Lab Exercise — Football Players

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. UML Diagrams
4. Sample Output
5. Test Code
6. Problem-Solving Tips

The test code represents a complete working C++ test program Read the problem description and examine the sample output; then study the test code. Using the problem-solving tips as a guide, write your C++ code. Compile and execute the program. Compare your output with the sample output provided.

Lab Objectives In this lab, you will practice:

- Using inheritance to create a football player hierarchy that includes a Player class, a FootballPlayer class, a GoalKeeper class, DefensivePlayer class, a MiddleFielderPlayer class and a ForwardPlayer class.
- Using private data members to limit access to data members.

Description of the Problem

The class hierarchy represent a football player team hierarchy. Every player belongs to a sepesific position and every one of them has spesific duties in the play. Your job is to implement this hierarchy and specific duties, using object oriented programming techniques (inheritance).
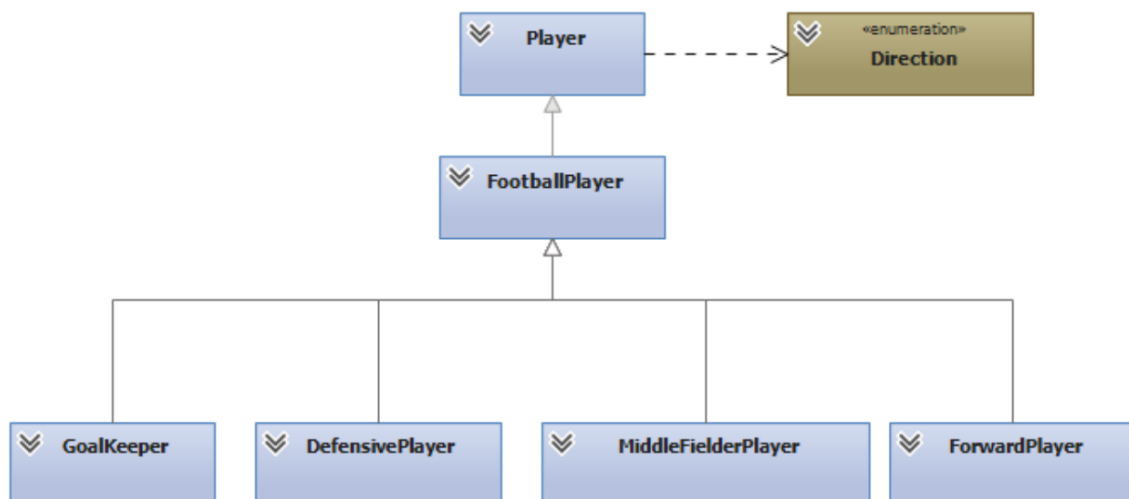
UML Diagrams
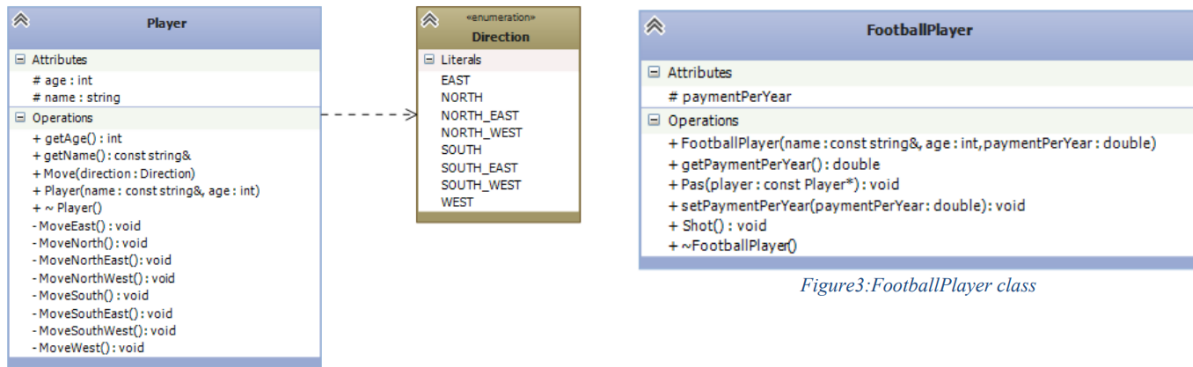

Figure 1:General UML Diagram

Player

Attributes
- # age : int
- # name : string

Operations
- + getAge() : int
- + getName() : const string&
- + Move(direction : Direction)
- + Player(name : const string&, age : int)
- + ~ Player()
- - MoveEast() : void
- - MoveNorth() : void
- - MoveNorthEast() : void
- - MoveNorthWest() : void
- - MoveSouth() : void
- - MoveSouthEast() : void
- - MoveSouthWest() : void
- - MoveWest() : void

Figure 2:Player class

«enumeration»
Direction

Literals
- EAST
- NORTH
- NORTH_EAST
- NORTH_WEST
- SOUTH
- SOUTH_EAST
- SOUTH_WEST
- WEST

FootballPlayer

Attributes
- # paymentPerYear

Operations
- + FootballPlayer(name : const string&, age : int, paymentPerYear : double)
- + getPaymentPerYear() : double
- + Pas(player : const Player*) : void
- + setPaymentPerYear(paymentPerYear : double) : void
- + Shot() : void
- + ~FootballPlayer()

Figure3:FootballPlayer class

GoalKeeper

Attributes

Operations
- + CalculatePaymentForThisYear() : double
- + FlyAndTryCatch() : void
- + GoalKeeper(name : const string&, age : int, paymentPerYear : double)
- + ~GoalKeeper()

Figure 4:GoalKeeper class

DefensivePlayer

Attributes
- - paymentPerMath : double
- - playedMatchCount : int

Operations
- + CalculatePaymentForThisYear() : double
- + Defense() : void
- + DefensivePlayer(name : const string&, age : int, paymentPerYear : double, paymentPerMatch : double)
- + getPlayedMatchCount() : int
- + setPlayerMatchCount(playedMatchCount : int) : void
- + TripUp(player : const Player*) : void
- + ~DefensivePlayer()

Figure 5:DefensivePlayer class

MiddleFielderPlayer

Attributes
- - assistCount : int
- - paymentPerAssist : double

Operations
- + CalculatePaymentForThisYear() : double
- + getAssistCount() : int
- + MiddleFielderPlayer(name : const string&, age : int, paymentPerYear : double, paymentPerAssist : double)
- + Press() : void
- + setAssistCount(assistCount : int) : void
- + TryThroughBall(player : const Player*) : void
- + ~MiddleFielderPlayer()
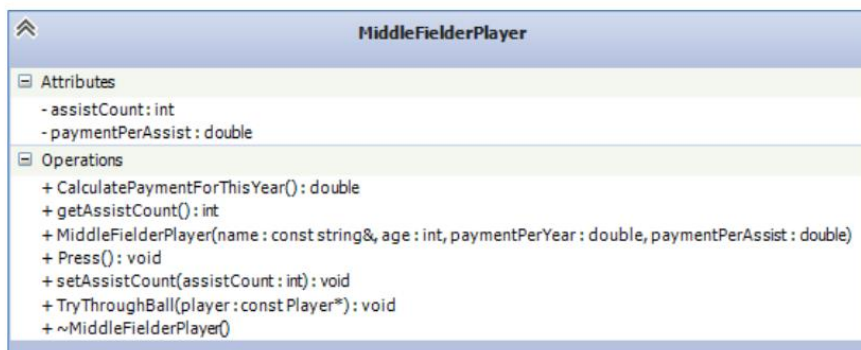
Figure 6:MiddleFielderPlayer

ForwardPlayer

Attributes
- - goalCount : int
- - paymentPerGoal : double

Operations
- + CalculatePaymentForThisYear() : double
- + ForwardPlayer(name : const string&, age : int, paymentPerYear : dou...
- + getGoalCount() : int
- + setGoalCount(goalCount : int) : void
- + ThrowYourSelf() : void
- + TryDribblePast(player : constPlayer*) : void
- + ~ForwardPlayer()
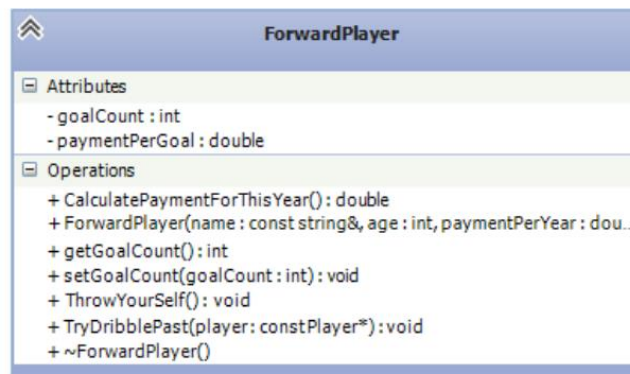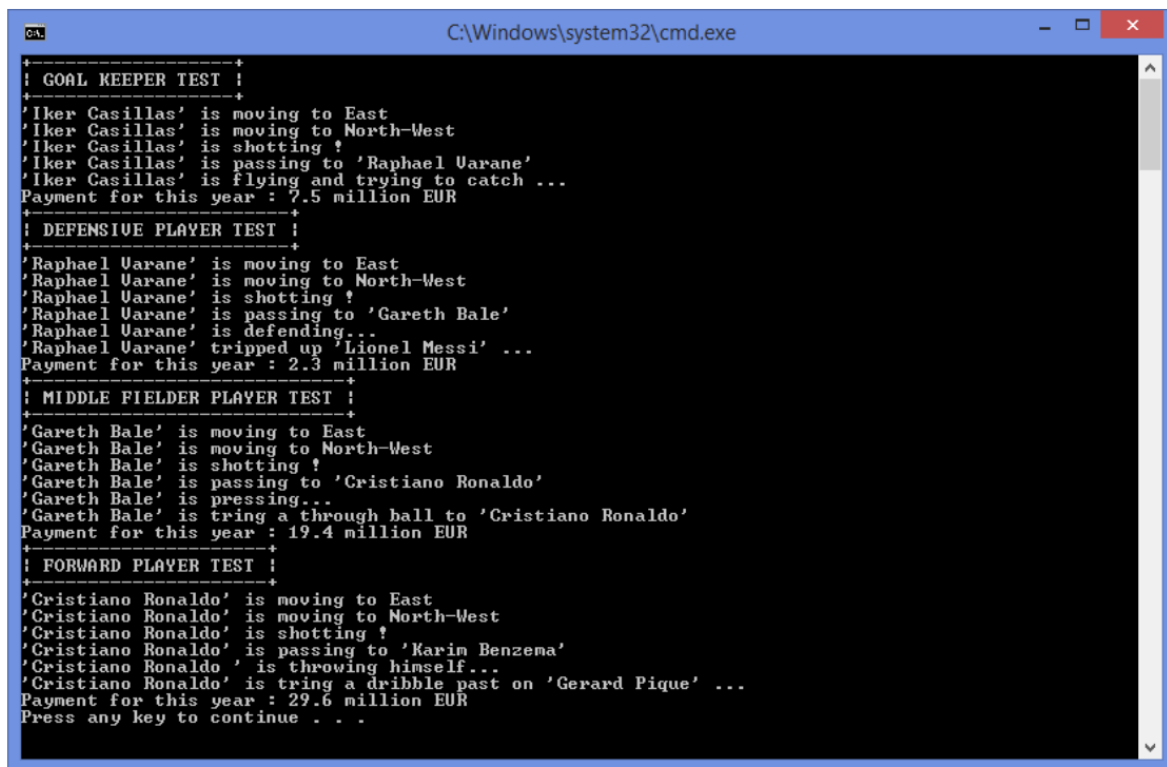
Figure 7:ForwardPlayer class

Sample Output

Problem-Solving Tips

1- Use given UML Diagrams and Test code as guide
2- Payment for a player can be calculated as following:

$$Payment_{GoolKeeper} = paymentPerYear$$

$$Payment_{DefensivePlayer} = paymentPerYear + paymentPerMatch * playedMatchCount$$

$$Payment_{MidFielderPlayer} = paymentPerYear + paymentPerAssist * assistCount$$

$$Payment_{ForwardPlayer} = paymentPerYear + paymentPerGoal * goalCount$$