

# OOPJ ASSIGNMENT - 3

Name - Nishant Nahar

Roll No - 241551078

Section - CSE-25

1. Write a Java program to demonstrate a default constructor that initializes data members with default values.

```
class Demo {
    int a;
    String b;
    Demo() {
        a = 241551078;
        b = "Hello Nishant Here";
    }
    static void main(String[] args) {
        System.out.println("Nishant Nahar : 241551078");
        Demo a = new Demo();
        System.out.println(a.a + " " + a.b);
    }
}
```

OUTPUT

```
• B:\java_lab\assignment_4 > main 0 ~1 java .\assign_1.java
Nishant Nahar : 241551078
241551078 Hello Nishant Here
```

2. Write a Java program using a parameterized constructor to initialize student details (roll, name, marks).

```
class Student_2 {
    int roll;
    String name;
    int marks;
    Student_2(int r, String n, int m) {
        roll = r;
        name = n;
        marks = m;
    }
    public static void main(String[] args) {
        System.out.println("nishant nahar 241551078");
        Student_2 s = new Student_2(1, "Amit", 90);
        System.out.println(s.roll + " " + s.name + " " + s.marks);
    }
}
```

OUTPUT

```
• B:\java_lab\assignment_4 > main 0 ~1 java .\assign_2.java
nishant nahar 241551078
1 Amit 90
```

3. Create a class Employee with a constructor that takes employee id and salary and displays them.

```
class emp_01 {
```

```

int id;
double salary;
emp_01(int i, double s) {
    id = i;
    salary = s;
}
public static void main(String[] args) {
    System.out.println("Nishant Nahar : 241551078");
    emp_01 e = new emp_01(1, 234560);
    emp_01 e1 = new emp_01(2, 75340);
    System.out.println("ID " + e.id + " Salary " + e.salary);
    System.out.println("ID " + e1.id + " Salary " + e1.salary);
}
}

```

#### OUTPUT

```

B:\java_lab\assignment_4 >main @ ~1 java .\assign_3.java
Nishant Nahar : 241551078
ID 1 Salary 234560.0
ID 2 Salary 75340.0

```

4. Write a program to show constructor overloading using a Box class (cube and cuboid).

```

class Box_2 {
    int area;
    Box_2(int a) {
        area = a * a;
    }
    Box_2(int l, int b) {
        area = l * b;
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar : 241551078");
        Box_2 cube = new Box_2(3);
        Box_2 cuboid = new Box_2(2, 8);
        System.out.println("The area of cube is " + cube.area);
        System.out.println("The area of cuboid is " + cuboid.area);
    }
}

```

#### OUTPUT

```

B:\java_lab\assignment_4 >main @ ~1 java assign_4.java
Nishant Nahar : 241551078
The area of cube is 9
The area of cuboid is 16

```

5. Create a class Book with a constructor that calculates total price based on quantity and unit price

```

class Book {
    double price_unit;
    int quantity;
    double total_price;
    Book(double p, int q) {
        price_unit = p;
        quantity = q;
        total_price = price_unit * quantity;
    }
}

```

```

}
public static void main(String[] args) {
    System.out.println("Nishant Nahar : 241551078");
    Book b1 = new Book(234.99, 20);
    Book b2 = new Book(560.87, 4);
    System.out.println("The total price of b1 is " + b1.total_price);
    System.out.println("The total price of b2 is " + b2.total_price);
}
}

```

## OUTPUT

```

• B:\java_lab\assignment_4 > /main @ ~1 java .\assign_5.java
Nishant Nahar : 241551078
The total price of b1 is 4699.8
The total price of b2 is 2243.48

```

6. Write a Java program where a constructor counts the number of objects created.

```

class Count {
    static int count = 0; // so that it doesn't change back to 0
    Count() {
        count++;
    }
    public static void main(String[] args) {
        new Count();
        new Count();
        new Count();
        System.out.println(count);
    }
}

```

## OUTPUT

```

• B:\java_lab\assignment_4 > /main @ ~1 java .\assign_6.java
3

```

7. Demonstrate constructor chaining using multiple constructors in the same class.

```

class Chaining {
    Chaining() {
        this(90);
        System.out.println("Hello Guys");
    }
    Chaining(int a) {
        System.out.println(a);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new Chaining();
    }
}

```

## OUTPUT

```

• B:\java_lab\assignment_4 > /main @ ~1 java .\assign_7.java
Nishant Nahar - 241551078
90
Hello Guys

```

8. Write a program to show the difference between constructor and method using execution order.

```

class Diff {
    Diff() {
        System.out.println("It is a constructor");
    }
    void meth() {
        System.out.println("It is a method");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Diff d = new Diff();
        d.meth();
    }
}

```

## OUTPUT

```

• B:\java_lab\assignment_4 > main @ w1 java .\assign_8.java
Nishant Nahar - 241551078
It is a constructor
It is a method

```

9. Create a class Time with constructor to accept hours, minutes, seconds and normalize time.

```

class t {
    int h, m, s;
    t(int hr, int mi, int se) {
        s = se % 60;
        mi += se / 60;
        m = mi % 60;
        h = h + mi / 60;
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        t ti = new t(1, 70, 80);
        System.out.println(ti.h + ":" + ti.m + ":" + ti.s);
    }
}

```

## OUTPUT

```

• B:\java_lab\assignment_4 > main @ w1 java .\assign_9.java
Nishant Nahar - 241551078
1:11:20

```

10. Write a Java program to initialize array elements using a constructor.

```

class arr {
    int a[];
    arr() {
        a = new int[] { 1, 2, 3 };
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        arr x = new arr();
        for (int i = 0; i < x.a.length; i++) {
            System.out.print(x.a[i] + " ");
        }
    }
}

```

## OUTPUT

```
• B:\java_lab\assignment_4 > \main @ ~1 java .\assign_10.java
Nishant Nahar - 241551078
1 2 3
```

11. Write a program demonstrating use of this to distinguish instance variables from parameters.

```
class demo_1 {
    int x;
    demo_1(int x) {
        this.x = x;
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        demo_1 d = new demo_1(10);
        System.out.println(d.x);
    }
}
```

## OUTPUT

```
• B:\java_lab\assignment_4 > \main @ ~1 java .\assign_11.java
Nishant Nahar - 241551078
10
```

12. Use this() to call another constructor within the same class.

```
class Cons {
    Cons() {
        this(34);
    }
    Cons(int x) {
        System.out.println(x);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new Cons();
    }
}
```

## OUTPUT

```
• B:\java_lab\assignment_4 > \main @ ~1 java .\assign_12.java
Nishant Nahar - 241551078
34
```

13. Create a class where a method returns this reference.

```
class Ret {
    Ret get() {
        return this;
    }

    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Ret r = new Ret();
        System.out.println(r.get());
    }
}
```

## OUTPUT

```
• B:\java_lab\assignment_4 > Jmain 0 ~1 java .\assign_13.java
Nishant Nahar - 241551078
Ret@51b279c9
```

14. Write a program to pass current object as method argument using this.

```
class Argu {
    void hello(Argu a) {
        System.out.println("Hello Ji");
    }
    void fail() {
        hello(this);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Argu a = new Argu();
        a.fail();
    }
}
```

## OUTPUT

```
• B:\java_lab\assignment_4 > Jmain 0 ~1 java .\assign_14.java
Nishant Nahar - 241551078
Hello Ji
```

15. Demonstrate method chaining using this.

```
class assign {
    assign a1() {
        System.out.println("a1");
        return this;
    }
    assign a2() {
        System.out.println("a2");
        return this;
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new assign().a1();
        new assign().a2();
    }
}
```

## OUTPUT

```
Ret@51b279c9
• B:\java_lab\assignment_4 > Jmain 0 ~1 java .\assign_15.java
Nishant Nahar - 241551078
a1
a2
```

16. Write a program showing how this improves code readability.

```
class A {
    int x;
    A(int x) {
        this.x = x;
    }
    void d() {
        System.out.println(x);
    }
}
```

```

}
public static void main(String[] args) {
    System.out.println("Nishant Nahar - 241551078");
    A o = new A(10);
    o.d();
}
}

```

## OUTPUT

```

netto JI
• B:\java_lab\assignment_4 > main @ ~1 java .\assign_16.java
Nishant Nahar - 241551078
10

```

17. Use this keyword inside constructor to avoid variable shadowing.

```

class B {
    int a;
    B(int a) {
        this.a = a;
        System.out.println(this.a);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new B(5);
    }
}

```

## OUTPUT

```

Value of a in object: 5
• B:\java_lab\assignment_4 > main @ ~2 java .\assign_17.java
Nishant Nahar - 241551078
5

```

18. Write a program to show incorrect behaviour without using this and correct it.

```

class C {
    int v;
    void bad(int v) {
        v = v;
        System.out.println("Inside bad (shadowing): " + v);
    }
    void good(int v) {
        this.v = v;
        System.out.println("Inside good (using this):" + this.v);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        C o = new C();
        o.bad(10);
        o.good(20);
    }
}

```

## OUTPUT

```

5
• B:\java_lab\assignment_4 > main @ ~2 java .\assign_18.java
Nishant Nahar - 241551078
Inside bad (shadowing): 10
Inside good (using this):20

```

19. Write a Java program to demonstrate single inheritance using Person and Student.

```
class Person {
    String n = "Nishant";
}
class stud extends Person {
    void d() {
        System.out.println(n);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new stud().d();
    }
}
```

OUTPUT

```
• B:\java_lab > main @ ?1 ~3 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
Nishant
```

20. Create a program to show multilevel inheritance using Vehicle → Car → Electric Car.

```
class Person {
    String n = "Nishant";
}
class stud extends Person {
    void d() {
        System.out.println(n);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new stud().d();
    }
}
```

OUTPUT

```
• B:\java_lab > main @ ?1 ~3 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
oaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\redhat.java\jdt
Nishant Nahar - 241551078
Vehicle
Car
Electric Battery
```

21. Write a program showing constructor calling sequence in inheritance.

```
class P {
    P() {
        System.out.println("Parent");
    }
}
class S extends P {
    S() {
        super();
        System.out.println("Child");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
    }
}
```



```

    new S();
}
}

```

## OUTPUT

```

B:\java_lab > Jmain @ ?1 ~3 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
Parent
Child

```

22. Write a Java program to demonstrate aggregation using Address and Employee.

```

class Add {
    String c = "City";
}
class Emp {
    Add a;
    Emp(Add a) {
        this.a = a;
        System.out.println("Address value: " + a.c);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Add ad = new Add();
        new Emp(ad);
    }
}

```

## OUTPUT

```

B:\java_lab > Jmain @ ?1 ~4 b.; cd 'b:\java_lab'; & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
es' '-op' 'C:\Users\nisha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
Address value: City

```

23. Create a class Engine and aggregate it inside a Car class.

```

class Eng {
    String t;
    Eng(String t) {
        this.t = t;
    }
}
class car_1 {
    String m;
    Eng e;
    car_1(String m, Eng e) {
        this.m = m;
        this.e = e;
    }
    void d() {
        System.out.println(m + " has " + e.t);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Eng en = new Eng("V8");
        car_1 c = new car_1("Mustang", en);
        c.d();
    }
}

```

```
}  
}
```

## OUTPUT

```
• B:\java_lab > main 0 ?1 w4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-  
sha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re  
Nishant Nahar - 241551078  
Mustang has V8
```

24. Write a program where a department has multiple Teacher objects.

```
class T {  
    String n;  
    T(String n) {  
        this.n = n;  
    }  
}  
class Dept {  
    String d;  
    T[] ts;  
    Dept(String d, T[] ts) {  
        this.d = d;  
        this.ts = ts;  
    }  
    void list() {  
        for (T t : ts)  
            System.out.println(d + " : " + t.n);  
    }  
    public static void main(String[] args) {  
        System.out.println("Nishant Nahar - 241551078");  
        T[] arr = { new T("Amit"), new T("Sumit") };  
        Dept d = new Dept("CS", arr);  
        d.list();  
    }  
}
```

## OUTPUT

```
Use show-help to display help  
• B:\java_lab > main 0 ?1 w4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-  
sha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re  
Nishant Nahar - 241551078  
CS : Amit  
CS : Sumit
```

25. Demonstrate aggregation using array of objects inside another class.

```
class Obj {  
    int id;  
    Obj(int i) {  
        id = i;  
    }  
}  
class Box {  
    Obj[] a;  
    Box(Obj[] a) {  
        this.a = a;  
    }  
    void show() {
```

```

        for (Obj obj : a) {
            System.out.println("ID: " + obj.id);
        }
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Obj[] list = { new Obj(1), new Obj(2) };
        new Box(list).show();
    }
}

```

#### OUTPUT

```

• B:\java_lab > java main 0 21 ~4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\ee01571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
ID: 1
ID: 2

```

26. Write a program showing difference between inheritance and aggregation.

```

class Base {
    void run() {
        System.out.println("Running");
    }
}
class Inh extends Base {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new Inh().run();
        new Agg().b.run();
    }
}
class Agg {
    Base b = new Base();
}

```

#### OUTPUT

```

• B:\java_lab > java main 0 21 ~4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\ee01571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
Running
Running

```

27. Create a college class that contains Student objects using aggregation.

```

class Stu {
    String n;
    Stu(String n) {
        this.n = n;
    }
}
class Col {
    Stu[] s;
    Col(Stu[] s) {
        this.s = s;
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
    }
}

```

```

    Stu[] list = { new Stu("S1"), new Stu("S2") };
    Col c = new Col(list);
    for (Stu student : c.s) {
        System.out.println(student.n);
    }
}
}

```

## OUTPUT

```

B:\java_lab > main 0 ?1 ~4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\ee01571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
S1
S2

```

28. Write a program to show HAS-A relationship using aggregation.

```

class CPU {
    void op() {
        System.out.println("CPU Op");
    }
}
class PC {
    CPU c = new CPU();
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        PC p = new PC();
        p.c.op();
    }
}

```

## OUTPUT

```

Use 'show-help' to display help
B:\java_lab > main 0 ?1 ~4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\ee01571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
CPU Op

```

29. Demonstrate aggregation with constructor-based object passing.

```

class Tool {
    String type;
    Tool(String t) {
        type = t;
    }
}
class Worker {
    Tool t;
    Worker(Tool t) {
        this.t = t;
    }

    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Worker w = new Worker(new Tool("Drill"));
        System.out.println(w.t.type);
    }
}

```

## OUTPUT

```
Use show-help to display help
• B:\java_lab > main @ ?1 ~4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
Drill
```

30. Write a program where destroying parent object does not destroy aggregated object.

```
class Part {
    void show() {
        System.out.println("Part is alive");
    }
}
class Whole {
    Part p;
    Whole(Part p) {
        this.p = p;
    }
}
public class assign_30 {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Part part = new Part();
        Whole whole = new Whole(part);
        whole.p.show();
        whole = null;
        part.show();
    }
}
```

## OUTPUT

```
• B:\java_lab > main @ ?1 ~4 & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-
sha\AppData\Roaming\Antigravity\User\workspaceStorage\eec1571208ddd9a5447730cd2b52d6f0\re
Nishant Nahar - 241551078
Part is alive
Part is alive
```