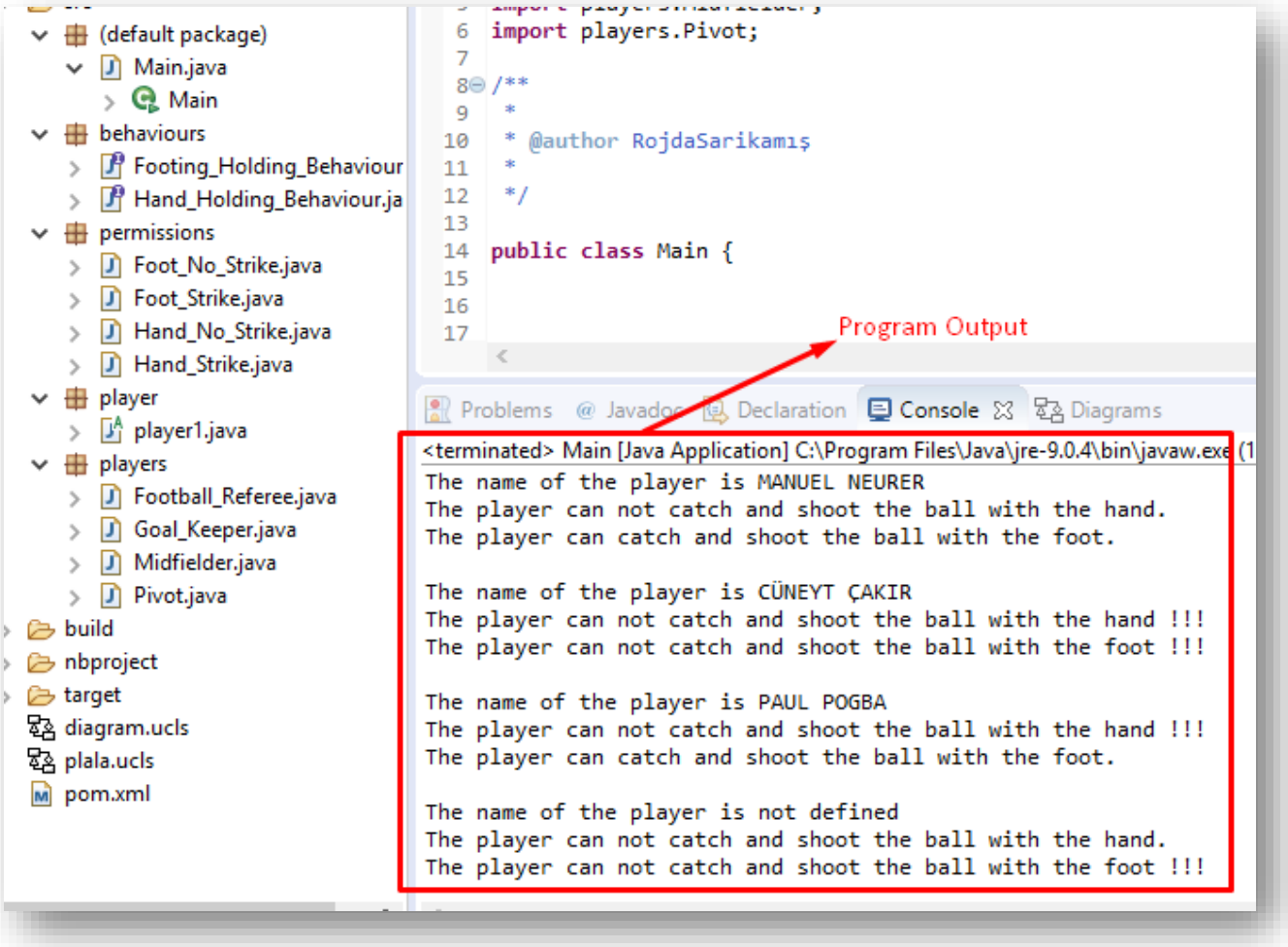
Statement of Work:

There are several types of players in this program. We can say there are players playing basketball or playing soccer. they also have different behaviors in each player. What I want is to call the methods from the classes that are related to the room, specifying only the properties for each different player type. I mean, I do not want to enter individual features for every different player. And behaviors can vary for each player. So while the goalkeeper can use his hand and foot in a soccer game, a midfielder can only use his foot. A referee can neither use his hand nor a foot while a basketball player can only use his hand. It is a very troublesome job to do this for each player individually. At this point I used a pattern to specify behaviors separately, I specified the player type separately. And I could easily do it with the hierarchy between them. Now it's just to add us new players. I could easily choose the behavior of the player.



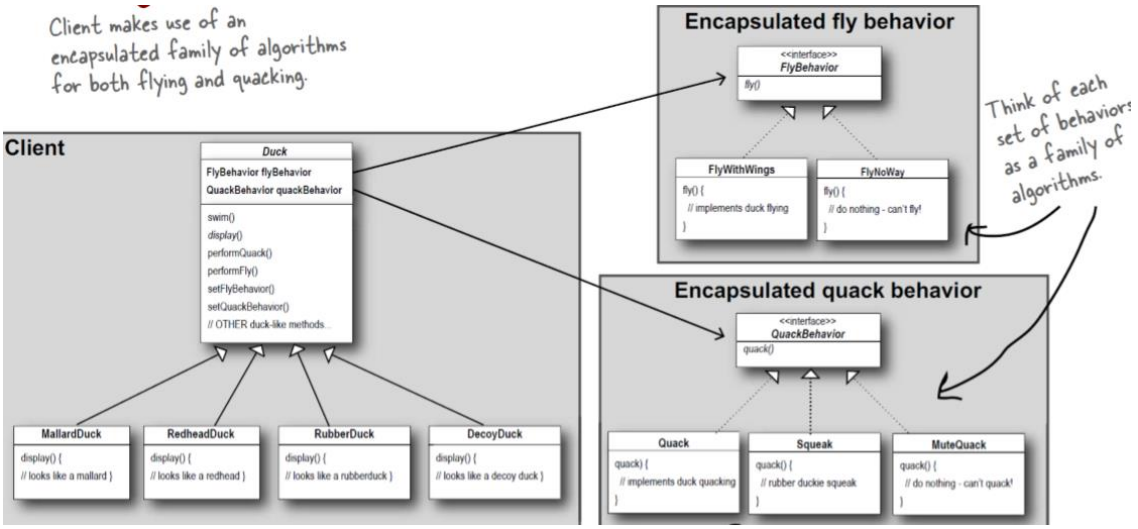
MANUEL NEURER is a goal keeeper and he can catch and shoot the ball with the hand and foot. CÜNEYT ÇAKIR is a football referee and he can not catch and shoot the ball with the hand and foot. PAUL POGBA is a midfielder and he can not catch and shoot the ball with the hand and he can catch and shoot the ball with the foot. So so..

Design Pattern:

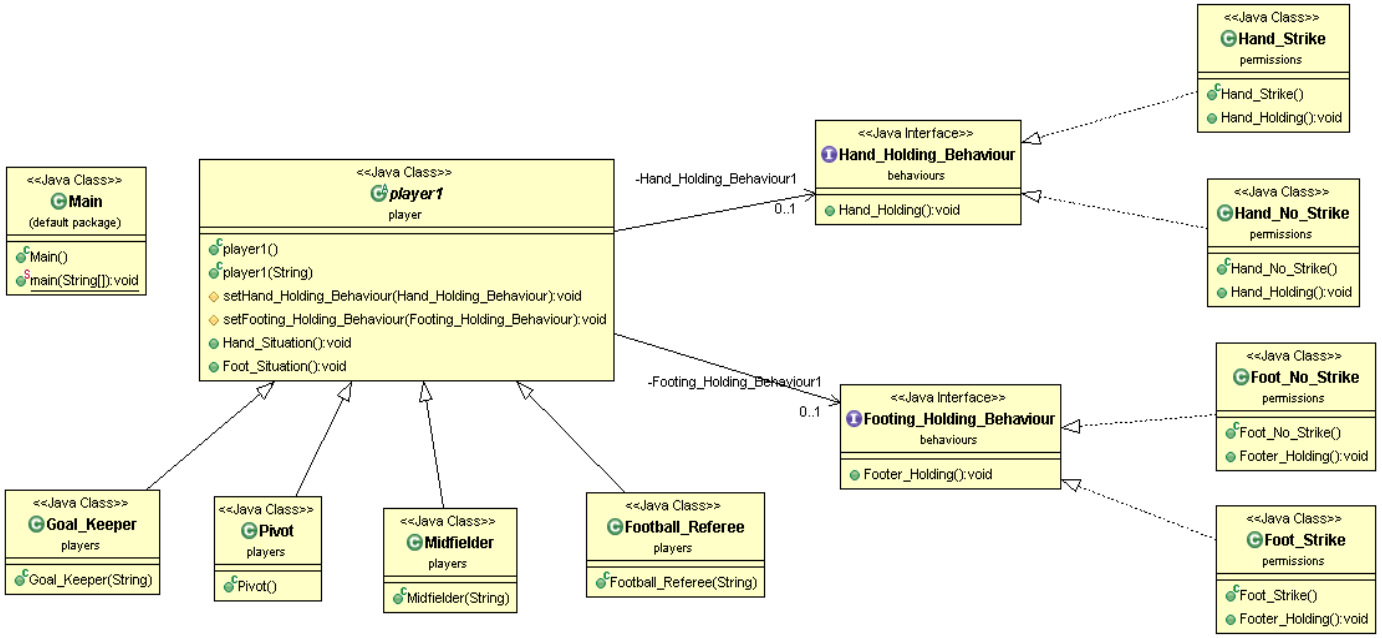
I chose The strategy pattern. The strategy pattern is a behavioral software design pattern that enables selecting an algorithm at runtime. The strategy pattern defines a family of algorithms, encapsulates each algorithm, and makes the algorithms interchangeable within that family. So I choose this pattern. This pattern can solve my problem. Strategy lets the algorithm vary independently from clients that use it.

For instance, a class that performs validation on incoming data may use the Strategy pattern to select a validation algorithm depending on the type of data, the source of the data, user choice, or other discriminating factors. These factors are not known until run-time and may require radically different validation to be performed. The validation algorithms (strategies), encapsulated separately from the validating object, may be used by other validating objects in different areas of the system without code duplication.

The essential requirement to implement the Strategy pattern is the ability to store a reference to some code in a data structure and retrieve it. This can be achieved by mechanisms such as the native function pointer, the first-class function, classes or class instances in object-oriented programming languages, or accessing the language implementation's internal storage of code via reflection.



UML:



Also I have the file ***"diagram.ucls"*** I Draw a detailed class diagram using UML Diagram Drawing Tool. You can look at uml in the program.

ROJDA SARİKAMIŞ

14888364024