**ASSIGNMENT-15**

**Question-1**

**Problem Statement:** Input two numbers as numerator and denominator for division. Write a program to show an ArithmeticException if the division is not possible when denominator is 0.

**Source Code**

import java.util.Scanner;

public class Division {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the numerator: ");

int numerator = sc.nextInt();

System.out.print("Enter the denominator: ");

int denominator = sc.nextInt();

try {

int result = numerator / denominator;

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

} catch (ArithmeticException e) {

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

}

}

}

**OUTPUT:**

Enter the numerator: 17

Enter the denominator: 0

Error: Division by zero is not allowed.

**Question-2**

**Problem Statement:** Define an array of size n and set some values to it. Show an ArrayIndexOutOfBoundException when trying to access the index that is more than size of the array.

**Source Code**

import java.util.Scanner;

public class ExceptionHandler {

public static void main(String[] args) {

//int[] array = new int[10];

int array[]={1,2,3,4,5,6,7,8,9,10};

try {

// This line of code will throw an ArrayIndexOutOfBoundsException if

// the specified index is greater than or equal to the length of the array.

int value = array[10];

} catch (ArrayIndexOutOfBoundsException e) {

System.err.println("Error: array index out of bounds");

System.err.println("Cause: " + e.getMessage());

}

}

}

**OUTPUT:**

Error: array index out of bounds

Cause: Index 10 out of bounds for length 10

**Question-3**

**Problem Statement:** Write a program to show the use of NullPointerException.

**Source Code**

public class NullPointerTest{

public static void main(String[] args) {

String str = null;

try {

// Trying to get the length of a null string

int length = str.length();

System.out.println("Length of the string: " + length);

} catch (NullPointerException e) {

System.out.println("NullPointerException caught: ");

}

}

}

**OUTPUT:**

NullPointerException caught.

**Question-4**

**Problem Statement:** Define Exceptions VowelException, BlankException, ExitException to restrict the input of vowel, space and ‘X’. Write another class TestException which reads a character from command line. If it is a vowel, throw VowelException, if it is a blank space throw BlankException and for a character ‘X’ throw an ExitException and terminate the program. For any other character, display “Valid character”.

**Source Code**

//Class VowelException

public class VowelException extends Exception {

public VowelException(String message) {

super(message);

}

}

//Class BlankException

public class BlankException extends Exception {

public BlankException(String message) {

super(message);

}

}

//Class ExitException

public class ExitException extends Exception {

public ExitException(String message) {

super(message);

}

}

//Class TestException(with main method)

public class TestException {

public static void main(String[] args) {

char input = args[0].charAt(0);

try {

if (input == 'X') {

throw new ExitException("Exit requested");

} else if (input == ' ') {

throw new BlankException("Blank space found");

} else if ("AEIOUaeiou".indexOf(input) != -1) {

throw new VowelException("Vowel found");

} else {

System.out.println("Valid character");

}

} catch (VowelException e) {

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

} catch (BlankException e) {

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

} catch (ExitException e) {

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

// Terminate the program

System.exit(0);

}

}

}

**OUTPUT:**

>>java TestException E

VowelException caught: Vowel found

**Question-5**

**Problem Statement:** Write a program which accepts two integers and an arithmetic operator from the command line and performs the operation. Check the following user defined exceptions:

i. If the number of arguments are less than 3 then throw “FewArgumentsException”.

ii. If the operator is not an Arithmetic operator, throw “InvalidOperatorException”.

iii. If result is –ve, then throw “NegativeResult” exception.

**Source Code**

//Class FewArgumentsException

public class FewArgumentsException extends Exception {

public FewArgumentsException(String message) {

super(message);

}

}

//Class InvalidOperatorException

public class InvalidOperatorException extends Exception {

public InvalidOperatorException(String message) {

super(message);

}

}

//Class NegativeResultException

public class NegativeResultException extends Exception {

public NegativeResultException(String message) {

super(message);

}

}

//Class Calculator(with main method)

public class Calculator {

public static void main(String[] args) {

try {

if (args.length < 3) {

throw new FewArgumentsException("Few arguments provided. Expected format: [operand1] [operator] [operand2]");

}

int operand1 = Integer.parseInt(args[0]);

String operator = args[1];

int operand2 = Integer.parseInt(args[2]);

int result = 0;

switch (operator) {

case "+":

result = operand1 + operand2;

break;

case "-":

result = operand1 - operand2;

break;

case "\*":

result = operand1 \* operand2;

break;

case "/":

if (operand2 == 0) {

throw new ArithmeticException("Division by zero");

}

result = operand1 / operand2;

break;

case "%":

result = operand1 % operand2;

break;

default:

throw new InvalidOperatorException("Invalid operator. Please use one of: +, -, \*, /, %");

}

if (result < 0) {

throw new NegativeResultException("Result is negative: " + result);

}

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

} catch (NumberFormatException e) {

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

} catch (FewArgumentsException | InvalidOperatorException | NegativeResultException e) {

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

} catch (ArithmeticException e) {

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

}

}

}

**OUTPUT:**

>>java Calculator 3 + 5

Result: 8

>>java Calculator 3 - 5

NegativeResultException: Result is negative: -2

**Question-6**

**Problem Statement:** Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not between 15 and 21 then generate user-defined exception “InvalidAgeException”. If name contains numbers or special characters raise exception “InvalidNameException”. Define the two exception classes.

**Source Code**

//Class InvalidAgeException

public class InvalidAgeException extends Exception {

public InvalidAgeException(String message) {

super(message);

}

}

//Class InvalidNameException

class InvalidNameException extends Exception {

public InvalidNameException(String message) {

super(message);

}

}

//Class Student(with main method)

public class Student {

private int rollNo;

private String name;

private int age;

private String course;

public Student(int rollNo, String name, int age, String course) throws InvalidAgeException, InvalidNameException {

if (age < 15 || age > 21) {

throw new InvalidAgeException("Invalid age. Age should be between 15 and 21.");

}

this.age = age;

if (!isValidName(name)) {

throw new InvalidNameException("Invalid name. Name should not contain numbers or special characters.");

}

this.name = name;

this.rollNo = rollNo;

this.course = course;

}

private boolean isValidName(String name) {

// Regular expression to check if the name contains only letters and spaces

return name.matches("[a-zA-Z ]+");

}

public void displayInfo() {

System.out.println("Roll No: " + rollNo);

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Course: " + course);

}

public static void main(String[] args) {

try {

// Creating a student object with valid data

Student student1 = new Student(1, "Ayushmaan Das", 19, "Chemistry ");

student1.displayInfo();

System.out.println();

// Creating a student object with invalid age

// This will throw InvalidAgeException

Student student2 = new Student(2, "Raj", 22, "Physics");

// Creating a student object with invalid name

// This will throw InvalidNameException

// Uncomment to test

// Student student3 = new Student(3, "Bob123", 20, "Mathematics");

} catch (InvalidAgeException | InvalidNameException e) {

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

}

}

}

**OUTPUT:**

Roll No: 1

Name: Ayushmaan Das

Age: 19

Course: Chemistry

Exception caught: Invalid age. Age should be between 15 and 21.