Objectives:

- To teach the students basics of JAVA programs and its execution.
- To teach the students the differences between C and Java programming.
- To make the students learn concepts like packages and interfaces.
- To make the student learn an object oriented way of solving problems using java.
- To make the students to write programs using multithreading concepts and handle exceptions concepts.

Programs:

P1: write a program to display Hello World message in console window.

Program:

```
class HelloWorld {
public static void main (String[] args) {
        System.out.println("Hello, World!");
        }
}
```

Output:

```
Hello, World!
```

P2: Write a program to perform arithmetic and bitwise operations in a single source program without object creation.

Program:

```
public class Operations {
         public static void main (String[] args) {
                int a = 10;
                int b = 5;

                // Arithmetic operations
                int sum = a + b;
                int difference = a - b;
                int product = a * b;
                int quotient = a / b;
                int remainder = a % b;

                System.out.println("Arithmetic Operations:");
                System.out.println("Sum: " + sum);
                System.out.println("Difference: " + difference);
                System.out.println("Product: " + product);
                System.out.println("Quotient: " + quotient);
                System.out.println("Remainder: " + remainder);

                // Bitwise operations
                int and = a & b;
                int or = a | b;
                int xor = a ^ b;
                int leftShift = a << 2;
                int rightShift = a >> 2;

                System.out.println("Bitwise Operations:");
                System.out.println("And: " + and);
                System.out.println("Or: " + or);
                System.out.println("Xor: " + xor);
                System.out.println("Left Shift: " + leftShift);
                System.out.println("Right Shift: " + rightShift);
                }
        }
```

Output:

```
Arithmetic Operations:
Sum: 15
Difference: 5
Product: 50
Quotient: 2
Remainder: 0
Bitwise Operations:
```

```
And: 0
Or: 15
Xor: 15
Left Shift: 40
Right Shift: 2
```

P3: Write a program to perform arithmetic and bitwise operations by creating individual methods and classes than create an object to execute the individual methods of each operation.

Program:

```
class ArithmeticOperations {
  int a;
  int b;

  public ArithmeticOperations(int a, int b) {
    this.a = a;
    this.b = b;
  }

  public int add() {
    return a + b;
  }

  public int subtract() {
    return a - b;
  }

  public int multiply() {
    return a * b;
  }

  public int divide() {
    return a / b;
  }
  public int modulo() {
    return a % b;
  }
}

class BitwiseOperations {
  int a;
  int b;

  public BitwiseOperations(int a, int b) {
    this.a = a;
    this.b = b;
  }

  public int and() {
    return a & b;
  }

  public int or() {
    return a | b;
  }

  public int xor() {
    return a ^ b;
  }

  public int not() {
    return ~a;
  }

  public int leftShift() {
    return a << b;
  }

  public int rightShift() {
    return a >> b;
  }
```

```
}

public class Main {
  public static void main(String[] args) {
    int a = 10;
    int b = 5;

    ArithmeticOperations arithmeticOperations = new ArithmeticOperations(a, b);
    BitwiseOperations bitwiseOperations = new BitwiseOperations(a, b);

    System.out.println("Arithmetic Operations:");
    System.out.println("Sum: " + arithmeticOperations.add());
    System.out.println("Difference: " + arithmeticOperations.subtract());
    System.out.println("Product: " + arithmeticOperations.multiply());
    System.out.println("Quotient: " + arithmeticOperations.divide());
    System.out.println("Remainder: " + arithmeticOperations.modulo());

    System.out.println("Bitwise Operations:");
    System.out.println("And: " + bitwiseOperations.and());
    System.out.println("Or: " + bitwiseOperations.or());
    System.out.println("Xor: " + bitwiseOperations.xor());
    System.out.println("Not: " + bitwiseOperations.not());
    System.out.println("Left Shift: " + bitwiseOperations.leftShift());
    System.out.println("Right Shift: " + bitwiseOperations.rightShift());
  }
}
```

Output:

```
Arithmetic Operations:
Sum: 15
Difference: 5
Product: 50
Quotient: 2
Remainder: 0
Bitwise Operations:
And: 0
Or: 15
```

```
Xor: 15
Left Shift: 320
Right Shift: 0
```

P4: Write a java program to display the employee details using Scanner class.

```
import java.util.Scanner;

public class Employee {
  String name;
  int age;
  String address;
  String phone;

  public Employee(String name, int age, String address, String phone) {
    this.name = name;
    this.age = age;
    this.address = address;
    this.phone = phone;
  }

  public void displayDetails() {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age);
    System.out.println("Address: " + address);
    System.out.println("Phone: " + phone);
  }

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

    System.out.println("Enter employee details:");
    System.out.print("Name: ");
    String name = scanner.nextLine();
    System.out.print("Age: ");
    int age = scanner.nextInt();
    System.out.print("Address: ");
    scanner.nextLine();
    String address = scanner.nextLine();
    System.out.print("Phone: ");
    String phone = scanner.nextLine();

    Employee employee = new Employee(name, age, address, phone);
    employee.displayDetails();

    scanner.close();
  }
}
```

Input:

```
Name:
Akshay
Age:
23
Address:
Post Limda, Waghodia, Gujarat 391760
Phone:
1111111111
```

Ouput:

```
Enter employee details:
Name: Akshay
Age: 23
Address: Post Limda, Waghodia, Gujarat 391760
Phone: 1111111111
```

P5: Write a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions?

```
import java.util.Scanner;
import java.lang.Math;
```

```
public class QuadraticEquation {
  private double a, b, c;

  public QuadraticEquation(double a, double b, double c) {
    this.a = a;
    this.b = b;
    this.c = c;
  }

  public double getDiscriminant() {
    return b * b - 4.0 * a * c;
  }

  public double getRoot1() {
    return (-b + Math.sqrt(getDiscriminant())) / (2.0 * a);
  }

  public double getRoot2() {
    return (-b - Math.sqrt(getDiscriminant())) / (2.0 * a);
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter a, b, and c:");
    double a = scanner.nextDouble();
    double b = scanner.nextDouble();
    double c = scanner.nextDouble();

    QuadraticEquation equation = new QuadraticEquation(a, b, c);
    double discriminant = equation.getDiscriminant();

    if (discriminant > 0) {
      System.out.println("The roots are real and different.");
      System.out.println("root1 = " + equation.getRoot1());
      System.out.println("root2 = " + equation.getRoot2());
    } else if (discriminant == 0) {
      System.out.println("The roots are real and same.");
      System.out.println("root = " + equation.getRoot1());
    } else {
      System.out.println("The roots are complex and different.");
    }
    scanner.close();
  }
}
```

Input:
```
Enter a, b, and c:
1.0 -3.0 2.0
```

Output:
```
The roots are real and different.
root1 = 2.0
root2 = 1.0
```

P6: The Fibonacci sequence is defined by the following rule. The first 2 values in the sequence are 1, 1. Every subsequent value is the sum of the 2 values preceding it. Write a Java program that uses both recursive and non-recursive functions to print the nth value of the Fibonacci sequence?

```
import java.util.Scanner;

public class Fibonacci {
  public static int recursiveFibonacci(int n) {
    if (n <= 1) {
      return n;
    } else {
      return recursiveFibonacci(n - 1) + recursiveFibonacci(n - 2);
    }
  }

  public static int nonRecursiveFibonacci(int n) {
    int f0 = 0;
    int f1 = 1;
    int fn = 1;
    for (int i = 2; i <= n; i++) {
      fn = f0 + f1;
      f0 = f1;
      f1 = fn;
    }

    return fn;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number: ");
    int n = scanner.nextInt();

    System.out.println("Recursive Fibonacci: " + recursiveFibonacci(n));
    System.out.println("Non-recursive Fibonacci: " + nonRecursiveFibonacci(n));

    scanner.close();
  }
}
```

```
Input:
Enter a number:
6

Output:
Recursive Fibonacci: 8
Non-recursive Fibonacci: 8
```

P7: Write a Java program that prompts the user for an integer and then prints out all the prime numbers up to that Integer?

```
import java.util.Scanner;

public class PrimeNumbers {
  public static boolean isPrime(int n) {
    if (n <= 1) {
      return false;
    }

    for (int i = 2; i < n; i++) {
      if (n % i == 0) {
        return false;
      }
    }

    return true;
  }

  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter an integer: ");
    int n = scanner.nextInt();

    System.out.println("Prime numbers up to " + n + ":");
    for (int i = 2; i <= n; i++) {
      if (isPrime(i)) {
        System.out.println(i);
      }
    }

    scanner.close();
  }
}
```

Input:

```
Enter an integer:
20
```

Output:

```
Prime numbers up to 20:
2
3
5
7
11
```

```
13
17
19
```

P8: Write a Java program to multiply two given matrices?

```
import java.util.Scanner;

public class MatrixMultiplication {
  public static int[][] multiply(int[][] a, int[][] b) {
    int rowsA = a.length;
    int columnsA = a[0].length;
    int columnsB = b[0].length;

    int[][] result = new int[rowsA][columnsB];

    for (int i = 0; i < rowsA; i++) {
      for (int j = 0; j < columnsB; j++) {
        for (int k = 0; k < columnsA; k++) {
          result[i][j] += a[i][k] * b[k][j];
        }
      }
    }

    return result;
  }

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

    System.out.print("Enter number of rows for matrix A: ");
    int rowsA = scanner.nextInt();
    System.out.print("Enter number of columns for matrix A: ");
    int columnsA = scanner.nextInt();
    int[][] matrixA = new int[rowsA][columnsA];

    System.out.println("Enter elements for matrix A:");
    for (int i = 0; i < rowsA; i++) {
      for (int j = 0; j < columnsA; j++) {
        matrixA[i][j] = scanner.nextInt();
      }
    }

    System.out.print("Enter number of rows for matrix B: ");
    int rowsB = scanner.nextInt();
    System.out.print("Enter number of columns for matrix B: ");
    int columnsB = scanner.nextInt();
    int[][] matrixB = new int[rowsB][columnsB];

    System.out.println("Enter elements for matrix B:");
    for (int i = 0; i < rowsB; i++) {
      for (int j = 0; j < columnsB; j++) {
        matrixB[i][j] = scanner.nextInt();
      }
    }

    if (columnsA != rowsB) {
      System.out.println("Error: The number of columns of matrix A must be equal
to the number of rows of matrix B");
    } else {
      int[][] result = multiply(matrixA, matrixB);

      System.out.println("Result:");
      for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < columnsB; j++) {
          System.out.print(result[i][j] + " ");
        }
        System.out.println();
      }
    }

    scanner.close();
  }
}
```

Input:

```
Enter number of rows for matrix A: 2
Enter number of columns for matrix A: 3
Enter elements for matrix A:
1 2 3
4 5 6
Enter number of rows for matrix B: 3
Enter number of columns for matrix B: 2
Enter elements for matrix B:
7 8
9 10
11 12
```

Output:

```
Result:
58 64
139 154
```

P9: Write a Java program for sorting a given list of names in ascending order?

Program:

```java
import java.util.Scanner;
import java.util.Arrays;

public class SortNames {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter number of names: ");
    int n = scanner.nextInt();
    String[] names = new String[n];

    System.out.println("Enter names:");
    for (int i = 0; i < n; i++) {
      names[i] = scanner.next();
    }
    Arrays.sort(names);

    System.out.println("Sorted names:");
    for (String name : names) {
      System.out.println(name);
    }

    scanner.close();
  }
}
```

Input:

```
Enter number of names: 4
Enter names:
John
Amy
David
Bob
```

Output:

```
Sorted names:
Amy
Bob
David
John
```

P10: Write a java program for Method overloading and Constructor overloading.

Program:

```
class OverloadingDemo {
  int num1;
  int num2;
  OverloadingDemo(int num1, int num2) {
    this.num1 = num1;
    this.num2 = num2;
  }
  OverloadingDemo(int num1) {
    this.num1 = num1;
    this.num2 = 0;
  }
  int add() {
    return num1 + num2;
  }
  int add(int num1, int num2) {
    return num1 + num2;
  }
  int add(int num1, int num2, int num3) {
    return num1 + num2 + num3;
  }
}
```

```
public class Main {

  public static void main(String[] args) {

    OverloadingDemo obj1 = new OverloadingDemo(5, 10);

    System.out.println("Sum using instance variables: " + obj1.add());

    OverloadingDemo obj2 = new OverloadingDemo(5);

    System.out.println("Sum using instance variables: " + obj2.add());

    System.out.println("Sum using method overloading: " + obj1.add(1, 2));

    System.out.println("Sum using method overloading: " + obj1.add(1, 2, 3));

  }

}
```

Output:

```
Sum using instance variables: 15

Sum using instance variables: 5

Sum using method overloading: 3

Sum using method overloading: 6
```

P11: Write a java program to represent Abstract class with example.

```
abstract class Shape {
  int x, y;

  Shape(int x, int y) {
    this.x = x;
    this.y = y;
  }

  abstract void draw();
}

class Circle extends Shape {
  int radius;

  Circle(int x, int y, int radius) {
    super(x, y);
    this.radius = radius;
  }

  void draw() {
    System.out.println("Drawing Circle at (" + x + "," + y + ") with radius " + radius);
  }
}

class Rectangle extends Shape {
  int width, height;

  Rectangle(int x, int y, int width, int height) {
    super(x, y);
    this.width = width;
    this.height = height;
  }

  void draw() {
    System.out.println("Drawing Rectangle at (" + x + "," + y + ") with width " +
width + " and height " + height);
  }
}

public class Main {
  public static void main(String[] args) {
    Shape circle = new Circle(1, 2, 3);
    Shape rectangle = new Rectangle(4, 5, 6, 7);
    circle.draw();
    rectangle.draw();
  }
}
```

Output:

```
Drawing Circle at (1,2) with radius 3
Drawing Rectangle at (4,5) with width 6 and height 7
```

P12: Write a program to implement multiple Inheritances.

Program:

```
interface A {
  void methodA();
}

interface B {
  void methodB();
}

class C implements A, B {
  public void methodA() {
    System.out.println("Method A of Class C");
  }

  public void methodB() {
    System.out.println("Method B of Class C");
  }
}

public class Main {
  public static void main(String[] args) {
    System.out.print("Enter a number (1 for Method A, 2 for Method B): ");
    Scanner scan = new Scanner(System.in);
    int choice = scan.nextInt();
    C obj = new C();

    switch (choice) {
      case 1:
        obj.methodA();
        break;
      case 2:
        obj.methodB();
        break;
      default:
        System.out.println("Invalid Choice");
        break;
    }
  }
}
```

Input & Output:

```
Enter a number (1 for Method A, 2 for Method B): 1
Method A of Class C
```

P13: write program to demonstrate method overriding and super keyword.

```
class Shape {
  void draw() {
    System.out.println("Drawing Shape");
  }
}

class Circle extends Shape {
  void draw() {
    super.draw();
    System.out.println("Drawing Circle");
  }
}

class Rectangle extends Shape {
  void draw() {
    super.draw();
    System.out.println("Drawing Rectangle");
  }
}

public class Main {
  public static void main(String[] args) {
    System.out.print("Enter a number (1 for Circle, 2 for Rectangle): ");
    Scanner scan = new Scanner(System.in);
    int choice = scan.nextInt();

    Shape obj;

    switch (choice) {
      case 1:
        obj = new Circle();
        obj.draw();
        break;
      case 2:
        obj = new Rectangle();
        obj.draw();
        break;
      default:
        System.out.println("Invalid Choice");
        break;
    }
  }
}
```

Input & Output:

```
Enter a number (1 for Circle, 2 for Rectangle): 1
Drawing Shape
Drawing Circle
```

P14: Write a java program to implement Interface using extends keyword.

Program:

```
interface A {
  void methodA();
}

interface B extends A {
  void methodB();
}

class C implements B {
  public void methodA() {
    System.out.println("Method A of Class C");
  }

  public void methodB() {
    System.out.println("Method B of Class C");
  }
}

public class Main {
  public static void main(String[] args) {
    System.out.print("Enter a number (1 for Method A, 2 for Method B): ");
    Scanner scan = new Scanner(System.in);
    int choice = scan.nextInt();

    C obj = new C();

    switch (choice) {
      case 1:
        obj.methodA();
        break;
      case 2:
        obj.methodB();
        break;
      default:
        System.out.println("Invalid Choice");
        break;
    }
  }
}
```

Input & Output:

```
Enter a number (1 for Method A, 2 for Method B): 1
Method A of Class C
```

P15: Write a java program to create inner classes.

Program:

```
class Outer {
  private int outerVariable = 100;

  class Inner {
    void display() {
      System.out.println("Outer Variable: " + outerVariable);
    }
  }

  void createInner() {
    Inner inner = new Inner();
    inner.display();
  }
}

public class Main {
  public static void main(String[] args) {
    Outer outer = new Outer();
    outer.createInner();
  }
}
```

Output:

```
Outer Variable: 100
```

P16: Write a java program to create user defined package.

Program:

```
package mypackage;
public class Hello {
  public void display() {
    System.out.println("Hello from Package");
  }
}

import mypackage.Hello;

public class Main {
  public static void main(String[] args) {
    Hello hello = new Hello();
    hello.display();
  }
}
```

Output:

```
Hello from Package
```

P17: Write a Java program that displays the number of characters, lines and words in a text?

Program:

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter a text: ");
    String text = scanner.nextLine();

    int characters = text.length();
    int words = text.split("\\s+").length;
    int lines = text.split("\n").length;

    System.out.println("Number of characters: " + characters);
    System.out.println("Number of words: " + words);
    System.out.println("Number of lines: " + lines);
  }
}
```

Input:

```
Hello World!
This is a sample text.
```

Output:

```
Number of characters: 32
Number of words: 7
Number of lines: 2
```

P18: Write a Java program that checks whether a given string is a palindrome or not. Ex: MADAM is a palindrome?

Program:

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter a string: ");
    String input = scanner.nextLine();

    StringBuffer buffer = new StringBuffer(input);
    buffer.reverse();

    if (input.equals(buffer.toString())) {
      System.out.println("The given string is a palindrome.");
    } else {
      System.out.println("The given string is not a palindrome.");
    }
  }
}
```

Input:

```
racecar
```

Output:

```
The given string is a palindrome.
```

P19: Write a Java program that reads a line of integers and then displays each integer and the sum of all integers. (Use StringTokenizer class)?

Program:

```
import java.util.Scanner;
import java.util.StringTokenizer;

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter a line of integers separated by spaces: ");
    String input = scanner.nextLine();

    StringTokenizer tokenizer = new StringTokenizer(input);
    int sum = 0;

    while (tokenizer.hasMoreTokens()) {
      int num = Integer.parseInt(tokenizer.nextToken());
      System.out.println("Integer: " + num);
      sum += num;
    }
    System.out.println("Sum: " + sum);
  }
}
```

Input:

```
1 2 3 4 5
```

Output:

```
Integer: 1
Integer: 2
Integer: 3
Integer: 4
Integer: 5
Sum: 15
```

P20: Write a java program for creating single try block with multiple catch blocks.

Program:

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter two numbers: ");

    try {
      int num1 = scanner.nextInt();
      int num2 = scanner.nextInt();
      int result = num1 / num2;
      System.out.println("Result: " + result);
    } catch (ArithmeticException e) {
      System.out.println("ArithmeticException: Cannot divide by zero.");
    } catch (Exception e) {
      System.out.println("Exception: Invalid input.");
    } finally {
      scanner.close();
      System.out.println("Scanner closed.");
    }
  }
}
```

Input:
10 0

Output:
```
ArithmeticException: Cannot divide by zero.
Scanner closed.
```

P21: write a program for multiple try blocks and multiple catch blocks including finally.

Program:

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter first number: ");
    int num1;

    try {
      num1 = scanner.nextInt();
    } catch (Exception e) {
      System.out.println("Exception: Invalid input for first number.");
      return;
    }

    System.out.println("Enter second number: ");
    int num2;

    try {
      num2 = scanner.nextInt();
    } catch (Exception e) {
      System.out.println("Exception: Invalid input for second number.");
      return;
    }

    try {
      int result = num1 / num2;
      System.out.println("Result: " + result);
    } catch (ArithmeticException e) {
      System.out.println("ArithmeticException: Cannot divide by zero.");
    } finally {
      scanner.close();
      System.out.println("Scanner closed.");
    }
  }
}
```

Input:

```
10 0
```

Ouput:

```
ArithmeticException: Cannot divide by zero.
Scanner closed.
```

P22: write a program to create user defined exception

Program:

```
import java.util.Scanner;

class InvalidAgeException extends Exception {
  public InvalidAgeException(String message) {
    super(message);
  }
}

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter your age: ");
    int age;
    try {
      age = scanner.nextInt();
      if (age < 0) {
        throw new InvalidAgeException("Age cannot be negative.");
      }
      System.out.println("Age: " + age);
    } catch (InvalidAgeException e) {
      System.out.println(e.getMessage());
    } catch (Exception e) {
      System.out.println("Exception: Invalid input for age.");
    } finally {
      scanner.close();
      System.out.println("Scanner closed.");
    }
  }
}
```

Input:

```
-10
```

Ouput:

```
Age cannot be negative.
Scanner closed.
```

P23: Write a java program for producer and consumer problem using Threads.

Program:

```
package test;
public class Producer implements Runnable{
	public StringBuffer sb=null;//Instance variable
	public Producer()//0-argument Constructor
	{
		sb = new StringBuffer();
	}
   public void run() {
	try {
```

```
                synchronized(sb) {
                        for(int i=1;i<=10;i++) {
                                sb.append(i+":");
                                System.out.println("Producer appeding data..");
                                Thread.sleep(1000);
                        }//end of loop
                        sb.notify();//sending msg_to_Waiting thread
                }//end of lock
        }catch(Exception e) {e.printStackTrace();}
   }
}

Consumer.java
package test;
public class Consumer implements Runnable{
  public Producer prod=null;
  public Consumer(Producer prod)
  {
          this.prod=prod;
  }
  public void run() {
          try {
                  synchronized(prod.sb) {
                          System.out.println
                          ("Consumer Activated..and Blocked");
                          prod.sb.wait();//Blocked
                          System.out.println("====Display using Consumer===");
                          System.out.println(prod.sb.toString());
                  }//end of lock
          }catch(Exception e) {e.printStackTrace();}
  }
}

DemoThread.java(MainClass)
package maccess;
import test.*;
public class DemoThread {
        public static void main(String[] args) {
```

```
        Producer p = new Producer();
        Consumer c = new Consumer(p);

        Thread t1 = new Thread(p);
        Thread t2 = new Thread(c);

        t2.start();//Consumer activated
        t1.start();
          }
}
```

Output:

```
Consumer Activated..and Blocked
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
Producer appeding data..
====Display using Consumer===
1:2:3:4:5:6:7:8:9:10:
```

P24: Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

```
import java.util.Random;

class Main {
  public static void main(String[] args) {
    NumberGenerator numberGenerator = new NumberGenerator();
    Thread t1 = new Thread(numberGenerator);
    SquareCalculator squareCalculator = new SquareCalculator();
    Thread t2 = new Thread(squareCalculator);
    CubeCalculator cubeCalculator = new CubeCalculator();
    Thread t3 = new Thread(cubeCalculator);
    t1.start();
    t2.start();
    t3.start();
  }
}

class NumberGenerator implements Runnable {
  public void run() {
    Random random = new Random();
    while (true) {
      int number = random.nextInt(100);
      System.out.println("Generated Number: " + number);
      if (number % 2 == 0) {
        synchronized (SquareCalculator.class) {
          SquareCalculator.number = number;
          SquareCalculator.class.notify();
        }
      } else {
        synchronized (CubeCalculator.class) {
          CubeCalculator.number = number;
          CubeCalculator.class.notify();
        }
      }
      try {
        Thread.sleep(1000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}
```

```
class SquareCalculator implements Runnable {
  static int number;

  public void run() {
    while (true) {
      synchronized (SquareCalculator.class) {
        try {
          SquareCalculator.class.wait();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        int square = number * number;
        System.out.println("Square of " + number + " is: " + square);
      }
    }
  }
}

class CubeCalculator implements Runnable {
  static int number;

  public void run() {
    while (true) {
      synchronized (CubeCalculator.class) {
        try {
          CubeCalculator.class.wait();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        int cube = number * number * number;
        System.out.println("Cube of " + number + " is: " + cube);
      }
    }
  }
}
```

Ouput:

```
Generated Number: 5
Cube of 5 is: 125
Generated Number: 78
Square of 78 is: 6084
Generated Number: 57
Cube of 57 is: 185193
Generated Number: 18
Square of 18 is: 32
Generated Number: 15
Cube of 15 is: 3375
```

P25: write a program to create dynamic array using ArrayList class and the print the contents of the array object.

```
import java.util.ArrayList;
import java.util.Scanner;

public class DynamicArray {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    ArrayList<Integer> array = new ArrayList<>();
    System.out.println("Enter number of elements: ");
    int n = sc.nextInt();
    for (int i = 0; i < n; i++) {
      System.out.println("Enter element " + (i + 1) + " : ");
      int element = sc.nextInt();
      array.add(element);
    }
    System.out.println("Contents of ArrayList: " + array);
  }
}
```

Input:

```
Enter number of elements:
5
Enter element 1 :
1
```

```
Enter element 2 :
2
Enter element 3 :
3
Enter element 4 :
4
Enter element 5 :
5

Ouput:
Contents of ArrayList: [1, 2, 3, 4, 5]
```

P26: Write programs to implement add, search and remove operation on ArrayList object.

```
import java.util.ArrayList;
import java.util.Scanner;

public class ArrayListOperations {
  public static void main(String[] args) {
    ArrayList<Integer> list = new ArrayList<>();
    Scanner sc = new Scanner(System.in);
    int choice, element;
    do {
      System.out.println("1. Add element");
      System.out.println("2. Search element");
      System.out.println("3. Remove element");
      System.out.println("4. Display elements");
      System.out.println("5. Exit");
      System.out.print("Enter your choice: ");
      choice = sc.nextInt();
      switch (choice) {
        case 1:
          System.out.print("Enter element to be added: ");
          element = sc.nextInt();
          list.add(element);
          break;
        case 2:
          System.out.print("Enter element to be searched: ");
          element = sc.nextInt();
          if (list.contains(element))
            System.out.println("Element found at index " + list.indexOf(element));
          else
            System.out.println("Element not found");
          break;
        case 3:
          System.out.print("Enter element to be removed: ");
          element = sc.nextInt();
          if (list.remove((Integer) element))
            System.out.println("Element removed");
          else
            System.out.println("Element not found");
          break;
        case 4:
          System.out.println("Elements in the list: " + list);
          break;
        case 5:
          break;
        default:
          System.out.println("Invalid choice");
          break;
      }
    } while (choice != 5);
    sc.close();
  }
}
```

```
1. Add element
2. Search element
3. Remove element
4. Display elements
5. Exit
Enter your choice: 1
Enter element to be added: 5
1. Add element
2. Search element
```

```
3. Remove element
4. Display elements
5. Exit
Enter your choice: 1
Enter element to be added: 3
1. Add element
2. Search element
3. Remove element
4. Display elements
5. Exit
Enter your choice: 1
Enter element to be added: 7
1. Add element
2. Search element
3. Remove element
4. Display elements
5. Exit
Enter your choice: 4
Elements in the list: [5, 3, 7]
1. Add element
2. Search element
3. Remove element
4. Display elements
5. Exit
Enter your choice: 2
Enter element to be searched: 7
Element found at index 2
1. Add element
2. Search element
3. Remove element
4. Display elements
5. Exit
Enter your choice: 3
Enter element to be removed: 5
Element removed
1. Add element
2. Search element
3. Remove element
```

```
4. Display elements
5. Exit
Enter your choice: 5
```