***CASE STUDY NO. 6***

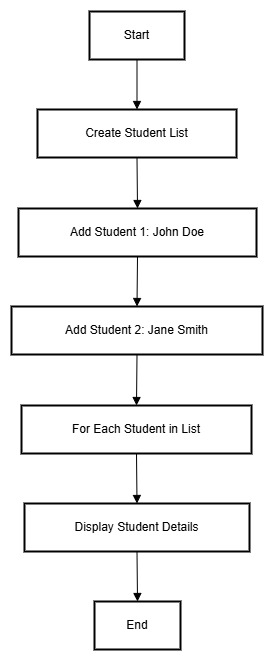
* **Title**
* Student Management System
* **Theory Concepts Of Java/OOP Used in This Case Study**

1. **ArrayList:**   
   A collection class in Java that allows dynamic resizing. It stores elements in an ordered manner. In this code, an ArrayList<Student> is used to store the students' data.
2. **Class:**   
   A blueprint to create objects that encapsulate both data and methods. The Student class is used to represent a student with attributes like name, age, and course.
3. **Object:**   
   An instance of a class. For example, new Student("John Doe", 20, "Computer Science") creates an object of the Student class.
4. **IndexOutOfBoundsException:**   
   The process of hiding the internal details of an object and providing controlled access through methods. The student attributes (name, age, and course) are private, and access is provided via the constructor and displayStudentDetails() method.
5. **Constructor**:  
   A special method used to initialize objects. In the Student class, the constructor initializes the name, age, and course attributes when creating a new Student object.
6. **Looping (For-each loop)**:  
   The for-each loop iterates over the students array list and calls displayStudentDetails() for each student.

* **Algorithm:**

1. **Start**
2. Define a Student class with the following:
   * Fields: name, age, and course.
   * A constructor to initialize name, age, and course.
   * A method displayStudentDetails to print student details.
3. Inside the main method of the StudentManagementSystem class:
   * Create an ArrayList to store Student objects.
4. Add new Student objects to the ArrayList:
   * Add a student with details: "John Doe", age 20, "Computer Science".
   * Add a student with details: "Jane Smith", age 22, "Mechanical Engineering".
5. Loop through each Student object in the ArrayList:
   * Call the displayStudentDetails method to print the details of each student.
6. **End**

* **Flowchart:**



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

class Student

{

    private String name;

    private int age;

    private String course;

    public Student(String name, int age, String course)

    {

        this.name = name;

        this.age = age;

        this.course = course;

    }

    public void displayStudentDetails()

    {

        System.out.println("Name: " + name + ", Age: " + age + ", Course: " + course);

    }

}

public class StudentManagementSystem

{

    public static void main(String[] args)

    {

        ArrayList<Student> students = new ArrayList<>();

        students.add(new Student("John Doe", 20, "Computer Science"));

        students.add(new Student("Jane Smith", 22, "Mechanical Engineering"));

        for (Student student : students)

        {

            student.displayStudentDetails();

        }

    }

}

* **Output:**
* Name: John Doe, Age: 20, Course: Computer Science

Name: Jane Smith, Age: 22, Course: Mechanical Engineering

* **Conclusion**
* The Student Management System demonstrates how Object-Oriented Programming principles can be applied to manage student data efficiently. The use of **classes** and **objects** models real-world entities (students) with encapsulation of their attributes. By utilizing **ArrayList**, the program can dynamically store and retrieve student information. The **for-each loop** simplifies the process of iterating through the student list, making the system scalable and easy to maintain.