SAPTHAGIRI NPS UNIVERSITY

No.14/5, Chikkasandra, Hesaraghatta Main Road
Bangalore – 560057

Department of Computer Science and Engineering – Allied Branches

Object Oriented Programming Laboratory Manual

Subject Code –24BEAML302

Semester – III

Academic Year

2025-2026

SAPTHAGIRI NPS UNIVERSITY, BENGALURU-57

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-AIML

LABORATORY CERTIFICATE

This is to certify that Mr. / Ms _____

has satisfactorily completed the course of Experiments in practical prescribed by the Department during the year _____ .

_____

Name of the Candidate : _____

SRN No. : _____ Semester : _____

Signature of the Staff in-charge :

Marks	
Max. Marks	Obtained

Date :

OOP Laboratory

Exp. No.	Date	Experiment Title
1a		Write a java program to calculate the total Student Marks & Grade Calculator. The program should also print the result of the students whether the student is pass or fail
1b		Write a Java Program to Implement a Basic ATM Machine with Deposit, Withdrawal, and Balance Check
2		Write a Java Program for Bank Account Management System using Encapsulation, Constructors, and Arrays
3		Develop a JAVA program to add TWO matrices of suitable order N (The value of N should beread from command line arguments).
4		Develop a program of Drawing Shapes using Java that demonstrates: • Inheritance (class hierarchy) • Method Overloading (same method name, different parameters) • Method Overriding (subclass changes behaviour of parent method)
5.		Develop a java program to demonstrate the abstract classes for doing calculations on areas & perimeters of certain shapes.
6a		Write a Java Program to demonstrate Exception Handling (Division Calculator (Handle Divide by Zero Exception))
6b		Write a Java Program to demonstrate Exception Handling (Square Root Calculator (Handle Negative Numbers))
7a		Write a Java Program to demonstrate Multithreading Applications
7b		Write a Java Program **using** Runnable **and** synchronized — this one simulates two threads depositing money into a bank account safely

8		Write a Java program using the **Collections Framework** (ArrayList, HashMap, HashSet) to simulate a **Student Course Management System**. The program should allow the user to: 1. Add students. 2. Enroll them into courses. 3. Record marks for courses. 4. Display all students with their enrolled courses and marks.
9		Write a Java program using **PriorityQueue** to simulate this project submission system.
10		Develop a **Java Swing program** that demonstrates how to create a simple GUI with buttons, labels, and images.

Program 1a:-

Write a java program to calculate the total Student Marks & Grade Calculator. The program should also print the result of the students whether the student is pass or fail

```
import java.util.Scanner;

public class StudentGradeCalculator {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("=== Student Marks & Grade Calculator ===");


      // Input student details
      System.out.print("Enter Student Name: ");
      String name = sc.nextLine();
      System.out.print("Enter number of subjects: ");
      int subjectCount = sc.nextInt();
      double totalMarks = 0;


      // Loop to take marks for each subject
      for (int i = 1; i <= subjectCount; i++) {
         System.out.print("Enter marks for Subject " + i + " (out of 100): ");
         double marks = sc.nextDouble();


         // Validate marks using conditional
         if (marks < 0 || marks > 100) {
            System.out.println("Invalid marks! Please enter again.");
            i--; // repeat this subject input
            continue;
         }
         totalMarks += marks;
      }


      // Calculate percentage
      double percentage = totalMarks / subjectCount;
```

```
        // Determine grade
        char grade;
        if (percentage >= 90) {
            grade = 'A';
        } else if (percentage >= 75) {
            grade = 'B';
        } else if (percentage >= 60) {
            grade = 'C';
        } else if (percentage >= 40) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        // Pass/Fail using logical operator
        boolean isPass = (percentage >= 40);
        // Display Result
        System.out.println("\n=== RESULT ===");
        System.out.println("Name: " + name);
        System.out.println("Total Marks: " + totalMarks + " / " + (subjectCount * 100));
        System.out.println("Percentage: " + percentage + "%");
        System.out.println("Grade: " + grade);
        System.out.println(isPass ? "Status: PASS" : "Status: FAIL");
        sc.close();
    }
}
```

Output:

```
=== Student Marks & Grade Calculator ===
Enter Student Name: Anil
Enter number of subjects: 5
Enter marks for Subject 1 (out of 100): 85
Enter marks for Subject 2 (out of 100): 92
Enter marks for Subject 3 (out of 100): 78
```

Enter marks for Subject 4 (out of 100): 88

Enter marks for Subject 5 (out of 100): 95

=== RESULT ===

Name: Anil

Total Marks: 438.0 / 500

Percentage: 87.6%

Grade: B

Status: PASS

Brainstorming:

- Replace “Subject 1”, “Subject 2” with real names like “Math”, “Science”, “English”.
- Instead of letters (A, B, C), show “Excellent”, “Very Good”, “Average”, “Pass”, “Fail”.

Program 1b

Write a Java Program to Implement a Basic ATM Machine with Deposit, Withdrawal, and Balance Check

```
import java.util.Scanner;
public class SimpleATM {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double balance = 1000.0; // initial balance
      int choice;
      do {
         System.out.println("\n=== ATM Menu ===");
         System.out.println("1. Check Balance");
         System.out.println("2. Deposit Money");
         System.out.println("3. Withdraw Money");
         System.out.println("4. Exit");
         System.out.print("Enter your choice: ");

         choice = sc.nextInt();

         switch(choice) {
            case 1:
               System.out.println("Your balance: $" + balance);
               break;
            case 2:
               System.out.print("Enter amount to deposit: ");
               double deposit = sc.nextDouble();
               balance += deposit;  // using + operator
               System.out.println("Deposited $" + deposit + ". New balance: $" + balance);
               break;
            case 3:
               System.out.print("Enter amount to withdraw: ");
               double withdraw = sc.nextDouble();
```

```
                if (withdraw <= balance) {
                    balance -= withdraw;  // using - operator
                    System.out.println("Withdrew $" + withdraw + ". Remaining balance: $" + balance);
                } else {
                    System.out.println("Insufficient balance!");
                }
                break;
            case 4:
                System.out.println("Thank you for using ATM. Goodbye!");
                break;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    } while(choice != 4);
    sc.close();
  }
}
```

Output:

```
=== ATM Menu ===
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
Enter your choice: 1
Your balance: $1000.0

=== ATM Menu ===
1. Check Balance
2. Deposit Money
3. Withdraw Money
4. Exit
```

Enter your choice: 2

Enter amount to deposit: 500

Deposited $500.0. New balance: $1500.0

=== ATM Menu ===

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Exit

Enter your choice: 3

Enter amount to withdraw: 200

Withdrew $200.0. Remaining balance: $1300.0

Brainstorming:

- the ATM warn if the balance falls below ₹500?
- Instead of directly exiting, can you ask, *"Are you sure you want to exit? (Y/N)"*?

Program 2

Write a Java Program for Bank Account Management System using Encapsulation, Constructors, and Arrays

```
import java.util.Scanner;
class BankAccount {
    private int accountNumber;
    private String accountHolder;
    private double balance;
    // Constructor
    public BankAccount(int accountNumber, String accountHolder, double balance) {
        this.accountNumber = accountNumber;
        this.accountHolder = accountHolder;
        this.balance = balance;
    }

    // Deposit
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + " | New Balance: " + balance);
    }

    // Withdraw
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew: " + amount + " | New Balance: " + balance);
        } else {
            System.out.println("Insufficient balance!");
        }
    }
    // Display Account Info
    public void display() {
```

```
        System.out.println("Account No: " + accountNumber +
                           " | Holder: " + accountHolder +
                           " | Balance: " + balance);
    }
    public int getAccountNumber() {
        return accountNumber;
    }
}

public class SimpleBankApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Create accounts
        BankAccount[] accounts = new BankAccount[2];
        accounts[0] = new BankAccount(101, "Alice", 5000);
        accounts[1] = new BankAccount(102, "Bob", 3000);
        while (true) {
            System.out.println("\n--- Menu ---");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Display Account");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();
            if (choice == 4) {
                System.out.println("Exiting... Goodbye!");
                break;
            }
            System.out.print("Enter Account Number: ");
            int accNo = sc.nextInt();
            BankAccount selectedAccount = null;
```

```
            // Find account directly here
            for (BankAccount acc : accounts) {
                if (acc.getAccountNumber() == accNo) {
                    selectedAccount = acc;
                    break;
                }
            }
            if (selectedAccount == null) {
                System.out.println("Account not found!");
                continue;
            }
            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double dep = sc.nextDouble();
                    selectedAccount.deposit(dep);
                    break;
                case 2:
                    System.out.print("Enter amount to withdraw: ");
                    double wit = sc.nextDouble();
                    selectedAccount.withdraw(wit);
                    break;
                case 3:
                    selectedAccount.display();
                    break;
                default:
                    System.out.println("Invalid choice!");
            }
        }
        sc.close();
    }
}
```

Output:

```
--- Menu ---
1. Deposit
2. Withdraw
3. Display Account
4. Exit
Enter choice: 1
Enter Account Number: 101
Enter amount to deposit: 1000
Deposited: 1000.0 | New Balance: 6000.0

--- Menu ---
1. Deposit
2. Withdraw
3. Display Account
4. Exit
Enter choice: 2
Enter Account Number: 102
Enter amount to withdraw: 500
Withdrew: 500.0 | New Balance: 2500.0

--- Menu ---
1. Deposit
2. Withdraw
3. Display Account
4. Exit
Enter choice: 3
Enter Account Number: 101
Account No: 101 | Holder: Alice | Balance: 6000.0

--- Menu ---
1. Deposit
2. Withdraw
3. Display Account
4. Exit
Enter choice: 4
Invalid choice!
```

Brainstorming:

- How can you add an option in the menu to display **all accounts at once?**
- What happens if a user enters a **negative amount**? How can you prevent that?

Program 3

Develop a JAVA program to add TWO matrices of suitable order N (The value of N should beread from command line arguments).

```
import java.util.Scanner;

public class Matrix {

public static void main(String[] args) { Scanner scanner = new Scanner(System.in);
System.out.println("Enter the value of N: "); int N = scanner.nextInt();

int[][] firstMatrix = new int[N][N]; int[][] secondMatrix = new int[N][N]; int[][]
resultMatrix = new int[N][N];

System.out.println("Enter the elements of first matrix: "); for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) { firstMatrix[i][j] = scanner.nextInt();

}

}

System.out.println("Enter the elements of second matrix: "); for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) { secondMatrix[i][j] = scanner.nextInt();

}

}

for (int i = 0; i < N; i++) { for (int j = 0; j < N; j++) {

resultMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];

}

}

System.out.println("The resultant matrix is: "); for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) { System.out.print(resultMatrix[i][j] + " ");

}

System.out.println();

}

}

}
```

1. Enter the value of N: 2

Enter the elements of first matrix:

3 4

4 5

Enter the elements of second matrix:

5 6

7 8

The resultant matrix is: 8 10
11 13

Program 4

Develop a program of Drawing Shapes using **Java** that demonstrates:

- **Inheritance** (class hierarchy)
- **Method Overloading** (same method name, different parameters)
- **Method Overriding** (subclass changes behaviour of parent method)

```
import java.util.Scanner;
// Base class
abstract class Shape {
    // Method Overloading for area
    double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }
    double calculateArea(double length, double breadth) {
        return length * breadth;
    }
    double calculateArea(double base, double height, boolean isTriangle) {
        return 0.5 * base * height;
    }
    // Method Overriding for perimeter
    abstract double calculatePerimeter();
}
// Circle class
class Circle extends Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }
    @Override
    double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}
```

```
// Rectangle class
class Rectangle extends Shape {
    double length, breadth;
    Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }
    @Override
    double calculatePerimeter() {
        return 2 * (length + breadth);
    }
}
// Triangle class
class Triangle extends Shape {
    double a, b, c;
    Triangle(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }
    @Override
    double calculatePerimeter() {
        return a + b + c;
    }
}
// Main class
public class EngineeringShapes {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("--- Engineering Drawing Shapes ---");
        System.out.println("Choose a shape: 1.Circle  2.Rectangle  3.Triangle");
        int choice = sc.nextInt();
        Shape shape = null;
        switch (choice) {
```

```
    case 1:
        System.out.print("Enter radius of circle: ");
        double r = sc.nextDouble();
        shape = new Circle(r);
        System.out.println("Area (Overloaded method): " + shape.calculateArea(r));
        break;
    case 2:
        System.out.print("Enter length of rectangle: ");
        double l = sc.nextDouble();
        System.out.print("Enter breadth of rectangle: ");
        double b = sc.nextDouble();
        shape = new Rectangle(l, b);
        System.out.println("Area (Overloaded method): " + shape.calculateArea(l, b));
        break;
    case 3:
        System.out.print("Enter base of triangle: ");
        double base = sc.nextDouble();
        System.out.print("Enter height of triangle: ");
        double h = sc.nextDouble();
        System.out.print("Enter side a: ");
        double a = sc.nextDouble();
        System.out.print("Enter side b: ");
        double bb = sc.nextDouble();
        System.out.print("Enter side c: ");
        double cc = sc.nextDouble();
        shape = new Triangle(a, bb, cc);
        System.out.println("Area (Overloaded method): " + shape.calculateArea(base, h, true));
        break;
    default:
        System.out.println("Invalid choice!");
}
if (shape != null) {
    System.out.println("Perimeter (Overridden method): " + shape.calculatePerimeter());
}
```

```
        sc.close();
    }
}
```

Output:

Choose a shape: 1.Circle 2.Rectangle 3.Triangle

1

Enter radius of circle: 5

Area (Overloaded method): 78.53981633974483

Perimeter (Overridden method): 31.41592653589793

Choose a shape: 1.Circle 2.Rectangle 3.Triangle

2

Enter length of rectangle: 10

Enter breadth of rectangle: 4

Area (Overloaded method): 40.0

Perimeter (Overridden method): 28.0

Choose a shape: 1.Circle 2.Rectangle 3.Triangle

3

Enter base of triangle: 6

Enter height of triangle: 4

Enter side a: 5

Enter side b: 6

Enter side c: 7

Area (Overloaded method): 12.0

Perimeter (Overridden method): 18.0

Brainstorming:

- Add new class like `Square`., inheriting from `Shape`.
- Override `area()` and `perimeter()` methods.

Program 5

Develop a java program to demonstrate the abstract classes for doing calculations on areas & perimeters of certain shapes.

```
import java.util.Scanner;
// Abstract class
abstract class Shape {
   abstract double calculateArea();
   abstract double calculatePerimeter();
}

// Circle class
class Circle extends Shape {
   double radius;
   Circle(double radius) {
      this.radius = radius;
   }
   @Override
   double calculateArea() {
      return Math.PI * radius * radius;
   }
   @Override
   double calculatePerimeter() {
      return 2 * Math.PI * radius;
   }
}
// Rectangle class
class Rectangle extends Shape {
   double length, width;
   Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }
   @Override
```

```
    double calculateArea() {
        return length * width;
    }
    @Override
    double calculatePerimeter() {
        return 2 * (length + width);
    }
}
// Triangle class
class Triangle extends Shape {
    double a, b, c; // sides
    Triangle(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }
    @Override
    double calculateArea() {
        // Heron's formula
        double s = (a + b + c) / 2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
    }
    @Override
    double calculatePerimeter() {
        return a + b + c;
    }
}
public class EngineeringShapes {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Choose a shape:\n1. Circle\n2. Rectangle\n3. Triangle");
        int choice = sc.nextInt();
        Shape shape = null;
```

```
        switch (choice) {
            case 1:
                System.out.print("Enter radius: ");
                double r = sc.nextDouble();
                shape = new Circle(r);
                break;
            case 2:
                System.out.print("Enter length: ");
                double l = sc.nextDouble();
                System.out.print("Enter width: ");
                double w = sc.nextDouble();
                shape = new Rectangle(l, w);
                break;
            case 3:
                System.out.print("Enter side a: ");
                double a = sc.nextDouble();
                System.out.print("Enter side b: ");
                double b = sc.nextDouble();
                System.out.print("Enter side c: ");
                double c = sc.nextDouble();
                shape = new Triangle(a, b, c);
                break;
            default:
                System.out.println("Invalid choice!");
                System.exit(0);
        }
        System.out.println("Area = " + shape.calculateArea());
        System.out.println("Perimeter = " + shape.calculatePerimeter());
        sc.close();
    }
}
```

Output:

Choose a shape:

1. Circle

2. Rectangle

3. Triangle

2

Enter length: 10

Enter width: 5

Area = 50.0

Perimeter = 30.0

Choose a shape:

1. Circle

2. Rectangle

3. Triangle

1

Enter radius: 7

Area = 153.93804002589985

Perimeter = 43.982297150257104

Brainstorming:

- Add new class like `Square`.

Program 6a

Write a Java Program to demonstrate Exception Handling (Division Calculator (Handle Divide by Zero Exception))

```
import java.util.Scanner;

public class DivisionCalculator {

   public static void main(String[] args) {

      Scanner sc = new Scanner(System.in);

      try {

         System.out.print("Enter numerator: ");

         int num = sc.nextInt();

         System.out.print("Enter denominator: ");

         int den = sc.nextInt();

         int result = num / den;

         System.out.println("Result = " + result);

      } catch (ArithmeticException e) {

         System.out.println("Error: Division by zero is not allowed!");

      } finally {

         System.out.println("Program finished.");

      }

   }

}
```

Output:

```
Enter numerator: 10

Enter denominator: 0

Error: Division by zero is not allowed!

Program finished.
```

Program 6b

Write a Java Program to demonstrate Exception Handling (Square Root Calculator (Handle Negative Numbers))

```
import java.util.Scanner;
public class SqrtCalculator {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      try {
         System.out.print("Enter a number: ");
         double num = sc.nextDouble();
         if (num < 0) {
            throw new IllegalArgumentException("Cannot find square root of negative number!");
         }
         double result = Math.sqrt(num);
         System.out.println("Square Root = " + result);

      } catch (IllegalArgumentException e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
}
```

Output:

Enter a number: -9

Error: Cannot find square root of negative number!

Brainstorming:

- Use a custom exception for invalid denominator(5a).
- Extend it to **calculate cube root or nth root** with proper validation(5b).

Program 7a

Write a Java Program to demonstrate Multithreading Applications

```
class ProcessTask extends Thread {
   private String taskName;
   ProcessTask(String taskName) {
      this.taskName = taskName;
   }
   // Each thread executes this run() method
   public void run() {
      for (int i = 1; i <= 5; i++) {
         System.out.println(taskName + " - Step " + i);
         try {
            Thread.sleep(500); // simulate processing time
         } catch (InterruptedException e) {
            System.out.println(taskName + " interrupted!");
         }
      }
      System.out.println(taskName + " completed!");
   }
}
public class ThreadMethodDemo {
   public static void main(String[] args) {
      // Create threads for different “methods/processes”
      ProcessTask process1 = new ProcessTask("Load Calculation");
      ProcessTask process2 = new ProcessTask("Temperature Monitoring");

      // Start the threads
      process1.start();
      process2.start();
   }
}
```

Output:

```
Load Calculation - Step 1
Temperature Monitoring - Step 1
Load Calculation - Step 2
Temperature Monitoring - Step 2
Load Calculation - Step 3
Temperature Monitoring - Step 3
Load Calculation - Step 4
Temperature Monitoring - Step 4
Load Calculation - Step 5
Temperature Monitoring - Step 5
Load Calculation completed!
Temperature Monitoring completed!
```

Program 7b

Write a Java Program using **Runnable** and **synchronized — this one simulates two threads depositing money into a bank account safely**

```
class BankAccount {
   private int balance = 0;
   // synchronized method to safely deposit
   public synchronized void deposit(int amount, String name) {
      balance += amount;
      System.out.println(name + " deposited " + amount + ", Balance = " + balance);
   }
}
class DepositTask implements Runnable {
   private BankAccount account;
   private String threadName;
   public DepositTask(BankAccount account, String threadName) {
      this.account = account;
      this.threadName = threadName;
   }
   @Override
   public void run() {
      for (int i = 1; i <= 3; i++) {
         account.deposit(100, threadName);
         try {
            Thread.sleep(100); // simulate delay
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
   }
}
public class BankDemo {
   public static void main(String[] args) {
```

```
        BankAccount account = new BankAccount();
        // Create two deposit threads
        Thread t1 = new Thread(new DepositTask(account, "Thread-1"));
        Thread t2 = new Thread(new DepositTask(account, "Thread-2"));
        t1.start();
        t2.start();
    }
}
```

Output:

```
Thread-1 deposited 100, Balance = 100
Thread-2 deposited 100, Balance = 200
Thread-1 deposited 100, Balance = 300
Thread-2 deposited 100, Balance = 400
Thread-1 deposited 100, Balance = 500
Thread-2 deposited 100, Balance = 600
```

Program 8

Write a Java program using the **Collections Framework** (ArrayList, HashMap, HashSet) to simulate a **Student Course Management System**.
The program should allow the user to:

5. Add students.
6. Enroll them into courses.
7. Record marks for courses.
8. Display all students with their enrolled courses and marks.

```
import java.util.*;
class Student {
   String name;
   HashSet<String> courses;              // Stores enrolled courses
   HashMap<String, Integer> marks;       // Stores marks for courses
   Student(String name) {
      this.name = name;
      this.courses = new HashSet<>();
      this.marks = new HashMap<>();
   }
   // Enroll in course
   void enrollCourse(String course) {
      courses.add(course);
   }
   // Add marks
   void addMarks(String course, int mark) {
      if (courses.contains(course)) {
         marks.put(course, mark);
      } else {
         System.out.println("Student not enrolled in " + course);
      }
   }
   // Display details
   void displayStudent() {
```

```
        System.out.println("Student: " + name);
        System.out.println("Courses Enrolled: " + courses);
        System.out.println("Marks: " + marks);
        System.out.println("-------------------------");
    }
}
public class StudentCourseSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<Student> students = new ArrayList<>();
        while (true) {
            System.out.println("\n===== Student Course Management System =====");
            System.out.println("1. Add Student");
            System.out.println("2. Enroll Course");
            System.out.println("3. Add Marks");
            System.out.println("4. Display All Students");
            System.out.println("5. Exit");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();
            sc.nextLine(); // consume newline
            switch (choice) {
                case 1:
                    System.out.print("Enter student name: ");
                    String name = sc.nextLine();
                    students.add(new Student(name));
                    System.out.println("Student added successfully!");
                    break;
                case 2:
                    System.out.print("Enter student name: ");
                    String sName = sc.nextLine();
                    System.out.print("Enter course name: ");
                    String course = sc.nextLine();
                    for (Student s : students) {
```

```
            if (s.name.equalsIgnoreCase(sName)) {
                s.enrollCourse(course);
                System.out.println("Course enrolled successfully!");
                break;
            }
        }
        break;
    case 3:
        System.out.print("Enter student name: ");
        String stName = sc.nextLine();
        System.out.print("Enter course name: ");
        String cName = sc.nextLine();
        System.out.print("Enter marks: ");
        int mark = sc.nextInt();
        sc.nextLine();
        for (Student s : students) {
            if (s.name.equalsIgnoreCase(stName)) {
                s.addMarks(cName, mark);
                System.out.println(" Marks added successfully!");
                break;
            }
        }
        break;
    case 4:
        for (Student s : students) {
            s.displayStudent();
        }
        break;
    case 5:
        System.out.println("Exiting... Goodbye!");
        sc.close();
        return;
    default:
```

```
                System.out.println(" Invalid choice. Try again!");
            }
        }
    }
}
```

Output:

```
===== Student Course Management System =====
1. Add Student
2. Enroll Course
3. Add Marks
4. Display All Students
5. Exit
Enter choice: 1
Enter student name: Anil
 Student added successfully!

Enter choice: 2
Enter student name: Anil
Enter course name: Java
 Course enrolled successfully!

Enter choice: 3
Enter student name: Anil
Enter course name: Java
Enter marks: 85
Marks added successfully!

Enter choice: 4
Student: Anil
Courses Enrolled: [Java]
Marks: {Java=85}
-------------------------
```

Program 9

Write a Java program using PriorityQueue **to simulate this project submission system.**

```
import java.util.PriorityQueue;
class Project implements Comparable<Project> {
    String studentName;
    String projectTitle;
    int priority; // 1 = Highest, 4 = Lowest
    // Constructor
    public Project(String studentName, String projectTitle, int priority) {
        this.studentName = studentName;
        this.projectTitle = projectTitle;
        this.priority = priority;
    }
    // Compare based on priority
    @Override
    public int compareTo(Project other) {
        return Integer.compare(this.priority, other.priority);
    }
    @Override
    public String toString() {
        return "Student: " + studentName +
               ", Project: " + projectTitle +
               ", Priority: " + priority;
    }
}
public class PriorityQueueExample {
    public static void main(String[] args) {
        // Creating a PriorityQueue
```

```
        PriorityQueue<Project> projectQueue = new PriorityQueue<>();
        // Adding projects (like students submitting work)
        projectQueue.add(new Project("Alice", "AI-based Attendance System", 1));
        projectQueue.add(new Project("Bob", "Mechanical CAD Model", 3));
        projectQueue.add(new Project("Charlie", "IoT Smart Parking System", 2));
        projectQueue.add(new Project("David", "Civil Bridge Design", 4));
        // Processing projects based on priority
        System.out.println("Processing projects in order of priority:\n");
        while (!projectQueue.isEmpty()) {
            System.out.println(projectQueue.poll());
        }
    }
}
```

Output:

Processing projects in order of priority:

Student: Alice, Project: AI-based Attendance System, Priority: 1

Student: Charlie, Project: IoT Smart Parking System, Priority: 2

Student: Bob, Project: Mechanical CAD Model, Priority: 3

Student: David, Project: Civil Bridge Design, Priority: 4

Program 10

Develop a Java Swing program that demonstrates how to create a simple GUI with buttons, labels, and images.

```
Import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class MyFrame extends JFrame implements ActionListener
{

   JButton jb,jb1,jb2;
   JLabel jl;
   MyFrame()
   {

      setLayout(new FlowLayout());
      jl=new JLabel();
      jb=new JButton("SCHOOL OF ENGINEERING");
      ImageIcon ii=new ImageIcon("snps.JPG");
       jb1=new JButton("SNPSU",ii);

      ImageIcon  ii2=new  ImageIcon("cs.JPG");
      jb2=new JButton("CSE", ii2);

      add(jb);   add(jb1); add(jb2); add(jl);

      jb.addActionListener(this);
      jb1.addActionListener(this);
      jb2.addActionListener(this);
   }

   public void actionPerformed(ActionEvent ae)
   {

      jl.setText("You Pressed:  "+ae.getActionCommand());
   }

}

class FrameDemo
{

   public static void main(String arg[])
   {

      MyFrame        f=new        MyFrame();
      f.setTitle("Welcome     to     Swings");
      f.setSize(500,500);
      f.setVisible(true);
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
```

}

Output: With image it dispalys