NAME :TALASANIYA NAVDIP R

ROLL NO :35

CLASS:SY D

Lab assignment -2

Subject : Programming with Java

---

1 Write a program to print numbers from 1 to 10, but stop printing when the number 7 is reached. Use the break statement to exit the loop when the number reaches 7.

```java
public class BreakExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 7) {
                break;
            }
            System.out.print(i + " ");
        }
    }
}

/*
output
 1 2 3 4 5 6
*/
```

2. Write a program to print the numbers from 1 to 10, but skip the number 5. Use the continue statement to skip printing when the number is 5.

```java
public class ContinueExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 5) {
                continue;
            }
            System.out.print(i + " ");
        }
    }
}

/*
output
1 2 3 4 6 7 8 9 10
*/
```

3. Write a program to calculate the sum of all the elements in a one-dimensional array of integers.

```java
public class ArraySum {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int sum = 0;

        for (int num : arr) {
            sum += num;
        }

        System.out.println("Sum of array elements: " + sum);
    }
}

/*
output
Sum of array elements: 15
*/
```

4. Write a program to calculate the sum of all elements in a jagged array of integers.

```java
public class JaggedArraySum {
    public static void main(String[] args) {
        int[][] jaggedArr = {
            {1, 2, 3},
            {4, 5},
            {6, 7, 8, 9}
        };

        int sum = 0;
        for (int[] row : jaggedArr) {
            for (int num : row) {
                sum += num;
            }
        }

        System.out.println("Sum of jagged array elements: " + sum);
    }
}

/*
output
```

```
Sum of jagged array elements: 45
*/
```

5. Write a program to create a Car class with the following attributes: brand (String), model (String), year (int) Define methods to set the values of these attributes and display the car's information.

```java
class Car {
    String brand;
    String model;
    int year;

    void setValues(String brand, String model, int year) {
        this.brand = brand;
        this.model = model;
        this.year = year;
    }

    void display() {
        System.out.println("Car Brand: " + brand);
        System.out.println("Car Model: " + model);
        System.out.println("Year: " + year);
    }

    public static void main(String[] args) {
        Car myCar = new Car();
        myCar.setValues("Toyota", "Corolla", 2022);
        myCar.display();
    }
}

/*
output
Car Brand: Toyota
Car Model: Corolla
Year: 2022

*/
```

6. Write a programming to implement encapsulation in a Person class with the name, age, and address attributes and create getter and setter methods to access and update the private variables.

```java
class Person {
    private String name;
    private int age;
    private String address;

    public String getName() { return name; }
    public int getAge() { return age; }
    public String getAddress() { return address; }

    public void setName(String name) { this.name = name; }
    public void setAge(int age) { this.age = age; }
    public void setAddress(String address) { this.address = address; }
}

public class EncapsulationExample {
    public static void main(String[] args) {
        Person p = new Person();
        p.setName("John");
        p.setAge(25);
        p.setAddress("New York");

        System.out.println("Name: " + p.getName());
        System.out.println("Age: " + p.getAge());
        System.out.println("Address: " + p.getAddress());
    }
}

/*
output
Name: John
Age: 25
Address: New York
*/
```

7. Write a program to demonstrate abstraction.

```java
abstract class Animal {
    abstract void makeSound();
}

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

public class AbstractionExample {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        myDog.makeSound();
    }
}
/*
output
Dog barks

*/
```

8. Write a program to demonstrate method overloading and method overriding.

```
class MathOperations {
    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }
}

class Parent {
    void show() {
        System.out.println("This is Parent class");
    }
}

class Child extends Parent {
    void show() {
        System.out.println("This is Child class");
    }
}

public class OverloadingOverriding {
    public static void main(String[] args) {
        MathOperations mo = new MathOperations();
        System.out.println("Addition of 2 numbers: " + mo.add(2, 3));
        System.out.println("Addition of 3 numbers: " + mo.add(2, 3, 4));

        Parent obj = new Child();
        obj.show();
    }
}
/*
output
```

```
Addition of 2 numbers: 5
Addition of 3 numbers: 9
This is Child class
*/
```

9. Demonstrate multilevel inheritance, where a class inherits from another class, which itself inherits from another class.

```java
class GrandParent {
    void grandParentMethod() {
        System.out.println("This is Grandparent class");
    }
}

class Parent extends GrandParent {
    void parentMethod() {
        System.out.println("This is Parent class");
    }
}

class Child extends Parent {
    void childMethod() {
        System.out.println("This is Child class");
    }
}

public class MultilevelInheritance {
    public static void main(String[] args) {
        Child obj = new Child();
        obj.grandParentMethod();
        obj.parentMethod();
        obj.childMethod();
    }
}

/*
output
This is Grandparent class
This is Parent class
This is Child class
*/
```

10. Demonstrate hierarchical inheritance, where multiple subclasses inherit from a single superclass.

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

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

class Cat extends Animal {
    void meow() {
        System.out.println("Cat meows");
    }
}

public class HierarchicalInheritance {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();
        d.bark();

        Cat c = new Cat();
        c.eat();
        c.meow();
    }
}

/*
output
This animal eats food
Dog barks
This animal eats food
Cat meows
*/
```

11. Write a program to demonstrate constructor.

```java
class DemoConstructor {
    DemoConstructor() {
        System.out.println("Constructor is called!");
    }

    public static void main(String[] args) {
        DemoConstructor obj = new DemoConstructor();
    }
}

/*
output
Constructor is called!
*/
```

12. Write a program to demonstrate constructor overloading.

```
class ConstructorOverloading {
    ConstructorOverloading() {
        System.out.println("Default Constructor");
    }

    ConstructorOverloading(int a) {
        System.out.println("Parameterized Constructor: " + a);
    }

    public static void main(String[] args) {
        new ConstructorOverloading();
        new ConstructorOverloading(10);
    }
}

/*
output
Default Constructor
Parameterized Constructor: 10

*/
```