Java Assignment -22610018-Vighnesh Potdar -S2Batch-

Q1. 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

ANS:

```java
 * main
 */
public class main {
    Run | Debug
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Caught Divide by Zero Exception: " + e.getMessage());
        }

        // Array Index Out Of Bound
        try {
            int[] arr = new int[5];
            int value = arr[10];
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught Array Index Out Of Bound Exception: " + e.
            getMessage());
        }

        // Number Format
        try {
            int number = Integer.parseInt("abc");
        } catch (NumberFormatException e) {
            System.out.println("Caught Number Format Exception: " + e.getMessage());
        }

        // Null Pointer
        try {
            String str = null;
            int length = str.length();
        } catch (NullPointerException e) {
            System.out.println("Caught Null Pointer Exception: " + e.getMessage());
        }

    }
```

```
9adf/redhat.java/jdt_ws/Java_c55baaa3/bin Assignments03.Q1.main
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Caught Divide by Zero Exception: / by zero
Caught Array Index Out Of Bound Exception: Index 10 out of bounds for length 5
Caught Number Format Exception: For input string: "abc"
Caught Null Pointer Exception: Cannot invoke "String.length()" because "str" is null
```

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

and multiple catch block.

```java
package Assignments03.Q2;

public class main {

    public static class CustomException extends Exception {
        public CustomException(String message) {
            super(message);
        }
    }
    Run | Debug
    public static void main(String[] args) {
        try {
            checkDivision();
        } catch (CustomException e) {
            System.out.println("Caught CustomException: " + e.getMessage());
        }
    }

    public static void checkDivision() throws CustomException {
        try {
            int result = 10 / 0; // This line will throw ArithmeticException
        } catch (ArithmeticException e) {
            throw new CustomException(message:"Division by zero occurred.");
        }
    }
}
```

```
3.Q2.main
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Caught CustomException: Division by zero occurred.
```

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

```java
package Assignments03.Q3;

public class main {
    Run | Debug
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Caught ArithmeticException: " + e.getMessage());
        } finally {
            System.out.println("Finally block executed.");
        }
    }
}
```

```
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Caught ArithmeticException: / by zero
Finally block executed.
```

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

```java
package Assignments03.Q4;

public class msin {

    Run | Debug
    public static void main(String[] args) {
        try {
            try {
                int result = 10 / 0;
            } catch (ArithmeticException e) {
                System.out.println("Inner Try Block: Caught ArithmeticException: " + e
            }
            int[] arr = new int[5];
            try {
                int value = arr[10];
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Outer Try Block: Caught ArrayIndexOutOfBoundsExcep
            }
        } catch (Exception e) {
            System.out.println("Caught Exception: " + e.getMessage());
        }
    }
}
```

```
9adf/redhat.java/jdt_ws/Java_c55baaa3/bin Assignments03.Q4.msin
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Inner Try Block: Caught ArithmeticException: / by zero
Outer Try Block: Caught ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5
```

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)

```java
package Assignments03.Q5;

public class main { // Corrected class name to start with an uppercase letter

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

    Run | Debug
    public static void main(String[] args) {
        int carSpeed = 130; // Example car speed

        try {
            if (carSpeed > 120) {
                throw new SpeedLimitExceededException("Speed Limit Exceeded! Speed: "
            } else {
                System.out.println("Car speed within limit.");
            }
        } catch (SpeedLimitExceededException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
3.Q5.main
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Speed Limit Exceeded! Speed: 130Km/hr
```

6. Differentiate in between throw and throws keyword.

`throw` keyword:

- The `throw` keyword is used to explicitly throw an exception.
- It is used within a method to throw an exception manually when a certain condition is met.
- It is followed by an instance of the `Throwable` class (or any of its subclasses), which represents the exception being thrown.

`throws` keyword:

- The `throws` keyword is used in method signatures to declare that a method may throw certain types of exceptions.
- It is used to delegate the responsibility of handling exceptions to the calling method or to propagate the exception up the call stack.
- Multiple exceptions can be declared using a comma-separated list.
- Example:

7. Explain exception handling mechanism.

ANS:

Exception handling is a mechanism in programming languages, including Java, that allows developers to gracefully handle unexpected or exceptional situations that occur during the execution of a program. These exceptional situations, known as exceptions, can include errors like division by zero, file not found, network connection failure, and many others.

1. **Exceptions**: Represent unexpected events or errors during program execution.
2. **Hierarchy**: Exceptions in Java are organized in a hierarchy, rooted at the `Throwable` class.
3. **Types**:
    - Checked Exceptions: Must be caught or declared using `throws` keyword.
    - Unchecked Exceptions (Runtime Exceptions): Not required to be caught or declared.
4. **try-catch Blocks**:
    - `try`: Contains code that may throw an exception.
    - `catch`: Handles exceptions thrown within the `try` block.
    - Multiple catch blocks can handle different types of exceptions.
5. **finally Block**: Optional block executed after `try-catch`, used for cleanup operations.
6. **Throwing Exceptions**:
    - `throw` keyword: Manually throws an exception.
    - Used to create and throw custom exceptions.
7. **Checked vs. Unchecked**:
    - Checked exceptions require handling or declaration.
    - Unchecked exceptions do not, but it's recommended to handle them for robustness.

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

```java
package Assignments03.Q6;

import java.io.FileNotFoundException;
import java.io.FileReader;

public class main {
    Run | Debug
    public static void main(String[] args) {
        try {
            readFile(filename:"example.txt");
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage()
        }
    }

    // Method that may throw a checked exception
    public static void readFile(String filename) throws FileNotFou
        FileReader fileReader = new FileReader(filename);
        // Other file reading operations...
        System.out.println("File " + filename + " successfully rea
    }
}
```

```
7f1fb10621092449adf/redhat.java/jdt_ws/Java_c55baaa3/bin Assignments0
3.Q6.main
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aat
ext=true
File not found: example.txt (No such file or directory)
```

```
└─$ java main.java
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aat
ext=true
File example.txt successfully read.
```