POO continuação

Prof. Dr. João Paulo Lemos Escola

Tópicos da aula

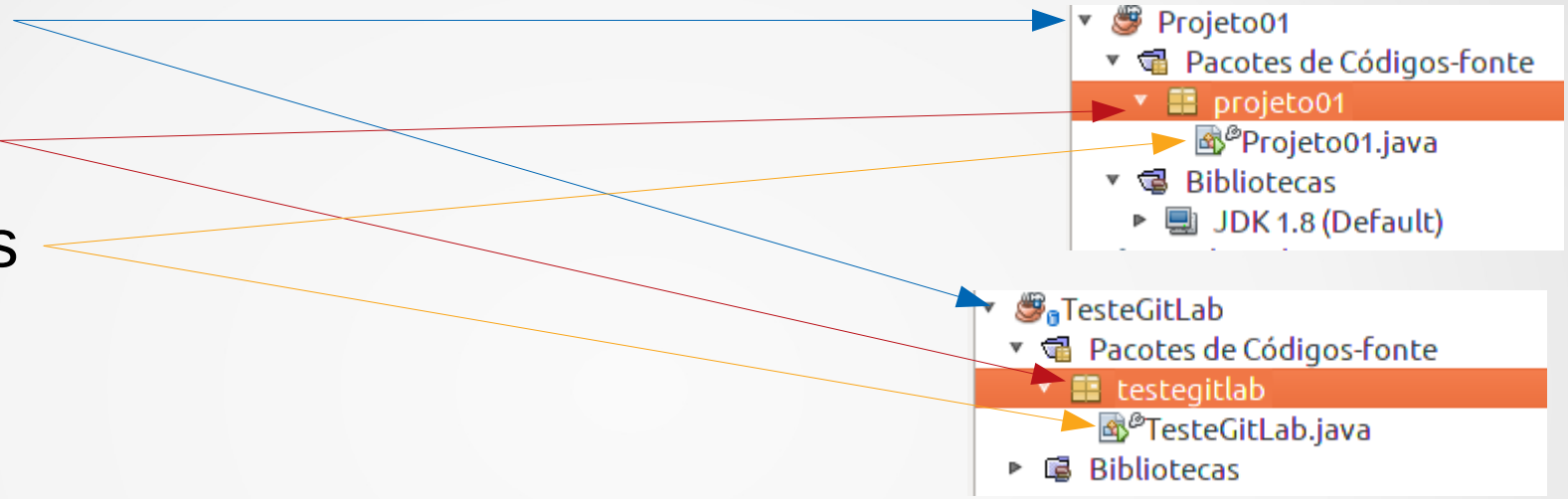
- Pacotes;
- Herança;
- Polimorfismo;
- Classe abstrata;
- Interfaces.

Pacote

- Um pacote é como uma pasta no projeto, utilizado para armazenar as classes com mesma finalidade;
- Toda classe pertence a algum pacote. Quando este não é especificado, é utilizado o pacote default, que é a pasta de códigos-fonte do projeto;
- Os nomes dos pacotes devem ser criados em letras minúsculas, a exemplo das variáveis;
- Classes devem iniciar com a declaração do pacote de que fazem parte, a partir da palavra reservada **package**;
- O padrão de desenvolvimento dita que os nomes dos pacotes sejam separados por pontos e com nome reverso ao domínio da empresa:
  - br.edu.ifsp.projeto.bean
  - com.google.android.projeto.dao

Exemplos de pacotes em projetos

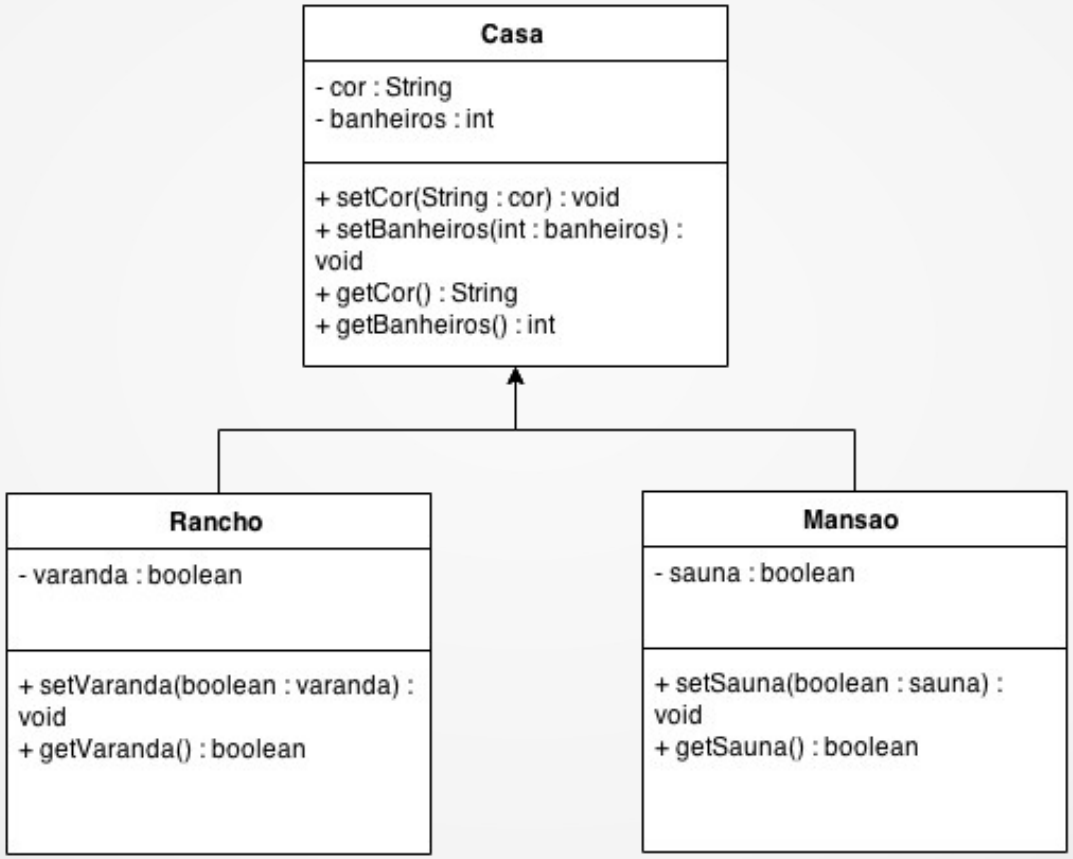
- Projeto
- Pacote
- Classes



Herança

- Assim como na vida real, a herança acontece quando uma classe (filha) recebe (herda) o conteúdo de outra classe (pai);
- As classes pais são chamadas de superclasses;
- As classes filhas são chamadas de subclasses;
- Essa técnica permite o reaproveitamento de código de uma classe em outra;
- Normalmente utilizamos herança quando queremos criar uma classe mais específica (especialista), com métodos não contemplados na classe original e que tenha características em comum com outra classe:
  - Classe Casa (pai), Classes Barraco, Mansão (filhas);
  - Classe Pessoa (pai), Classes Vendedor, Usuario, Cliente (filhas);
  - Classe Veiculo (pai), Classes Carro, Caminhão (filhas).

Diagrama de herança



Código de utilização de herança

- A declaração de uma classe filha deve utilizar a palavra reservada **extends** seguido do nome da classe pai:

```
modificador class nome-da-classe extends classe-pai {
   // código da classe
}
```

A8ex1.jar

- Implemente a classe Pessoa e, utilizando herança, as classes Vendedor e Cliente;
- Utilize construtor completo e vazio;
- Instancie os objetos vendedor1 e cliente1;
- Mostre os atributos dos objetos.

Polimorfismo

- Polimorfismo significa “pode variar de forma”;
- Este conceito, em linguagens orientadas a objeto permite que objetos sejam declarados com um determinado tipo e implementados como outro, desde que tenham relação (como no caso da herança);
- Para isso, é necessário que aconteça o overriding de métodos, ou seja, ambas as classes devem ter os mesmos métodos, com os mesmos tipos de retorno e parâmetros;
- Esta técnica é importante quando temos classes que devem ter comportamentos diferentes de acordo com determinada decisão ou escolha em tempo de execução:
  - Classes de acesso a banco de dados utilizando diferentes SGBDs;
  - Classes de generalização de acesso a dados;

Exemplo de Polimorfismo

- Criando uma variável do tipo Object;
- Utilizando como objeto String:

```
public static void main(String[] args) {
    Object teste;
    teste = "oi";
    System.out.println(teste);

}
```

A8ex2.jar

- Crie a classe Movel (pai) e a classe Mesa (filha);
- Implemente o método “meuNome” em ambas as classes, que retorne uma String “eu sou um móvel” ou “eu sou uma mesa” de acordo com a classe construída;
- Declare um objeto “mesinha” da classe Movel;
- Instancie o objeto e chame o método “meuNome”;
- Desabilite a instanciação, instancie com o construtor Mesa() e chame o método “meuNome” novamente.

Classes abstratas

- Não permite instanciação de objetos;
- Normalmente utilizada como base para criação de outras classes;
- Sintaxe:

```
modificador abstract class nome-da-classe{
   // código da classe
}
```

A8ex3.jar

- Copie as classes Movel e Mesa para o projeto atual;
- Transforme a classe Movel para abstrata;
- Tente instanciar um objeto movel1 da classe Movel (deve dar erro);
- Instancie o objeto mesa1 da classe Mesa (deve funcionar).

Interface

- Utilizadas para definir os métodos que devem existir nas classes derivadas;
- Funciona como uma definição de métodos para as classes, pois, na interface, somente são declarados os nomes dos métodos e não o seu conteúdo;
- As classes que implementam a interface devem utilizar a palavra reservada “implements” no lugar da “extends” utilizada em herança;
- Em projetos de software são comuns quando se há uma definição do escopo mas ainda não há a definição final das tarefas que cada método irá executar:

```
modificador interface nome-da-interface{
   // definição dos métodos
}
```

A8ex4.jar

- Crie uma interface BancoDeDados com os métodos salvar e excluir;
- Crie a classe Produto e a classe Venda implementando a interface BancoDeDados;
- Os métodos devem imprimir mensagens fictícias, como “Produto salvo!” e “Produto excluído!”

A8ex5.jar

Crie as classes Fabricante (codigo, nome), Componente (codigo, nome, modelo,Fabricante), Monitor (tamanho, tipo), Processador (Fabricante, velocidade, nucleos), Memoria (quantidade). Utilize herança entre as classes quando for possível.

O que aprendemos?

- Pacotes;
- Herança;
- Polimorfismo;
- Classe abstrata;
- Interfaces.

Na próxima aula...

- Classes Date e DateFormat;
- Conversão de String em Date;
- Calendar;
- SimpleDateFormat.