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

Construtores, Modificadores de Acesso e Encapsulamento

Prof: Diego Corrêa

Construtores

Construtor

- Utilizado para definir informações mínimas para a construção/instanciação dos objetos
- Quais informações são essenciais e sem elas o objeto/sistema fica inconsistente?
- Um classe pode ter diversos construtores
- São “métodos” executados durante a instanciação de um objeto (new)

Construtor Exemplo

```
class Conta {
    int numero;
    String titular;
    double saldo;
    double limite;
    double limiteFixo;
    Conta() {
            System.out.println("Construtor da conta");
    }
}
```

Construtor com parâmetro

No momento de instanciar um construtor também pode receber parâmetros

```
Conta(double limite) {
    this.limite = limite;
}
```

Torna-se obrigatório a passagem de um valor para este parâmetro:

```
Conta conta1 = new Conta(5000);
```

Sobrecarga Construtor

- Basicamente cria variações de um mesmo método
- A criação de dois ou mais métodos com nomes totalmente iguais em uma classe
- A Sobrecarga permite que utilizemos o mesmo nome em mais de um método
- Contanto que suas listas de argumentos sejam diferentes

Sobrecarga Exemplo

```
class Conta {
        int numero;
        String titular;
        double saldo;
        double limite;
        double limiteFixo;
    int tipoConta; // [0 - 3] Estudante, Trabalhador, Especial,
        Ilimitado
    Conta() {
            System.out.println("Construtor da conta");
    }
}
```

Sobrecarga Exemplo

```
Conta(int numero, String titular, double saldo, double limiteFixo) {
    this.numero = numero;
    this.titular = titular;
    this.saldo = saldo;
    this.limite = limiteFixo;
    this.limiteFixo = limiteFixo;
    this.tipoConta = 0;
}

Conta(int numero, String titular, double saldo,double limiteFixo, int tipoConta) {
    this.numero = numero;
    this.titular = titular;
    this.saldo = saldo;
    this.limite = limiteFixo;
    this.limiteFixo = limiteFixo;
    this.tipoConta = tipoConta;
}

Conta(int numero, String titular, double saldo) {
    this.titular = titular;
    this.saldo = saldo;
    this.limite = 500.0;
    this.limiteFixo = 500.0;
    this.tipoConta = 0;
}
```

Sobrecarga
Exemplo

```
Conta conta_sem_nada = new Conta();
conta_sem_nada.exibeSaldo();
Conta conta_estudante = new Conta(111, "Cleber", 100.00);
conta_estudante.exibeSaldo();
Conta conta_especial = new Conta(222, "Suzana", 500000.00, 1000000.00, 2);
conta_especial.exibeSaldo();
```

Modificadores de Acesso

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;
```

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!

Public

A instrução public indica que a classe, método ou variável assim declarada possa ser acessada em qualquer lugar e a qualquer momento da execução do programa

No UML sinal de adição (“+”) no atributo, método ou classe deve ser codificado como **public**

Private

O modificador de acesso "private" quando aplicado a um atributo ou a um método indica que os mesmos só podem ser acessados de dentro da classe que os criou (encapsulamento)

No UML sinal de subtração (“-”) no atributo ou método deve ser codificado como **private**

Protect

Quando um membro da classe é declarado assim, ele se torna acessível por classes do mesmo pacote ou através de herança. Os membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados

No UML sinal de jogo da velha (“#”) no atributo ou método deve ser codificado como **protect**

Default

A classe e/ou seus membros são acessíveis somente por classes do mesmo pacote

Static

Na declaração de uma variável dentro de uma classe, para se criar uma variável que poderá ser acessada por todas as instâncias de objetos desta classe como um variável comum

No UML o sublinhado (“_”) no atributo ou método deve ser codificado como **static**

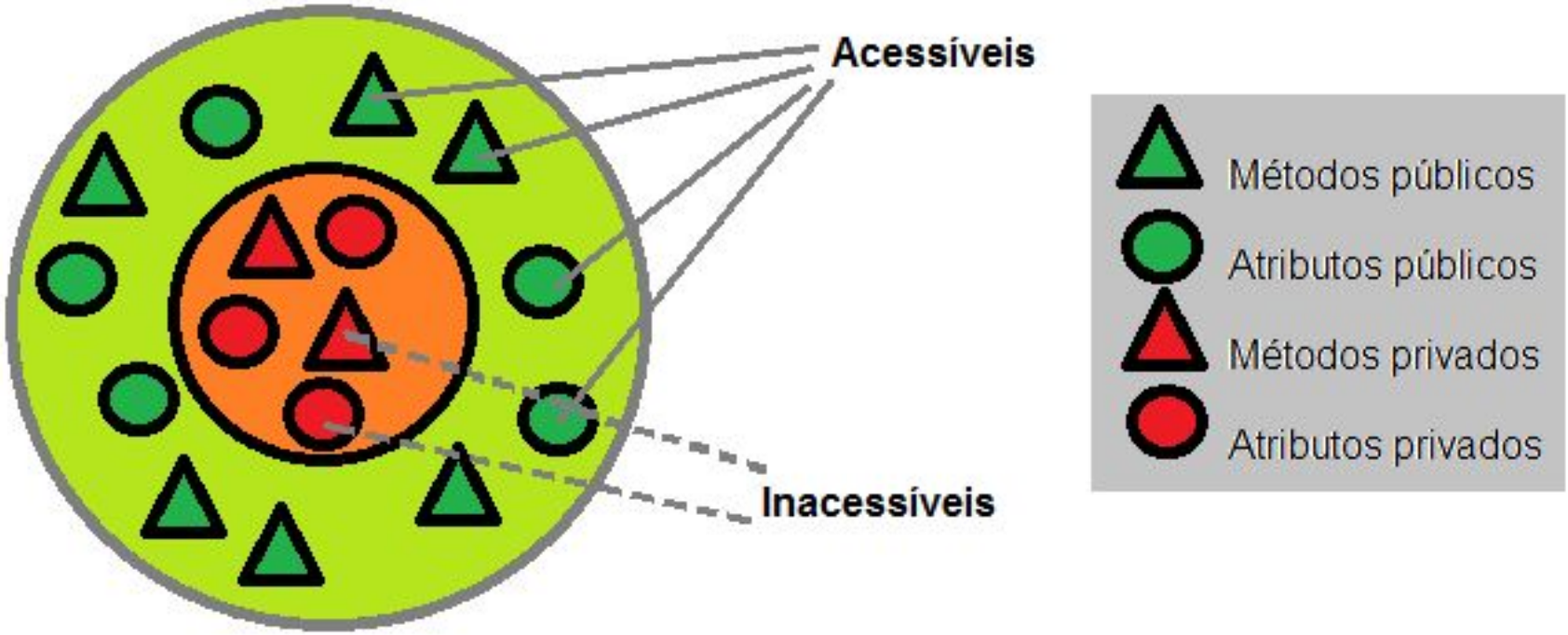
Final

A instrução final indica que a classe, método ou variável assim declarada têm uma única atribuição que se mantém constante, ou seja, não pode ser alterada no decorrer do processamento

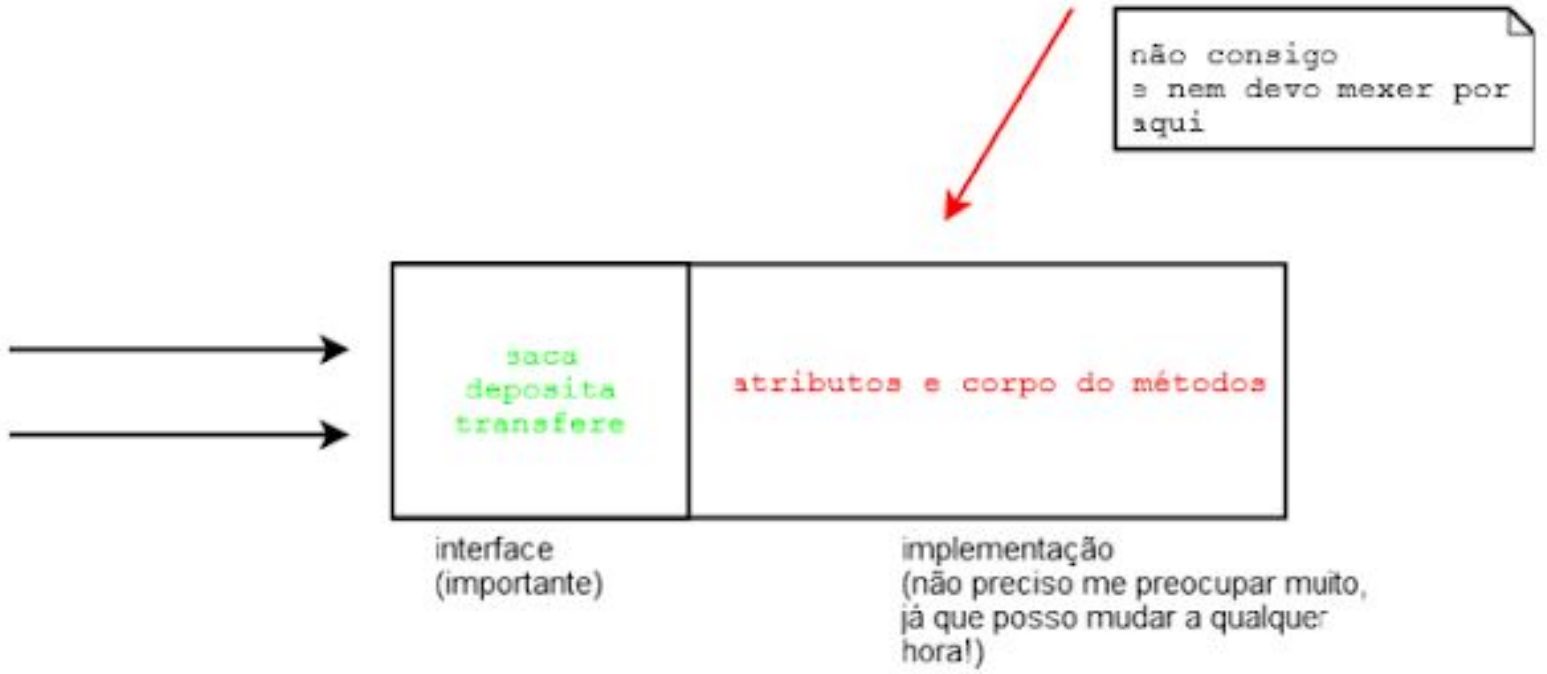
Este modificador declara o que chamamos, em programação, de constante

No UML o atributo escrito em CAIXA ALTA deve ser codificado como **final**

Encapsulamento



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.

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Encapsulamento

- Separar o programa em partes, o mais isoladas possível

- Uma grande vantagem do encapsulamento é que toda parte encapsulada pode ser modificada sem que os usuários da classe em questão sejam afetados

Get e Set

- 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?

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Modificadores

```
public class Conta {
    private final int NUMERO;
    private final String TITULAR;
    private double saldo;
    private double limite;
    private double limiteFixo;
    private int tipoConta; // [0 - 3] Estudante, Trabalhador, Especial, Ilimitado
```

Construtores

```java
Conta(int numero, String titular, double saldo) {
    this.NUMERO = numero;
    this.TITULAR = titular;
    this.saldo = saldo;
    this.limite = 0.0;
    this.limiteFixo = 0.0;
    this.tipoConta = 1;
}

Conta(int numero, String titular, double saldo, double limiteFixo) {
    this.NUMERO = numero;
    this.TITULAR = titular;
    this.saldo = saldo;
    this.limite = limiteFixo;
    this.limiteFixo = limiteFixo;
    this.tipoConta = 1; // Trabalhador
}

Conta(int numero, String titular, double saldo,double limiteFixo, int tipoConta)
    this.NUMERO = numero;
    this.TITULAR = titular;
    this.saldo = saldo;
    this.limite = limiteFixo;
    this.limiteFixo = limiteFixo;
    this.tipoConta = tipoConta;
}
```

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Encapsulamento Get e Set

```
public String getTitular(){
    return this.TITULAR;
}

public int getNumero(){
    return this.NUMERO;
}

public double getSaldo() {
    return this.saldo;
}

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

public double getLimiteFixo() {
    return this.limiteFixo;
}

public int getTipoConta() {
    return this.tipoConta;
}
```

Atividade

- Implementar a classe conta do diagrama a seguir:
  - Saque deve retirar do saldo e depois do limite
  - Deposito deve cobrir primeiro o limite depois o saldo
  - O mesmo citado acima serve para a transferencia
  - Alterar o limite deve considerar o valor prévio do limite na conta e adaptar o valor ao novo limite

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Atividade