Ex 2	**Built–in and user–defined exception handling using try, catch, finally, throw, and throws**

Aim

To create a Java application that includes built–in and user–defined exception handling using try, catch, finally, throw, and throws.

Definitions

Built–in exception handling

Built-in exception handling refers to the predefined mechanisms and classes provided by a programming language or its standard library to manage and respond to errors or exceptional conditions that occur during program execution. These are distinct from user-defined exceptions, which are custom exceptions created by programmers for specific application needs.

User–defined exception handling

User-defined exception handling involves creating and utilizing custom exception classes to manage specific error conditions within an application that are not adequately covered by built-in exception types. This approach provides greater clarity, specificity, and maintainability in error management.

try

In Java, the try block is a fundamental component of exception handling. It is used to enclose a section of code that might potentially throw an exception.

catch

In Java exception handling, the catch block is used to handle exceptions that might occur within a try block. It provides a mechanism to gracefully manage runtime errors, preventing the program from crashing and allowing for recovery or alternative actions.

finally

The finally block in Java exception handling serves to ensure that a specific block of code is executed regardless of whether an exception occurs in the try block or is caught by a catch block.

throw

In Java, the throw keyword is used to explicitly throw an exception from a method or any block of code. This allows for custom error handling and the creation of custom exceptions.

throws

In Java exception handling, the throws keyword is used in a method signature to declare the types of checked exceptions that a method might throw during its execution. This declaration serves as a contract, informing any calling code that it needs to either handle these declared exceptions (using try-catch blocks) or further propagate them by adding them to its own throws clause.

Procedure

open NetBeans IDE.

To create a Project go to File Menu → choose New Project → choose Java from Categories →choose Java Application from Projects →click next →specify the project name as ExceptionHandling → uncheck Create Main Class Check-box → click Finish.

Right click on Source Packages Folder → choose New → select Java Class → specify the class name as ExceptionHandlingDemo →click Finish.

Type the following codes in ExceptionHandlingDemo.java:

ExceptionHandlingDemo.java

```
// Custom exception class

class InvalidAgeException extends Exception {

    public InvalidAgeException(String message) {

        super(message);

    }

}

public class ExceptionHandlingDemo {


    // Method that declares it might throw a custom exception

    public static void validateAge(int age) throws InvalidAgeException {

        if (age < 0 || age > 120) {

            // Explicitly throw a custom exception

            throw new InvalidAgeException("Age must be between 0 and 120.");

        } else {

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

        }

    }

    // Method that might throw a built-in exception

    public static void divideNumbers(int numerator, int denominator) {

        try {

            int result = numerator / denominator; // Potential ArithmeticException

            System.out.println("Division result: " + result);
        } catch (ArithmeticException e) {
            // Catching a built-in exception
            System.err.println("Error: Cannot divide by zero. " + e.getMessage());
        } finally {
            System.out.println("Division operation attempted.");
        }
    }
    public static void main(String[] args) {

        // Demonstrate built-in exception handling
        System.out.println("--- Demonstrating Built-in Exception Handling ---");
        divideNumbers(10, 2);
        divideNumbers(5, 0); // This will cause an ArithmeticException

        System.out.println("\n--- Demonstrating User-defined Exception Handling ---");
        try {
            validateAge(25); // Valid age
            validateAge(-5); // This will cause an InvalidAgeException
        } catch (InvalidAgeException e) {
            // Catching the custom exception
            System.err.println("Error: " + e.getMessage());
        } finally {
            System.out.println("Age validation process completed.");
        }

        System.out.println("\n--- Demonstrating another Built-in Exception ---");
        try {
            int[] numbers = {1, 2, 3};
            System.out.println("Accessing element at index 5: " + numbers[5]); // Potential
ArrayIndexOutOfBoundsException
        } catch (ArrayIndexOutOfBoundsException e) {
            System.err.println("Error: Array index out of bounds. " + e.getMessage());
        } finally {
            System.out.println("Array access attempt completed.");
        }
    }
}
```

Right click on ExceptionHandlingDemo.java file → choose Run File. You can see the following result in the output window.

Output

```
run:
--- Demonstrating Built-in Exception Handling ---
Error: Cannot divide by zero. / by zero
Division result: 5
Error: Age must be between 0 and 120.
Division operation attempted.
Error: Array index out of bounds. 5
Division operation attempted.

--- Demonstrating User-defined Exception Handling ---
Age is valid: 25
Age validation process completed.

--- Demonstrating another Built-in Exception ---
Array access attempt completed.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Result

Thus, a Java application that includes built–in and user–defined exception handling using try, catch, finally, throw, and throws has been created.