# OOPJ LAB ASSIGNMENT – 6

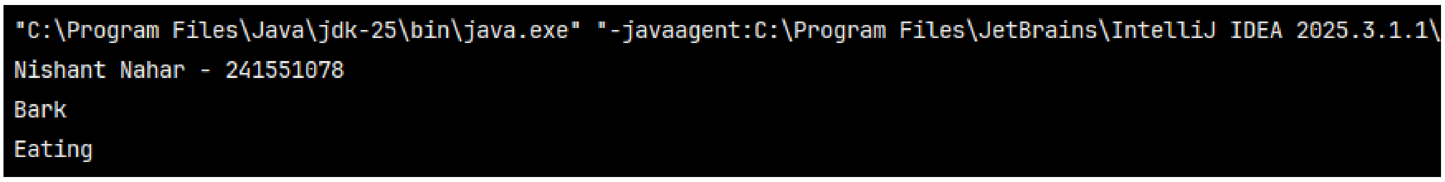
Name – Nishant Nahar

Roll No – 241551078

1. Write a Java program using an abstract class that contains:
    a. An abstract method
    b. A non-abstract method (Demonstrate method implementation in a subclass.)

```
abstract class Animal {
    abstract void so();
    void eat() {
        System.out.println("Eating");
    }}
class Dog extends Animal {
    void so() {
        System.out.println("Bark");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Animal a = new Dog();
        a.so();
        a.eat();
    }}
```

OUTPUT

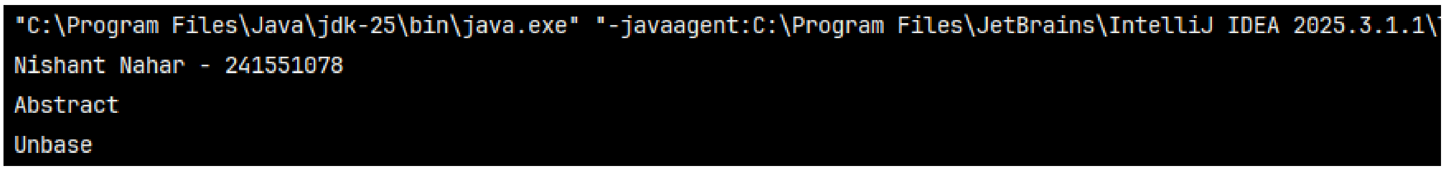


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Bark
Eating
```

2. Create an abstract class with a constructor and show how it is invoked during object creation.

```
abstract class Bas {
    Bas() {
        System.out.println("Abstract");
    }}
class Unbase extends Bas {
    Unbase() {
        System.out.println("Unbase");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new Unbase();
    }}
```

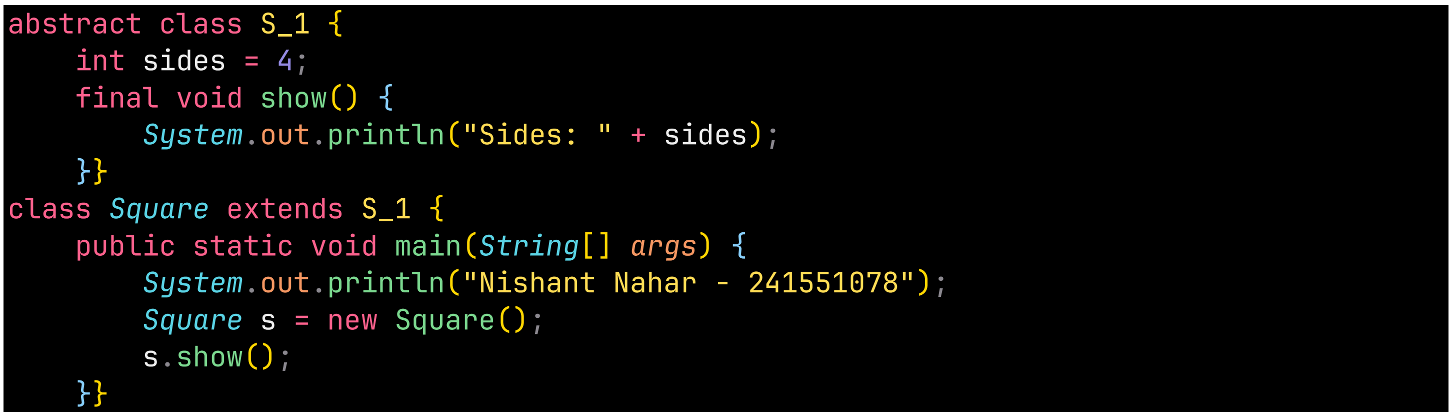
OUTPUT



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

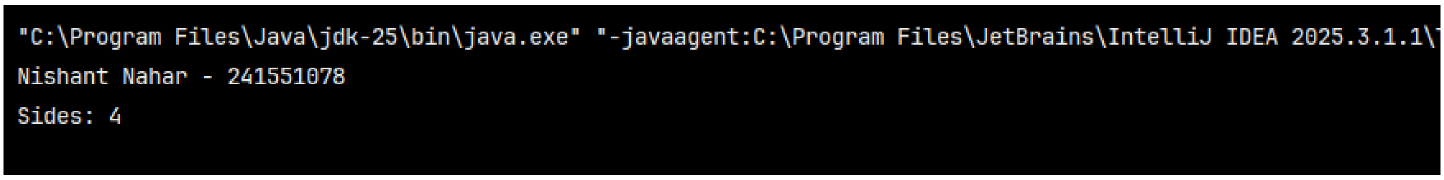
3. Write a program where an abstract class has:
    a. Instance variables

b. Final methods   Show how subclasses interact with them.



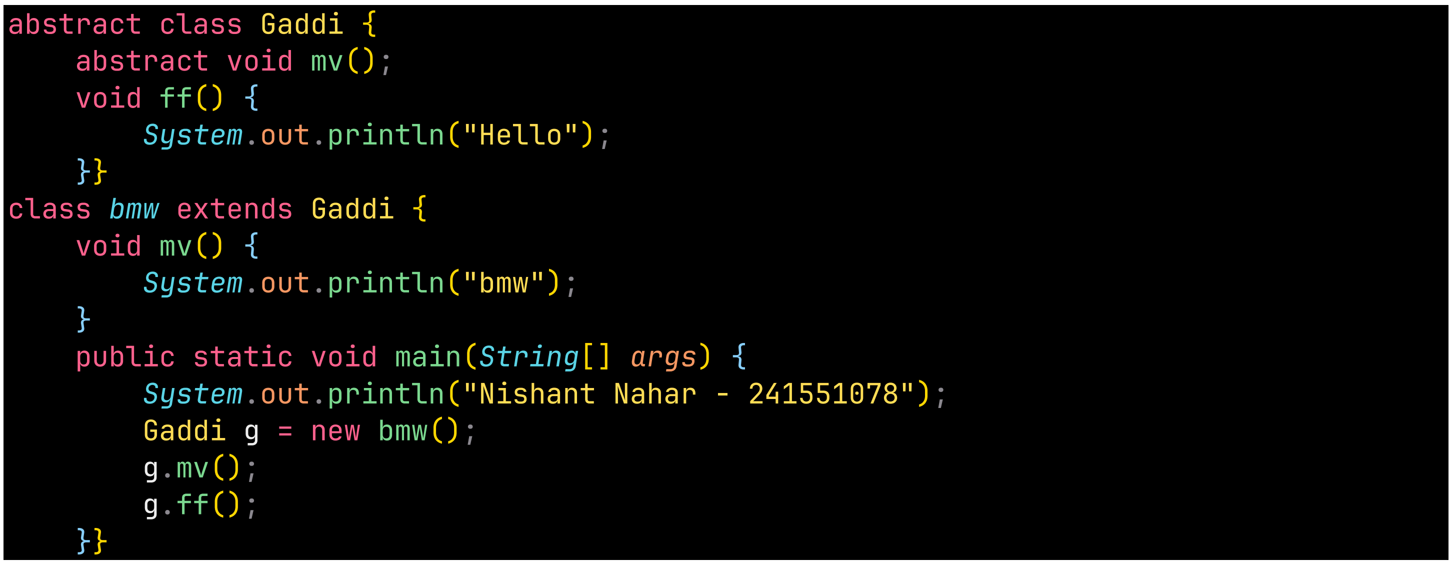
```
abstract class S_1 {
    int sides = 4;
    final void show() {
        System.out.println("Sides: " + sides);
    }}
class Square extends S_1 {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Square s = new Square();
        s.show();
    }}
```

OUTPUT



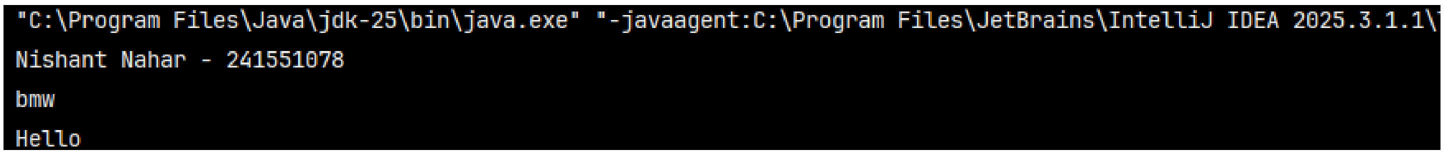
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Sides: 4
```

4. Demonstrate partial abstraction using an abstract class and explain why complete abstraction is not possible.



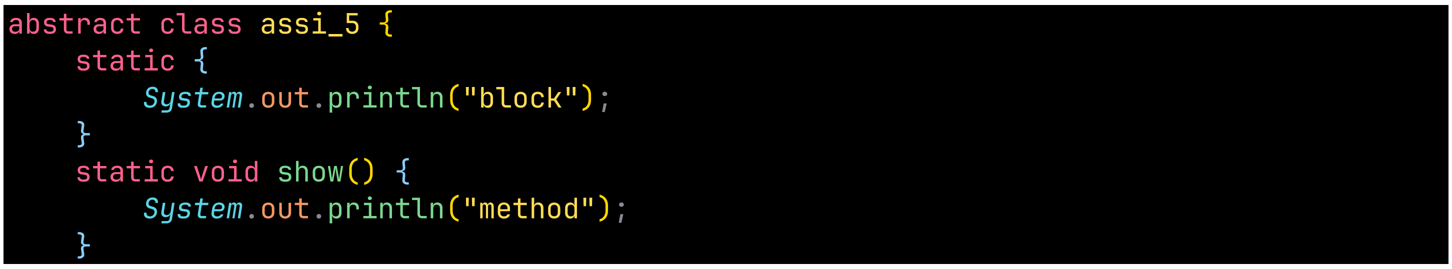
```
abstract class Gaddi {
    abstract void mv();
    void ff() {
        System.out.println("Hello");
    }}
class bmw extends Gaddi {
    void mv() {
        System.out.println("bmw");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Gaddi g = new bmw();
        g.mv();
        g.ff();
    }}
```

OUTPUT

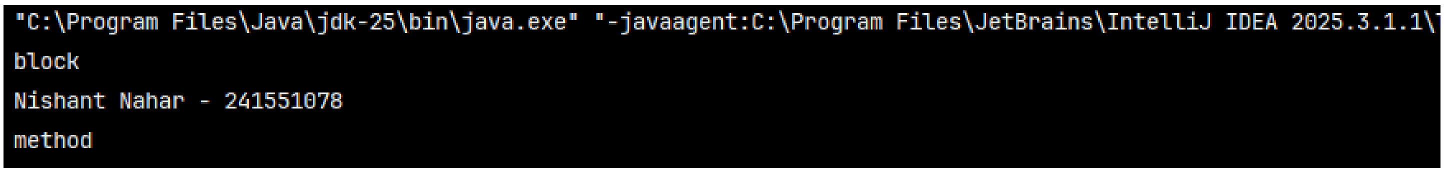


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
bmw
Hello
```

5. Write a program to show that an abstract class can have:
   a. Static methods
   b. Static blocks and demonstrates their execution.

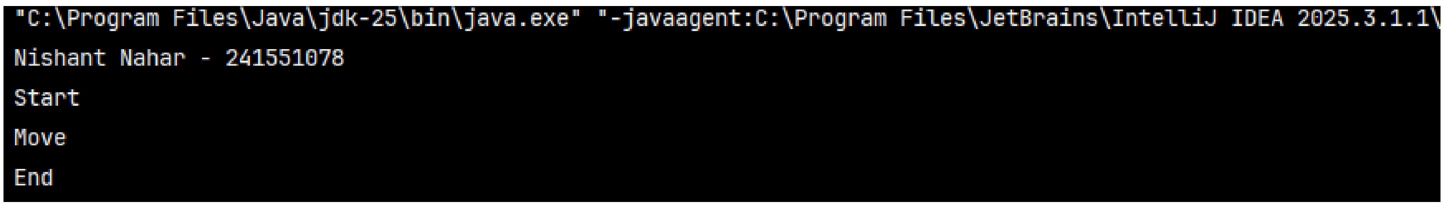


```
abstract class assi_5 {
    static {
        System.out.println("block");
    }
    static void show() {
        System.out.println("method");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        assi_5.show();
    }}
```
OUTPUT


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

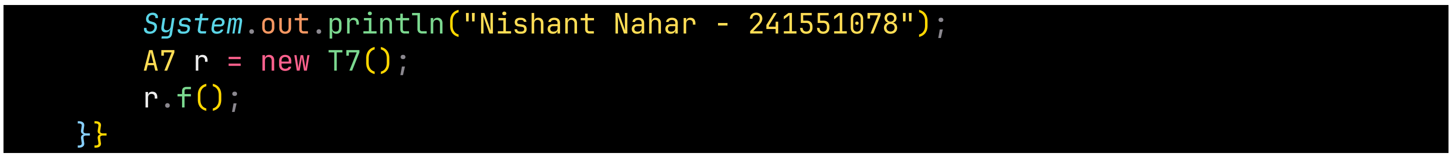
6. Create an abstract class with a template method pattern and implement different behaviors in subclasses.

```
abstract class Game {
    final void play() {
        start();
        move();
        end();
    }
    abstract void move();
    void start() {
        System.out.println("Start");
    }
    abstract void end();
}
class VC extends Game {
    void move() {
        System.out.println("Move");
    }
    void end() {
        System.out.println("End");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Game g = new VC();
        g.play();
    }}
```
OUTPUT


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

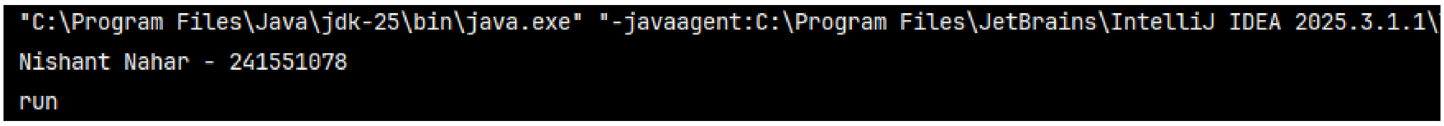
7. Write a program to show runtime polymorphism using an abstract class reference.

```
abstract class A7 {
    abstract void f();
}
class T7 extends A7 {
    void f() {
        System.out.println("run");
    }
    public static void main(String[] a) {
```



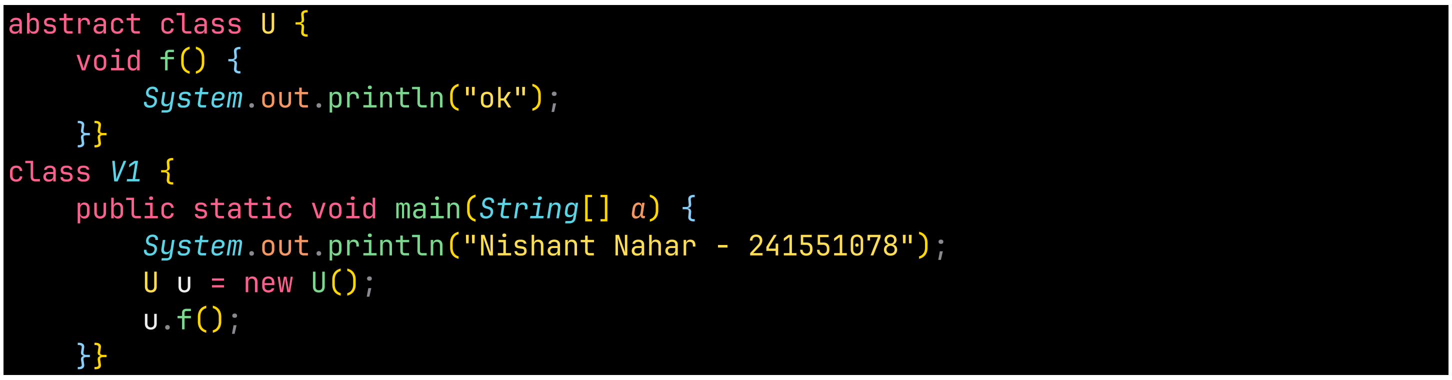
```java
        System.out.println("Nishant Nahar - 241551078");
        A7 r = new T7();
        r.f();
    }}
```

OUTPUT



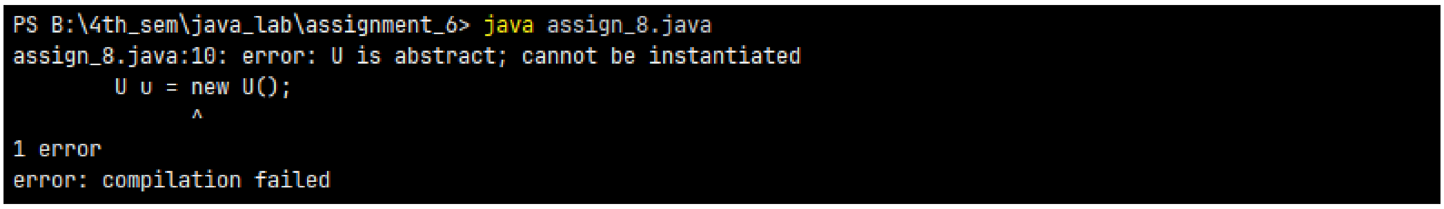
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
run
```

8. Demonstrate why an abstract class cannot be instantiated, but its reference can be used.



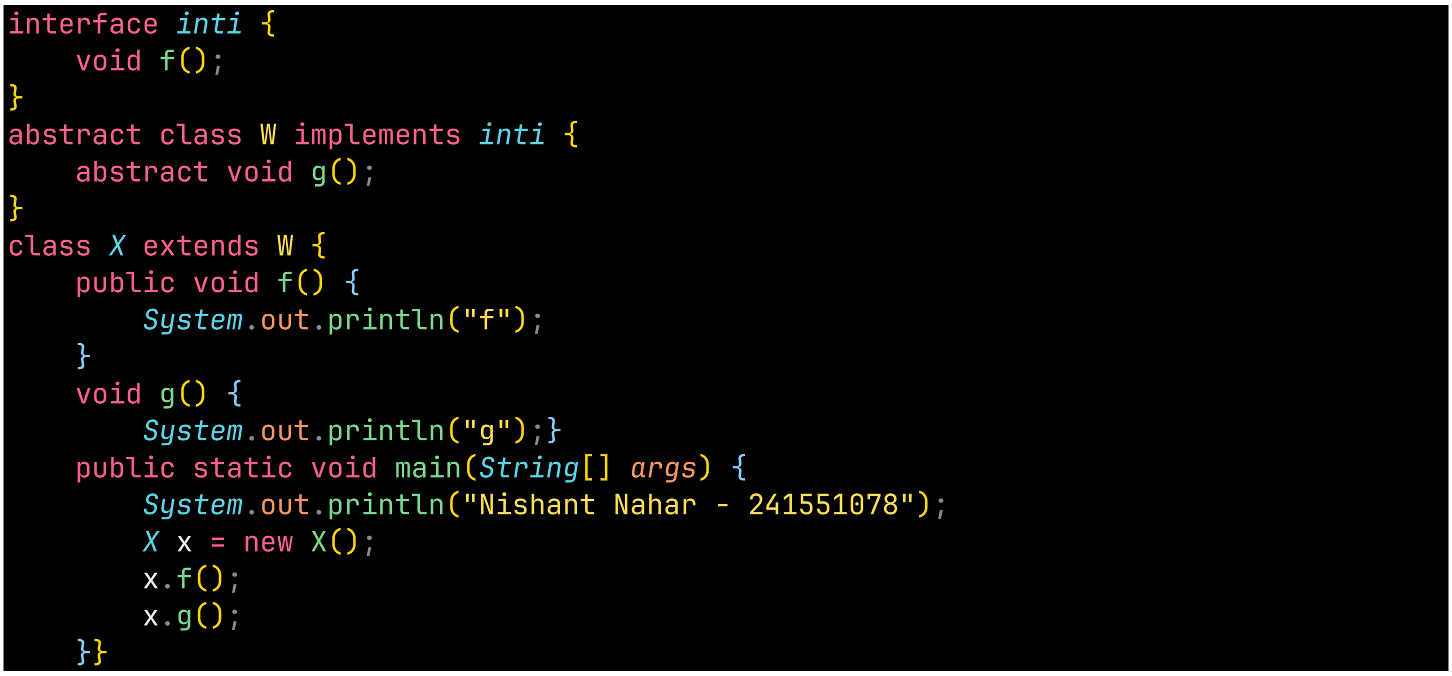
```java
abstract class U {
    void f() {
        System.out.println("ok");
    }}
class V1 {
    public static void main(String[] a) {
        System.out.println("Nishant Nahar - 241551078");
        U u = new U();
        u.f();
    }}
```

OUTPUT



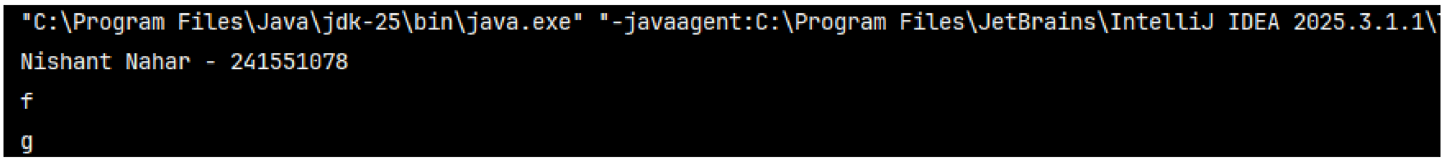
```
PS B:\4th_sem\java_lab\assignment_6> java assign_8.java
assign_8.java:10: error: U is abstract; cannot be instantiated
        U u = new U();
              ^
1 error
error: compilation failed
```

9. Write a program where an abstract class implements an interface and a subclass completes the implementation.



```java
interface inti {
    void f();
}
abstract class W implements inti {
    abstract void g();
}
class X extends W {
    public void f() {
        System.out.println("f");
    }
    void g() {
        System.out.println("g");}
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        X x = new X();
        x.f();
        x.g();
    }}
```

OUTPUT

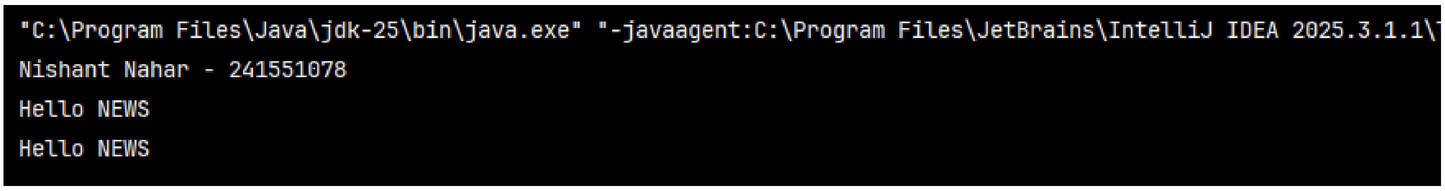


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
f
g
```

10. Show how abstract classes help avoid code duplication using inheritance.

```java
abstract class ani {
    void bre() {
        System.out.println("Hello NEWS");
    }}
class dni extends ani {}
class cni extends ani {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        dni d = new dni();
        cni c = new cni();
        d.bre();
        c.bre();
    }}
```

OUTPUT

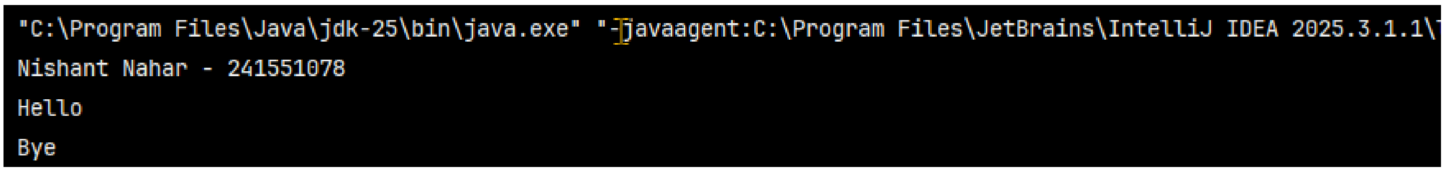


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Hello NEWS
Hello NEWS
```

11. Write a Java program to demonstrate:
    a. Multiple inheritance using interfaces
    b. Conflict resolution of default methods.

```java
interface ina {
    void hello();
}
interface inb {
    void bye();
}
class MyClass implements ina, inb {
    public void hello() {
        System.out.println("Hello");
    }
    public void bye() {
        System.out.println("Bye");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        MyClass obj = new MyClass();
        obj.hello();
        obj.bye();
    }}
```

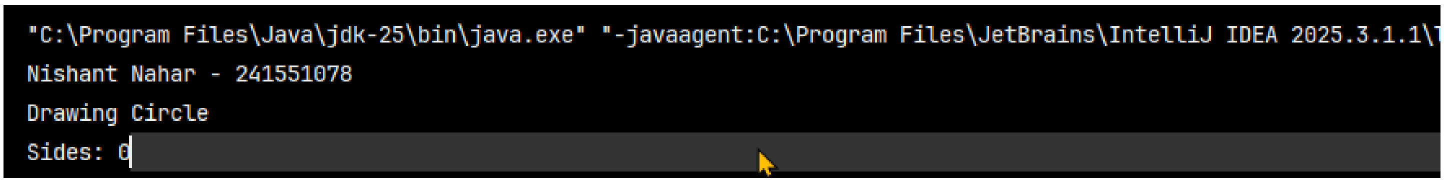
OUTPUT



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

12. Create an interface with:
    a. Constants
    b. Abstract methods and show how they are accessed in implementing classes.

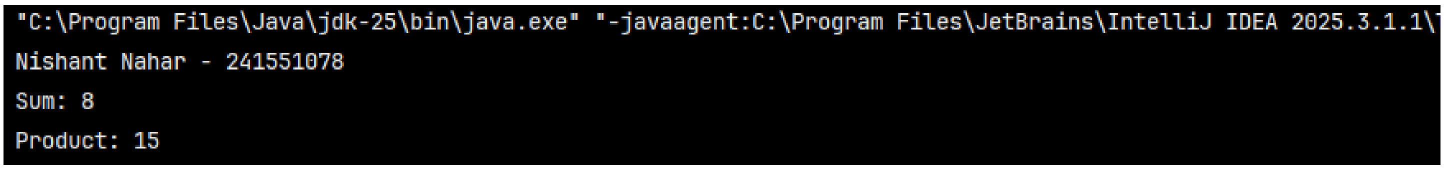
```java
interface shp {
    int sd = 0;

    void draw();
}
class crl implements shp {
    public void draw() {
        System.out.println("Drawing Circle");
        System.out.println("Sides: " + sd);
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        crl c = new crl();
        c.draw();
    }}
```
OUTPUT



```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Drawing Circle
Sides: 0
```

13. Write a program to demonstrate functional interfaces and lambda expressions.

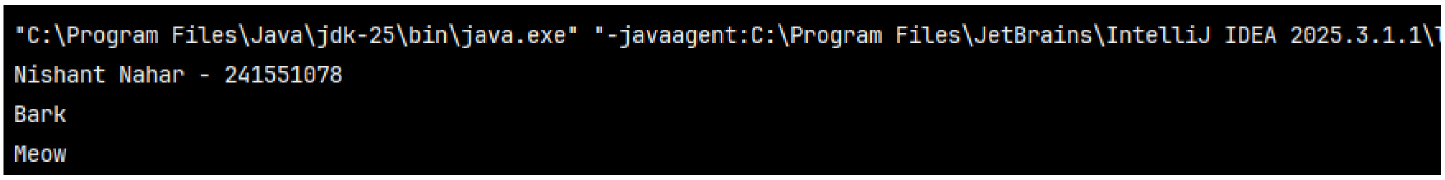
```java
interface Operation {
    int calculate(int a, int b);
}
class LambdaExample {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Operation add = (a, b) -> a + b;
        Operation multiply = (a, b) -> a * b;
        System.out.println("Sum: " + add.calculate(5, 3));
        System.out.println("Product: " + multiply.calculate(5, 3));
    }}
```
OUTPUT



```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Sum: 8
Product: 15
```

14. Show that an interface reference can refer to different implementing class objects and invoke overridden methods.

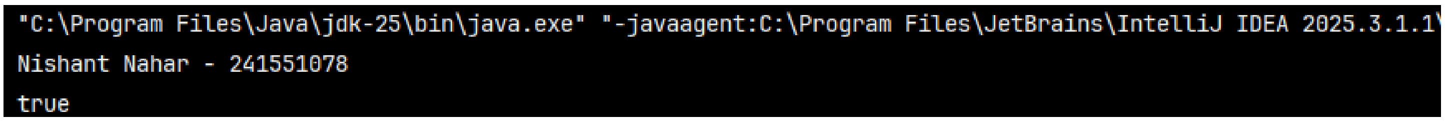
```java
interface assi_1 {
    void sound();
}
class d1 implements assi_1 {
    public void sound() {
        System.out.println("Bark");
    }}
class c1 implements assi_1 {
    public void sound() {
        System.out.println("Meow");
    }
public static void main(String[] args)
        System.out.println("Nishant Nahar - 241551078");
        assi_1 ar;
        ar = new d1();
        ar.sound();
        ar = new c1();
        ar.sound();
    }}
```
OUTPUT



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

15. Write a program to demonstrate marker interfaces and their use in Java.

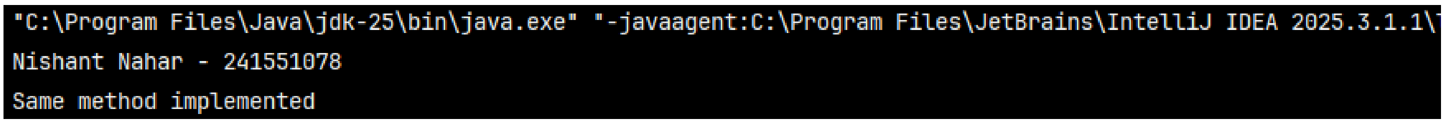
```
interface Marked {}
class pp implements Marked {}
class Test15 {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar – 241551078");
        pp p = new pp();
        System.out.println(p instanceof Marked);
    }}
```
OUTPUT



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

16. Create two interfaces with the same method signature and implement them in a single class.

```
interface inta {
    void show();
}
interface intb {
    void show();
}
class clc implements inta, intb {
    public void show() {
        System.out.println("Same method implemented");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        clc obj = new clc();
        obj.show();
    }}
```
OUTPUT

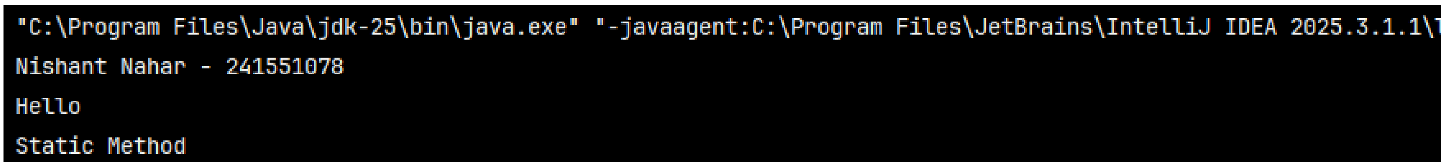


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Same method implemented
```

17. Demonstrate default and static methods in interfaces.

```java
interface myia {
    default void helo() {
        System.out.println("Hello");
    }
    static void msgg() {
        System.out.println("Static Method");
    }}
class deda implements myia {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        deda d = new deda();
        d.helo();
        myia.msgg();
    }}
```

OUTPUT

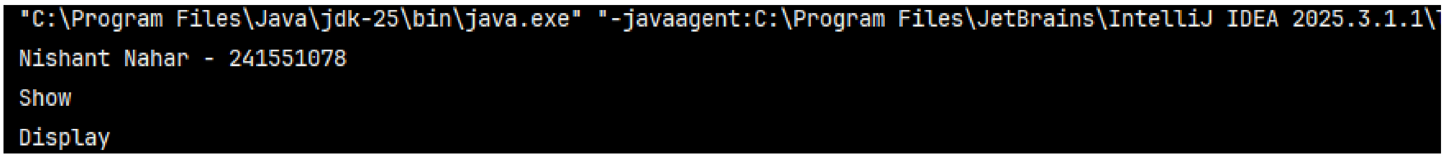


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Hello
Static Method
```

18. Write a program where an interface extends another interface and a class implements the child interface.

```java
interface pp1 {
    void show();
}
interface cc1 extends pp1 {
    void display();
}
 class tt1 implements cc1 {
    public void show() {
        System.out.println("Show");
    }
    public void display() {
        System.out.println("Display");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        tt1 t = new tt1();
        t.show();
        t.display();
    }}
```

OUTPUT

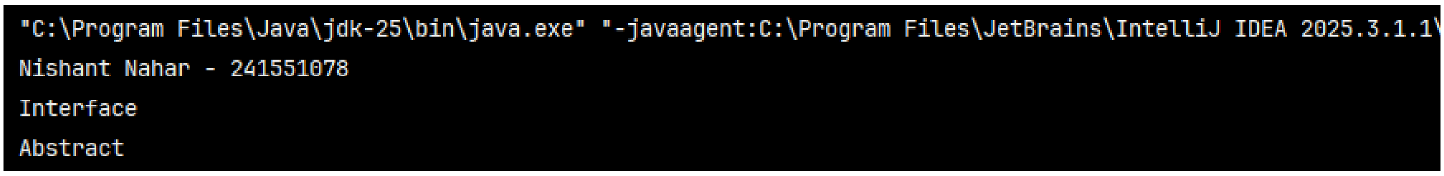


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Show
Display
```

19. Show the difference between interface and abstract class using a program.

```java
interface Ia {
    void m1();
}
abstract class Ib {
    abstract void m2();
}
class a19 extends Ib implements Ia {
    public void m1() {
        System.out.println("Interface");
    }
    public void m2() {
        System.out.println("Abstract");
    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        a19 t = new a19();
        t.m1();
        t.m2();
    }}
```

OUTPUT

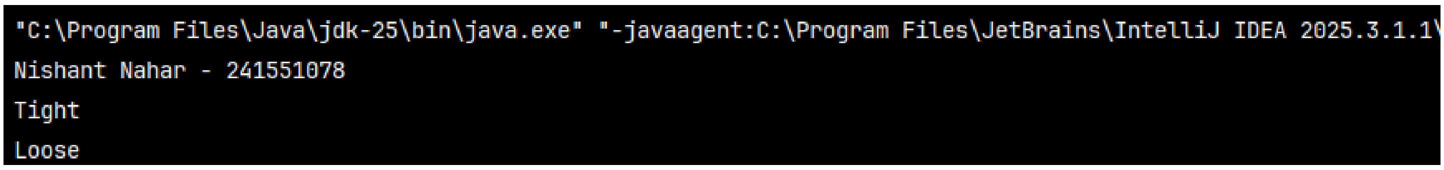


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 241551078
Interface
Abstract
```

20. Write a program to demonstrate tight vs loose coupling using interfaces.

```
class deed {
    static class K {
        void type() {
            System.out.println("Tight");
        }    }
    static class CT {
        void work() {
            new K().type();
        }    }
    interface I {
        void input();
    }
    static class KL implements I {
        public void input() {
            System.out.println("Loose");
        }    }
    static class CL {
        void work(I i) {
            i.input();
        }    }
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        new CT().work();
        new CL().work(new KL());
    }}
```

OUTPUT

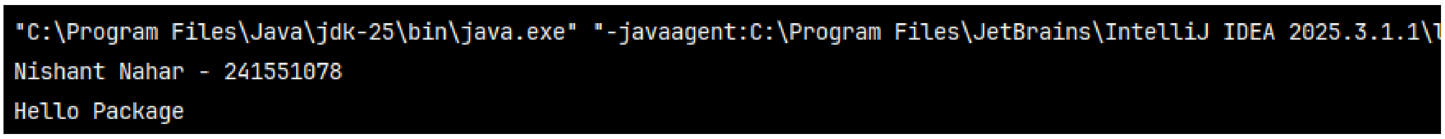


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1
Nishant Nahar - 241551078
Tight
Loose
```

21. Create a user-defined package, compile it, and access its classes from another package.

```java
package mypack;
public class assign_21 {
    public void msg() {
        System.out.println("Hello Package");
    }}
import mypack.assign_21;
class Test21 {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        assign_21 h = new assign_21();
        h.msg();
    }}
```

OUTPUT

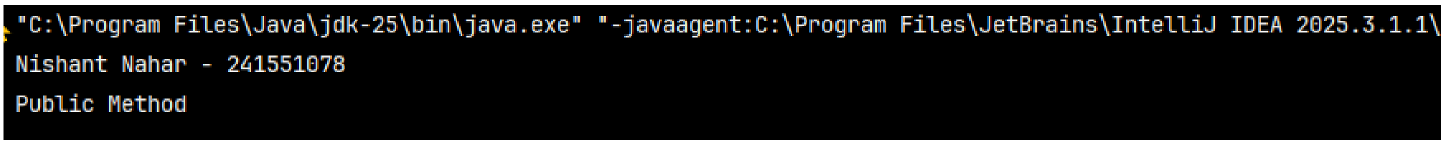


```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\l
Nishant Nahar - 241551078
Hello Package
```

22. Write a program to demonstrate access control (public, default, protected) across packages.

```java
package pack1;
public class assign_22 {
    public static void show() {
        System.out.println("Public Method");
    }}
import pack1.assign_22;
class Test22 {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        assign_22 a = new assign_22();
        assign_22.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
Public Method
```

23. Create a package containing both interface and abstract class, and implement them in another package.

    a. P1

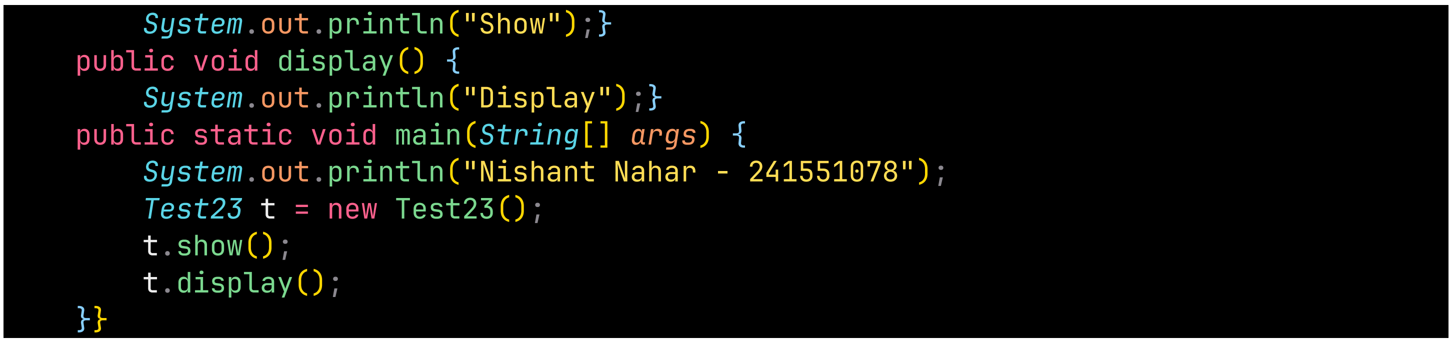
```java
package p1;
public interface assign_23 {
  void show();
}
```

    b. P1

```java
package p1;
public abstract class assign_23_1 {
    public abstract void display();
}
```

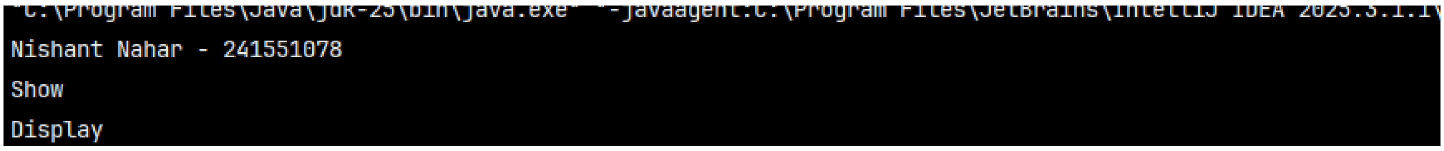
    c. Code

```java
class Test23 extends p1.assign_23_1 implements p1.assign_23 {
    public void show() {
```



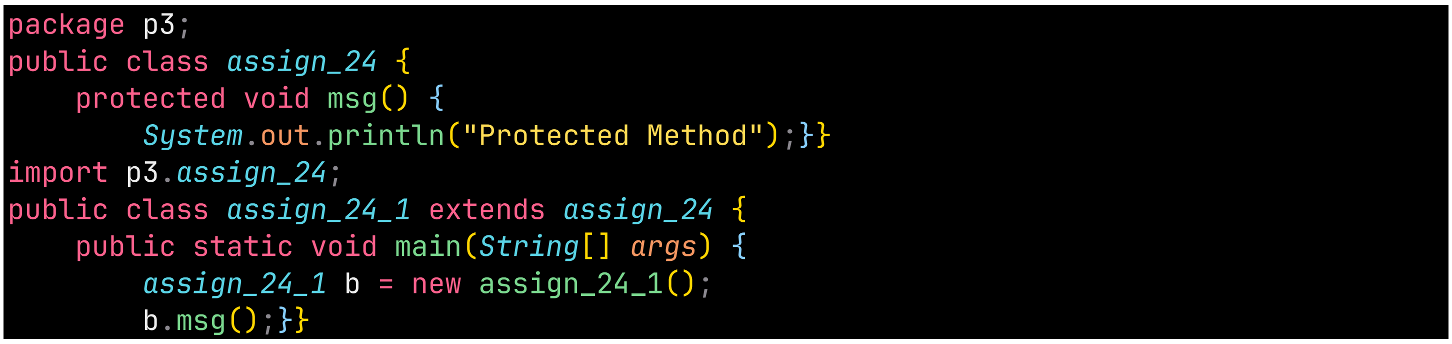
```java
        System.out.println("Show");}
    public void display() {
        System.out.println("Display");}
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        Test23 t = new Test23();
        t.show();
        t.display();
    }}
```

OUTPUT



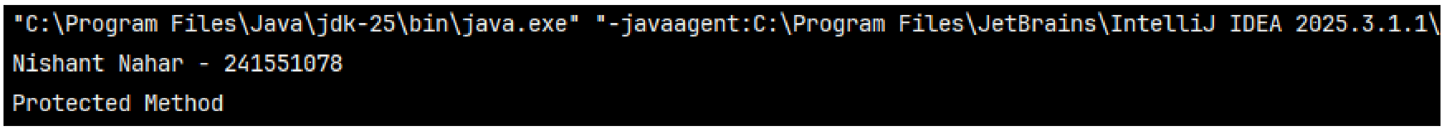
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Show
Display
```

24. Write a program to show how protected members behave in inheritance across packages.



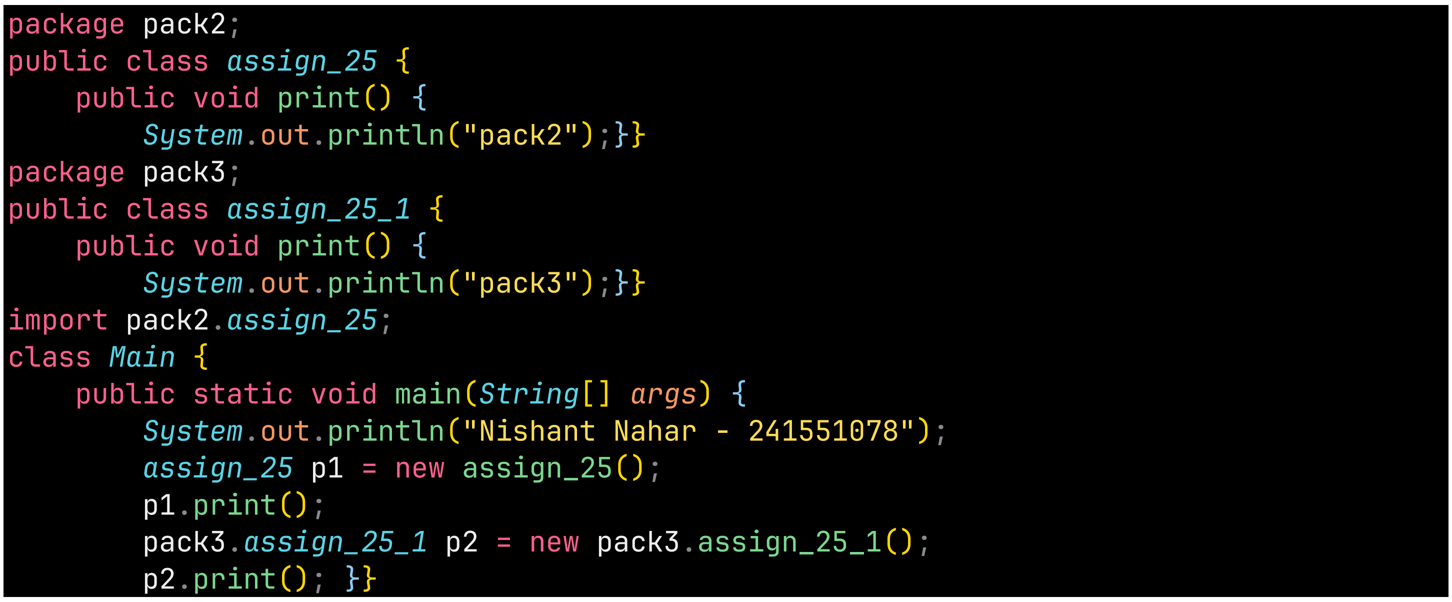
```java
package p3;
public class assign_24 {
    protected void msg() {
        System.out.println("Protected Method");}}
import p3.assign_24;
public class assign_24_1 extends assign_24 {
    public static void main(String[] args) {
        assign_24_1 b = new assign_24_1();
        b.msg();}}
```

OUTPUT



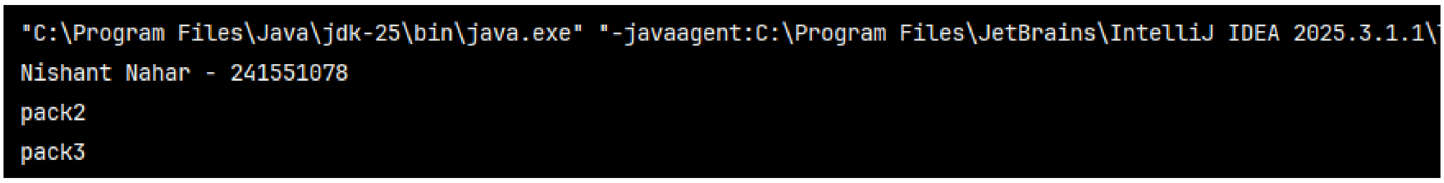
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Protected Method
```

25. Demonstrate name conflict resolution using packages.



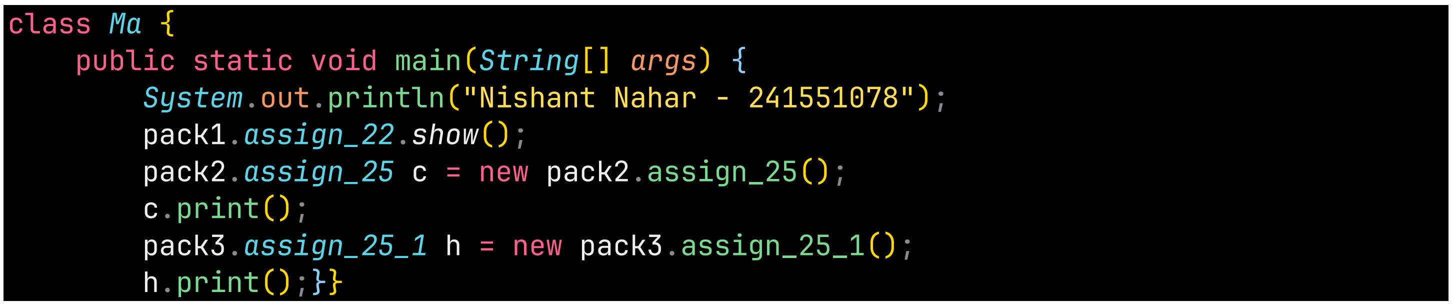
```java
package pack2;
public class assign_25 {
    public void print() {
        System.out.println("pack2");}}
package pack3;
public class assign_25_1 {
    public void print() {
        System.out.println("pack3");}}
import pack2.assign_25;
class Main {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        assign_25 p1 = new assign_25();
        p1.print();
        pack3.assign_25_1 p2 = new pack3.assign_25_1();
        p2.print(); }}
```

OUTPUT



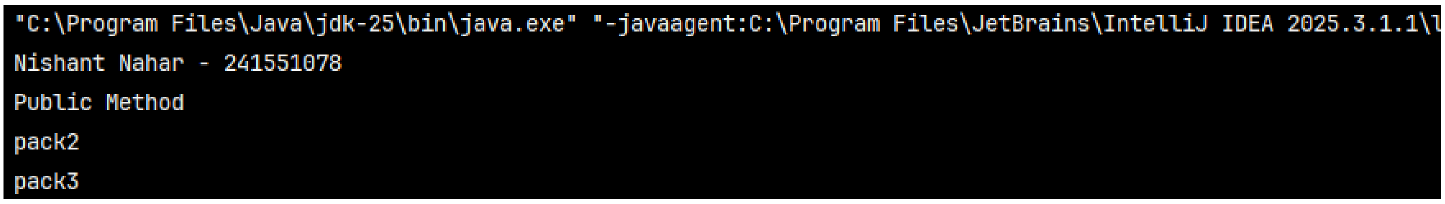
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
pack2
pack3
```

26. Write a program to organize a project using multiple packages and explain its benefits.



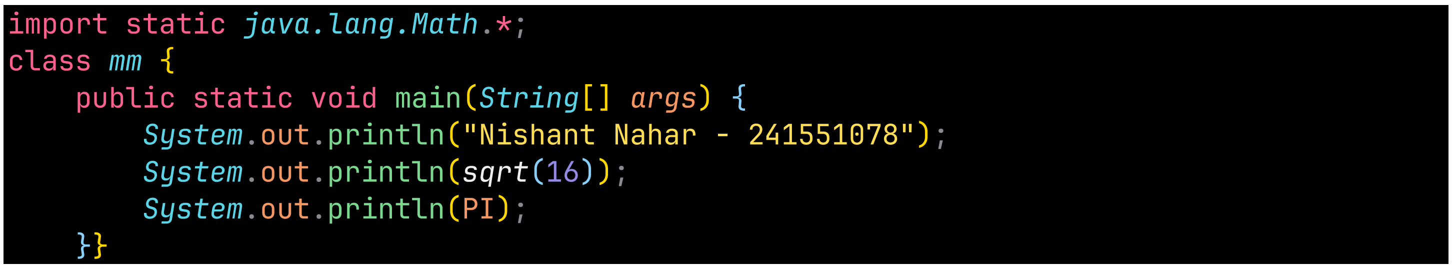
```java
class Ma {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        pack1.assign_22.show();
        pack2.assign_25 c = new pack2.assign_25();
        c.print();
        pack3.assign_25_1 h = new pack3.assign_25_1();
        h.print();}}
```

OUTPUT



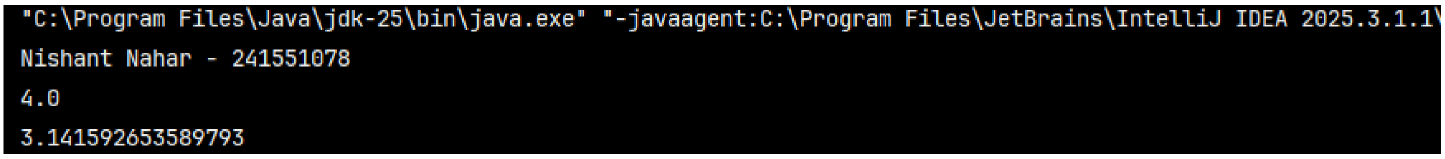
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
Public Method
pack2
pack3
```

27. Show how to use static import and explain when it is useful.



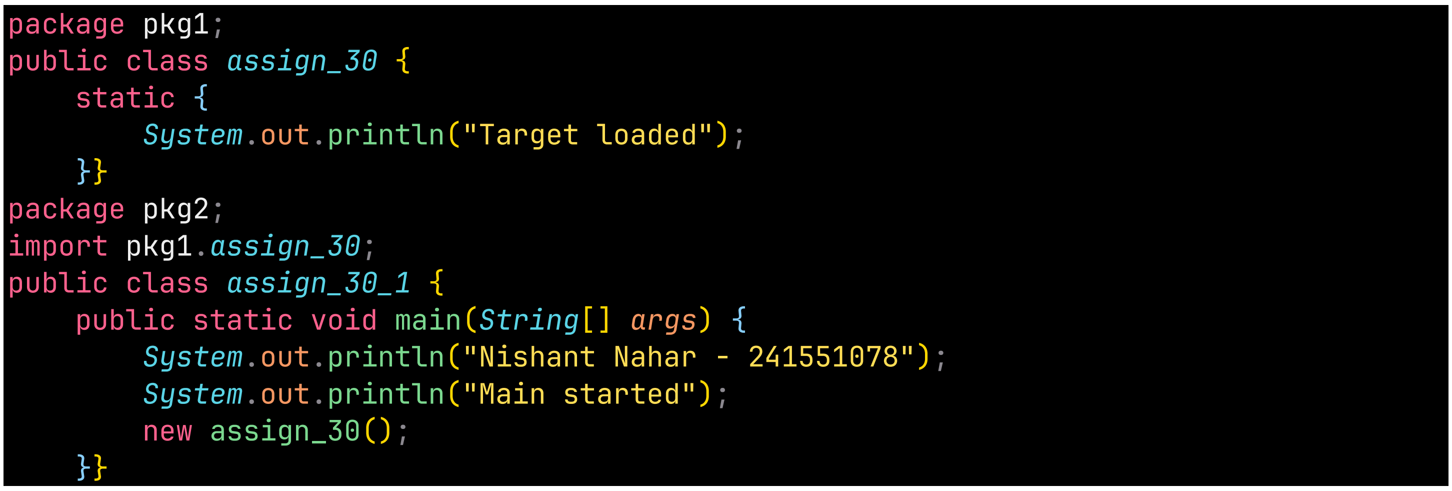
```java
import static java.lang.Math.*;
class mm {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        System.out.println(sqrt(16));
        System.out.println(PI);
    }}
```

OUTPUT



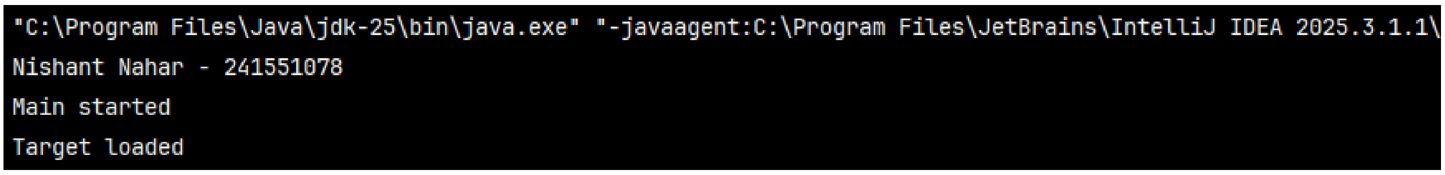
```
"C:\Program Files\Java\jdk-25\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.1.1\
Nishant Nahar - 241551078
4.0
3.141592653589793
```

28. Write a program to show class loading behavior across packages.



```java
package pkg1;
public class assign_30 {
    static {
        System.out.println("Target loaded");
    }}
package pkg2;
import pkg1.assign_30;
public class assign_30_1 {
    public static void main(String[] args) {
        System.out.println("Nishant Nahar - 241551078");
        System.out.println("Main started");
        new assign_30();
    }}
```

OUTPUT



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