bank.Account");
        Account account = (Account) class1.newInstance();
    }
}
```

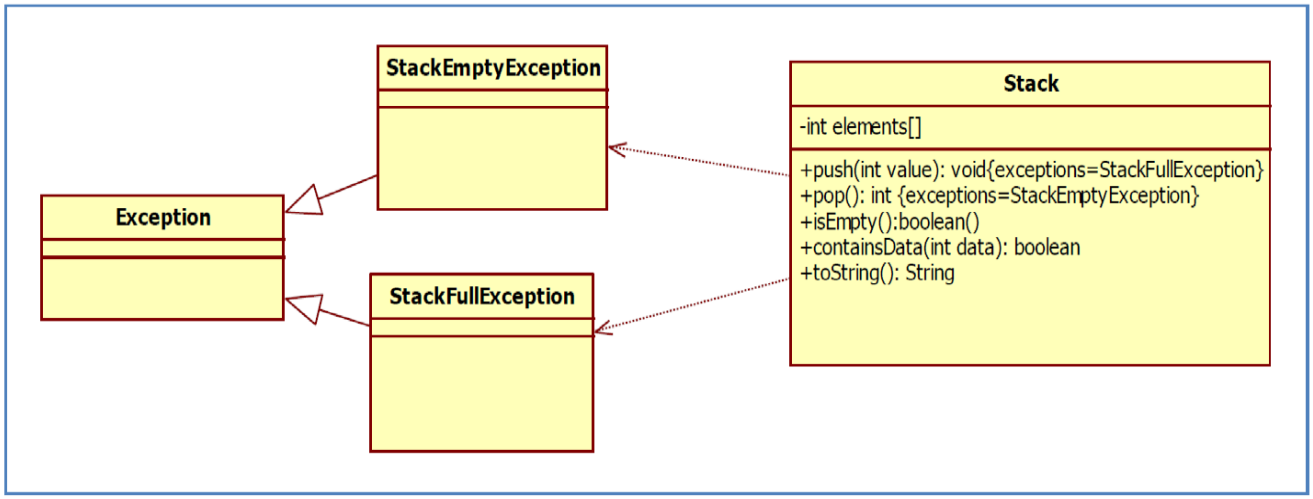
Execute the above code and observe the result. In case of compilation error, fix it using appropriate try and catch block.

Assignment 10: User Defined Exception

Problem description:

Stack is a LIFO (LAST-IN-FIRST-OUT) data structure.

Examples of real-time stack implementations are Calculator, removing a book from a pile of books, etc. Write an application which implements stack data structure.
Below is the UML diagram of same:



Note:

a) Do not use built-in legacy java.util.Stack class.
b) Create exception classes StackFullException and StackEmptyException.
c) Create a Stack class to implement FILO with the following methods:

1. public void push(int value) throws StackFullException
   -) Should add the data into an integer array
   -) Should throw StackFullException, if stack the number of elements added exceed the Stack size which is mentioned while stack creation.

2. public int pop( ) throws StackEmptyException.
   -) Should return the last added element and remove that element from Stack
   -) Should throw StackEmptyException if no elements exist

3. public boolean isEmpty()
   -) returns true if stack is empty

4. public boolean containsData(int data)
   -) returns true if specified element exists in stack

5. Override toString()
   -) Prints all the elements in Stack