Programação Orientada a Objetos

Aula: Introdução a Classes e Objetos.

Prof. Anderson Augusto Bosing

O que é um Paradigma?

O que é um Paradigma?

Um exemplo que serve como modelo ou padrão.

O que são Paradigmas da Programação?

O que são Paradigmas da Programação?

Um paradigma é um estilo de programação, um modelo, uma metodologia.
Não se trata de uma linguagem, mas a forma como você soluciona problemas usando uma determinada linguagem de programação.

Paradigmas da Programação
Procedural

Consiste em um modelo de programação que funciona como uma espécie de lista de instruções que são executadas em forma de passo a passo.

Ex:

C

Pascal

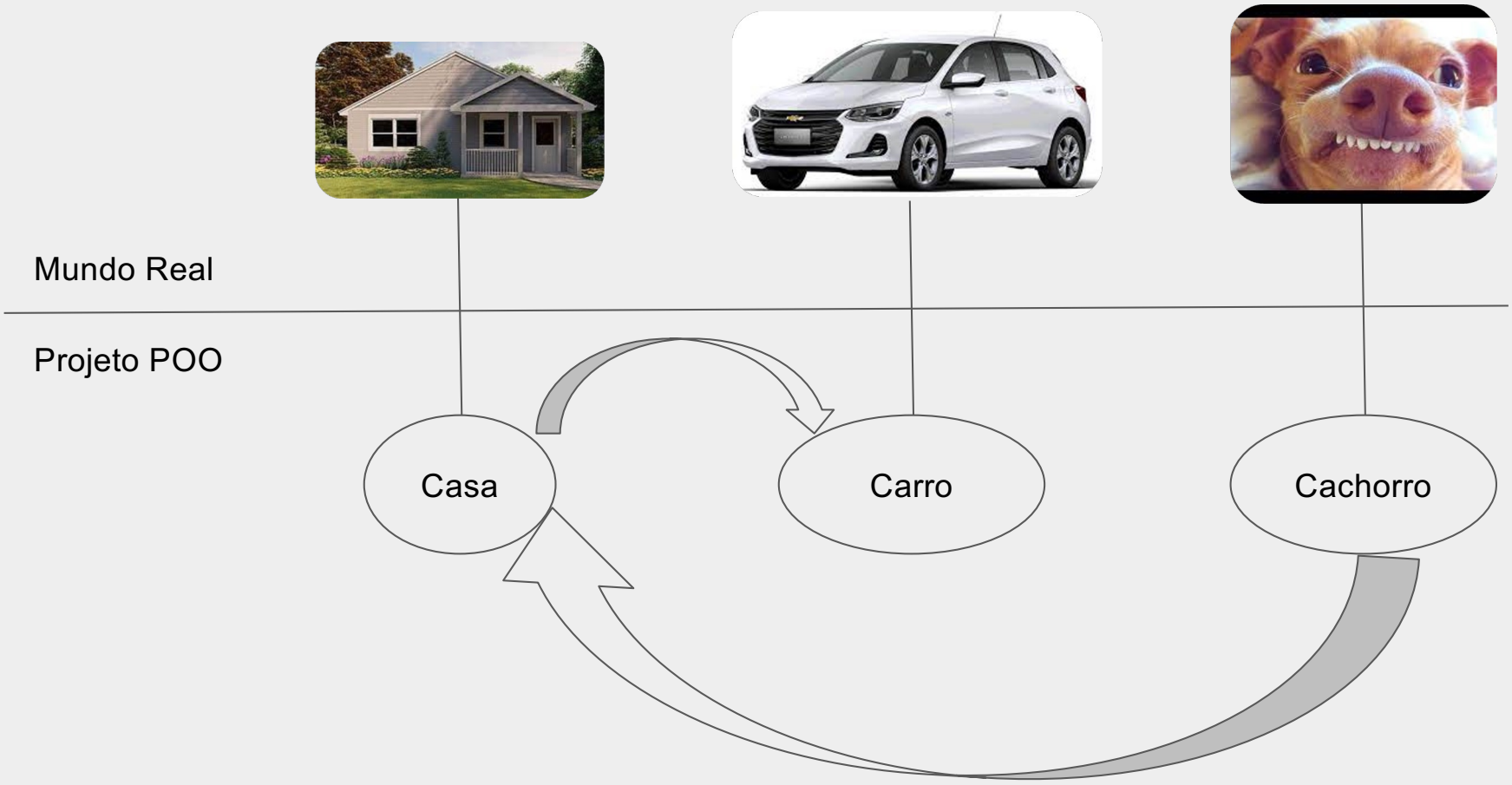
Paradigmas da Programação
POO

Este paradigma é o que mais reflete os problemas atuais. Um programa OO consistem em objetos que enviam mensagens uns para os outros. Estes objetos no programa correspondem diretamente a objetos atuais, tais como pessoas, máquinas, departamentos, documentos e assim por diante.

Ex:

Java
C#

POO



Introdução à Tecnologia de Objetos

Hoje, como a demanda por software novo e mais poderoso está aumentando, construir softwares de maneira rápida, correta e econômica continua a ser um objetivo indefinido.

Objetos ou, mais precisamente, as classes de onde os objetos são essencialmente componentes reutilizáveis de software.
Há objetos data, objetos data/hora, objetos áudio, objetos vídeo, objetos automóvel, objetos pessoas etc.

Quase qualquer substantivo pode ser razoavelmente representado como um objeto de software em termos dos atributos (por exemplo, nome, cor e tamanho) e comportamentos (por exemplo, calcular, mover e comunicar).

Introdução à Tecnologia de Objetos

Quando programamos estamos modelando aspectos do 'mundo real' utilizando uma linguagem de programação. Assim sempre temos um aspecto do 'mundo real', a representação deste aspecto no 'mundo computacional' (em tempo de execução) e a descrição deste aspecto no 'mundo linguístico' (em uma linguagem de programação).

O automóvel como um objeto

Vamos supor que você queira dirigir um carro e fazê-lo andar mais rápido pisando no pedal acelerador. O que deve acontecer antes que você possa fazer isso?

O automóvel como um objeto

Vamos supor que você queira dirigir um carro e fazê-lo andar mais rápido pisando no pedal acelerador. O que deve acontecer antes que você possa fazer isso?

Alguém precisa projetá-lo.

Criar uma especificação que vai servir como base para fabricação dos carros que vão ser utilizados nas ruas.

Considerando um software para um banco.

O que toda conta corrente tem que é importante para nós?

Considerando um software para um banco.

O que toda conta corrente tem que é importante para nós?

- **Número da conta**
- **Nome do titular da conta**
- **Saldo**
- **Limite**

Considerando um software para um banco.

O que toda conta corrente faz que é importante para nós?

O que pedimos à conta corrente?

Considerando um software para um banco.

O que toda conta corrente faz que é importante para nós?

O que pedimos à conta corrente?

- **Saca uma quantidade x;**
- **Deposita uma quantidade x;**
- **Consultar o saldo atual;**
- **Imprimir extrato.**
- **Transfere uma quantidade x para uma outra conta y;**

Considerando um software para um banco.

De acordo com o que levantamos então temos uma especificação de uma conta. Sendo ela:

Atributos de Uma conta

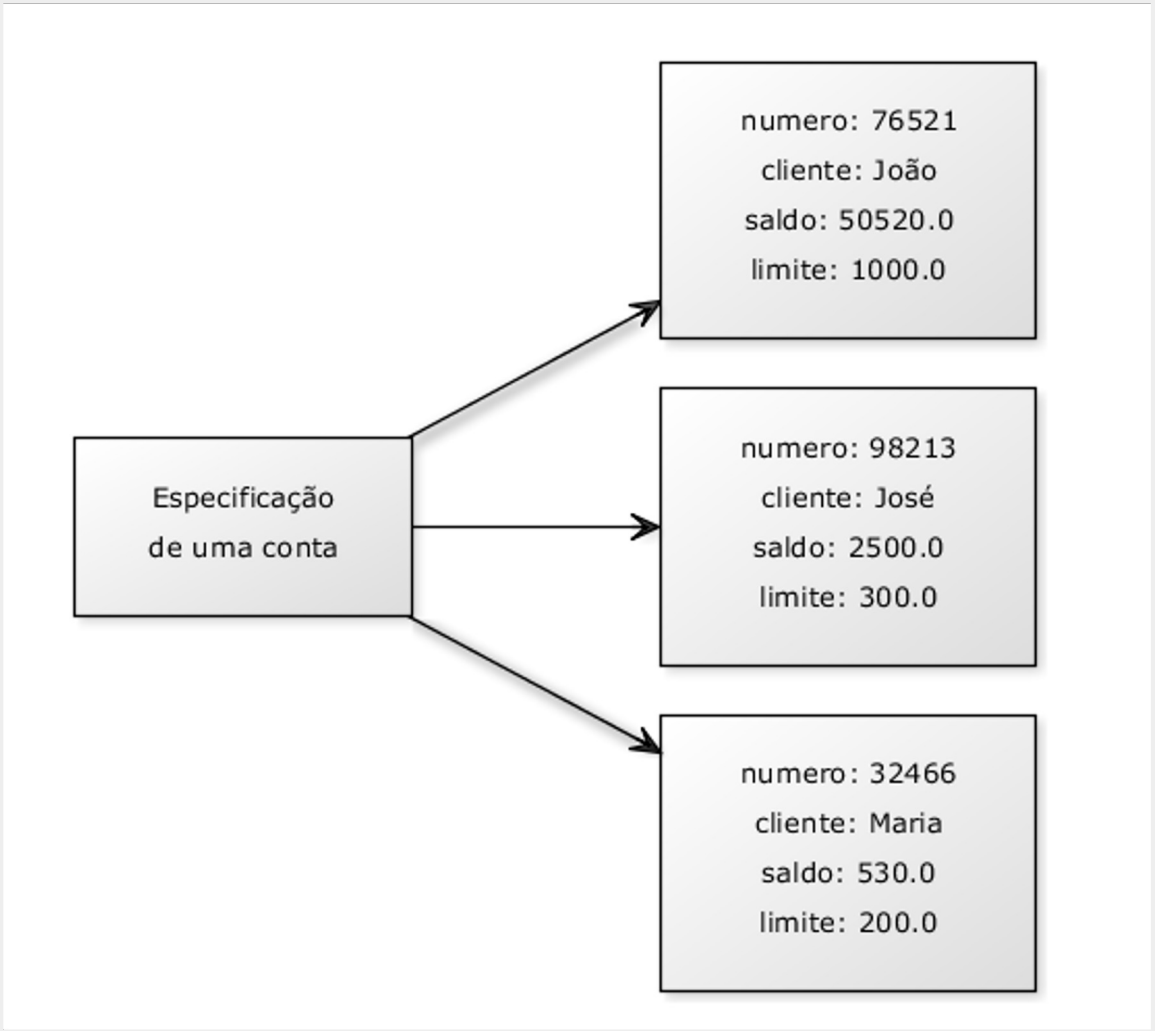
- **Número da conta**
- **Nome do titular da conta**
- **Saldo**
- **Limite**

Métodos de uma Conta

- **Saca uma quantidade x;**
- **Deposita uma quantidade x;**
- **Consultar o saldo atual;**
- **Imprimir extrato.**
- **Transfere uma quantidade x para uma outra conta y;**

Mas o que temos ainda é o projeto dela, antes de a utilizarmos precisamos construir uma conta.

Considerando um software para um banco.



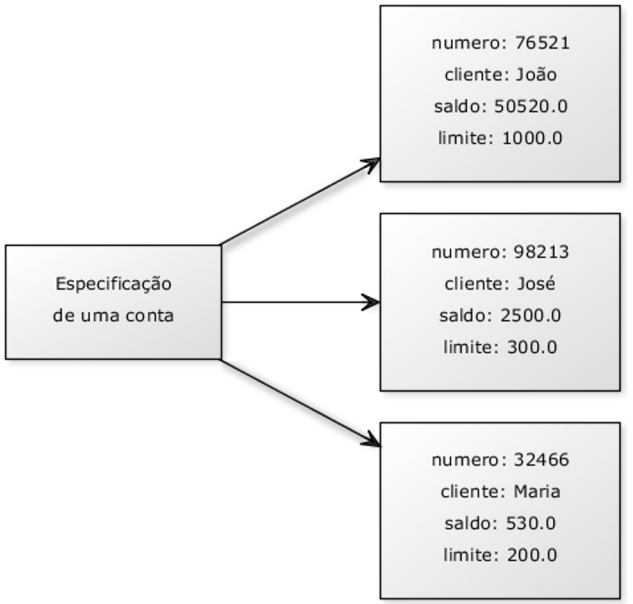
Considerando um software para um banco.

Na figura anterior podemos observar que ao lado esquerdo temos a especificação de uma conta, mas essa especificação é uma conta ?

Nós depositamos e sacamos dinheiro desse papel?

Não. Utilizamos a especificação da Conta para poder criar instâncias que realmente são contas, nas quais podemos realizar as operações que criamos.

Apesar de declararmos que toda conta tem um saldo, um número e uma agência no pedaço de, são nas instâncias desse projeto em que realmente há espaço para armazenar esses valores.

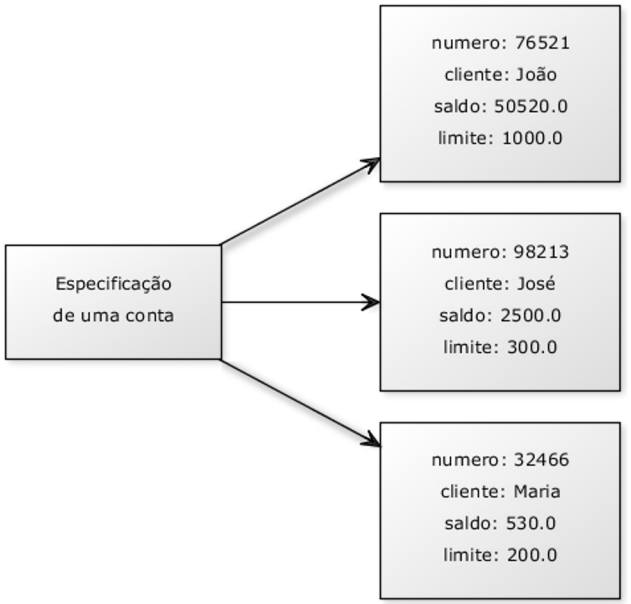


Considerando um software para um banco.

Ao projeto da conta, isto é, à especificação da conta, damos o nome de classe. Ao que podemos construir a partir desse projeto que são as contas de verdade, damos o nome de objetos.

As características da conta damos o nome de atributos(Numero, Saldo, Cliente).

E aos comportamentos desta conta damos o nome de métodos(Sacar, depositar, imprimir extrato).



Um outro exemplo: uma receita de bolo

A pergunta é certeira: você come uma receita de bolo? Não.

Precisamos instanciá-la e fazer um objeto bolo a partir dessa especificação (a classe) para utilizá-la.

Podemos criar centenas de bolos com base nessa classe (a receita, no caso). Eles podem ser bem semelhantes, alguns até idênticos, mas são objetos diferentes.

Um outro exemplo: planta de uma casa

A planta de uma casa é uma casa? Definitivamente, não.

Não podemos morar dentro da planta de uma casa nem podemos abrir sua porta ou pintar suas paredes.

Precisamos, antes, construir instâncias a partir dessa planta. Essas instâncias, sim, podemos pintar, decorar ou morar dentro.

Definições Técnicas

Classe

Assim como os objetos do mundo real, uma classe agrupa os objetos pelos seus comportamentos e atributos comuns.

Uma classe define os atributos e comportamentos comuns compartilhados por uma tipo de objeto. Os objetos de certo tipo ou classificação compartilham os mesmos comportamentos e atributos.

Definições Técnicas

Objeto

Um objeto é uma instância de uma classe.

Atributo

Atributos são as características de uma classe visíveis externamente. A cor de um carro e o seu modelo são exemplos de atributos.

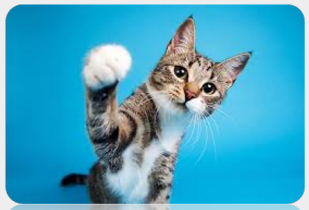
Método

Métodos definem as habilidades dos objetos

Vamos codificar.

- Criar uma classe câmera.

- Criar uma classe gato.



NetBeans

Vamos Praticar?

Lista 1 - Exercicios Orientação a Objetos com Java.

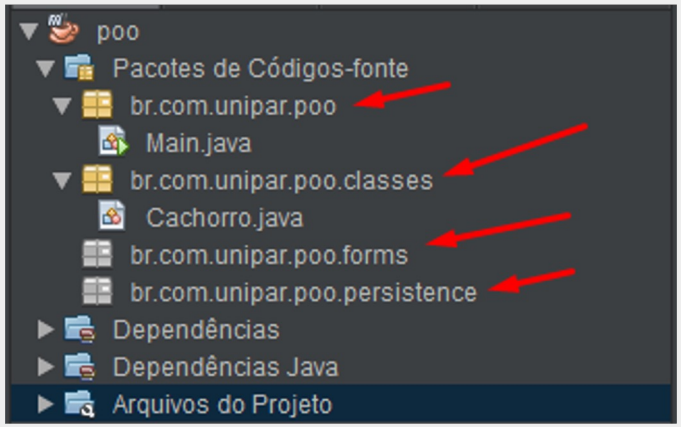
Programação Orientada a Objetos

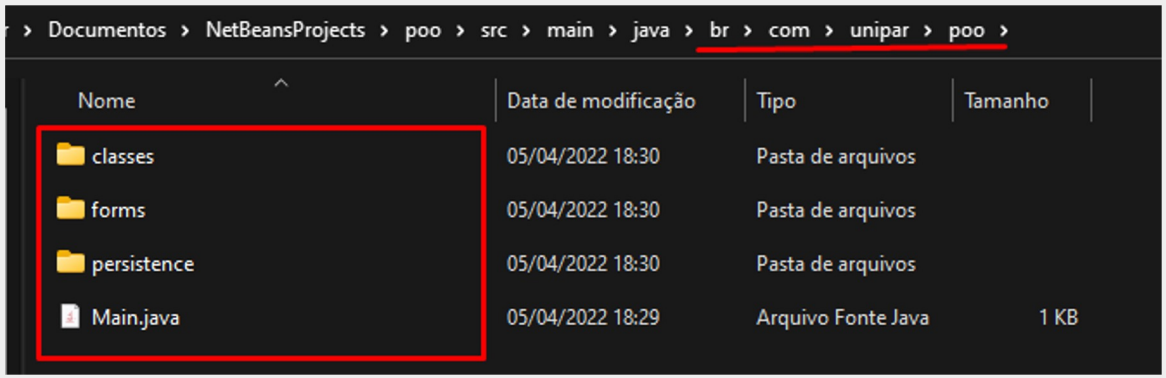
Aula: Modificadores de acesso,
Métodos Acessores.

Prof. Anderson Augusto Bosing

Pacotes

Pacotes java são utilizados para organizar as classes da sua aplicação. Um programa pode, facilmente, ter mais de centenas de classes. Então é muito importante que todos os seus componentes fiquem organizados. Podemos pensar nos pacotes como uma pasta do seu sistema de arquivos.





Modificadores de Acesso a Nível de Classe - public

Modificador public torna uma classe visível:

Para qualquer outra classe e em qualquer pacote.

```
 */
public class Cachorro {

}
```

Modificadores de Acesso a Nível de Classe - default

Modificador vazio ou default torna uma classe visível:

Torna uma classe visível apenas para classes do mesmo pacote.

```
 */
class Cachorro {

}
```

Modificadores de Acesso a Nível de Atributos e Métodos(Membros) - public

public torna um membro acessível:

Em qualquer lugar e a qualquer outra classe que possa visualizar a classe que contém o membro.

```
public class Cachorro {

    public String nome;

}
```

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        Cachorro dog = new Cachorro();
        dog.nome = "bob";


    }

}
```

Modificadores de Acesso a Nível de Atributos e Métodos(Membros) - protected

protected torna um membro acessível às classes:

Do mesmo pacote.

Os membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados.

```
*/
public class Cachorro {

    protected String nome;

}
```

```
public static void main(String[] args) {

    Cachorro dog = new Cachorro();
    dog.nome = "bob";

}
```

```
*/
public sta
    Cachor
    dog.nome = "bob";

}
```

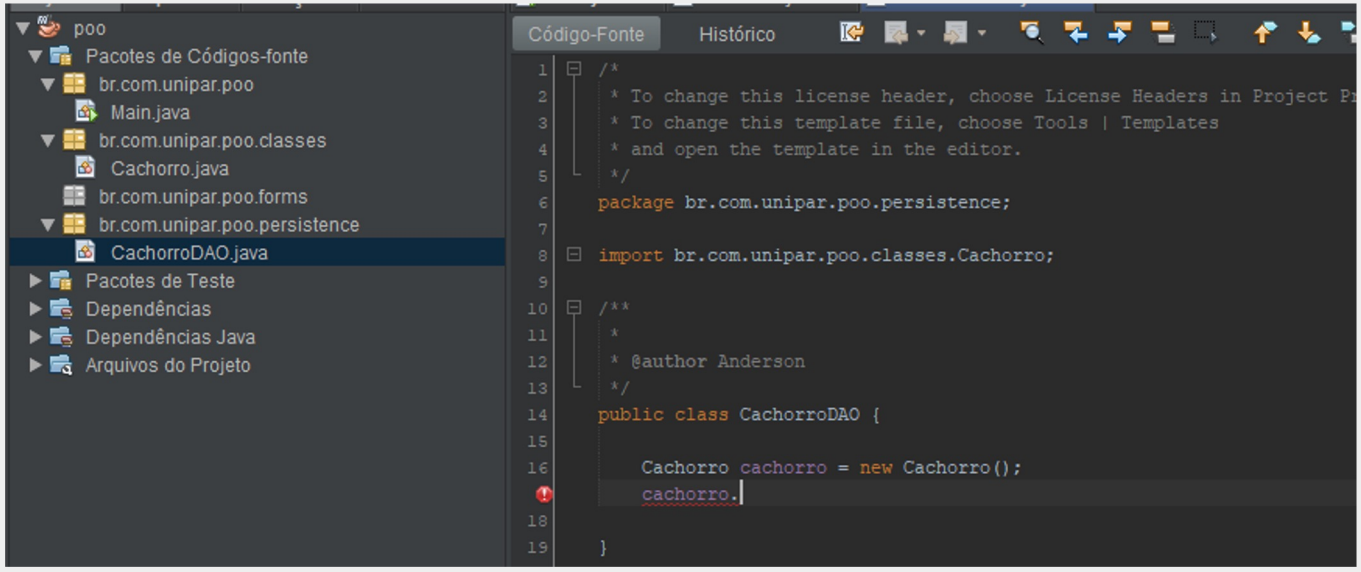
nome has protected access in Cachorro

(Alt-Enter mostra dicas)

Modificadores de Acesso a Nível de Atributos e Métodos(Membros) - default

default (sem modificador explícito) torna um membro acessível apenas para classes do mesmo pacote.

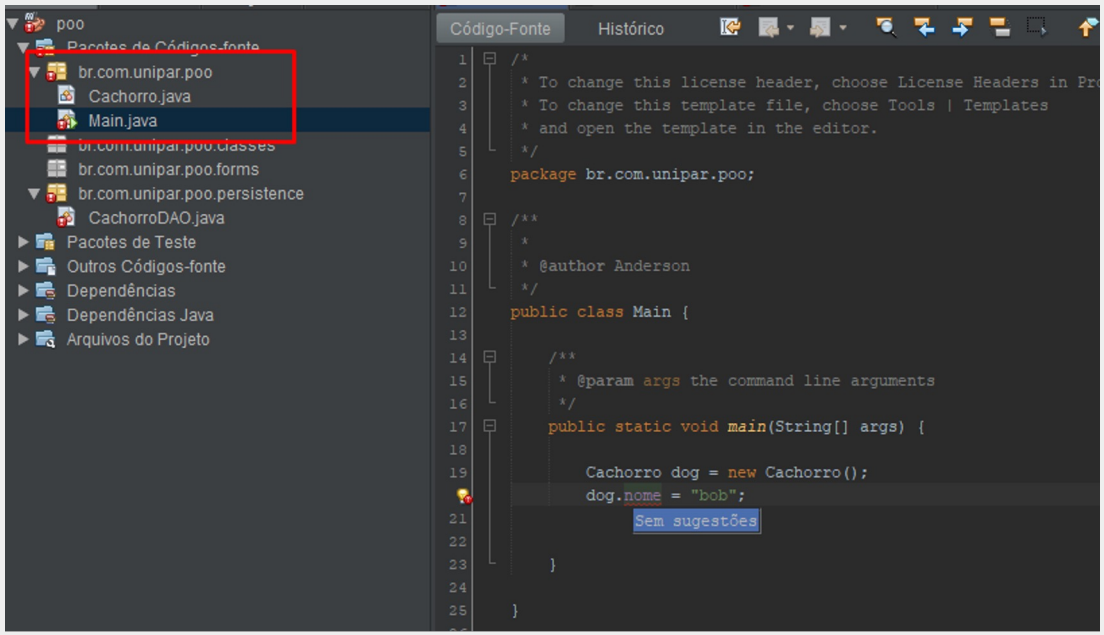
```
 */
public class Cachorro {

    String nome;

}
```



Modificadores de Acesso a Nível de Atributos e Métodos(Membros) - private

private torna um membro acessível apenas para a classe que o contém.

```
public class Cachorro {

    private String nome;

}
```



Objetos não devem alterar os estados dos outros ou seus atributos.

Exemplos:

Pessoa e um Carro.

Pessoa e um Porta.

Professor, mas se um objeto não pode alterar diretamente os atributos e o estado de outro objeto, como fazemos nosso código abaixo?

```
 */
public class Cachorro {

    String nome;

}
```

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        Cachorro dog = new Cachorro();
        dog.nome = "bob";


    }

}
```

Professor, mas se um objeto não pode alterar diretamente os atributos e o estado de outro objeto, como fazemos nosso código abaixo?

Através dos métodos acessores.

Métodos Acessores(Get e Set)

Também conhecidos como getter e setters, são métodos utilizados para que seja possível acessar e modificar os valores de variáveis com modificador de acesso private.

Getters

O propósito de um getter é obter o valor de uma variável declarada como private e permitir sua leitura a partir de outra classe.

Setters

Um setter é um método que permite modificar o valor de um atributo da classe que não seja acessível diretamente por ser privado.

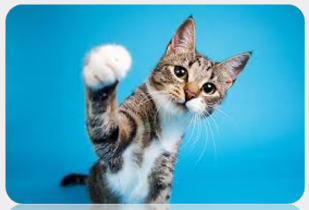
Getters e Setters

```
 */
public class Cachorro {
   //Atributo privado
    private String nome;

    //No caso do setter o retorno do metodo é void ou seja sem retorno pois
    //usamos o metodo para setar o valor a um atributo privado
    //Acessor do metodo + Tipo de Retorno + nomeMetodo(Tipo e Parametro de Entrada)
    public void setNome(String nome) {
        this.nome = nome;//this identifica o contexto do que está sendo setado
                         //nesse caso o nosso contexto é a própria classe
                         //fazendo com que this.nome seja diferente da variavel de entrada nome
    }

    //No caso do getter como apenas queremos buscar o valor do atributo privado não temos
    //parametros de entrada no metodo mas temos o tipo de retorno
    //nesse caso como o atributo se trata de uma string
    //o tipo de retorno é uma string
    //Acessor do metodo + Tipo de Retorno + nomeMetodo()
    public String getNome() {
        return nome; //return é um comando reservado que identifica no java qual será o retorno do metodo
    }

}
```

Vamos codificar.

- Criar uma classe câmera.

- Criar uma classe gato.



NetBeans

Vamos Praticar?

- **Lista 2 - Exercicios Orientação a Objetos com Java.docx**

Programação Orientada a Objetos

Aula: Encapsulamento, Construtores e Destrutores.

Prof. Anderson Augusto Bosing

Encapsulamento

Classes (e seus objetos) encapsulam, isto é, contêm seus atributos e métodos. Os atributos e métodos de uma classe (e de seu objeto) estão intimamente relacionados.

Os objetos podem se comunicar entre si, mas eles em geral não sabem como outros objetos são implementados — os detalhes de implementação permanecem ocultos dentro dos próprios objetos. (Deitel, 2016).

Encapsulamento

Encapsular é tornar o código dentro de uma classe acessível ou inacessível para objetos fora da classe. A lógica que suporta este comportamento é que cada classe deve ter um significado e por si só deve descrever o comportamento de um objeto.

A palavra chave this

Palavra reservada do Java que referencia atributos e métodos da própria classe.

Objetos e Mais Objetos

É comum que sejam criadas classes para representar objetos do mundo real, e tão comum quanto isso são que os atributos das classes também seja identificados como outras classes.

Vamos a um exemplo.

Se nós temos uma classe que representa um carro, esse carro tem uma marca que tem seus próprios atributos. E essa marca consequentemente pode possuir um endereço que também possui seus próprios atributos.

Mas e professor, como isso seria codificado?

Objetos e Mais Objetos

Vamos a outro exemplo.

Um banco possui muitas agencias e estas agencias possuem muitos correntistas, sendo contas correntes ou contas poupanças. Cada correntista possui um endereço para onde deve ser enviado o cartão.

Mas e professor, como isso seria codificado?

Por que Utilizar o Encapsulamento?

Organização de código.

Quando criamos classes, cada uma delas possui uma função específica. Ou seja, elas descrevem um objeto específico, sendo assim, classes coesas não realizam várias funções.

Manutenção de código.

- **Depois de pronto, todo código, principalmente os mais extensos, são propensos a sofrer manutenções.**
- **Com o encapsulamento, isso passa a ser mais fácil, uma vez que, com a devida proteção de acesso aos dados, a pessoa programadora achará mais rápido algum ponto onde o código precisa ser melhorado.**

Por que utilizar o Encapsulamento?

Reuso de Código.

Com o encapsulamento, o programa terá mais chances de ter o código reaproveitado em outros projetos, poupando bastante tempo da equipe de desenvolvimento.

Simplificação da codificação.

O encapsulamento transforma a implementação de alguns códigos em uma espécie de caixa preta. Na prática, isso significa que as classes externas não precisam acessar alguns dados de forma direta. Assim, o desenvolvimento dos sistemas passa a ficar simplificado e acelerado.