DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis and Design

SPORTS LEAGUE MANAGEMENT SYSTEM

by

Cafer Hakan Acar

Muhammed Aydoğan

Umut Devrim Kaya

CHAPTER ONE

INTRODUCTION

The subject of this project is to create a league system in any sports branch we have determined to code. In line with this sport branch we choose, all the necessary information for the league will be taken from the file. After all teams and players have been created, our league will begin. Until the league system is completed, the teams will be able to transfer player according to their transfer budgets. At the end of this whole league, one team will win the league as a champion. The rules within the league will vary according to the sports branch we have determined to code. As a group, our current idea is to form a league system on the football branch. However, since the logic of the league systems is very close to each other, this algorithm we have developed will be very easily implemented in other systems.

The aim of our project is to be a champion according to the rules mentioned before.In this project you have your own team. You can transfer the players from other teams if the rules allow also create your players as you want and also project have a league and you can match the other teams like a real league and collect the points to end of the final match.The algorithm of that determines who will win will be determined according to the strengths of the teams. If user wants to be a champion of the league he/she should built the team carefully and should manage transfer relations.

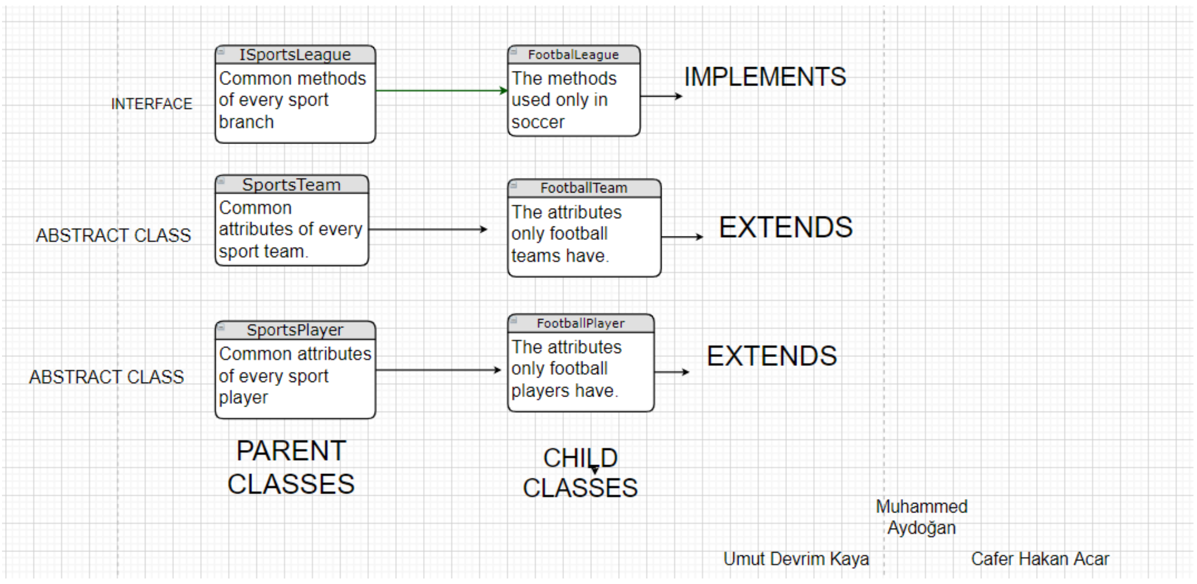
Project’s scope can be defined as customer’s order depending on the sport branch. Customer can always get feedback about the statement of their progress,league,transfer system or economical situation of the team. Given timeline allows to develop a well-designed system for our strong-designed project. In final stages of the project, it will be reviewed for additional features depend on the customer feedback. Any kind of algorithm problem will be handled from a programmer’s view according to algortihm efficiency techniques.

CHAPTER TWO

REQUIREMENTS

The strength of the teams will be formed according to the strength of the players in the team (attack, defense, morale, etc.). Team’s winning matches will contribute to the morale of the players and the team's budget. File processes will be impletemented using java reading functions and results will be written to the text file to store log.

Project’s main structure can be shown as follows:


INTERFACE
ISportsLeague
Common methods of every sport branch
FootbalLeague
The methods used only in soccer
IMPLEMENTS
ABSTRACT CLASS
SportsTeam
Common attributes of every sport team.
FootballTeam
The attributes only football teams have.
EXTENDS
ABSTRACT CLASS
SportsPlayer
Common attributes of every sport player
FootballPlayer
The attributes only football players have.
EXTENDS
PARENT CLASSES
CHILD CLASSES
Muhammed Aydoğan
Umut Devrim Kaya
Cafer Hakan Acar

```
public interface ILeague {

	public FootballTeam matchTeams(FootballTeam team1,FootballTeam team2);
	public void increasePoint(FootballTeam team);
	public void createAllTeams();
	public void createFixture();
	public boolean isBudgetEnough(FootballTeam team) {}
	public boolean isSizeofTeamConvenient(FootballTeam team) {}
	public void makeTransfer(Player playerToTransfer) {}

}
```

```
public abstract class SportsTeam {

	private String name;
	private int point=0;
	private int winCount=0;
	private int lossCount=0;
	private ArrayList<Player> players;
	private int MAX_PLAYER;
	private int teamStrength;
	private int teamMoral;
	private int budget;
	private int currentPlayer;



	public int getPoint() {
		return point;
	}
	public void setPoint(int point) {
		this.point = point;
	}

	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getWinCount() {
		return winCount;
	}
	public void setWinCount(int winCount) {
		this.winCount = winCount;
	}
	public int getLossCount() {
		return lossCount;
	}
	public void setLossCount(int lossCount) {
		this.lossCount = lossCount;
	}
	public ArrayList<Player> getPlayers() {
		return players;
	}
	public void setPlayers(ArrayList<Player> players) {
		this.players = players;
	}
	public int getMAX_PLAYER() {
		return MAX_PLAYER;
	}
	public void setMAX_PLAYER(int mAX_PLAYER) {
		MAX_PLAYER = mAX_PLAYER;
	}
	public int getTeamStrength() {
		return teamStrength;
	}
	public void setTeamStrength(int teamStrength) {
		this.teamStrength = teamStrength;
	}
	public int getTeamMoral() {
		return teamMoral;
	}
	public void setTeamMoral(int teamMoral) {
		this.teamMoral = teamMoral;
	}
	public int getBudget() {
		return budget;
	}
	public void setBudget(int budget) {
		this.budget = budget;
	}
	public int getCurrentPlayer() {
		return currentPlayer;
	}
	public void setCurrentPlayer(int currentPlayer) {
		this.currentPlayer = currentPlayer;
	}

}

public abstract class SportsPlayer {

	private int moral;
	private int strength;
	private String name;
	private int age;
	private SportsTeam team;



	public int getMoral() {
		return moral;
	}
	public void setMoral(int moral) {
		this.moral = moral;
	}
	public int getStrength() {
		return strength;
	}
	public void setStrength(int strength) {
		this.strength = strength;
	}
	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;
	}
	public SportsTeam getTeam() {
		return team;
	}
	public void setTeam(SportsTeam team) {
		this.team = team;
	}

}


public class FootballLeague implements ILeague {


	private ArrayList<FootballTeam> teams;




	@Override
	public FootballTeam matchTeams(FootballTeam team1, FootballTeam team2) {



	}
	@Override
	public void increasePoint(FootballTeam team) {


	}
	@Override
	public void createAllTeams() {

	}
	@Override
	public void createFixture() {


	}
	@Override
	public boolean isBudgetEnough(FootballTeam team) {
		// TODO Auto-generated method stub
		return false;
	}
	@Override
	public boolean isSizeofTeamConvenient(FootballTeam team) {
		// TODO Auto-generated method stub
		return false;
	}
	@Override
	public void makeTransfer(Player playerToTransfer) {
		// TODO Auto-generated method stub

	}
```

```
}



public class FootballTeam extends SportsTeam {



	private int tieCount;
	private int goalScored;
	private int goalTaken;

	private int determineTeamPower(ArrayList<Player> players) {

	}

	private int determineTeamMoral(ArrayList<Player> players) {

	}


}

public class Player extends SportsPlayer {

	private String position;
	private boolean foot;
	public String getPosition() {
		return position;
	}
	public void setPosition(String position) {
		this.position = position;
	}
	public boolean isFoot() {
		return foot;
	}
	public void setFoot(boolean foot) {
		this.foot = foot;
	}

}
```