Object Oriented Programming with Java (OOPJ)

Session 8: OOPS Pillar

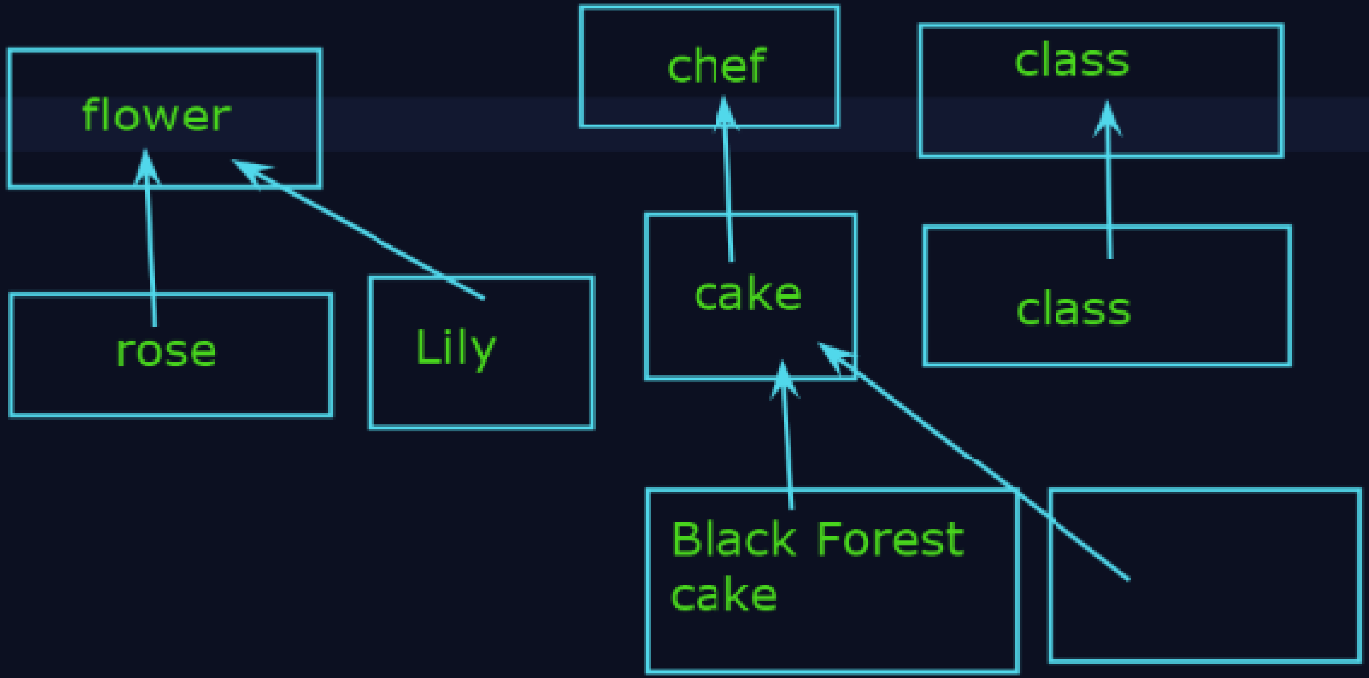
Kiran Waghmare

Sub class (child class) : The class that inherits properties from another cl
Reusability: Using existing methods and filelds of a super class in a subcla

Inheritance:
-It is a mechanism in Java where a class(child class) acquire properties(fi
behaviour(methods) from another class (PArent class).
-It promotes reusability, hierarcical organization and polymorphism
-REpresents IS-A relationship
-Helps avoid redundant code by reusing common functionalities.

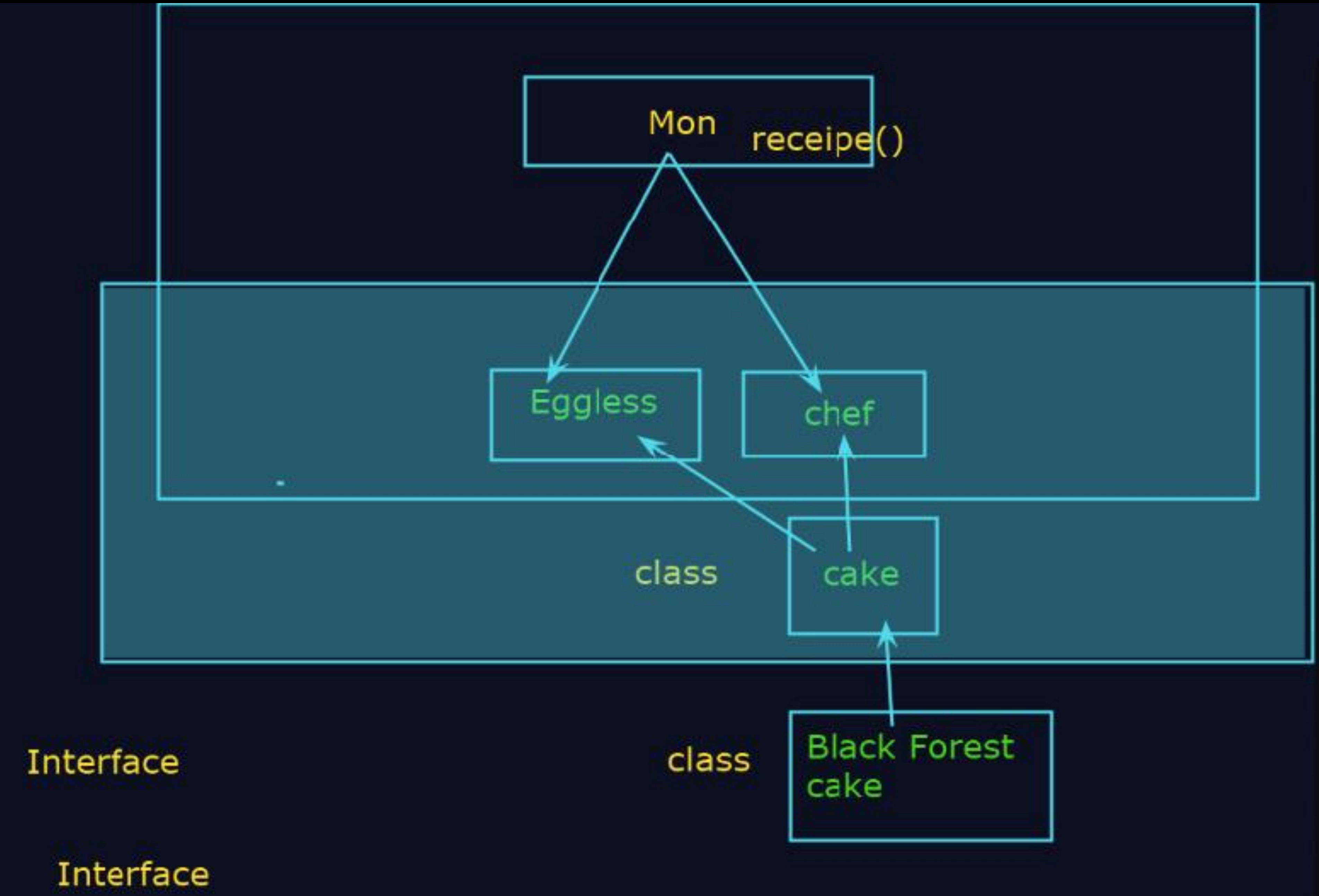
Syntax:

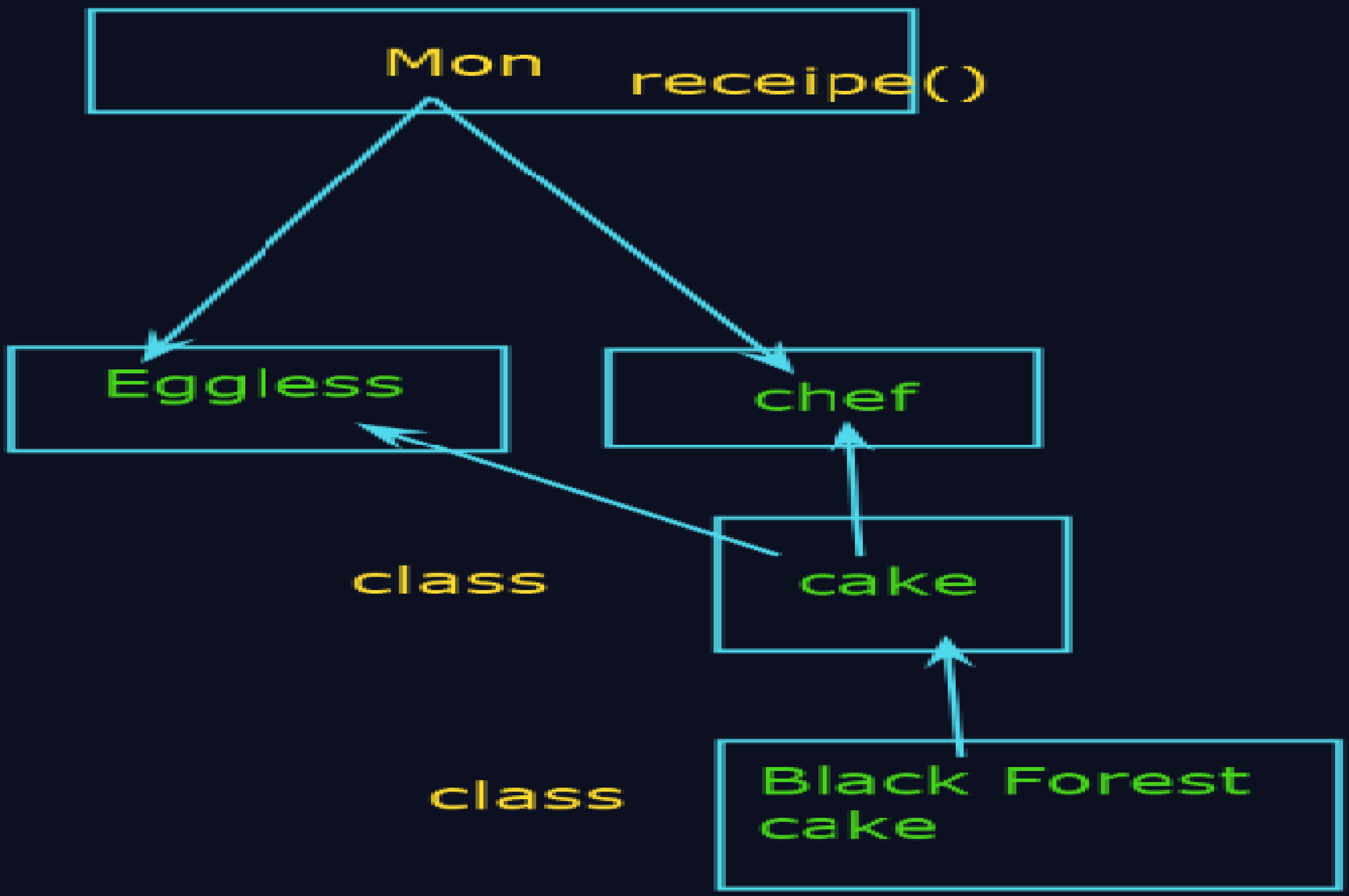
```
class Parent{
}


class Child extends Parent{
}


class Demo extends Child{
}
```

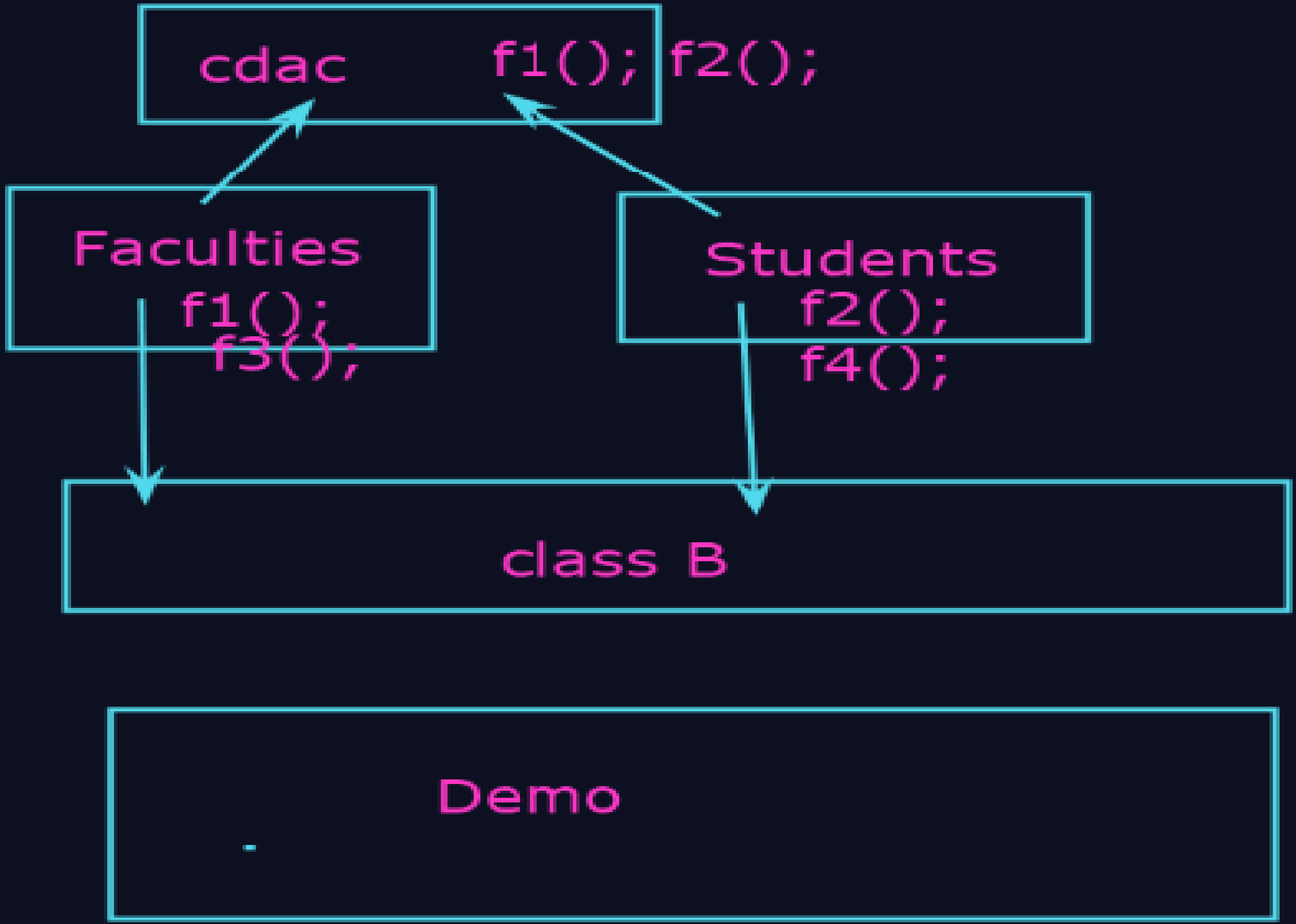


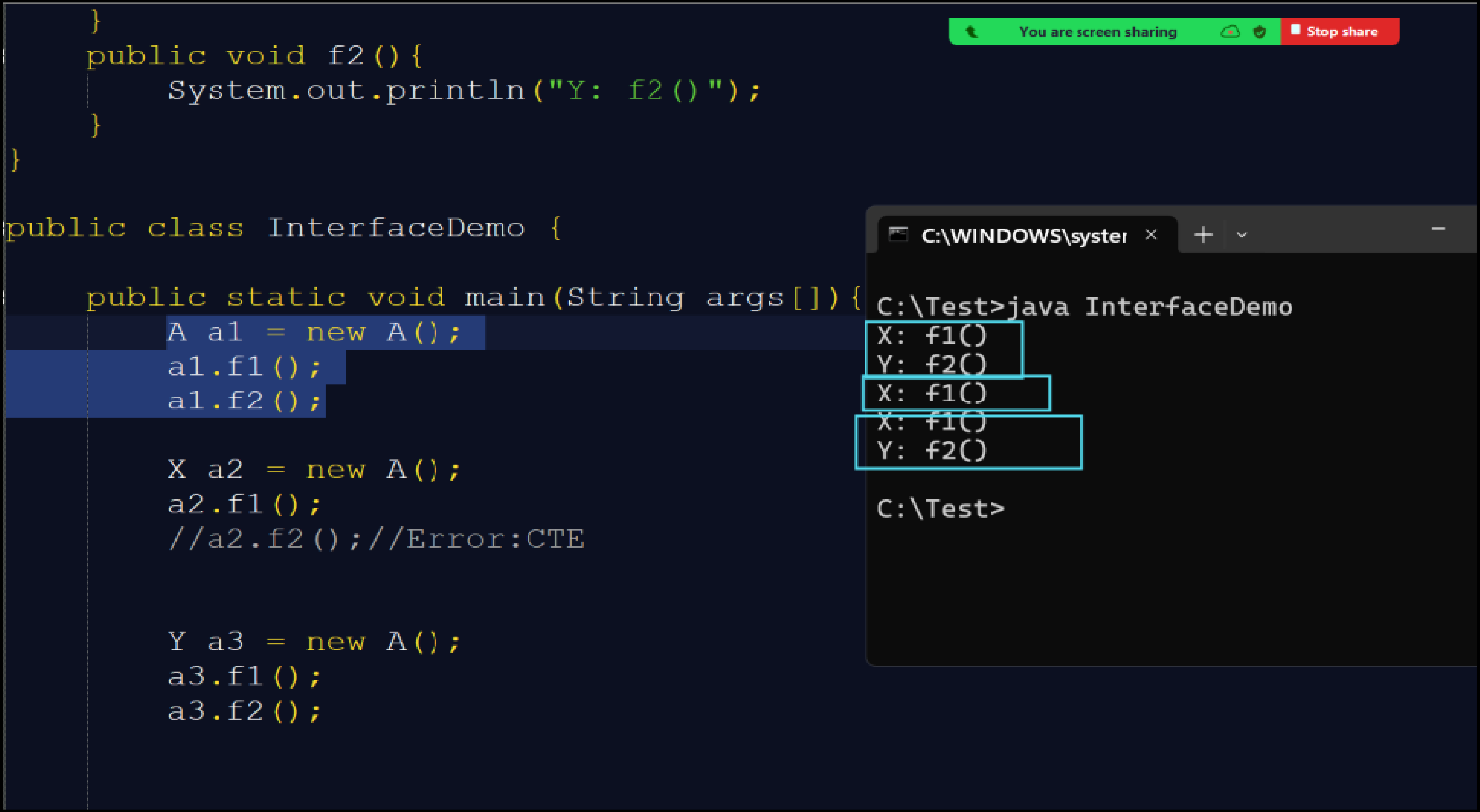
Types of Inheritance:

-Association




Mon
receipe()
Eggless
chef
class
cake
class
Black Forest cake

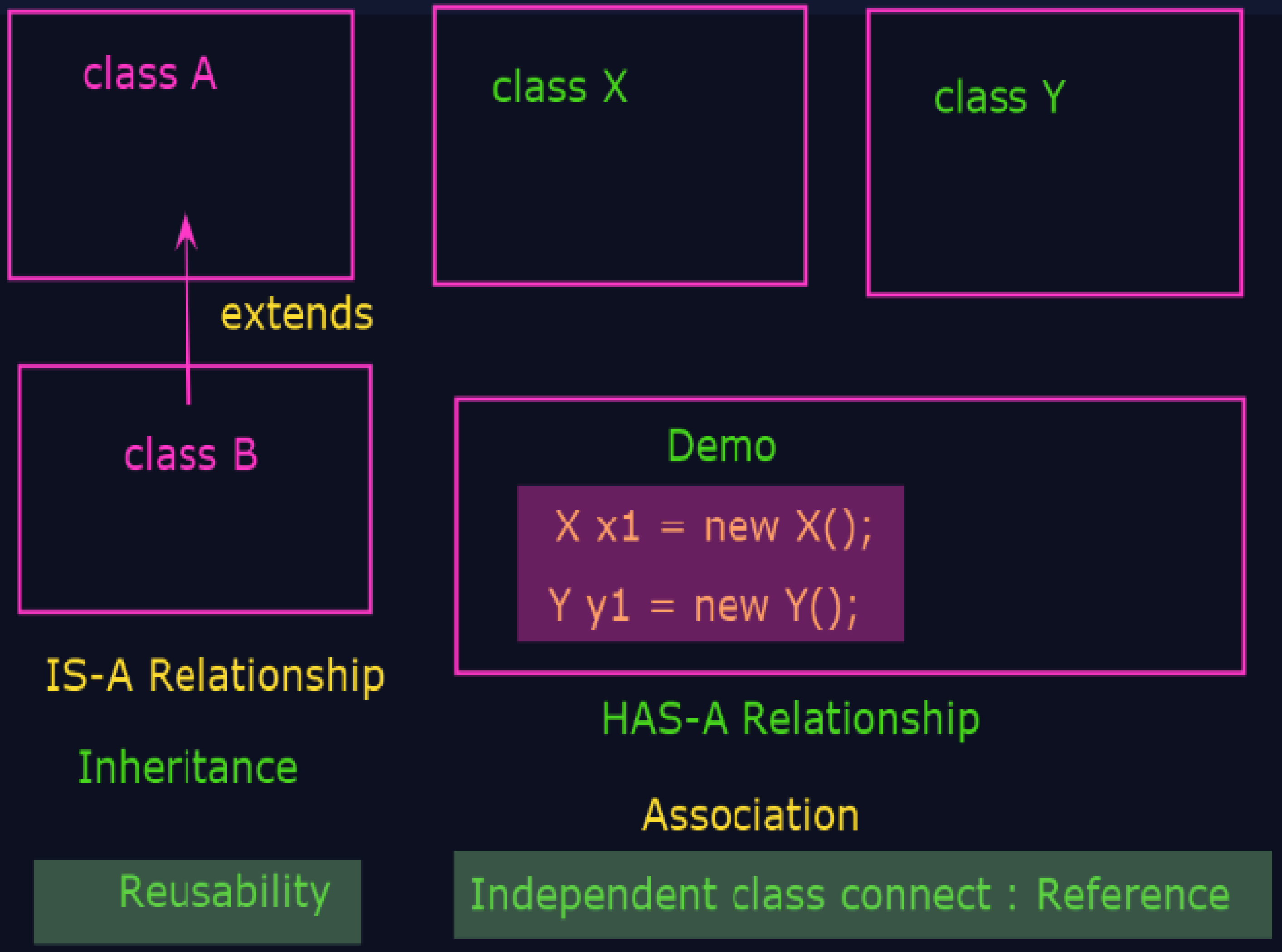

cdac
f1(); f2();
Faculties
f1();
f3();
Students
f2();
f4();
class B
Demo



```
    }
    public void f2(){
        System.out.println("Y: f2()");
    }
}

public class InterfaceDemo {

    public static void main(String args[]){
        A a1 = new A();
        a1.f1();
        a1.f2();

        X a2 = new A();
        a2.f1();
        //a2.f2();//Error:CTE


        Y a3 = new A();
        a3.f1();
        a3.f2();
```

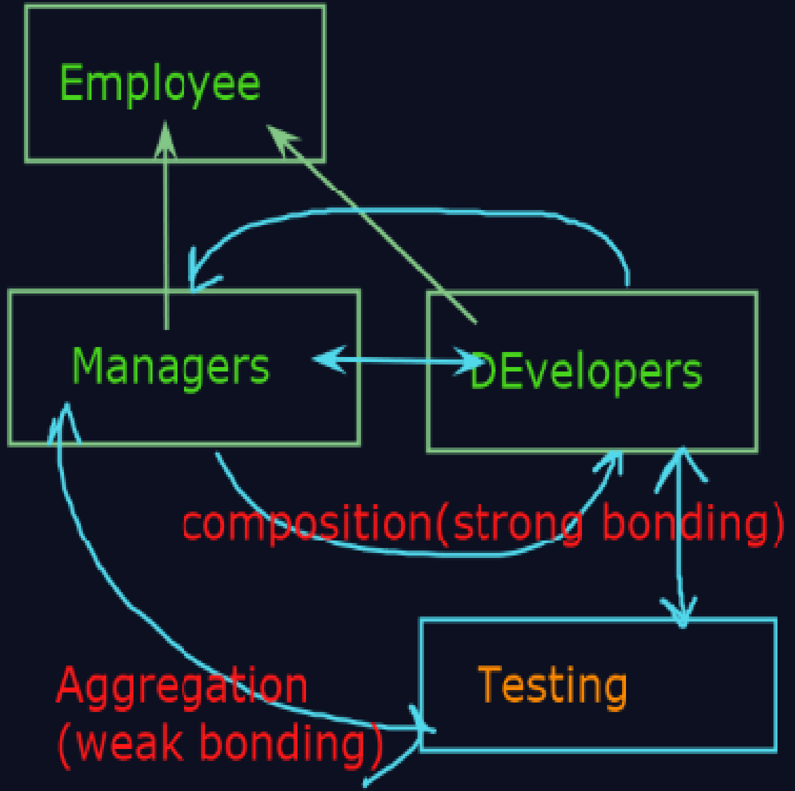
```
C:\Test>java InterfaceDemo
X: f1()
Y: f2()
X: f1()
X: f1()
Y: f2()

C:\Test>
```

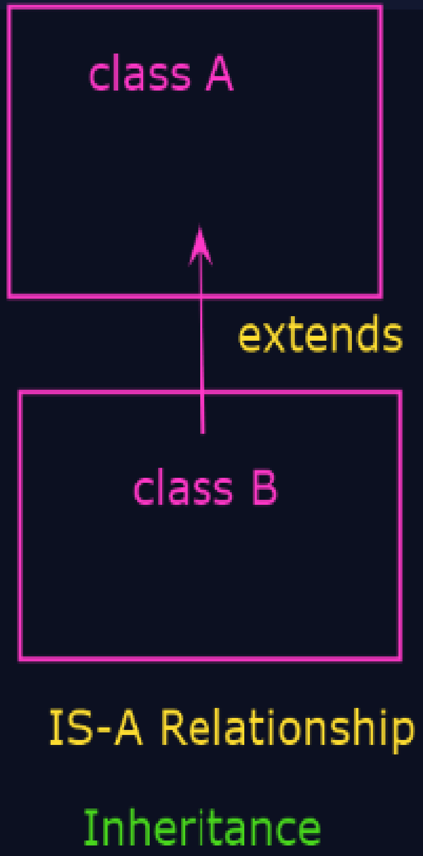
Association:

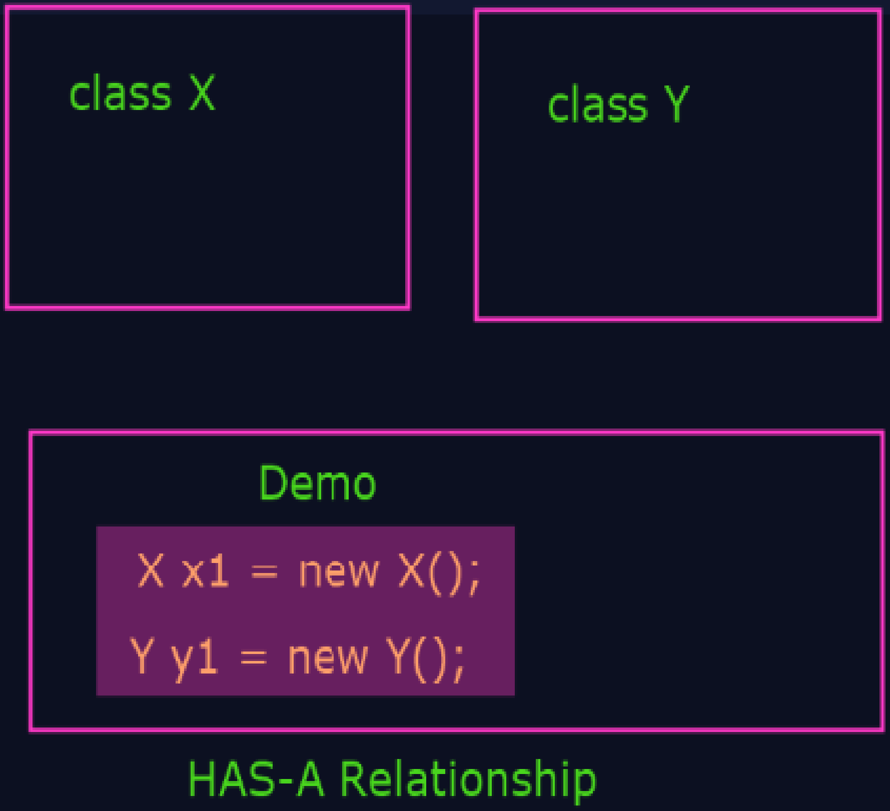


```
X x1 = new X();
Y y1 = new Y();
```

Association:







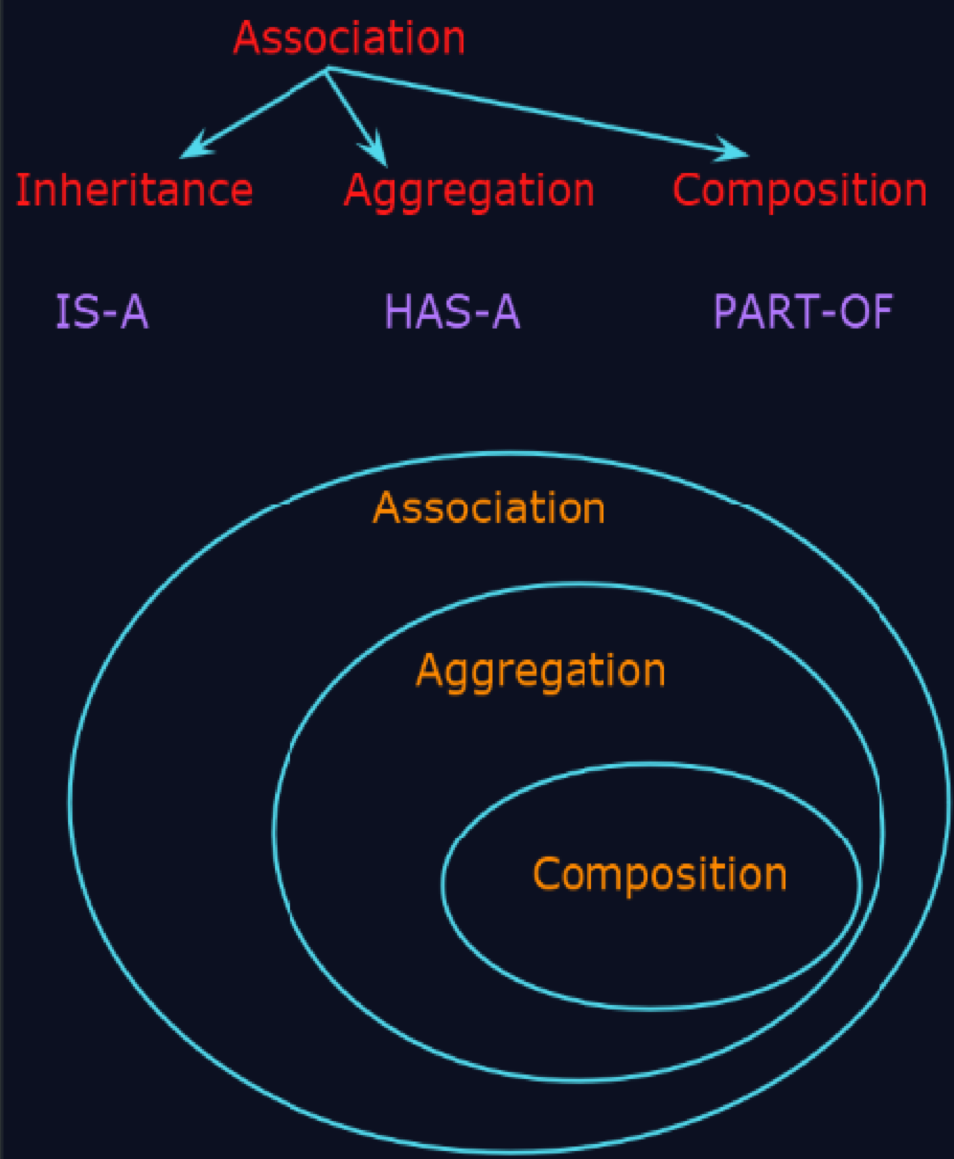
```
X x1 = new X();
Y y1 = new Y();
```

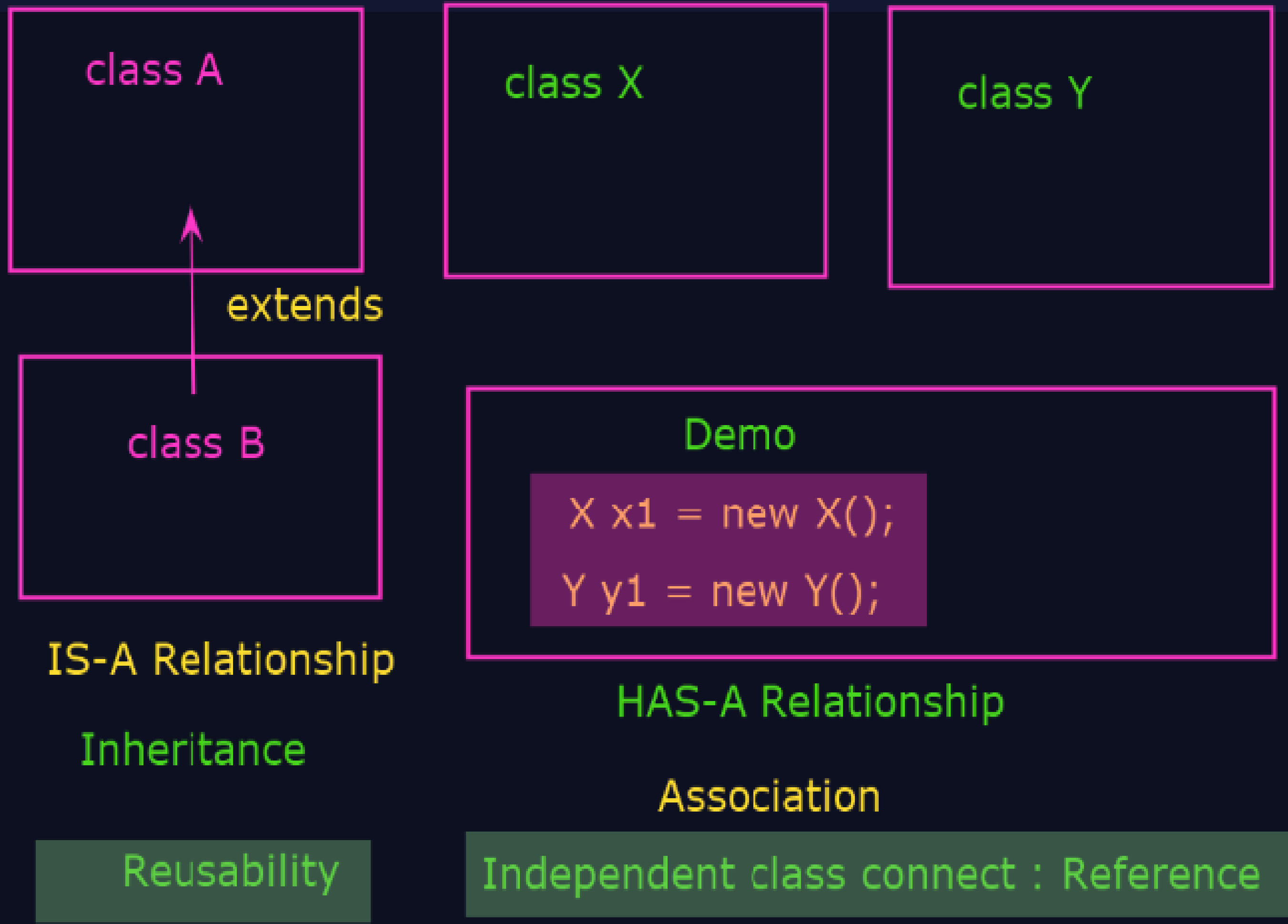
Association

Reusability

Independent class connect : Reference

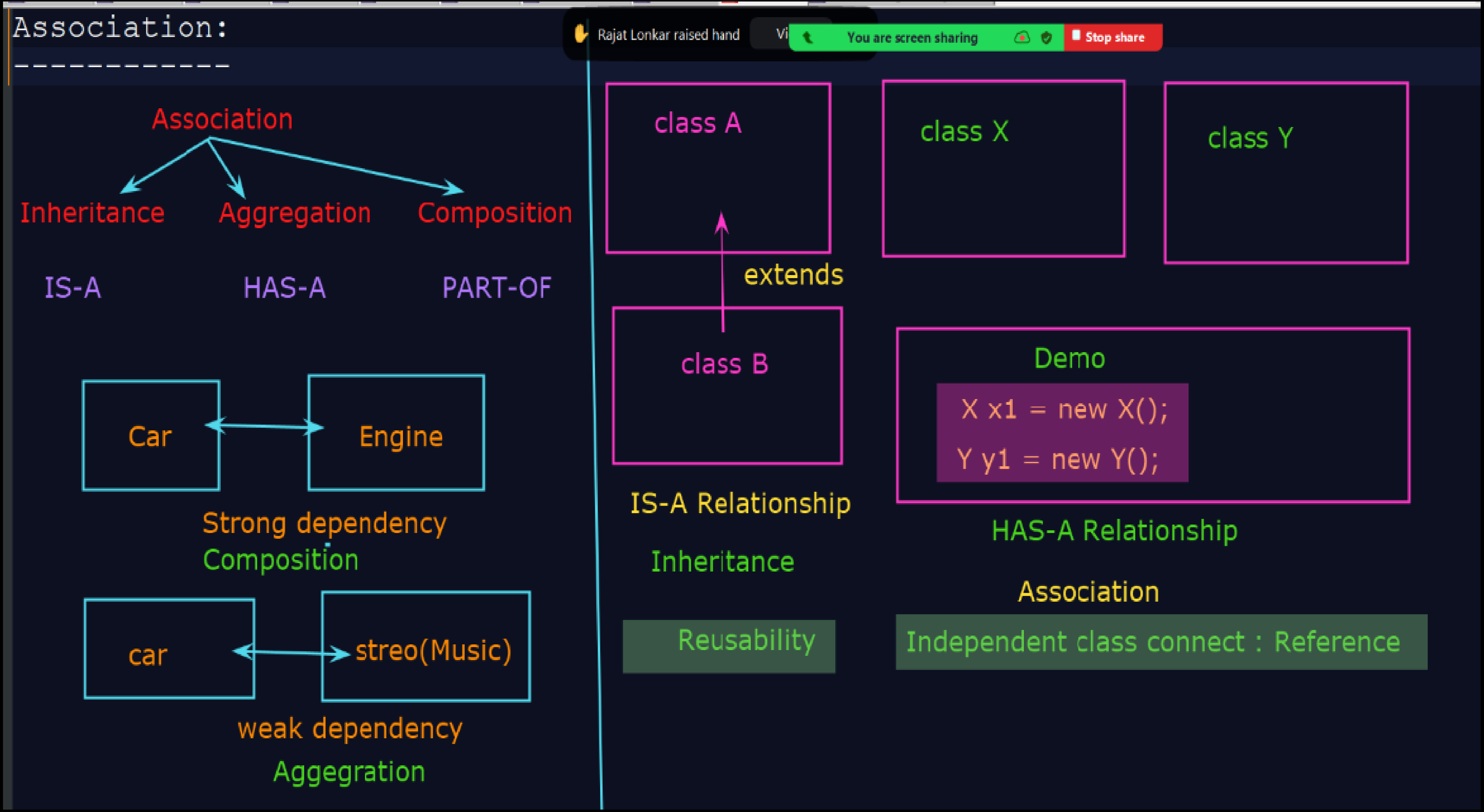
Association:





```
X x1 = new X();
Y y1 = new Y();
```

Independent class connect : Reference



Association:

Association

Inheritance Aggregation Composition

IS-A HAS-A PART-OF

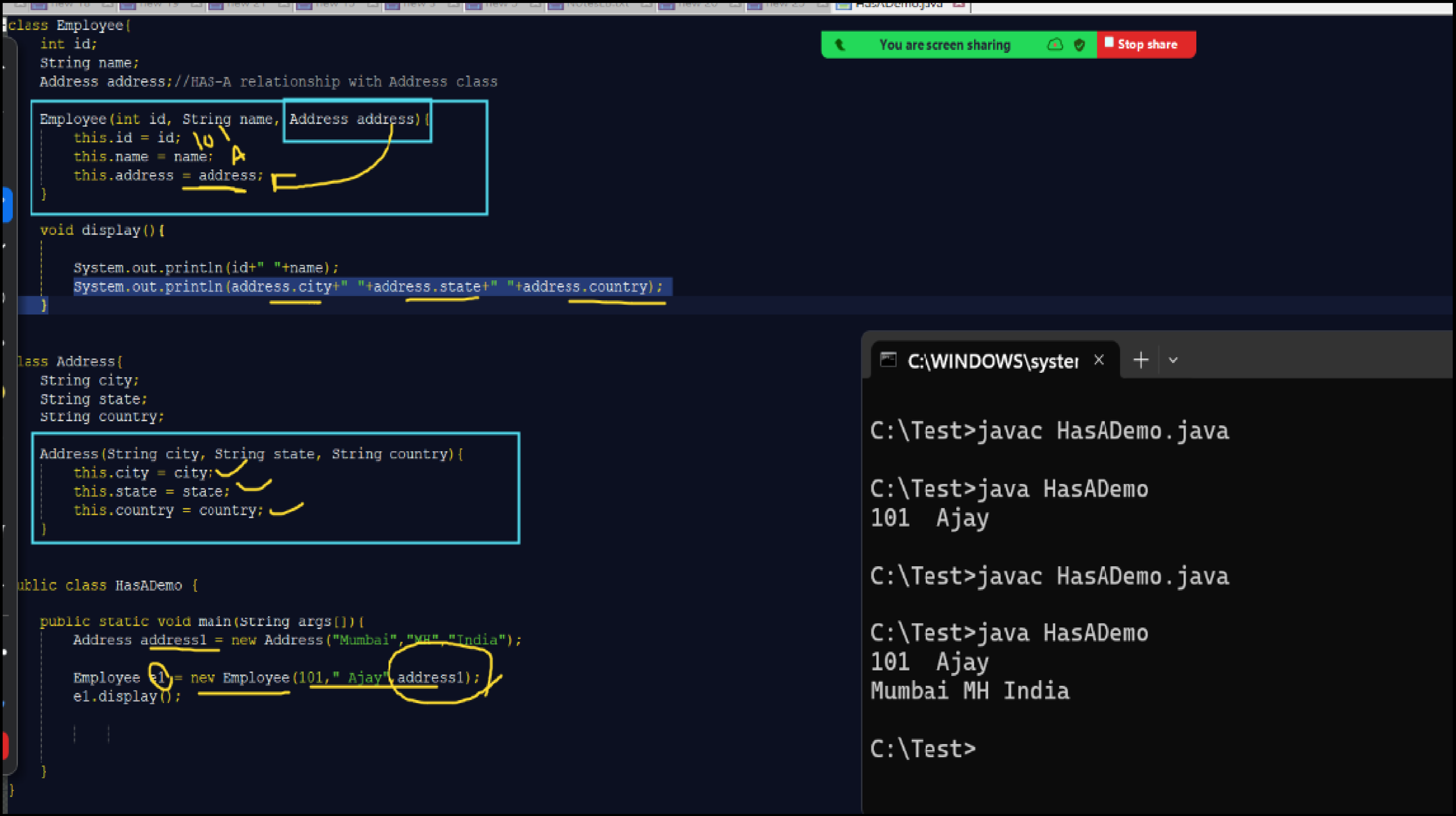
Strong dependency
Composition

weak dependency
Aggegration

```
X x1 = new X();
Y y1 = new Y();
```

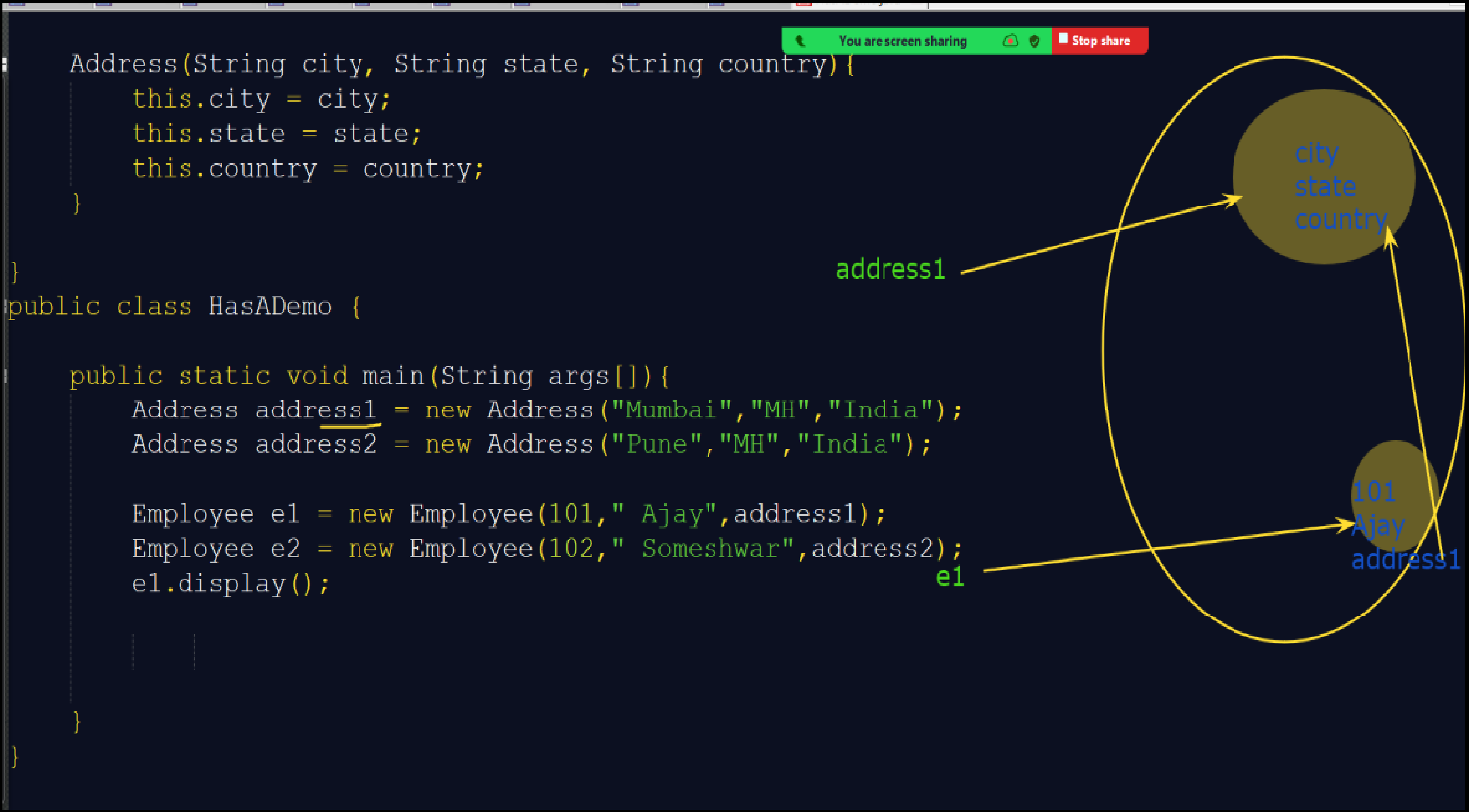
IS-A Relationship
Inheritance
Reusability

HAS-A Relationship
Association
Independent class connect : Reference

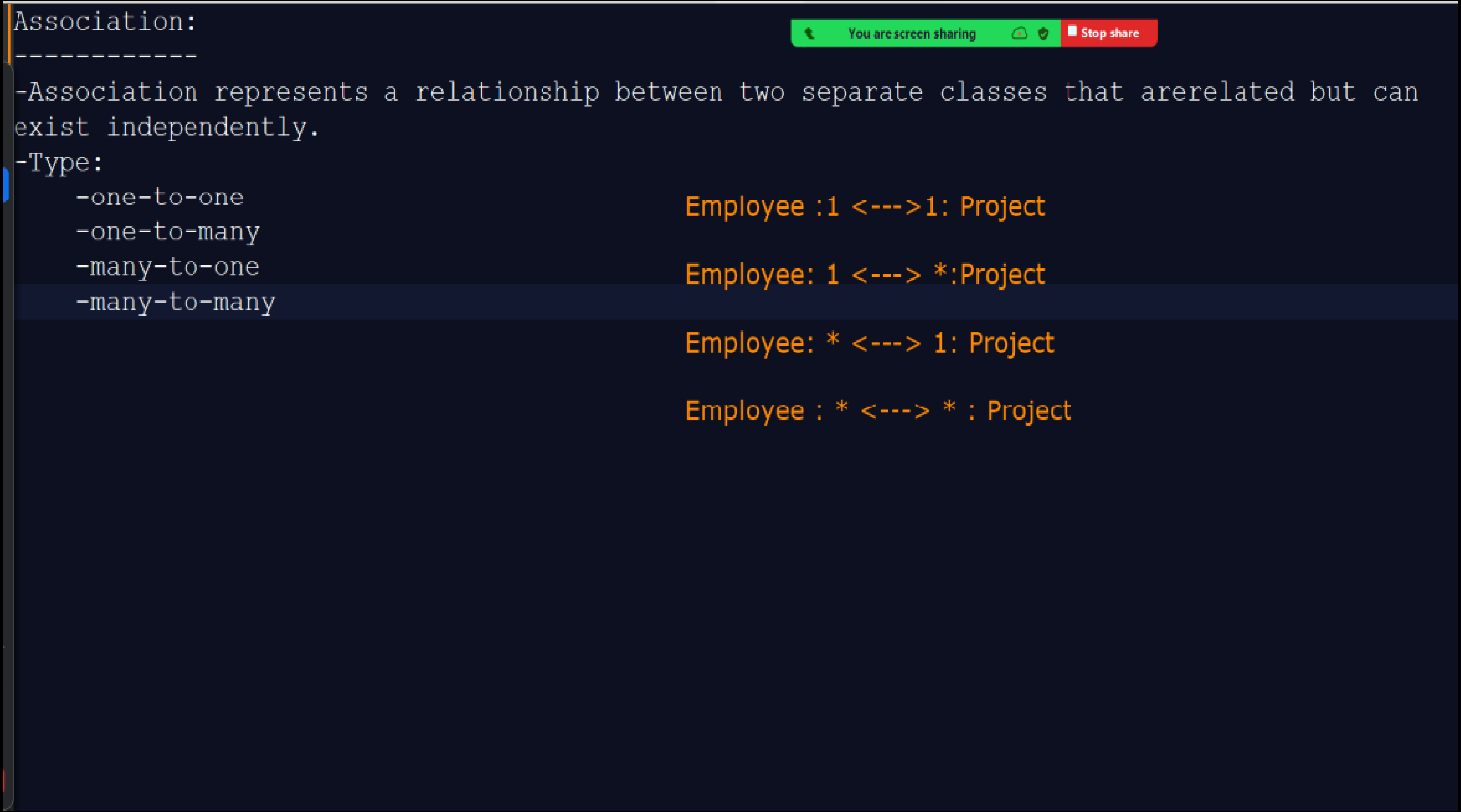


```java
class Employee{
    int id;
    String name;
    Address address;//HAS-A relationship with Address class

    Employee(int id, String name, Address address){
        this.id = id;
        this.name = name;
        this.address = address;
    }

    void display(){

        System.out.println(id+" "+name);
        System.out.println(address.city+" "+address.state+" "+address.country);
    }


lass Address{
    String city;
    String state;
    String country;

    Address(String city, String state, String country){
        this.city = city;
        this.state = state;
        this.country = country;
    }

ublic class HasADemo {

    public static void main(String args[]){
        Address address1 = new Address("Mumbai","MH","India");

        Employee e1 = new Employee(101," Ajay",address1);
        e1.display();



    }
}
```

```
C:\Test>javac HasADemo.java

C:\Test>java HasADemo
101  Ajay

C:\Test>javac HasADemo.java

C:\Test>java HasADemo
101  Ajay
Mumbai MH India

C:\Test>
```



```
    Address(String city, String state, String country){
        this.city = city;
        this.state = state;
        this.country = country;
    }

}
public class HasADemo {

    public static void main(String args[]){
        Address address1 = new Address("Mumbai","MH","India");
        Address address2 = new Address("Pune","MH","India");

        Employee e1 = new Employee(101," Ajay",address1);
        Employee e2 = new Employee(102," Someshwar",address2);
        e1.display();


    }
}
```



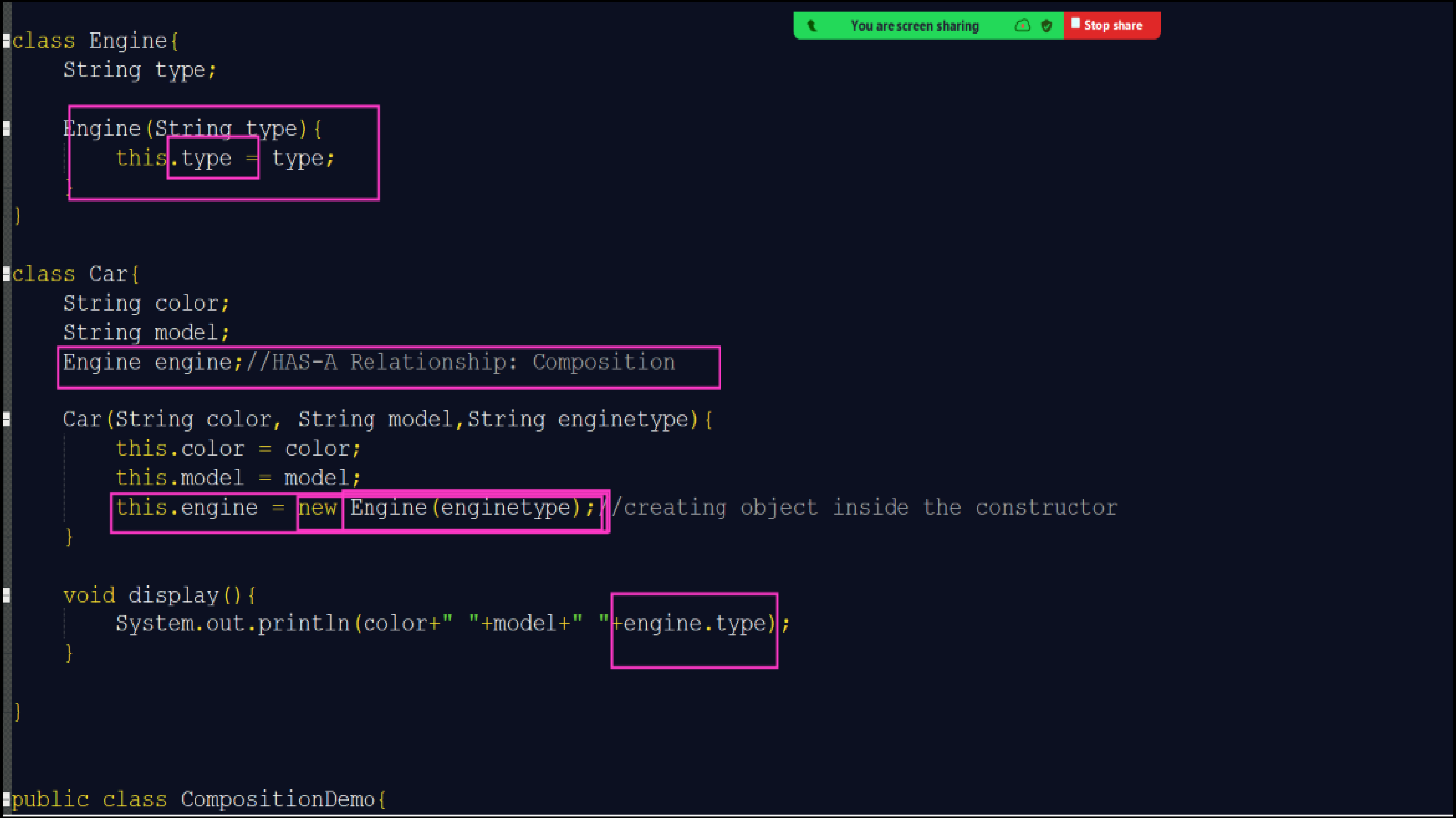
```
Association:
------------
-Association represents a relationship between two separate classes that arerelated but can
exist independently.
-Type:
    -one-to-one
    -one-to-many
    -many-to-one
    -many-to-many
```

Employee :1 <--->1: Project

Employee: 1 <---> *:Project

Employee: * <---> 1: Project

Employee : * <---> * : Project



```java
class Engine{
    String type;

    Engine(String type){
        this.type = type;
    }
}

class Car{
    String color;
    String model;
    Engine engine;//HAS-A Relationship: Composition

    Car(String color, String model,String enginetype){
        this.color = color;
        this.model = model;
        this.engine = new Engine(enginetype);//creating object inside the constructor
    }

    void display(){
        System.out.println(color+" "+model+" "+engine.type);
    }

}


public class CompositionDemo{
```