BITS PILANI, DUBAI CAMPUS
DUBAI INTERNATIONAL ACADEMIC CITY, DUBAI

FIRST SEMESTER 2025 – 2026
COURSE: CSF213/ECOM213/MACF212 (Object Oriented Programming)
COMPONENT: Lab 11 **Week: 12**

Aim:
To learn how to handle runtime errors in Java using exception handling mechanisms such as try, catch, finally, throw, and throws, and to understand how these features help maintain program stability and avoid abnormal termination.

Objective:
To learn how to use try and catch blocks to handle errors safely, use finally for guaranteed execution, to generate exceptions using throw, to declare possible exceptions using throws, and create and use user-defined exceptions for customized error handling.

Problem Statements:

1. Write a Java program that demonstrates how to use try–catch blocks to handle runtime errors safely. Create a class called SafeCalculator that has a method divide(int a, int b) which returns the result of division. The method should perform the division inside a try block, and it should catch an ArithmeticException if the user tries to divide a number by zero. When an exception occurs, display a proper error message instead of stopping the program. In the main method, ask the user to enter two integers: a numerator and a denominator. Then call the divide() method and print the result if the division is successful. This problem helps to understand how Java prevents abnormal termination by using exception handling.

   Expected Input:
   ```
   Numerator: 10
   Denominator: 0
   ```
   Expected Output:
   ```
   Error: Cannot divide by zero.
   Program continues safely.
   ```

2. Write a Java program that demonstrates how to use throw and throws with built-in exceptions. Create a class called FileReaderUtility with a method readFile(String fileName) that is responsible for opening and reading a file. If the fileName is empty, the method should throw an IllegalArgumentException with the message "Filename cannot be empty." The method should also declare throws IOException, because reading a file may cause an input/output error. Inside the method, simulate file reading (no need for real file operations). If the file name is "missing.txt", manually throw an IOException with the message "File not found. In the main() method: Create an array of filenames, including valid names, an empty string, and "missing.txt". For each filename, call readFile() inside a try–catch block. Catch and display the appropriate messages for each exception. This problem shows how Java uses throw to manually generate exceptions and throws to declare exceptions that a method may pass to the caller.
   Expected Input:
   ```
   Files: ["data.txt", "", "missing.txt"]
   ```
   Expected Output:
   ```
   Reading file: data.txt
   Error: Filename cannot be empty.
   Error: File not found
   ```