Lab5. Programació Orientada a Objectes

Jhonatan Barcos Gambaro - P201 G09

Índice

1. Introducción

Este proyecto tiene como objetivo implementar un sistema de gestión de equipos y partidos de fútbol en C++. Se han definido varias clases clave para representar jugadores, equipos y partidos, incorporando conceptos fundamentales de la programación orientada a objetos, como la herencia:

- ## Country

Atributos

- name (string): Almacena el nombre del país.

Métodos

- Country(std::string n): Constructor que recibe el nombre del país.
- std::string getName(): Método que devuelve el nombre del país.
-

- ## Player

Atributos

- _isFemale (bool): Indica si el jugador es femenino.
- name (string): Almacena el nombre del jugador.
- age (int): Almacena la edad del jugador.
- nacionality (Country*): Puntero al país de origen del jugador.
- noMatches (int): Número de partidos jugados.

Métodos

- Player(bool i, std::string n, int a, Country* c): Constructor que inicializa los atributos básicos.
- bool isFemale() const: Método que devuelve si el jugador es femenino.
- std::string getName(): Método que devuelve el nombre del jugador.
- int getAge(): Método que devuelve la edad del jugador.

- Country* getNacionality(): Método que devuelve el país de origen del jugador.
- int getNoMatches(): Método que devuelve el número de partidos jugados.
- virtual void updateStats(Match* m) = 0: Método virtual puro para actualizar las estadísticas del jugador.
- virtual void printStats() = 0: Método virtual puro para imprimir las estadísticas del jugador.
-

• Goalkeeper

Atributos

- noSaves (int): Número de salvadas realizadas por el portero.
- noGoalsLet (int): Número de goles permitidos por el portero.

Métodos

- void updateStats(Match* m): Implementación específica para actualizar las estadísticas del portero.
- void printStats(): Implementación específica para imprimir las estadísticas del portero.

Herencia

- Deriva de la clase Player
-

• Outfielder.

Atributos

- noTrackles (int): Número de tackles realizados por el jugador de campo.
- noPasses (int): Número de pases realizados por el jugador de campo.
- noShots (int): Número de tiros realizados por el jugador de campo.
- noAssists (int): Número de asistencias realizadas por el jugador de campo.
- noGoals (int): Número de goles anotados por el jugador de campo.

Métodos

- void updateStats(Match* m): Implementación específica para actualizar las estadísticas del jugador de campo.
- void printStats(): Implementación específica para imprimir las estadísticas del jugador de campo.

Herencia

- Deriva de la clase Player.

- **Team**

  Atributos

  - name (string): Almacena el nombre del equipo.
  - country (Country*): Puntero al país al que pertenece el equipo.
  - gender (Gender): Enumeración que representa el género del equipo (MALE, FEMALE, MIXED).
  - players (list<Player*>)`: Lista de jugadores pertenecientes al equipo.

  Métodos

  - Team(std::string n, Country* c, Gender g): Constructor que inicializa los atributos básicos.
  - std::string getName(): Método que devuelve el nombre del equipo.
  - Country* getCountry(): Método que devuelve el país al que pertenece el equipo.
  - Gender getGender(): Método que devuelve el género del equipo.
  - std::list<Player*>& getPlayers(): Método que devuelve la lista de jugadores del equipo.
  - void addPlayer(Player* p): Método que agrega un jugador al equipo según las restricciones de género.

- **Match**

  Atributos

  - teamOne (Team*): Puntero al primer equipo en el partido.
  - teamTwo (Team*): Puntero al segundo equipo en el partido.
  - goalOne (int): Número de goles del primer equipo.
  - goalTwo (int): Número de goles del segundo equipo.
  - scorersOne (list<Player*>)`: Lista de jugadores que marcaron goles para el primer equipo.
  - scorersTwo (list<Player*>)`: Lista de jugadores que marcaron goles para el segundo equipo.

  Métodos

  - Match(Team* o, Team* t): Constructor que recibe los dos equipos participantes.
  - Team* getTeamOne(): Método que devuelve el primer equipo.

- Team* getTeamTwo(): Método que devuelve el segundo equipo.
- int getGoalOne(): Método que devuelve el número de goles del primer equipo.
- int getGoalTwo(): Método que devuelve el número de goles del segundo equipo.
- std::list<Player*>& getScorersOne(): Método que devuelve la lista de goleadores del primer equipo.
- std::list<Player*>& getScorersTwo(): Método que devuelve la lista de goleadores del segundo equipo.
- void simulateMatch(): Método que simula un partido, generando resultados aleatorios y estadísticas de goleadores.
- void printMatch(): Método que imprime el resultado del partido.

2. Descripción soluciones alternativas

Durante la implementación, se discutió la posibilidad de utilizar una clase separada para el manejo de estadísticas y resultados de partidos. Sin embargo, se eligió incorporar estas funciones directamente en la clase Match para simplificar la estructura y mejorar la cohesión.

En cuanto a la representación de los jugadores, se consideró la opción de tener clases separadas para diferentes posiciones en el campo (defensa, mediocampo, delantero, etc.). Sin embargo, se optó por tener una clase base Player y derivar clases específicas (Goalkeeper y Outfielder) para representar las diferentes posiciones. Esta elección facilita la extensión del sistema a posiciones adicionales en el futuro.

La herencia se aplicó eficazmente en las clases Goalkeeper y Outfielder, que derivan de la clase base Player. Esto permite compartir atributos y métodos comunes mientras se extiende la funcionalidad para satisfacer las necesidades específicas de cada tipo de jugador.

Durante la implementación, se superaron desafíos menores relacionados con la gestión de punteros y la simulación aleatoria de estadísticas. Sin embargo, estos problemas se resolvieron de manera eficiente.

3. Conclusión

La implementación demuestra una representación coherente y estructurada de los elementos del fútbol mediante el uso efectivo de la herencia y la encapsulación. Las pruebas realizadas en el main() indican que las clases funcionan correctamente, generando resultados simulados de partidos y estadísticas de jugadores.

En cuanto a nuestro main(). Veamos paso a paso como hemos simulado este programa a partir de los requisitos de la práctica:

1. **Creación de Países:**

Se crean dos países, "Portugal" y "Spain", utilizando la clase Country.

```
// Create countries
Country * c1 = new Country("Portugal");
Country * c2 = new Country("Spain");
```

2. **Creación de Jugadores de Campo:**

Se crean cuatro jugadores de campo, dos de cada país, utilizando la clase Outfielder.

```
// Create outfielders from different countries
Player * o1 = new Outfielder(false, "Cristiano Ronaldo", 35, c1);
Player * o2 = new Outfielder(false, "Bernardo Silva", 26, c1);
Player * o3 = new Outfielder(false, "Sergio Ramos", 34, c2);
Player * o4 = new Outfielder(false, "Sergio Busquets", 32, c2);
```

3. **Creación de Equipos:**

Se crean dos equipos, "Portugal Male Team" y "Spain Male Team", utilizando la clase Team.
Se añaden jugadores de campo a cada equipo mediante el método addPlayer().

```
//Create two teams with the outfielders
Team * team1 = new Team("Portugal Male Team", c1, Team::Gender::MALE);
team1->addPlayer(o1);
team1->addPlayer(o2);

Team * team2 = new Team("Spain Male Team", c2, Team::Gender::MALE);
team2->addPlayer(o3);
team2->addPlayer(o4);
```

4. **Creación de un Partido:**

Se crea un objeto de la clase Match representando un partido entre los dos equipos.

```
//Create a match between two teams
Match * m = new Match(team1, team2);
```

5. **Simulación y Visualización del Partido:**

Se simula el partido mediante el método simulateMatch() de la clase Match.
Se imprime el resultado del partido mediante el método printMatch().

```
//Simulate match
m->simulateMatch();

//Print match stats
m->printMatch();
```

```
Simulating match...

Printing match...
Portugal Male Team-Spain Male Team: 5-5
```

6. **Actualización y Visualización de Estadísticas de Jugadores:**

Se actualizan las estadísticas de cada jugador con el método updateStats() después de jugar el partido. Se imprimen las estadísticas actualizadas de cada jugador mediante el método printStats().

```
//Update stats of each player
o1->updateStats(m);
o2->updateStats(m);
o3->updateStats(m);
o4->updateStats(m);

//Print stats of each player
o1->printStats();
o2->printStats();
o3->printStats();
o4->printStats();
```

```
Updating stats...

Printing stats...

* Printing stats of Cristiano Ronaldo
Trackles: 1
Passes: 7
Shots: 1
Assists: 1
Goals: 5

* Printing stats of Bernardo Silva
Trackles: 2
Passes: 7
Shots: 6
Assists: 1
Goals: 4
```

En resumen, la solución implementada cumple con los requisitos del problema y proporciona una base sólida para futuras extensiones o mejoras en el sistema de gestión de equipos y partidos de fútbol en C++.