Name : Nikam Khushi Sahebrao.
PRN : 22620004
Branch : Information Technology
Batch : S1
Subject : Java Programming

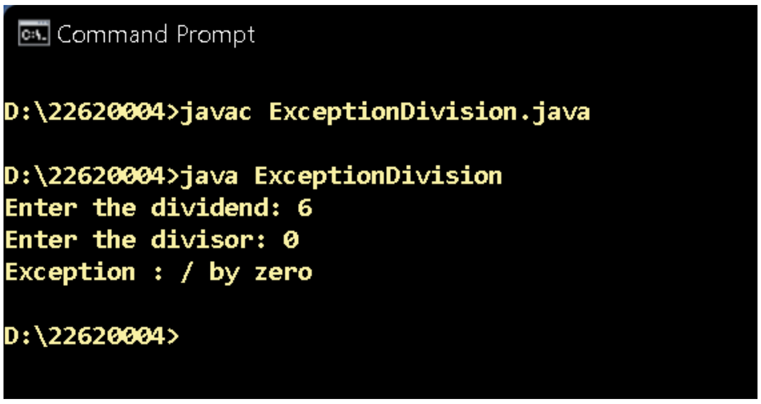
Assignment : 04

1. Write a program in java to handle below exceptions.

a. Divide by Zero
Program :

```java
import java.util.Scanner;

public class ExceptionDivision {
public static void main(String args[]) {

Scanner sc = new Scanner(System.in);
System.out.print("Enter the dividend: ");
int d = sc.nextInt();

System.out.print("Enter the divisor: ");
int div = sc.nextInt();

try {
  int result = d/ div;
  System.out.println("Result: " + result);
} catch (Exception e) {
  System.out.println("Exception : " + e.getMessage());
}
}
}
```

Output :



Command Prompt

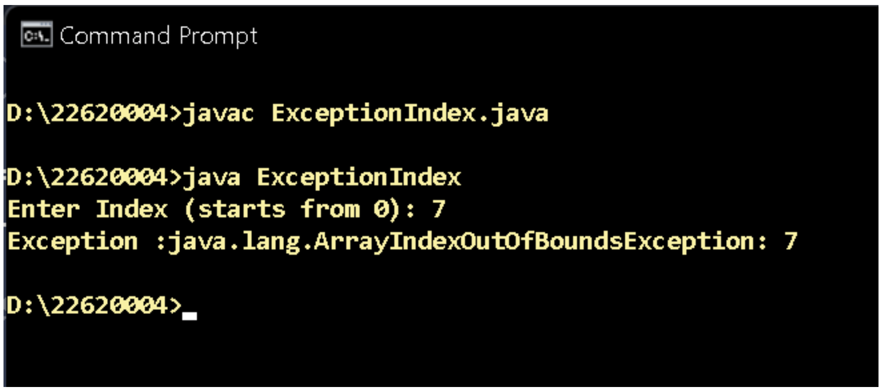
```
D:\22620004>javac ExceptionDivision.java

D:\22620004>java ExceptionDivision
Enter the dividend: 6
Enter the divisor: 0
Exception : / by zero

D:\22620004>
```

b. Array Index Out Of Bound

Program:

```
import java.util.Scanner;
public class ExceptionIndex{
    public static void main(String args[]){
        int[] arr = {10,30,70,90,22};
        int index;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter Index (starts from 0): ");
        index=sc.nextInt();
     try {
      System.out.println("Element at " +index+" : "+ arr[index]);
     } catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception :"+e);
     }
    }
}
```

Output :



Command Prompt

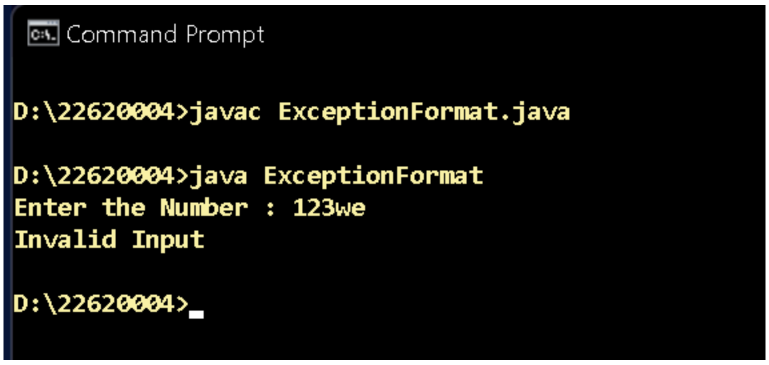
```
D:\22620004>javac ExceptionIndex.java

D:\22620004>java ExceptionIndex
Enter Index (starts from 0): 7
Exception :java.lang.ArrayIndexOutOfBoundsException: 7

D:\22620004>_
```

c. Number format

Program :

```
import java.util.Scanner;

public class ExceptionFormat{
public static void main(String args[]) {

Scanner sc = new Scanner(System.in);
System.out.print("Enter the Number : ");
String d = sc.next();
try {
    int num = Integer.parseInt(d);
    System.out.println("The number is: " + num);
} catch (NumberFormatException e) {
    System.out.println("Invalid Input");
}
}
}
```

Output :



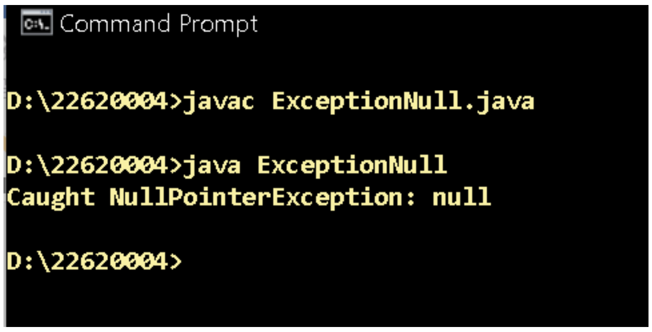
```
Command Prompt

D:\22620004>javac ExceptionFormat.java

D:\22620004>java ExceptionFormat
Enter the Number : 123we
Invalid Input

D:\22620004>
```

d. Null Pointer

Program :

```
import java.util.Scanner;
public class ExceptionNull{
    public static void main(String args[]){
        try {
    String str = null;
    str.toUpperCase();
} catch (NullPointerException e) {
    System.out.println("Caught NullPointerException: " + e.getMessage());
}
    }
}
```

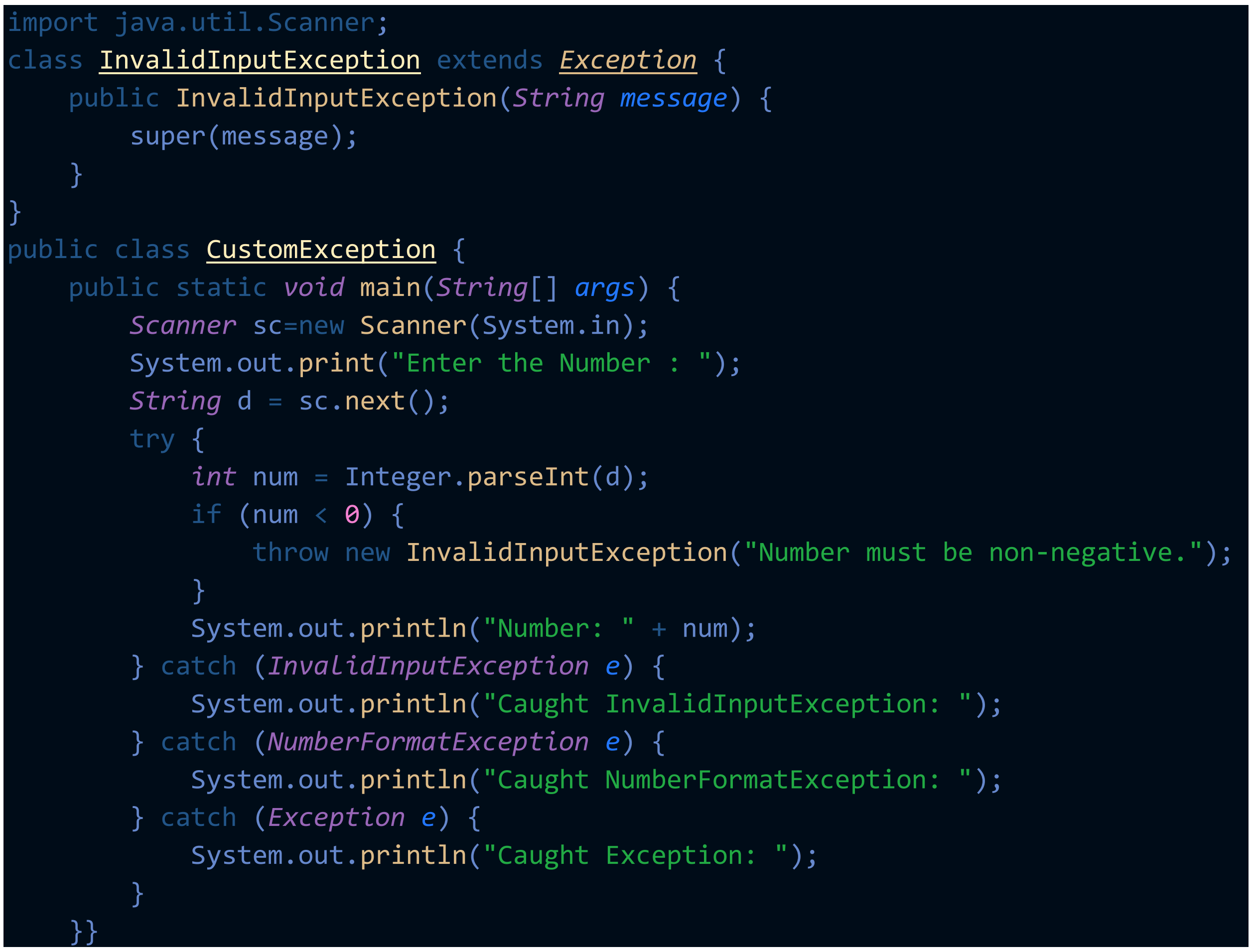
Output :



```
Command Prompt

D:\22620004>javac ExceptionNull.java

D:\22620004>java ExceptionNull
Caught NullPointerException: null

D:\22620004>
```

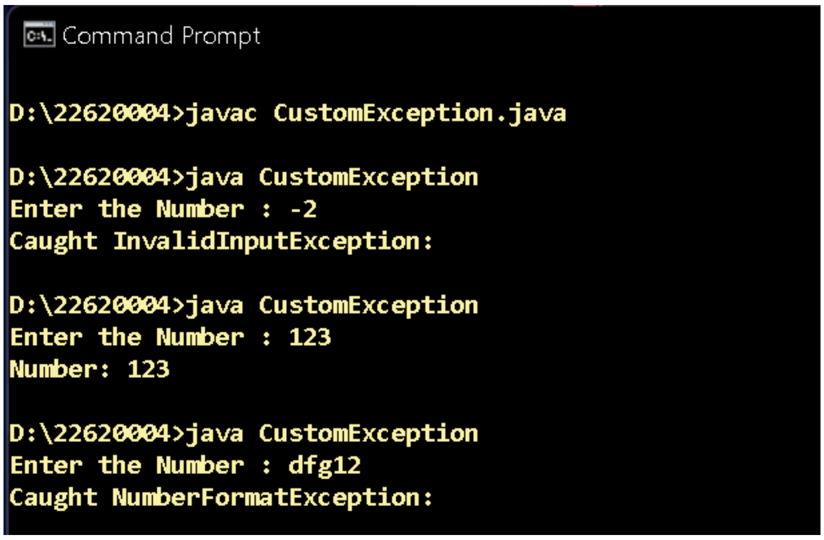
2. Write a program in java to handle custom exception with single try block and multiple catch block.

Program :



```java
import java.util.Scanner;
class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}
public class CustomException {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the Number : ");
        String d = sc.next();
        try {
            int num = Integer.parseInt(d);
            if (num < 0) {
                throw new InvalidInputException("Number must be non-negative.");
            }
            System.out.println("Number: " + num);
        } catch (InvalidInputException e) {
            System.out.println("Caught InvalidInputException: ");
        } catch (NumberFormatException e) {
            System.out.println("Caught NumberFormatException: ");
        } catch (Exception e) {
            System.out.println("Caught Exception: ");
        }
    }}
```

Output :



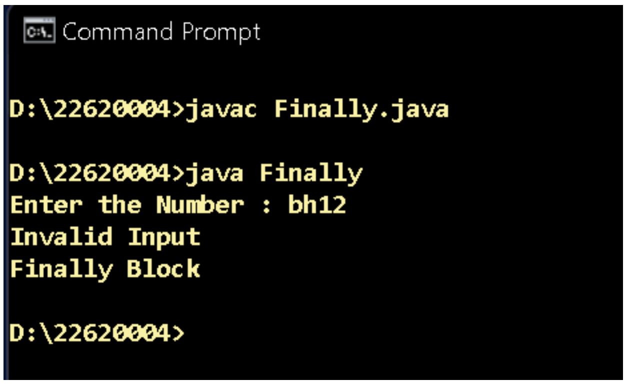
```
D:\22620004>javac CustomException.java

D:\22620004>java CustomException
Enter the Number : -2
Caught InvalidInputException:

D:\22620004>java CustomException
Enter the Number : 123
Number: 123

D:\22620004>java CustomException
Enter the Number : dfg12
Caught NumberFormatException:
```

3. Write a program in java to show the use of finally keyword.

Program :

```
import java.util.Scanner;

public class Finally{
public static void main(String args[]) {

Scanner sc = new Scanner(System.in);
System.out.print("Enter the Number : ");
String d = sc.next();
try {
    int num = Integer.parseInt(d);
    System.out.println("The number is: " + num);
} catch (NumberFormatException e) {
    System.out.println("Invalid Input");
}finally{
    System.out.println("Finally Block");
}
}
}
```

Output :



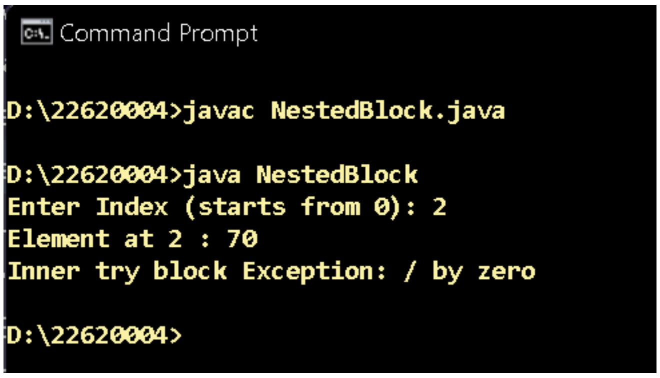
```
Command Prompt

D:\22620004>javac Finally.java

D:\22620004>java Finally
Enter the Number : bh12
Invalid Input
Finally Block

D:\22620004>
```

4. Write a program in java for handling exceptions with nested try block.

Program :

```
import java.util.Scanner;
public class NestedBlock{
    public static void main(String args[]){
        int[] arr = {10,30,70,90,22};
        int index;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter Index (starts from 0): ");
        index=sc.nextInt();
     try {
      System.out.println("Element at " +index+" : "+ arr[index]);
      try {
                int y = 5 / 0;
            } catch (ArithmeticException e) {
                System.out.println("Inner try block Exception: " +
e.getMessage());
            }

     } catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception :"+e);
     }
    }
}
```

Output :

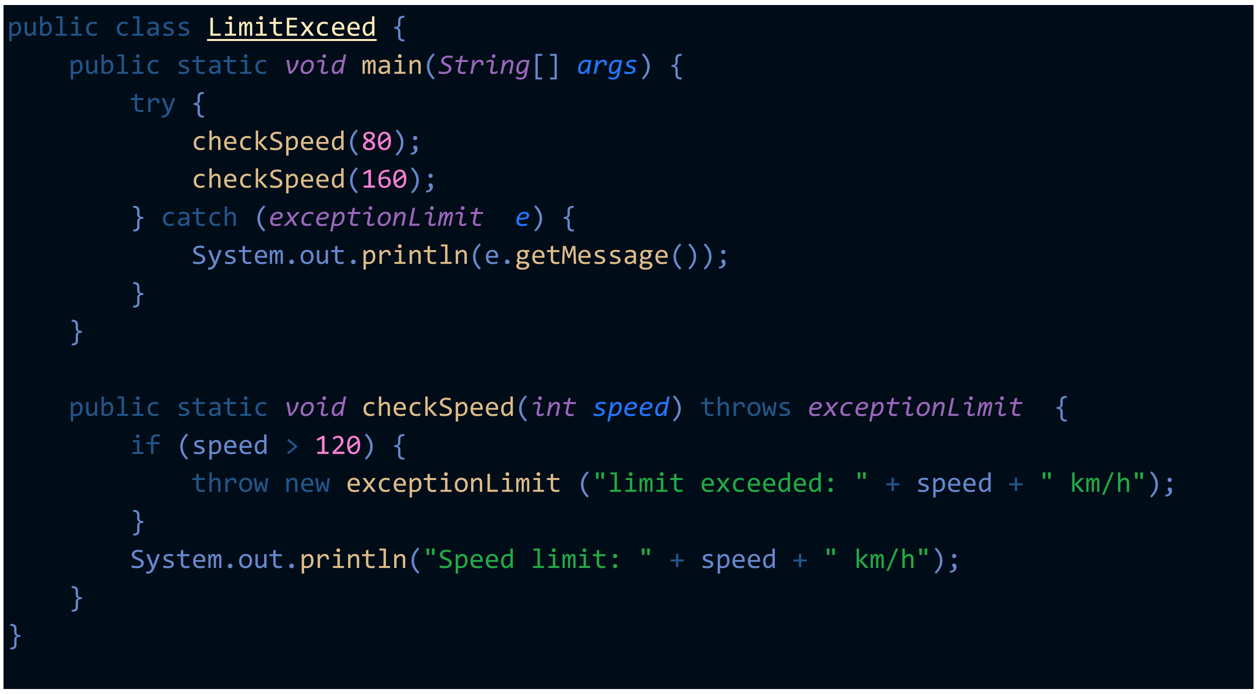


```
D:\22620004>javac NestedBlock.java

D:\22620004>java NestedBlock
Enter Index (starts from 0): 2
Element at 2 : 70
Inner try block Exception: / by zero

D:\22620004>
```

5. Write a program in java for custom exception to check speed of car on highway, if speed exceeds 120Km/hr then throw a 'Speed Limit Exceeded'exception. (use throw)

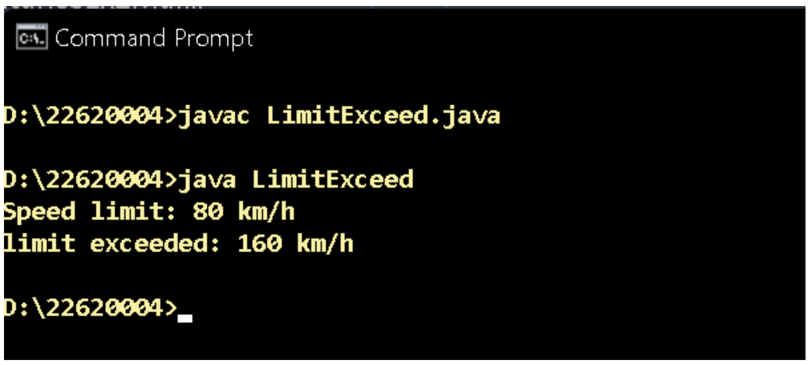
Program :

```
class exceptionLimit  extends Exception {
    public exceptionLimit (String msg) {
        super(msg);
    }
}
```



```java
public class LimitExceed {
    public static void main(String[] args) {
        try {
            checkSpeed(80);
            checkSpeed(160);
        } catch (exceptionLimit  e) {
            System.out.println(e.getMessage());
        }
    }

    public static void checkSpeed(int speed) throws exceptionLimit  {
        if (speed > 120) {
            throw new exceptionLimit ("limit exceeded: " + speed + " km/h");
        }
        System.out.println("Speed limit: " + speed + " km/h");
    }
}
```

Output :



```
D:\22620004>javac LimitExceed.java

D:\22620004>java LimitExceed
Speed limit: 80 km/h
limit exceeded: 160 km/h

D:\22620004>_
```

6. Differentiate in between throw and throws keyword.

throw	throws
1. throw is a keyword used to explicitly throw an exception from a method or a block of code.	1. throws is a keyword used to declare the types of exceptions that a method may throw.
2. Followed by an instance of an exception object that is being thrown.	2. followed by one or more exception types separated by commas.
3. It is used to indicate that an error has occurred during the execution of a method or a block of code	3. It is used to indicate the types of errors that may occur during the execution of a method.
4. It is used to propagate an exception up the call stack to the nearest	4. It is used to inform the caller of a method that the method may

catch block that can handle it	throw a certain type of exception, and the caller must either handle it or declare it in its own throws clause.
5. It is used within the body of a method or a block of code.	5. It is used in the method signature to declare the types of exceptions that the method may throw.

7. Explain exception handling mechanism.

Ans :

1. Exceptions are runtime errors that occur during the execution of a program, such as divide-by-zero, null pointer references, or file not found errors.
2. The exception handling mechanism in Java provides a way to gracefully handle these errors, preventing the program from crashing unexpectedly.
3. The try block is used to enclose a block of code that may throw an exception.
4. If an exception is thrown, it is caught by one or more catch blocks that are designed to handle that specific type of exception.
5. The finally block is executed regardless of whether an exception is thrown or not, and is used to release any resources held by the try block, such as files or network connections.
6. To handle checked exceptions, which are exceptions that must be caught or declared to be thrown by the calling method, the throws keyword is used to declare that the method may throw a specific type of exception.
7. Custom exceptions can be created by extending the Exception class or one of its subclasses, allowing programmers to define their own exception types that can be thrown and caught using the same exception handling mechanism.
8. Proper exception handling can improve the reliability and robustness of a program by allowing it to handle errors gracefully, provide informative error messages to users, and recover from errors without crashing.

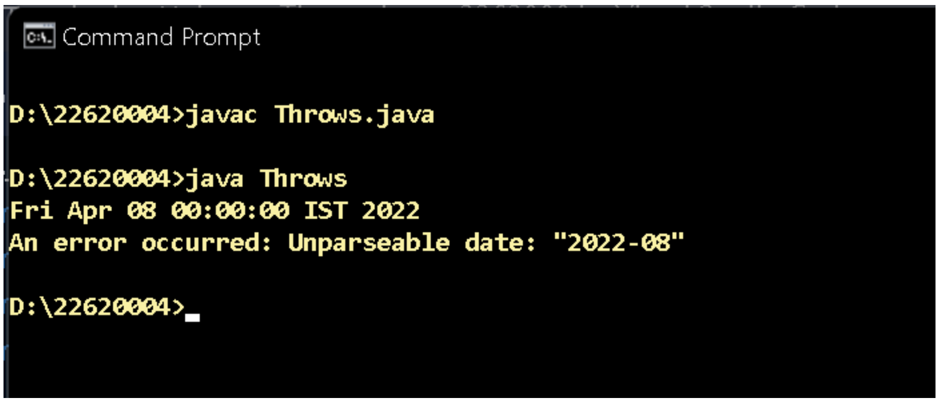
8. Write a program in java for handling checked exceptions using throws keyword.

Program :

```
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```java
public class Throws {
    public static void main(String[] args) throws ParseException {
        String dateString = "2022-04-08";
        String dateString2 = "2022-08";
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date date1= null;
        Date date2= null;
        try {
            date1 = dateFormat.parse(dateString);
            System.out.println(date1);
            date2 = dateFormat.parse(dateString2);
            System.out.println(date2);
        } catch (ParseException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

Output :



```
D:\22620004>javac Throws.java

D:\22620004>java Throws
Fri Apr 08 00:00:00 IST 2022
An error occurred: Unparseable date: "2022-08"

D:\22620004>_
```