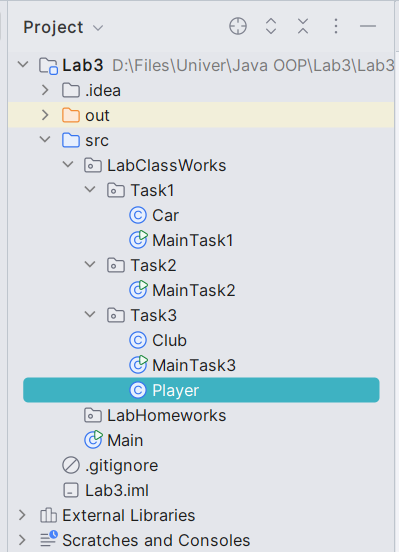
Ospan Ramazan it2-2310, 38045

*Class Work:*



Work structure

Task1.

Create a class called **Car**, which has parameters and methods:

String name; String model; int maxSpeed;

double engineVolume;

Car() //default constructor

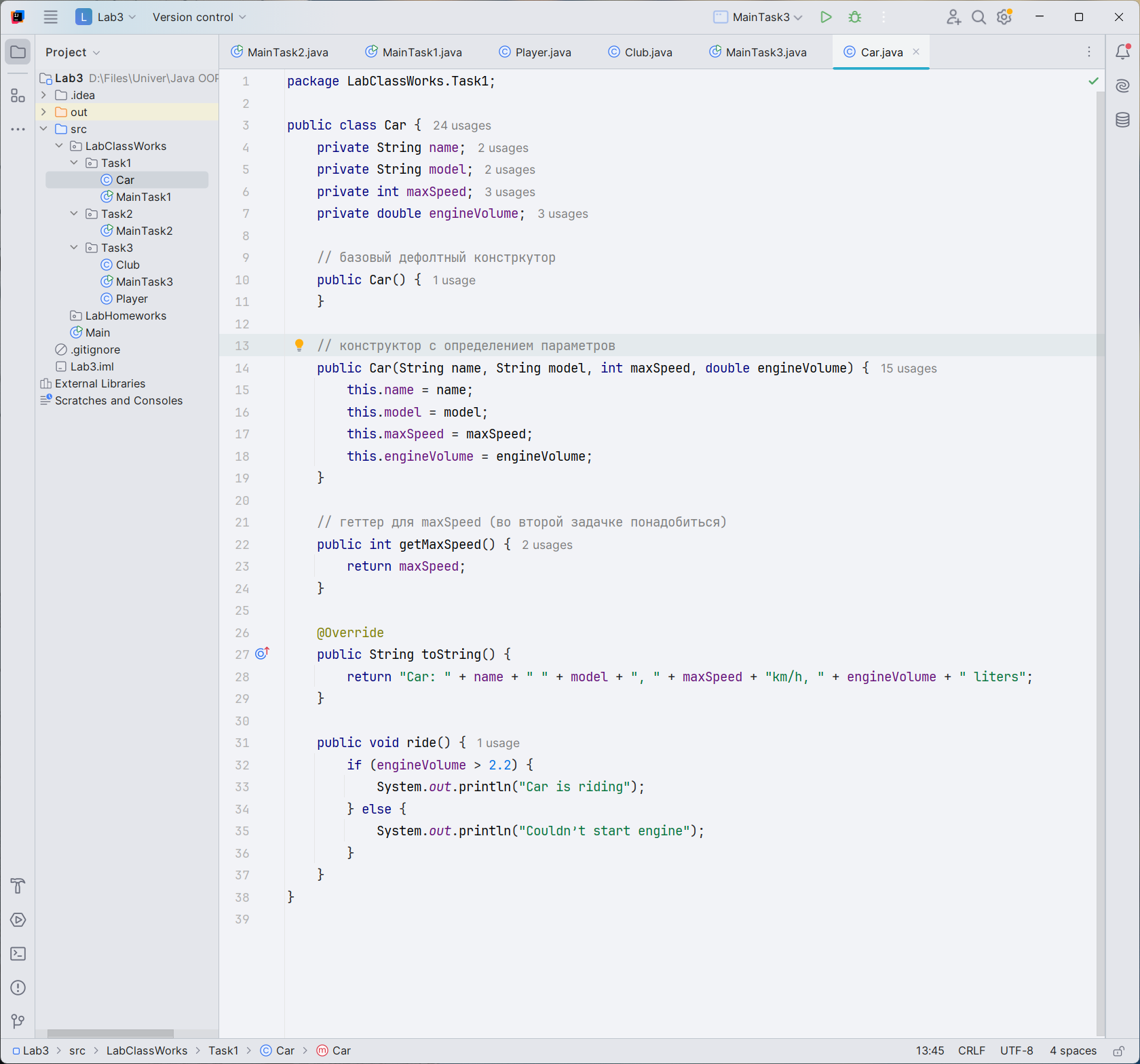
Car(String name, String model, int maxSpeed,double engineVolume)

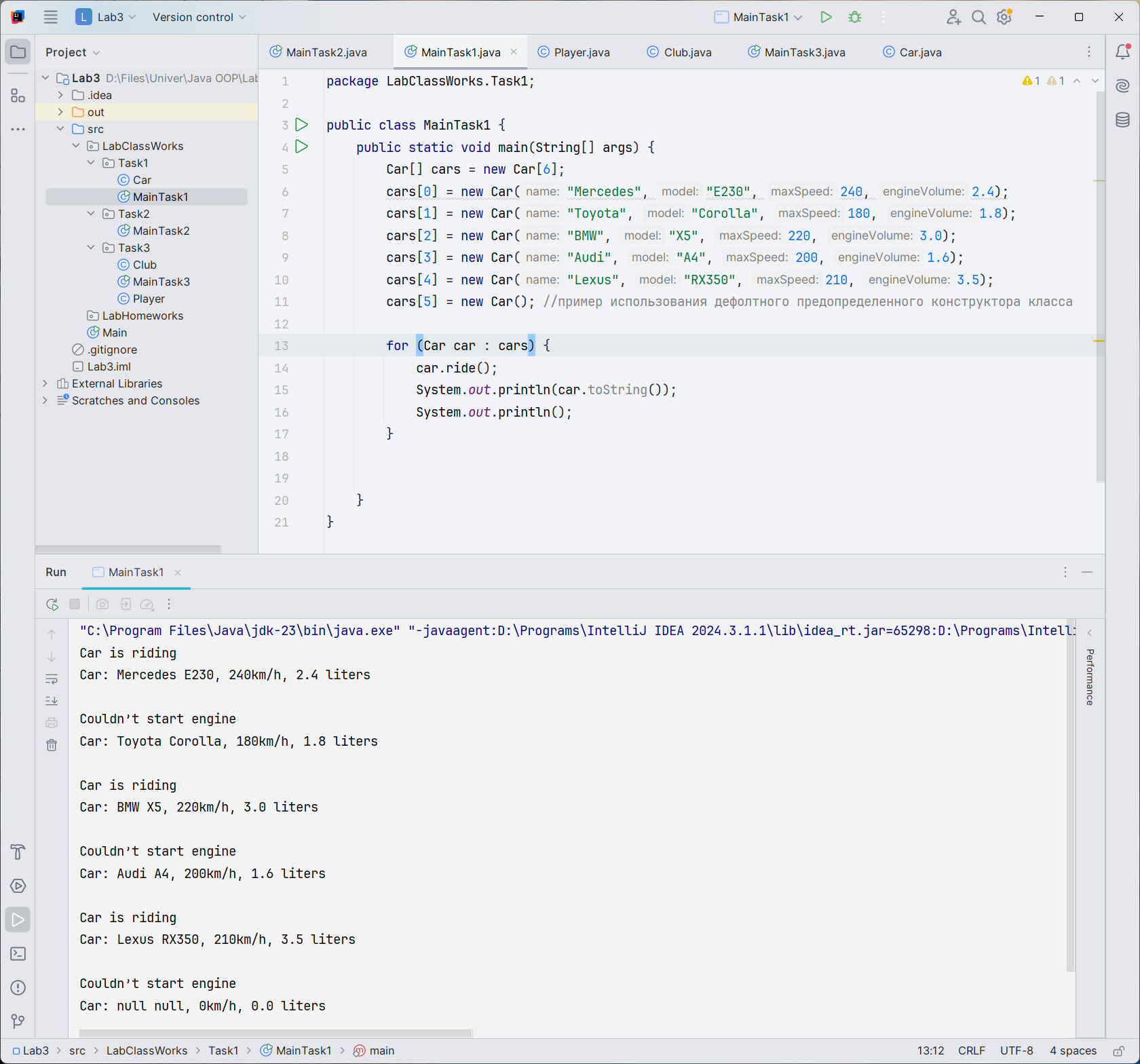
**String toString()** // This method returns all data of car in one row in following format

**Example:** Car: Mercedes E230, 240km/h, 2.4 liters

**void ride()** // This method prints "Car is riding" if engineVolume of a car will be greater than 2.2 liters, otherwise print "Couldn’t start engine".

In a main class, you should create at least five objects of Car class and put them into array. After putting objects, call **ride()** method for all objects in array by using loop.

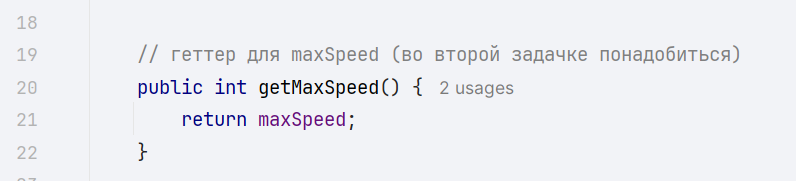




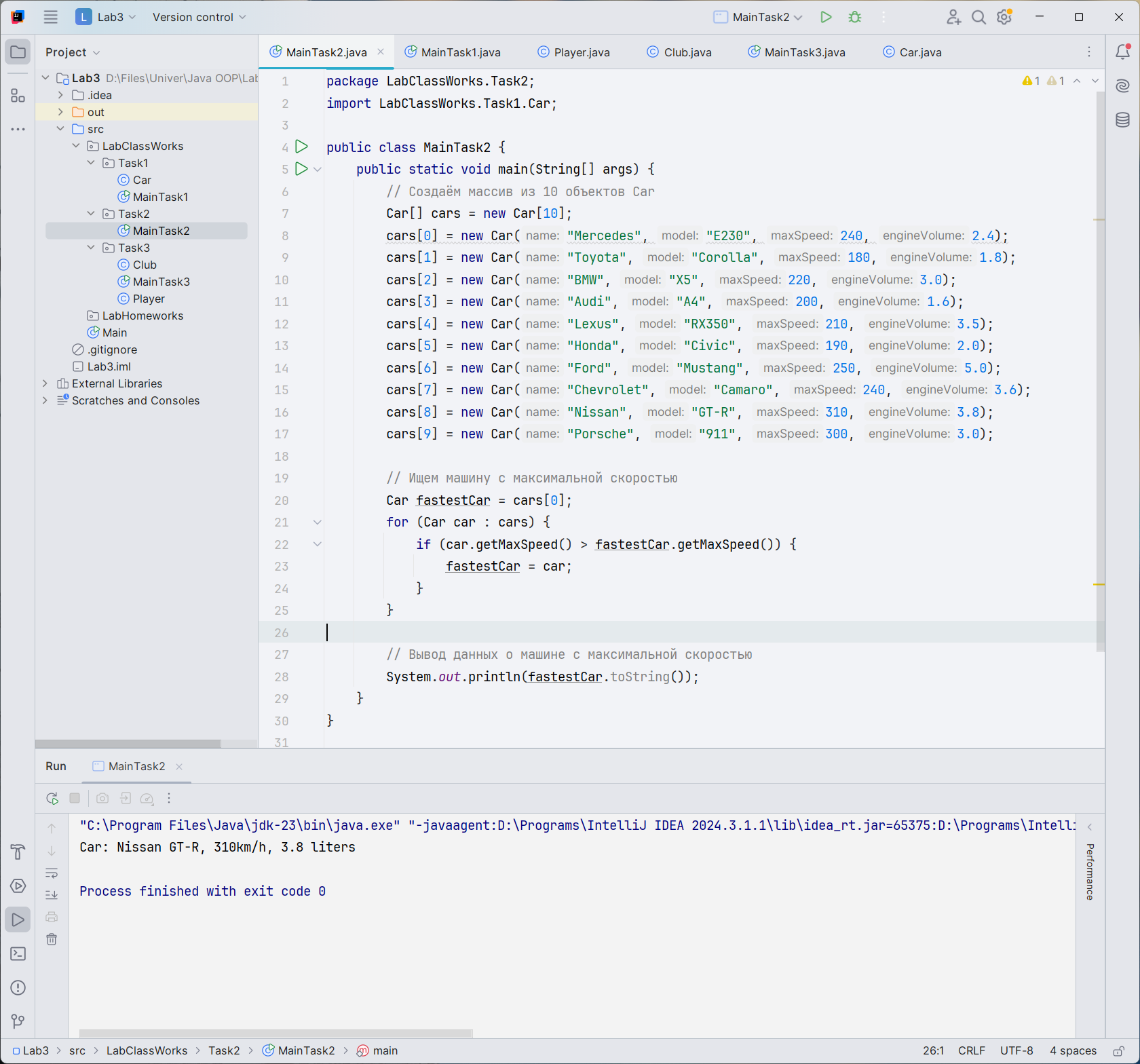
Task2.

Use previous Student class. Create at least 10 objects and put them into array.

You should sort array according to their gpa in descending order, and print their data. Use bubble sort.



*Из предыдущего класса, поскольку ответ на второе задание у меня находится в другом package, чтобы не нарушать принципы ООП проектирования, я решил добавить данный getter для вызова данных из другого класса. Это позволило сохранить инкапсуляцию, обеспечивая доступ к полю через метод. Я рассматривал возможность изменения уровня доступа атрибута на* ***public****, но впоследствии выяснилось, что это могло бы стать грубым нарушением принципов ООП, так как нарушается защита данных, инкапсуляция и возможен неконтролируемый доступ к атрибуту.*



Task3.

Create a class called **Player** with parameters:

String name; String position; int number;

boolean isInjured; // true if player is injured, otherwise, false

Create default and parameterized constructors

String getPlayerData(); // this method returns all fields of **Player** class Create a class **Club** with parameters:

String name; Player []players;

Create default and parameterized constructors void printClubData();

This method prints club data, including **Player** data. For **Player** data, use getPlayerData() method of

**Player** object;

void printSquad(); // print squad of players from club

This method prints player’s number, name and position in following format if player is not injured.

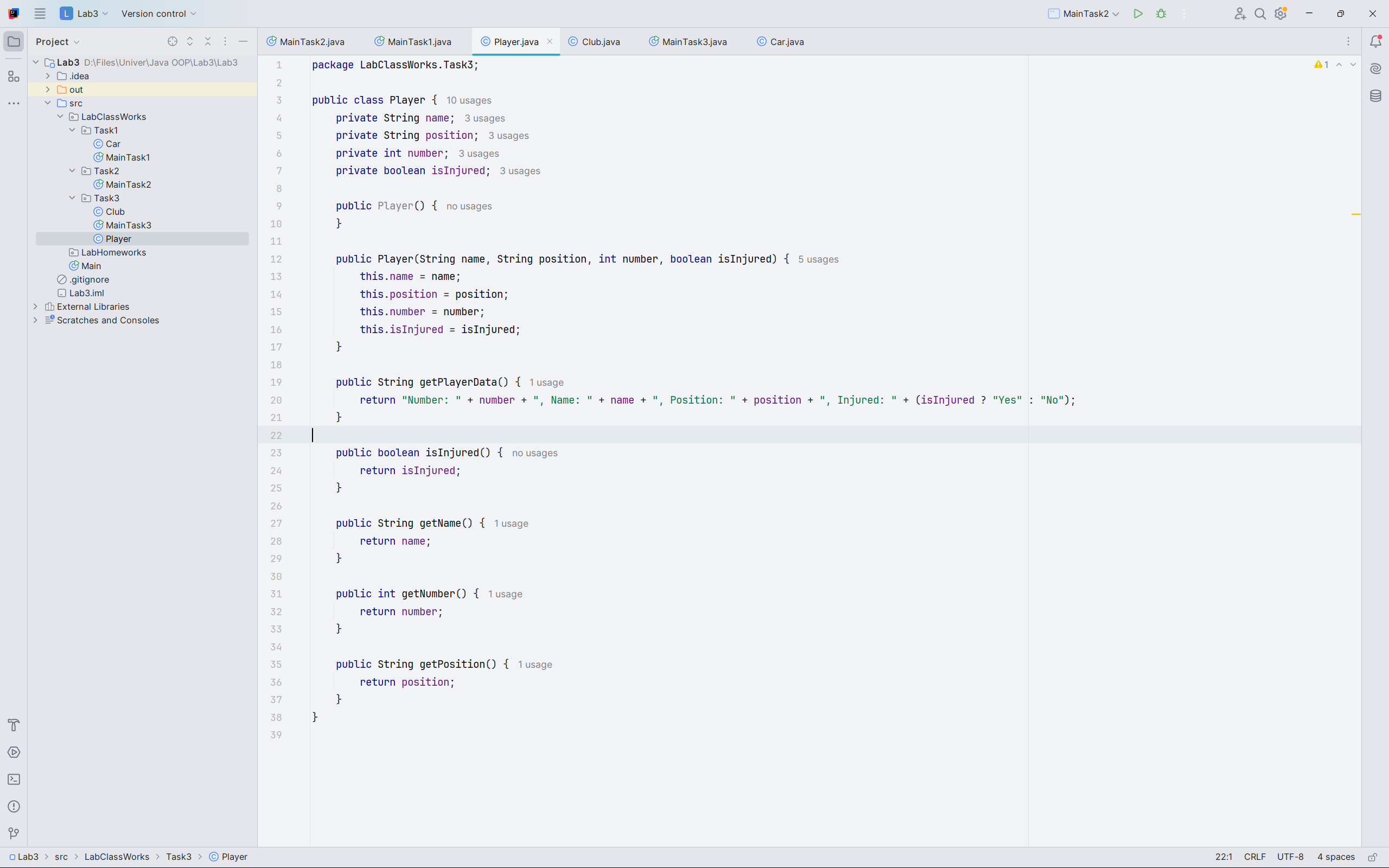
1 Neuer GK

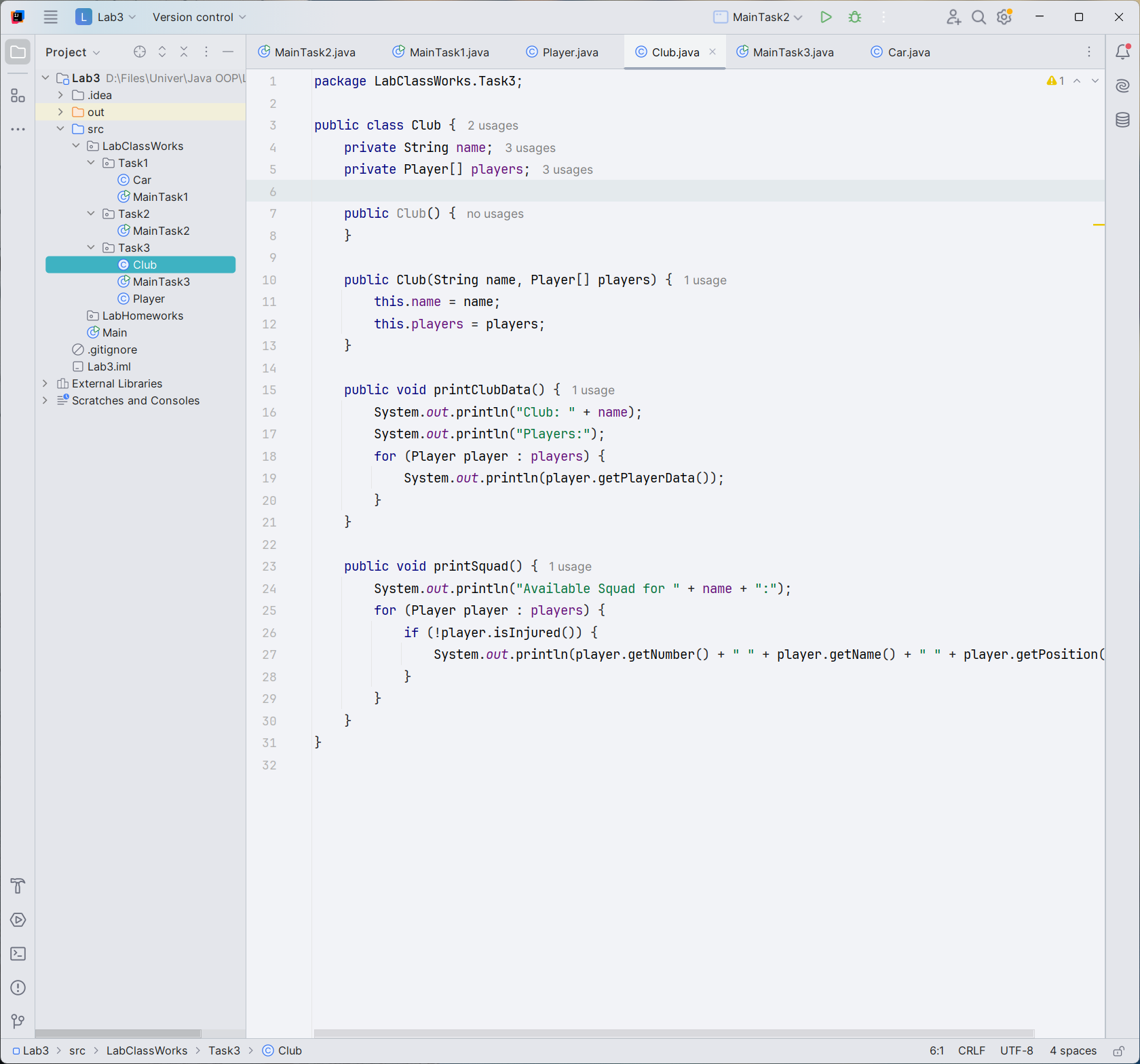
2 Carvajal RB

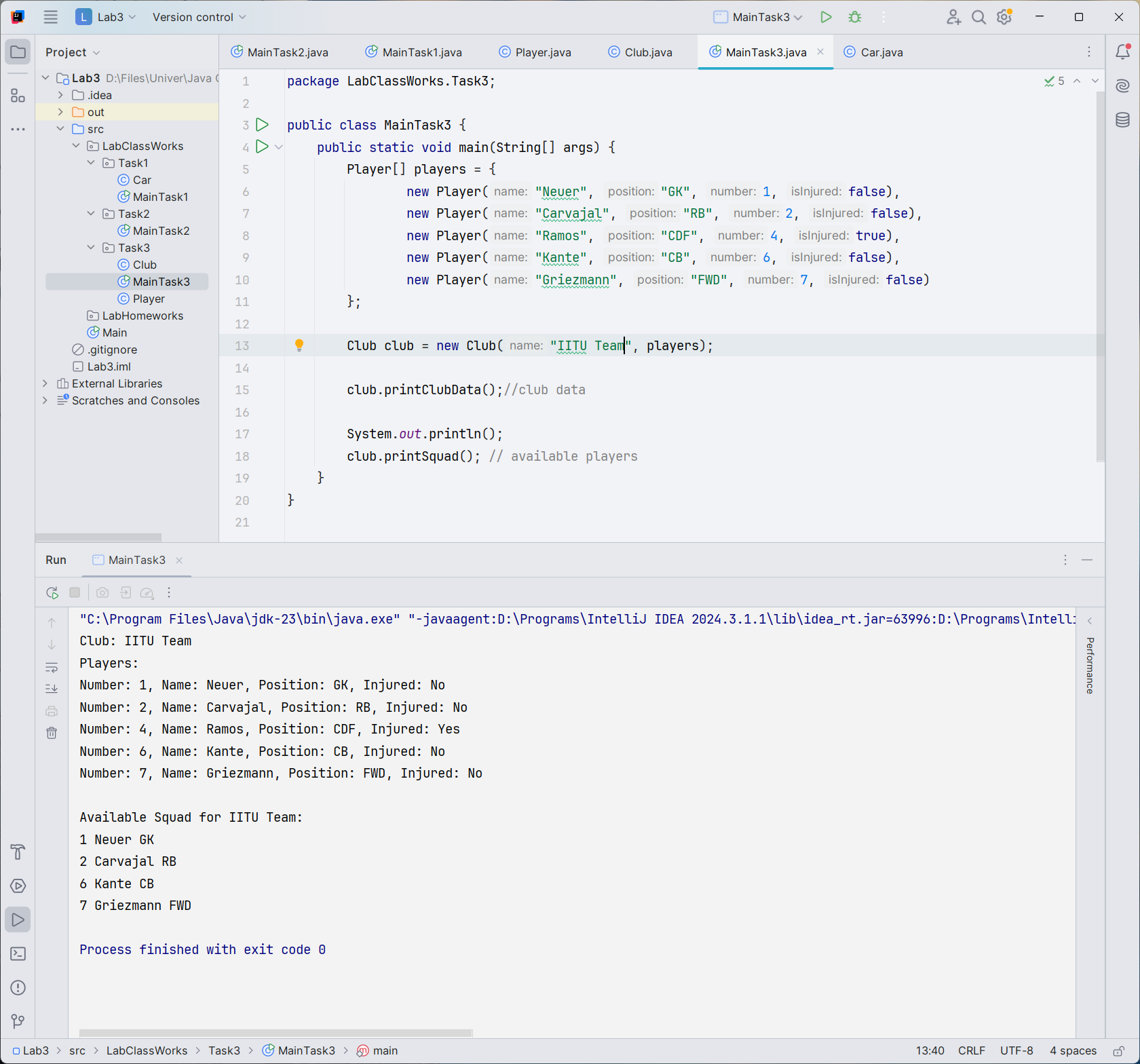
4 Ramos CDF

6 Kante CB

7 Griezmann FWD







File sources: