OOPJ LAB ASSIGNMENT 5

Name – Nishant Nahar

Roll No – 241551078

1. Write a Java program where a child class calls a parameterized parent constructor using super().

```
class A_1 {
    A_1(int x) {
        System.out.println(x);
    }
}
class B_10 extends A_1 {
    B_10(int x) {
        super(x);
    }
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 2415551078");
        new B(5);
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 2415551078
5
```

2. Demonstrate accessing a hidden parent class variable using super.

```
class A_2 {
    int x = 10;
}
class B_2 extends A_2 {
    int x = 20;
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        B_2 b = new B_2();
        System.out.println(b.x);
        b.show();
    }
    void show() {
        System.out.println(super.x);
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
20
10
```

3. Override a method in a child class and invoke the parent version using super.method().

```java
class par {
    void show() {
        System.out.println("A");
    }
}
class chi extends par {
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new chi().show();
    }
    void show() {
        super.show();
        System.out.println("B");
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
A
B
```

4. Show that super() must be the first statement in a constructor.

```java
class A_0 {
    A_0() {
        System.out.println("A");
    }
}
class assig_4 extends A_0 {
    assig_4() {
        System.out.println("B");
        super();
    }
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new assig_4();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
B
A
```

5. Demonstrate constructor execution order using super.

```java
class A_3 {
    A_3() {
        System.out.println("A");
    }
}
class B_1 extends A_3 {
    B_1() {
        super();
        System.out.println("B");
    }
}
```

```java
class C_0 extends B_1 {
    C_0() {
        super();
        System.out.println("C");
    }
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new C_0();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
A
B
C
```

6. Write a program where omission of super() causes a compilation error.

```java
class par_1 {
    par_1(int x) {
        System.out.println(x);
    }
}
class chi_1 extends par_1 {
    chi_1() {
        System.out.println("chi_1");
    }
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        chi_1 c = new chi_1();
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_6.java
assign_6.java:8: error: constructor par_1 in class par_1 cannot be applied to given types;
    chi_1() {
            ^
  required: int
  found:    no arguments
  reason: actual and formal argument lists differ in length
1 error
error: compilation failed
```

7. Use super in multilevel inheritance.

```java
class A_31 {
    A_31() {
        System.out.println("A");
    }
}
class B_11 extends A_31 {
    B_11() {
        super();
        System.out.println("B");
    }
}
class C_01 extends B_11 {
    C_01() {
```

```
        super();
        System.out.println("C");
    }
    static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new C_01();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
A
B
C
```

8. Demonstrate variable shadowing and resolution using super.

```
class Parent {
    int x = 10;
}
class Child extends Parent {
    int x = 20;
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new Child().show();
    }
    void show() {
        System.out.println(x);
        System.out.println(super.x);
    }}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
20
10
```

9. Show method and variable name conflict resolution using super.

```
class par_1 {
    int x = 10;
    void show() {
        System.out.println("Parent");
    }
}
class chi_1 extends par_1 {
    int x = 20;
    chi_1() {
        System.out.println("Nishant Nahar - 241551078");
        System.out.println(super.x);
        super.show();
    }
    public static void main(String[] a) {
        new chi_1();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
10
Parent
```

10. Distinguish between this and super through a program.

```java
class superr {
    int x = 10;
}
class thiss extends superr {
    int x = 20;
    thiss() {
        System.out.println("Nishant Nahar - 241551078");
        System.out.println(this.x);
        System.out.println(super.x);
    }
    public static void main(String[] a) {
        new thiss();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
20
10
```

11. Demonstrate that a final variable cannot be modified.

```java
class one {
    final int x = 10;
}
class two extends one {
    void show() {
        x = 20;
    }
    public static void main(String[] a) {
        two t = new two();
        t.show();
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_11.java
assign_11.java:7: error: cannot assign a value to final variable x
        x = 20;
        ^
1 error
error: compilation failed
```

12. Initialize a final variable using a constructor.

```java
class n {
    final int x;
    n() {
        x = 10;
```

```
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        n obj = new n();
        System.out.println(obj.x);
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_12.java
Nishant Nahar - 241551078
10
```

13. Show that a final method cannot be overridden.

```
class final_over {
    final void show() { System.out.println("P"); }
}
class no_over extends final_over {
    void show() { System.out.println("C"); }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        new no_over().show();
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_13.java
assign_13.java:6: error: show() in no_over cannot override show() in final_over
    void show() { System.out.println("C"); }
         ^
  overridden method is final
1 error
error: compilation failed
```

14. Demonstrate that a final class cannot be inherited.

```
final class final_1 {
    void show() {
    }
}
class no_1 extends final_1 {   // Compile-time error
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_14.java
assign_14.java:6: error: cannot inherit from final final_1
class no_1 extends final_1 {   // Compile-time error
                   ^
1 error
error: compilation failed
```

15. Final

    a. Show behavior of a final reference variable.

```java
class beh {
    int x;
    beh(int x) { this.x = x; }
}
class beh_1 {
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        final beh r = new beh(10);
        r.x = 20;
        System.out.println(r.x);
        // r = new beh(30); // Not allowed, final reference cannot point to new object
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\l
Nishant Nahar - 241551078
20
```

b. When ref is Used

```java
class beh {
    int x;
    beh(int x) {
        this.x = x;
    }
}
class beh_1 {
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        final beh r = new beh(10);
        r.x = 20;
        System.out.println(r.x);
        r = new beh(30); // Not allowed, final reference cannot point to new object
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_15.java
assign_15.java:15: error: cannot assign a value to final variable r
        r = new beh(30); // Not allowed, final reference cannot point to new object
        ^
1 error
error: compilation failed
```

16. Use final keyword with constructor parameters.

```java
class cons_final {
    int x;
    cons_final(final int x) {
        this.x = x;
    }
}
class cons_final_1 {
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        cons_final c = new cons_final(10);
        System.out.println(c.x);
```

```
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
10
```

17. Demonstrate how final restricts runtime polymorphism.

```
class final_poly {
    final void show() {
    }
}
class final_poly_1 extends final_poly {
    void show() {
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_17.java
assign_17.java:7: error: show() in final_poly_1 cannot override show() in final_poly
    void show() {
         ^
  overridden method is final
1 error
error: compilation failed
```

18. Show compile-time error when overriding a final method.

```
class final_poly {
    final void show() {
    }
}
class final_poly_1 extends final_poly {
    void show() {
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_18.java
assign_18.java:7: error: show() in final_poly_1 cannot override show() in final_poly
    void show() {
         ^
  overridden method is final
1 error
error: compilation failed
```

19. Initialize a final variable inside an instance initializer block.

```
class bloc {
    final int x;
    {
```

```
        x = 10;
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        bloc b = new bloc();
        System.out.println(b.x);
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_19.java
Nishant Nahar - 241551078
10
```

20.Write a program to demonstrate execution of an instance initializer block.

```
class init_block {
    {
        System.out.println("1");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        new init_block();
        new init_block();
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_20.java
Nishant Nahar - 241551078
1
1
```

21. Show execution order of instance initializer block and constructor.

```
class order_block {
    {
        System.out.println("1");
    }
    order_block() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        new order_block();
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_21.java
Nishant Nahar - 241551078
1
2
```

22.Demonstrate multiple instance initializer blocks and their execution order.

```
class multi_block {
    {
```

```
        System.out.println("1");
    }
    {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        new multi_block();
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_22.java
Nishant Nahar - 241551078
1
2
```

23. Show instance initializer block execution in inheritance.

```
class blk_1 {
    {
        System.out.println("1");
    }
}
class blk_2 extends blk_1 {
    {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        new blk_2();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
1
2
```

24. Initialize instance variables using an instance initializer block.

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

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_24.java
Nishant Nahar - 241551078
10
```

25. Write a program to demonstrate runtime polymorphism using method overriding.

```java
class me_1 {
    void show() {
        System.out.println("1");
    }
}
class me_2 extends me_1 {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        me_1 obj = new me_2(); // runtime polymorphism
        obj.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
2
```

26. Demonstrate dynamic method dispatch using parent reference and child object.

```java
class d_par {
    void show() {
        System.out.println("1");
    }
}
class d_chi extends d_par {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        d_par obj = new d_chi();
        obj.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
2
```

27. Show that runtime polymorphism does not apply to instance variables.

```java
class var_par {
    int x = 10;
}
class var_chi extends var_par {
    int x = 20;
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        var_par obj = new var_chi();
        System.out.println(obj.x);
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
10
```

28. Demonstrate runtime polymorphism in multilevel inheritance.

```
class a_a {
    void show() {
        System.out.println("1");
    }
}
class a_b extends a_a {
    void show() {
        System.out.println("2");
    }
}
class a_c extends a_b {
    void show() {
        System.out.println("3");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        a_a obj = new a_c();
        obj.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
3
```

29. Show that constructors do not support runtime polymorphism.

```
class con_par {
    con_par() {
        System.out.println("1");
    }
}
class con_chi extends con_par {
    con_chi() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        con_par obj = new con_chi();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
1
2
```

30. Demonstrate that static methods are not polymorphic.

```java
class stat_par {
    static void show() {
        System.out.println("1");
    }
}
class stat_chi extends stat_par {
    static void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        stat_par obj = new stat_chi();
        obj.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
1
```

31. Show effect of final method on runtime polymorphism.

```java
class poly_1 {
    final void show() {
        System.out.println("1");
    }
}
class poly_2 extends poly_1 {
    void show() {
    } // Compile-time error
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
    }
}
```

OUTPUT

```
PS B:\java_lab\assignment_5> java assign_31.java
assign_31.java:8: error: show() in poly_2 cannot override show() in poly_1
    void show() {
         ^
  overridden method is final
1 error
error: compilation failed
```

32. Demonstrate runtime binding of overridden methods.

```java
class bind_1 {
    void show() {
        System.out.println("1");
    }
}
class bind_2 extends bind_1 {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        bind_1 obj = new bind_2();
        obj.show();
```

```
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
2
```

33. Write a program where a parent reference invokes different child Implementations.

```
class imp_p {
    void show() {
        System.out.println("1");
    }
}
class imp_c extends imp_p {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        imp_p p1 = new imp_p();
        imp_p p2 = new imp_c();
        p1.show();
        p2.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 241551078
1
2
```

34. Demonstrate polymorphism using arrays of parent-class references.

```
class arr_p {
    void show() {
        System.out.println("1");
    }
}
class arr_c extends arr_p {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        arr_p[] arr = {new arr_p(), new arr_c()};
        for (arr_p x : arr) x.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
1
2
```

35. Show method call resolution when overridden methods throw exceptions.

```java
class exc_p {
    void show() throws Exception {
        System.out.println("1");
    }
}
class exc_c extends exc_p {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) throws Exception {
        System.out.println("Nishant Nahar - 241551078");
        exc_p obj = new exc_c();
        obj.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 241551078
2
```

36. Demonstrate polymorphism with method parameters and return types.

```java
class ret_me_1 {
    ret_me_1 show(ret_me_1 r) {
        System.out.println("1");
        return r;
    }
}
class ret_me_2 extends ret_me_1 {
    ret_me_2 show(ret_me_2 r) {
        System.out.println("2");
        return r;
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        ret_me_1 p = new ret_me_2();
        p.show(new ret_me_1());
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
1
```

37. Write a program showing upcasting and its role in polymorphism.

```java
class up_1 {
    void show() {
        System.out.println("1");
    }
}
class up_2 extends up_1 {
    void show() {
        System.out.println("2");
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
```

```
        up_1 obj = new up_2();
        obj.show();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 241551078
2
```

38. Demonstrate why downcasting can cause ClassCastException.

```
class down_1 {
}
class down_2 extends down_1 {
}
class test_down {
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        down_1 obj = new down_1();
        down_2 c = (down_2) obj;
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\lib\idea_rt.jar=59923" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=U
Nishant Nahar - 241551078
Exception in thread "main" java.lang.ClassCastException Create breakpoint : class down_1 cannot be cast to class down_2 (down_1 and down_2 are in unnamed module of loader 'app')
    at test_down.main(assign_38.java:11)
```

39. Write a single program combining super + final + runtime polymorphism and explain the output.

```
class s_f_p {
    final void show() {
        System.out.println("1");
    }
    int x = 10;
}
class s_f_c extends s_f_p {
    int x = 20;
    void display() {
        System.out.println(super.x);
        super.show();
    }
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        s_f_p obj = new s_f_c();
        ((s_f_c) obj).display();
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 241551078
10
1
```