1.Write a Java Program to find Math.pow(n,p). - If either n or p is negative, then the method must throw an exception which says "n or p should not be negative.". - If both n and p are zero, then the method must throw an exception which says "n and p should not be zero."

import java.util.Scanner;

// Class to compute power with exception handling

class PowerCalculator {

// Method to compute power with exception handling

public static double power(int n, int p) throws Exception {

if (n < 0 || p < 0) {

throw new Exception("n or p should not be negative.");

}

if (n == 0 && p == 0) {

throw new Exception("n and p should not be zero.");

}

return Math.pow(n, p);

}

}

// Main class to test power calculation

public class MathPowerDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter base (n): ");

int n = scanner.nextInt();

System.out.print("Enter exponent (p): ");

int p = scanner.nextInt();

try {

double result = PowerCalculator.power(n, p);

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

} catch (Exception e) {

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

}

scanner.close();

}

}

2. Write a Java Program that reads two integers by passing arguments to main() and find sum of squares of two integers. Handle ArrayIndexOutOfBoundsException. (result=x2+y2)

public class SumOfSquares {

public static void main(String[] args) {

try {

// Read integers from command line arguments

int x = Integer.parseInt(args[0]);

int y = Integer.parseInt(args[1]);

// Calculate sum of squares

int result = (x \* x) + (y \* y);

// Print the result

System.out.println("Sum of squares: " + result);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Please provide two integer arguments.");

} catch (NumberFormatException e) {

System.out.println("Error: Please enter valid integers.");

}

}

}

**How to Run the Program**

1. **Compile the program**:

javac SumOfSquares.java

1. **Run the program with two integer arguments**:

java SumOfSquares 3 4

**Output:**

Sum of squares: 25

1. **Handling missing arguments**:

java SumOfSquares 5

**Output:**

Error: Please provide two integer arguments.

1. **Handling non-integer input**:

java SumOfSquares 5 a

**Output:**

Error: Please enter valid integers.

3. Write a Java Program to count number of invalid entries. Example: java example 5 2 a 1 b => Invalid = 2

public class CountInvalidEntries {

public static void main(String[] args) {

int invalidCount = 0;

for (String arg : args) {

try {

Integer.parseInt(arg); // Attempt to parse as integer

} catch (NumberFormatException e) {

invalidCount++; // Increment count if parsing fails

}

}

System.out.println("Invalid entries: " + invalidCount);

}

}

**How to Run the Program**

1. **Compile the program**:

javac CountInvalidEntries.java

1. **Run the program with mixed inputs**:

java CountInvalidEntries 5 2 a 1 b

**Output:**

Invalid entries: 2

4.Write a Java Program to demonstrate try with multiple catch block. (ArithmeticException,NullPointerException, ArrayIndexOutOfBoundsException)

public class MultipleCatchDemo {

public static void main(String[] args) {

try {

// Example of ArithmeticException

int a = 10, b = 0;

int result = a / b; // Division by zero

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

// Example of NullPointerException

String str = null;

System.out.println(str.length()); // Accessing length of null

// Example of ArrayIndexOutOfBoundsException

int[] arr = {1, 2, 3};

System.out.println(arr[5]); // Accessing out-of-bound index

} catch (ArithmeticException e) {

System.out.println("Error: Division by zero is not allowed.");

} catch (NullPointerException e) {

System.out.println("Error: Null reference encountered.");

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Array index is out of bounds.");

}

System.out.println("Program continues after exception handling.");

}

}

**How it Works**

1. **ArithmeticException**: Occurs when dividing by zero.
2. **NullPointerException**: Occurs when trying to access a method on a null reference.
3. **ArrayIndexOutOfBoundsException**: Occurs when accessing an array index that does not exist.

**Expected Output**

Error: Division by zero is not allowed.

Program continues after exception handling.

(Only the first exception is caught because after an exception occurs, control jumps to the catch block.)

5. Write a program to perform integer divisions. The user enters two numbers, Num1 and Num2. The division of Num1 and Num2 is displayed. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException. Display the exception.

import java.util.Scanner;

public class IntegerDivision {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

// Taking user input

System.out.print("Enter first number (Num1): ");

int num1 = Integer.parseInt(scanner.next());

System.out.print("Enter second number (Num2): ");

int num2 = Integer.parseInt(scanner.next());

// Performing division

int result = num1 / num2;

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

} catch (NumberFormatException e) {

System.out.println("Error: Please enter valid integers.");

} catch (ArithmeticException e) {

System.out.println("Error: Division by zero is not allowed.");

} finally {

scanner.close();

}

}

}

### ****How It Works****

1. **Reads two numbers (Num1 and Num2)** from the user.
2. **Parses them as integers**. If they are not valid integers, it throws a NumberFormatException.
3. **Performs division (Num1 / Num2)**. If Num2 is 0, it throws an ArithmeticException.
4. **Catches and displays exceptions** when invalid input or division by zero occurs.

### ****Example Runs****

#### ****Valid Input****

Enter first number (Num1): 10

Enter second number (Num2): 2

Result: 5

#### ****Invalid Input (Non-Integer)****

Enter first number (Num1): ten

Error: Please enter valid integers.

#### ****Division by Zero****

Enter first number (Num1): 5

Enter second number (Num2): 0

Error: Division by zero is not allowed.

6. Write a Java Program to demonstrate **NegativeArraySizeException.**

import java.util.Scanner;

public class NegativeArraySizeDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter array size: ");

int size = scanner.nextInt();

// Attempting to create an array with a negative size

int[] arr = new int[size];

System.out.println("Array created successfully with size " + size);

} catch (NegativeArraySizeException e) {

System.out.println("Error: Array size cannot be negative.");

} finally {

scanner.close();

}

}

}

### ****How It Works****

1. **User enters an integer** for the array size.
2. **If the size is negative**, a NegativeArraySizeException is thrown.
3. **Catches the exception** and displays an error message.
4. **Ensures scanner is closed** in the finally block.

### ****Example Runs****

#### ****Valid Input****

Enter array size: 5

Array created successfully with size 5

#### ****Negative Input****

Enter array size: -3

Error: Array size cannot be negative.

7. Write a Java Program to demonstrate **NullPointerException**

public class NullPointerExceptionDemo {

public static void main(String[] args) {

try {

String str = null; // Initializing string with null

System.out.println(str.length()); // This will throw NullPointerException

} catch (NullPointerException e) {

System.out.println("Error: Attempted to access a null reference.");

}

}

}

**How It Works**

1. A String variable is set to null.
2. Attempting to access its length() method results in a NullPointerException.
3. The exception is caught and an appropriate message is displayed.

**Example Output**

Error: Attempted to access a null reference.

8.Write a Java Program to demonstrate **StringIndexOutOfBoundsException**

public class StringIndexOutOfBoundsDemo {

public static void main(String[] args) {

try {

String str = "Hello";

// Attempting to access an index that is out of bounds

char ch = str.charAt(10);

System.out.println("Character at index 10: " + ch);

} catch (StringIndexOutOfBoundsException e) {

System.out.println("Error: String index is out of bounds.");

}

}

}

**How It Works**

1. A String variable is initialized with "Hello".
2. The program attempts to access the character at index 10, which is out of bounds.
3. This results in a StringIndexOutOfBoundsException, which is caught and handled gracefully.

**Example Output**

Error: String index is out of bounds.

9.Write a Java Program to demonstrate **IOException**

import java.io.\*;

public class IOExceptionDemo {

public static void main(String[] args) {

try {

// Attempting to read from a non-existent file

BufferedReader reader = new BufferedReader(new FileReader("nonexistentfile.txt"));

String line = reader.readLine();

System.out.println("File content: " + line);

reader.close();

} catch (IOException e) {

System.out.println("Error: An I/O exception occurred.");

}

}

}

**How It Works**

1. The program tries to open a file named "nonexistentfile.txt".
2. If the file does not exist, an IOException is thrown.
3. The exception is caught, and an error message is displayed.

**Example Output**

Error: An I/O exception occurred.

10. Write a Java Program to demonstrate **FileNotFoundException**

import java.io.\*;

public class FileNotFoundExceptionDemo {

public static void main(String[] args) {

try {

// Attempting to open a file that does not exist

FileReader file = new FileReader("nonexistentfile.txt");

BufferedReader reader = new BufferedReader(file);

System.out.println("File opened successfully.");

reader.close();

} catch (FileNotFoundException e) {

System.out.println("Error: File not found.");

}

}

}

**How It Works**

1. The program attempts to open a file named "nonexistentfile.txt".
2. If the file does not exist, a FileNotFoundException is thrown.
3. The exception is caught, and an error message is displayed.

**Example Output**

Error: File not found.