NOTE: Classes from package java.lang. do not need to be imported, for example String , Object Classes which are from java.lang package. The java.lang package is implicitly imported by every one of the Java source files so no need to import. **You need to explicitly** import Exception Classes contained in other Java packages like the exceptions inside the Java io package e.g. FileNotFoundException, IOException, InterruptedIOException.
Also note that Java defines several exception classes inside the standard package java.lang and **you don't need** to import these Exceptions:NullPointerException, ArithmeticException, IllegalStateException, IndexOutOfBoundsException etc.

1) import java.util.Scanner;

```
public class Calculator {
   private double num1, num2;

   // No-argument constructor
   public Calculator() {
      this.num1 = 0.0;
      this.num2 = 0.0;
   }

  public Calculator(double num1, double num2) {
      this.num1 = num1;
      this.num2 = num2;
   }

   // Accessor (Getter) for num1
   public double getNum1() {
      return num1;
   }

   // Accessor (Getter) for num2
   public double getNum2() {
      return num2;
   }

   // Mutator (Setter) for num1
   public void setNum1(double num1) {
      this.num1 = num1;
   }

   // Mutator (Setter) for num2
   public void setNum2(double num2) {
      this.num2 = num2;
```

```
    }

    // Method for addition
    public double Add() throws ArithmeticException {
        if (num1 < 0 || num2 < 0) {
            throw new ArithmeticException("Addition Error: Numbers should be positive.");
        }
        return num1 + num2;
    }

    // Method for subtraction
    public double Subtract() throws ArithmeticException {
        if (num1 < 0 || num2 < 0) {
            throw new ArithmeticException("Subtraction Error: Numbers should be positive.");
        }
        return num1 - num2;
    }

    // Method for multiplication
    public double Multiply() throws ArithmeticException {
        if (num1 == 0 || num2 == 0) {
            throw new ArithmeticException("Multiplication Error: Numbers should not be zero.");
        }
        return num1 * num2;
    }

    // Method for division
    public double Divide() throws ArithmeticException {
        if (num2 == 0) {
            throw new ArithmeticException("Division Error: Cannot divide by zero.");
        }
        return num1 / num2;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) { //keep looping until a non-numeric input is given
            try {
                // Calculator objects for different operations
                Calculator obj1 = new Calculator();
                Calculator obj2 = new Calculator();
```

```
            Calculator obj3 = new Calculator();
            Calculator obj4 = new Calculator();

            // Taking input for first object (Addition)
            System.out.println("Enter two numbers for addition (obj1): ");
            obj1.setNum1(Double.parseDouble(scanner.next())); //converts any number to double. If
                                                              //string is given, will throw NumberFormatException

            obj1.setNum2(Double.parseDouble(scanner.next()));
            System.out.println("Addition result: " + obj1.Add());

            // Taking input for second object (Subtraction)
            System.out.println("Enter two numbers for subtraction (obj2): ");
            obj2.setNum1(Double.parseDouble(scanner.next()));
            obj2.setNum2(Double.parseDouble(scanner.next()));
            System.out.println("Subtraction result: " + obj2.Subtract());

            // Taking input for third object (Multiplication)
            System.out.println("Enter two numbers for multiplication (obj3): ");
            obj3.setNum1(Double.parseDouble(scanner.next()));
            obj3.setNum2(Double.parseDouble(scanner.next()));
            System.out.println("Multiplication result: " + obj3.Multiply());

            // Taking input for fourth object (Division)
            System.out.println("Enter two numbers for division (obj4): ");
            obj4.setNum1(Double.parseDouble(scanner.next()));
            obj4.setNum2(Double.parseDouble(scanner.next()));
            System.out.println("Division result: " + obj4.Divide());

        } catch (NumberFormatException e) {
            System.out.println("Input Error: Please provide valid numeric values.");
            break; // Exit the loop if a non-numeric input is provided
        } catch (ArithmeticException e) {
            // Catch arithmetic exceptions such as division by zero or negative numbers
            System.out.println(e.getMessage());
        }
    }

  }
}
```

//The NumberFormatException occurs when an attempt is made to convert a string with improper format into a numeric value.

2)

```
// Define the FruitException class that extends Exception which is built-in
class FruitException extends Exception {
    // Constructor that accepts a string message
    public FruitException(String message) {
        // Pass the message to the superclass (Exception) constructor
        super(message);
    }
}

// Define the Fruit class
public class Fruit {

    // Method to calculate fruit price and throw exceptions based on conditions
    public void fruitPrice(double price, double weight) throws FruitException {
        // Check the price and throw the appropriate exception or print the total price
        if (price < 50) {
            throw new FruitException("Available Fruit!");
        } else if (price > 500) {
            throw new FruitException("Rare Fruit!");
        } else {
            // Calculate and print the total price of the fruit
            double totalPrice = price * weight;
            System.out.println("The total price of the fruit is: $" + totalPrice);
        }
    }

    // Main method to demonstrate the usage
    public static void main(String[] args) {
        // Create an instance of the Fruit class
        Fruit fruit = new Fruit();

        // Test cases to show different behaviors
        try {
            // Test with price less than 50
            fruit.fruitPrice(30, 2); // This should throw "Available Fruit!"
        } catch (FruitException e) {
            System.out.println(e.getMessage());
        }

        try {
            // Test with price greater than 500
            fruit.fruitPrice(600, 1); // This should throw "Rare Fruit!"
```

```
        } catch (FruitException e) {
            System.out.println(e.getMessage());
        }

        try {
            // Test with price between 50 and 500
            fruit.fruitPrice(100, 3); // This should print the total price
        } catch (FruitException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

FruitException Class:

- Inherits from Exception.
- Uses a constructor to pass the message to the Exception superclass.

System.err.println() sends the output to the error stream (System.err), not the standard output (System.out).
The output is typically displayed in red in most IDEs or terminals configured to differentiate error streams. It is an output stream like System.out, but it is specifically meant for error messages and diagnostics.
Printing ex (an instance of Throwable, typically an exception like Exception or Error) directly will call its toString() method. By default, Throwable.toString() returns a string in the format:
<ClassName>: <Message>

3)

```
public class MultiCatchExample {

    public static void main(String[] args) {
        try {
            // Method call that may throw multiple exceptions
            performOperations();
        } catch (ArithmeticException e) {
            System.out.println("Caught an ArithmeticException: " + e.getMessage());
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught an ArrayIndexOutOfBoundsException: " + e.getMessage());
        } catch (NullPointerException e) {
            System.out.println("Caught a NullPointerException: " + e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Caught a NumberFormatException: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Caught a general Exception: " + e.getMessage());
```

```
        } finally {
            // This block will always execute regardless of whether an exception was thrown or not
            System.out.println("This block always executes, regardless of an exception.");
        }
    }

    public static void performOperations() {
        // Example code that might throw different types of exceptions

        // ArithmeticException (division by zero)
        int result = 10 / 0;

        // ArrayIndexOutOfBoundsException (accessing an invalid index)
        int[] numbers = new int[5];
        numbers[10] = 100;

        // NullPointerException (dereferencing a null object)
        String text = null;
        System.out.println(text.length());

        // NumberFormatException (parsing an invalid number format)
        int number = Integer.parseInt("invalid_number");//giving string as input but expecting int.

        // Example of a generic exception (will not be reached in this case)
        throw new RuntimeException("Generic exception for demonstration");
    }
}

4)

public class NestedTryCatchExample {

    public static void main(String[] args) {
        try {
            // Outer try block
            System.out.println("Starting outer try block.");

            try {
                // First nested try block
                System.out.println("Starting first nested try block.");
                performFirstLevelOperations();
            } catch (ArithmeticException e) {
                System.out.println("Caught ArithmeticException in first nested try: " +
e.getMessage());
```

```
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught ArrayIndexOutOfBoundsException in first nested try: " +
e.getMessage());
        }

        try {
            // Second nested try block
            System.out.println("Starting second nested try block.");
            performSecondLevelOperations();
        } catch (NullPointerException e) {
            System.out.println("Caught NullPointerException in second nested try: " +
e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Caught NumberFormatException in second nested try: " +
e.getMessage());
        }

    } catch (Exception e) {
        System.out.println("Caught general Exception in outer try: " + e.getMessage());
    } finally {
        // Finally block for the outer try
        System.out.println("This block always executes, regardless of an exception in outer
try.");
    }
  }

  public static void performFirstLevelOperations() {
    try {
        // Third nested try block
        System.out.println("Starting third nested try block.");
        int result = 10 / 0; // This will cause an ArithmeticException

        try {
            // Fourth nested try block
            System.out.println("Starting fourth nested try block.");
            int[] numbers = new int[5];
            numbers[10] = 100; // This will cause an ArrayIndexOutOfBoundsException

            try {
                // Fifth nested try block
                System.out.println("Starting fifth nested try block.");
                String text = null;
                System.out.println(text.length()); // This will cause a NullPointerException
            } catch (NullPointerException e) {
```

```
                System.out.println("Caught NullPointerException in fifth nested try: " +
e.getMessage());
            }

        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught ArrayIndexOutOfBoundsException in fourth nested try: " +
e.getMessage());
        }

    } catch (ArithmeticException e) {
        System.out.println("Caught ArithmeticException in third nested try: " + e.getMessage());
    }
  }

  public static void performSecondLevelOperations() {
    try {
        // Another block of code that might throw exceptions
        System.out.println("Performing second level operations.");

        int number = Integer.parseInt("invalid_number"); // This will cause
NumberFormatException
    } catch (NumberFormatException e) {
        System.out.println("Caught NumberFormatException in second level operations: " +
e.getMessage());
    }
  }
}

5)

// Custom exception class MyException
class MyException extends Exception {
  private int param;

  // Constructor to initialize the exception with a parameter
  public MyException(int param) {
    this.param = param;
  }

  // Override the getMessage method to return the custom message
  @Override
  public String getMessage() {
    return "MyException[" + param + "]";
  }
```

```
}
```

//Main Class with Exception Handling

```
public class ExceptionHandlingExample {

    public static void main(String[] args) {
        try {
            // Code that may throw exceptions
            performOperations();
        } catch (ArithmeticException e) {
            // Handle division by zero
            System.out.println("Invalid division");
        } catch (NumberFormatException e) {
            // Handle string parsed to a numeric variable
            System.out.println("Format mismatch");
        } catch (StringIndexOutOfBoundsException e) {
            // Handle accessing an invalid index in string
            System.out.println("Index is invalid");
        } catch (ArrayIndexOutOfBoundsException e) {
            // Handle accessing an invalid index in array
            System.out.println("Array index is invalid");
        } catch (MyException e) {
            // Handle MyException
            System.out.println(e.getMessage());
        } catch (Exception e) {
            // Handle any other exception
            System.out.println("Exception encountered");
        } finally {
            // This block will always execute
            System.out.println("Exception Handling Completed");
        }
    }

    public static void performOperations() throws MyException {
        // Example operations that might throw exceptions
        int[] numbers = new int[5];
        String text = "Example";

        // Division by zero
        int result = 10 / 0;

        // Parsing a string to an integer
        int number = Integer.parseInt("invalid_number");
```

```
        // Accessing an invalid index in string
        char c = text.charAt(100);

        // Accessing an invalid index in array
        numbers[10] = 100;

        // Throwing a custom exception
        throw new MyException(5);
    }
}
```