**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 :-)**

GITHUB LINK: <https://github.com/AnuraagGanji/rg-assignments/tree/feature-java/Week2>



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.

**Explanation:** The method calculateTax() is not a pure function as its return value is dependent on the value of the rate variable which may change depending on the state of the program. One way to make it pure is to take the rate also as one of the parameters of the method, like calculateTax(double amount, double rate).

2)

What will be the output for the 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)

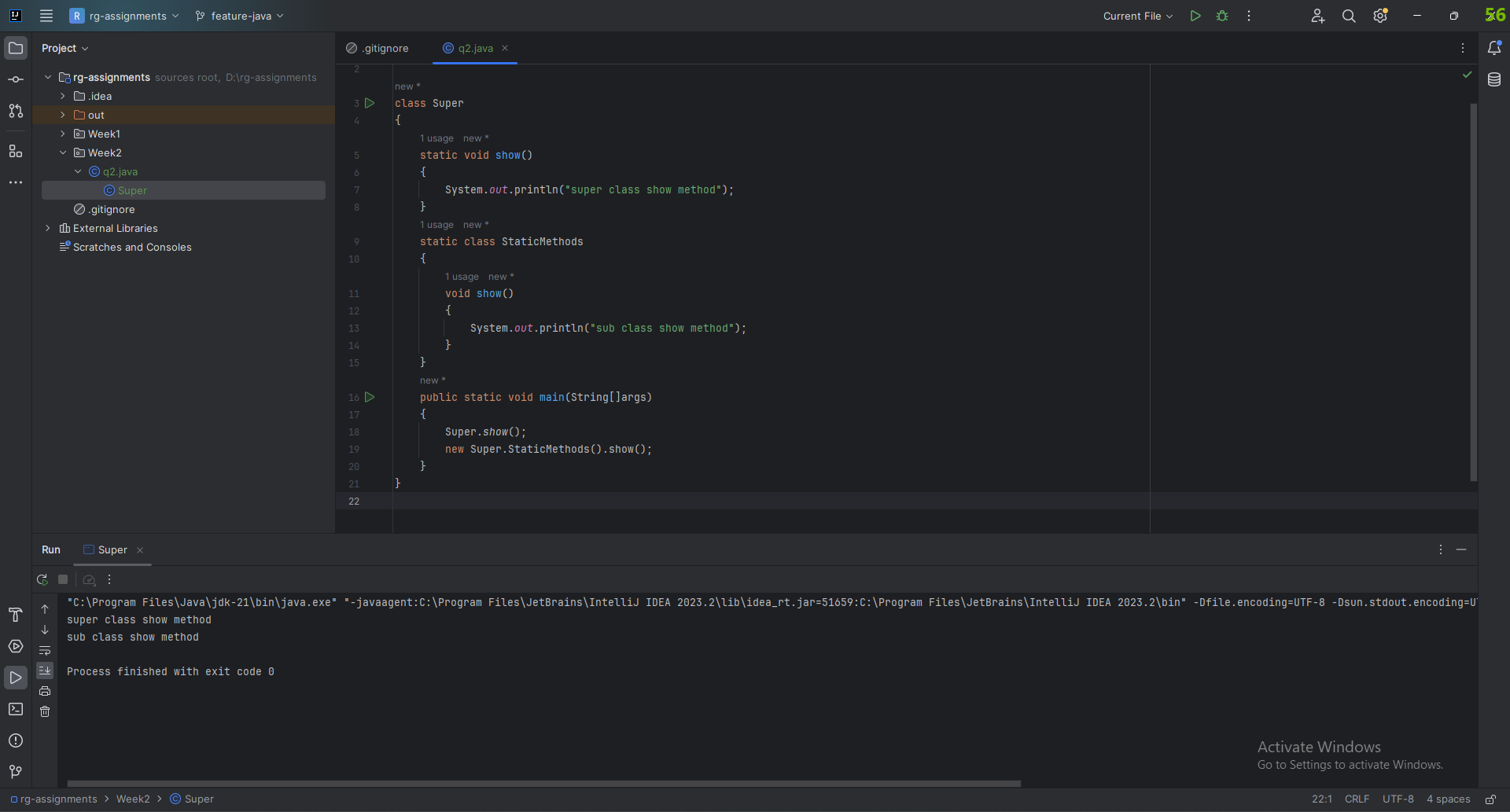
{

Super.show();

new Super.StaticMethods().show();

}

}



**Explanation:** In Super.show(), we are calling the static method of the Super class and in Line 2, we are instantiating an object of the nested class StaticMethods and we are calling it’s instance method show().

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)

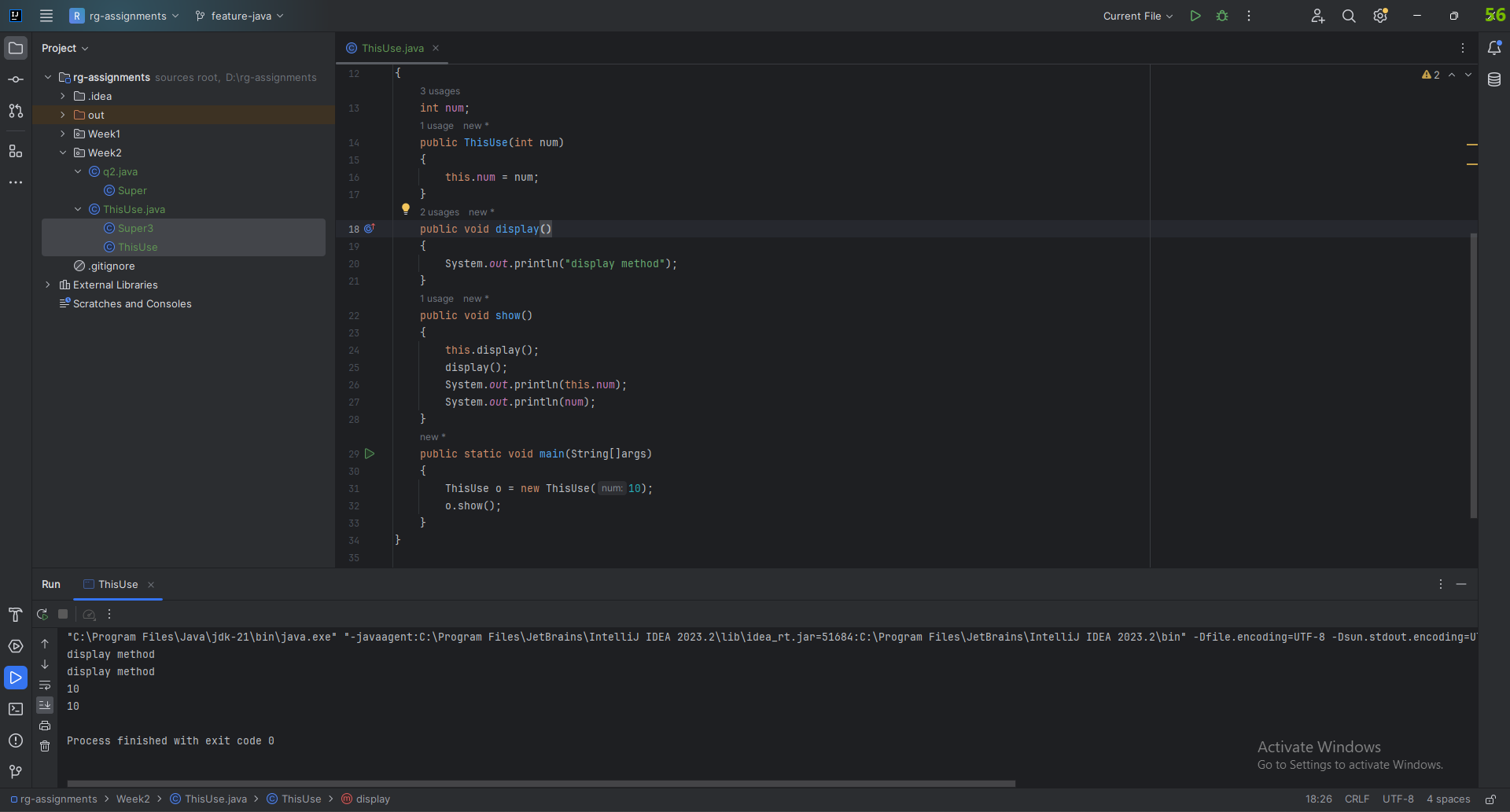
{

ThisUse o=new ThisUse(10);

o.show();

}

}

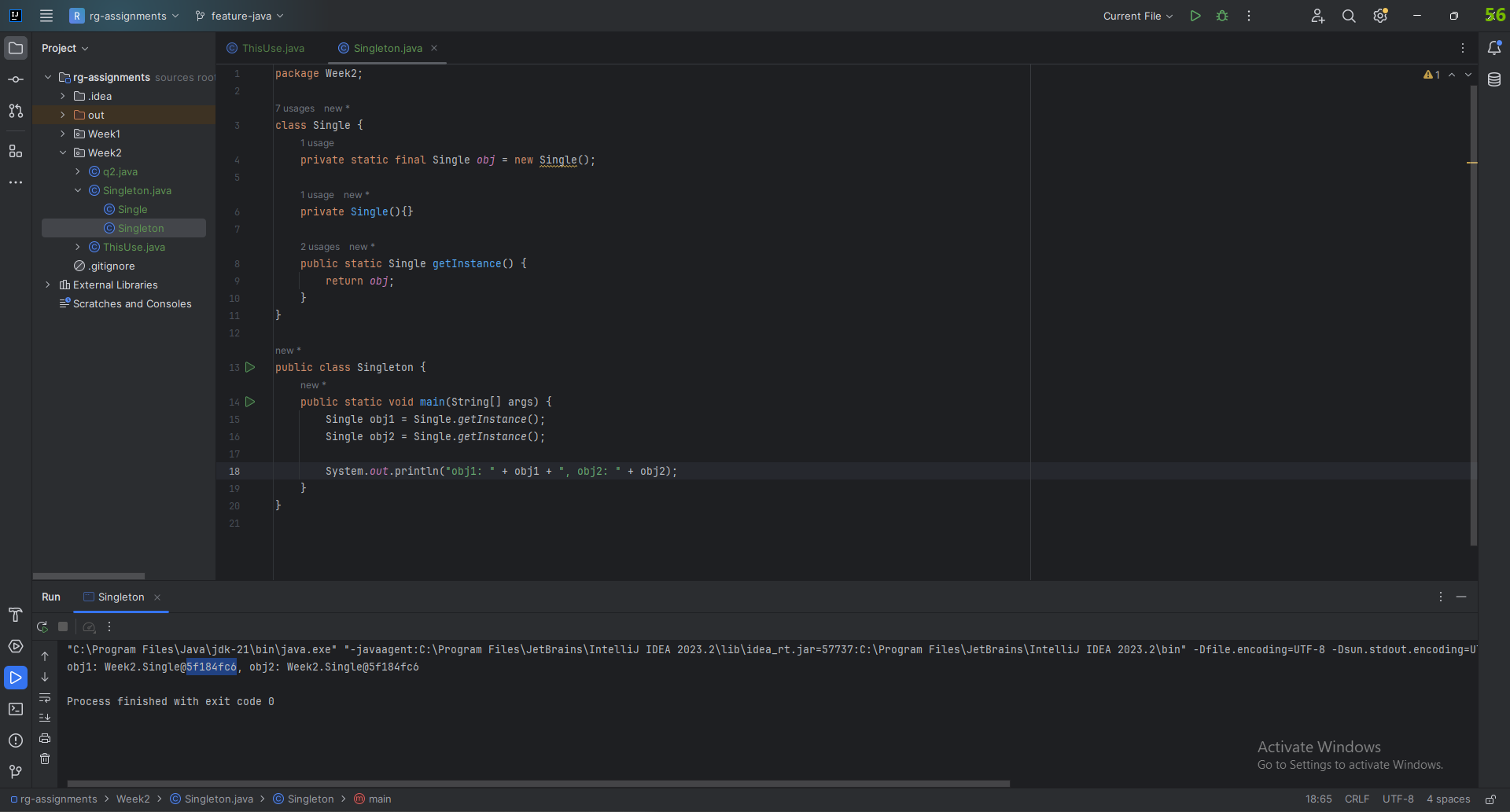


**Explanation:** ThisUse is a subclass of Super. The display() method in ThisUse is overriding the display() method of Super. In the show() method, calling this.display() and display() is equivalent. It wouldn’t be if there was any other method called display() in the same scope, which is not the case here. Similar logic applies for the num variable. Hence, the output is as shown. It would’ve been different if we were calling either Super.display() or printing Super.num.

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

The idea of singleton design pattern is to have only one instance of a class at any time. Only one object is created and it is used for the life of the program.

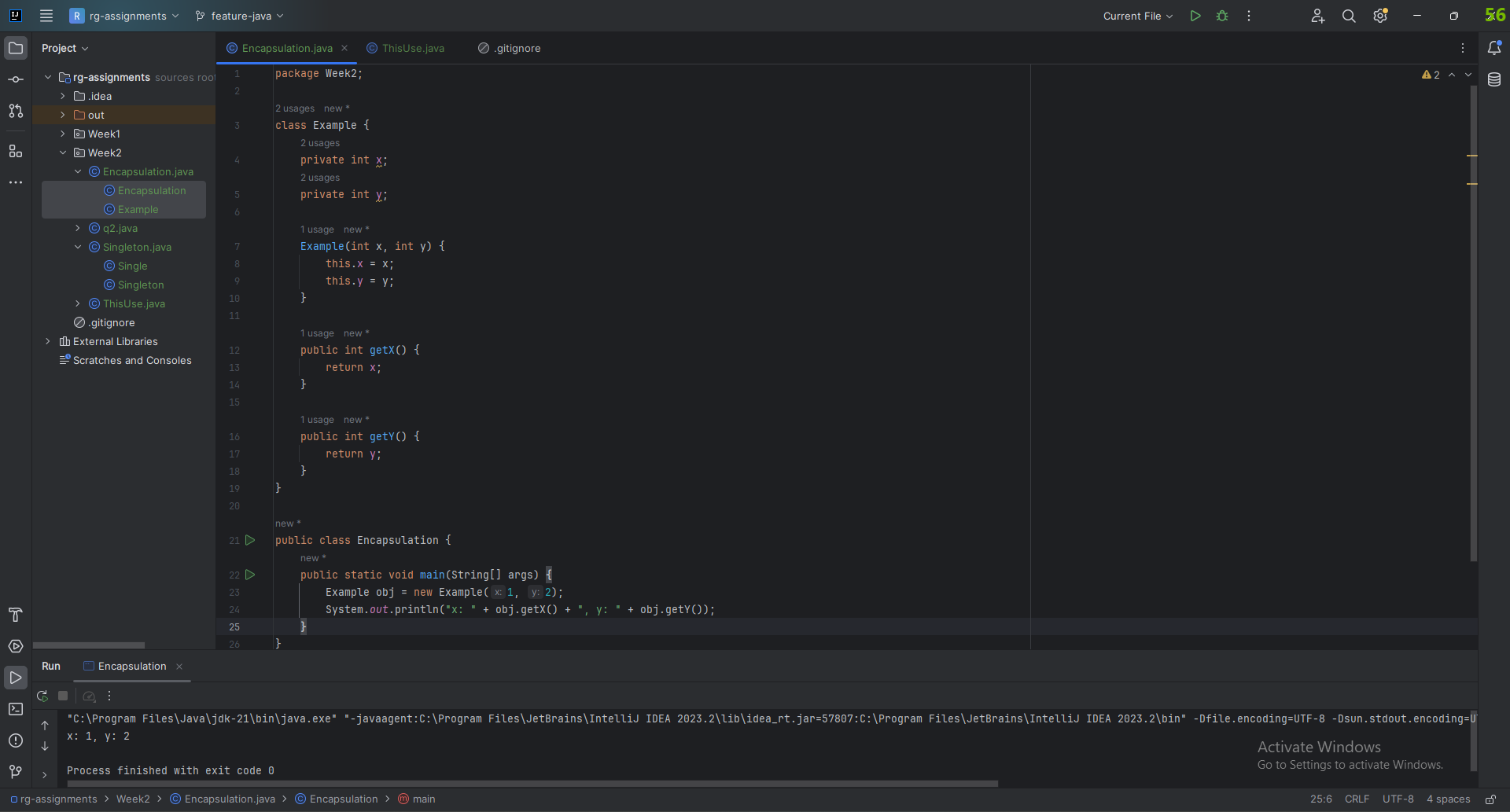
Making a singleton class:



As one can see, both obj1 and obj2 have the same hashcode.

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

We use access modifiers (private, protected, public, default) to implement encapsulation in Java.



Here we can’t directly access x and y from outside the class using obj.x and obj.y. We have to use the getters in order to do that since x and y are private. This example shows encapsulation.

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

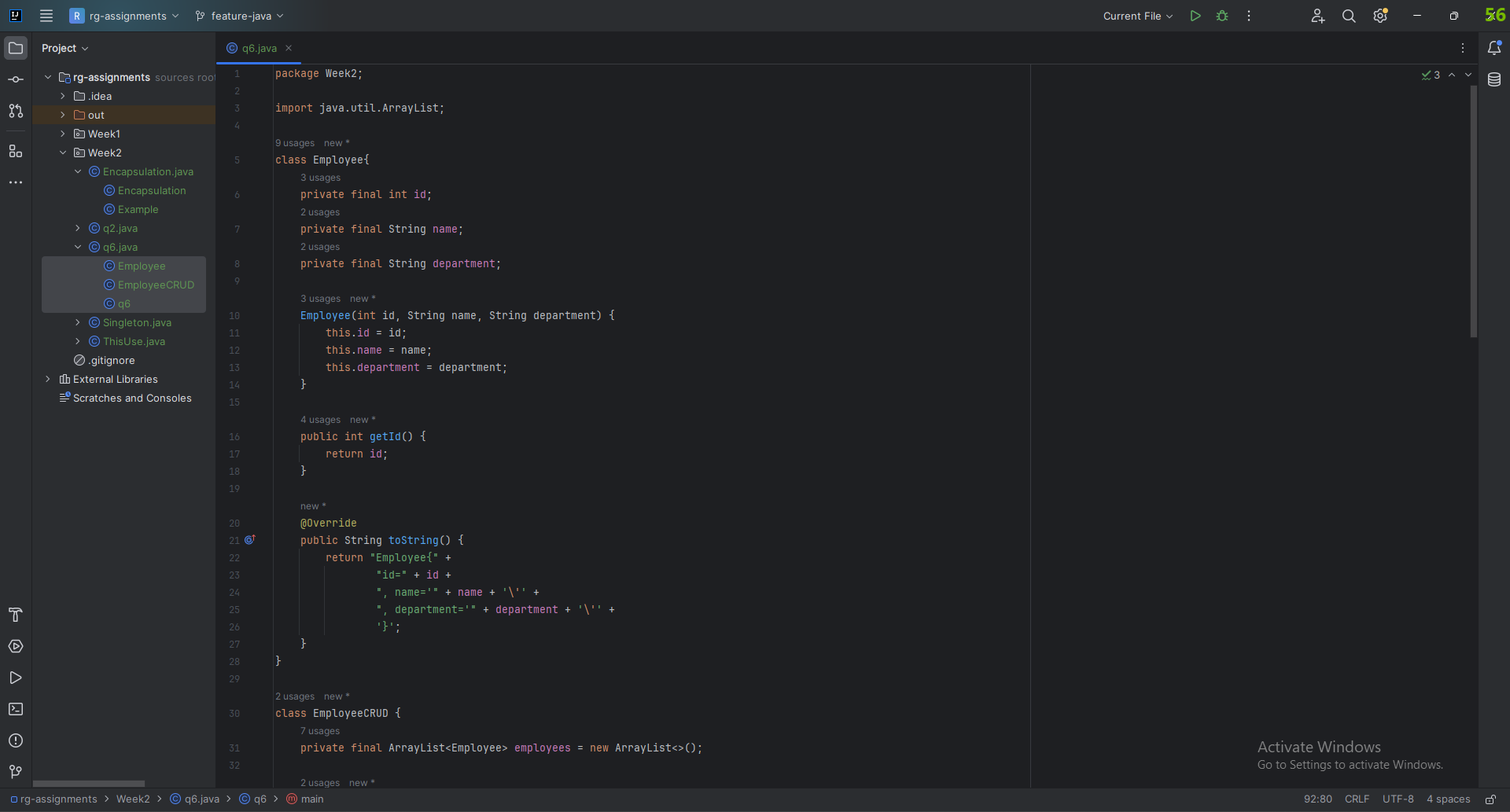
class Employee{

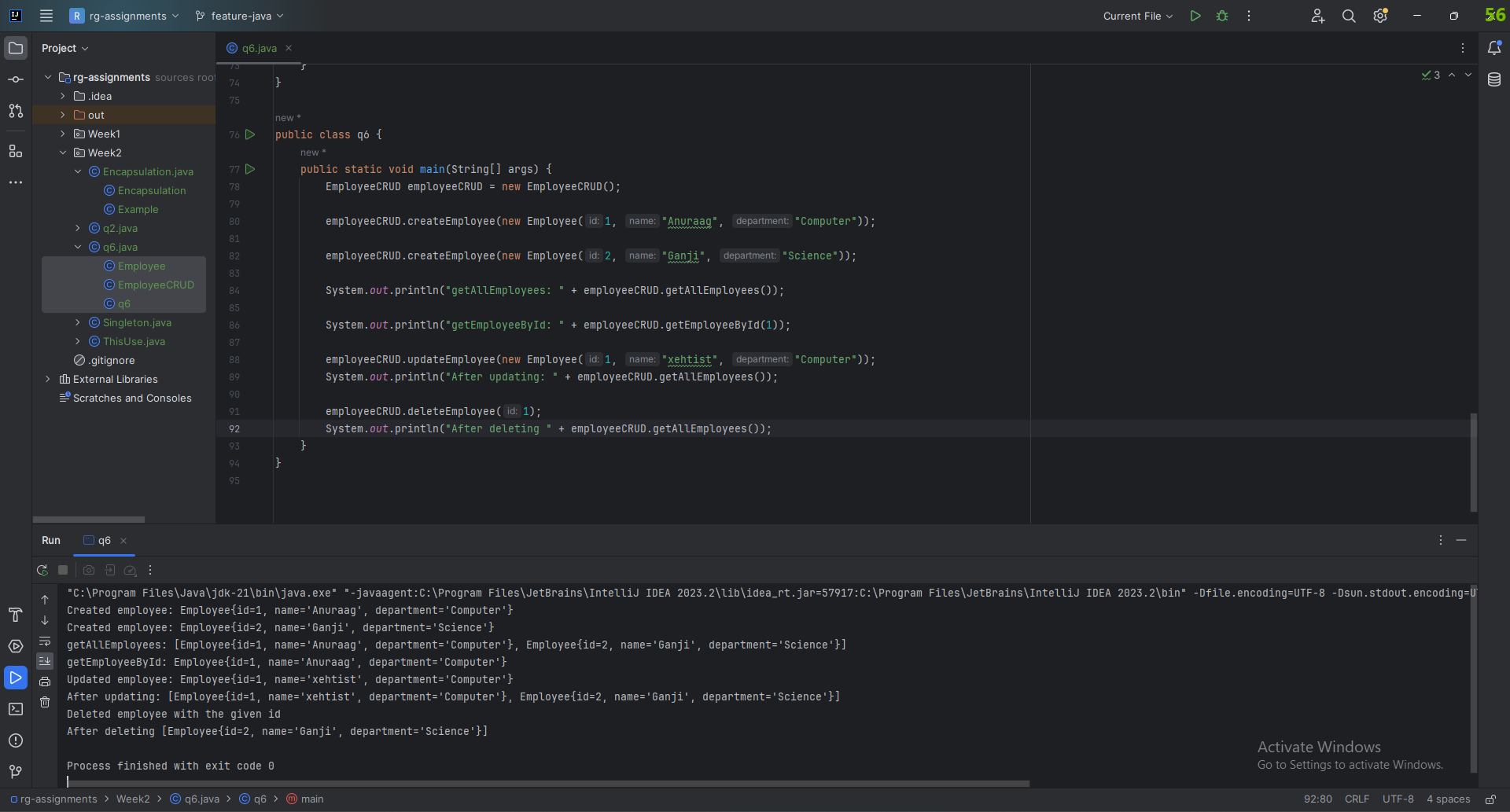
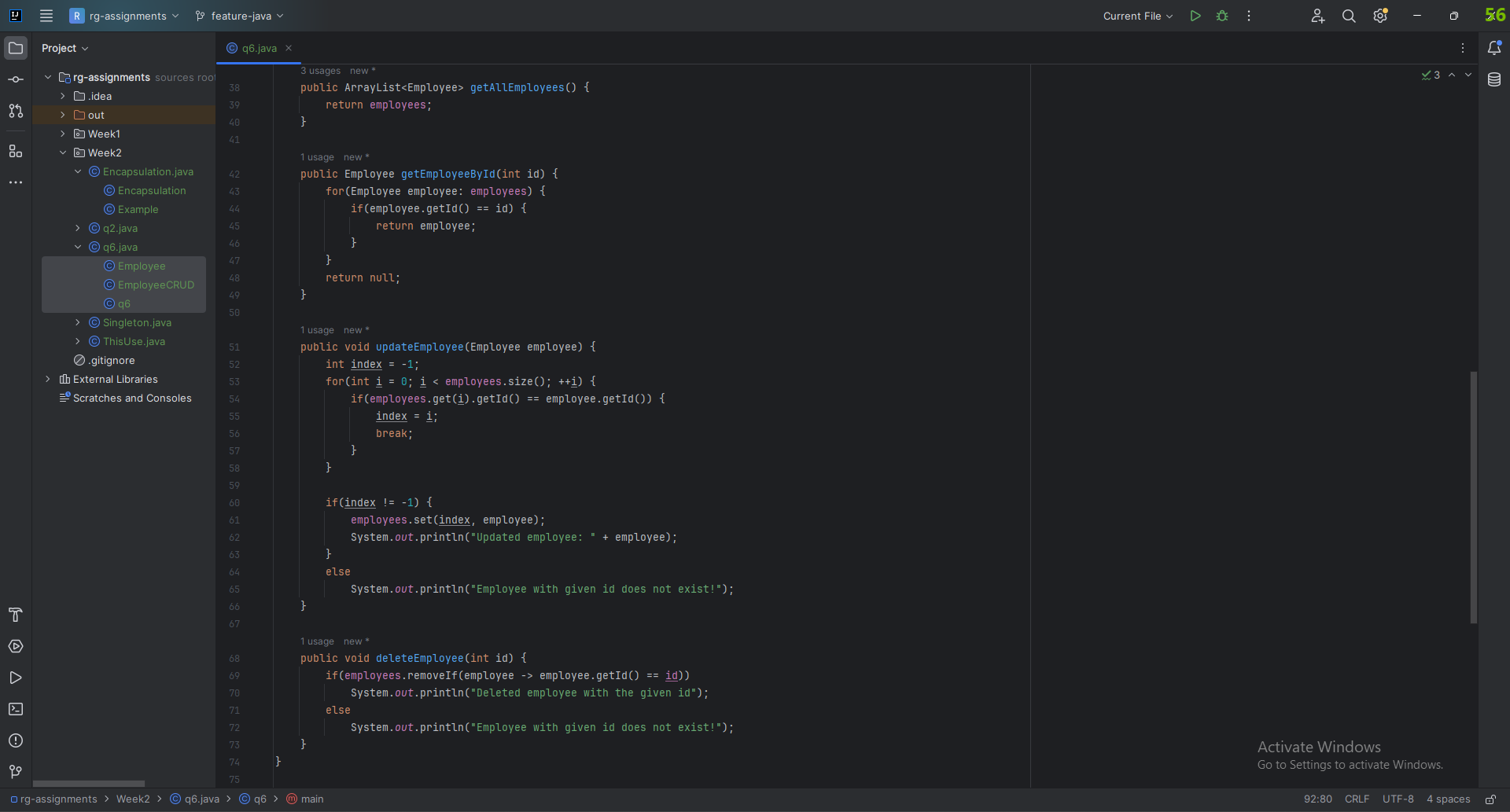
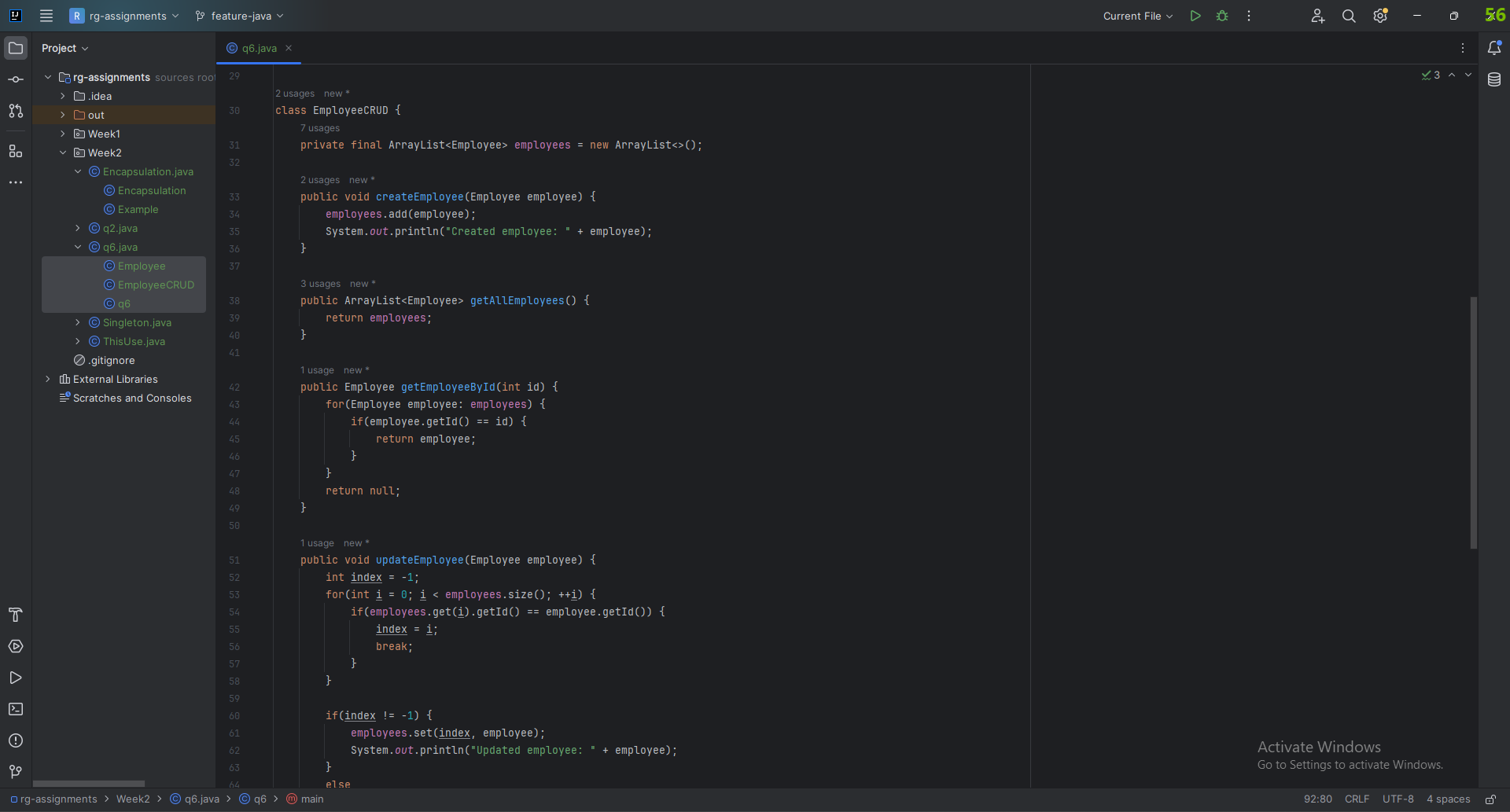
private int id;

private String name;

private String department;

}





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;

}

Looked about it here: <https://dotnettutorials.net/lesson/crud-operations-in-java-using-jdbc/>