**Please make sure to save/push all your code in the branch feature-java created in the previous week assignment as part of your github repo rg-assignments**

**Please share your output screenshots in the assignment document along with the github link for each question. Provide an explanation wherever possible as part of your response :-)**



Given:

public class TaxUtil {

double rate = 0.15;

public double calculateTax(double amount) {

return amount \* rate;

}

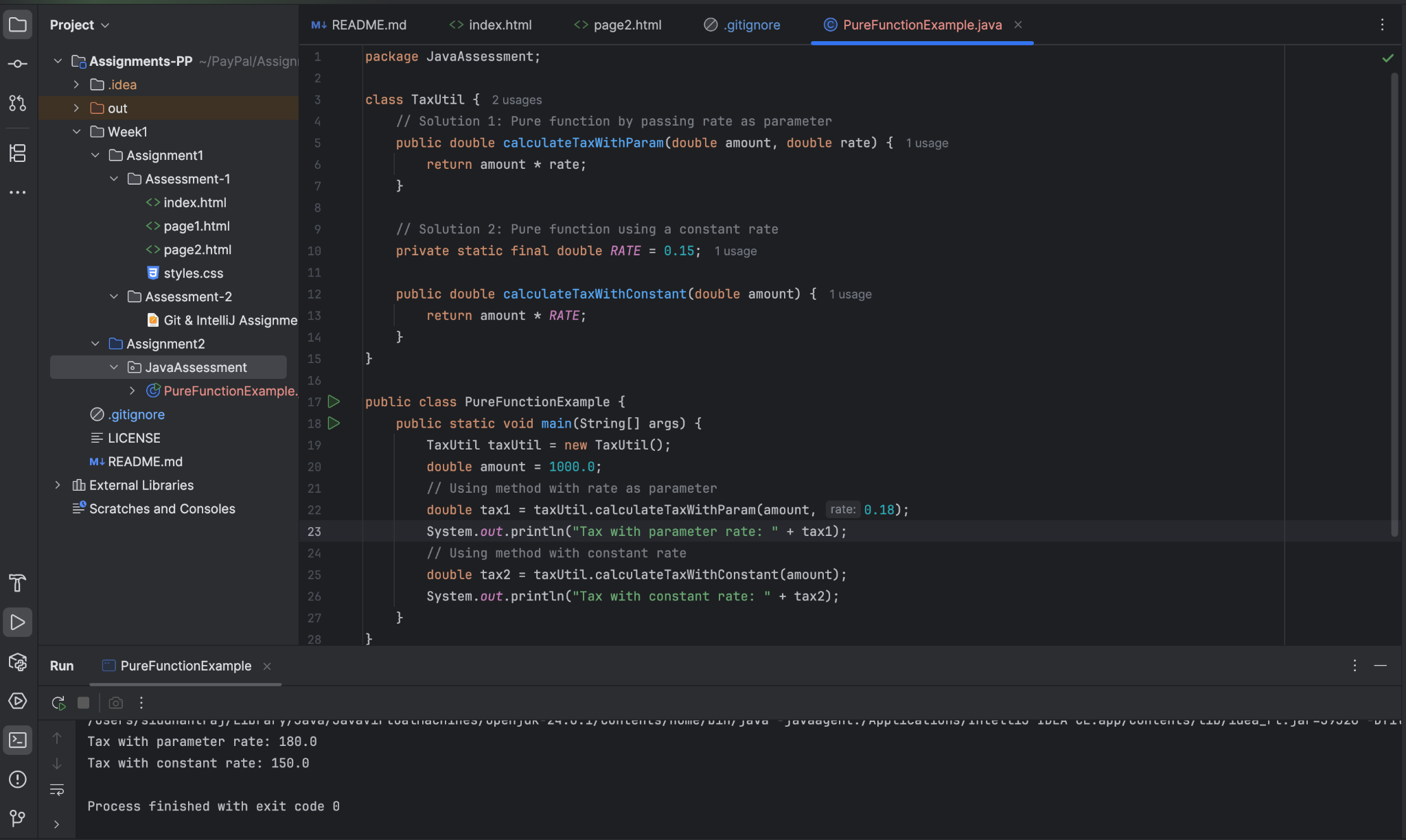
}

Would you consider the method calculateTax() a 'pure function'? Why or why not?

If you claim the method is NOT a pure function, please suggest a way to make it pure.

**Ans :-** No, calculateTax() is not a pure function because it depends on the mutable instance variable rate, which can change and affect the output. There are two ways to make it pure :-

* Pass rate as a parameter
* Make rate a final constant



2)

What will be the output for following code?

class Super

{

static void show()

{

System.out.println("super class show method");

}

static class StaticMethods

{

void show()

{

System.out.println("sub class show method");

}

}

public static void main(String[]args)

{

Super.show();

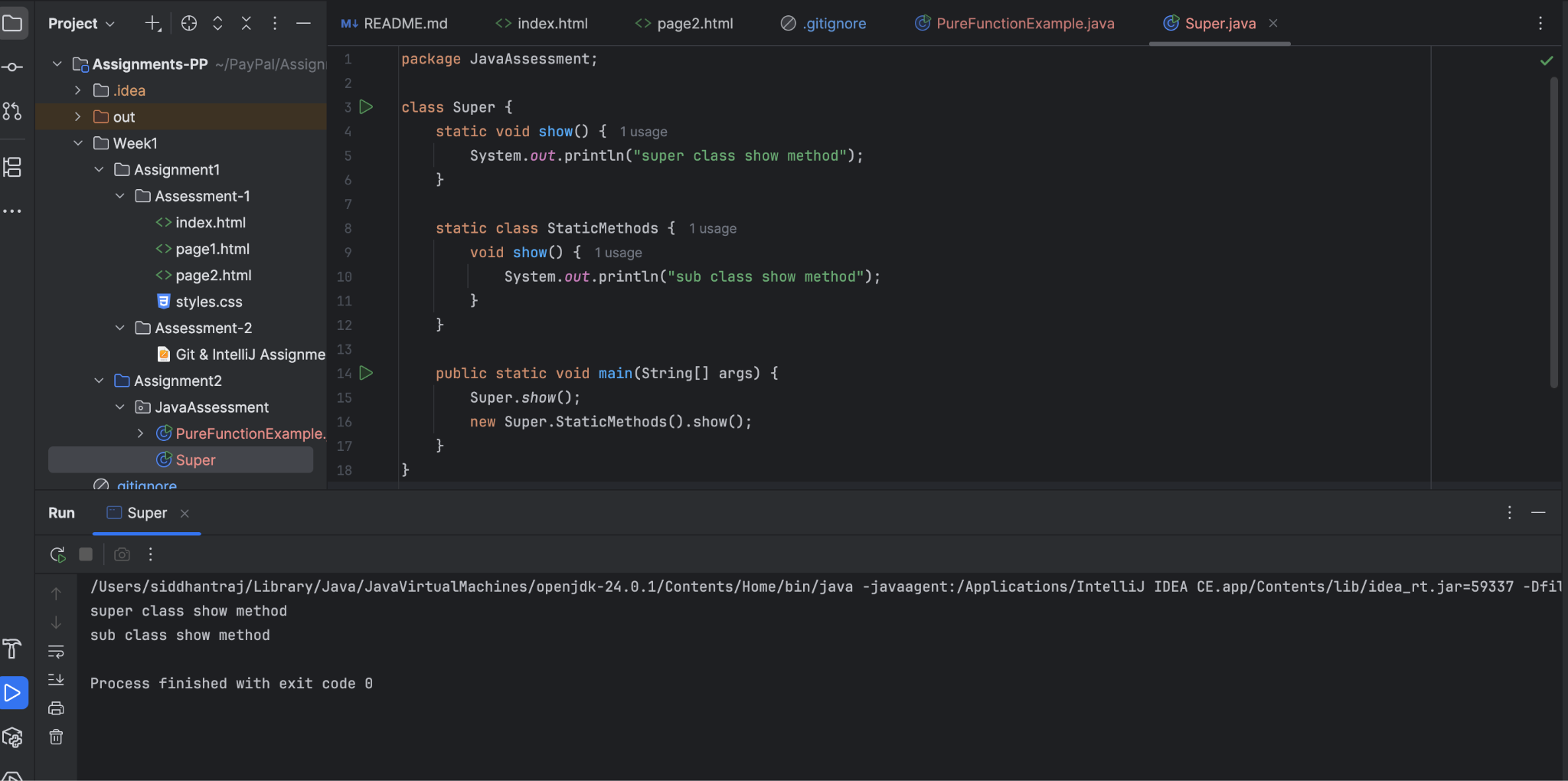
new Super.StaticMethods().show();

}

}

**Ans :-** super class show method

sub class show method



3)

What will be the output for the following code?

class Super

{

int num=20;

public void display()

{

System.out.println("super class method");

}

}

public class ThisUse extends Super

{

int num;

public ThisUse(int num)

{

this.num=num;

}

public void display()

{

System.out.println("display method");

}

public void Show()

{

this.display();

display();

System.out.println(this.num);

System.out.println(num);

}

public static void main(String[]args)

{

ThisUse o=new ThisUse(10);

o.show();

}

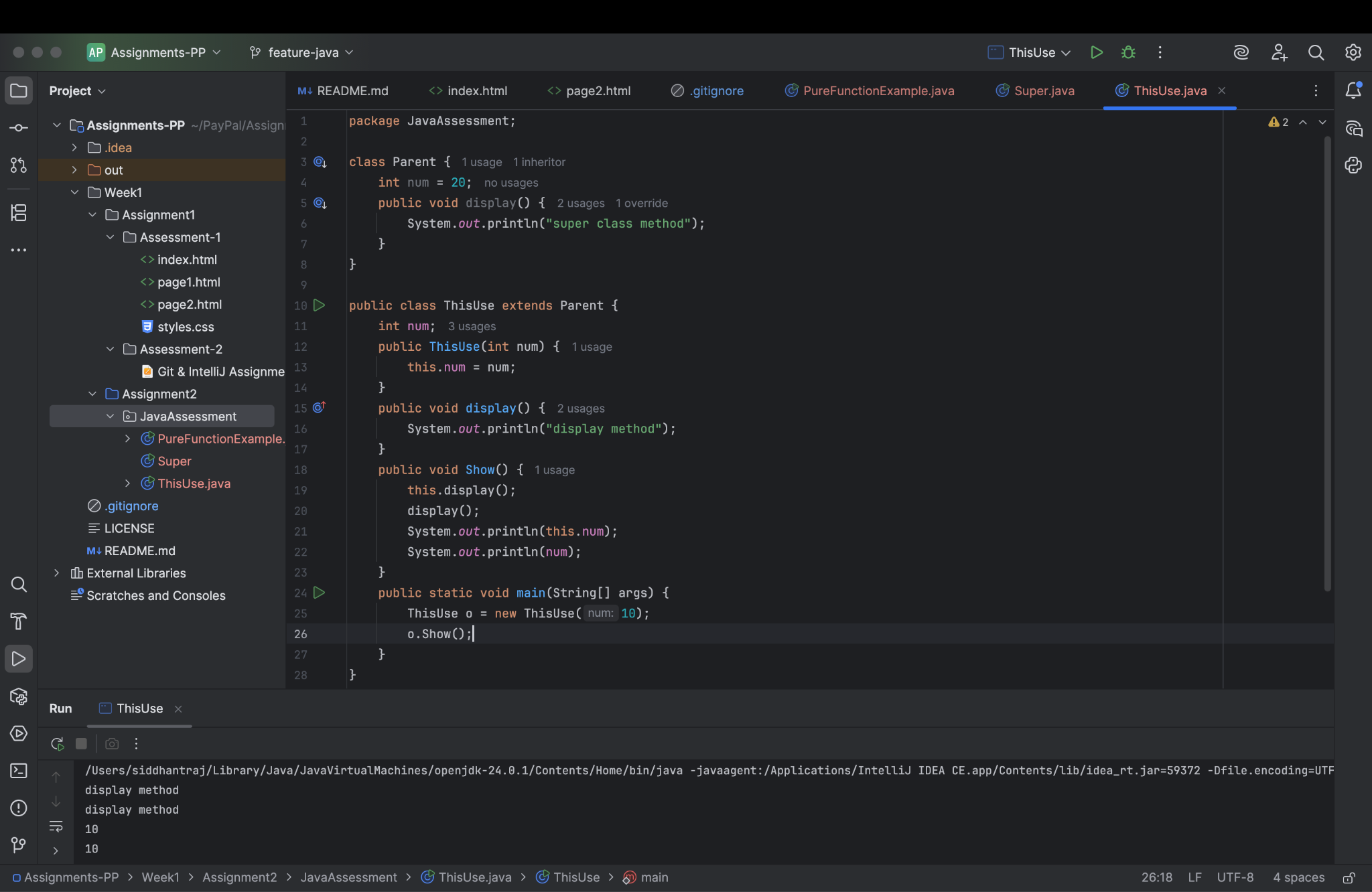
}

**Ans :-** display method

display method

10

10



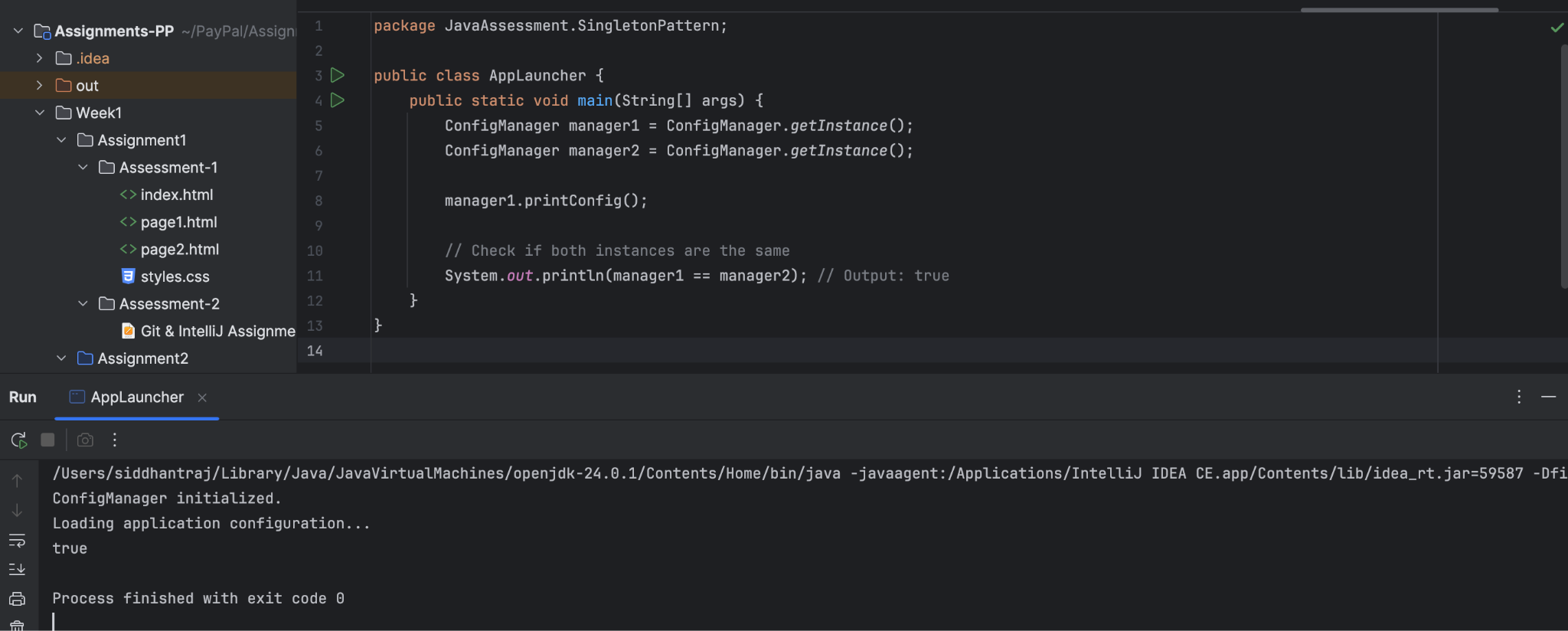
4) What is the singleton design pattern? Explain with a coding example.

**Ans :-** The Singleton Pattern ensures that a class has only one instance and provides a global point of access to it. It’s commonly used in scenarios like logging, configuration management, or database connections, where having multiple instances can cause problems.

Key Characteristics:

* Only one instance of the class exists.
* Provides a static method to get the instance.
* Constructor is private to prevent external instantiation.

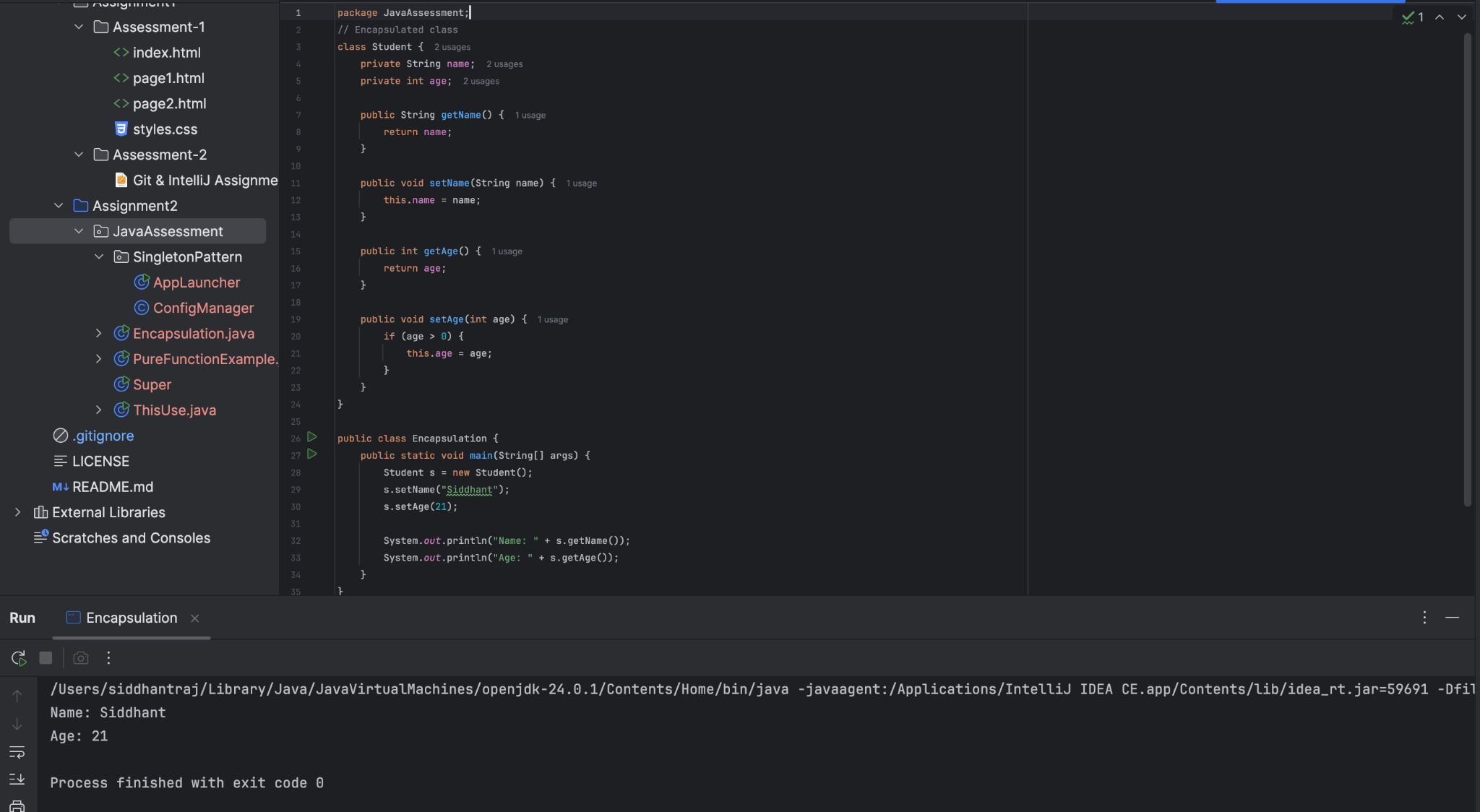




5) How do we make sure a class is encapsulated? Explain with a coding example.

**Ans :-** Encapsulation means wrapping data (variables) and methods (functions) into a single unit (class) and restricting direct access to some of the object's components. To ensure a class is encapsulated :-

* Make all data members private.
* Provide public getter and setter methods to access and update the values.
* Hide internal implementation details from other classes.



6)Perform CRUD operation using ArrayList collection in an EmployeeCRUD class for the below Employee

class Employee{

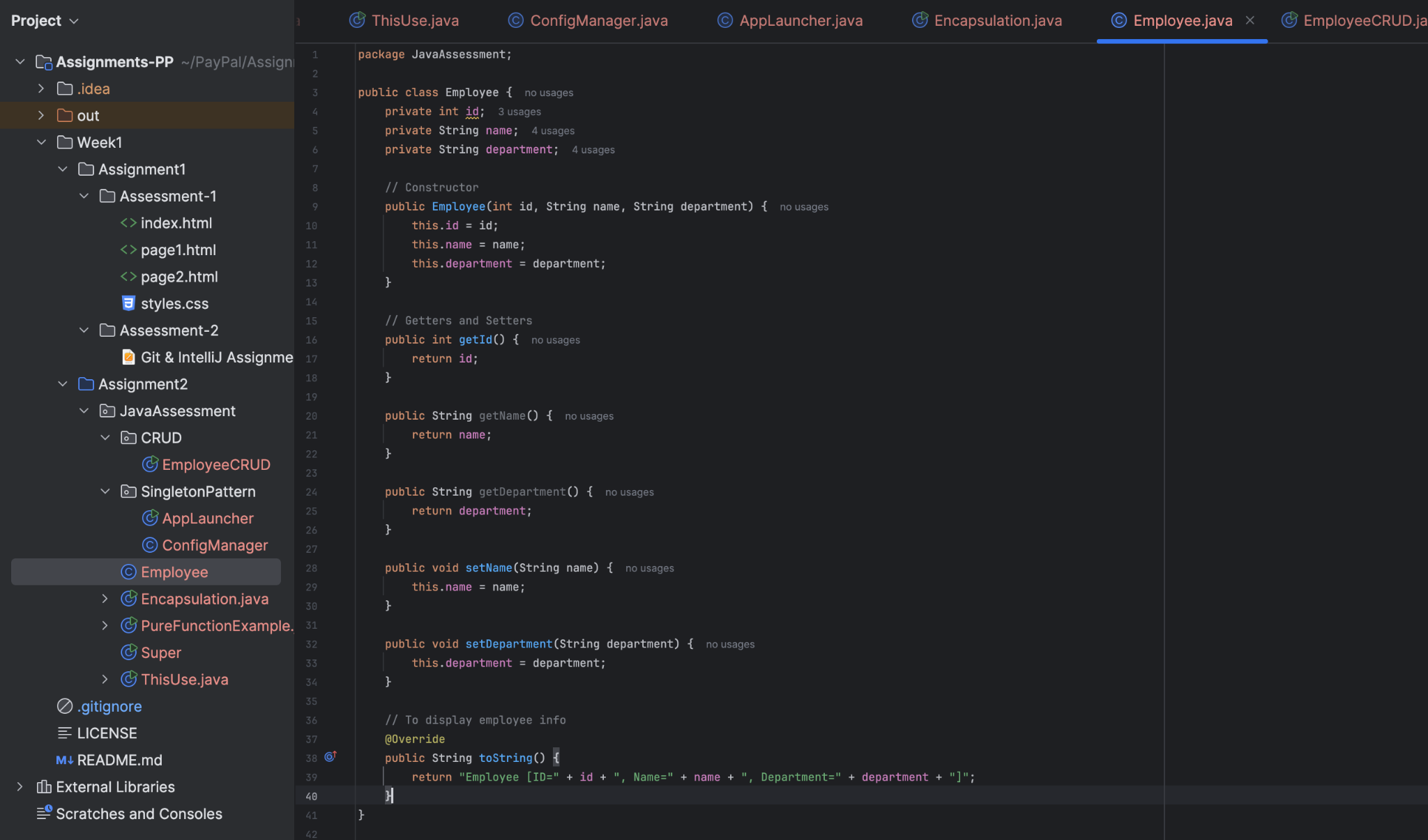
private int id;

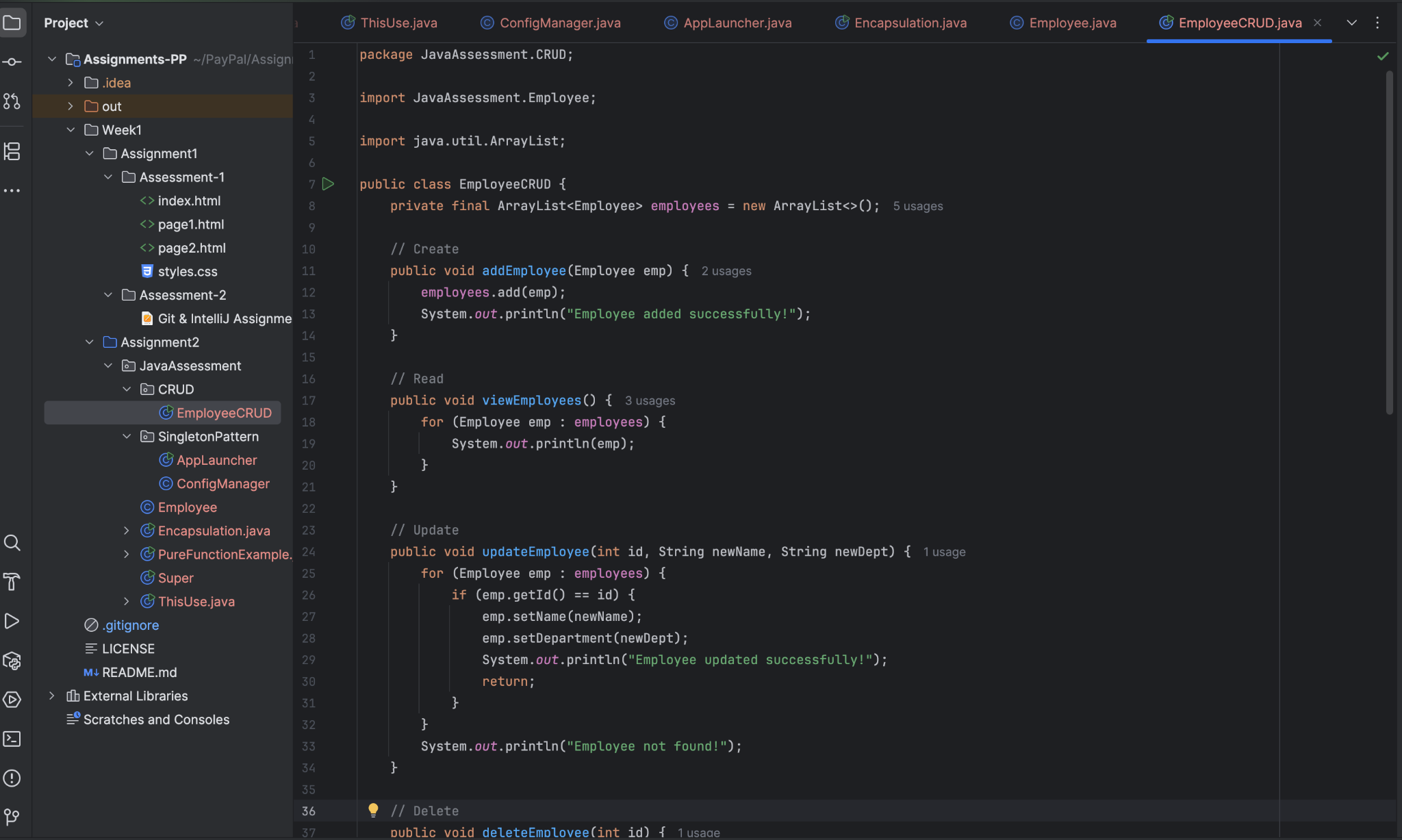
private String name;

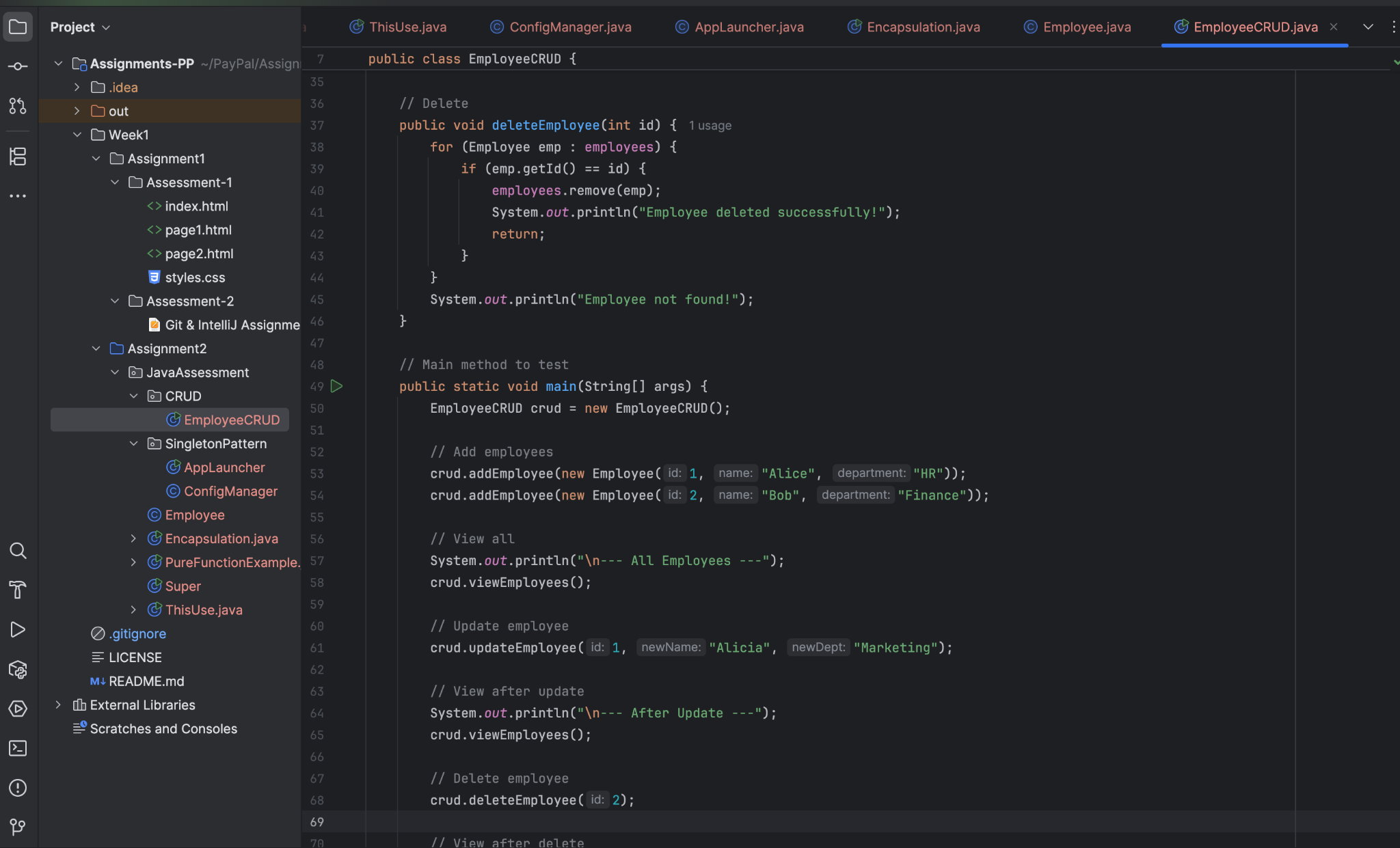
private String department;

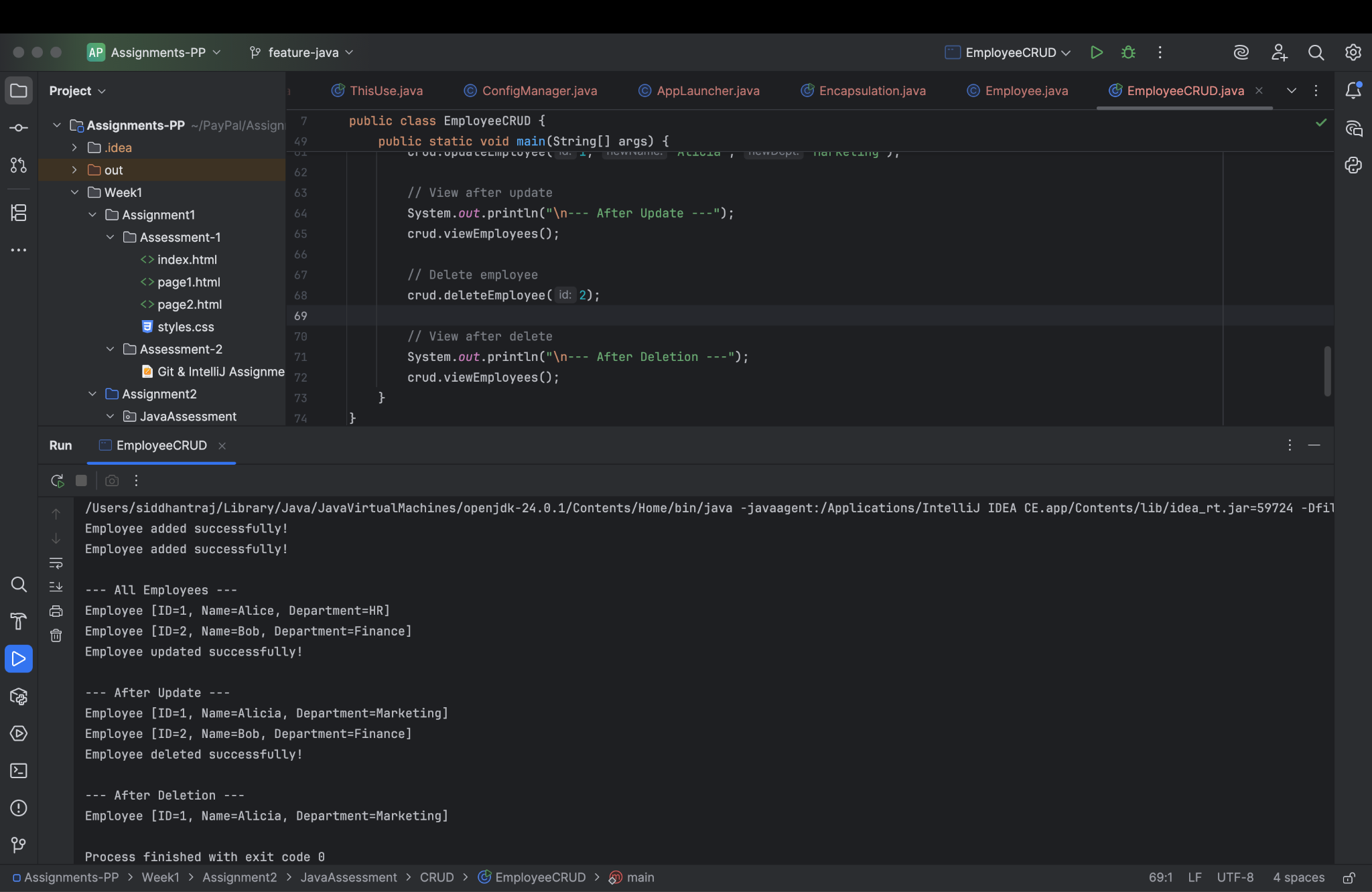
}

**Ans :-**









7) Perform CRUD operation using JDBC in an EmployeeJDBC class for the below Employee

class Employee{

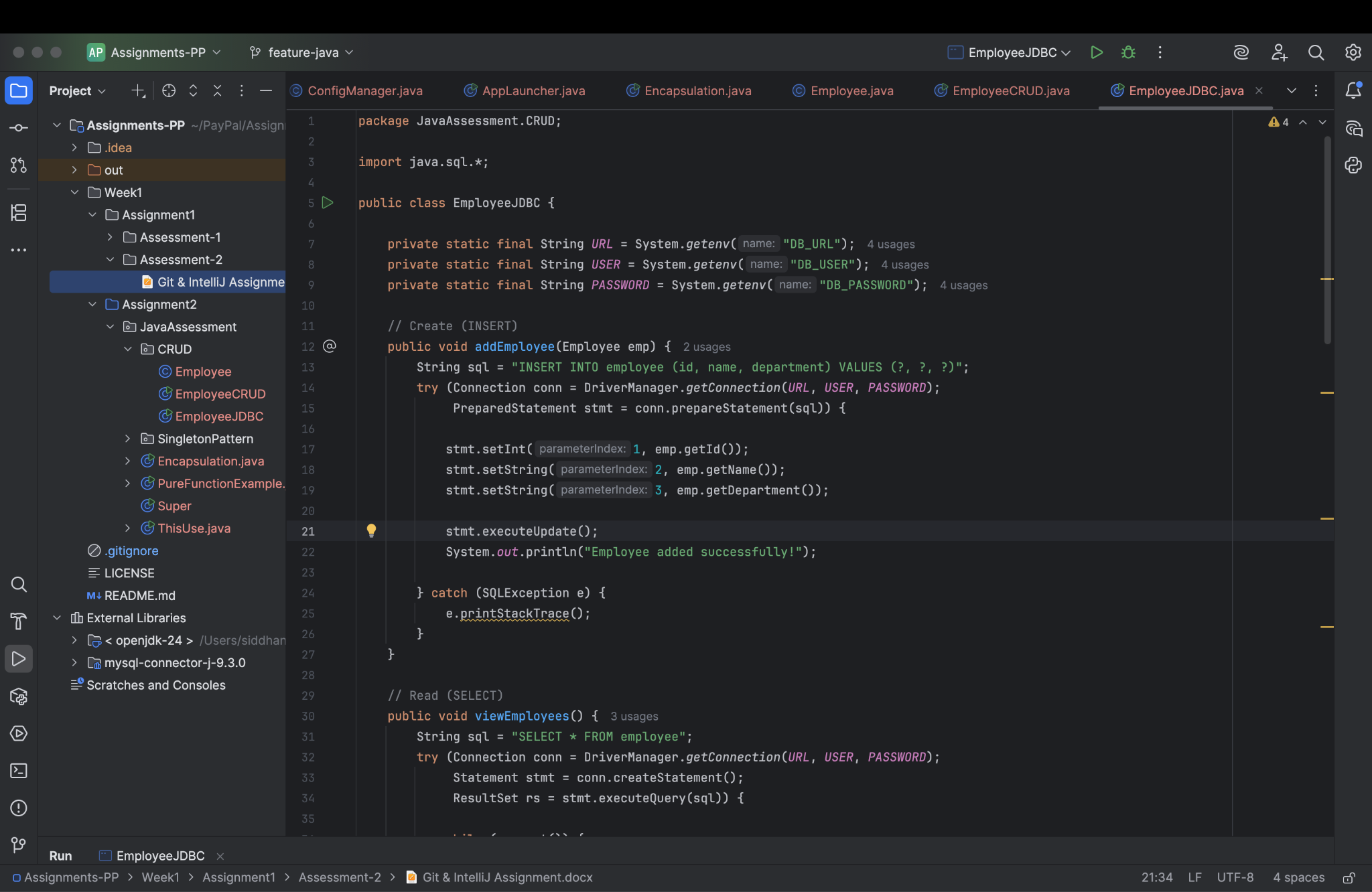
private int id;

private String name;

private String department;

}

**Ans :-**

****

