

Programação Orientada por Objetos

Revisões

Prof. José Cordeiro,

Prof. Cédric Grueau,

Prof. Laercio Junior

Departamento de Sistemas e Informática

Escola Superior de Tecnologia de Setúbal – Instituto Politécnico de Setúbal

2019/2020

- Revisões
 - Exemplo Xadrez
 - Exemplo Premier League

Exemplo — Xadrez

□ Requisitos do protótipo:

- Representar os componentes do jogo sem implementar as regras ou o desenrolar do jogo.
- Representar as peças: peão, torre, cavalo, rei, rainha e bispo.
- Representar o tabuleiro de jogo com as posições.
- Deve ser possível obter em texto a posição de cada peça usando a notação algébrica (ex: e5 – peão na casa e5, ou Te7 – torre na casa e7).



Exemplo — Xadrez

□ Representação:

- Cada peça poderá ser representada por uma classe
- Peças: peão (pawn), torre (rook), cavalo (knight), rei (king), rainha (queen) e bispo (bishop).
- O tabuleiro de jogo corresponde a outra classe.



Exemplo — Xadrez

- Representação do peão – Classe **Pawn**



- Atributos:

- **Colour** (branco ou preto)
- **Position** (coordenadas x e y)

Usar o tipo enumerado

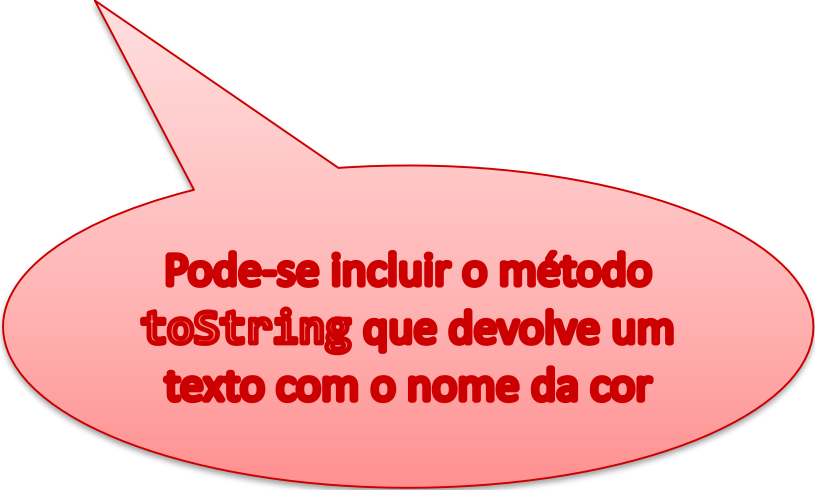
- Métodos

- Seletores e modificadores
- **toString** – para devolver a peça e a posição utilizando a notação algébrica

Exemplo — Xadrez

□ Tipo enumerado **Colour**

```
public enum Colour {  
    WHITE, BLACK  
}
```



**Pode-se incluir o método
toString que devolve um
texto com o nome da cor**

Exemplo — Xadrez

□ Tipo enumerado **Colour**

```
public enum Colour {  
  
    WHITE, BLACK;  
  
    @Override  
    public String toString() {  
        if (this == WHITE) {  
            return "Branco";  
        } else if (this == BLACK) {  
            return "Preto";  
        }  
        return "";  
    }  
}
```

Exemplo — Xadrez

□ Classe **Pawn**

```
public class Pawn {  
  
    private Colour colour;  
    private char x;  
    private int y;  
  
    // restante código omitido  
}
```

A posição das peças irá ser usada em vários locais: pode ser definida numa classe separada, utilizando-se depois a composição de classes.

Exemplo — Xadrez

□ Classe **Position**

```
public class Position {  
    private char x;  
    private int y;
```

Coordenadas de uma posição
num tabuleiro de xadrez

```
    public Position(char x, int y) {  
        this.x = x;  
        this.y = y;  
    }
```

Construtor com os argumentos **x** e **y**

```
    public Position() {  
        this.x = 'a';  
        this.y = 1;  
    }
```

Construtor sem argumentos

```
    // restante código omitido
```

```
}
```

Exemplo — Xadrez

- Classe **Position** – métodos **seletores** e **modificadores**

```
public char getX() {  
    return x;  
}
```

```
public int getY() {  
    return y;  
}
```

```
public void setX(char x) {  
    this.x = x;  
}
```

```
public void setY(int y) {  
    this.y = y;  
}
```

Exemplo — Xadrez

- Classe **Position** – método **toString**

```
@Override  
public String toString() {  
    return "" + x + y;  
}
```



Porquê as aspas?

Exemplo — Xadrez

□ Classe **Position** – exemplo de utilização

```
Position pos1 = new Position('e', 7); //posicao x e y  
System.out.println("Posicao: " + pos1);
```

```
Position pos2 = new Position();  
System.out.println("Posicao: " + pos2.getX() + pos2.getY());
```

Posicao: e7
Posicao: a1

Exemplo — Xadrez

□ Classe **Pawn**

```
public class Pawn {  
  
    private Colour colour;  
    private Position position;  
  
    public Pawn(Colour colour, Position position) {  
        this.colour = colour;  
        if (position != null) {  
            this.position = position;  
        } else {  
            this.position = new Position();  
        }  
    }  
  
    // restante código omitido  
}
```

Exemplo — Xadrez

□ Classe **Pawn** – métodos **seletores** e **modificadores** (1/2)

```
public Colour getColour() {  
    return colour;  
}  
  
public Position getPosition() {  
    return new Position(position.getX(), position.getY());  
}  
  
public void setPosition(char x, int y) {  
    position.setX(x);  
    position.setY(y);  
}  
  
public void setPosition(Position position) {  
    this.position.setX(position.getX());  
    this.position.setY(position.getY());  
}
```

Exemplo — Xadrez

- Classe **Pawn** – métodos **seletores** e **modificadores** (2/2)

```
public void setY(int y) {  
    position.setY(y);  
}
```

```
public char getX() {  
    return position.getX();  
}
```

```
public int getY() {  
    return position.getY();  
}
```

Exemplo — Xadrez

- ❑ Classe **Pawn** – métodos **toString** e **getName**

```
@Override
public String toString() {
    return position.toString();
}

public String getName() {
    return "Peão";
}
```


Exemplo — Xadrez

❑ Classe **Pawn** – exemplo de utilização

```
Pawn p1 = new Pawn(Colour.BLACK, new Position('e', 7));  
Pawn p2 = new Pawn(Colour.WHITE, new Position());  
p2.setPosition('d', 2);
```

```
System.out.println("Peão 1 - " + p1);  
System.out.println("Peão 2 - " + p2);
```

```
System.out.print("Peão 1 - ");  
System.out.print(p1.getName() + " na posicao: ");  
System.out.println("" + p1.getX() + p1.getY());
```

```
System.out.print("Peão 2 - ");  
System.out.print(p2.getName() + " na posicao: ");  
System.out.println("" + p2.getX() + p2.getY());
```

```
Peão 1 - e7  
Peão 2 - d2  
Peão 1 - Peão na posicao: e7  
Peão 2 - Peão na posicao: d2
```

Exemplo — Xadrez

□ Classe **Rook**

```
public class Rook {  
  
    private Colour colour;  
    private Position position;  
  
    public Rook(Colour colour, Position position) {  
        this.colour = colour;  
        if (position != null) {  
            this.position = position;  
        } else {  
            this.position = new Position();  
        }  
    }  
  
    public Colour getColour() {  
        return colour;  
    }  
  
    // restante código omitido  
}
```

**Os mesmos atributos
que a classe Pawn e
muito código idêntico
ao da classe Pawn**

Exemplo — Xadrez

□ Classe **Rook**

```
public class Rook {  
  
    // restante código omitido  
  
    public Position getPosition() {  
        return new Position(position.getX(), position.getY());  
    }  
  
    public void setPosition(char x, int y) {  
        position.setX(x);  
        position.setY(y);  
    }  
  
    public void setPosition(Position position) {  
        this.position.setX(position.getX());  
        this.position.setY(position.getY());  
    }  
  
    // restante código omitido  
}
```

Exemplo — Xadrez

□ Classe **Rook**

```
public class Rook {  
  
    // restante código omitido  
    public void setY(int y) {  
        position.setY(y);  
    }  
  
    public char getX() {  
        return position.getX();  
    }  
  
    public int getY() {  
        return position.getY();  
    }  
  
    public String getName() {  
        return "Torre";  
    }  
  
    @Override  
    public String toString() {  
        return "T" + position.toString();  
    }  
}
```

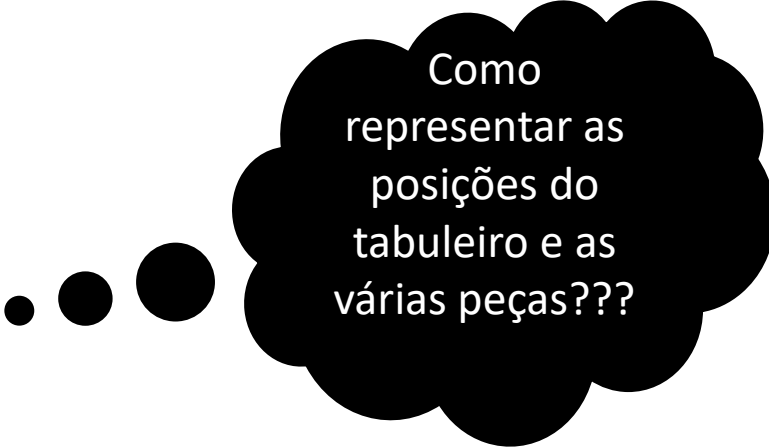
**getName e toString
são os únicos métodos
diferentes em relação à
classe Pawn**

Exemplo — Xadrez

- ❑ Classes **Queen, King, Bishop, Knight**
 - Omitidas: são semelhantes às anteriores
- ❑ Classe **Chessboard**

```
public class Chessboard {
```

```
    // construtor e métodos  
}
```



Como
representar as
posições do
tabuleiro e as
várias peças???

Exemplo — Xadrez

□ Classe **ChessBoard**

```
public class Chessboard {  
  
    ArrayList<Pawn> pawns;  
    ArrayList<Knight> knights;  
    ArrayList<Rook> rooks;  
    ArrayList<Queen> queens;  
    ArrayList<Bishop> bishops;  
    ArrayList<King> kings;  
  
    // construtor e métodos  
}
```

**Podemos usar listas
para guardar as várias
peças, mas...**

**... temos de ter uma
lista por cada tipo de
peça!!!**

E as posições
do tabuleiro?

Exemplo — Xadrez

❑ Classe **ChessBoard**

```
public class Chessboard {  
  
    ArrayList<Pawn> pawns;  
    ArrayList<Knight> knights;  
    ArrayList<Rook> rooks;  
    ArrayList<Queen> queens;  
    ArrayList<Bishop> bishops;  
    ArrayList<King> kings;  
  
    // construtor e métodos  
}
```

- ❑ A representação das posições com o que sabemos do Java utilizando as classes definidas seria complexa.
 - Não podemos ter uma coleção única de posições com peças associadas.
 - Uma representação possível seria utilizar uma coleção **HashMap** para as posições e peças associadas. Não é possível porque temos vários tipos de peças.

Exemplo — Xadrez

□ Classe **ChessBoard**

```
public class Chessboard {  
  
    ArrayList<Pawn> pawns;  
    ArrayList<Knight> knights;  
    ArrayList<Rook> rooks;  
    ArrayList<Queen> queens;  
    ArrayList<Bishop> bishops;  
    ArrayList<King> kings;  
  
    // construtor e métodos  
}
```

- Uma alternativa será não representar as posições e gerir os movimentos das peças para que as suas posições sejam sempre válidas. Neste caso como só pretendemos a representação dos componentes do Xadrez iremos omitir essa implementação.

Exemplo — Xadrez

❑ Classe **ChessBoard**

```
public class Chessboard {  
  
    ArrayList<Pawn> pawns;  
    ArrayList<Knight> knights;  
    ArrayList<Rook> rooks;  
    ArrayList<Queen> queens;  
    ArrayList<Bishop> bishops;  
    ArrayList<King> kings;  
  
    public Chessboard() {  
        pawns = new ArrayList<>();  
        knights = new ArrayList<>();  
        rooks = new ArrayList<>();  
        queens = new ArrayList<>();  
        kings = new ArrayList<>();  
        bishops = new ArrayList<>();  
  
        setup();  
    }  
    // restante código  
}
```

Criação das peças com a sua
colocação nas posições iniciais

Exemplo — Xadrez

❑ Classe ChessBoard – método **setup**

```
private void setup() {  
    for (char x = 'a'; x <= 'h'; x++) {  
        pawns.add(new Pawn(Colour.WHITE, new Position(x, 2)));  
        pawns.add(new Pawn(Colour.BLACK, new Position(x, 7)));  
    }  
    int line = 1;  
    Colour colour = Colour.WHITE;  
    rooks.add(new Rook(colour, new Position('a', line)));  
    knights.add(new Knight(colour, new Position('b', line)));  
    bishops.add(new Bishop(colour, new Position('c', line)));  
    queens.add(new Queen(colour, new Position('d', line)));  
    kings.add(new King(colour, new Position('e', line)));  
    bishops.add(new Bishop(colour, new Position('f', line)));  
    knights.add(new Knight(colour, new Position('g', line)));  
    rooks.add(new Rook(colour, new Position('h', line)));  
  
    line = 8;  
    colour = Colour.BLACK;  
    rooks.add(new Rook(colour, new Position('a', line)));  
    knights.add(new Knight(colour, new Position('b', line)));  
    bishops.add(new Bishop(colour, new Position('c', line)));  
    queens.add(new Queen(colour, new Position('d', line)));  
    kings.add(new King(colour, new Position('e', line)));  
    bishops.add(new Bishop(colour, new Position('f', line)));  
    knights.add(new Knight(colour, new Position('g', line)));  
    rooks.add(new Rook(colour, new Position('h', line)));  
}
```

Exemplo — Xadrez

- Problemas da solução encontrada:
 - Duplicação de código nas classes das peças.
 - A representação do tabuleiro ficou complexa.
 - Outros?

Exemplo — Xadrez (2)

□ Solução alternativa:

- não ter classes para cada tipo de peça
- Ter uma classe única para as peças: a classe **Piece**
 - Neste caso definia-se um atributo **PieceType** dum tipo enumerado que especificava qual a peça que se estava a representar:

```
public class Piece {  
  
    private Colour colour;  
    private Position position;  
    private PieceType pieceType;  
  
    // restante código omitido  
}
```

Exemplo — Xadrez (2)

□ Tipo enumerado **PieceType**

```
public enum PieceType {  
  
    PAWN, ROOK, KNIGHT, BISHOP, KING, QUEEN;  
  
    @Override  
    public String toString() {  
        switch (this) {  
            case PAWN:  
                return "Peão";  
            case ROOK:  
                return "Torre";  
            case KNIGHT:  
                return "Cavalo";  
            case BISHOP:  
                return "Bispo";  
            case QUEEN:  
                return "Rainha";  
            case KING:  
                return "Rei";  
        }  
        return "";  
    }  
}
```

Exemplo — Xadrez (2)

□ Classe **Piece**

```
public class Piece {  
  
    private Colour colour;  
    private Position position;  
    private PieceType pieceType;  
  
    public Piece(PieceType pieceType, Colour colour, Position position) {  
        this.pieceType = pieceType;  
        this.colour = colour;  
        if (position != null) {  
            this.position = position;  
        } else {  
            this.position = new Position();  
        }  
    }  
  
    // restante código omitido  
}
```

O construtor passa a receber o tipo de peça

Exemplo — Xadrez (2)

□ Classe **Piece** – métodos **seletores** e **modificadores** (1/2)

```
public Colour getColour() {  
    return colour;  
}
```

```
public PieceType getPieceType() {  
    return pieceType;  
}
```

```
public void setPieceType(PieceType pieceType) {  
    this.pieceType = pieceType;  
}
```

```
public Position getPosition() {  
    return new Position(position.getX(), position.getY());  
}
```

Exemplo — Xadrez (2)

□ Classe **Piece** – métodos **seletores** e **modificadores** (2/2)

```
public void setPosition(char x, int y) {  
    position.setX(x);  
    position.setY(y);  
}
```

```
public void setPosition(Position position) {  
    this.position.setX(position.getX());  
    this.position.setY(position.getY());  
}
```

```
public void setY(int y) {  
    position.setY(y);  
}
```


```
public char getX() {  
    return position.getX();  
}
```

```
public int getY() {  
    return position.getY();  
}
```


Exemplo — Xadrez (2)

□ Classe **Piece** – método **toString**

```
@Override
public String toString() {
    String text = "";
    switch(pieceType){
        case ROOK:
            text += 'T';
            break;
        case KNIGHT:
            text += 'C';
            break;
        case BISHOP:
            text += 'B';
            break;
        case QUEEN:
            text += 'D';
            break;
        case KING:
            text += 'R';
            break;
    }
    text += position.toString();
    return text;
}
```



Usa um `switch` para tratar separadamente os textos de cada tipo de peça!

Exemplo — Xadrez (2)

- Classe **Piece** – método **getName**

```
public String getName() {  
    return pieceType.toString();  
}
```

Exemplo — Xadrez (2)

□ Classe **Chessboard**

```
public class Chessboard {  
  
    ArrayList<Piece> pieces;  
  
    public Chessboard() {  
        pieces = new ArrayList<>();  
        setup();  
    }  
}
```

**Já é possível colocar
todas as peças numa
única coleção**

Exemplo — Xadrez (2)

□ Classe Chessboard – método **setup**

```
private void setup() {
    for (char x = 'a'; x <= 'h'; x++) {
        pieces.add(new Piece(PieceType.PAWN, Colour.WHITE, new Position(x, 2)));
        pieces.add(new Piece(PieceType.PAWN, Colour.BLACK, new Position(x, 7)));
    }
    int line = 1;
    Colour colour = Colour.WHITE;
    pieces.add(new Piece(PieceType.ROOK, colour, new Position('a', line)));
    pieces.add(new Piece(PieceType.KNIGHT, colour, new Position('b', line)));
    pieces.add(new Piece(PieceType.BISHOP, colour, new Position('c', line)));
    pieces.add(new Piece(PieceType.QUEEN, colour, new Position('d', line)));
    pieces.add(new Piece(PieceType.KING, colour, new Position('e', line)));
    pieces.add(new Piece(PieceType.BISHOP, colour, new Position('f', line)));
    pieces.add(new Piece(PieceType.KNIGHT, colour, new Position('g', line)));
    pieces.add(new Piece(PieceType.ROOK, colour, new Position('h', line)));

    line = 8;
    colour = Colour.BLACK;
    pieces.add(new Piece(PieceType.ROOK, colour, new Position('a', line)));
    pieces.add(new Piece(PieceType.KNIGHT, colour, new Position('b', line)));
    pieces.add(new Piece(PieceType.BISHOP, colour, new Position('c', line)));
    pieces.add(new Piece(PieceType.QUEEN, colour, new Position('d', line)));
    pieces.add(new Piece(PieceType.KING, colour, new Position('e', line)));
    pieces.add(new Piece(PieceType.BISHOP, colour, new Position('f', line)));
    pieces.add(new Piece(PieceType.KNIGHT, colour, new Position('g', line)));
    pieces.add(new Piece(PieceType.ROOK, colour, new Position('h', line)));
}
```

Exemplo — Xadrez (2)

- Problemas da solução encontrada:
 - Classe **Piece** complexa. Tem problemas de coesão.
 - Na classe **Chessboard** será necessário usar vários **switch**, um por cada vez que se quiser escolher entre os vários tipos de peça.
 - Exemplo: na movimentação das peças.
 - Outros?

Exemplo — Premier League

□ Requisitos da aplicação:

- Pretende-se um programa para gerir os clubes e jogadores da primeira liga inglesa.
- Os jogadores são caracterizados pelo seu nome, idade e número da camisola, e os clubes pelo seu nome e cidade.
- A idade mínima dos jogadores é de 16 anos.
- Os clubes devem ser guardados numa coleção.
- Os jogadores devem ser guardados separadamente relacionando-os com o seu clube.



Exemplo — Premier League

□ Representação do jogador - Classe **Player**

```
public class Player {  
    private String name;  
    private int number;  
    private int age;  
  
    private static final int MINIMUM_AGE = 16;  
    public Player(String name, int age) {  
        if (name != null) {  
            this.name = name;  
        } else {  
            this.name = "";  
        }  
        if (age < MINIMUM_AGE) {  
            this.age = MINIMUM_AGE;  
        } else {  
            this.age = age;  
        }  
        number = 0;  
    }  
    // restante código omitido  
}
```

Constante para a
idade mínima

Exemplo — Premier League

□ Classe **Player** – métodos **seletores** e **modificadores** (1/2)

```
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    if (name != null) {  
        this.name = name;  
    }  
}  
  
public int getNumber() {  
    return number;  
}  
  
public void setNumber(int number) {  
    if (number > 0) {  
        this.number = number;  
    }  
}
```


Exemplo — Premier League

- Classe **Player** – métodos **seletores** e **modificadores** (2/2)

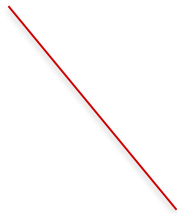
```
public int getAge() {  
    return age;  
}
```

```
public void setAge(int age) {  
    if (age >= MINIMUM_AGE) {  
        this.age = age;  
    }  
}
```

Exemplo — Premier League

□ Representação do clube - Classe **Club**

```
public class Club {  
  
    private String name;  
    private String city;  
  
    public Club(String name, String city) {  
        if (name == null || name.trim().equals("")) {  
            this.name = "";  
        } else {  
            this.name = name;  
        }  
  
        this.city = city;  
    }  
    // restante código omitido  
}
```



Eliminar espaços
em branco

Exemplo — Premier League

□ Classe **Club** – métodos **seletores** e **modificadores**

```
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    if (!(name == null || name.trim().equals("")))) {  
        this.name = name;  
    }  
}  
  
public String getCity() {  
    return city;  
}  
  
public void setCity(String city) {  
    this.city = city;  
}
```

Exemplo — Premier League

□ Representação da liga - Classe **PremierLeague**

```
public class PremierLeague {  
  
    private HashSet<Club> clubs;  
  
    public PremierLeague(HashSet<Club> clubs) {  
        if (clubs != null) {  
            this.clubs = new HashSet<>(clubs);  
        } else {  
            this.clubs = new HashSet();  
        }  
    }  
  
    // restante código omitido  
}
```

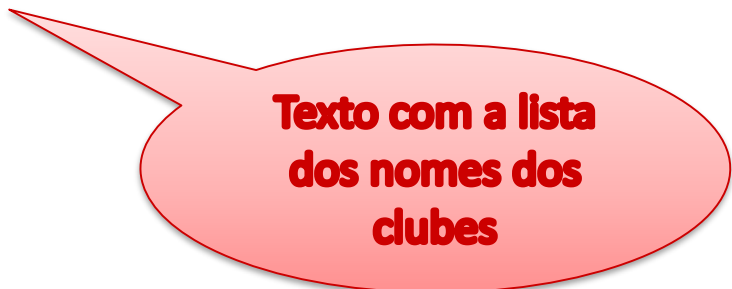
Vamos começar por incluir apenas os clubes guardando-os numa coleção HashSet

No construtor podemos receber uma coleção inicial de clubes.

Exemplo — Premier League

□ Representação da liga - Classe **PremierLeague**

```
public class PremierLeague {  
  
    private HashSet<Club> clubs;  
  
    public String clubsNames() {  
        String info = "Club List:";  
  
        for (Club club : clubs) {  
            info += "\n" + club.getName();  
        }  
  
        return info;  
    }  
  
    // restante código omitido  
}
```



**Texto com a lista
dos nomes dos
clubes**

Exemplo – Premier League

❑ Classe **PremierLeague** – exemplo de utilização

```
HashSet<Club> clubs = new HashSet<>();

clubs.add(new Club("Chelsea", "London"));
clubs.add(new Club("Chelsea", "London"));

clubs.add(new Club("Manchester United", "Manchester"));
clubs.add(new Club("Manchester City", "Manchester"));
clubs.add(new Club("Liverpool", "Liverpool"));

PremierLeague league = new PremierLeague(clubs);

System.out.println(league.clubsNames());
```

Club List:
Manchester United
Liverpool
Chelsea
Chelsea
Manchester City

Desordenados,
repetidos,
porquê?

Exemplo — Premier League

□ Representação do clube - Classe **Club**

```
public class Club {  
    private String name;  
    private String city;  
  
    public int hashCode() {  
        int hash = 7;  
        hash = 53 * hash + Objects.hashCode(this.name);  
        hash = 53 * hash + Objects.hashCode(this.city);  
        return hash;  
    }  
  
    public boolean equals(Object obj) {  
        if (obj == null) {  
            return false;  
        }  
        if (getClass() != obj.getClass()) {  
            return false;  
        }  
        final Club other = (Club) obj;  
        return this.name.equals(other.name) && this.city.equals(other.city);  
    }  
  
    // restante código omitido  
}
```

**Necessário definir
equals e hashCode**

Exemplo – Premier League

❑ Classe **PremierLeague** – exemplo de utilização

```
HashSet<Club> clubs = new HashSet<>();

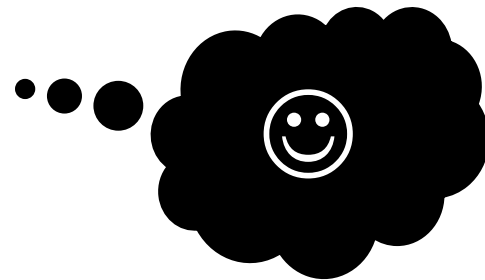
clubs.add(new Club("Chelsea", "London"));
clubs.add(new Club("Chelsea", "London"));

clubs.add(new Club("Manchester United", "Manchester"));
clubs.add(new Club("Manchester City", "Manchester"));
clubs.add(new Club("Liverpool", "Liverpool"));

PremierLeague league = new PremierLeague(clubs);

System.out.println(league.clubsNames());
```

Club List:
Manchester City
Chelsea
Liverpool
Manchester United



Exemplo — Premier League

□ Representação da liga - Classe **PremierLeague**

```
public class PremierLeague {  
  
    private HashSet<Club> clubs;  
    private HashMap<Player, Club> players;  
  
    public PremierLeague(HashSet<Club> clubs) {  
        if (clubs != null) {  
            this.clubs = new HashSet<>(clubs);  
        } else {  
            this.clubs = new HashSet();  
        }  
        players = new HashMap<>();  
    }  
  
    // restante código omitido  
}
```

**Juntamos a coleção
de jogadores
associados a clubes**

Criamos a coleção.

Exemplo — Premier League

- Requisitos da aplicação (2):
 - Adicionar, remover jogadores e obter a lista dos nomes dos jogadores de um clube (como texto).
 - Só é possível adicionar jogadores se se fornecer o clube associado. O clube deve existir na coleção de clubes e o jogador, se existir, não pode estar associado a nenhum clube.



Exemplo — Premier League

- Classe **PremierLeague** – métodos **addPlayer** e **removePlayer**

```
public boolean addPlayer(Player player, Club club) {  
    if (!clubs.contains(club)) {  
        return false;  
    }  
  
    if (players.get(player) != null) {  
        return false;  
    }  
  
    players.put(player, club);  
    return true;  
}  
  
public void removePlayer(Player player) {  
    players.remove(player);  
}
```

Exemplo — Premier League

□ Classe **PremierLeague** – método **clubPlayers**

```
public String clubPlayers(Club club) {  
    if (club == null || !clubs.contains(club)) {  
        return "";  
    }  
    String info = "Players from " + club.getName();  
  
    for (Map.Entry<Player, Club> entry : players.entrySet()) {  
        if (entry.getValue().equals(club)) {  
            info += "\n" + entry.getKey().getName();  
        }  
    }  
  
    return info;  
}
```

Exemplo — Premier League

- Requisitos da aplicação (3):
 - Adicionar jogadores.
 - A informação é o nome do clube e a lista de jogadores a adicionar.
 - Obter os jogadores de um clube.



Exemplo — Premier League

□ Classe **PremierLeague** – método **addPlayers**

```
public void addPlayers(String clubName, ArrayList<Player> players) {  
    for (Club club : clubs) {  
        if (club.getName().equals(clubName)) {  
            for (Player player : players) {  
                this.players.put(player, club);  
            }  
        }  
    }  
}
```

Exemplo — Premier League

□ Classe **PremierLeague** – método **getPlayers**

```
public HashSet<Player> getPlayers(Club club) {  
    HashSet<Player> clubPlayers = new HashSet<>();  
  
    for (Player player : players.keySet()) {  
        if (players.get(player).equals(club)) {  
            clubPlayers.add(player);  
        }  
    }  
  
    return clubPlayers;  
}
```

Exemplo — Premier League

- Requisitos da aplicação (4):
 - Adicionar e remover clubes
 - Remover um clube não remove os jogadores de um clube, remove apenas a sua relação com esse clube.



Exemplo — Premier League

- ❑ Classe **PremierLeague** – métodos **addClub** e **removeClub**

```
public void addClub(Club club) {  
    if (club != null) {  
        clubs.add(club);  
    }  
}  
  
public void removeClub(Club club) {  
    if (club == null) {  
        return;  
    }  
    clubs.remove(club);  
  
    for (Player player : getPlayers(club)) {  
        players.put(player, null);  
    }  
}
```

Exemplo — Premier League

- Requisitos da aplicação (5):
 - Utilizando o processamento funcional:
 - Obter todos os jogadores que tenham um determinado texto no nome.
 - Saber quantos jogadores sem clube existem
 - Usar o processamento funcional em métodos anteriores?



Exemplo — Premier League

□ Classe **PremierLeague** – método **getPlayers**

```
public ArrayList<Player> getPlayers(String text) {  
    ArrayList<Player> foundPlayers = new ArrayList<>();  
  
    players.keySet().stream()  
        .filter(p -> p.getName().contains(text))  
        .forEach(p -> foundPlayers.add(p));  
  
    return foundPlayers;  
}
```

Exemplo — Premier League

- Classe **PremierLeague** – método **playersWithoutClub**

```
public int playersWithoutClub() {  
    return players.entrySet().stream()  
        .filter(pair -> pair.getValue() == null)  
        .map(pair -> 1)  
        .reduce(0, (runningSum, count) -> runningSum + 1);  
}
```

Exemplo — Premier League

❑ Classe **PremierLeague** – método **getPlayers**

```
public HashSet<Player> getPlayers(Club club) {  
    HashSet<Player> clubPlayers = new HashSet<>();  
  
    for (Player player : players.keySet()) {  
        if (players.get(player).equals(club)) {  
            clubPlayers.add(player);  
        }  
    }  
  
    return clubPlayers;  
}
```

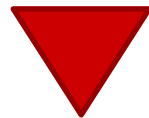


```
public HashSet<Player> getPlayersFunctional(Club club) {  
    HashSet<Player> clubPlayers = new HashSet<>();  
  
    players.keySet().stream()  
        .filter(player -> players.get(player).equals(club))  
        .forEach(player -> clubPlayers.add(player));  
  
    return clubPlayers;  
}
```

Exemplo — Premier League

❑ Classe **PremierLeague** – método **removeClub**

```
public void removeClub(Club club) {  
    if (club == null) {  
        return;  
    }  
    clubs.remove(club);  
  
    for (Player player : getPlayers(club)) {  
        players.put(player, null);  
    }  
}
```



```
public void removeClub(Club club) {  
    if (club == null) {  
        return;  
    }  
    clubs.remove(club);  
  
    getPlayers(club).stream()  
        .forEach(player -> players.put(player, null));  
}
```

Dúvidas?

