1. Write a Java program to demonstrate control statements and iterative statements.

2. Write a Java program that demonstrates the use of type conversion and casting.

3. Write a Java program that defines a class and the class methods and access variables through its objects.

4. Create a class Calculator with multiple overloaded methods named add. Implement polymorphism by providing different implementations of the add method that accept different numbers of parameters (e.g., two numbers, three numbers, or an array of numbers) and perform addition accordingly. Demonstrate polymorphism by creating instances of the Calculator class and calling different versions of the add method.

5. Write a Java program to illustrate the concept off multilevel inheritance and method overriding. Write a Java program using inheritance with the help of 'super' keyword.

6. Demonstrate polymorphism by creating instances of the Calculator class and calling different versions of the add method.

7. Create a base class Vehicle with properties like model and year. Make the Vehicle class final to prevent further inheritance. Then, attempt to create a subclass, Car, that tries to inherit from Vehicle. Observe and document the compiler error that occurs due to the final keyword.
8.

a. Write a Java program that illustrates the multiple inheritance by using interfaces
b. b. Write a Java program to illustrate the concept of abstract class.

9. Create a package named com.myapp.math with a class named Calculator that contains methods for basic mathematical operations (e.g., addition, subtraction). Import and use the Calculator class in a different package to perform mathematical calculations.

10. Define an interface Drawable with a draw method. Create a class Circle that implements the Drawable interface and provides an implementation of the draw method to draw a circle. Demonstrate how to implement and apply interfaces in Java by creating an instance of Circle and calling its draw method.

11. Write a Java program that demonstrates the exception handling mechanism and nested try statements.

12. Write a Java program Creating a thread, Multithreading and illustrating inter thread communication.

13. Create a Java program that uses the List interface from the Java Collections Framework with generics. Implement a generic list to store various types of objects, and demonstrate how this approach ensures type safety.

14. Implement a Java program that uses a thread pool to manage and reuse a fixed number of threads for executing tasks concurrently.

15. Write a program that demonstrates the need for thread synchronization. Create multiple threads accessing a shared resource simultaneously without synchronization. Then, modify the program to use synchronization (e.g., synchronized keyword) and show how it resolves conflicts.

16. Create an ATM class to demonstrate encapsulation by making the account balance private and providing methods to access and modify it

17. Create a simple weather forecast program that provides weather conditions (e.g., sunny, rainy, cloudy) for a given location. Use control statements to simulate weather changes and provide forecasts for multiple days.

18. Write a Java program that interacts with a database using JDBC to perform the following operations: Create: Insert a new record into a database table. Prompt the user to enter data for the new record (e.g., name, age), and then insert it into the database . Read: Retrieve data from a specified database table based on user-defined criteria. Allow the user to specify a search criterion (e.g., name) and enter the corresponding value. Retrieve and display records from the database table that match the criteria. Delete: Delete a record from the database table. Prompt the user to specify the record to be deleted (e.g., by ID) and execute an SQL DELETE statement to remove it.

1. Write a Java program to demonstrate control statements and iterative statements.

```
import java.util.Scanner;

public class ControlAndIterativeStatementsDemo {

  public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);


    System.out.println("Choose an operation:");

    System.out.println("1. If-Else Statement");

    System.out.println("2. Switch Statement");

    System.out.println("3. For Loop");

    System.out.println("4. While Loop");

    System.out.print("Enter your choice: ");

    int choice = scanner.nextInt();


    switch (choice) {

      case 1:

        ifElseDemo();

        break;

      case 2:

        switchDemo();

        break;

      case 3:

        forLoopDemo();

        break;

      case 4:
            whileLoopDemo();

            break;

        default:

            System.out.println("Invalid choice. Please choose a valid option (1-4).");

    }

}
```

public static void ifElseDemo() {

```
    Scanner scanner = new Scanner(System.in);


    System.out.print("Enter a number: ");

    int num = scanner.nextInt();


    if (num % 2 == 0) {

        System.out.println(num + " is even.");

    } else {

        System.out.println(num + " is odd.");

    }

}
```

public static void switchDemo() {

```
    Scanner scanner = new Scanner(System.in);


    System.out.print("Enter a day of the week (1-7): ");

    int day = scanner.nextInt();


    String dayName;
```

```
switch (day) {
    case 1:
        dayName = "Sunday";
        break;
    case 2:
        dayName = "Monday";
        break;
    case 3:
        dayName = "Tuesday";
        break;
    case 4:
        dayName = "Wednesday";
        break;
    case 5:
        dayName = "Thursday";
        break;
    case 6:
        dayName = "Friday";
        break;
    case 7:
        dayName = "Saturday";
        break;
    default:
        dayName = "Invalid day";
}
```

```
        System.out.println("The day is " + dayName);
    }

    public static void forLoopDemo() {
        System.out.println("Counting from 1 to 10 using a for loop:");
        for (int i = 1; i <= 10; i++) {
            System.out.print(i + " ");
        }
        System.out.println();
    }

    public static void whileLoopDemo() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a positive integer: ");
        int n = scanner.nextInt();

        int i = 1;
        System.out.println("Counting from 1 to " + n + " using a while loop:");
        while (i <= n) {
            System.out.print(i + " ");
            i++;
        }
        System.out.println();
    }
}
```

2. Write a Java program that demonstrates the use of type conversion and casting.

```java
public class TypeConversionAndCasting {

    public static void main(String[] args) {

        // Implicit Type Conversion (Widening)

        int intValue = 42;

        double doubleValue = intValue; // int to double (automatic conversion)

        System.out.println("Implicit Type Conversion (Widening):");

        System.out.println("int to double: " + doubleValue);


        // Explicit Type Conversion (Narrowing) - Casting

        double doubleNum = 123.456;

        int intNum = (int) doubleNum; // double to int (casting)

        System.out.println("\nExplicit Type Conversion (Narrowing) - Casting:");

        System.out.println("double to int: " + intNum);


        // Type Conversion between Different Data Types

        int x = 10;

        byte y = (byte) x; // int to byte (casting)

        System.out.println("");

        System.out.println("\nType Conversion between Different Data Types:");

        System.out.println("int to byte: " + y);


        // Overflow

        int largeInt = 130;

        byte smallByte = (byte) largeInt; // Overflow will occur, resulting in data loss
```

```
            System.out.println("");

        System.out.println("\nOverflow and Underflow:");
        System.out.println("int to byte (overflow): " + smallByte);

        // Casting with Care
        double num1 = 123.67;
        int num2 = (int) num1; // Fractional part is lost
        System.out.println("\nCasting with Care:");
        System.out.println("double to int (fractional part lost): " + num2);

    }
}
```

3. Write a Java program that defines a class and the class methods and access variables through its objects.

```
class Student {
    // Class variables (attributes)
    String name;
    int age;

    // Constructor to initialize object properties
    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }
    // Class method to display student information
    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
public class Main {
    public static void main(String[] args) {
        // Create objects (instances) of the "Student" class
        Student student1 = new Student("Alice", 20);
        Student student2 = new Student("Bob", 22);

        // Access and modify object properties
        student1.age = 21;  // Modify the age of student1

        // Access class methods to display information
        System.out.println("Student 1 Information:");
        student1.displayInfo();

        System.out.println("\nStudent 2 Information:");
        student2.displayInfo();
    }
}
```

4. Create a class Calculator with multiple overloaded methods named add. Implement polymorphism by providing different implementations of the add method that accept different numbers of parameters (e.g., two numbers, three numbers, or an array of numbers) and perform addition accordingly. Demonstrate polymorphism by creating instances of the Calculator class and calling different versions of the add method.

```
public class Calculator {

  // Method to add two numbers

  public int add(int a, int b) {

    return a + b;

  }

  // Method to add three numbers

  public int add(int a, int b, int c) {

    return a + b + c;

  }

  // Method to demonstrate polymorphism

  public void performAddition() {

    System.out.println("Adding two numbers: " + add(5, 10));

    System.out.println("Adding three numbers: " + add(5, 10, 15));

  }


  public static void main(String[] args) {

    Calculator calculator = new Calculator();

    calculator.performAddition();

  }

}
```

5. [A] Write a Java program to illustrate the concept off multilevel inheritance and method overriding

```
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

class GermanShepherd extends Dog {
    @Override
    void sound() {
        System.out.println("German Shepherd barks loudly");
    }
}

public class MultilevelInheritanceExample {
    public static void main(String[] args) {
        Animal animal = new Animal();
        animal.sound(); // Output: Animal makes a sound
        Dog dog = new Dog();
        dog.sound();    // Output: Dog barks

        GermanShepherd shepherd = new GermanShepherd();
        shepherd.sound(); // Output: German Shepherd barks loudly
    }
}
```

5. [B] Write a Java program using inheritance with the help of 'super' keyword.

```
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    @Override
    void makeSound() {
        super.makeSound(); // Calls the superclass method
        System.out.println("Dog barks");
    }
}
public class SuperKeywordExample {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.makeSound();
    }
}
```

7. **Create a base class Vehicle with properties like model and year. Make the Vehicle class final to prevent further inheritance. Then, attempt to create a subclass, Car, that tries to inherit from Vehicle. Observe and document the compiler error that occurs due to the final keyword.**

```
final class Vehicle {
    private String model;
    private int year;

    public Vehicle(String model, int year) {
        this.model = model;
        this.year = year;
    }

    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }
}
class Car extends Vehicle { // Error: Cannot inherit from final 'Vehicle'
    private String brand;
    public Car(String model, int year, String brand) {
        super(model, year);
        this.brand = brand;
    }

    public String getBrand() {
        return brand;
    }
}

public class InheritanceExample {
    public static void main(String[] args) {
        Car car = new Car("Sedan", 2022, "Toyota");
        System.out.println("Model: " + car.getModel());
        System.out.println("Year: " + car.getYear());
        System.out.println("Brand: " + car.getBrand());
    }
}
```