SE 2217 Software Engineering Principles – Lab #12 (Labwork 6)

Class Diagrams

Class diagram is a graphical notation used to construct and visualize object oriented systems. A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

In this labwork, you are expected to generate a design type class diagram for the given problem.

Hockey League Simulation

Consider the Java code below.

```
abstract class Person {
    private String name;
    private String address;

    public Person(String name, String address) {
        this.name = name;
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
```

```java
class Player extends Person{

    private int playerNumber;
    private String position;
    private int speed;

    public Player(String name, String address, int playerNumber, String position, int speed) {
        super(name, address);
        this.playerNumber = playerNumber;
        this.position = position;
        this.speed = speed;
    }
    public void speedUp(){
        this.speed += 5;
    }
    public void slowDown(){
        this.speed -= 5;
    }
    public String shootAtGoal(){
        Random rand = new Random();
        int goalProb = rand.nextInt( bound: 100) + 1;
        if(goalProb < 75)
        {
            return "Player number " + playerNumber + " at " + position + " position shot, but missed.";
        }
        else {
            // TO DO: Increase the team score by 1.
            return "Player number " + playerNumber + " at " + position + " position shot and scored!";
        }
    }
    public String passTheBall(int toPlayer){
        return "Player " + playerNumber + " sent the ball to the player " + toPlayer;
    }
}
```

```java
class Coach extends Person{

    private int levelOfAccreditation;
    private int yearsOfExperience;

    public Coach(String name, String address, int levelOfAccreditation, int yearsOfExperience) {
        super(name, address);
        this.levelOfAccreditation = levelOfAccreditation;
        this.yearsOfExperience = yearsOfExperience;
    }

    public int getLevelOfAccreditation() {
        return levelOfAccreditation;
    }

    public void setLevelOfAccreditation(int levelOfAccreditation) {
        this.levelOfAccreditation = levelOfAccreditation;
    }

    public int getYearsOfExperience() {
        return yearsOfExperience;
    }

    public void setYearsOfExperience(int yearsOfExperience) {
        this.yearsOfExperience = yearsOfExperience;
    }
}
```

```java
class Captain extends Player {
    public Captain(String name, String address, int playerNumber, String position, int speed) {
        super(name, address, playerNumber, position, speed);
    }
}
```

```java
class Team {

    private String teamName;
    private int teamPoint;

    public Team(String teamName, int teamPoint) {
        this.teamName = teamName;
        this.teamPoint = teamPoint;
    }

    public void practice() {
        System.out.println(teamName + " is practicing.");
    }

    public void incrementScore(){
        this.teamPoint += 3;
        System.out.println(teamName + "wins the match and increased league point by 3.");
    }

    public void incrementScoreForDraw(String otherTeamName){
        this.teamPoint += 1;
        System.out.println(teamName + " drawn with team " + otherTeamName + "and both teams' league point " +
                "increased by 1.");
    }

    public String getName(){
        return teamName;
    }
}
```

```java
class League {
    public Team[] teams;
}
```

```java
class Match {
    private String finalScore = "0-0";
    private Team team1;
    private Team team2;

    public Match(Team team1, Team team2) {
        this.team1 = team1;
        this.team2 = team2;
    }

    public void setFinalScore(String finalScore)
    {
        this.finalScore = finalScore;
    }

    public String getFinalScore(){
        return finalScore;
    }

    public Team getWinner(){
        // Considering the finalScore attribute is "number"-"number", e.g. 4-3

        int dashIndex = finalScore.indexOf("-");
        int team1Score = Integer.parseInt(finalScore.substring(0, dashIndex));
        int team2Score = Integer.parseInt(finalScore.substring( beginIndex: dashIndex + 1));

        if(team1Score > team2Score)
        {
            return team1;
        }
        else if (team1Score < team2Score){
            return team2;
        }
        else return null;
    }}
```

Class Relations

It is known that **one** hockey league occurs with **four to seven** hockey teams and **fifteen** matches, and **each** team has a **single** captain. Players in teams composed to **six to twelve** players (including substitute players) for each team.

Each team led by a **single** coach, but coaches can train **multiple** teams (or does not train any team).

Two hockey teams play games against each other in any number.

Hint: If attributes in constructor are taken from a parent class with the super(…) method, there is no need to show the attributes again for the child class in the diagram.

Additionally, ignore the color difference in methods (blue- grey) and attributes (pink-grey), and take all of them as class methods/attributes.

In line with the given information, draw the necessary <u>design type</u> class diagram.

While exporting, please choose your export type as PDF (diagram per page) to gather all your work in a single document. Then, name it as **YourStudentNumber.pdf** and upload it under the related assignment.