1. Write a program that tries to access an element outside the bounds of an array and handles the ArrayIndexOutOfBoundsException by printing a user-friendly message.

Program:-

**package** anudip;

**public** **class** ArrayAccessException

{

**public** **static** **void** main(String[] args)

{

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

// Create an array with 3 elements

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

// Try to access an element outside the bounds of the array

**try**

{

// This line will throw ArrayIndexOutOfBoundsException

**int** value = numbers[5];

System.***out***.println("Value at index 5: " + value);

}

**catch** (ArrayIndexOutOfBoundsException e)

{

// Handle the exception and print a user-friendly message

System.***out***.println("Error: Attempted to access an element

outside the bounds of the array.");

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

}

// Continue with the rest of the program

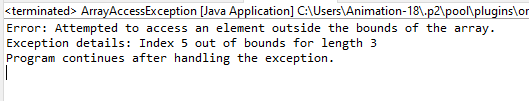
System.***out***.println("Program continues after handling the

exception.");

}

}

Output:-



1. Write a program that attempts to divide a number by zero and handles the ArithmeticException by printing a message that division by zero is not allowed.

Program:-

**package** anudip;

**public** **class** Division

{

**public** **static** **void** main(String[] args)

{

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

**int** numerator = 10;

**int** denominator = 0;

// This will cause ArithmeticException

**try**

{

// Attempt to divide by zero

**int** result = numerator / denominator;

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

}

**catch**(ArithmeticException e)

{

// Handle the exception and print a user-friendly message

System.***out***.println("Error: Division by zero is not

allowed.");

}

// Continue with the rest of the program

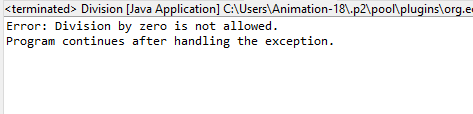
System.***out***.println("Program continues after handling the

exception.");

}

}

Output:-



1. Write a Java program that reads an integer input from the user and throws an IllegalArgumentException if the input is negative. Display an appropriate message when the exception is caught

Program:-

**package** anudip;

**import** java.util.Scanner;

**public** **class** InputValidation

{

**public** **static** **void** main(String[] args)

{

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

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

System.***out***.print("Please enter an integer: ");

**try**

{

// Read the integer input from the user

**int** userInput = scanner.nextInt();

// Check if the input is negative

**if** (userInput < 0)

{

**throw** **new** IllegalArgumentException("Negative values are

not allowed.");

}

// If input is valid, print the input

System.***out***.println("You entered: " + userInput);

}

**catch** (IllegalArgumentException e)

{

// Handle the exception and print a user-friendly message

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

}

**catch**(Exception e)

{

// Handle any other exceptions (e.g., non-integer input)

System.***out***.println("Error: Invalid input. Please enter a

valid integer.");

}

**finally**

{

// Close the scanner

scanner.close();

}

// Continue with the rest of the program

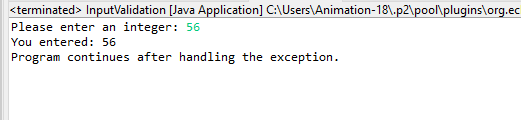
System.***out***.println("Program continues after handling the

exception.");

}

}

Output:-



1. Create a Java method that divides two numbers and declares that it throws an ArithmeticException. Handle the exception in the main method.

Program:-

**package** anudip;

**public** **class** DivisionWithException

{

// Method that divides two numbers and declares that it throws

ArithmeticException

**public** **static** **int** divide(**int** numerator, **int** denominator) **throws**

ArithmeticException

{

**if** (denominator == 0)

{

**throw** **new** ArithmeticException("Division by zero is not

allowed.");

}

**return** numerator / denominator;

}

**public** **static** **void** main(String[] args)

{

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

**int** num1 = 10;

**int** num2 = 0;

// This will cause ArithmeticException

**try**

{

// Attempt to divide the numbers

**int** result = *divide*(num1, num2);

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

}

**catch** (ArithmeticException e)

{

// Handle the ArithmeticException and print a user-friendly

message

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

}

// Continue with the rest of the program

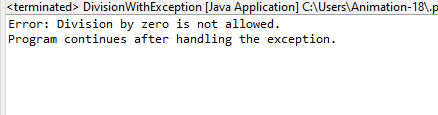
System.***out***.println("Program continues after handling the

exception.");

}

}

Output:-



1. Define a custom exception called InvalidAgeException. Write a Java program that throws this exception if the age provided is less than 18. Handle the exception and display an appropriate message.

Program:-

**package** anudip;

//Define the custom exception

**class** InvalidAgeException **extends** Exception

{

// Constructor that accepts a custom message

**public** InvalidAgeException(String message)

{

**super**(message);

}

}

**public** **class** AgeValidation

{

// Method to check age and throw InvalidAgeException if age is less than

18

**public** **static** **void** validateAge(**int** age) **throws** InvalidAgeException

{

**if**(age < 18)

{

**throw** **new** InvalidAgeException("Age must be 18 or older.

Provided age: " + age);

}

System.***out***.println("Age is valid.");

}

**public** **static** **void** main(String[] args)

{

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

**int** age = 16;

// Example age that is less than 18

**try**

{

// Call the method to validate the age

*validateAge*(age);

}

**catch**(InvalidAgeException e)

{

// Handle the custom exception and display the error message

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

}

// Continue with the rest of the program

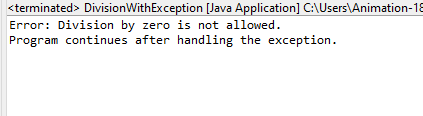
System.***out***.println("Program continues after handling the

exception.");

}

}

Output:-



1. Write a Java program that has a method to validate a user's email address. The method should throw a custom exception InvalidEmailException if the email does not contain @ and .. Handle the exception in the main method.

Program:-

**package** anudip;

//Define the custom exception

**class** InvalidEmailException **extends** Exception

{

// Constructor that accepts a custom message

**public** InvalidEmailException(String message)

{

**super**(message);

}

}

**public** **class** EmailValidation

{

// Method to validate the email address

**public** **static** **void** validateEmail(String email) **throws**

InvalidEmailException

{

// Check if the email contains both '@' and '.'

**if**(email == **null** || !email.contains("@") || !email.contains("."))

{

**throw** **new** InvalidEmailException("Invalid email address: must

contain '@' and '.'");

}

System.***out***.println("Email is valid.");

}

**public** **static** **void** main(String[] args)

{

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

String email = "exampleemail.com"; // Example email address

**try**

{

// Call the method to validate the email

*validateEmail*(email);

}

**catch**(InvalidEmailException e)

{

// Handle the custom exception and display the error message

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

}

// Continue with the rest of the program

System.***out***.println("Program continues after handling the

exception.");

}

}

Output:-

