Pilares da Orientação à Objeto

Encapsulamento, herança e polimorfismo

Encapsulamento

DEIXAR OS ITENS DE DADOS E AS FUNÇÕES QUE OS MANIPULAM DEFINIDOS DENTRO DE UMA MESMA ESTRUTURA

EM JAVA É COMUM CONSTRUIR OS ATRIBUTOS COMO PRIVADOS E MÉTODOS GETTERS E SETTERS PARA OBTER OU ALTERAR O VALOR DE UM ATRIBUTOS

```java
package placarTOP;

public class PlacarBasico {

    private String nomeTimeA;
    private String nomeTimeB;
    private int pontuacaoA;
    private int pontuacaoB;

    public PlacarBasico(String ta, String tb) {
        this.setNomeTimeA(ta);
        this.setNomeTimeB(tb);
        this.setPontuacaoA(0);
        this.setPontuacaoB(0);
    }

    public String getNomeTimeA() {
        return nomeTimeA;
    }

    public void setNomeTimeA(String nomeTimeA) {
        this.nomeTimeA = nomeTimeA;
    }

    public String getNomeTimeB() {
        return nomeTimeB;
    }

    public void setNomeTimeB(String nomeTimeB) {
        this.nomeTimeB = nomeTimeB;
    }

    public int getPontuacaoA() {
        return pontuacaoA;
    }

    public void setPontuacaoA(int pontuacaoA) {
        this.pontuacaoA = pontuacaoA;
    }

    public int getPontuacaoB() {
        return pontuacaoB;
    }

    public void setPontuacaoB(int pontuacaoB) {
        this.pontuacaoB = pontuacaoB;
    }

    public void pontuar(int time) {
        if (time == 0) {
            this.setPontuacaoA(this.getPontuacaoA()+1);
        } else {
            this.setPontuacaoB(this.getPontuacaoB()+1);
        }
    }

    public void mostrarPlacar() {
        System.out.printf("%s X %s\n", this.nomeTimeA, this.nomeTimeB);
        System.out.printf("%03d - %03d", this.pontuacaoA, this.pontuacaoB);
    }

}
```

Atributos encapsulados

Getters e Setters

```java
package placarTOP;

public class PlacarBasico {

    private String nomeTimeA;
    private String nomeTimeB;
    private int pontuacaoA;
    private int pontuacaoB;

    public PlacarBasico(String ta, String tb) {
        this.setNomeTimeA(ta);
        this.setNomeTimeB(tb);
        this.setPontuacaoA(0);
        this.setPontuacaoB(0);
    }

    public String getNomeTimeA() {
        return nomeTimeA;
    }

    public void setNomeTimeA(String nomeTimeA) {
        this.nomeTimeA = nomeTimeA;
    }

    public String getNomeTimeB() {
        return nomeTimeB;
    }

    public void setNomeTimeB(String nomeTimeB) {
        this.nomeTimeB = nomeTimeB;
    }

    public int getPontuacaoA() {
        return pontuacaoA;
    }

    public void setPontuacaoA(int pontuacaoA) {
        this.pontuacaoA = pontuacaoA;
    }

    public int getPontuacaoB() {
        return pontuacaoB;
    }

    public void setPontuacaoB(int pontuacaoB) {
        this.pontuacaoB = pontuacaoB;
    }

    public void pontuar(int time) {
        if (time == 0) {
            this.setPontuacaoA(this.getPontuacaoA()+1);
        } else {
            this.setPontuacaoB(this.getPontuacaoB()+1);
        }
    }

    public void mostrarPlacar() {
        System.out.printf("%s X %s\n", this.nomeTimeA, this.nomeTimeB);
        System.out.printf("%03d - %03d", this.pontuacaoA, this.pontuacaoB);
    }

}
```

Herança

1

Uma classe filha herda todos os comportamentos e característica do pai

2

Ela deixa o comportamento mais específico, enquanto o pai é mais generalista

3

Também é conhecida como generalização ou especificação

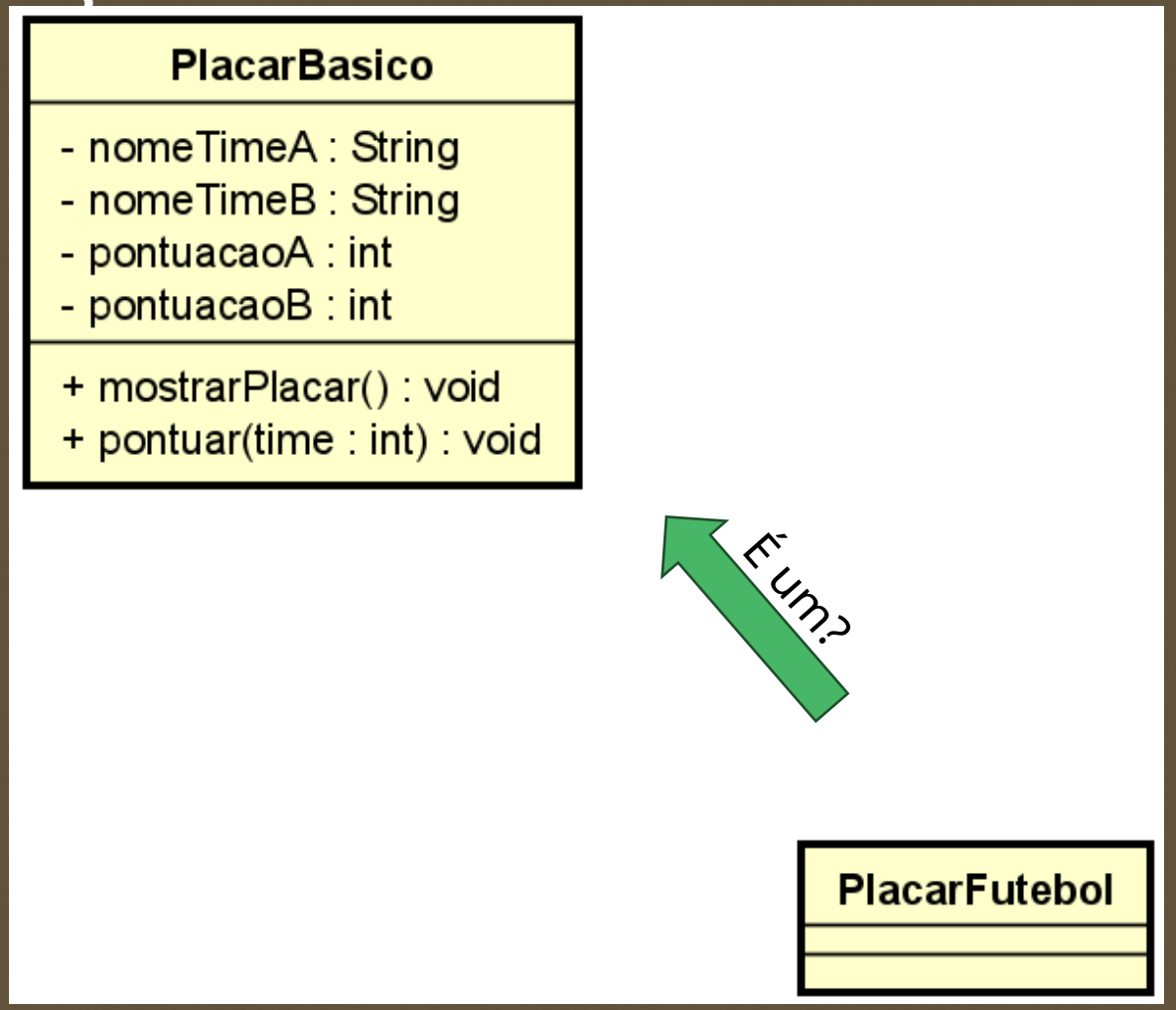
Herança

A classe filha deve ser uma coisa do mesmo tipo do pai

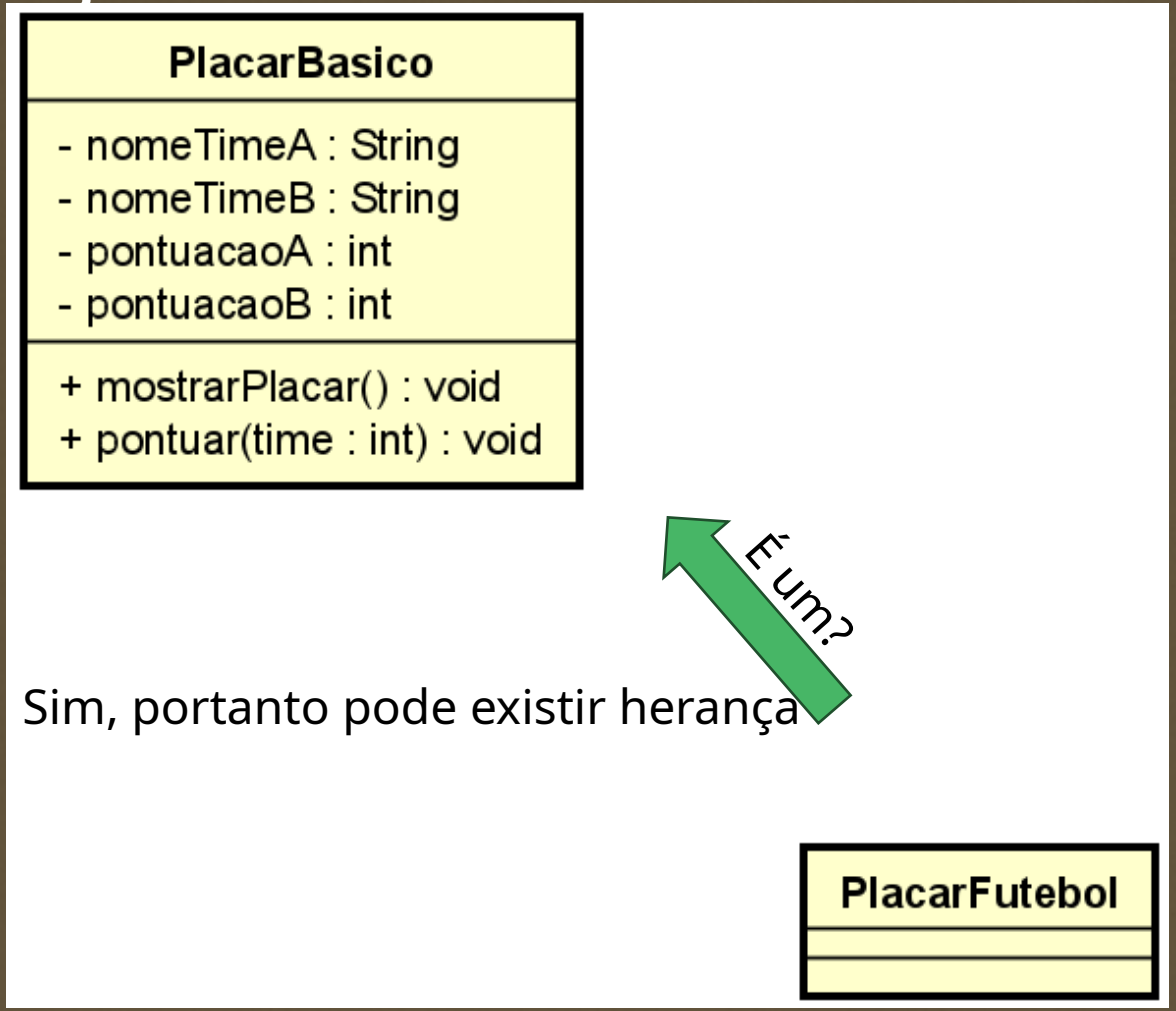
A classe filha pode reescrever comportamentos da classe pai

Sobrescrita de métodos

Herança


PlacarBasico
- nomeTimeA : String
- nomeTimeB : String
- pontuacaoA : int
- pontuacaoB : int
+ mostrarPlacar() : void
+ pontuar(time : int) : void
É um?
PlacarFutebol

Herança



Sim, portanto pode existir herança

Herança

```
1  package placarTOP;
2
3  public class PlacarFutebol extends PlacarBasico{
4
5      public PlacarFutebol(String ta, String tb) {
6          super(ta, tb);
7      }
8
9  }
```

Herança

```
1  package placarTOP;
2
3  public class PlacarFutebol extends PlacarBasico{
4
5      public PlacarFutebol(String ta, String tb) {
6          super(ta, tb);
7      }
8
9  }
```

Cria classe PlacarFutebol

Estende (herda)

PlacarBasico

Chama o construtor da classe pai

Herança

```
1  package placarTOP;
2
3  public class PlacarFutebol extends PlacarBasico{
4
5      public PlacarFutebol(String ta, String tb) {
6          super(ta, tb);
7      }
8
9  }
```

Cria classe PlacarFutebol

Estende (herda)

PlacarBasico

```
public PlacarBasico(String ta, String tb) {
    this.setNomeTimeA(ta);
    this.setNomeTimeB(tb);
    this.setPontuacaoA(0);
    this.setPontuacaoB(0);
}
```

Herança

É possível implementar outros comportamentos em métodos com os mesmos nomes

```java
package placarTOP;

public class PlacarFutebol extends PlacarBasico{

    public PlacarFutebol(String ta, String tb) {
        super(ta, tb);
    }

    @Override
    public void mostrarPlacar() {
        System.out.printf("%s %d\n",this.getNomeTimeA(),this.getPontuacaoA());
        System.out.printf("%s %d\n",this.getNomeTimeB(),this.getPontuacaoB());
    }

}
```

```java
package placarTOP;

public class PlacarFutebol extends PlacarBasico{

    public PlacarFutebol(String ta, String tb) {
        super(ta, tb);
    }

    @Override
    public void mostrarPlacar() {
        System.out.printf("%s %d\n",this.getNomeTimeA(),this.getPontuacaoA());
        System.out.printf("%s %d\n",this.getNomeTimeB(),this.getPontuacaoB());
    }

}
```

Anota que o método está sobrescrevendo um método da classe pai

Mesmo nome e parâmetros da classe pai

Anotações

- Metadados sobre o código
- Não realiza efeito direto no código anotado
- Pode ser utilizado para:
  - Informar comportamentos para o compilador
  - Processamento em tempo de compilação e deploy
  - Processamento em tempo de execução

```java
import java.util.Scanner;

public class Display {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Digite o nome do time A:");
        String nomeA = sc.nextLine();
        System.out.println("Digite o nome do time B:");
        String nomeB = sc.nextLine();
        sc.close();

        PlacarBasico pb = new PlacarBasico(nomeA, nomeB);
        pb.pontuar(0);
        pb.pontuar(0);
        pb.pontuar(0);
        pb.mostrarPlacar();
    }

}
```

Problems Servers Terminal Data Source Explorer Properties Console

<terminated> Display [Java Application] C:\Users\joaoc\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\

```
Digite o nome do time A:
MAC
Digite o nome do time B:
NOR
MAC X NOR
003 - 000
```

```java
import java.util.Scanner;

public class Display {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.println("Digite o nome do time A:");
        String nomeA = sc.nextLine();
        System.out.println("Digite o nome do time B:");
        String nomeB = sc.nextLine();
        sc.close();

        PlacarBasico pb = new PlacarFutebol(nomeA, nomeB);
        pb.pontuar(0);
        pb.pontuar(0);
        pb.pontuar(0);
        pb.mostrarPlacar();
    }

}
```

Problems Servers Terminal Data Source Explorer Properties Console

<terminated> Display [Java Application] C:\Users\joaoc\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jr

```
Digite o nome do time A:
MAC
Digite o nome do time B:
NOR
MAC 3
NOR 0
```

Polimorfismo

A CAPACIDADE DE OBJETOS SE ADEQUAREM A CONTEXTOS

MUDA O COMPORTAMENTO DURANTE A EXECUÇÃO DO PROGRAMA

GARANTE MUITA FLEXIBILIDADE NO CÓDIGO

```java
public class Display {

    public static void main(String[] args) {
        PlacarBasico pb = null;
        Scanner sc = new Scanner(System.in);
        System.out.println("Digite o nome do time A:");
        String nomeA = sc.nextLine();
        System.out.println("Digite o nome do time B:");
        String nomeB = sc.nextLine();
        System.out.println("Qual placar você prefere? 1 ou 2");
        int pType = sc.nextInt();
        if (pType == 1) {
            pb = new PlacarBasico(nomeA, nomeB);
        }else {
            pb = new PlacarFutebol(nomeA, nomeB);
        }

        int parar = 0;
        while (parar == 0) {
            pb.mostrarPlacar();
            System.out.println();
            System.out.println("O jogo terminou?");
            parar = sc.nextInt();
            if (parar == 1) {
                break;
            }
            System.out.println("Qual time fez gol?");
            int time = sc.nextInt();
            pb.pontuar(time);
        }
        sc.close();

    }

}
```

```java
public class Display {

    public static void main(String[] args) {
        PlacarBasico pb = null;
        Scanner sc = new Scanner(System.in);
        System.out.println("Digite o nome do time A:");
        String nomeA = sc.nextLine();
        System.out.println("Digite o nome do time B:");
        String nomeB = sc.nextLine();
        System.out.println("Qual placar você prefere? 1 ou 2");
        int pType = sc.nextInt();
        if (pType == 1) {
            pb = new PlacarBasico(nomeA, nomeB);
        }else {
            pb = new PlacarFutebol(nomeA, nomeB);
        }

        int parar = 0;
        while (parar == 0) {
            pb.mostrarPlacar();
            System.out.println();
            System.out.println("O jogo terminou?");
            parar = sc.nextInt();
            if (parar == 1) {
                break;
            }
            System.out.println("Qual time fez gol?");
            int time = sc.nextInt();
            pb.pontuar(time);
        }
        sc.close();

    }

}
```

Ambos objetos ficam na mesma variável

Exercício

- Crie uma classe "Livro" com atributos como título, autor e número de páginas. Instancie vários objetos desta classe e exiba suas informações.
- Modifique a classe "Livro" para tornar seus atributos privados. Implemente métodos getters e setters para acessar e modificar esses atributos.
- Adicione um construtor à classe "Livro" que inicialize todos os atributos. Crie outro construtor que aceite apenas título e autor.
- Crie uma classe "LivroDigital" que herde de "Livro". Adicione um atributo específico como "tamanhoEmMB" e um método para exibir todas as informações.
- Na classe "LivroDigital", sobrescreva o método de exibição de informações para incluir o tamanho do arquivo.