**TASK-11**

1. **What are the four access modifiers available in java and what is their significance in term of class, method, and variable accessibility?**

1) **Public:** The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

2) **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

3) **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

4) **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

Similarly a default method or variable is also accessible inside the package in which they are defined and not outside the package.

1. **What is the difference between Exception and Error?**

In Java, errors and exceptions are both types of throw able objects, but they represent different types of problems that can occur during the execution of a program.

* An exception is an event that occurs during program execution, interrupting the normal flow of instructions.
* Exceptions can happen at compile time or run time & Error can happen at compile time.
* Exceptions are typically caused by issues in the code written by developers.
* Exceptions can be recovered using mechanisms like try-catch blocks or by declaring them with the throws keyword.
* ExceptionincludesInputMismatchException, NullPointerException, and ArrayIndexOutOfBoundsException.
* There are two types of exceptions:
  + Checked exceptions: These must be handled explicitly by the programmer (SQL, IO Exceptions).
  + Unchecked exceptions: These do not require explicit handling.
* Errors are usually caused by serious problems that are outside the control of the program, such as running out of memory or a system crash.
* An exception is an event that occurs during program execution, interrupting the normal flow of instructions.
* Exceptions can happen at compile time or run time & Error can happen at compile time.
* Exceptions are typically caused by issues in the code written by developers.
* Exceptions can be recovered using mechanisms like try-catch blocks or by declaring them with the throws keyword.
* ExceptionincludesInputMismatchException, NullPointerException, and ArrayIndexOutOfBoundsException.
* There are two types of exceptions:
  + Checked exceptions: These must be handled explicitly by the programmer (SQL, IO Exceptions).
  + Unchecked exceptions: These do not require explicit handling.
* Errors are usually caused by serious problems that are outside the control of the program, such as running out of memory or a system crash.
* In java, both Errors and Exceptions are the subclasses of java.lang.Throwable class.
* Errors are irrecoverable and cannot be caught or handled.
* Errors are always unchecked & Exceptions Include both Checked and unchecked.
* Error include OutOfMemoryError, StackOverflowError, and LinkageError.
* Errors are mostly caused by the environment in which program is running while in Exceptions Program itself is responsible for causing exceptions.

1. **What is the difference between checked exception and unchecked exception?**

**Checked Exception:**

1. Checked exceptions happen at compile time when the source code is

Transformed into an executable code.

1. The checked exception is checked by the compiler.
2. Checked exceptions can be created manually
3. This exception is counted as a sub-class of the class.
4. Java Virtual Machine requires the exception to be caught or handled.

**Examples of Checked exceptions:**

* File Not Found Exception
* No Such Field Exception
* Interrupted Exception
* No Such Method Exception
* Class Not Found Exception

**Unchecked Exception:**

1. Unchecked exceptions happen at runtime when the executable program starts running.
2. These types of exceptions are not checked by the compiler.
3. They can also be created manually
4. This exception happens in runtime, and hence it is not included in the exception class.
5. Java Virtual Machine does not need the exception to be caught or handled.

**Examples of Unchecked Exceptions:**

* No Such Element Exception
* Undeclared Throwable Exception
* Empty Stack Exception
* Arithmetic Exception
* Null Pointer Exception
* Array Index Out of Bounds Exception
* Security Exception

1. **Write a java program that reads user input for two integer and performs division. Handle the exception that is thrown when the second number is zero, and display an error message to user?**

package task11;

public class HandleDivideByZero {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

int numerator=2;

int denominator=0;

try {

int result=numerator/denominator;

System.***out***.println("Result"+result);

}

catch (ArithmeticException e) {

System.***out***.println("Error:"+e.getMessage());

System.***out***.println("Cannot divide by zero");

}

}

}

1. **Write a code for ArrayIndexOutOfBoundException and StringIndexOutOfBoundException**?

**ArrayIndexOutOfBoundException:**

package task11;

import java.util.Arrays;

import java.util.\*;

public class AIOBException {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

int[] myArray = { 1234, 1589, 4687, 1547, 6556, 5445, 7896 };

System.***out***.println("Elements in the array are:");

System.***out***.println(Arrays.*toString*(myArray));

Scanner sc = new Scanner(System.***in***);

System.***out***.println("Enter the index of the required element:");

int element = sc.nextInt();

System.***out***.println("Element in the given index is ::" + myArray[element]);

}

}

Elements in the array are:

[1234, 1589, 4687, 1547, 6556, 5445, 7896]

Enter the index of the required element:

7

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 7 out of bounds for length 7

at task11.AIOBException.main(AIOBException.java:18)

**StringIndexOutOfBoundException:**

package task11;

public class Sample {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

String str = "Hi how are you";

System.***out***.println("Length of the String: "+str.length());

for(int i=0; i<str.length(); i++) {

System.***out***.println(str.charAt(i));

}

System.***out***.println(str.charAt(40));

}

}

1. **You are building a login system for a website using Java .If the user enters an incorrect password, you want to display a message informing them of the error. How would you use exception handling to handle this situation?**
2. package task11;
3. public class Login {
4. String correctPassword;
5. public Login(String xyz123) {
6. this.correctPassword = correctPassword;
7. }
8. public void loginFn(String inCorrectPw) throws Exception {
9. if (!inCorrectPw.equals(correctPassword)) {
10. throw new Exception("Incorrect Password please try again");
11. } else {
12. System.***out***.println("Login successful!");
13. }
14. }
15. public static void main(String[] args) {
17. Login obj = new Login("correct\_password");
18. try {
19. obj.loginFn("Wrong Password");
20. } catch (Exception e) {
21. e.printStackTrace();
22. }
23. }
24. }

**Output:**

java.lang.Exception: Incorrect Password please try again

at task11.Login.loginFn(Login.java:17)

at task11.Login.main(Login.java:32)

**7.** **Create a custom exception in Java called “ InvalidAgeException” that is thrown when the user enter an age less than 18. Implement exception handling in a Java program to catch the “ InvalidAgeException” and display an error message.**

package task11;

public class AgeValidation {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

InvalidAgeException obj = new InvalidAgeException("Exception for age");

obj.ageValidation(15);

}

}

package task11;

public class InvalidAgeException extends Exception {

public InvalidAgeException(String str) {

super(str);

}

public void ageValidation(int age) {

try {

if (age < 18) {

throw new InvalidAgeException("Your age is less than 18, so you are not eligible to vote");

}

else {

System.***out***.println("You age is " + age + "so you are not eligible to vote");

}

} catch (InvalidAgeException e) {

e.printStackTrace();

}

}

}

task11.InvalidAgeException: Your age is less than 18, so you are not eligible to vote

at task11.InvalidAgeException.ageValidation(InvalidAgeException.java:15)

at task11.AgeValidation.main(AgeValidation.java:9)

**8. Implement exception handling in a Java program that reads data from a file. If the file does not exist, throw a “ FileNotFoundException” and display an error message to the user.**

package task11;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.Properties;

public class FileException {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

FileException obj = new FileException();

obj.readData();

}

public void readData() {

String value = null;

Properties prop = new Properties();

try {

FileInputStream ip = new FileInputStream("C:\\Users\\Subasri Suresh\\OneDrive\\Desktop\\JavaFile.txt");

prop.load(ip);

} catch (FileNotFoundException e) {

System.***out***.println("File does not exits");

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

value = prop.getProperty("name");

System.***out***.println(value);

}

}

**Output:**

File does not exits

java.io.FileNotFoundException: C:\Users\Subasri Suresh\OneDrive\Desktop\JavatestFile.txt (The system cannot find the file specified)

at java.base/java.io.FileInputStream.open0(Native Method)

at java.base/java.io.FileInputStream.open(FileInputStream.java:213)

at java.base/java.io.FileInputStream.<init>(FileInputStream.java:152)

at java.base/java.io.FileInputStream.<init>(FileInputStream.java:106)

at task11.FileException.readData(FileException.java:23)

at task11.FileException.main(FileException.java:14)

null