SENAI CIMATEC

SISTEMA FIEB

TECNOLOGIA E INOVAÇÃO EM PROL DA INDÚSTRIA

SENAI CIMATEC — SISTEMA FIEB

PROAJ – Desenvolvedor Web com JAVA

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Desenvolvimento de Sistemas I – 100h

Profª: Francisleide Almeida

Modificadores de acesso

- Até agora temos realizado operações no método principal que tem gerado algumas “inconsistências”.
- Observe o exemplo abaixo:

```
minhaConta.saldo = 900.0;
minhaConta.limite = 100;
minhaConta.sacaValor(500);
minhaConta.exibeSaldo();
minhaConta.saldo = 30.0;
```

Modificadores de acesso

- Até agora temos realizado operações no método principal que tem gerado algumas “inconsistências”.
- Observe o exemplo abaixo:

```
minhaConta.saldo = 900.0;
minhaConta.limite = 100;
minhaConta.sacaValor(500);
minhaConta.exibeSaldo();
minhaConta.saldo = 30.0;
```

Ok, tenho 900 reais

Modificadores de acesso

- Até agora temos realizado operações no método principal que tem gerado algumas “inconsistências”.
- Observe o exemplo abaixo:

```
minhaConta.saldo = 900.0;
minhaConta.limite = 100;
minhaConta.sacaValor(500);
minhaConta.exibeSaldo();
minhaConta.saldo = 30.0;
```

Beleza, saquei 500...

Modificadores de acesso

- Até agora temos realizado operações no método principal que tem gerado algumas “inconsistências”.
- Observe o exemplo abaixo:

```
minhaConta.saldo = 900.0;
minhaConta.limite = 100;
minhaConta.sacaValor(500);
minhaConta.exibeSaldo();
minhaConta.saldo = 30.0;
```

Exibirá 400 reais

Modificadores de acesso

- Até agora temos realizado operações no método principal que tem gerado algumas “inconsistências”.
- Observe o exemplo abaixo:

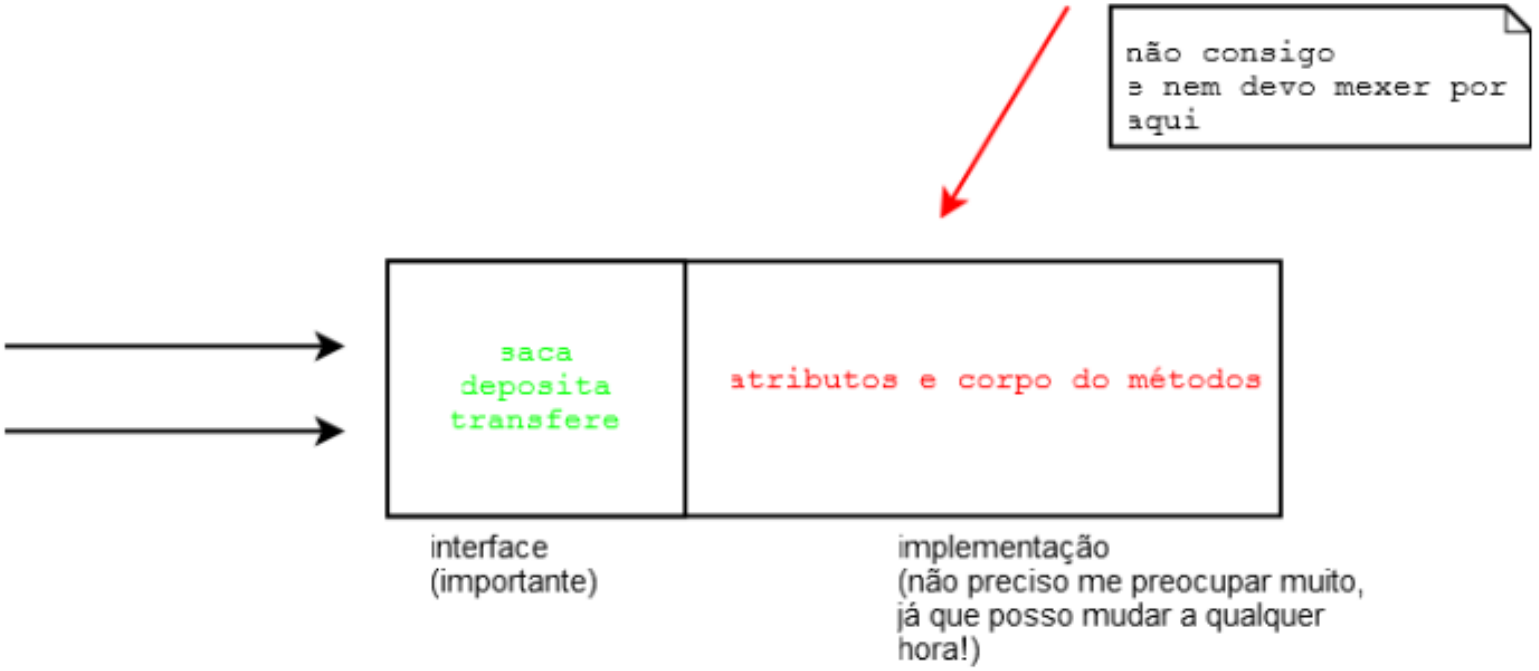
```
minhaConta.saldo = 900.0;
minhaConta.limite = 100;
minhaConta.sacaValor(500);
minhaConta.exibeSaldo();
minhaConta.saldo = 30.0;
```

Só tenho 30??

Resumindo...

- Não podemos deixar com que os atributos da nossa classe fiquem a mercê da atribuição de qualquer outra classe;
- Vamos criar modificadores de acesso para os atributos da classe;
- Modificações vão ser possíveis apenas através de métodos!

Encapsulamento

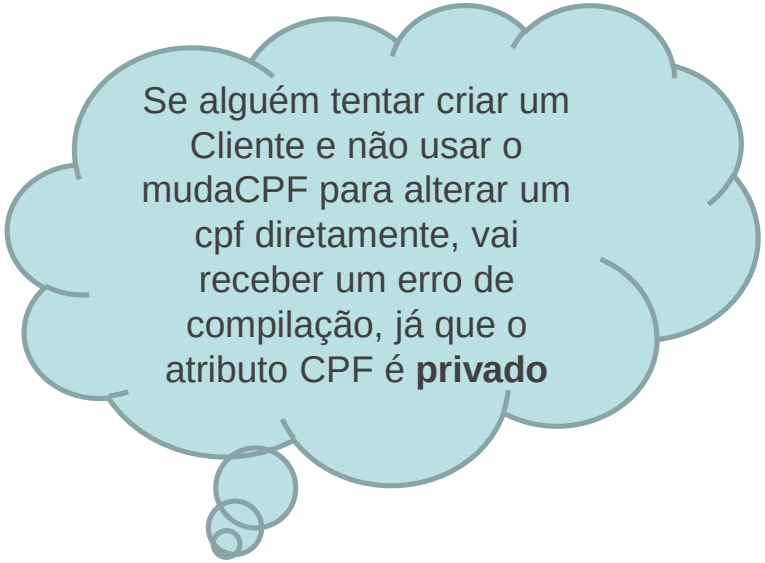


- O conjunto de métodos públicos de uma classe é também chamado de **interface da classe**, pois esta é a única maneira a qual você se comunica com objetos dessa classe.

Ex1:

```
class Cliente {
    private String nome;
    private String endereco;
    private String cpf;
    private int idade;

public boolean mudaCPF(String cpf) {
    boolean verifica = validaCPF(cpf);
    if(verifica){
            this.cpf = cpf;
    } return false;
}
```



Significado de cada Modificador

- **Public**
  - Pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.
- **Private**
  - Não podem ser acessados ou usados por nenhuma outra classe.
- **Protected**
  - torna o membro acessível às classes do mesmo pacote ou através de herança, seus membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados
- **default**
  - A classe e/ou seus membros são acessíveis somente por classes do mesmo pacote, na sua declaração não é definido nenhum tipo de modificador, sendo este identificado pelo compilador.

GETTERS E SETTERS

- O modificador private faz com que ninguém consiga modificar, nem mesmo ler, o atributo em questão;

Mas, e agora? Como fazer para mostrar o saldo de uma Conta, já que nem mesmo podemos acessá-lo para leitura?

- Usando métodos... Ex:

```
class Conta {
        private double saldo;

        public double getLimite() {
                return this.limite;
        }
}
```

```
class Conta {
        private double saldo;

        public void setLimite(double limite) {
                this.limite = limite;
        }
}
```

Atividade

- Criar um programa que armazene a estrutura de casas, contendo:
  - Dono;
  - Número;
  - Quantidade de moradores;

Com atributos privados e crie métodos de acesso getters e setters para cada um destes atributos. Em seguida crie um objeto solicitando tais dados para o usuário na tela e exiba-os em seguida.

Construtores

- Quem cria o objeto da classe;
- Possui o mesmo nome da classe;
- Não é um método.

Ex:

```
class Cliente {
    private String nome;
    private String endereco;
    private String cpf;
    private int idade;
Conta() {
        System.out.println("Construindo uma
    conta.");
 }
```

Construtor com parâmetro

- Conta(double limite) {

  this.limite = limite;

}

No momento de instanciar, torna-se obrigatório a passagem de um valor para este parâmetro:

Conta conta1 = new Conta(5000);

Mais de um construtor

- Você pode ter mais de um construtor, contato que ele não possua a mesma assinatura;
- Para voltar a instanciar um objeto sem parâmetros, depois de já ter criado o construtor passando parâmetros, é preciso criar um construtor vazio (o criado automaticamente, deixará de existir, quando criado um novo)

Mais de um construtor

```
public class Cachorro {
    private int idade;
    private String nome;
    private String raca;
    private String cor;
    private String porte;
    private boolean castrado;
    private char sexo;
    private boolean doado;

    public Cachorro(){


}
    public Cachorro(String raca, int idade){
        this.raca = raca;
        this.idade = idade;
    }
    public Cachorro(int idade, char sexo){
        this.idade = idade;
        this.sexo = sexo;
    }
```

Construtor vazio

Construtores com assinaturas diferentes

Atividade

- Crie um classe Celular que possua atributos de preço, modelo, marca, Gigas de armazenamento. Deixe todos os atributos privados e crie os getters e setters para cada uma deles. Crie também dois construtores: um que seja necessário para criação do objeto possuir modelo, marca e gigas e outro que seja necessário todos atributos para instanciar um novo celular, pois um será usado apenas pelo fabricante (não precisa atribuir preço) e outro será usado pela loja.