

//1.If else Demo

```
public class ifelse
{
    public static void main(String[]args)
    {
        int age=20;if(age>18)
        {
            System.out.print("Age is greater than 18");
        }
        else{
            System.out.print("You Are Below 18");
        }
    }
}
```

//2.do while example

```
public class DoWhileExample {
    public static void main(String[] args) {
        int i = 1;
        do {
            System.out.println(i);
            i++;
        } while (i <= 10);
    }
}
```

// 3.for loop

```
public class ForExample {
    public static void main(String[] args) {
        //Code of Java for loop for(int i=1;i<=10;i++){
        System.out.println(i);
        }
    }
}
```

// 4. This keyword example

```
class Student{
    int rollno; String name; float fee;
    Student(int rollno,String name,float fee){ this.rollno=rollno;
    this.name=name; this.fee=fee;
    }
    void display(){System.out.println(rollno+" "+name+" "+fee);}
}
```

```
class TestThis2 {
```

```

    public static void main(String args[]) {
        Student s1 = new Student(111, "ankit", 5000f);
        Student s2 = new Student(112, "sumit", 6000f);
        s1.display();
        s2.display();
    }
}

// 5.new keyword in java
public class NewExample1 {
    void display()
    {
        System.out.println("Invoking Method");
    }
    public static void main(String[] args) {
        NewExample1 obj = new NewExample1();
        obj.display();
    }
}

// 6.default constructor class Bike1{
// creating a default constructor Bike1(){System.out.println("Bike is
// created");}
// main method

    public static void main(String args[])
        // calling a default constructor Bike1 b=new Bike1();
    }
}

// 7.parameterized constructor

// Java Program to demonstrate the use of the parameterized constructor. class
// Student4{
int id;
String name;
// creating a parameterized constructor Student4(int i,String n){
id=i;name=n;
}

// method to display the values

void display() {
    System.out.println(id + " " + name);
}

    public static void main(String args[]) {
        // creating objects and passing values Student4 s1 = new Student4(111,"Karan");

```

```

        // Student4 s2 = new Student4(222,"Aryan");
        // calling method to display the values of object s1.display();
        s2.display();
    }
}

// 8.copy Constructor
public class Fruit
{
    private double fprice;
    private String fname;
    // constructor to initialize roll number and name of the student Fruit(double
    // fPrice, String fName)
    {

        fprice=fPrice;fname=fName;}
    // creating a copy constructor Fruit(Fruit fruit)
    {
        System.out.println("\nAfter invoking the Copy Constructor:\n");fprice=fruit.fprice;fname=fruit.fname;
    }
    // creating a method that returns the price of the fruit double showPrice()
    {
        return fprice;
    }

    // creating a method that returns the name of the fruit String showName()
    {
        return fname;
    }

    // class to create student object and print roll number and name of the student
    // public static void main(String args[])
    {
        Fruit f1 = new Fruit(399, "Ruby Roman Grapes");
        System.out.println("Name of the first fruit: "+f1.showName());System.out.println("Price of the first
fruit: "+f1.showPrice());
        // passing the parameters to the copy constructor Fruit f2 = new Fruit(f1);
        System.out.println("Name of the second fruit: "+f2.showName());System.out.println("Price of the
second fruit: "+f2.showPrice());}
    }

// 9.constructor overloading
public class Student {
    // instance variables of the class int id;
    String name;

    Student(){
        System.out.println("this a default constructor");
    }
}

```

```

    }

    Student(int i, String n){ id = i;
        name = n;
    }

    public static void main(String[] args) {
        // object creation
        Student s = new Student();
        System.out.println("\nDefault Constructor values: \n");
        System.out.println("Student Id : " + s.id + "\nStudent Name : " + s.name);
        System.out.println("\nParameterized Constructor values: \n");
        Student student = new Student(10, "David");
        System.out.println("Student Id : " + student.id + "\nStudent Name : " + student.name);
    }
}

// 10.static variable program
// Java Program to demonstrate the use of static variable class Student{
int rollno;// instance variable String name;
static String college = "ITS";// static variable

// constructor Student(int r, String n){ rollno = r;
name=n;
}

// method to display the values
void display() {
    System.out.println(rollno + " " + name + " " + college);
}
}

// Test class to show the values of objects public class TestStaticVariable1{
public static void main(String args[]) {
    Student s1 = new Student(111, "Karan");
    Student s2 = new Student(222, "Aryan");
    // we can change the college of all objects by the single line of code
    // Student.college="BBDIT"; s1.display();
    s2.display();
}
}

// 11 . static method program
// Java Program to demonstrate the use of a static method.
class Student{
    int rollno;
    String name;
    static String college = "ITS";
    // static method to change the value of static variable static void change(){

```

```

college="BBDIT";
}
// constructor to initialize the variable Student(int r, String n){
rollno=r;name=n;}
// method to display values

void display() {
    System.out.println(rollno + " " + name + " " + college);
}
}

// Test class to create and display the values of object public class
// TestStaticMethod{
public static void main(String args[]) {
    Student.change();// calling change method
    // creating objects

    Student s1 = new Student(111, "Karan");
    Student s2 = new Student(222, "Aryan");
    Student s3 = new Student(333, "Sonoo");
    // calling display method s1.display();
    s2.display();
    s3.display();
}
}

// 12.static block program without main
class A3{
    static{
        System.out.println("static block is invoked"); System.exit(0);
    }
}

// 13.single inheritance class Employee{

    float salary = 40000;

}

class Programmer extends Employee {
    int bonus = 10000;

    public static void main(String args[]) {
        Programmer p = new Programmer();
        System.out.println("Programmer salary is:" + p.salary);
        System.out.println("Bonus of Programmer is:" + p.bonus);
    }
}

```

```
// 14.multilevel inheritance class Animal{
    void eat() {
        System.out.println("eating...");
    }
}

class Dog extends Animal {

    void bark() {
        System.out.println("barking...");
    }

}

class BabyDog extends Dog {

    void weep() {
        System.out.println("weeping...");
    }

}

class TestInheritance2 {

    public static void main(String args[]){ BabyDog d=new BabyDog(); d.weep();
        d.bark();

        d.eat();

    }}

// 15. Java polymorphism : - method overloading

// 1. method class Adder{
static int add(int a, int b) {
    return a + b;
}

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

}

class TestOverloading1 {

    public static void main(String[] args){ System.out.println(Adder.add(11,11));
```

```
System.out.println(Adder.add(11,11,11));
    }}

// 2.method class Adder{
static int add(int a, int b) {
    return a + b;
}

static double add(double a, double b) {
    return a + b;
}

}

class TestOverloading2 {

    public static void main(String[] args) {
        System.out.println(Adder.add(11, 11));
        System.out.println(Adder.add(12.3, 12.6));
    }
}
```