#### 1.

#### It is not a pure function. A pure function must return the same output for the same input and have no side effects.

#### public class TaxUtil {

#### double rate = 0.15;

#### public double calculateTax(double amount) {

#### return amount \* rate;

#### }

#### }

calculateTax() depends on instance variable rate, so if rate is changed, the same amount would give a different result… Therefore not pure.

To make it pure, pass the rate as function parameter - so that it relies only on input parameters

public class TaxUtil {

public double calculateTax(double amount, double rate) {

return amount \* rate;

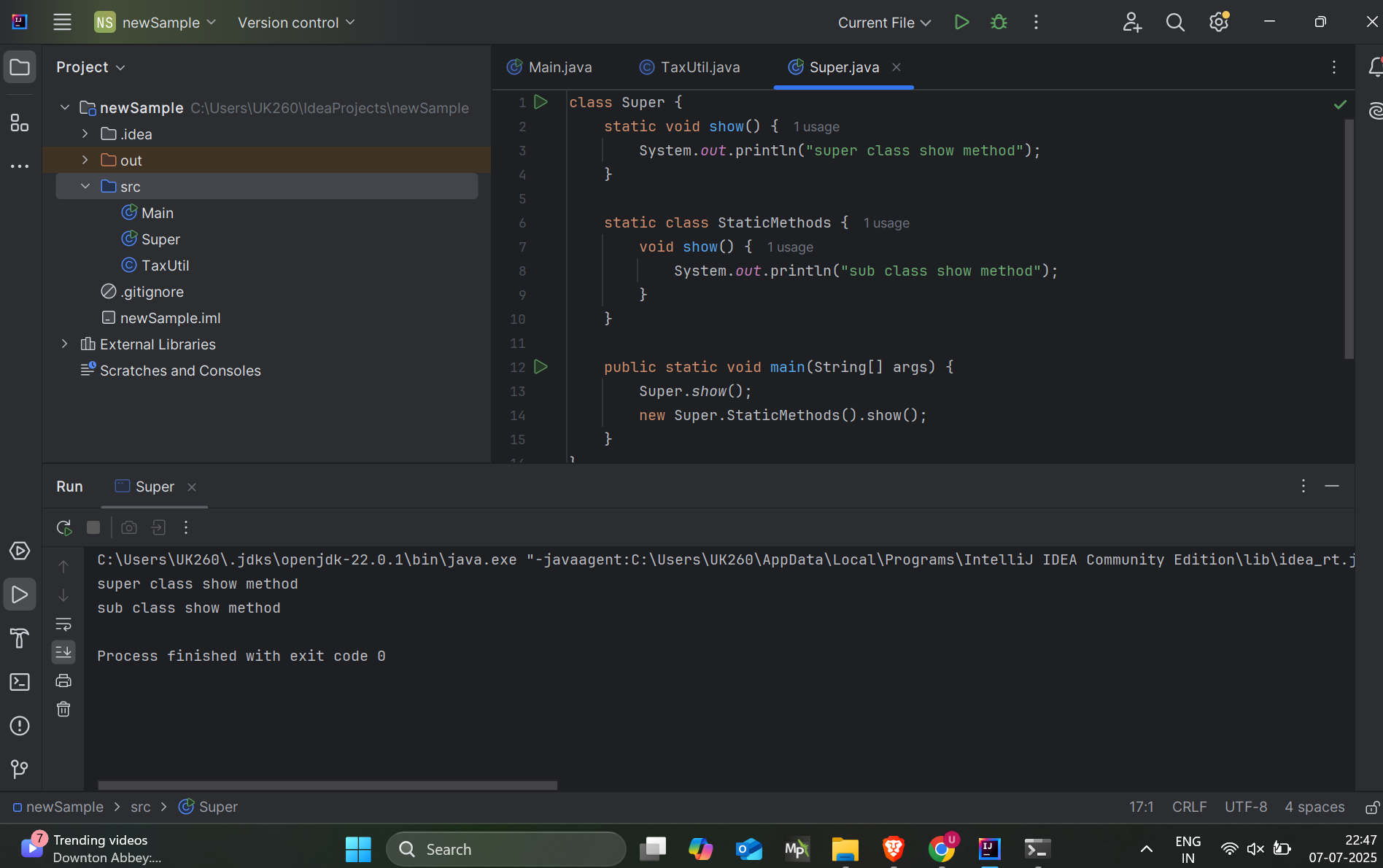
}

}

2.

Super.show() calls the static method.

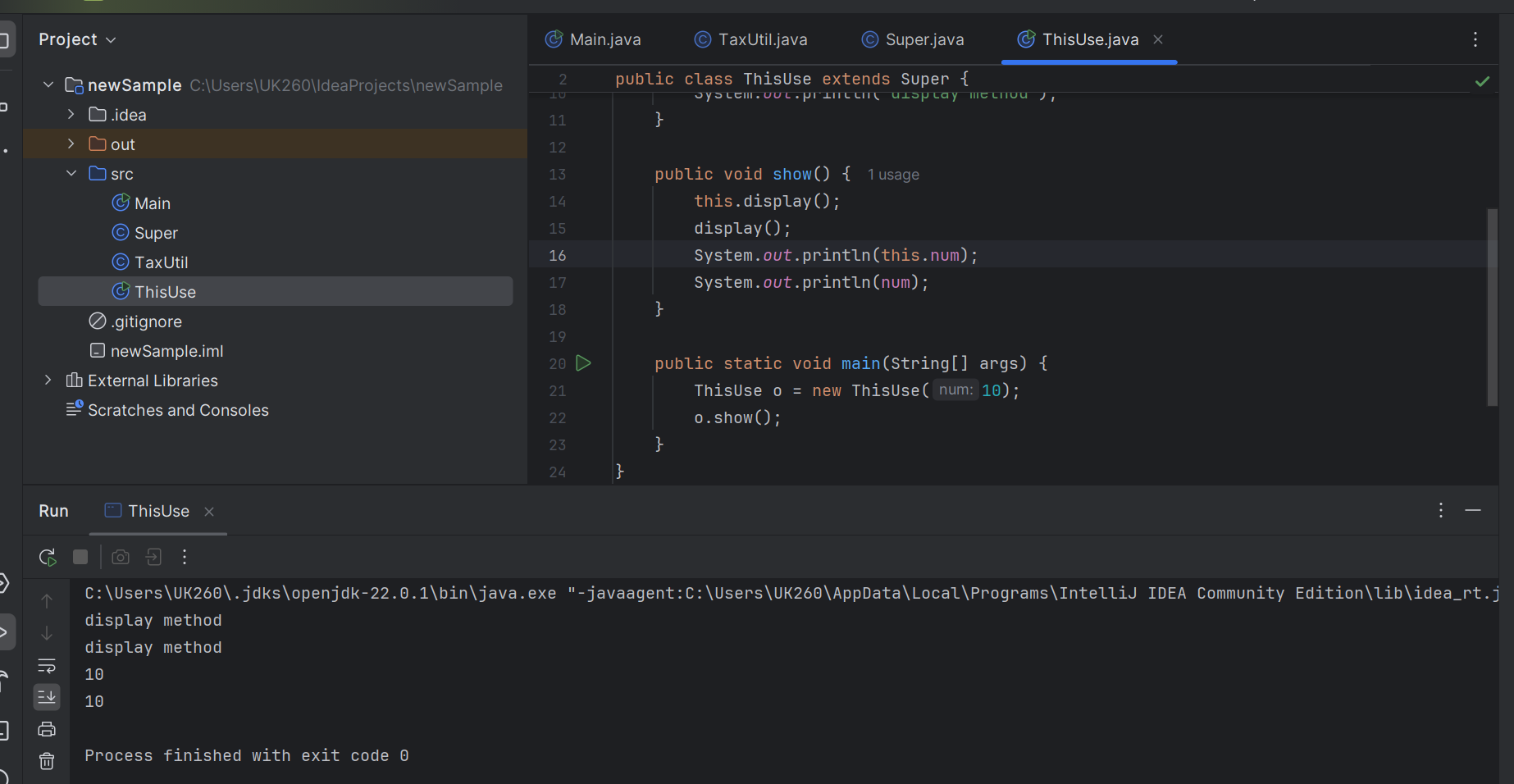
StaticMethods.show() is an instance method, so we create an object and call it.



3.

this.display() and display() both call the overridden method in ThisUse.

this.num and num refer to the num in ThisUse, not Super.



4.

Singleton ensures only one instance of a class exists in the application.

public class Singleton {

private static Singleton instance;

private Singleton() {

// private constructor

}

public static Singleton getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

}

5.

Encapsulation = wrapping fields with private access and exposing them via public getters/setters.

Here, name and age are encapsulated (hidden).

You control access via public methods.

public class Student {

private String name;

private int age;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

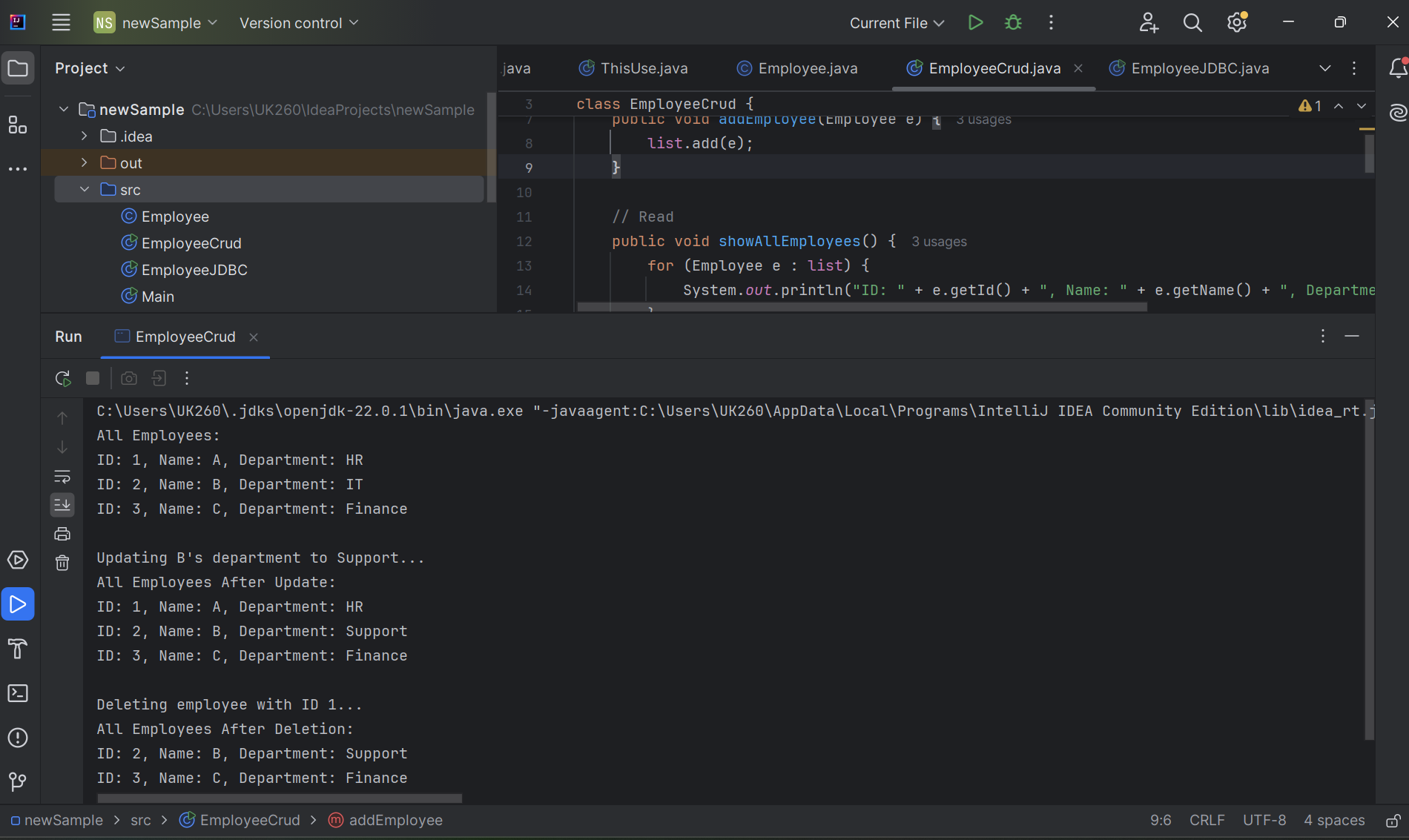
public void setAge(int age) {

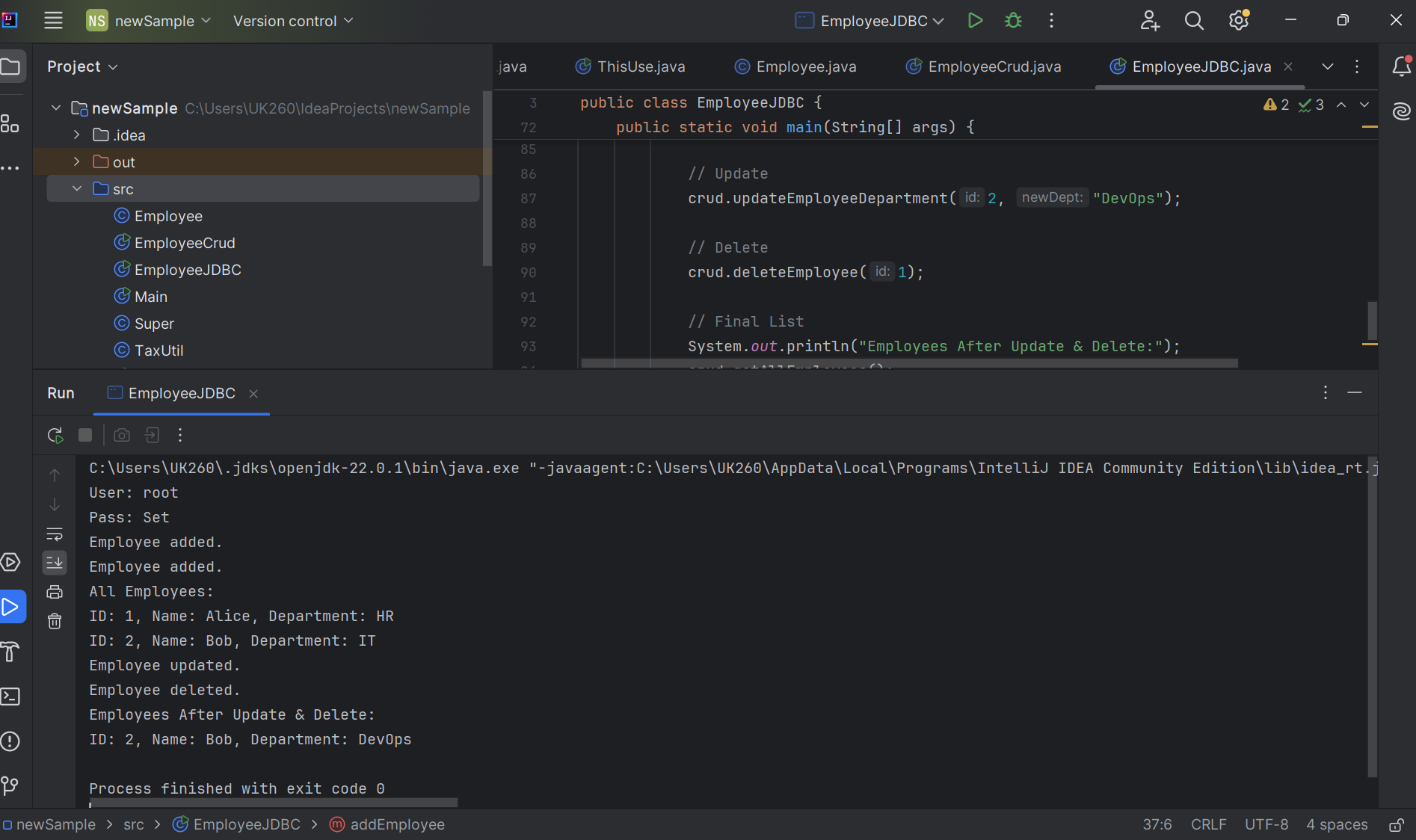
this.age = age;

}

}

6. , 7.





JavaScript

