Laboratorio 4

El objetivo del laboratorio es:

En esta sesión de laboratorio el objetivo es implementar un mecanismo para clasificar la tabla de ligas y la listas de goleadores.

Entrega:

- Código del proyecto (Programas en Java)
- Documentación de la implementación.

Universitat Pompeu Fabra Barcelona

Documentación

1. Una **introducción** donde se describe el problema.
   a. ¿Qué hace el programa?
   b. ¿Qué clases se definieron?
   c. ¿Cuáles métodos se implementaron?
2. Una **descripción** de **soluciones alternativas** que se discutieron, y una descripción de la **solución elegida**, así como el motivo de la elección de esta solución en lugar de otras. También es una buena idea mencionar los **aspectos teóricos** de los conceptos de programación orientada a objetos que se aplicaron como parte de la solución.
3. Una **conclusión** que describa el **funcionamiento** de la práctica, es decir, ¿las pruebas mostraron que las clases se implementaron correctamente? Se puede mencionar cualquier **dificultad** durante la implementación, así como la solución.

El código fuente y la documentación deben cargarse en un directorio de nombre **Lab4** de su repositorio Git ***antes de la próxima sesión de laboratorio***. ***Importante, el nombre del directorio deberá ser exactamente Lab4.***

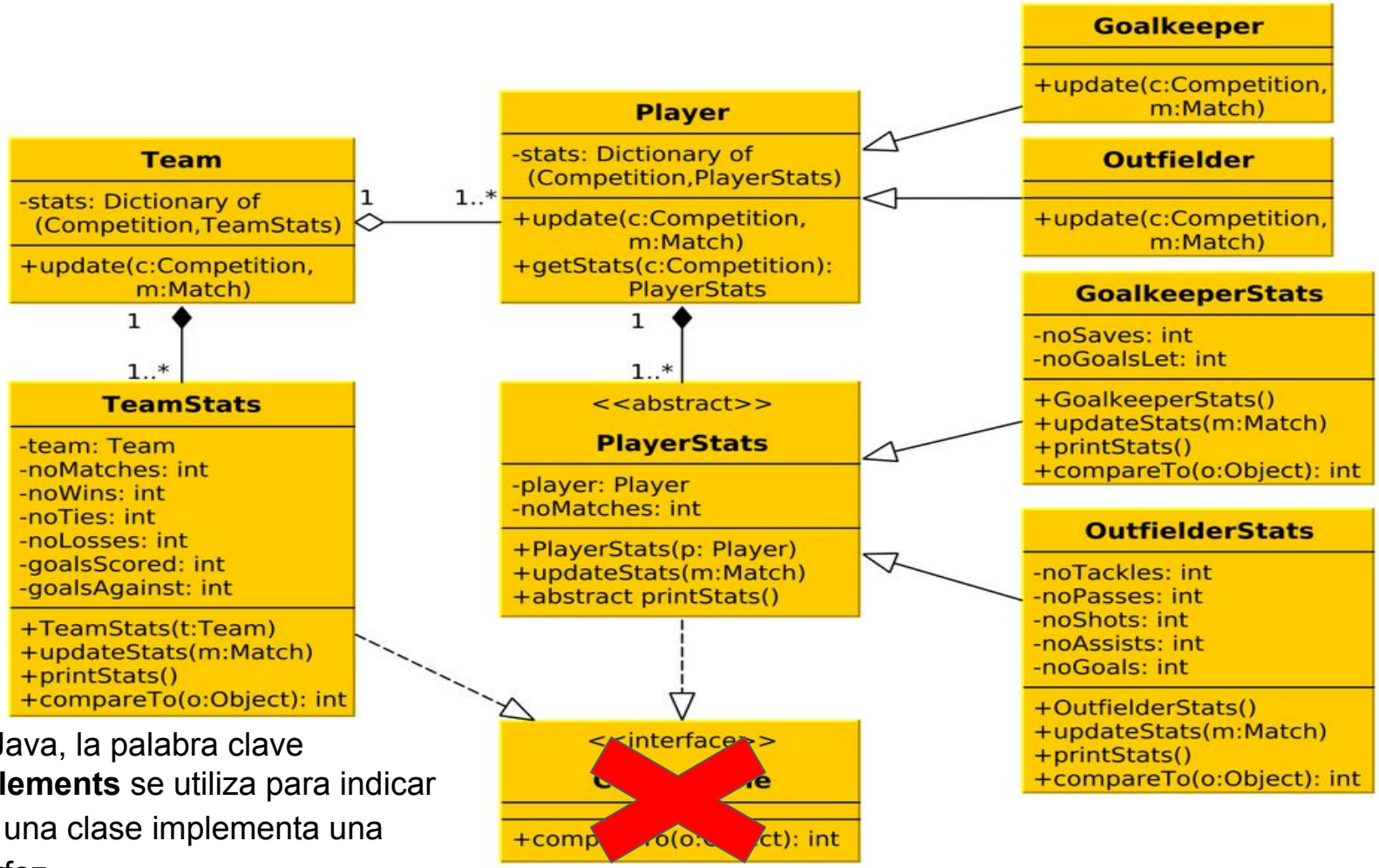
Universitat Pompeu Fabra Barcelona

Estadísticas de equipos y jugadores

- Primero hay que implementar las clases relacionadas con las estadísticas del equipo y las estadísticas de los jugadores.
- **Mover** algunos **atributos** de Team a TeamStats, de GoalKeeper a GoalkeeperStats, y de Outfielder a OutfielderStats.
- También mover los métodos **updateStats** and **printStats**.

Universitat Pompeu Fabra Barcelona

Diagrama propuesto



En Java, la palabra clave **implements** se utiliza para indicar que una clase implementa una interfaz.

Ordenar tablas de clasificación

En una liga determinada, los criterios para determinar el orden de clasificación de los equipos es el siguiente:

a. **Más puntos**, definidos como **3 ∗ noWins + noTies**.

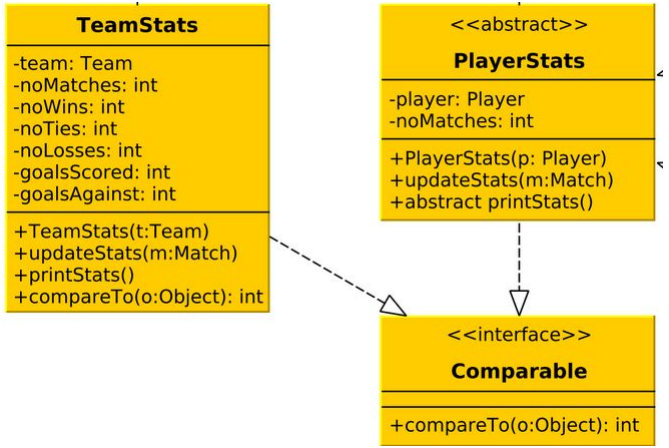
b. Mejor **diferencia de goles**, definida como goles a favor – goles en contra. (goalsFor − goalsAgainst)

c. Más **goles marcados**, es decir, goles a favor. (goalsFor)

Si dos **equipos** son **iguales** según los tres criterios, se pueden **ordenar arbitrariamente.**

Ordenar tablas de clasificación

```
public class TeamStats implements Comparable<TeamStats> {
```



En **TeamStats**, la idea es anular el método **compareTo** para poder ordenar correctamente los equipos.

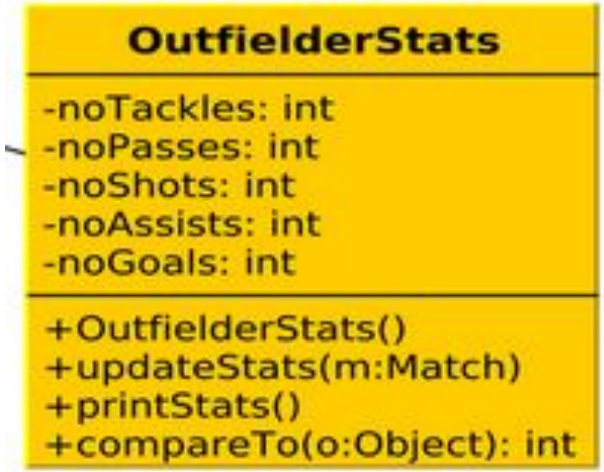
```
public int compareTo(TeamStats ts) {
```

CompareTo debería devolver:

- −1 si el valor actual de la instancia de TeamStats debe ordenarse antes que otra,
- 0 si son iguales.
- 1 si la instancia actual debe ordenarse después de otra.

Clasificación de goleadores

```
public class OutfielderStats extends PlayerStats {
```



Para determinar si un outfielder ha marcado más goles que otro hay que **comparar** el valor del atributo **noGoals** en la clase OutfielderStats. Para ello hay que sobreescribir el método compareTo en OutfielderStats.

```
public int compareTo(PlayerStats ps) {
    if (ps instanceof OutfielderStats) {
        OutfielderStats os = (OutfielderStats)ps;
```

downcast

CompareTo debería devolver:

- **−1** si el valor actual de la instancia de TeamStats debe ordenarse antes que otra,
- **0** si son iguales.
- **1** si la instancia actual debe ordenarse después de otra.

Implementando un diccionario

El siguiente paso es modificar las clases Equipo y Jugador para incorporar un diccionario desde competiciones hasta estadísticas.

- Se puede implementar un diccionario en Java usando la clase existente **HashMap**, que debe importarse en la clase.
- Observar los métodos definidos en HashMap, especialmente **get y put.**
- El diccionario debe crearse en el constructor de Team y Player.

https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

Implementando un diccionario

- Importar HashMap.

```
import java.util.HashMap;
```

- Crear atributo

```
protected HashMap<Competition, TeamStats> stats;
```

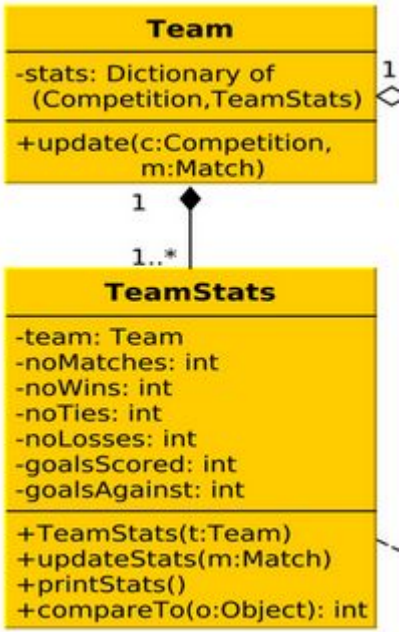
- Crear diccionario en el constructor.

```
stats = new HashMap<Competition, TeamStats>();
```

Update de la clase Team

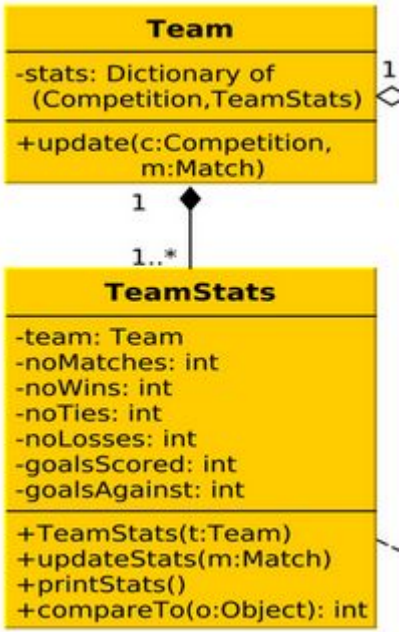
Actualiza las estadísticas del equipo cuando juega un partido en una competición determinada.

- Esto se hace mirando primero en el diccionario para ver si el equipo tiene alguna estadística asociada con la competición.
- De lo contrario, debe crear una nueva instancia de TeamStats y agregarla al diccionario para la competencia actual.



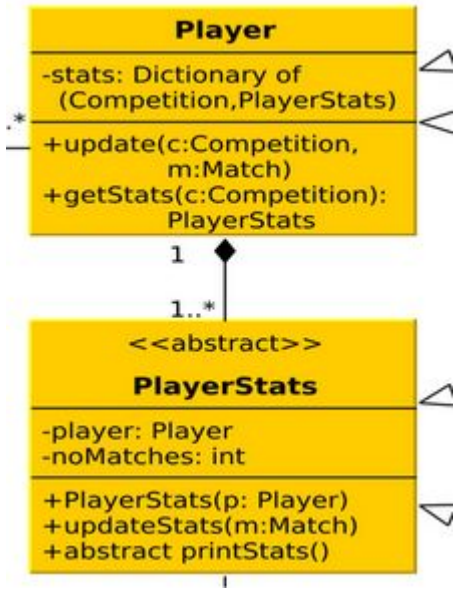
Update de la clase Team

- Después se llama al método updateStats de TeamStats con el partido como parámetro.
- La actualización debera también llamar al método de update de Player para todos los jugadores del equipo.



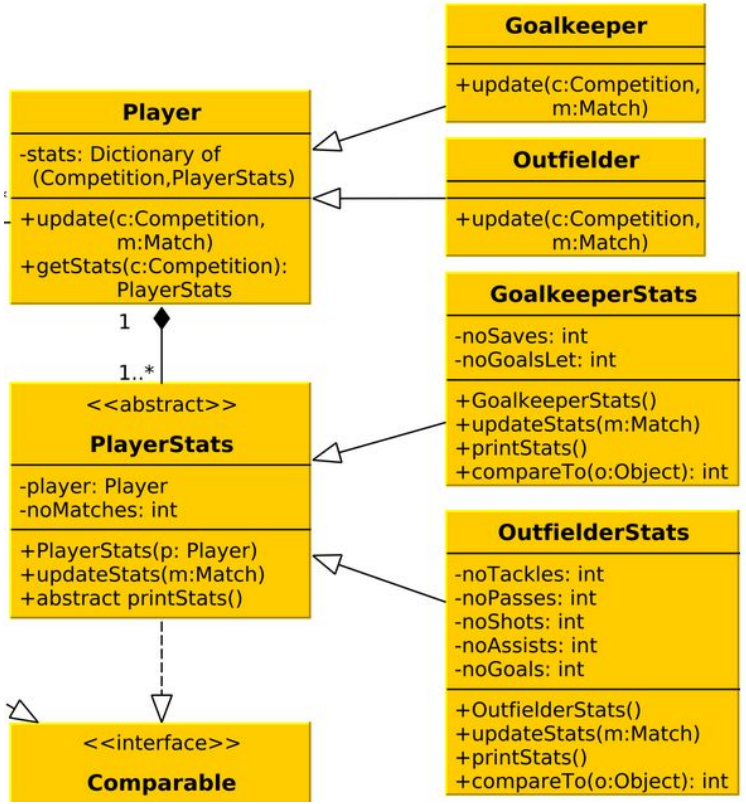
Update de la clase Player

- Asimismo, el método de **update** del jugador deberá actualizar las estadísticas del jugador como resultado de jugar un partido en una competición determinada.
- En este método, primero se busca en el diccionario para ver si el jugador tiene alguna estadística asociada con la competencia. Si no, se deberá crear una nueva instancia de PlayerStats y añadirlo al diccionario de la competencia actual.
- Tener en cuenta que las clases Goalkeeper y Outfielder sobreescriben el método update.
- La razón es que el método puede crear una instancia de GoalkeeperStats si el jugador es un Goalkeeper y una instancia de OutfielderStats si el jugador es Outfielder.



Update de la clase Player

- Una vez que las estadísticas de los jugadores de una competición existen en el diccionario, se puede llamar al método **updateStats** de **PlayerStats** con el partido que se desea actualizar.
- Como antes, la implementación de updateStats es diferente para los GoalKeeper (en GoalkeeperStats) y Outfielders(en OutfielderStats).



Impresión de tablas de clasificación y goleadores

- El último paso de la sesión de laboratorio es imprimir las tablas de clasificación y las listas de goleadores.
- En **League**, implementar el método **printTable** creando una lista de **TeamStats** y llenarlo con las estadísticas de todos los equipos en la competencia actual (llamando al método getStats del Equipo).

```
LinkedList<TeamStats> stats = new LinkedList<TeamStats>();
```

- Después, se debe ordenar la lista de TeamStats utilizando el método **Collections.sort**.

```
import java.util.Collections;
```

```
Collections.sort(stats);
```

- Para imprimir la tabla en un formato bonito, se puede llamar al método **printStats** de **TeamStats**.

Impresión de tablas de clasificación y goleadores

- Finalmente, se ha de implementar el método **printGoalScorers** de la clase **Competition**.
- Iplementar un método **getOutfielderStats**, que devuelve una lista de **OutfielderStats** para todos los jugadores outfielder de todos los equipos en la competencia.
- Una vez que se tenga la **lista**, simplemente hay que **ordenarla** usando Collections.sort e **imprimir los primeros k jugadores** (usando el método printStats de OutfielderStats).