DIN Departamento de Informática

UEM Universidade Estadual de Maringá

Polimorfismo

Lilian Passos Scatalon
lpscatalon2@uem.br

Conceitos de POO

Abstração

Encapsulamento

Herança

Polimorfismo

Polimorfismo

poli = muitas, *morphos* = formas

Mecanismo que permite tratar um objeto da subclasse como se fosse da superclasse

Programar “no geral”, em vez de “no específico”

Tipos de polimorfismo

Polimorfismo em tempo de compilação

Funções com o mesmo nome, mas com assinaturas diferentes (nome + lista de parâmetros)

```
static int soma(int x, int y) {
    return x+y;
}

static double soma(double x, double y) {
    return x+y;
}


public static void main(String[] args) {

    System.out.println("Soma 1 = "+soma(4,5));
    System.out.println("Soma 2 = "+soma(3.1, 5.2));

}
```

Polimorfismo em tempo de execução

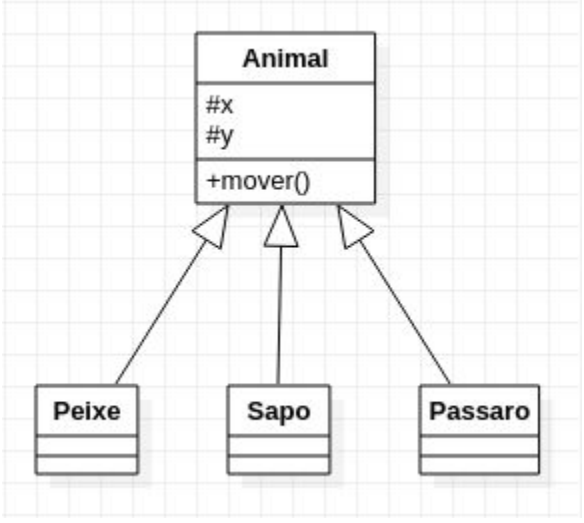
Suponha um programa para simular o movimento de diversos tipos de animais

A classe Animal contém o método mover() e mantém a posição atual como uma coordenada (x,y)

Classes Peixe, Sapo e Pássaro estendem Animal

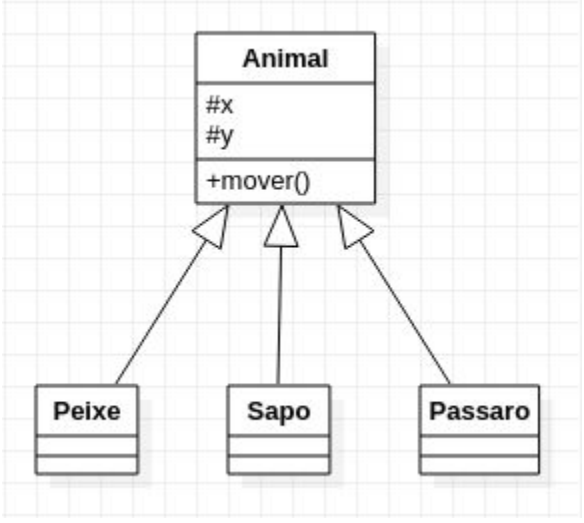
Polimorfismo

Cada subclasse implementa o método mover() com o respectivo comportamento específico (o peixe nada 1m, o sapo pula 0,5m e o pássaro voa 3m)



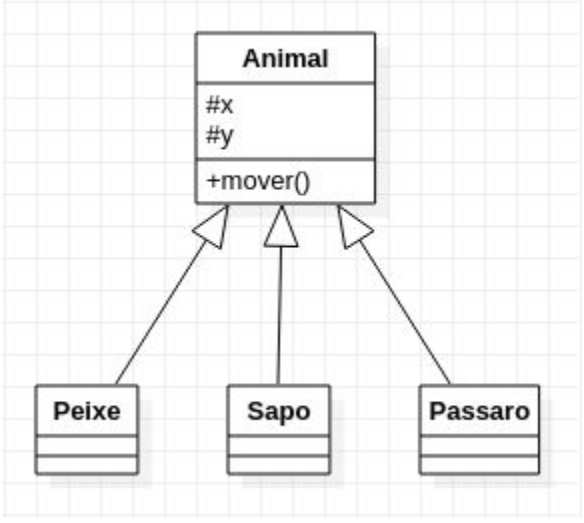
Polimorfismo

Pelo polimorfismo, o programa pode chamar o mesmo método mover() de modo genérico a uma referência de Animal, mas cada objeto específico vai saber como modificar suas coordenadas para o tipo de movimento feito pelo animal em questão



Polimorfismo

Pelo polimorfismo, o programa pode chamar o mesmo método mover() de modo genérico a uma referência de Animal, mas cada objeto específico vai saber como modificar suas coordenadas para o tipo de movimento feito pelo animal em questão



Polimorfismo

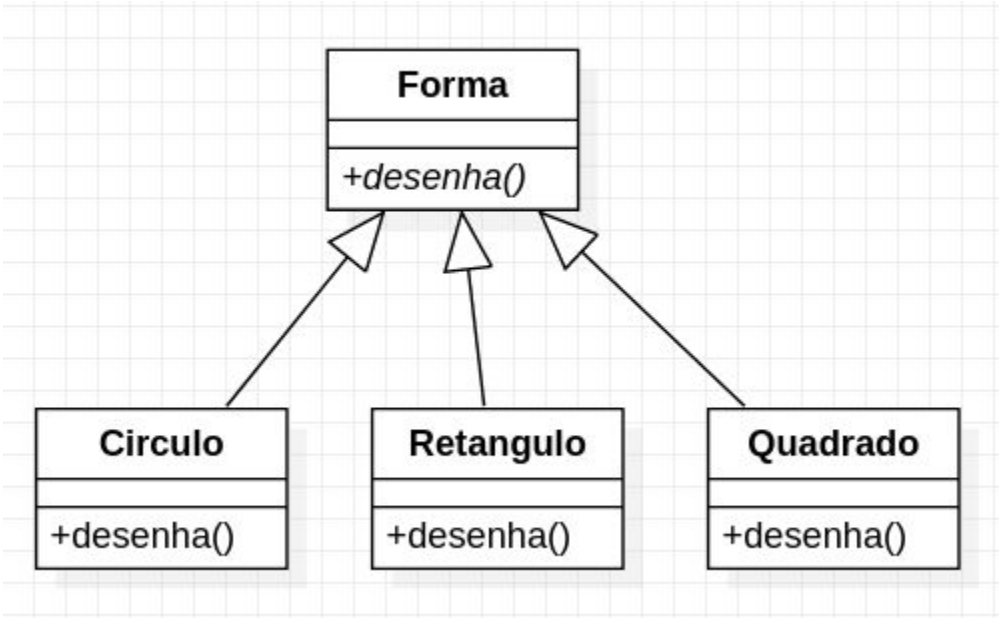
Usa-se uma referência da superclasse com um objeto da subclasse

Um objeto da subclasse pode ser tratado como um objeto da superclasse (por causa do relacionamento é-um)

```
Animal a1 = new Peixe();
Animal a2 = new Sapo();
Animal a3 = new Passaro();

a1.move();
a2.move();
a3.move();
```

Polimorfismo



Método sobrescrita

```
public class Forma {

    public void desenha() {
        System.out.println("Desenha forma");
    }

}
```

Método sobrescrita

```java
public class Quadrado extends Forma{

    public void desenha() {
        System.out.println("Desenha quadrado");
    }

}
```

```java
public class Triangulo extends Forma{

    public void desenha() {
        System.out.println("Desenha triangulo");
    }

}
```

```java
public class Circulo extends Forma{

    public void desenha() {
        System.out.println("Desenha círculo");
    }

}
```

Método sobrecarregado X Método sobrescrito

Para métodos sobrecarregados, só é possível fazer a chamada a partir do objeto da subclasse

O método sobrescrito pode ser chamado a partir de uma referência para a superclasse

Classes abstratas

Definimos classes para que objetos sejam instanciados (a classe é o tipo do objeto)

Porém, há situações em que é útil definir uma classe que nunca será instanciada

Exemplos: Forma, Animal

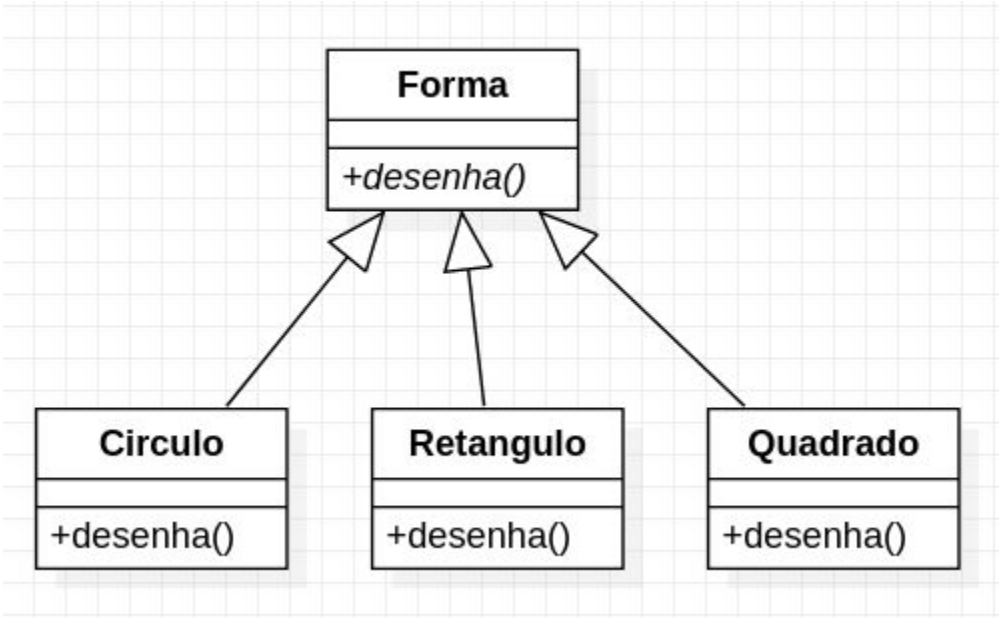
Classes abstratas

Classes que não podem ser instanciadas

Geralmente são usadas como superclasse em uma hierarquia de herança

Classes abstratas

A classe abstrata é uma classe básica a partir da qual outras classes podem herdar



Classes abstratas

Uma classe abstrata é declarada com a palavra-chave abstract

```
public abstract class Forma {

    // ...

}
```

Classes abstratas

Para que uma classe em Java seja abstrata, é preciso adicionar a palavra-chave abstract à sua declaração

```
public abstract class Forma {

}
```

Classes abstratas

A classe abstrata não pode ser instanciada

```
Forma f = new Forma();
```

```
'Forma' is abstract; cannot be instantiated
Implement methods Alt+Shift+Enter   More actions... Alt+Enter

public abstract class Forma
extends Object
```

Classes abstratas

É possível ter referências do tipo da classe abstrata

```
Forma f = new Circulo( raio: 5);
```

Métodos abstratos

Classes abstratas podem ter métodos abstratos

São métodos apenas declarados, mas sem implementação (com ponto-e-vírgula)

Usa-se a palavra-chave abstract também

Definem um comportamento a ser implementado por subclasses concretas

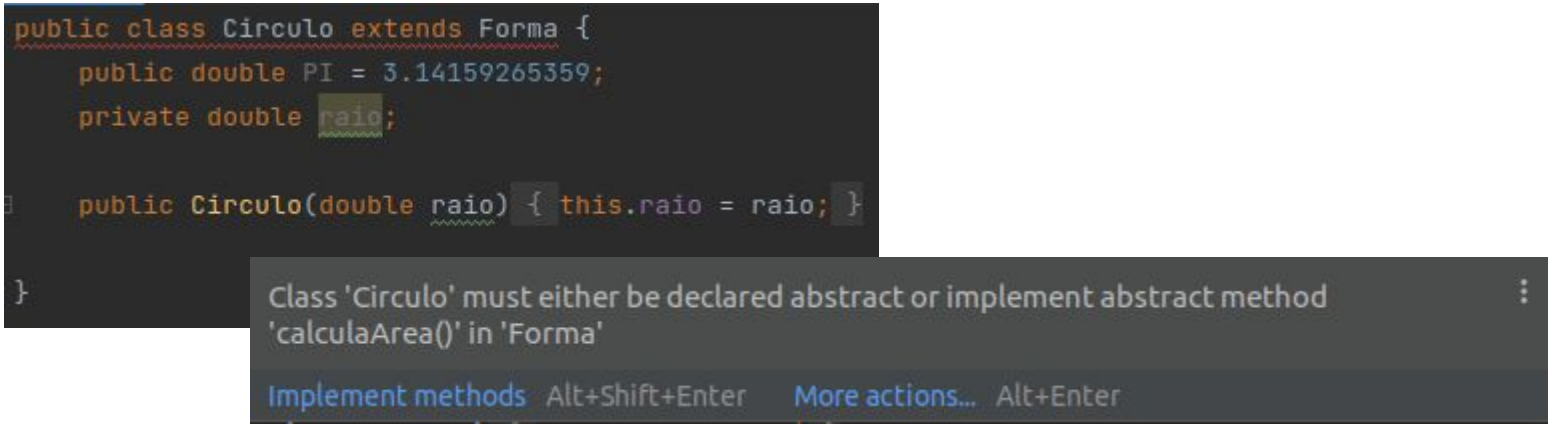
```
public abstract class Forma {

    public abstract double calculaArea();

}
```

Métodos abstratos

Se a classe derivada não o implementa, também deve ser uma classe abstrata



Métodos abstratos

O método abstrato deve ser implementado por classes derivadas da classe abstrata

```
public class Circulo extends Forma {
    public double PI = 3.14159265359;
    private double raio;

    public Circulo(double raio) { this.raio = raio; }

    public double calculaArea(){
        return PI*raio*raio;
    }
}
```

Classes abstratas

Uma classe abstrata pode ter construtores

```java
public abstract class Forma {

    private double area;

    public Forma(){
        area = 0;
    }

    public abstract double calculaArea();

}
```

```java
public class Circulo extends Forma {
    public double PI = 3.14159265359;
    private double raio;

    public Circulo(double raio){
        super();
        this.raio = raio;
    }
```

Bibliografia

H. Schildt. **Java: A Beginner's Guide**. McGraw-Hill Education, 8th edition, 2018.

C. S. Horstmann. **Core Java SE 9 for the Impatient**. Addison-Wesley, 2nd Edition, 2018.

Caelum. Apostila **Java e Orientação a Objetos**. Disponível em https://www.alura.com.br/apostila-java-orientacao-objetos