

Polimorfismo

Carlos Arruda Baltazar
UNIP – Cidade Universitária

A palavra polimorfismo vem do grego e significa aquilo que pode tomar várias formas. Esta característica é um dos conceitos essenciais da programação orientada a objeto (POO). Enquanto a herança se refere às classes e a sua hierarquia, o polimorfismo diz respeito aos métodos dos objetos. Existem três tipos de polimorfismo:

O polimorfismo por sobrecarga permite ter métodos com o mesmo nome, com funcionalidades similares, em classes sem nenhuma relação entre elas.

Classe1
- atributo1:int - atributo2:int
+ Classe1() + SetAtributo1(atributo1:int):void + SetAtributo2(atributo2:int):void + GetAtributo1():int + GetAtributo2():int + Soma():int

Classe2
- atributo1:int - atributo2:int
+ Classe2() + SetAtributo1(atributo1:int):void + SetAtributo2(atributo2:int):void + GetAtributo1():int + GetAtributo2():int + Soma():int



```
public class Classe1
{
    private int atributo1;
    private int atributo2;

    public Classe1()
    {
    }

    public int getAtributo1() {
        return atributo1;
    }

    public void setAtributo1(int atributo1) {
        this.atributo1 = atributo1;
    }

    public int getAtributo2() {
        return atributo2;
    }

    public void setAtributo2(int atributo2) {
        this.atributo2 = atributo2;
    }

    public int Soma()
    {
        return this.atributo1 + this.atributo2;
    }
}
```

```
public class Classe2 {
    private int atributo1;
    private int atributo2;

    public Classe2()
    {
    }

    public int getAtributo1() {
        return atributo1;
    }

    public void setAtributo1(int atributo1) {
        this.atributo1 = atributo1;
    }

    public int getAtributo2() {
        return atributo2;
    }

    public void setAtributo2(int atributo2) {
        this.atributo2 = atributo2;
    }

    public int Soma()
    {
        int a = 3;
        int b = 5;
        return a + b;
    }
}
```

O polimorfismo por sobrecarga permite definir métodos cuja utilização será diferente de acordo com o tipo de configurações que lhes são próprias. Por isso, é possível sobrecarregar o método e fazê-lo realizar ações diferentes para cada uma das operações.

O polimorfismo paramétrico representa a possibilidade de definir várias funções do mesmo nome, porém com parâmetros diferentes (em número e/ou tipo). O polimorfismo paramétrico torna possível a escolha automática do método a ser adotado em função do tipo de dado passado em parâmetro.

Classe1
- atributo1:int
+ Classe1() + SetAtributo1(atributo1:int):void + SetAtributo2(atributo2:int):void + GetAtributo1():int + GetAtributo2():int + Soma(int a, int b):int + Soma(float a, float b):float + Soma(double a, double b):double



```
public class Classe1
{
    private int atributo1;
    private int atributo2;

    public Classe1()
    {
    }

    public int getAtributo1() {
        return atributo1;
    }

    public void setAtributo1(int atributo1) {
        this.atributo1 = atributo1;
    }

    public int getAtributo2() {
        return atributo2;
    }

    public void setAtributo2(int atributo2) {
        this.atributo2 = atributo2;
    }

    public int Soma(int a, int b)
    {
        return a + b;
    }

    public float Soma(float a, float b)
    {
        return a + b;
    }

    public double Soma(double a, double b)
    {
        return a + b;
    }
}
```

Assim, podemos definir vários métodos homônimos que são diferenciados pelos seus parâmetros.

O polimorfismo por substituição traz a possibilidade de redefinir um método em classes herdeiras de uma classe básica é chamada de especialização. Assim sendo, é possível chamar o método de um objeto sem se preocupar com o seu tipo intrínseco de polimorfismo de inclusão. Isso permite fazer abstração dos detalhes das classes especializadas de uma família de objetos, ocultando-os através de uma interface comum, chamada de classe básica.



```
public class Classe1
{
    private int atributo1;
    private int atributo2;

    public Classe1()
    {
    }

    public int getAtributo1() {
        return atributo1;
    }

    public void setAtributo1(int atributo1) {
        this.atributo1 = atributo1;
    }

    public int getAtributo2() {
        return atributo2;
    }

    public void setAtributo2(int atributo2) {
        this.atributo2 = atributo2;
    }

    public int Soma()
    {
        return this.atributo1 + this.atributo2;
    }
}
```



```
public class Classe2 extends Classe1
{
    private int atributo3;

    public Classe2()
    {
    }

    public int getAtributo3() {
        return atributo3;
    }

    public void setAtributo3(int atributo3) {
        this.atributo3 = atributo3;
    }

    public int Soma(int a, int b)
    {
        return a + b;
    }
}
```



```
public class Classe3 extends Classe1
{
    private int atributo4;

    public Classe3()
    {
    }

    public int getAtributo4() {
        return atributo4;
    }

    public void setAtributo4(int atributo4) {
        this.atributo4 = atributo4;
    }

    public double Soma(double a, double b)
    {
        return a + b;
    }
}
```

OBRIGADO