Orientação a Objetos

.:. Revisão .:.

Professora Ana Paula Lemke

Paradigmas de Programação

- Paradigma é a “filosofia” adotada na construção de software.
- Exemplos de paradigmas:
  - Procedimental
  - Lógico
  - Funcional
  - Orientado a agentes
  - **Orientado a Objetos**

Paradigma Orientado a Objetos

- Paradigma de programação no qual um programa é estruturado em **classes** e **objetos**.
  - Possibilita que objetos do mundo real sejam mapeados em objetos no computador e pressupõe que o mundo é composto por objetos.
  - Não existe uma maneira “correta” de decompor um problema em objetos; esta decomposição depende do julgamento do projetista e da natureza do problema.
  - O funcionamento de um sistema OO se dá através do relacionamento e troca de mensagens entre os objetos.

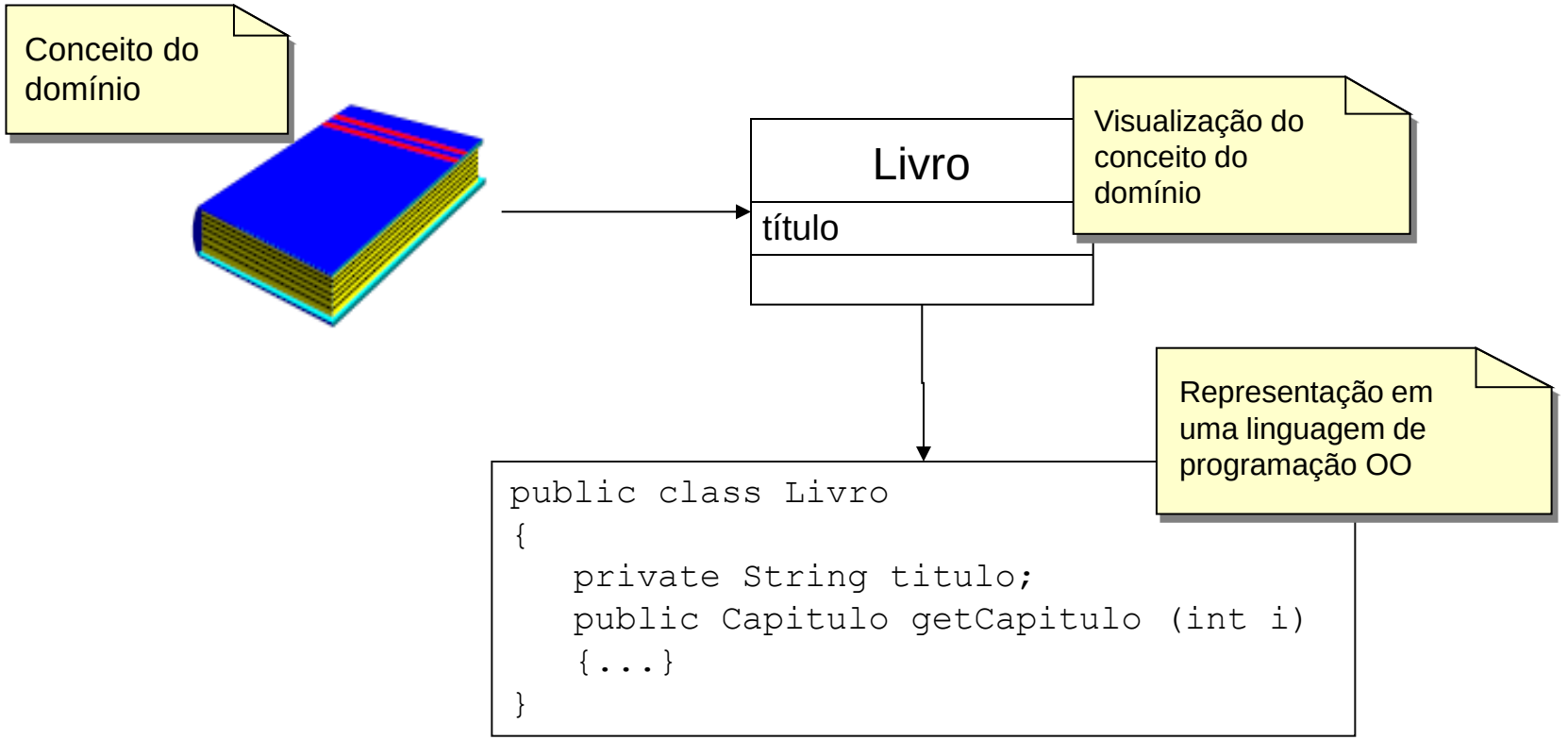
Vantagens da utilização de OO

- Alta reutilização de código.
- Redução no tempo de manutenção.
- Redução da complexidade através da melhoria do grau de abstração.
- Aumento de qualidade e produtividade, pois oferece maiores facilidades ao desenvolvedor.
- Aceitação comercial crescente.

Desenvolvimento Orientado a Objetos

- **Análise orientada a objetos**: desenvolvimento de um modelo orientado a objetos do domínio da aplicação.
- **Projeto orientado a objetos**: desenvolver um modelo orientado a objetos de um sistema de software para implementar os requisitos identificados. Está relacionado à solução do problema.
- **Programação orientada a objetos**: realização de um projeto de software utilizando uma linguagem de programação orientada a objetos, como Java.

Desenvolvimento Orientado a Objetos



Adaptado do livro "Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and the Unified Process", de Craig Larman, 2001

Cenário exemplo

Classes

"Carro vermelho"
- Características?
- Ações?

"Tibúrcio"
- Características?
- Ações?

Objetos

"Carro verde"
- Características?
- Ações?

Quando o carro anda?
Quando Tibúrcio passa uma ***mensagem*** contendo sua requisição*: acelerar!*

Mensagem

O carro verde tem a ***responsabilidade*** de, através *de* algum ***método****,* cumprir a requisição de seu motorista. O método utilizado pelo carro verde pode estar **oculto** de Tibúrcio..

Responsabilidades de objetos

Encapsulamento

Conceitos e definições

Conceitos da Orientação a Objetos

- Objeto
- Classe
- Atributo
- Método
- Mensagem
- Sobrecarga
- Herança

- Abstração
- Encapsulamento
- Polimorfismo
- ...

Objeto

"Qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e métodos que os manipulam". (*Odell & Martin*, 1995)

- Um **objeto** é uma entidade que possui *identidade,* estado e comportamentos.
- Podem representar entidades concretas ou conceituais.
  - Entidade física: caminhão, carro, bicicleta, etc.
  - Entidade conceitual: processo químico, matrícula, etc
  - Entidade de software: lista encadeada, arquivo, etc.
- Dois objetos são **distintos** mesmo que eles apresentem as mesmas características.

Objeto

- O **estado** do objeto é representado pelas informações encapsuladas por ele.
- Os **comportamentos** do objeto são as requisições que ele pode cumprir.

Tibúrcio é uma *Pessoa*.

Características:
- nome = Tibúrcio
- peso = 80 kg
- altura = 1,60 m

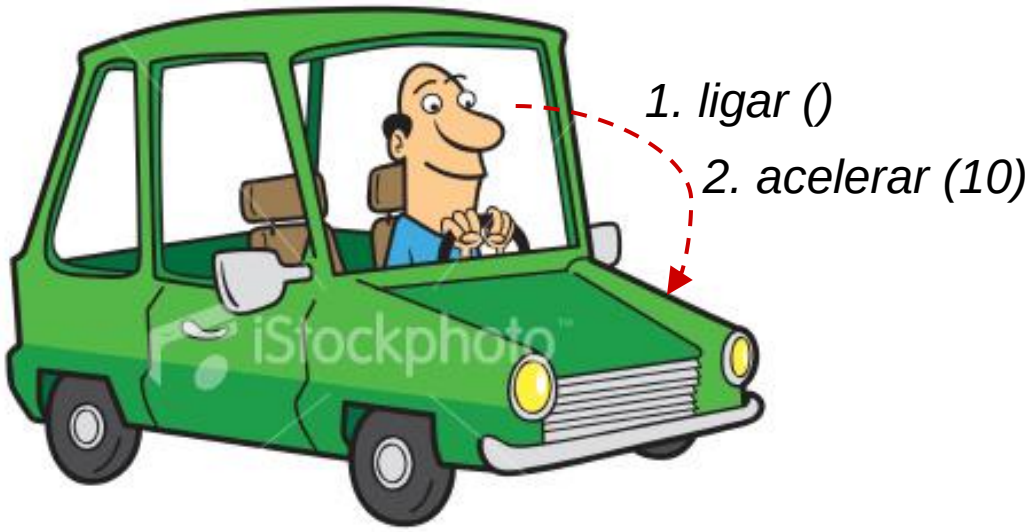
CarroVerde é um *Carro*.

Características:
- cor = Verde
- dono = 

Comportamentos:
-ligar, acelerar, freiar, qtdeCombustivel

- **Mensagens** são utilizadas para invocar um dos métodos de um objeto, ativando um de seus comportamentos.

Comunicação entre objetos



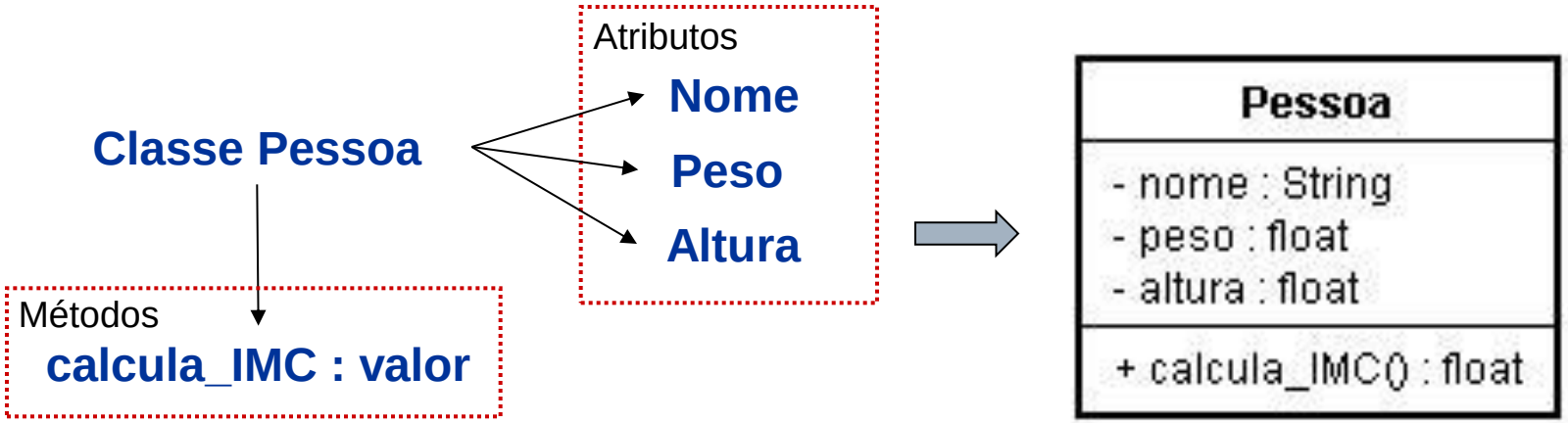
- Uma ação se inicia através do envio de uma mensagem para um objeto, que é o responsável por executar o comportamento.
- A mensagem carrega uma requisição, além de toda a informação necessária (argumentos) para que a ação seja executada.

Encapsulamento

- Esconder os detalhes da implementação de um objeto é chamado **encapsulamento**.
  - O encapsulamento permite que um objeto seja visto como uma caixa-preta pelo usuário: ele não sabe o que há dentro do objeto, sabe apenas para que ele serve e quais os métodos disponíveis para a manipulação deste.
  - Exemplo: quando Tibúrcio acelera o carro, diversos mecanismos entram em ação. Para Tibúrcio, os detalhes de como os dados são processados para que o carro se locomova, são irrelevantes.
- Benefícios:
  - O código cliente pode usar apenas a interface para a operação.
  - A implementação do objeto pode mudar (para corrigir erros, aumentar desempenho, etc) sem que seja necessário modificar o código do cliente.
  - A manutenção é mais fácil e menos custosa.

Classe

- Descreve um conjunto (possivelmente infinito) de objetos individuais que possuem as mesmas características e comportamentos.
- Uma classe define o conjunto de atributos (características) mantidos pelos objetos e os métodos (ações) que um objeto pode executar.



Representação gráfica (UML)

Atributos e Métodos

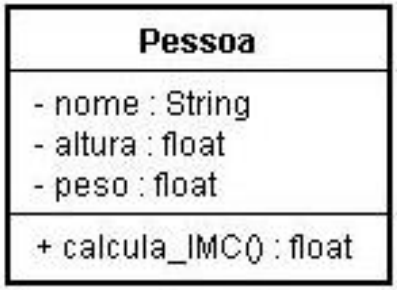
- Atributos contêm, normalmente, duas informações:
  - Nome do atributo
  - Tipo de dado (como *integer*, *float, boolean*)
- Métodos podem receber ou não parâmetros (valores usados durante as suas execuções) e podem retornar valores.
- Os métodos podem permitir a leitura e a modificação dos atributos.
  - Método de acesso*:* permite apenas a leitura das informações.
  - Método de modificação: permite alterar as informações armazenadas no contexto do objeto.

Um método em uma classe é apenas uma definição. A ação só ocorre quando o método é invocado através do objeto.

Classes e Objetos

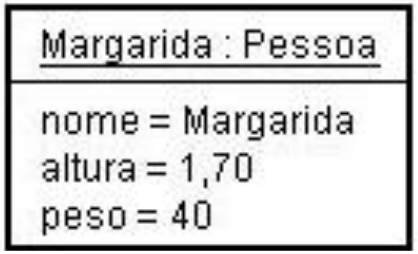
- Objetos são *instâncias* de classes.

- Exemplo:
  - Classe **Pessoa**:
    - nome: string;
    - peso: float;
    - altura: float;



Representação gráfica (UML) de uma classe

  - Objeto **Margarida** (instância de Pessoa)
    - nome = “Margarida”;
    - peso = 40 ;
    - Altura = 1,70;



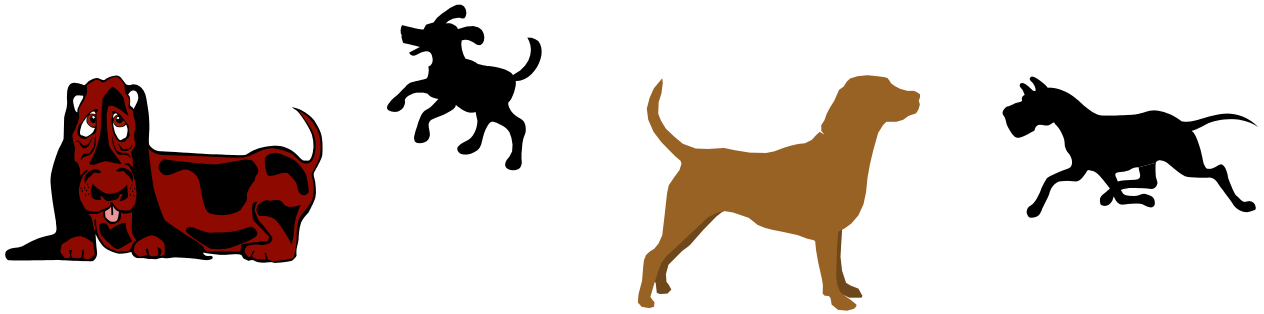
Representação gráfica (UML) de um objeto

Classificação, Abstração e Instanciação

- O ser humano, no início de sua infância, aprende e pensa de uma maneira orientada a objetos, representando seu conhecimento por meio de classificações e abstrações.
- Uma criança aprende conceitos simples como pessoa, carro e casa e ao fazer isso define **classes**.
  - Qualquer coisa que possua cabeça tronco e membros passa a ser uma pessoa, qualquer construção onde as pessoas possam entrar passa ser uma casa.
  - Esse processo já exige um grande esforço de abstração, pelas diferenças que os objetos de cada classe podem apresentar. Porém, mesmo diferentes, possuem a mesma classificação.

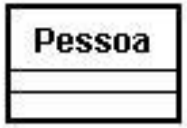
Classificação, Abstração e Instanciação

- Classificar: definir os objetos que pertencem a mesma classe.
- Abstrair: enxergar o **conceito**, chegando à conclusão que um termo (classe) se refere a muitos objetos que possuem **algumas** características semelhantes.
- Instanciar: criar um novo item do conjunto representado por uma classe.



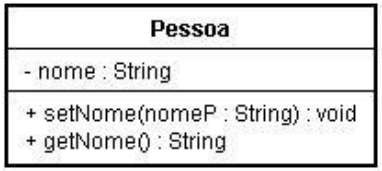
Classes e Objetos em Java

- Em Java, a declaração de novas classes é feita através da construção **class**.




```
public class Pessoa{

}
```




```
public class Pessoa {
    private String nome;

    public String getNome() {
        return this.nome;
    }

    public void setNome(String nomeP) {
        this.nome = nomeP;
    }
}
```

Classes e Objetos em Java

- Uma vez definida uma classe, uma nova instância (objeto) pode ser criada através do comando **new**.

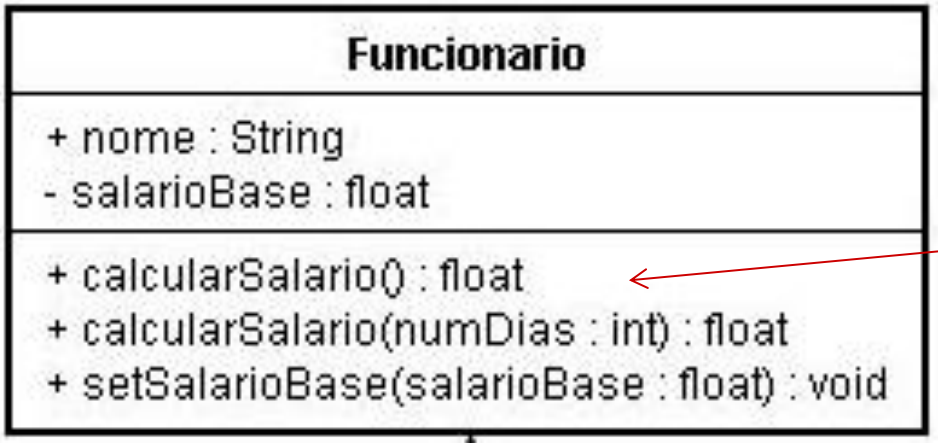
```
Pessoa ana = new Pessoa ();
```

- Em Java, o envio de uma mensagem é feito através de uma chamada de método com passagem de parâmetros.
  - Por exemplo, a mensagem que permite indicar o nome de uma pessoa é a chamada do método **setNome**.

```
ana.setNome ("Ana Paula");
```

Sobrecarga

- Sobrecarregar um método significa prover mais de uma versão de um mesmo método.
- As versões devem, necessariamente, possuir listas de parâmetros diferentes, seja no tipo ou no número desses parâmetros.



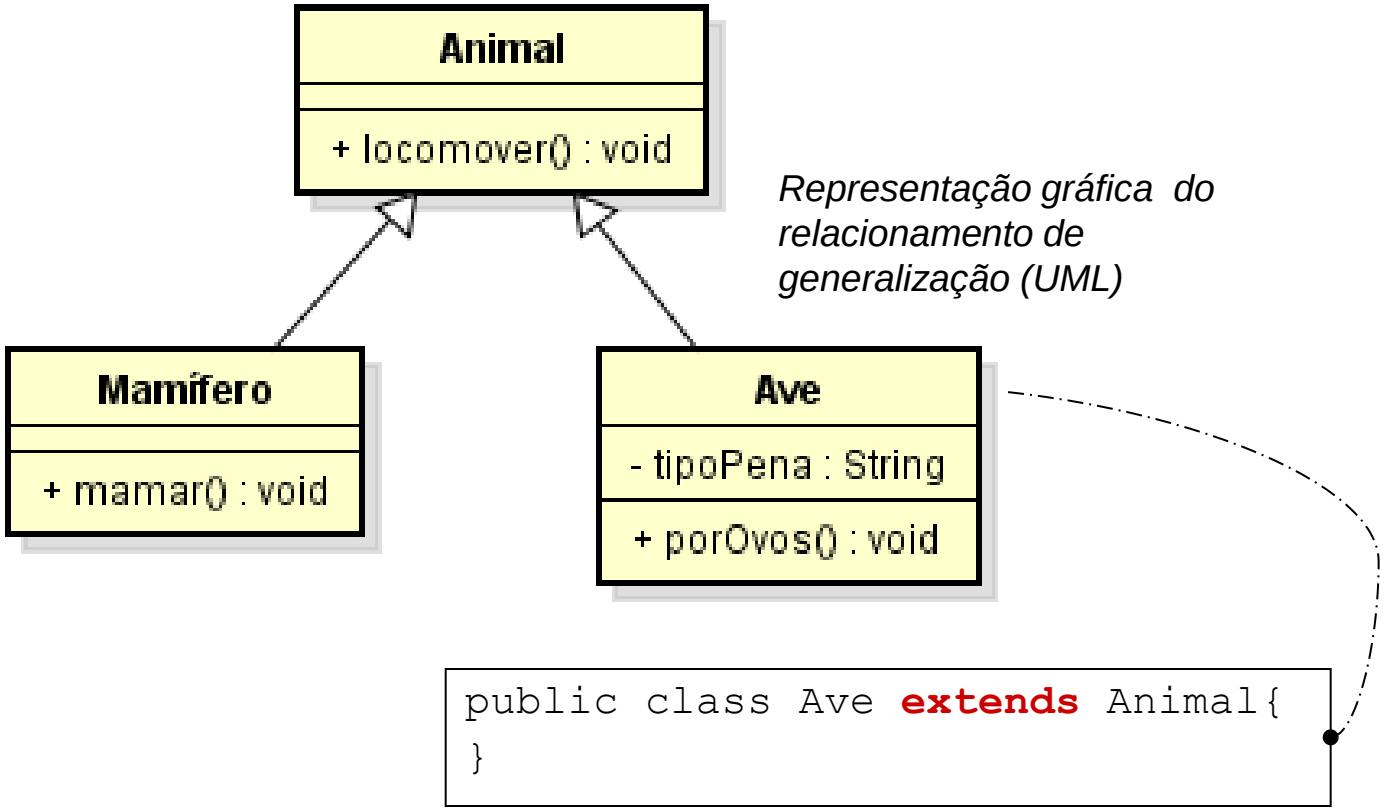
Método calcularSalario esta sobrecarregado (dois métodos com o mesmo nome, mas diferente conjunto de parâmetros)

Herança

- Permite o reaproveitamento de atributos e métodos.
- Trabalha com o conceito de superclasse e sub-classe:
  - Superclasses (classe pai): classe que possui outras classes derivadas dela.
  - Sub-classes (classe filha): classes derivadas de uma superclasse que herdam seus atributos e métodos.

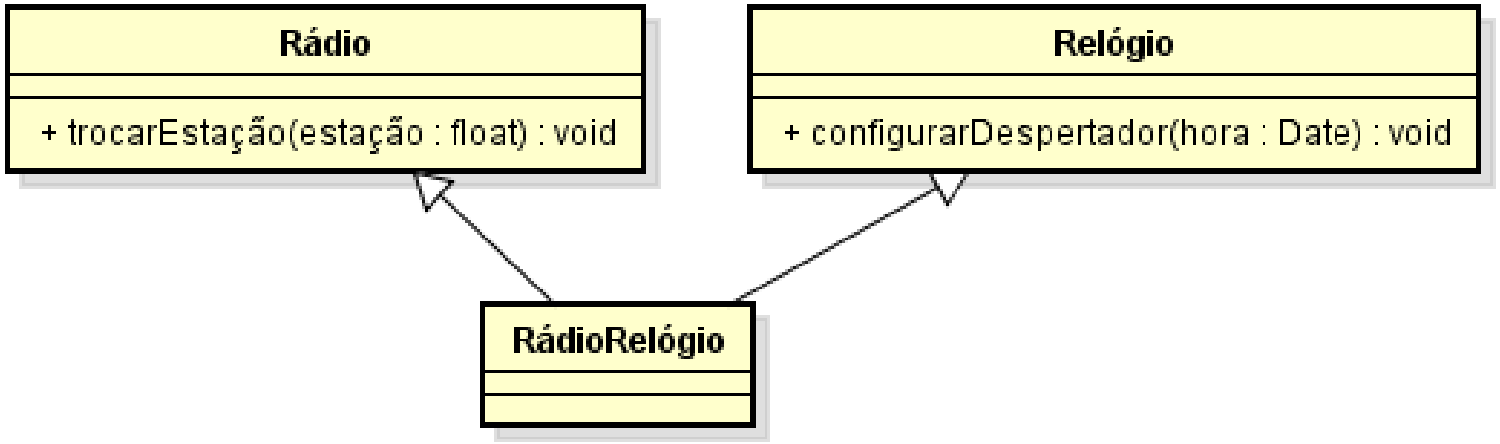
- Vantagens:
  - Otimiza o tempo de desenvolvimento;
  - Diminui linhas de código;
  - Facilita manutenções futuras.

Herança


Animal
+ locomover() : void
Mamífero
+ mamar() : void
Ave
- tipoPena : String
+ porOvos() : void
Representação gráfica do relacionamento de generalização (UML)
public class Ave extends Animal{
}

Herança Múltipla

- Há herança múltipla quando uma subclasse possui mais de uma superclasse.

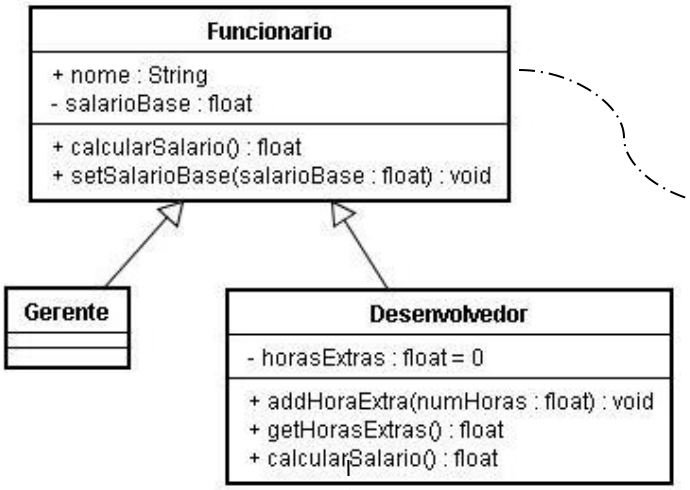


- Observação: a linguagem Java não suporta herança múltipla.

Polimorfismo

- Polimorfismo = múltiplas formas.
- Múltiplas formas de fazer algo.
  - Trabalha com a redeclaração de métodos herdados de uma superclasse.
  - Embora semelhantes, esses métodos diferem na forma de implementação. Dessa maneira podem existir dois ou mais métodos com a mesma nomenclatura mas com implementações diferentes.
- Capacidade de um objeto tomar diferentes formas.
  - Ao estendermos ou especializarmos uma classe, não perdemos compatibilidade com a superclasse.
    - Exemplo: a subclasse Ave de Animal é, além de outras coisas, um tipo de Animal.
  - Permite a manipulação de instâncias de classes diferentes (mas que herdem de uma mesma classe ancestral) de forma unificada.

Polimorfismo




```
public class Funcionario {
      ...
      public float calcularSalario ()
      {
            System.out.println ("Cálculo padrão");
            ...
      }
}
```

```
public class Desenvolvedor extends Funcionario{
      ...
      public float calcularSalario ()
      {
            System.out.println ("Cálculo com horas Extras");
            ...
      }
}
```

Polimorfismo

```
Funcionario funcionario1 = new Funcionario ();
Funcionario funcionario2 = new Gerente ();
Funcionario funcionario3 = new Desenvolvedor ();
```

Capacidade de um objeto ter diferentes formas

- Como saber qual método invocar para calcular o salário de cada funcionário?
  - A decisão sobre qual o método que deve ser selecionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução, através do **mecanismo de ligação tardia**.

```
funcionario1.calcularSalario();
funcionario2.calcularSalario();
funcionario3.calcularSalario();
```

  - Resposta:

```
Cálculo padrão...
Cálculo padrão...
Cálculo com horas extras...
```

Tarefa 1

- Identifique classes, objetos e atributos nos parágrafos abaixo.
  - “Todas os funcionários de uma empresa recebem um salário (valor em dinheiro) e o depositam em uma conta corrente. Eles consultam o saldo de sua conta corrente frequentemente, sacando dinheiro quando necessário”.
  - “José é um funcionário da empresa e como tal possui a ‘Conta corrente do José’. Em determinado momento, sua conta corrente possui um saldo de ‘R$ 500’. Já seu colega Pedro possui apenas ‘R$ 30,00’ de saldo em sua conta corrente”.

Bibliografia básica

- Santos, Rafael. **Introdução à Programação Orientada a Objetos usando Java**. Rio de Janeiro: Elsevier, 2003.
- Deitel, H., Deitel, P.J. **Java Como Programar**. São Paulo: Prentice-Hall, 2006, 6ª edição.