PSG COLLEGE OF TECHNOLOGY, COIMBATORE - 641 004
Department of Applied Mathematics and Computational Sciences

MSc Software Systems - Semester V
20XW57- Java Programming Lab
PROBLEM SHEET 6 - Exception Handling

Completion Date: 16/08/2024
***Note: Use Text Editors/IntelliJ IDEA/Apache NetBeans tools to develop, compile and execute the* below programs**

1. Write a program for example of try and catch block. In this check whether the given array size is negative or not.

Code:

```
import java.util.Scanner;

public class TryCatch
{
   public static void main(String[] args)
   {
       try
       {
           System.out.print("Enter Array Size: ");
           Scanner sc = new Scanner(System.in);
           int num = sc.nextInt();
           int[] a = new int[num];
           System.out.println("Array Size: " + a.length);
       }
       catch(NegativeArraySizeException n)
       {
           System.out.println("Generated exception: " + n);
```

```
            System.out.println("Since the exception was caught,
code after the try block will continue to run");
        }


    }
}
```

Output:

```
Enter Array Size: 3
Array Size: 3
```

```
Enter Array Size: -7
Generated exception: java.lang.NegativeArraySizeException: -7
Since the exception was caught, code after the try block will continue to run
```

2. Write a program for example of multiple catch statements occurring in a program.

Code:

```
import java.util.Scanner;
public class TryMultipleCatch
{
   public static void main(String[] args)
   {
        Scanner sc = new Scanner(System.in);
        try
        {
            System.out.print("Enter array size: ");
            int num = Integer.parseInt(sc.nextLine());
            int[] array = new int[num];
        }
        catch(NegativeArraySizeException n)
        {
            System.out.println("generated exception: " + n);
        }
        catch (NumberFormatException e)
        {
```

```
            System.out.println("generated exception: " + e);
        }
    }
}
```

Output:

```
Enter array size: -12
generated exception: java.lang.NegativeArraySizeException: -12
```

```
Enter array size: 1ad3f
generated exception: java.lang.NumberFormatException: For input string: "1ad3f"
```

3. Write a program to illustrate sub class exception precedence over base class.

Code:

```
public class ExceptionPrecedence {
    public static void main(String[] args) {
        try {
            throw new ArithmeticException("Arithmetic
Exception");
        } catch (ArithmeticException e) {
            System.out.println("Caught ArithmeticException: " +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Caught Exception: " +
e.getMessage());
        }
    }
}
```

Output:

```
Caught ArithmeticException: Arithmetic Exception
```

4. Write a program to do division of two numbers. Make function called doDivision( )
to do division of two numbers. During the division process if runtime error may
occur throw error to calling function. Handle ArithmeticException,
ArrayIndexOutofBoundsException, NumberFormatException in main( ) method only.
Code:

```
public class DivisionHandling {

   public static void doDivision(int numerator, int denominator)
throws ArithmeticException {
       if (denominator == 0) {
           throw new ArithmeticException("Division by zero");
       }
       int result = numerator / denominator;
       System.out.println("Result: " + result);
   }

   public static void main(String[] args) {
       try {
           doDivision(10, 0); // This will throw
ArithmeticException
       } catch (ArithmeticException e) {
           System.out.println("Caught ArithmeticException: " +
e.getMessage());
       }

       try {
           String[] arr = new String[5];
           System.out.println(arr[10]); // This will throw
ArrayIndexOutOfBoundsException
       } catch (ArrayIndexOutOfBoundsException e) {
           System.out.println("Caught
ArrayIndexOutOfBoundsException: " + e.getMessage());
       }
```

```
        try {
            int number = Integer.parseInt("abc"); // This will
throw NumberFormatException
        } catch (NumberFormatException e) {
            System.out.println("Caught NumberFormatException: " +
e.getMessage());
        }
    }
}
```

Output:

```
Caught ArithmeticException: Division by zero
Caught ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5
Caught NumberFormatException: For input string: "abc"
```

5. Write a program in Java to display name and roll number of students. Initialize respective array variables for 10 students. Handle ArrayIndexOutOfBoundsExeption, so that any such problem doesn't cause illegal termination of program.

Code:

```
public class Student {
    String rollNo;
    String name;

    Student(String rollNo, String name)
    {
        this.name = name;
        this.rollNo = rollNo;
    }

    public static void main(String[] args)
    {
        Student[] s = new Student[10];
```

```
        for(int i = 0; i < 10; i++)
        {
            s[i] = new Student("D22PWX", "Dummy");
        }

        try
        {
            for (int i = 0; i <= 10; i++)
            {
                System.out.println("Name: " + s[i].name + "\t
Roll Number: " + s[i].rollNo);
            }
        }
        catch(ArrayIndexOutOfBoundsException a)
        {
            System.out.println("EXCEPTION: " + a);
        }
    }
}
```

Output:

```
Name: Dummy      Roll Number: D22PWX
Name: Dummy      Roll Number: D22PWX
EXCEPTION: java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10
```

6. Create a user-defined exception class called "NoMatchException" using extends keyword.
Write a constructor for this class that takes a string argument and stores it inside the object
with a string handle. Write a method that prints out the stored string. Create a try-catch clause
to exercise the created exception when a user entered string is not equal to "India";

Code:

```
import java.util.Scanner;
```

```
class NoMatchException extends Exception {
   private String message;

   NoMatchException(String message) {
       this.message = message;
   }

   public void printMessage() {
       System.out.println(message);
   }
}

public class Example {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter a country name: ");
       String userInput = sc.nextLine();

       try {
           if (!userInput.equals("India")) {
               throw new NoMatchException("Input does not match
'India'");
           }
           System.out.println("Input matches 'India'");
       } catch (NoMatchException e) {
           e.printMessage();
       }
   }
}
```

Output:

```
Enter a country name: Japan
Input does not match 'India'
```

```
Enter a country name: India
Input matches 'India'
```

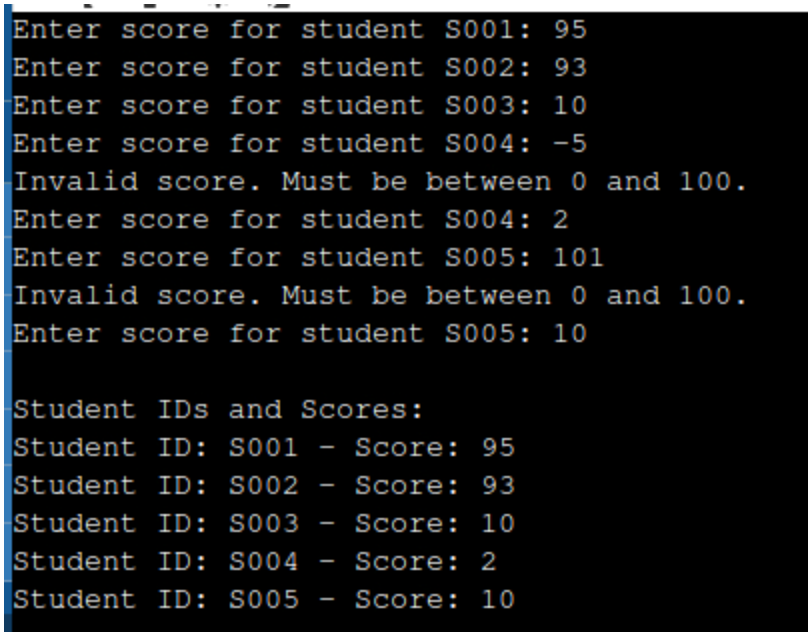
7. Write an application that displays a series of at least five student ID numbers (that you have stored in an array) and asks the user to enter a numeric test score for the student. Create a ScoreException class, and throw a ScoreException for the class if the user does not enter a valid score (less than or equal to 100). Catch the ScoreException, display an appropriate message, and then store a 0 for the student's score. At the end of the application, display all the student IDs and scores.

Code:

```
import java.util.Scanner;

class ScoreException extends Exception {
   public ScoreException(String message) {
       super(message);
   }
}

public class StudentScores {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       String[] studentIDs = {"S001", "S002", "S003", "S004",
"S005"};
       int[] scores = new int[studentIDs.length];

       for (int i = 0; i < studentIDs.length; i++) {
           while (true) {
               try {
                   System.out.print("Enter score for student " +
studentIDs[i] + ": ");
```

```
                    int score = Integer.parseInt(sc.nextLine());

                    if (score < 0 || score > 100) {
                        throw new ScoreException("Invalid score.
Must be between 0 and 100.");
                    }
                    scores[i] = score;
                    break;
                } catch (ScoreException e) {
                    System.out.println(e.getMessage());
                } catch (NumberFormatException e) {
                    System.out.println("Invalid input. Please
enter a numeric value.");
                }
            }
        }

        System.out.println("\nStudent IDs and Scores:");
        for (int i = 0; i < studentIDs.length; i++) {
            System.out.println("Student ID: " + studentIDs[i] + "
- Score: " + scores[i]);
        }
    }
}
```

Output:



8. A typical requirement of a custom exception called "WeakPasswordException" would be for validation purposes. In this exercise, Let's validate a password input. A password is said to be strong if it satisfies the following criteria

i) It should be a minimum of 10 characters and a maximum of 20 characters.

ii) It should contain at least one digit.

iii) It should contain at least one special character (non-numeric, non-alphabetic).

iv) It should contain at least one letter.

If the password fails any one of the criteria, it is considered as weak.

Code:

```
import java.util.Scanner;

class WeakPasswordException extends Exception {
   public WeakPasswordException(String message) {
       super(message);
   }
}
```

```java
public class PasswordValidator {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.print("Enter password: ");
       String password = sc.nextLine();

       try {
           validatePassword(password);
           System.out.println("Password is strong.");
       } catch (WeakPasswordException e) {
           System.out.println(e.getMessage());
       }
   }

   public static void validatePassword(String password) throws
WeakPasswordException {
       if (password.length() < 10 || password.length() > 20) {
           throw new WeakPasswordException("Password must be
between 10 and 20 characters.");
       }

       boolean hasDigit = false;
       boolean hasSpecialChar = false;
       boolean hasLetter = false;

       for (char c : password.toCharArray()) {
           if (Character.isDigit(c)) {
               hasDigit = true;
           } else if (!Character.isLetterOrDigit(c)) {
               hasSpecialChar = true;
           } else if (Character.isLetter(c)) {
               hasLetter = true;
           }
       }

       if (!hasDigit) {
```

```
            throw new WeakPasswordException("Password must
contain at least one digit.");
        }
        if (!hasSpecialChar) {
            throw new WeakPasswordException("Password must
contain at least one special character.");
        }
        if (!hasLetter) {
            throw new WeakPasswordException("Password must
contain at least one letter.");
        }
    }
}
```

Output:

```
Enter password: hahahawewewe
Password must contain at least one digit.
```

```
Enter password: 121212121212!@
Password must contain at least one letter.
```

```
Enter password: javalanguage12@
Password is strong.
```