***CASE STUDY NO. 5***

* **Title**
* Exception handling and generic programming using array list (ArrayList class)
* **Theory Concepts Of Java/OOP Used in This Case Study**

1. **ArrayList:**   
   An ArrayList is a resizable array-like data structure in Java that belongs to the java.util package. Unlike traditional arrays, it can dynamically grow or shrink as elements are added or removed. It provides methods for insertion, deletion, and retrieval of elements. In this program, ArrayList<Integer> is used to store a list of integer values
2. **Generics:**   
   Generics in Java allow you to define classes, interfaces, and methods with a placeholder for the type of data they operate on. This feature enforces type safety at compile-time, preventing runtime errors and unnecessary type casting. For example, ArrayList<Integer> ensures only integers are stored in the list, reducing type-related-errors.
3. **Exception:**   
   Exception handling is a programming mechanism that manages runtime errors gracefully to ensure the program does not crash unexpectedly. The try-catch block is used to identify and handle exceptions, allowing the program to recover or take alternative actions. In this program, exception handling is demonstrated for managing invalid index access.
4. **IndexOutOfBoundsException:**   
   This is a runtime exception that occurs when trying to access an array or collection element with an invalid index—either negative or beyond the size of the collection. The program uses a try-catch block to detect and handle this exception, preventing abrupt termination when accessing an invalid index.

* **Algorithm:**

1. **Start**
2. Create an ArrayList of type Integer named numbers.
3. Begin a try block:
   * Add the integer 10 to the numbers list.
   * Add the integer 20 to the numbers list.
   * Add the integer 30 to the numbers list.
   * Print the content of the numbers list.
   * Attempt to access the element at index 3 of the numbers list.
   * Print the retrieved element.
4. If an IndexOutOfBoundsException occurs:
   * Enter the catch block.
   * Print the exception message using e.getMessage().
5. **End**

* **Flowchart:**

A diagram of a flowchart

Description automatically generated

* **Program/Code:**
* import java.util.ArrayList;

public class GenericArrayListExample

{

    public static void main(String[] args)

    {

        ArrayList<Integer> numbers = new ArrayList<>();

        try

        {

            numbers.add(10);

            numbers.add(20);

            numbers.add(30);

            System.out.println("Numbers: " + numbers);

            int number = numbers.get(3); // Accessing invalid index

            System.out.println("Number at index 3: " + number);

        }

        catch (IndexOutOfBoundsException e)

        {

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

        }

    }

}

* **Output:**
* Numbers: [10, 20, 30]

Exception caught: Index 3 out of bounds for length 3

* **Conclusion**
* This program illustrates the integration of **exception handling** with **generic programming** using the ArrayList class in Java. By leveraging generics, type safety is ensured while storing and retrieving data. Exception handling adds robustness, gracefully managing runtime errors such as invalid index access. This modular and type-safe approach enhances code reliability and usability.