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

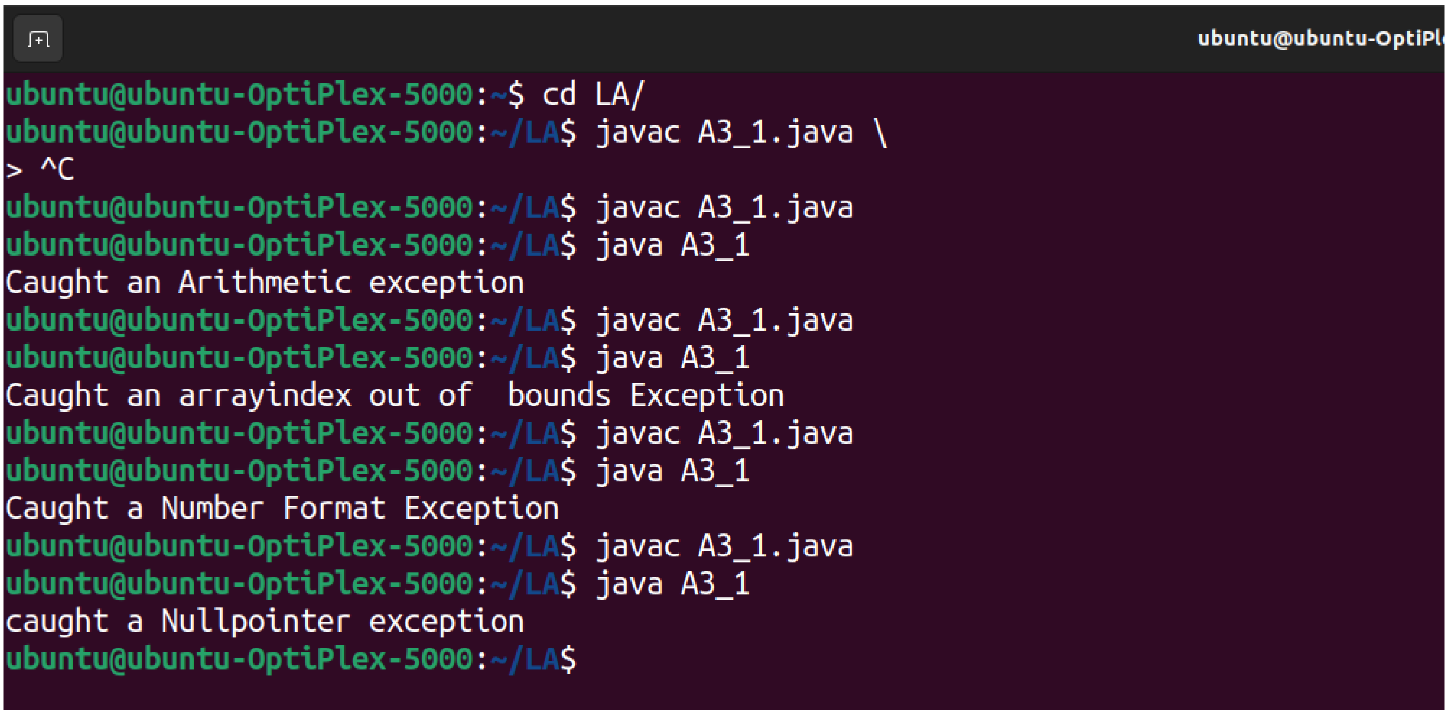
a. Divide by Zero

b. Array Index Out Of Bound

c. Number Format

d. Null Pointer

```
//22610027
public class A3_1 {


    public static void main(String[] args) {
        try {
            int i = 10/0;
            int[] arr = {1,2,3};
            int val = arr[5];


            String num = "hello";
            int numval = Integer.parseInt(num);


            String str = null;
            str.length();


        }catch(ArithmeticException e){
            System.out.println("Caught an Arithmetic exception");
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Caught an arrayindex out of  bounds Exception");


        }catch(NumberFormatException e){
            System.out.println("Caught a Number Format Exception");
        }catch(NullPointerException e){
            System.out.println("caught a Nullpointer exception");
        }
    }
}
```



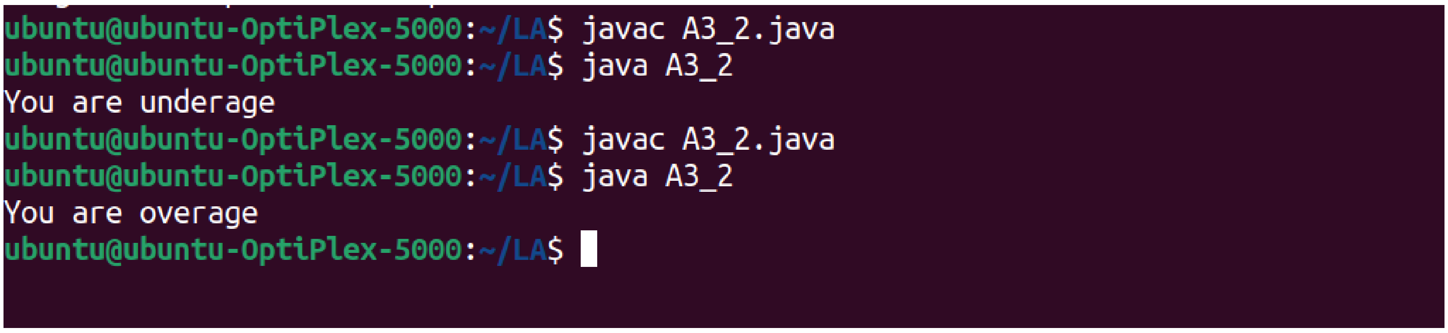
```
ubuntu@ubuntu-OptiPl
ubuntu@ubuntu-OptiPlex-5000:~$ cd LA/
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_1.java \
> ^C
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_1.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_1
Caught an Arithmetic exception
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_1.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_1
Caught an arrayindex out of  bounds Exception
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_1.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_1
Caught a Number Format Exception
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_1.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_1
caught a Nullpointer exception
ubuntu@ubuntu-OptiPlex-5000:~/LA$
```

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

//22610027

```
public class A3_2 {

        public static void main(String[] args) {
            try {
                int age = 65;
                checkAge(age);
            } catch (UnderAgeException e) {
```

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

    private static void checkAge(int age) throws UnderAgeException, OverAgeException {
        if (age < 18) {
            throw new UnderAgeException("You are underage");
        } else if (age > 60) {
            throw new OverAgeException("You are overage");
        } else {
            System.out.println("Your age is valid");
        }
    }
}

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

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

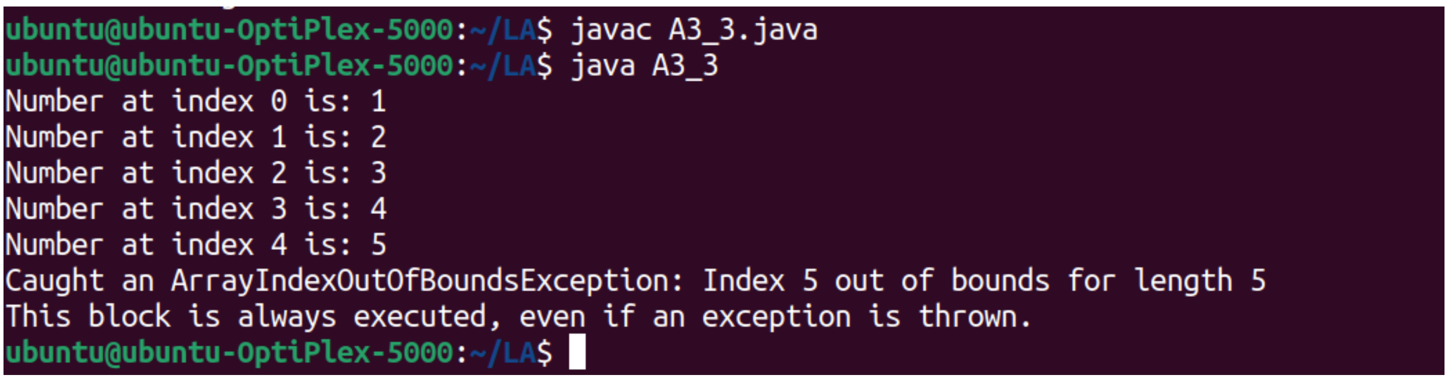


```
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_2.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_2
You are underage
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_2.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_2
You are overage
ubuntu@ubuntu-OptiPlex-5000:~/LA$
```

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

//22610027

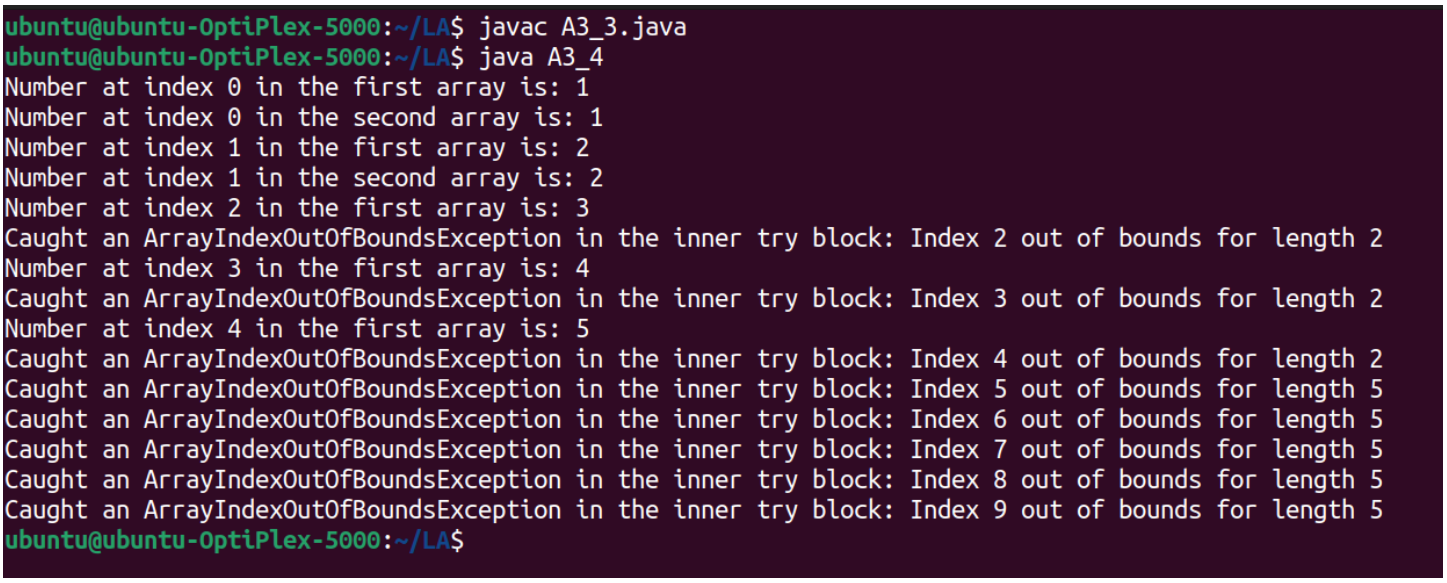
```
public class A3_3 {

    public static void main(String[] args) {

        int[] numbers = {1, 2, 3, 4, 5};

        try {

            for (int i = 0; i < 10; i++) {

                System.out.println("Number at index " + i + " is: " + numbers[i]);

            }

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Caught an ArrayIndexOutOfBoundsException: " + e.getMessage());

        } finally {

            System.out.println("This block is always executed, even if an exception is thrown.");

        }

    }

}
```


```
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_3.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_3
Number at index 0 is: 1
Number at index 1 is: 2
Number at index 2 is: 3
Number at index 3 is: 4
Number at index 4 is: 5
Caught an ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
This block is always executed, even if an exception is thrown.
ubuntu@ubuntu-OptiPlex-5000:~/LA$ 
```

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

//22610027

```
public class A3_4 {

    public static void main(String[] args) {

        int[] numbers = {1, 2, 3, 4, 5};

        int[] numbers2 = {1, 2};

        try {

            for (int i = 0; i < 10; i++) {

                try {

                    System.out.println("Number at index " + i + " in the first array is: " + numbers[i]);

                    System.out.println("Number at index " + i + " in the second array is: " + numbers2[i]);

                } catch (ArrayIndexOutOfBoundsException e) {

                    System.out.println("Caught an ArrayIndexOutOfBoundsException in the inner try block: "
+ e.getMessage());

                }

            }

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Caught an ArrayIndexOutOfBoundsException in the outer try block: " +
e.getMessage());

        }

    }

}
```

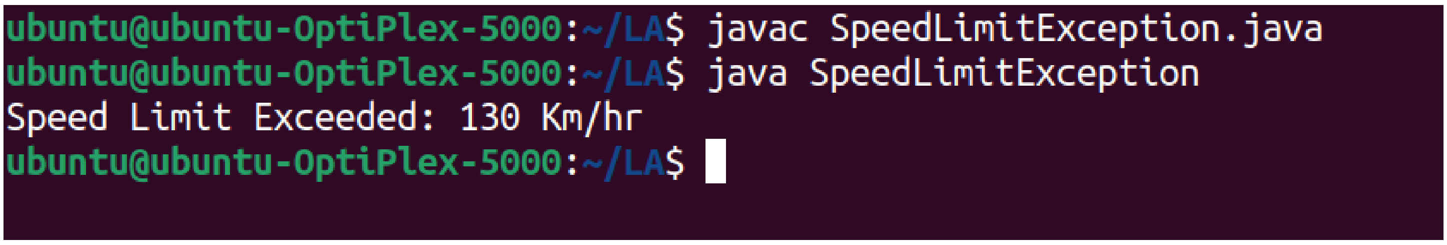


```
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A3_3.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A3_4
Number at index 0 in the first array is: 1
Number at index 0 in the second array is: 1
Number at index 1 in the first array is: 2
Number at index 1 in the second array is: 2
Number at index 2 in the first array is: 3
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 2 out of bounds for length 2
Number at index 3 in the first array is: 4
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 3 out of bounds for length 2
Number at index 4 in the first array is: 5
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 4 out of bounds for length 2
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 5 out of bounds for length 5
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 6 out of bounds for length 5
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 7 out of bounds for length 5
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 8 out of bounds for length 5
Caught an ArrayIndexOutOfBoundsException in the inner try block: Index 9 out of bounds for length 5
ubuntu@ubuntu-OptiPlex-5000:~/LA$
```

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)

//22610027

```
public class SpeedLimitException extends Exception {

    public SpeedLimitException(String message) {

        super(message);

    }



    public static void main(String[] args) {

        int speed = 130;

        try {

            checkSpeed(speed);

        } catch (SpeedLimitException e) {

            System.out.println(e.getMessage());

        }

    }



    private static void checkSpeed(int speed) throws SpeedLimitException {
        if (speed > 120) {
            throw new SpeedLimitException("Speed Limit Exceeded: " + speed + " Km/hr");
        } else {
            System.out.println("Speed is within the limit: " + speed + " Km/hr");
        }
    }
}
```



```
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac SpeedLimitException.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java SpeedLimitException
Speed Limit Exceeded: 130 Km/hr
ubuntu@ubuntu-OptiPlex-5000:~/LA$ 
```

6]Differentiate in between throw and throws keyword.

//22610027

In Java, throw and throws are both used for exception handling, but they serve different purposes.

The throw keyword is used to explicitly throw an exception within a method or a block of code. When an exception is thrown, the normal flow of the program is interrupted and the exception is propagated up the call stack until it is handled by a catch block or reaches the top-level Thread object, which will print the stack trace to the console

The throws keyword, on the other hand, is used to declare that a method may throw one or more exceptions. It is used in the method signature and indicates that the method may propagate an exception to its caller. The throws keyword does not actually throw an exception; it only declares that an exception may be thrown.

7]Explain exception handling mechanism.

//22610027

Exception handling is a mechanism in Java that allows a program to handle unexpected conditions that may occur during the execution of a program. These unexpected conditions are called exceptions, and they can be caused by various factors such as user input errors, network failures, hardware failures, or programming errors.

The Java exception handling mechanism consists of the following components:

1. **Exceptions**: An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. Exceptions can be of two types: checked exceptions and unchecked exceptions. Checked exceptions are exceptions that are checked by the compiler at compile-time, while unchecked exceptions are exceptions that are not checked by the compiler and can occur at runtime.
2. **Throwable class**: The Throwable class is the superclass of all errors and exceptions in Java. It has two subclasses: Error and Exception. The Error class represents errors that are outside the control of the application, such as OutOfMemoryError or StackOverflowError. The Exception class represents exceptions that can be handled by the application, such as IOException or NullPointerException.
3. **try-catch block**: The try-catch block is used to handle exceptions in Java. The try block contains the code that may throw an exception, and the catch block contains the code that handles the exception. The catch block must immediately follow the try block and can handle one or more exceptions.
4. **finally block**: The finally block contains the code that is always executed, whether an exception is thrown or not. It is used to clean up resources, such as closing file handles or network connections.
5. **throw keyword**: The throw keyword is used to explicitly throw an exception within a method or a block of code.
6. **throws keyword**: The throws keyword is used to declare that a method may throw one or more exceptions. It is used in the method signature and indicates that the method may propagate an exception to its caller.

The Java exception handling mechanism allows a program to handle unexpected conditions gracefully and provide meaningful error messages to the user. It also allows the program to continue executing after the exception has been handled, rather than crashing or producing incorrect results.

When an exception is thrown, the Java runtime looks for the nearest catch block that can handle the exception. If no such catch block is found, the exception is propagated up the call stack until it is handled or reaches the top-level Thread object, which will print the stack trace to the console.

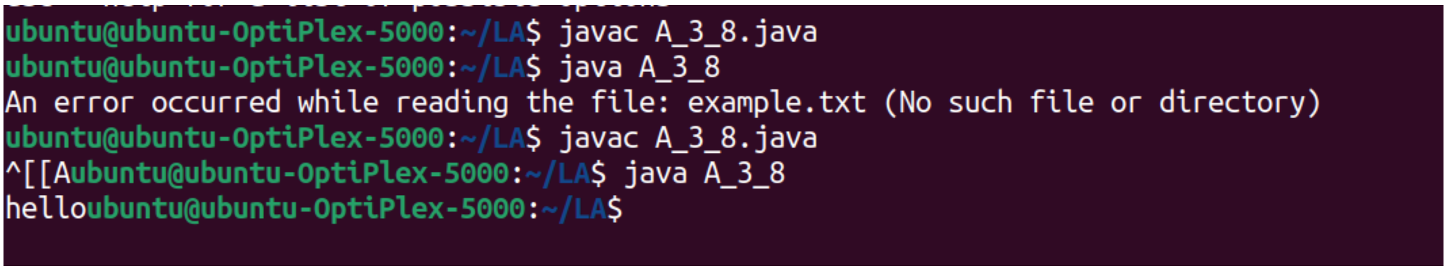
In summary, the Java exception handling mechanism is a powerful feature that allows a program to handle unexpected conditions, clean up resources, and provide meaningful error messages to the user.

```
    }
```

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

//22610027

```
import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;


public class A_3_8 {

    public static void main(String[] args) {

        try {

            readFile("example.txt");

        } catch (IOException e) {

            System.out.println("An error occurred while reading the file: " + e.getMessage());

        }

    }


    private static void readFile(String fileName) throws FileNotFoundException, IOException {

        FileInputStream inputStream = new FileInputStream(fileName);

        int data;

        while ((data = inputStream.read()) != -1) {

            System.out.print((char) data);

        }

        inputStream.close();

    }

}
```



```
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A_3_8.java
ubuntu@ubuntu-OptiPlex-5000:~/LA$ java A_3_8
An error occurred while reading the file: example.txt (No such file or directory)
ubuntu@ubuntu-OptiPlex-5000:~/LA$ javac A_3_8.java
^[[Aubuntu@ubuntu-OptiPlex-5000:~/LA$ java A_3_8
helloubuntu@ubuntu-OptiPlex-5000:~/LA$
```