LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(An Autonomous Institution since 2010)
Accredited by NAAC with 'A' Grade & NBA (Under Tier - I),
An ISO 21001:2018,14001:2015,50001:2018 Certified Institution
Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada
L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.
hodcse@lbrce.ac.in, cseoffice@lbrce.ac.in, Phone: 08659-222 933, Fax: 08659-222931
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

OBJECT-ORIENTED PROGRAMMING THROUGH JAVA LAB MANUAL

II B.TECH III SEM

(R23 Regulations)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Name & Code : **Object Oriented Programming Through Java lab (23CS54)**
L-T-P Structure : **0-0-3** **Credits:** 1.5
Program/Sem/Sec : B.Tech-CSE / III SEM / A,B, C & D SECTIONS **A.Y :** 2024-25

PREREQUISITE : **C Programming Language**

Course Educational Objectives:

This course aims to equip students with a strong foundation in Object-Oriented Programming (OOP) using Java. Students will learn to implement classes, objects, methods, inheritance, runtime polymorphism, and exception handling, including user-defined exceptions. The course also covers JDBC for database connectivity, multithreading, event handling, and organizing code using packages, along with developing interactive user interfaces using JavaFX.

Course Outcomes (COs): ***At the end of this course, the student will be able to***

- **CO1:** Implement basic concepts of the java programming language. **(Apply-L3)**
- **CO2:** Implement object-oriented programming concepts and exception handling **(Apply- L3)**
- **CO3:** Design multithreaded and GUI based applications. **(Apply-L3)**
- **CO4:** Improve individual / teamwork skills, communication & report writing skills with ethical values.

COURSE ARTICULATION MATRIX (Correlation of Cos & POs, PSOs):

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2	-	2	-	-	-	-	-	-	2	2	-	-
CO2	3	2	2	-	2	-	-	-	-	-	-	2	2	-	2
CO3	3	2	2	-	2	-	-	-	-	-	-	2	2	-	2
CO4	-	-	-	-	-	-	-	2	2	2	-	-	-	-	-

Note: Enter Correlation Levels 1 or 2 or 3. If there is no correlation, put '-'
1- Slight (Low), 2 – Moderate (Medium), 3 - Substantial (High).

TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

Vision of the Department

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department

- **DM1:** To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.
- **DM2:** To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.
- **DM3:** Expose the students to cutting edge technologies which enhance their employability and knowledge.
- **DM4:** Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs)

- **PEO1:** Pursue higher education, entrepreneurship and research to compete at global level.
- **PEO2:** Design and develop products innovatively in computer science and engineering and in other allied fields.
- **PEO3:** Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all the levels with ethics and necessary attitude.
- **PEO4:** Serve ever-changing needs of society with a pragmatic perception.

TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

PROGRAMME OUTCOMES (POs):

PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success oforganization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO 3	To inculcate an ability to analyse, design and implement database applications.

Modules/ Exercises

Exercise - 1:

a. Write a JAVA program to display default value of all primitive data type of JAVA
b. Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

Exercise - 2

a. Write a JAVA program to search for an element in a given list of elements using binary search mechanism.
b. Write a JAVA program to sort for an element in a given list of elements using bubble sort
c. Write a JAVA program using String Buffer to delete, remove character.

Exercise - 3

a. Write a JAVA program to implement class mechanism. Create a class, methods and invoke them inside main method.
b. Write a JAVA program implements method overloading.
c. Write a JAVA program to implement constructor.
d. Write a JAVA program to implement constructor overloading.

Exercise - 4

a. Write a JAVA program to implement Single Inheritance.
b. Write a JAVA program to implement multilevel Inheritance.
c. Write a JAVA program for abstract class to find areas of different shape.

Exercise - 5

a. Write a JAVA program give example for "super" keyword.
b. Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?
c. Write a JAVA program that implements Runtime polymorphism.

Exercise - 6

a. Write a JAVA program that describes exception handling mechanism.
b. Write a JAVA program Illustrating Multiple catch clauses.
c. Write a JAVA program for creation of Java Built-in Exceptions.
d. Write a JAVA program for creation of User Defined Exception.

Exercise - 7

a. Write a JAVA program that creates threads by extending Thread class. First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds (Repeat the same by implementing Runnable).
b. Write a program illustrating is Alive and join ()
c. Write a Program illustrating Daemon Threads.
d. Write a JAVA program Producer Consumer Problem.

Exercise – 8

a. Write a JAVA program that import and use the user defined packages.
b. Without writing any code, build a GUI that display text in label and image in an ImageView (use JavaFX).
c. Build a Tip Calculator app using several JavaFX components and learn how to respond to user interactions with the GUI.

Exercise-9:

a. Implement the programs using List Interface and its implemented classes.
b. Implement the programs using Set Interface and its implemented classes.
c. Implement the programs using Map Interface and its implemented classes.

Exercise - 1:

a. Write a JAVA program to display default value of all primitive data type of JAVA

```
class DefaultValues {
   byte b;
   short s;
   int i;
   long l;
   float f;
   double d;
   char c;
   boolean bool;

   void displayDefaults() {
      System.out.println("Default Values of Primitive Data Types:");
      System.out.println("Byte: " + b);
      System.out.println("Short: " + s);
      System.out.println("Int: " + i);
      System.out.println("Long: " + l);
      System.out.println("Float: " + f);
      System.out.println("Double: " + d);
      System.out.println("Char: [" + c + "] (empty character)");
      System.out.println("Boolean: " + bool);
   }

   public static void main(String[] args) {
      DefaultValues obj = new DefaultValues();
      obj.displayDefaults();
   }
}
```

Expected Output:

```
Default Values of Primitive Data Types:
Byte: 0
Short: 0
Int: 0
Long: 0
Float: 0.0
Double: 0.0
Char: [] (empty character)
Boolean: false
```

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING * MYLAVARAM * TRUTH ALWAYS TRIUMPHS HARD WORK PAYS Estd: 1998

b. Write a java program that display the roots of a quadratic equation ax2+bx=0. Calculate the discriminate D and basing on value of D, describe the nature of root.

```
import java.util.Scanner;

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

      // Input coefficients
      System.out.print("Enter coefficient a: ");
      double a = sc.nextDouble();

      System.out.print("Enter coefficient b: ");
      double b = sc.nextDouble();

      System.out.print("Enter coefficient c: ");
      double c = sc.nextDouble();

      // Calculate discriminant
      double D = b * b - 4 * a * c;
      System.out.println("Discriminant (D) = " + D);

      // Determine nature of roots
      if (D > 0) {
         System.out.println("Roots are real and distinct.");
         double root1 = (-b + Math.sqrt(D)) / (2 * a);
         double root2 = (-b - Math.sqrt(D)) / (2 * a);
         System.out.println("Root 1: " + root1);
         System.out.println("Root 2: " + root2);
      } else if (D == 0) {
         System.out.println("Roots are real and equal.");
         double root = -b / (2 * a);
         System.out.println("Root: " + root);
      } else {
         System.out.println("Roots are complex and imaginary.");
         double realPart = -b / (2 * a);
         double imaginaryPart = Math.sqrt(-D) / (2 * a);
         System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
         System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
      }

      sc.close();
   }
}
```

Example Output:

```
Enter coefficient a: 1
Enter coefficient b: -7
Enter coefficient c: 10
Discriminant (D) = 9.0
Roots are real and distinct.
Root 1: 5.0
Root 2: 2.0
```

Exercise - 2

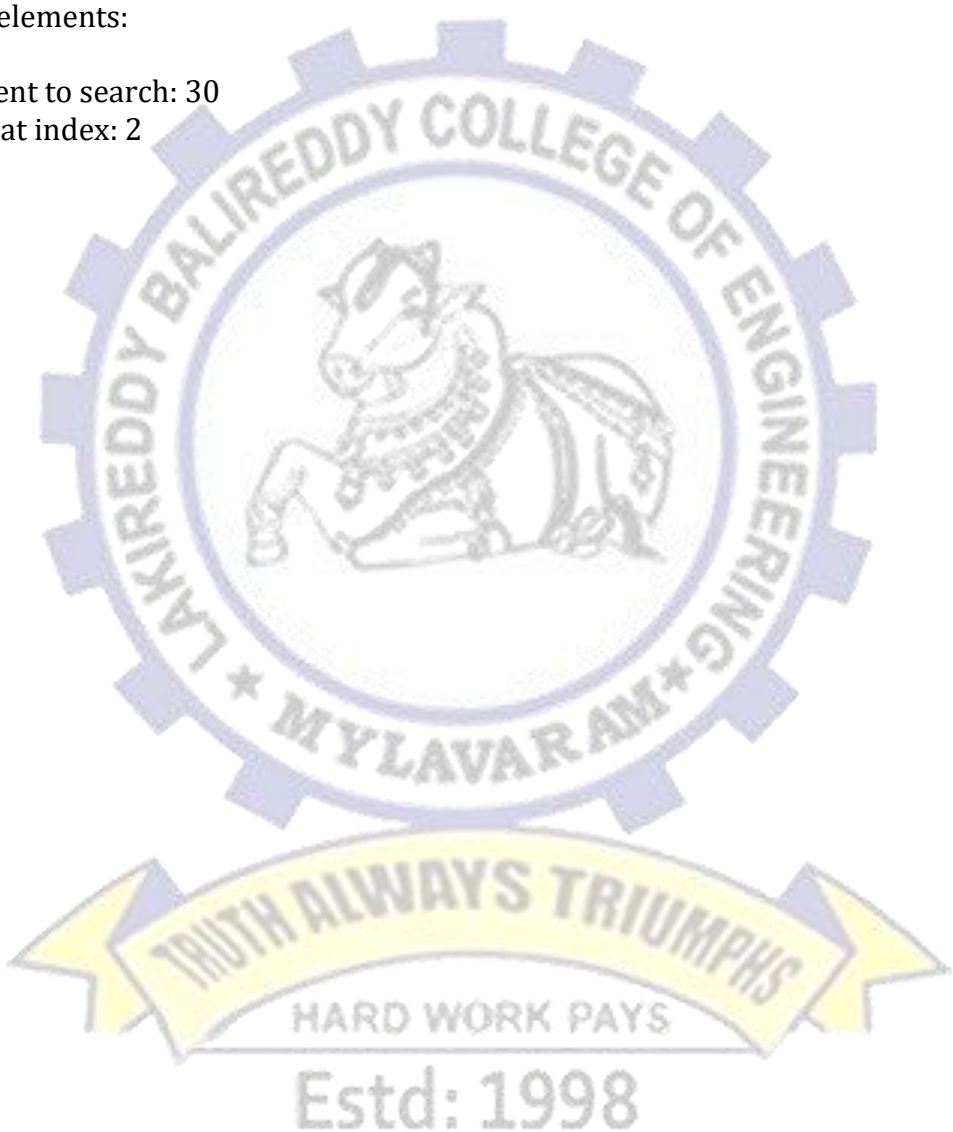
a. Write a JAVA program to search for an element in a given list of elements using binary search mechanism.

```
import java.util.Scanner;

class BinarySearch {
  // Method to perform binary search
  public static int binarySearch(int[] arr, int key) {
    int left = 0, right = arr.length - 1;

    while (left <= right) {
      int mid = left + (right - left) / 2;

      // Check if key is at mid
      if (arr[mid] == key) {
        return mid; // Return index of the key
      }
      // If key is smaller, search in left half
      else if (arr[mid] > key) {
        right = mid - 1;
      }
      // If key is larger, search in right half
      else {
        left = mid + 1;
      }
    }
    return -1; // Key not found
  }

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

    // Input size of array
    System.out.print("Enter the number of elements: ");
    int n = sc.nextInt();

    int[] arr = new int[n];

    // Input sorted array elements
    System.out.println("Enter " + n + " sorted elements:");
    for (int i = 0; i < n; i++) {
      arr[i] = sc.nextInt();
    }

    // Input element to search
    System.out.print("Enter the element to search: ");
    int key = sc.nextInt();

    // Perform binary search
    int result = binarySearch(arr, key);

    // Display result
    if (result != -1) {
```

```
            System.out.println("Element found at index: " + result);
        } else {
            System.out.println("Element not found in the list.");
        }

        sc.close();
    }
}
```

Expected Output:

```
Enter the number of elements: 5
Enter 5 sorted elements:
10 20 30 40 50
Enter the element to search: 30
Element found at index: 2
```



b. Write a JAVA program to sort for an element in a given list of elements using bubble sort.

```
import java.util.Scanner;
class BubbleSort {
   // Method to perform Bubble Sort
   public static void bubbleSort(int[] arr) {
      int n = arr.length;
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
               // Swap arr[j] and arr[j+1]
               int temp = arr[j];
               arr[j] = arr[j + 1];
               arr[j + 1] = temp;
            }
         }
      }
   }

   // Method to display the array
   public static void displayArray(int[] arr) {
      for (int num : arr) {
         System.out.print(num + " ");
      }
      System.out.println();
   }

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

      // Input size of array
      System.out.print("Enter the number of elements: ");
      int n = sc.nextInt();

      int[] arr = new int[n];

      // Input array elements
      System.out.println("Enter " + n + " elements:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
      // Sorting the array
      System.out.println("Array before sorting:");
      displayArray(arr);

      bubbleSort(arr);

      // Displaying sorted array
      System.out.println("Array after sorting:");
      displayArray(arr);
      sc.close();
   }
}
```

Expected Output:

Enter the number of elements: 5
Enter 5 elements:
64 34 25 12 22
Array before sorting:
64 34 25 12 22
Array after sorting:
12 22 25 34 64

c. Write a JAVA program using String Buffer to delete, remove character.

```
import java.util.Scanner;
class StringBufferExample {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      // Input string
      System.out.print("Enter a string: ");
      StringBuffer sb = new StringBuffer(sc.nextLine());
      // Display original string
      System.out.println("Original String: " + sb);
      // Deleting a substring
      System.out.print("Enter start and end index to delete substring: ");
      int start = sc.nextInt();
      int end = sc.nextInt();
      sb.delete(start, end);
      System.out.println("String after deletion: " + sb);
      // Removing a character
      System.out.print("Enter index to delete a character: ");
      int index = sc.nextInt();
      sb.deleteCharAt(index);
      System.out.println("String after removing character: " + sb);
      sc.close();
   }
}
```

Expected Output:

```
Enter a string: Hello World
Original String: Hello World
Enter start and end index to delete substring: 6 11
String after deletion: Hello
Enter index to delete a character: 2
String after removing character: Helo
```

Exercise - 3

a. Write a JAVA program to implement class mechanism. Create a class, methods and invoke them inside main method.

```
// Define a class
class Student {
   // Data members (instance variables)
   String name;
   int age;

   // Method to set student details
   void setDetails(String n, int a) {
      name = n;
      age = a;
   }

   // Method to display student details
   void displayDetails() {
      System.out.println("Student Name: " + name);
      System.out.println("Student Age: " + age);
   }
}

public class ClassMechanismExample {
   public static void main(String[] args) {
      // Creating an object of Student class
      Student s1 = new Student();

      // Calling method to set details
      s1.setDetails("Alice", 20);

      // Calling method to display details
      System.out.println("Student Information:");
      s1.displayDetails();
   }
}
```

Expected Output:

```
Student Information:
Student Name: Alice
Student Age: 20
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING
MYLAVARAM
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

b. Write a JAVA program implements method overloading.

```
class MathOperations {
   // Method to add two integers
   int add(int a, int b) {
      return a + b;
   }

   // Method to add three integers
   int add(int a, int b, int c) {
      return a + b + c;
   }

   // Method to add two double numbers
   double add(double a, double b) {
      return a + b;
   }
}

public class MethodOverloadingExample {
   public static void main(String[] args) {
      MathOperations obj = new MathOperations();

      // Calling overloaded methods
      System.out.println("Sum of two integers: " + obj.add(5, 10));
      System.out.println("Sum of three integers: " + obj.add(5, 10, 15));
      System.out.println("Sum of two double values: " + obj.add(5.5, 2.5));
   }
}
```

Expected Output:

```
Sum of two integers: 15
Sum of three integers: 30
Sum of two double values: 8.0
```

LAKIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

c. Write a JAVA program to implement constructor.

```
class Student {
   String name;
   int age;

   // Constructor to initialize values
   Student(String n, int a) {
      name = n;
      age = a;
   }

   // Method to display student details
   void display() {
      System.out.println("Student Name: " + name);
      System.out.println("Student Age: " + age);
   }
}

public class ConstructorExample {
   public static void main(String[] args) {
      // Creating an object of Student class (Constructor is called automatically)
      Student s1 = new Student("Alice", 20);

      // Display student details
      s1.display();
   }
}
```

Expected Output:

```
Student Name: Alice
Student Age: 20
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

d. Write a JAVA program to implement constructor overloading.

```
class Student {
   String name;
   int age;

   // Constructor with no parameters (Default Constructor)
   Student() {
      name = "Unknown";
      age = 0;
   }

   // Constructor with one parameter
   Student(String n) {
      name = n;
      age = 18; // Default age
   }

   // Constructor with two parameters
   Student(String n, int a) {
      name = n;
      age = a;
   }

   // Method to display student details
   void display() {
      System.out.println("Student Name: " + name);
      System.out.println("Student Age: " + age);
   }
}

public class ConstructorOverloadingExample {
   public static void main(String[] args) {
      // Creating objects using different constructors
      Student s1 = new Student();
      Student s2 = new Student("Alice");
      Student s3 = new Student("Bob", 21);

      // Displaying student details
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM * AUTONOMOUS TRIUMPHS HARD WORK PAYS Estd: 1998

```
        System.out.println("Student 1:");
        s1.display();

        System.out.println("\nStudent 2:");
        s2.display();

        System.out.println("\nStudent 3:");
        s3.display();
    }
}
```

Expected Output:

Student 1:
Student Name: Unknown
Student Age: 0

Student 2:
Student Name: Alice
Student Age: 18

Student 3:
Student Name: Bob
Student Age: 21

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING
MYLAVARAM
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

Exercise - 4

a. Write a JAVA program to implement Single Inheritance.

```
// Parent class
class Animal {
   void eat() {
      System.out.println("This animal eats food.");
   }
}

// Child class inheriting from Animal
class Dog extends Animal {
   void bark() {
      System.out.println("The dog barks.");
   }
}

public class SingleInheritanceExample {
   public static void main(String[] args) {
      // Creating an object of Dog class
      Dog myDog = new Dog();

      // Calling methods from both parent and child classes
      myDog.eat();  // Inherited method
      myDog.bark(); // Child class method
   }
}
```

Expected Output:

```
This animal eats food.
The dog barks.
```

LAKIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

b. Write a JAVA program to implement multilevel Inheritance.

```
// Base class (Parent)
class Animal {
   void eat() {
      System.out.println("This animal eats food.");
   }
}

// Derived class (Child of Animal)
class Mammal extends Animal {
   void walk() {
      System.out.println("Mammals can walk.");
   }
}

// Derived class (Child of Mammal)
class Dog extends Mammal {
   void bark() {
      System.out.println("The dog barks.");
   }
}

public class MultilevelInheritanceExample {
   public static void main(String[] args) {
      // Creating an object of Dog class
      Dog myDog = new Dog();

      // Calling methods from all levels of inheritance
      myDog.eat();  // Inherited from Animal
      myDog.walk(); // Inherited from Mammal
      myDog.bark(); // Defined in Dog
   }
}
```

Expected Output:

```
This animal eats food.
Mammals can walk.
The dog barks.
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING
MYLAVARAM
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

c. Write a JAVA program for abstract class to find areas of different shape.

```
// Abstract class
abstract class Shape {
   // Abstract method to calculate area
   abstract void calculateArea();
}

// Subclass for Circle
class Circle extends Shape {
   double radius;

   // Constructor
   Circle(double r) {
      radius = r;
   }

   // Implementation of abstract method
   void calculateArea() {
      double area = Math.PI * radius * radius;
      System.out.println("Area of Circle: " + area);
   }
}

// Subclass for Rectangle
class Rectangle extends Shape {
   double length, width;

   // Constructor
   Rectangle(double l, double w) {
      length = l;
      width = w;
   }

   // Implementation of abstract method
   void calculateArea() {
      double area = length * width;
      System.out.println("Area of Rectangle: " + area);
   }
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM * TRUTH ALWAYS TRIUMPHS HARD WORK PAYS Estd: 1998

```
}

// Subclass for Triangle
class Triangle extends Shape {
   double base, height;

   // Constructor
   Triangle(double b, double h) {
      base = b;
      height = h;
   }

   // Implementation of abstract method
   void calculateArea() {
      double area = 0.5 * base * height;
      System.out.println("Area of Triangle: " + area);
   }
}

public class AbstractClassExample {
   public static void main(String[] args) {
      // Creating objects of different shapes
      Shape circle = new Circle(5);
      Shape rectangle = new Rectangle(4, 6);
      Shape triangle = new Triangle(3, 8);

      // Calculating areas
      circle.calculateArea();
      rectangle.calculateArea();
      triangle.calculateArea();
   }
}
```

Expected Output:

Area of Circle: 78.53981633974483
Area of Rectangle: 24.0
Area of Triangle: 12.0

Exercise - 5

a. Write a JAVA program give example for "super" keyword.

```
// Parent class
class Animal {
   String name = "Animal";

   // Constructor of parent class
   Animal() {
      System.out.println("Animal constructor is called.");
   }

   // Method in parent class
   void makeSound() {
      System.out.println("Animals make different sounds.");
   }
}
// Child class
class Dog extends Animal {
   String name = "Dog";

   // Constructor of child class
   Dog() {
      // Calling parent class constructor using super()
      super();
      System.out.println("Dog constructor is called.");
   }

   // Method in child class
   void display() {
      System.out.println("Child class variable: " + name);
      System.out.println("Parent class variable: " + super.name); // Access parent class variable

      super.makeSound(); // Call parent class method
   }
}
public class SuperKeywordExample {
   public static void main(String[] args) {
      // Creating an object of Dog class
      Dog myDog = new Dog();

      // Calling method to demonstrate super keyword usage
      myDog.display();
   }
}
```

Expected Output:

```
Animal constructor is called.
Dog constructor is called.
Child class variable: Dog
Parent class variable: Animal
Animals make different sounds.
```

b. Write a JAVA program to implement Interface. What kind of Inheritance can be achieved?

```
// Interface definition
interface Shape {
   // Abstract method (no implementation)
   void calculateArea();
}

// Class implementing the interface
class Circle implements Shape {
   double radius;

   // Constructor
   Circle(double r) {
      radius = r;
   }

   // Implementing interface method
   public void calculateArea() {
      double area = Math.PI * radius * radius;
      System.out.println("Area of Circle: " + area);
   }
}

// Another class implementing the same interface
class Rectangle implements Shape {
   double length, width;

   // Constructor
   Rectangle(double l, double w) {
      length = l;
      width = w;
   }

   // Implementing interface method
   public void calculateArea() {
      double area = length * width;
      System.out.println("Area of Rectangle: " + area);
   }
}

public class InterfaceExample {
   public static void main(String[] args) {
      // Creating objects of different shapes
      Shape circle = new Circle(5);
      Shape rectangle = new Rectangle(4, 6);

      // Calculating areas
      circle.calculateArea();
      rectangle.calculateArea();
   }
}
```

Expected Output:

Area of Circle: 78.53981633974483
Area of Rectangle: 24.0

```
// First interface
interface Animal {
   void eat();
}

// Second interface
interface Bird {
   void fly();
}

// Class implementing multiple interfaces
class Bat implements Animal, Bird {
   public void eat() {
      System.out.println("Bat eats insects and fruits.");
   }

   public void fly() {
      System.out.println("Bat can fly.");
   }
}

public class MultipleInheritanceExample {
   public static void main(String[] args) {
      // Creating an object of Bat class
      Bat myBat = new Bat();

      // Calling methods from both interfaces
      myBat.eat();
      myBat.fly();
   }
}
```

Expected Output:

Bat eats insects and fruits.
Bat can fly.

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING
MYLAVARAM
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

c. Write a JAVA program that implements Runtime polymorphism.

```
// Parent class
class Animal {
   // Method to be overridden
   void makeSound() {
      System.out.println("Animals make different sounds.");
   }
}

// Child class 1
class Dog extends Animal {
   // Overriding makeSound() method
   void makeSound() {
      System.out.println("Dog barks.");
   }
}
// Child class 2
class Cat extends Animal {
   // Overriding makeSound() method
   void makeSound() {
      System.out.println("Cat meows.");
   }
}
public class RuntimePolymorphismExample {
   public static void main(String[] args) {
      // Parent class reference pointing to Child class objects (Upcasting)
      Animal myAnimal;
      myAnimal = new Dog();
      myAnimal.makeSound();  // Calls Dog's version of makeSound()
      myAnimal = new Cat();
      myAnimal.makeSound();  // Calls Cat's version of makeSound()
   }
}
```

Expected Output:

```
Dog barks.
Cat meows.
```

Exercise - 6

a. Write a JAVA program that describes exception handling mechanism.

```
public class ExceptionHandlingExample {
   public static void main(String[] args) {
      try {
         // Code that may cause an exception
         int a = 10, b = 0;
         int result = a / b; // This will cause ArithmeticException
         System.out.println("Result: " + result);
      } catch (ArithmeticException e) {
         // Handling the exception
         System.out.println("Exception caught: Division by zero is not allowed.");
      } finally {
         // Finally block always executes
         System.out.println("Execution completed.");
      }
   }
}
```

Expected Output:

```
Exception caught: Division by zero is not allowed.
Execution completed.
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

b. Write a JAVA program Illustrating Multiple catch clauses.

```
public class MultipleCatchExample {
   public static void main(String[] args) {
      try {
         int arr[] = {10, 20, 30};
         int a = 10, b = 0;

         // This will cause ArithmeticException
         int result = a / b;
         System.out.println("Result: " + result);

         // This will cause ArrayIndexOutOfBoundsException
         System.out.println("Array Element: " + arr[5]);

      } catch (ArithmeticException e) {
         System.out.println("Exception caught: Division by zero is not allowed.");
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Exception caught: Array index is out of bounds.");
      } catch (Exception e) {
         System.out.println("Exception caught: General exception occurred.");
      } finally {
         System.out.println("Execution completed.");
      }
   }
}
```

Expected Output:

```
Exception caught: Division by zero is not allowed.
Execution completed.
```

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

c. Write a JAVA program for creation of Java Built-in Exceptions.

```
public class BuiltInExceptionsExample {
   public static void main(String[] args) {
      try {
         // 1. ArithmeticException (Division by zero)
         int a = 10, b = 0;
         int result = a / b; // Causes ArithmeticException
         System.out.println("Result: " + result);

      } catch (ArithmeticException e) {
         System.out.println("Exception caught: " + e);
      }

      try {
         // 2. ArrayIndexOutOfBoundsException (Invalid index access)
         int arr[] = {1, 2, 3};
         System.out.println("Array Element: " + arr[5]); // Causes ArrayIndexOutOfBoundsException

      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Exception caught: " + e);
      }

      try {
         // 3. NullPointerException (Accessing method on null object)
         String str = null;
         System.out.println("String length: " + str.length()); // Causes NullPointerException

      } catch (NullPointerException e) {
         System.out.println("Exception caught: " + e);
      }

      System.out.println("Program execution continued...");
   }
}
```

Expected Output:

```
Exception caught: java.lang.ArithmeticException: / by zero
Exception caught: java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3
Exception caught: java.lang.NullPointerException: Cannot invoke "String.length()" because "str" is null
Program execution continued...
```

d. Write a JAVA program for creation of User Defined Exception.

```
// Creating a user-defined exception by extending Exception class
class InvalidAgeException extends Exception {
   // Constructor to display custom message
   InvalidAgeException(String message) {
      super(message);
   }
}

public class UserDefinedExceptionExample {
   // Method to check age
   static void validateAge(int age) throws InvalidAgeException {
      if (age < 18) {
         throw new InvalidAgeException("Age must be 18 or above to vote.");
      } else {
         System.out.println("You are eligible to vote.");
      }
   }

   public static void main(String[] args) {
      try {
         validateAge(16); // This will throw the custom exception
      } catch (InvalidAgeException e) {
         System.out.println("Exception caught: " + e.getMessage());
      }

      System.out.println("Program execution continued...");
   }
}
```

Expected Output:

Exception caught: Age must be 18 or above to vote.
Program execution continued...

Exercise - 7

a. Write a JAVA program that creates threads by extending Thread class. First thread display "Good Morning "every 1 sec, the second thread displays "Hello "every 2 seconds and the third display "Welcome" every 3 seconds (Repeat the same by implementing Runnable).

```
// Thread 1: Prints "Good Morning" every 1 second
class GoodMorningThread extends Thread {
   public void run() {
      try {
         while (true) {
            System.out.println("Good Morning");
            Thread.sleep(1000); // 1 second delay
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}
// Thread 2: Prints "Hello" every 2 seconds
class HelloThread extends Thread {
   public void run() {
      try {
         while (true) {
            System.out.println("Hello");
            Thread.sleep(2000); // 2 seconds delay
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}

// Thread 3: Prints "Welcome" every 3 seconds
class WelcomeThread extends Thread {
   public void run() {
      try {
         while (true) {
            System.out.println("Welcome");
            Thread.sleep(3000); // 3 seconds delay
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}

public class ThreadExample {
   public static void main(String[] args) {
      // Creating and starting threads
      new GoodMorningThread().start();
      new HelloThread().start();
      new WelcomeThread().start();
   }
}
```

Expected Output:

```
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
...
```

```
// Thread 1: Prints "Good Morning" every 1 second
class GoodMorningRunnable implements Runnable {
   public void run() {
      try {
         while (true) {
            System.out.println("Good Morning");
            Thread.sleep(1000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}

// Thread 2: Prints "Hello" every 2 seconds
class HelloRunnable implements Runnable {
   public void run() {
      try {
         while (true) {
            System.out.println("Hello");
            Thread.sleep(2000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}

// Thread 3: Prints "Welcome" every 3 seconds
class WelcomeRunnable implements Runnable {
   public void run() {
      try {
         while (true) {
            System.out.println("Welcome");
            Thread.sleep(3000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}

public class RunnableExample {
```

```
    public static void main(String[] args) {
        // Creating thread objects using Runnable
        Thread t1 = new Thread(new GoodMorningRunnable());
        Thread t2 = new Thread(new HelloRunnable());
        Thread t3 = new Thread(new WelcomeRunnable());

        // Starting threads
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Expected Output:

```
Good Morning
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
...
```

b. Write a program illustrating is Alive and join ().

```
class MyThread extends Thread {
   public void run() {
      try {
         for (int i = 1; i <= 3; i++) {
            System.out.println(Thread.currentThread().getName() + " - Count: " + i);
            Thread.sleep(1000); // 1 second delay
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}
public class ThreadJoinIsAliveExample {
   public static void main(String[] args) {
      // Creating threads
      MyThread t1 = new MyThread();
      MyThread t2 = new MyThread();
      // Starting threads
      t1.start();
      t2.start();
      // Checking if threads are alive
      System.out.println("Thread t1 is alive: " + t1.isAlive());
      System.out.println("Thread t2 is alive: " + t2.isAlive());
      try {
         // Main thread waits for t1 and t2 to finish
         t1.join();
         t2.join();
      } catch (InterruptedException e) {
         System.out.println("Main thread interrupted: " + e);
      }

      // After threads finish
      System.out.println("Thread t1 is alive: " + t1.isAlive());
      System.out.println("Thread t2 is alive: " + t2.isAlive());
      System.out.println("Main thread execution completed.");
   }
}
```

Expected Output:

```
Thread t1 is alive: true
Thread t2 is alive: true
Thread-0 - Count: 1
Thread-1 - Count: 1
Thread-0 - Count: 2
Thread-1 - Count: 2
Thread-0 - Count: 3
Thread-1 - Count: 3
Thread t1 is alive: false
Thread t2 is alive: false
Main thread execution completed.
```

c. Write a Program illustrating Daemon Threads.

```
class DaemonThreadExample extends Thread {
   public void run() {
      // Checking if the thread is daemon or not
      if (isDaemon()) {
         System.out.println(Thread.currentThread().getName() + " is a Daemon Thread.");
      } else {
         System.out.println(Thread.currentThread().getName() + " is a User Thread.");
      }

      try {
         for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " - Count: " + i);
            Thread.sleep(1000); // Sleep for 1 second
         }
      } catch (InterruptedException e) {
         System.out.println("Thread Interrupted: " + e);
      }
   }
}
public class DaemonThreadDemo {
   public static void main(String[] args) {
      // Creating daemon and user threads
      DaemonThreadExample t1 = new DaemonThreadExample(); // User thread
      DaemonThreadExample t2 = new DaemonThreadExample(); // Daemon thread

      t2.setDaemon(true); // Setting thread t2 as daemon

      // Starting both threads
      t1.start();
      t2.start();

      // Main thread sleeps for 3 seconds
      try {
         Thread.sleep(3000);
      } catch (InterruptedException e) {
         System.out.println("Main thread interrupted.");
      }
      System.out.println("Main thread finished execution.");
   }
}
```

Expected Output:

```
Thread-0 is a User Thread.
Thread-1 is a Daemon Thread.
Thread-0 - Count: 1
Thread-1 - Count: 1
Thread-0 - Count: 2
Thread-1 - Count: 2
Thread-0 - Count: 3
Thread-1 - Count: 3
Main thread finished execution.
```

d. Write a JAVA program Producer Consumer Problem.

```
import java.util.LinkedList;
class SharedBuffer {
   private LinkedList<Integer> buffer = new LinkedList<>();
   private int capacity = 5; // Max buffer size

   // Producer method
   public synchronized void produce(int item) throws InterruptedException {
      while (buffer.size() == capacity) {
         System.out.println("Buffer is full. Producer waiting...");
         wait(); // Wait if buffer is full
      }

      buffer.add(item);
      System.out.println("Produced: " + item);
      notify(); // Notify consumer
   }

   // Consumer method
   public synchronized int consume() throws InterruptedException {
      while (buffer.isEmpty()) {
         System.out.println("Buffer is empty. Consumer waiting...");
         wait(); // Wait if buffer is empty
      }

      int item = buffer.removeFirst();
      System.out.println("Consumed: " + item);
      notify(); // Notify producer
      return item;
   }
}

// Producer thread
class Producer extends Thread {
   private SharedBuffer buffer;

   Producer(SharedBuffer buffer) {
      this.buffer = buffer;
```

```
    }

    public void run() {
        int item = 1;
        try {
            while (true) {
                buffer.produce(item++);
                Thread.sleep(1000); // Simulate delay
            }
        } catch (InterruptedException e) {
            System.out.println("Producer interrupted.");
        }
    }
}

// Consumer thread
class Consumer extends Thread {
    private SharedBuffer buffer;

    Consumer(SharedBuffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        try {
            while (true) {
                buffer.consume();
                Thread.sleep(1500); // Simulate delay
            }
        } catch (InterruptedException e) {
            System.out.println("Consumer interrupted.");
        }
    }
}

// Main class
public class ProducerConsumerDemo {
    public static void main(String[] args) {
```

```
        SharedBuffer buffer = new SharedBuffer();

        // Creating Producer and Consumer threads
        Producer producer = new Producer(buffer);
        Consumer consumer = new Consumer(buffer);

        // Starting threads
        producer.start();
        consumer.start();
    }
}
```

Expected Output:

```
Produced: 1
Consumed: 1
Produced: 2
Produced: 3
Consumed: 2
Produced: 4
Produced: 5
Consumed: 3
Produced: 6
Buffer is full. Producer waiting...
Consumed: 4
Produced: 7
...
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING
MYLAVARAM
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

Exercise – 8

a. Write a JAVA program that import and use the user defined packages.

```
// Declare the package
package mypackage;
public class Calculator {
   // Method for addition
   public int add(int a, int b) {
      return a + b;
   }
   // Method for subtraction
   public int subtract(int a, int b) {
      return a - b;
   }

   // Method for multiplication
   public int multiply(int a, int b) {
      return a * b;
   }
   // Method for division
   public double divide(int a, int b) {
      if (b != 0) {
         return (double) a / b;
      } else {
         System.out.println("Error: Division by zero!");
         return 0;
      }
   }
}

// Import the user-defined package
import mypackage.Calculator;
public class MainProgram {
   public static void main(String[] args) {
      // Creating an object of the Calculator class
      Calculator calc = new Calculator();
      // Using the methods
      System.out.println("Addition: " + calc.add(10, 5));
      System.out.println("Subtraction: " + calc.subtract(10, 5));
      System.out.println("Multiplication: " + calc.multiply(10, 5));
      System.out.println("Division: " + calc.divide(10, 5));
   }
}
```

Expected Output:

```
javac MainProgram.java
java MainProgram

Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
```

b. Without writing any code, build a GUI that display text in label and image in an ImageView (use JavaFX).

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class JavaFXLabelImageView extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Create a Label
    Label label = new Label("Hello, JavaFX!");

    // Load an image (Ensure the image file exists in the project directory)
    Image image = new Image("file:example.jpg"); // Change the path accordingly
    ImageView imageView = new ImageView(image);

    // Set ImageView properties
    imageView.setFitWidth(300); // Set width
    imageView.setPreserveRatio(true); // Maintain aspect ratio

    // Create a VBox layout and add Label & ImageView
    VBox root = new VBox(10, label, imageView);
    root.setStyle("-fx-alignment: center; -fx-padding: 20;");

    // Set the Scene and Stage
    Scene scene = new Scene(root, 400, 400);
    primaryStage.setTitle("JavaFX Label and ImageView");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

  public static void main(String[] args) {
    launch(args);
  }
}
```

Expected Output:

A **JavaFX window** opens with:

1. A **Label** at the top displaying **"Hello, JavaFX!"**.
2. An **ImageView** below the label showing an image (example.jpg).
3. The **VBox layout** centers the elements and applies padding.

c. Build a Tip Calculator app using several JavaFX components and learn how to respond to user interactions with the GUI.

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class TipCalculatorApp extends Application {

  @Override
  public void start(Stage primaryStage) {
    // Label and TextField for Bill Amount
    Label billLabel = new Label("Enter Bill Amount:");
    TextField billInput = new TextField();
    billInput.setPromptText("Enter amount");

    // ChoiceBox for Tip Percentage
    Label tipLabel = new Label("Select Tip Percentage:");
    ChoiceBox<Integer> tipChoiceBox = new ChoiceBox<>();
    tipChoiceBox.getItems().addAll(10, 15, 20);
    tipChoiceBox.setValue(15); // Default tip percentage

    // Button to calculate tip
    Button calculateButton = new Button("Calculate Tip");

    // Labels for displaying results
    Label tipAmountLabel = new Label("Tip Amount: $0.00");
    Label totalAmountLabel = new Label("Total Amount: $0.00");

    // Event Handling - Calculate tip on button click
    calculateButton.setOnAction(e -> {
      try {
        double billAmount = Double.parseDouble(billInput.getText());
        int tipPercentage = tipChoiceBox.getValue();
        double tipAmount = billAmount * tipPercentage / 100;
        double totalAmount = billAmount + tipAmount;
```

```
            // Update Labels
            tipAmountLabel.setText(String.format("Tip Amount: $%.2f", tipAmount));
            totalAmountLabel.setText(String.format("Total Amount: $%.2f", totalAmount));
        } catch (NumberFormatException ex) {
            tipAmountLabel.setText("Invalid input! Enter a valid number.");
            totalAmountLabel.setText("");
        }
    });

    // Layout using VBox
    VBox root = new VBox(10, billLabel, billInput, tipLabel, tipChoiceBox, calculateButton, tipAmountLabel, totalAmountLabel);
    root.setPadding(new Insets(20));
    root.setStyle("-fx-alignment: center;");

    // Scene and Stage Setup
    Scene scene = new Scene(root, 300, 250);
    primaryStage.setTitle("Tip Calculator");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

  public static void main(String[] args) {
    launch(args);
  }
}
```

Expected Output & Functionality:

1. **User enters the bill amount** in the text field.
2. **User selects a tip percentage** (10%, 15%, 20%) from the **ChoiceBox**.
3. **User clicks "Calculate Tip"**, and:
    - **Tip Amount** is displayed.
    - **Total Amount** (bill + tip) is displayed.

Exercise-9:

a. Implement the programs using List Interface and its implemented classes.

```
import java.util.ArrayList;

public class ArrayListExample {
   public static void main(String[] args) {
      // Creating an ArrayList
      ArrayList<String> fruits = new ArrayList<>();

      // Adding elements
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Mango");

      // Accessing elements
      System.out.println("Fruits List: " + fruits);

      // Removing an element
      fruits.remove("Banana");
      System.out.println("After Removing Banana: " + fruits);

      // Iterating using for-each loop
      System.out.println("Iterating over elements:");
      for (String fruit : fruits) {
         System.out.println(fruit);
      }
   }
}
```

Expected Output:

```
Fruits List: [Apple, Banana, Mango]
After Removing Banana: [Apple, Mango]
Iterating over elements:
Apple
Mango
```

LAKIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

```
import java.util.LinkedList;

public class LinkedListExample {
   public static void main(String[] args) {
      // Creating a LinkedList
      LinkedList<Integer> numbers = new LinkedList<>();

      // Adding elements
      numbers.add(10);
      numbers.add(20);
      numbers.addFirst(5); // Adding at the beginning
      numbers.addLast(30); // Adding at the end

      // Displaying elements
      System.out.println("LinkedList: " + numbers);

      // Removing elements
      numbers.removeFirst();
      numbers.removeLast();

      // Iterating using forEach method
      System.out.println("After Removing First and Last Elements:");
      numbers.forEach(System.out::println);
   }
}
```

Expected Output:

```
LinkedList: [5, 10, 20, 30]
After Removing First and Last Elements:
10
20
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

```
import java.util.Vector;

public class VectorExample {
   public static void main(String[] args) {
      // Creating a Vector
      Vector<String> cities = new Vector<>();

      // Adding elements
      cities.add("New York");
      cities.add("London");
      cities.add("Tokyo");

      // Display elements
      System.out.println("Cities: " + cities);

      // Checking size and capacity
      System.out.println("Size: " + cities.size());
      System.out.println("Capacity: " + cities.capacity());

      // Removing an element
      cities.remove("London");

      // Iterating using a traditional for-loop
      System.out.println("After Removing London:");
      for (int i = 0; i < cities.size(); i++) {
         System.out.println(cities.get(i));
      }
   }
}
```

Expected Output:

```
Cities: [New York, London, Tokyo]
Size: 3
Capacity: 10
After Removing London:
New York
Tokyo
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

```
import java.util.Stack;

public class StackExample {
   public static void main(String[] args) {
      // Creating a Stack
      Stack<String> books = new Stack<>();

      // Pushing elements onto the stack
      books.push("Java Programming");
      books.push("Data Structures");
      books.push("Operating Systems");

      // Displaying stack
      System.out.println("Stack: " + books);

      // Popping an element
      System.out.println("Popped: " + books.pop());

      // Peeking at the top element
      System.out.println("Top Element: " + books.peek());

      // Checking if stack is empty
      System.out.println("Is Stack Empty? " + books.isEmpty());
   }
}
```

Expected Output:

```
Stack: [Java Programming, Data Structures, Operating Systems]
Popped: Operating Systems
Top Element: Data Structures
Is Stack Empty? False
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING
MYLAVARAM
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

b. Implement the programs using Set Interface and its implemented classes.

```
import java.util.HashSet;

public class HashSetExample {
   public static void main(String[] args) {
      // Creating a HashSet
      HashSet<String> colors = new HashSet<>();

      // Adding elements
      colors.add("Red");
      colors.add("Blue");
      colors.add("Green");
      colors.add("Yellow");
      colors.add("Blue"); // Duplicate element (ignored)

      // Displaying elements
      System.out.println("HashSet: " + colors);

      // Removing an element
      colors.remove("Green");

      // Checking if an element exists
      System.out.println("Contains Blue? " + colors.contains("Blue"));

      // Iterating using forEach loop
      System.out.println("Iterating through HashSet:");
      for (String color : colors) {
         System.out.println(color);
      }
   }
}
```

Expected Output:

```
HashSet: [Red, Blue, Yellow, Green]  (Order may vary)
Contains Blue? true
Iterating through HashSet:
Red
Blue
Yellow
```

LAKIREDDY COLLEGE OF ENGINEERING * MYLAVARAM * TRUTH ALWAYS TRIUMPHS HARD WORK PAYS Estd: 1998

```
import java.util.LinkedHashSet;

public class LinkedHashSetExample {
   public static void main(String[] args) {
      // Creating a LinkedHashSet
      LinkedHashSet<Integer> numbers = new LinkedHashSet<>();

      // Adding elements
      numbers.add(10);
      numbers.add(20);
      numbers.add(30);
      numbers.add(10); // Duplicate (ignored)

      // Displaying elements (preserves insertion order)
      System.out.println("LinkedHashSet: " + numbers);

      // Removing an element
      numbers.remove(20);

      // Iterating using forEach
      System.out.println("After Removal:");
      numbers.forEach(System.out::println);
   }
}
```

Expected Output:

```
LinkedHashSet: [10, 20, 30]
After Removal:
10
30
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

```
import java.util.TreeSet;

public class TreeSetExample {
   public static void main(String[] args) {
      // Creating a TreeSet
      TreeSet<Integer> sortedNumbers = new TreeSet<>();

      // Adding elements
      sortedNumbers.add(50);
      sortedNumbers.add(10);
      sortedNumbers.add(30);
      sortedNumbers.add(40);
      sortedNumbers.add(20);

      // Displaying sorted elements
      System.out.println("TreeSet (Sorted): " + sortedNumbers);

      // Checking first and last element
      System.out.println("First Element: " + sortedNumbers.first());
      System.out.println("Last Element: " + sortedNumbers.last());

      // Removing an element
      sortedNumbers.remove(30);

      // Iterating using for-each loop
      System.out.println("After Removal:");
      for (int num : sortedNumbers) {
         System.out.println(num);
      }
   }
}
```

Expected Output:

```
TreeSet (Sorted): [10, 20, 30, 40, 50]
First Element: 10
Last Element: 50
After Removal:
10
20
40
50
```

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

c. Implement the programs using Map Interface and its implemented classes.

```
import java.util.HashMap;
import java.util.Map;

public class HashMapExample {
   public static void main(String[] args) {
      // Creating a HashMap
      HashMap<Integer, String> students = new HashMap<>();

      // Adding key-value pairs
      students.put(101, "Alice");
      students.put(102, "Bob");
      students.put(103, "Charlie");
      students.put(104, "David");

      // Displaying the HashMap
      System.out.println("HashMap: " + students);

      // Accessing value using key
      System.out.println("Student with ID 102: " + students.get(102));

      // Removing an entry
      students.remove(103);

      // Iterating using forEach loop
      System.out.println("Iterating over HashMap:");
      for (Map.Entry<Integer, String> entry : students.entrySet()) {
         System.out.println("ID: " + entry.getKey() + ", Name: " + entry.getValue());
      }
   }
}
```

Expected Output:

```
HashMap: {101=Alice, 102=Bob, 103=Charlie, 104=David}
Student with ID 102: Bob
Iterating over HashMap:
ID: 101, Name: Alice
ID: 102, Name: Bob
ID: 104, Name: David
```

TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

```
import java.util.LinkedHashMap;
import java.util.Map;

public class LinkedHashMapExample {
   public static void main(String[] args) {
      // Creating a LinkedHashMap
      LinkedHashMap<Integer, String> countries = new LinkedHashMap<>();

      // Adding key-value pairs
      countries.put(1, "USA");
      countries.put(2, "India");
      countries.put(3, "UK");
      countries.put(4, "Canada");

      // Displaying the LinkedHashMap
      System.out.println("LinkedHashMap: " + countries);

      // Removing an entry
      countries.remove(3);

      // Iterating using forEach
      System.out.println("Iterating over LinkedHashMap:");
      countries.forEach((key, value) -> System.out.println("Key: " + key + ", Value: " + value));
   }
}
```

```
LinkedHashMap: {1=USA, 2=India, 3=UK, 4=Canada}
Iterating over LinkedHashMap:
Key: 1, Value: USA
Key: 2, Value: India
Key: 4, Value: Canada
```

LAKIREDDY BALIREDDY COLLEGE OF ENGINEERING * MYLAVARAM *
TRUTH ALWAYS TRIUMPHS
HARD WORK PAYS
Estd: 1998

```
import java.util.TreeMap;
import java.util.Map;

public class TreeMapExample {
   public static void main(String[] args) {
      // Creating a TreeMap
      TreeMap<Integer, String> employees = new TreeMap<>();

      // Adding key-value pairs
      employees.put(2003, "John");
      employees.put(2001, "Emma");
      employees.put(2002, "Michael");
      employees.put(2005, "Sophia");

      // Displaying the TreeMap (sorted by key)
      System.out.println("TreeMap (Sorted by Key): " + employees);

      // Accessing first and last key-value pairs
      System.out.println("First Entry: " + employees.firstEntry());
      System.out.println("Last Entry: " + employees.lastEntry());

      // Removing an entry
      employees.remove(2002);

      // Iterating using for-each loop
      System.out.println("After Removal:");
      for (Map.Entry<Integer, String> entry : employees.entrySet()) {
         System.out.println("ID: " + entry.getKey() + ", Name: " + entry.getValue());
      }
   }
}
```

Expected Output:

```
TreeMap (Sorted by Key): {2001=Emma, 2002=Michael, 2003=John, 2005=Sophia}
First Entry: 2001=Emma
Last Entry: 2005=Sophia
After Removal:
ID: 2001, Name: Emma
ID: 2003, Name: John
ID: 2005, Name: Sophia
```

********** **The End** **********