프로그래밍 언어론 과제

디지털 꾸러미 설계

2022년 카타르 월드컵 포지션 별 선수 정보 꾸러미 및 소속 국가 팀 정보 꾸러미 만들기

중앙대학교 소프트웨어학부

20193802

조명근

목 차

1.서론

객체 지향 프로그래밍(OOP)은 데이터와 동작(메서드)을 모두 포함하는 데이터 구조인 "객체" 개념을 기반으로 하는 프로그래밍 패러다임이다. 객체 지향 프로그래밍 등장 이전에 프로그래밍 패러다임은 절차 지향 프로그래밍이었다. 절차 지향 프로그래밍에서는 특정 작업을 수행하는 프로시저 또는 함수로 구성된 코드를 작성하고, 이때 문제를 더 작고 관리하기 쉬운 조각으로 나누고 각 조각을 개별적으로 해결하는 것에 중점을 뒀다. 이 접근 방식은 더 작고 독립적인 작업으로 쉽게 나눌 수 있는 문제를 해결하는데 주로 사용되었다.

반대로 객체 지향 프로그래밍은 데이터와 동작을 캡슐화하는 자체 포함 단위인 "객체" 개념을 기반으로 하는 프로그래밍 패러다임이다. 객체 지향 프로그래밍에서 프로그래머는 객체의 데이터 유형과 적용할 수 있는 메서드를 정의한다. 이런 객체의 데이터와 메서드를 바탕으로 객체가 복잡한 방식으로 서로 상호 작용할 수 있도록 하는 상속, 다형성 및 캡슐화와 같은 기능을 바탕으로 프로그램이 구성된다.

캡슐화: 객체 지향 프로그래밍은 캡슐화를 사용한다. 즉, 개체의 구현 세부 정보가 외부 세계에서 숨겨지고 개체의 메서드를 통해서만 액세스할 수 있다. 이렇게 하면 개체의 데이터를 보호하고 프로그램의 나머지 부분에 영향을 주지 않고 구현을 쉽게 변경할 수 있다. 구조 프로그래밍에는 이 개념이 존재하지않는다.

상속: 객체 지향 프로그래밍을 사용하면 개체가 부모 개체에서 속성과 동작을 상속할 수 있으므로 코드를 재사용할 수 있으므로 시간과 노력을 절약할 수 있다. 구조 프로그래밍에는 이 개념이 존재하지 않는다.

다형성: 객체 지향 프로그래밍을 사용하면 호출되는 상황에 따라 이름은 같지만 동작이 다른 여러 메서드를 정의할 수 있다. 이것을 다형성이라고 한다. 구조 프로그래밍에는 이 개념이 존재하지 않는다.

추상화 : 실제 실세계의 존재하는 것을 객체 또는 개념의 단순화된 표현으로 생성하는 것이다. 컴퓨터과학에서 많이 쓰이는 개념이고, OOP에서는 추상화는 개체의 구현 세부 정보를 숨기고 필요한 특성과 동작만 노출하는 데 사용된다.

따라서 이런 상속, 다형성, 캡슐화, 추상화와 같은 기능은 문제 해결을 위해 객체와 객체의 상호 작용을 사용할 수 있게 해주고, 이는 객체 지향 프로그래밍을 강력하게 만들어준다.

따라서 객체 지향 프로그래밍은 재사용 가능한 코드 생성과 프로그래밍에 대한 보다 모듈화된 접근 방식을 허용하므로 보다 복잡한 소프트웨어 시스템을 설계하는 데 자주 사용된다. 또한 개체 간의 계층적 관계를 생성하고 코드를 더 작고 관리하기 쉬운 단위로 모듈화할 수 있으므로 유지보수가 쉽고 이에 따라서 크고 복잡한 프로젝트를 처리하는 데 더 적합하다.

따라서 이런 객체 지향 프로그래밍의 특징을 이용하여 Kaggle의 큰 데이터셋을 객체화하하여 객체 지향 프로그래밍들의 특징을 활용하고 이런 특징들을 확인하는 것이 이번 프로젝트의 목표이다.

따라서 추상화를 이용하여 실제 세계에 존재하는 모든 데이터를 객체의 단순화된 표현으로 속성과 동작을 재정의하고, 상속을 바탕으로 데이터 서로 간의 관계가 있는 경우 상속 단위로 묶거나 객체에 포함시켜 데이터 간의 관계를 정의할 것이다. 이렇게 정의한 데이터를 바탕으로 이에 따른 동작을 구현하고. 동작의 다형성(오버라이딩)과 멤버 변수의 캡슐화(private, getter,setter 메서드)를 바탕으로 큰 데이터셋을 객체 지향 프로그래밍의 특징을 이용하여 객체 단위로 프로그래밍하는 것이 이번 프로젝트의 목표이다.

2. 배경 – 데이터를 선택한 배경

2022년 현재 전 세계는 코로나-19 대유행 이후 처음으로 열리는 전 지구인의 축제인 월드컵을 즐기고 있다. 월드컵은 32개 국가의 여러 선수들이 경쟁하며 우승을 목표로 경기를 치룬다. 이 때 매 경기마다 축구공에 부착된 센서, 선수들의 몸에 착용된 센서를 바탕으로 선수들에 대한 데이터를 수집한다. 이런 수집된 정보들은 선수들 각각의 데이터로 저장이 되고 이는 선수들을 경기 이후 바라보

는 지표가 될 수 있다. 이 때, 정말 거의 모든 세세한 정보들이 선수들의 경기와 함께 기록된다. 예를 들어 골을 넣은 횟수, 어시스트를 한 횟수와 같이 간단한 정보도 담기지만, XG와 같이 특정 위치와 특정 상황에서 골을 넣기 위해 슈팅하였을 때 득점할 확률을 나타낸 정보 , 패스 횟수, 패스가 이동한 길이 , 슛 찬스를 만드는 움직임 , 골 찬스를 만드는 움직임 ,공중볼 경합 횟수와 같이 정말 많은 데이터가 선수와 함께 기록이 된다.

야구에서는 세이버매트릭스라는 이론이 있는데. 이 세이버매트릭스는 게임 이론과 통계학적 방법론을 도입하여 선수의 가치에 대한 객관적인 평가를 돕는다. 이 때 세이버 매트릭스는 선수와 관련된 모든 데이터를 바탕으로 이를 포지션에 맞게 추상화하여 이를 바탕으로 선수의 가치에 대한 평가를 진행한다. 이와 비슷하게 현대 축구에서도 데이터를 바탕으로 선수의 가치를 평가한다.

그래서 kaggle에서 이번 월드컵 경기에서도 이런 데이터들이 없을까 검색을 진행하였는데, 월드컵 경기에 참여한 모든 선수들의 데이터를 담은 데이터셋을 발견하였다. 이 데이터 안에는 defense.csv, passing.csv, gca.csv(goal created activity) 와 같은 다양한 데이터 셋이 존재하였고, 선수 이름과 함께 모든 csv파일을 합치면 273개의 속성 즉 선수 별로 273개의 지표가 즉 정말 많은 데이터가 있었다.

그림 1

따라서 이를 바탕으로 선수에게 필요한 지표를 추상화하여 필요한 데이터만을 가져온 후, 이를 바탕으로 선수의 이번 월드컵 내에서의 가치를 평가하면 좋을 것 같다고 생각했다. 또한 포지션 별로 중요한 능력치가 다 다르기 때문에 공통된 선수 정보는 상위 클래스에 넣어두고, 포지션 별 정보를 상속을 통해 표현하기도 좋다고 생각하였다. 또한 이런 선수 정보를 바탕으로 선수들의 모음인 팀을

만들어서 캡슐화도 가능할 것 같다고 생각하였다. 따라서 이 프로젝트의 주제를 “2022년 카타르 월드컵 포지션 별 선수 정보 꾸러미 및 소속 국가 팀 정보 꾸러미 만들기”로 결정하였다.

3. 구현 내용

Dataset의 정보를 abstract하는 과정

먼저 데이터셋을 읽어서 선수가 공통으로 갖고 있는 정보들을 찾아보았다. 그래서 player_stats라는 csv파일을 읽어서 이름, 팀, 생년월일, 소속팀, 국가를 얻어내서 이를 선수가 공통으로 갖고 있는 특징으로 정의했다. 또한 포지션별 특징을 추상화하기 위해 노력했다. 포지션별로 중요한 능력치가 다 다르기 때문에 데이터셋에서 어떤 것을 가져올 지를 결정하기 위해서 피파 온라인 4라는 게임을 참고했다. 여러 포지션에서 중요하게 가져야할 스탯을 도출할 수 있는 정보를 *그림 1.* 에 있는 csv files에서 parsing한 후 정보를 선정한 후 가져와야 했다. 따라서 포지션 별로 가져와야 할 정보를 다음과 같이 정의하였다.

1. 공격수

공격수를 추상화한다고 생각하면, 골을 만들 수 있는 능력, 볼 컨트롤, 오프더볼 움직임과 같은 능력이 중요하다고 생각하였다.

따라서 이와 관련된 데이터인 골을 몇 개 넣었는지?, 90분 경기당 몇 번 슛을 했는지?, 90분 경기당 유효슈팅을 얼마나 만들었는지?, 패스를 몇 번 받았는지?, 드리블의 성공 퍼센트가 몇인지?의 정보를 여러 데이터셋을 파싱하여 정보를 가져왔다. 총 5개의 정보(goals, shoots on per 90 , shoots on target on per 90, pass received, dribbles completed percent)를 가져와서 이를 공격수를 평가하는 지표로 판단하였다.

2. 미드필더

미드필더는 볼 컨트롤, 활동량 , 패스 횟수와 같은 능력이 중요하다고 생각하였다.

따라서 이와 관련된 데이터인 패스 성공률이 얼마인지? , 총 패스가 이루어진 길이가 몇인지? , 공격과 관련된 패스가 얼마나 이루어졌는지? , 드리블의 성공 퍼센트가 몇인지?, 공을 소유한 횟수가 몇

회인지?, 패스를 받은 횟수가 몇인지?의 정보를 여러 데이터셋을 파싱하여 이에 관련된 정보를 가져왔다. 총 6개의 정보(passes_pct, passes total distance, passes about attack, dribbles completed pct, touches, pass received)를 가져와서 이를 미드필더를 평가하는 지표로 판단하였다.

3. 수비수

수비수는 골을 향한 공을 막거나, 태클로 공을 뺏거나, 공을 상대방에게서 가져오는 수비 능력이 중요하다고 생각하였고, 공격을 지연하는 움직임이나, 체격을 바탕으로 한 공중볼 소유와 같은 능력이 중요하다고 판단하여서 위의 설명을 담은 6개의 정보 (blocks- 공의 막은 횟수 , tackle_won – 태클 성공 횟수, intercept- 상대방에게서 공 소유를 뺏는 행위, clearance – 수비 지역에서 공 걷어낸 횟수, arials_won_pct - 공중볼 경합 승리 횟수, ball_recovery – 수비 리커버리를 간 횟수)를 여러 데이터셋을 파싱하여 이에 관련된 정보를 가져와서 이를 수비수를 평가하는 지표로 판단하였다.

4. 골키퍼

골키퍼는 골을 막은 횟수, 골키퍼에게서의 패스 성공률이 중요하다고 생각하였다.

따라서 골키퍼는 골 이나 슛 찬스가 상대방에게서 발생했을 때 막은 횟수가 얼마인지? , 경기 당 몇 번의 클린시트가 있었는지? , 골키퍼의 발에서 출발한 패스의 정확도의 퍼센트가 얼마인지? 의 정보를 골키퍼 정보가 담긴 데이터셋을 파싱하여 이에 관련된 정보를 가져왔다. 총 3개의 정보 (save_pct, clean_sheet_pct , gk_pass_pct)를 가져와서 이를 골키퍼를 평가하는 지표로 판단하였다.

Class 구현 부 – 선수에 관한 구현

먼저 모든 포지션의 부모 클래스가 되는 player class를 abstract class로 정의하였다. 이 player class의 정의가 굉장히 중요한데, 이는 자식의 메서드와 데이터들을 결정할 수 있기 때문이다.

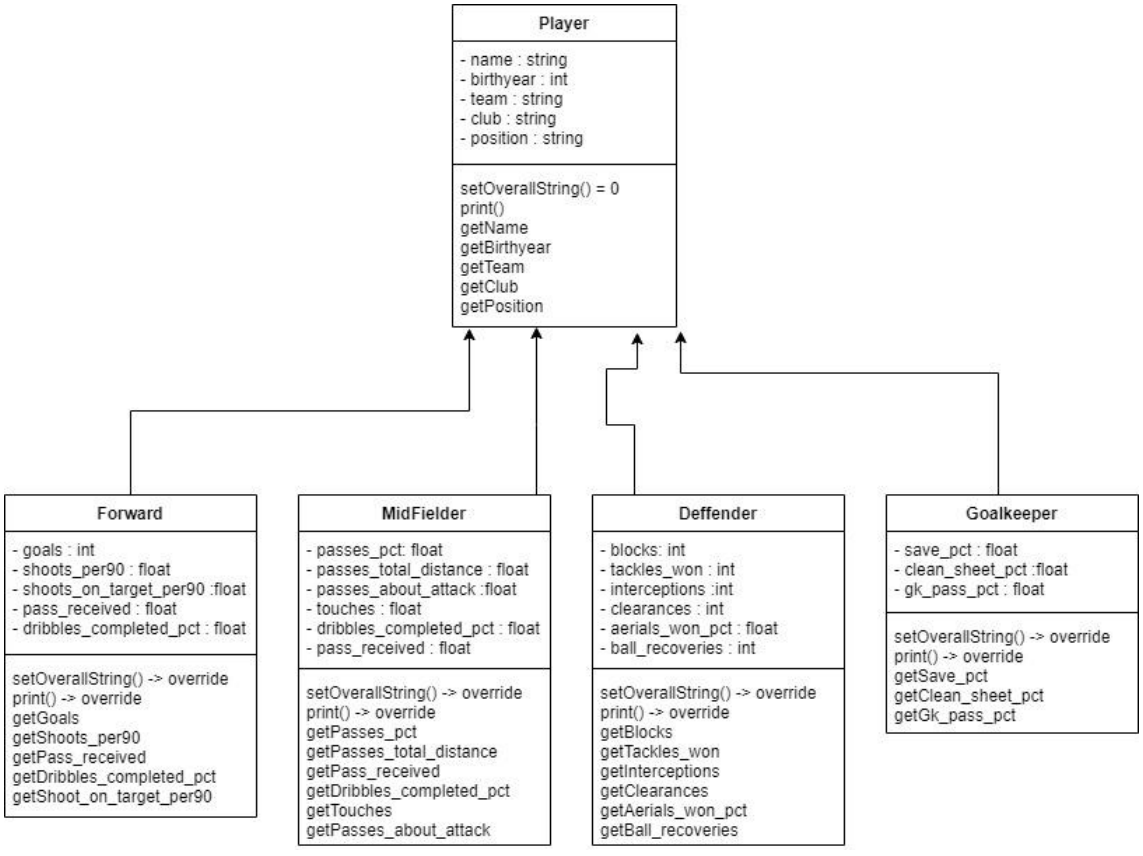
따라서 이 부모 클래스인 player에는 포지션과 상관 없는 즉 상속과 상관 없는 데이터인 이름, 팀, 생년월일, 소속팀, 국가가 private 멤버 변수로 정의되어 캡슐화 되어있다. 또한 abstract class는 무조건 상속이 되어야하기 때문에 position을 멤버 변수로 넣는데, 이 때, 상속된 class에서 자동으로 할당될 수 있게 구현하였다. 따라서 이에 대한 정보를 받을 수 있게 getter를 각각 멤버 변수에 구현하였다. 선수에 관한 class 구현에서는 동적으로 할당하는 멤버 변수가 존재하지 않기 때문에

따로 소멸자는 구현하지 않았다.

상속 과 다형성을 구현하기 위해서 Player를 abstract 클래스로 만들었는데, abstract class를 만들기 위해서는 pure virtual function을 부모 클래스에서 정의한 후 자식 클래스에서 이에 대한 구현을 해줘야 한다. 따라서 Player를 상속한 Forward, Defense, Midfielder, Goalkeeper에서 선수 포지션 별로 필요한 정보들이 다 다르기 때문에 이 각각 다른 정보를 파싱하고 생성자에 넣는 동작을 pure virtual function으로 만들어서 각각 상속된 class에서 구현하였다. 이는 setOverallRating()함수로 정의 되었다. 이 뿐만 아니라 virtual function print를 만들어서 각각 상속된 class에서 이를 변형하여 쓸 수 있도록 하였다. 이런 구현은 override를 통해 다형성을 구현한 행위이다.

또한 class에서 포지션 별 정보를 담은 멤버 변수들이 private에 의해 캡슐화 되어 있기 때문에 이를 getter 메서드를 바탕으로 제공할 수 있게 구현하였다.

따라서 Player에 와 포지션 별 상속에 관한 class의 UML는 다음과 같다.



Class 구현 부 - 팀에 관한 구현

이렇게 선수들의 포지션 별 정보를 class로 만들어서 객체를 생성할 수 있게 된다면, 이런 정보가 잘 담겨 있는지 테스트를 할 수 있다. 또한 선수들을 묶어 놓은 팀을 구현할 수 있다. 축구 팀은 포지션 별 선수들의 그룹으로 이루어져 있기 때문에 Team이라는 클래스를 정의하여 각 포지션 별로 속한 선수들의 그룹을 만들어주었다. 이를 구현하기 위해 먼저 멤버변수에 Forward, Midfielder, Goalkeeper, Defender 객체 배열을 정의하였고 팀의 이름을 정의하였다. 이 class의 목표는 소속된 나라의 대표팀을 표현하는 것이다. 따라서 선수 객체에서 국가 정보를 가져온 후 포지션 별로 객체 배열에 담아야 한다. 이 때 동적으로 배열의 메모리를 할당하기 위해 생성자에서 해당 국가의 각 포지션 별로 몇 명의 선수가 있는지 확인한 후 이에 맞게 동적으로 배열을 할당해준다. 그 후 포지션 별로 선수 객체를 넣어주어 객체 배열 안에 선수 객체를 할당해주면 team의 멤버 변수에 팀 내 포지션 별 선수를 확인할 수 있다. 멤버 변수는 이렇게 정의할 수 있고, 이 멤버 변수를 다루기 위해 메서드 또한 정의해 줘야한다.

그래서 정의된 메서드는 각 멤버변수에 대한 getter, 그리고 각 포지션 별 선수를 print해주는 메서드, 그리고 전체 팀을 print해주는 메서드가 존재한다. 또한 동적으로 배열을 할당했기 때문에 소멸자 또한 정의해야하는데, 동적으로 할당된 선수들의 객체 배열을 각각 소멸자에서 delete를 통해 지워주었다.

Main 구현 부

메인 함수에서는 player_stats.csv 파일 즉 월드컵의 참가한 모든 선수들의 정보를 담은 csv파일을 파싱하여 각각 포지션 선수 객체 마다 vector에 선수 객체를 넣어준다. 따라서 4개의 벡터에 포지션별로 모든 선수의 객체가 담긴다.

그 후 프로그램이 잘 돌아가는지 테스트 하기 위해 한국 대표팀의 정보를 담을 Team 클래스 teamkr 객체를 만들어주고 정보가 잘 들어가 있는지 확인하기 위해 print 메서드를 불러서 프린트 해준다. 또한 이번 월드컵 우승팀인 아르헨티나 팀의 정보를 Team class를 통해 arge 객체를 만들어주고, 정보가 잘 들어가 있는지 확인하기 위해 print 메서드를 불러서 프린트 해준다. 이 내용은 결과 부분에서 더 자세히 확인할 수 있다.

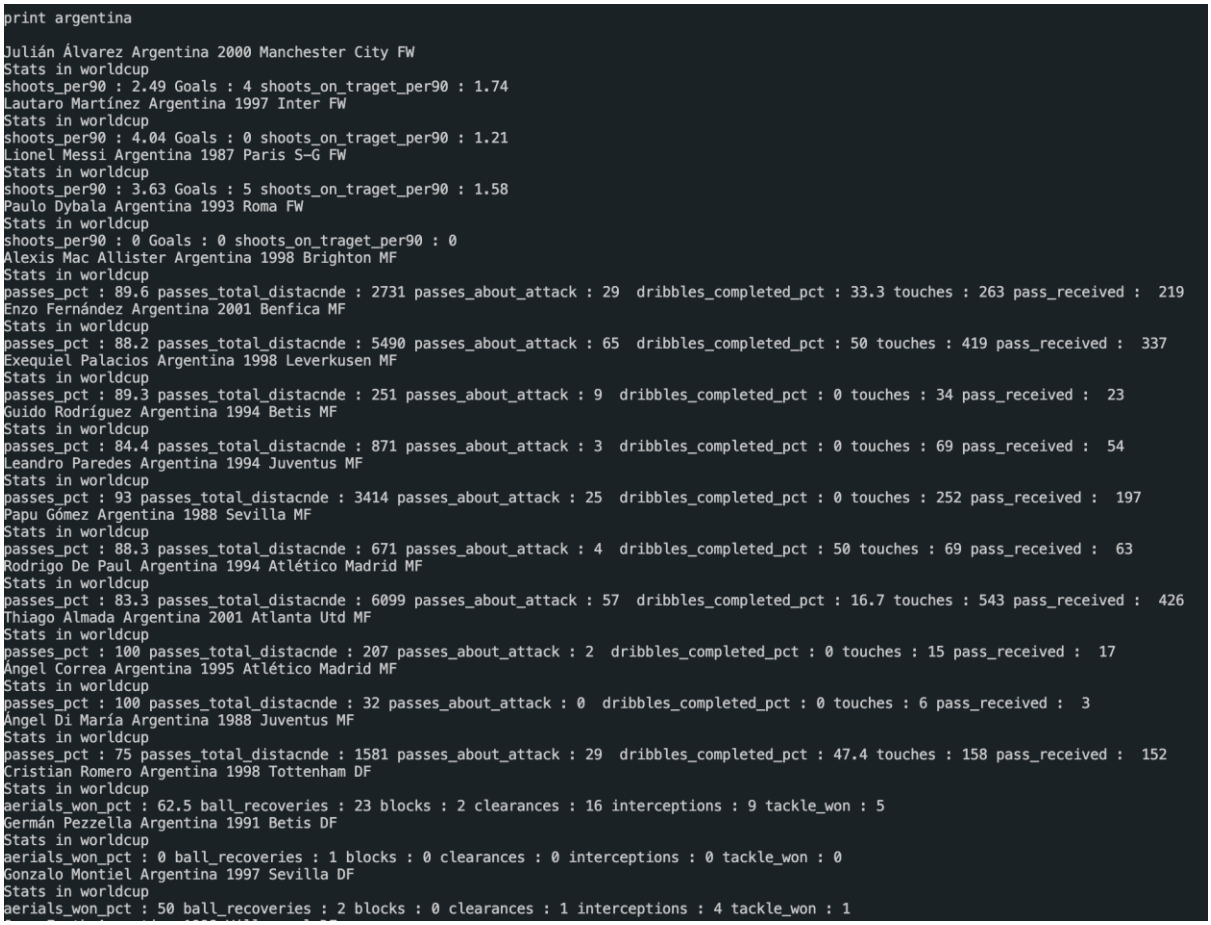
4. 결과

```
string kd = "Korea Republic";
Team teamkr(kd, fw, mf, df, gk);
// 모든 선수 출력
cout << endl;
teamkr.print();
cout << endl;

// 우승팀 아르헨티나 정보 적재
cout << "print argentina" << endl;

string arge = "Argentina";
Team teamarg(arge, fw, mf, df, gk);
cout << endl;
// 아르헨티나 대표팀 내 모든 선수 출력
teamarg.print();
cout << endl;
cout << "print fw in argentina";
teamarg.printFW();

cout << "ended" << endl;

cout << "test player object";
// 메서드 작동 여부 check team 객체의 공격수 0번째 값
Forward test = teamkr.getFwarr()[0];
test.print();

Midfielder test1 = teamkr.getMfarr()[1];
test1.print();

Goalkeeper test2 = teamarg.getGkarr()[0];
test2.print();
```

main함수 내에서 프로그램 내의 class가 잘 실행이 되는지 확인하기 위해서 한국 선수단의 정보를 담을 Team class의 teamkr객체를 선언해주었다. 그 이후 이번 월드컵 우승팀인 아르헨티나 대표팀의 정보를 담을 Team class의 teamarg객체를 선언해주었다. 각각 선수단의 모든 선수를 print할 수 있도록 print 메서드를 실행해주었고, 객체 내 다른 메서드도 잘 작동하는지 확인하기 위해 선수 객체 배열을 받아오는 getFwarr함수, getMfarr()함수를 실행하고, 이 배열 안의 선수정보가 잘 들어가 있는지 확인하기 위해서 선수객체의 print 함수를 실행하였다. 이외에도 많은 객체의 행동을 정의하는 메서드들(getter 등)이 많지만, print함수로 표현이 가능해서 이렇게 테스트를 진행하였다.

```
Cho Guesung Korea Republic 1998 Jeonbuk FW
Stats in worldcup
shoots_per90 : 2.95 Goals : 2 shoots_on_traget_per90 : 1.64
Cho Yumin Korea Republic 1996 Daejeon Citizen FC FW
Stats in worldcup
shoots_per90 : 0 Goals : 0 shoots_on_traget_per90 : 0
Hwang Ui-jo Korea Republic 1992 Olympiacos FW
Stats in worldcup
shoots_per90 : 0.85 Goals : 0 shoots_on_traget_per90 : 0
Kwon Chang-hoon Korea Republic 1994 Sangju Sangmu FW
Stats in worldcup
shoots_per90 : 0 Goals : 0 shoots_on_traget_per90 : 0
Na Sang-ho Korea Republic 1996 FC Seoul FW
Stats in worldcup
shoots_per90 : 0.76 Goals : 0 shoots_on_traget_per90 : 0.76
Son Heung-min Korea Republic 1992 Tottenham FW
Stats in worldcup
shoots_per90 : 2.25 Goals : 0 shoots_on_traget_per90 : 0.75
Hwang Hee-chan Korea Republic 1996 Wolves MF
Stats in worldcup
passes_pct : 76.3 passes_total_distacnde : 595 passes_about_attack : 2  dribbles_completed_pct : 30 touches : 84 pass_received :  72
Hwang In-beom Korea Republic 1996 Olympiacos MF
Stats in worldcup
passes_pct : 78.5 passes_total_distacnde : 3456 passes_about_attack : 60  dribbles_completed_pct : 0 touches : 276 pass_received :  206
Jeong Woo-yeong Korea Republic 1999 Freiburg MF
Stats in worldcup
passes_pct : 80 passes_total_distacnde : 181 passes_about_attack : 2  dribbles_completed_pct : 0 touches : 20 pass_received :  18
Jung Woo-young Korea Republic 1989 Al Sadd SC MF
Stats in worldcup
passes_pct : 84.4 passes_total_distacnde : 3599 passes_about_attack : 41  dribbles_completed_pct : 0 touches : 239 pass_received :  169
Lee Jae-sung Korea Republic 1992 Mainz 05 MF
Stats in worldcup
passes_pct : 86.6 passes_total_distacnde : 1285 passes_about_attack : 4  dribbles_completed_pct : 66.7 touches : 116 pass_received :  92
Lee Kangin Korea Republic 2001 Mallorca MF
Stats in worldcup
passes_pct : 80.5 passes_total_distacnde : 1225 passes_about_attack : 24  dribbles_completed_pct : 50 touches : 91 pass_received :  64
Paik Seung-ho Korea Republic 1997 Jeonbuk MF
Stats in worldcup
passes_pct : 74.1 passes_total_distacnde : 260 passes_about_attack : 6  dribbles_completed_pct : 0 touches : 30 pass_received :  19
Son Jun-ho Korea Republic 1992 Shandong Lunen MF
Stats in worldcup
passes_pct : 93.2 passes_total_distacnde : 1190 passes_about_attack : 15  dribbles_completed_pct : 0 touches : 68 pass_received :  48
Hong Chul Korea Republic 1990 Daegu DF
Stats in worldcup
aerials_won_pct : 100 ball_recoveries : 2 blocks : 0 clearances : 1 interceptions : 2 tackle_won : 2
Kim Jin-su Korea Republic 1992 Jeonbuk DF
Stats in worldcup
aerials_won_pct : 71.4 ball_recoveries : 12 blocks : 1 clearances : 5 interceptions : 16 tackle_won : 2
```

➔ 한국 선수단 정보 출력한 결과(일부 생략)



```
print argentina

Julián Álvarez Argentina 2000 Manchester City FW
Stats in worldcup
shoots_per90 : 2.49 Goals : 4 shoots_on_traget_per90 : 1.74
Lautaro Martínez Argentina 1997 Inter FW
Stats in worldcup
shoots_per90 : 4.04 Goals : 0 shoots_on_traget_per90 : 1.21
Lionel Messi Argentina 1987 Paris S-G FW
Stats in worldcup
shoots_per90 : 3.63 Goals : 5 shoots_on_traget_per90 : 1.58
Paulo Dybala Argentina 1993 Roma FW
Stats in worldcup
shoots_per90 : 0 Goals : 0 shoots_on_traget_per90 : 0
Alexis Mac Allister Argentina 1998 Brighton MF
Stats in worldcup
passes_pct : 89.6 passes_total_distacnde : 2731 passes_about_attack : 29  dribbles_completed_pct : 33.3 touches : 263 pass_received :  219
Enzo Fernández Argentina 2001 Benfica MF
Stats in worldcup
passes_pct : 88.2 passes_total_distacnde : 5490 passes_about_attack : 65  dribbles_completed_pct : 50 touches : 419 pass_received :  337
Exequiel Palacios Argentina 1998 Leverkusen MF
Stats in worldcup
passes_pct : 89.3 passes_total_distacnde : 251 passes_about_attack : 9  dribbles_completed_pct : 0 touches : 34 pass_received :  23
Guido Rodríguez Argentina 1994 Betis MF
Stats in worldcup
passes_pct : 84.4 passes_total_distacnde : 871 passes_about_attack : 3  dribbles_completed_pct : 0 touches : 69 pass_received :  54
Leandro Paredes Argentina 1994 Juventus MF
Stats in worldcup
passes_pct : 93 passes_total_distacnde : 3414 passes_about_attack : 25  dribbles_completed_pct : 0 touches : 252 pass_received :  197
Papu Gómez Argentina 1988 Sevilla MF
Stats in worldcup
passes_pct : 88.3 passes_total_distacnde : 671 passes_about_attack : 4  dribbles_completed_pct : 50 touches : 69 pass_received :  63
Rodrigo De Paul Argentina 1994 Atlético Madrid MF
Stats in worldcup
passes_pct : 83.3 passes_total_distacnde : 6099 passes_about_attack : 57  dribbles_completed_pct : 16.7 touches : 543 pass_received :  426
Thiago Almada Argentina 2001 Atlanta Utd MF
Stats in worldcup
passes_pct : 100 passes_total_distacnde : 207 passes_about_attack : 2  dribbles_completed_pct : 0 touches : 15 pass_received :  17
Ángel Correa Argentina 1995 Atlético Madrid MF
Stats in worldcup
passes_pct : 100 passes_total_distacnde : 32 passes_about_attack : 0  dribbles_completed_pct : 0 touches : 6 pass_received :  3
Ángel Di María Argentina 1988 Juventus MF
Stats in worldcup
passes_pct : 75 passes_total_distacnde : 1581 passes_about_attack : 29  dribbles_completed_pct : 47.4 touches : 158 pass_received :  152
Cristian Romero Argentina 1998 Tottenham DF
Stats in worldcup
aerials_won_pct : 62.5 ball_recoveries : 23 blocks : 2 clearances : 16 interceptions : 9 tackle_won : 5
Germán Pezzella Argentina 1991 Betis DF
Stats in worldcup
aerials_won_pct : 0 ball_recoveries : 1 blocks : 0 clearances : 0 interceptions : 0 tackle_won : 0
Gonzalo Montiel Argentina 1997 Sevilla DF
Stats in worldcup
aerials_won_pct : 50 ball_recoveries : 2 blocks : 0 clearances : 1 interceptions : 4 tackle_won : 1
```

➔ 아르헨티나 대표팀 선수 정보 출력한 결과(일부 생략)

```
test player objectCho Guesung Korea Republic 1998 Jeonbuk FW
Stats in worldcup
shoots_per90 : 2.95 Goals : 2 shoots_on_traget_per90 : 1.64
Hwang In-beom Korea Republic 1996 Olympiacos MF
Stats in worldcup
passes_pct : 78.5 passes_total_distacnde : 3456 passes_about_attack : 60  dribbles_completed_pct : 0 touches : 276 pass_received :  206
Emiliano Martínez Argentina 1992 Aston Villa GK
Stats in worldcup
clean_sheet_pct : 50 save_pct : 37.5 gk_pass_pct : 23
Argentinateam destructor called
Korea Republicteam destructor called
```

➔ 객체배열을 받아오고, index를 바탕으로 선수 객체에 접근하여 선수 객체의 메서드를 이용하여 출력한 결과. 또한 소멸자가 실행된 것을 프린트되어서 확인할 수 있는 실행 결과

5.논의

-UI(user interface)에 관한 논의

현재 프로그램 내에서 구현된 내용은 main함수 내에 월드컵에 출전한 모든 선수의 데이터를 적재해 놓고, 이 데이터(선수 객체 벡터)를 바탕으로 nationteam 객체를 만들어서 nationteam객체 의 정보가 정확히 저장되어 있는지, 또 각각 nationteam 객체 안의 배열의 선수 객체의 정보가 정확하게 담겨 있는지 확인하는 것이다. 즉 만들어낸 class들의 생성자 및 메서드가 잘 작동하는지 확인하

는 것에 중점을 두었다.

따라서 이 프로그램을 더 발전시킬 수 있을 것이라 생각했다. 현재는 interface부분을 구현해 놓지 않아서 사용자 인풋을 받을 수 없는 것이 한계이다. 사용자 input을 받아서 원하는 선수의 정보를 받을 수 있도록 하거나, 원하는 국가대표팀의 정보를 받을 수 있다면 좋을 것이다. 즉 인터페이스의 부재가 프로그램에 존재하기 때문에 객체지향 프로그래밍의 특징 중 하나인 추상화, 즉 사용자가 인터페이스만 보고 내부 구현을 알 필요가 없는 추상화의 아이디어가 이 부분에서 부족하다고 생각한다.

-프로그램의 성능

프로그램의 성능 부분에 대해서 생각했는데, 처음 메인함수에서 반복문을 돌려서 월드컵에 출전한 모든 선수를 각 포지션 벡터에 담을 수 있도록 하였는데, 이 각각 생성자가 불릴 때 마다 2개의 파일을 반복적으로 접근하여서 처음 모든 선수의 정보를 담을 때 굉장히 오래걸린다. 따라서 이 부분을 프로그램 내에서 개선해야한다. 따라서 이는 선수 class를 상속한 모든 포지션 별 class에서 이런 로직을 바탕으로 프로그램이 돌아가기 때문에, csv파일을 개선하거나, 포지션 별로 다시 묶어서 받아올 수 있으면 비교연산 횟수 및 반복 횟수가 줄어들어 프로그램의 성능을 개선할 수 있을 것 같다.

-Team class -> 벡터 vs 동적할당 배열

Team class를 구현할 때, 동적할당한 배열을 사용할지, 벡터를 사용할지에 대한 고민을 하였다. 동적할당을 하게 되면 메모리를 차지하는 공간은 적겠지만, 할당을 하기 전에 각 배열에 필요한 크기가 얼만지 모르기 때문에 main의 선수 벡터에서 나라별 선수의 명수를 세온다음에 그 크기를 할당했다. 반면 벡터를 쓰게 된다면 이런 동적할당 과정 없이, 즉 메인 함수의 선수 벡터에서 선수를 세는 과정이 필요없이 그냥 벡터에 선수 정보를 넣을 수 있다. 따라서 공간 복잡도와 시간 복잡도의 교환이 발생한다고 생각했다. 이는 더 고민해보아야 할 문제인 것 같다.

-Team class -> Team class의 확장 가능성

Team class를 구현할 때, 국가별 정보만 담을 수 있게 class를 구현하였는데, 피파23과 같은 컴퓨터 게임에서는 원하는 선수를 원하는 스탯, 포지션별로 필터링하여 자신 만의 팀을 구현한다. 따라

서 Team class를 부모 클래스로 하고, 국가별 정보를 담는 nationTeam class를 상속하도록 하고, 또한 부모 클래스가 된 Team class를 상속하여 Myteam class 즉, 사용자가 원하는 선수를 찾아서 팀을 꾸릴 수 있는 class를 만들 수 있을 것 같다고 생각했다. 이는 구현을 하게 되면 프로그램을 더 의미 있게 만들 수 있을 것 같다.

-2026년 월드컵

2026년 월드컵 또한 선수 정보를 비슷하거나 동일한 데이터셋으로 제공될 수도 있다. 객체를 바탕으로 2022년 월드컵에서 활약한 선수들의 정보를 담았기 때문에 2026년에도 class를 유지한 채 dataset만 변경해주면 동일하게 프로그램을 사용할 수 있을 것 같다. 이는 절차지향 프로그래밍과는 다르게 객체지향 프로그래밍이라서 가능할 것이다.

6.결론

객체 지향 프로그래밍을 바탕으로 Kaggle의 2022년 카타르 월드컵의 선수와 팀 클래스를 만들었다. 그 후 이 클래스를 바탕으로 메인함수를 실행하였다.

이 프로그램을 만들 때, 가장 고려한 점은 객체 지향의 특징들인 **추상화, 다형성, 캡슐화, 상속**이다.

선수 클래스는 상속을 바탕으로 이루어진다. 또한 추상화를 부모 클래스인 Player은 abstract class를 바탕으로 한 추상화, 그리고 선수의 모든 정보를 담는 것이 아닌 필요한 정보를 축소하는 과정에서의 추상화를 구현하였다.

다형성은 virtual function을 통해 상속될 때 자연스럽게 다형성을 형성할 수 있도록 구현하였다. 이는 Player클래스가 상속되었을 때 선수의 포지션 별로 객관적으로 평가하는 지표가 다 다르기 때문에 이런 정보를 받아오는 함수를 상속된 자식 클래스에서 재정의해야하고, 객체별로 프린트하는 방법도 다르기 때문에 이를 오버라이딩으로 재정의하여 다형성을 구현하였다.

팀 클래스도 마찬가지로 위의 선수들의 객체를 소유하여 선수단에 대한 데이터와 이에 따른 메서드(동작)을 구현하였다. 이 때 나라별 선수 명수가 다를 수 있기 때문에 이는 동적으로 메모리를 할당하여 메모리 관리를 해 주었다.

이 두가지 클래스에서 공통적으로 캡슐화를 통해 정보은닉을 하였다.

프로그램을 이렇게 객체 지향의 4가지 특징을 확인할 수 있도록 구성하였다.

이렇게 구현한 두 개의 클래스를 바탕으로 main()함수에서 벡터에 월드컵에서 뛴 선수들을 객체를 바탕으로 적재하였다. 이 적재된 벡터를 바탕으로 한국 대표팀 객체, 아르헨티나 대표팀 객체를 만들어서 위에서 구현한 두 개의 클래스의 메서드가 잘 작동하는지 확인하였다.