

Técnicas Avançadas de Programação

Desenvolvimento para Web e Dispositivos Móveis

1º Ano, 2º Semestre

Joana Fialho

E-mail: jfialho@estgv.ipv.pt

Nuno Costa

E-mail: ncosta@estgv.ipv.pt

Carlos Simões

E-mail: csimoes@estgv.ipv.pt

Escola Superior de Tecnologia e Gestão de Viseu
2019-2020

Herança

A herança permite criar uma classe por herança (ou extensão) de outra classe existente. Essa classe herda as variáveis de instância e os métodos de instância; acrescenta novos membros.

Um objeto desta nova classe contém os membros de instância da classe existente mais os membros novos. Por exemplo, criar a classe Aluno que herda da classe Pessoa. A classe Aluno contém todas as características da Pessoa mais as específicas de Aluno. Aluno é subclasse de Pessoa; Pessoa é superclasse de Aluno. Aluno é uma especialização de Pessoa. Aluno É uma Pessoa.

Herança

Em Java todas as classes herdam a classe Object com todos os seus membros e métodos.

A palavra reservada super é usada quando queremos chamar a superclasse:

- Com parâmetros, apenas no construtor e como 1ª instrução
- Seguida de (.) e método da superclasse para distinguir o método local do método da superclasse (quando existem com o mesmo nome).

```
public class Aluno extends Pessoa {  
    private int numMec;  
    private String curso;  
  
    public Aluno(int n, String nif)  
    {  
        super(nif);  
        this.numMec=n;  
        this.curso="";  
    }  
  
    public void setNumMec(int n){numMec=n;}  
  
    public void setCurso(String str){curso=str;}  
  
    public int getNumMec(){return numMec;}  
  
    public String getCurso(){return curso;}  
}
```

Se uma classe for definida como final (na assinatura), não pode ser herdada.

Polimorfismo

Em JAVA como na generalidade da POO, uma classe pode redefinir, i.e., modificar, alguns métodos herdados da sua classe base.

A esta ideia chama-se de polimorfismo, a possibilidade de tratar da mesma maneira objetos de tipos diferentes.

Utiliza-se cada objeto como se ele fosse da classe de base, mas o seu comportamento efetivo depende da sua classe efetiva (derivada desta classe de base), em particular da maneira de como os seus próprios métodos foram redefinidos.

O polimorfismo permite acrescentar novos objetos num cenário preestabelecido e, eventualmente, escrito tendo tido conhecimento do tipo exato destes objetos.

O polimorfismo tem a ver com a capacidade da mesma instrução executar métodos diferentes em função dos objetos em causa.

Polimorfismo - Exemplo prático – Operações Matemáticas

Assim, vamos ter a classe OperacaoMatematica:

```
public class OperacaoMatematica {  
    public double calcular(double x, double y){  
        return 0;  
    }  
}
```

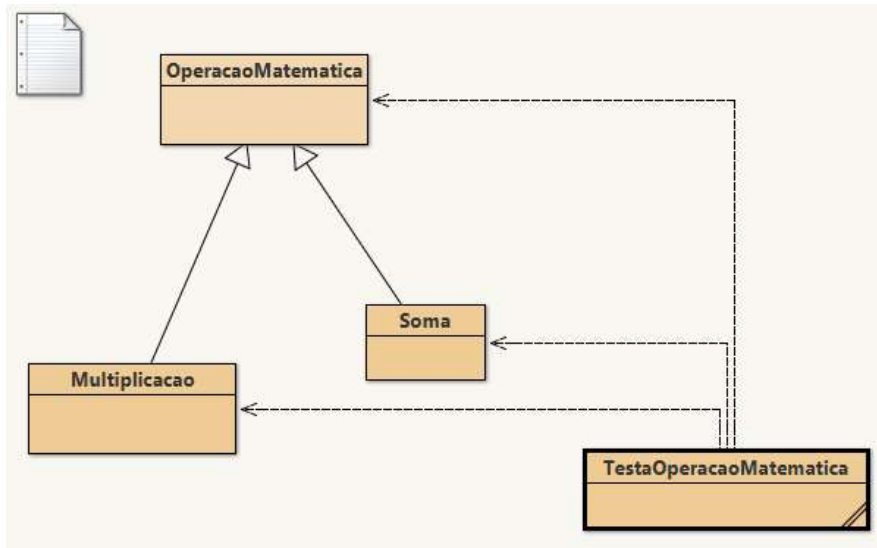
E duas classes que vão herdar:

A classe Multiplicacao:

```
public class Multiplicacao extends OperacaoMatematica {  
    public double calcular(double x, double y){  
        return x * y;  
    }  
}
```

E a classe Soma:

```
public class Soma extends OperacaoMatematica {  
    public double calcular(double x, double y){  
        return x + y;  
    }  
}
```



Polimorfismo - Exemplo prático – Operações Matemáticas

Por fim, testamos através da classe TestaOperacaoMatematica:

```
public class TestaOperacaoMatematica {  
    //executa a operação com polimorfismo  
    public static void calculaOperacao(OperacaoMatematica o, double x, double y){  
        System.out.println(o.calcular(x, y));  
    }  
    public static void main(String[] args) {  
        calculaOperacao (new Soma(), 1500, 300);  
        calculaOperacao (new Multiplicacao(), 12, 16);  
    }  
}
```

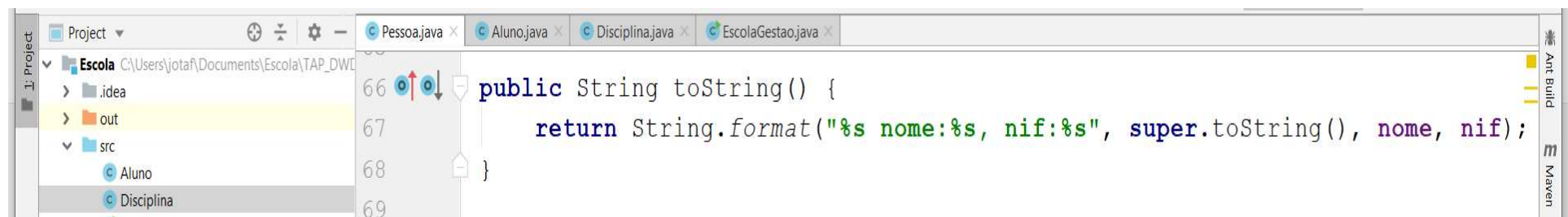
Executamos a aplicação e obtemos:



Redefinição de métodos

Os métodos de uma classe podem ser definidos ou redefinidos (*override*). Os métodos definidos existem apenas na classe e não nas superclasses. Os métodos redefinidos são redefinições de métodos existentes nas superclasses. A assinatura do método redefinido tem que ser igual ao da superclasse, mas altera alguma funcionalidade do método. Métodos redefinidos não podem diminuir a visibilidade.

Se um método é definido como final, não pode ser redefinido.

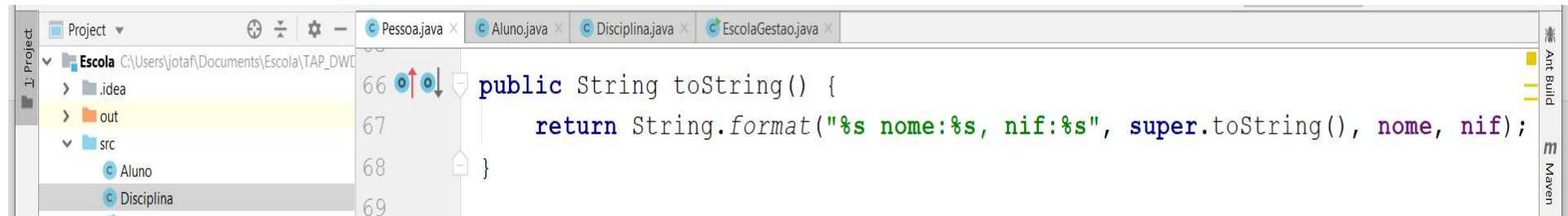


Redefinição de métodos

Os métodos de uma classe podem ser definidos ou redefinidos (*override*). Os métodos definidos existem apenas na classe e não nas superclasses. Os métodos redefinidos são redefinições de métodos existentes nas superclasses. A assinatura do método redefinido tem que ser igual ao da superclasse, mas altera alguma funcionalidade do método. Métodos redefinidos não podem diminuir a visibilidade.

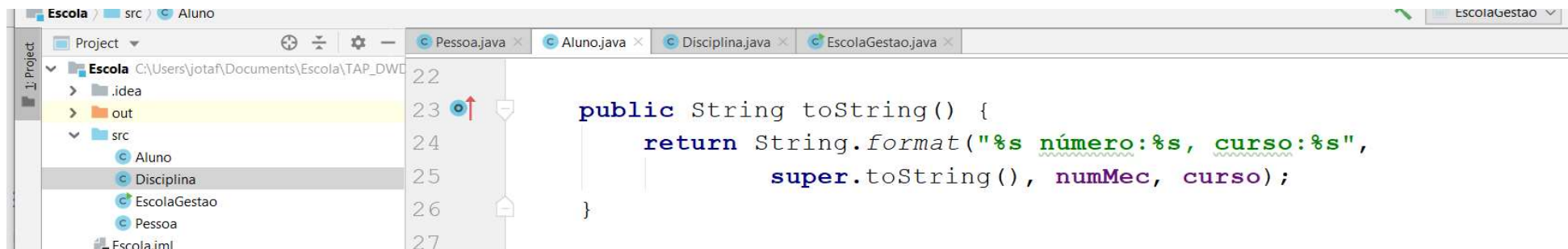
Se um método é definido como final, não pode ser redefinido.

Redefinição de métodos



The screenshot shows an IDE with a project named 'Escola'. The 'src' directory contains files for 'Aluno', 'Disciplina', and 'EscolaGestao'. The 'Pessoa.java' file is open, showing the following code:

```
66 public String toString() {  
67     return String.format("%s nome:%s, nif:%s", super.toString(), nome, nif);  
68 }  
69
```



The screenshot shows the same IDE with the 'Aluno.java' file open. The 'toString()' method is being defined, showing how it overrides the method in 'Pessoa.java' by adding 'numero' and 'curso' attributes.

```
22  
23 public String toString() {  
24     return String.format("%s número:%s, curso:%s",  
25         super.toString(), numMec, curso);  
26 }  
27
```

Não confundir a redefinição e polimorfismo com sobrecarga/overloading. Este último permite a definição de múltiplos métodos/construtores com o mesmo nome, na mesma classe.