Centro Tecnológico
Departamento de Informática

Prof. Vítor E. Silva Souza
http://www.inf.ufes.br/~vitorsouza

[Desenvolvimento OO com Java]

Orientação a objetos básica

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Conteúdo do curso

- O que é Java;
- Variáveis primitivas e controle de fluxo;
-  Orientação a objetos básica;
- Um pouco de vetores;
- Modificadores de acesso e atributos de classe;
- Herança, reescrita e polimorfismo;
- Classes abstratas e interfaces;
- Exceções e controle de erros;
- Organizando suas classes;
- Utilitários da API Java.

Estes slides foram baseados na apostila do curso FJ-11: Java e Orientação a Objetos da Caelum e na apostila Programação Orientada a Objetos em Java do prof. Flávio Miguel Varejão.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Questão de responsabilidade

- Imagine um sistema...
  - *Dezenas de formulários;*
  - *CPFs são validados – função `validar(cpf)` deve ser chamada em cada formulário;*
  - *Todos os desenvolvedores são responsáveis!*
  - *Define-se que idade deve ser >= 18. Validação simples (um if), mas onde adicioná-la?*
- Procedural: responsabilidade espalhada;
- OO: responsabilidade concentrada, polimorfismo.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Classes e objetos

- Desde Aristóteles que o ser humano classifica os objetos do mundo;
- Juntamos objetos com mesmas características em categorias que chamamos de “classes”:
  - *Todas as contas de banco tem um saldo, mas cada conta pode ter um saldo diferente;*
  - *Todas as contas de banco podem sofrer depósitos ou serem encerradas.*
- Classes são usadas por linguagens OO para modelar tipos compostos. São modelos abstratos que definem os objetos da classe.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Classes e objetos

Classes são projetos / especificações	Objetos são instâncias de verdade
Homo Sapiens	Um ser humano
Receita de bolo	Um bolo feito com a receita
Planta de uma casa	Uma casa construída a partir da planta

Definem um conjunto de características e comportamentos comuns.

Possuem valores para as características (olhos verdes, calda de chocolate, cor azul) e podem realizar o comportamento (correr, alimentar, abrir a porta)

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Definição de uma classe

- Uso da palavra reservada `class`;
- Significado: “segue abaixo a especificação de como objetos deste tipo devem se comportar”;

```
class NomeDaClasse {
  /* Especificação da classe vai aqui. */
}
```

- Depois de definida a classe, podemos definir variáveis (referências) e criar objetos:

```
NomeDaClasse obj = new NomeDaClasse();
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Criação de objetos

- Objetos são criados com o operador new:
  - *Cria o objeto na memória (monte/heap);*
  - *Retorna uma referência ao objeto criado.*
- Construtores:
  - *Métodos especiais que executam durante a criação;*
  - *Podem especificar valores iniciais aos atributos.*
- Inicialização:
  - *Atributos são "zerados" quando um objeto é construído;*
  - *Podem também ser declarados com valores iniciais.*

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Destruição de objetos

- Um objeto é alocado dinamicamente;
- Qual é o tempo de vida de um objeto?
- Antes: qual é o tempo de vida de uma variável?

```
{
  int x = 12;
  // x está disponível.
  {
    int q = 96;
    // x e q estão disponíveis.
  }
  // x está disponível, q fora de escopo.
}
// x e q fora de escopo.
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Destruição de objetos

- Uma variável é destruída quando acaba seu escopo (funcionamento da pilha);
- E quando um objeto é destruído?

```
{
  String s;
  // referência s disponível.
  {
    String r = new String("Olá!");
    // s, r, objeto disponíveis.
    s = r;
  }
  // r não existe mais. Destruir o objeto?
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Destruição de objetos

- Ao contrário das variáveis, um objeto não é destruído quando o escopo acaba;
- Um objeto é destruído automaticamente pelo Coletor de Lixo (*Garbage Collector* – *GC*) quando ele se torna inacessível;
- Um objeto é inacessível quando não há referências (diretas ou indiretas) para ele na pilha.

```
{
  String s = new String("Olá!");
}
// Impossível acessar a string "Olá!"
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Membros da classe

- Uma classe pode ter dois tipos de membro:
  - *Variáveis (em jargão OO: "atributos");*
  - *Funções (em jargão OO: "métodos").*
- Atributos são como partes de um tipo composto;
- Métodos são funções que são executadas no contexto de uma classe/objeto.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Atributos

- Definidos como variáveis no escopo da classe:

```
class Conta {
  int numero;
  String dono;
  double saldo;
  double limite;
  // ...
}
```

Repare que as variáveis não são declaradas dentro de um bloco de função, mas diretamente no bloco da classe.

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Atributos

- Acesso via operador de seleção (“.”):

```
public class Programa {
  public static void main(String[] args) {
    Conta minhaConta;
    minhaConta = new Conta();

    minhaConta.dono = "Duke";
    minhaConta.saldo = 1000.0;

    System.out.println("Saldo: " + minhaConta.saldo);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Métodos

- Um método é uma função que opera no contexto de uma classe (mensagem que o objeto recebe);
- É a maneira (método) de se fazer algo num objeto:

```
class Conta {
  // Atributos já declarados...
  void sacar(double qtd) {
    double novoSaldo = this.saldo - qtd;
    this.saldo = novoSaldo;
  }
  void depositar(double qtd) {
    this.saldo += qtd;
  }
}
```

Métodos

Não retorna valor.

Argumento(s)

Variável local

```
class Conta {
  // Atributos já declarados...

  void sacar(double qtd) {
    double novoSaldo = this.saldo - qtd;
    this.saldo = novoSaldo;
  }

  void depositar(double qtd) {
    this.saldo += qtd;
  }
}
```

Atributo (neste caso, `this` é opcional)

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Métodos

- E/Invocação também via operador de seleção (“.”):

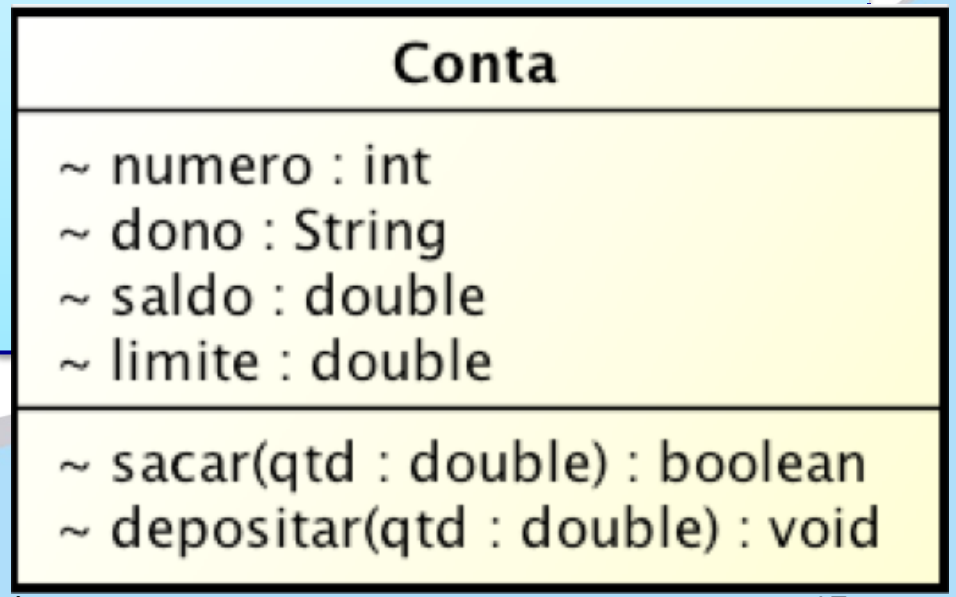
```
public class Programa {
  public static void main(String[] args) {
    Conta minhaConta = new Conta();
    minhaConta.dono = "Duke";
    minhaConta.saldo = 1000;

    minhaConta.sacar(200);
    minhaConta.depositar(500);

    // Saldo: 1300.0
    System.out.println("Saldo: " + minhaConta.saldo);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Métodos

- Um método pode retornar um valor:

```
class Conta {
  // Atributos já declarados...

  boolean sacar(double qtd) {
    if (saldo < qtd) return false;

    saldo = saldo - qtd;
    return true;
  }
}
```



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

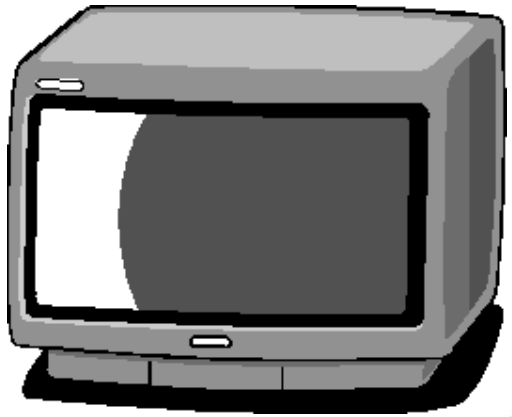
Uma classe, múltiplas instâncias

- Podemos criar quantos objetos quisermos...

```
public class Programa {
  public static void main(String[] args) {
    Conta minhaConta = new Conta();
    minhaConta.saldo = 1000;
    if (minhaConta.sacar(2000))
      System.out.println("Consegui");
    else System.out.println("Não deu...");

    Conta meuSonho = new Conta();
    meuSonho.saldo = 1_000_000_000.0;

    // Usando a mesma referência.
    minhaConta = new Conta();
  }
}
```

Manipulação de objetos em Java

- Em Java trabalhamos com referências para objetos, ao contrário de C++ (manipulação direta ou ponteiros);
- Analogia:
  - *A TV é o objeto;*
  - *O controle é a referência;*
  - *Você só carrega a referência;*
  - *A referência pode existir sem o objeto.*



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Referência e objeto

```
public class Coordenadas {
  int x;
  int y;
  int z;

  public static void main(String[] args) {
    // Só a referência. Não dá pra fazer nada...
    Coordenadas coord;

    // Agora temos um objeto, podemos usá-lo.
    coord = new Coordenadas();
    coord.x = 10;
    coord.y = 15;
    coord.z = 18;
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Atribuição de valores

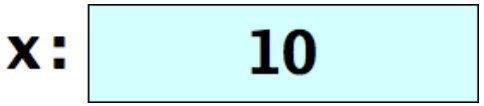
- Quando realizamos uma atribuição:

```
x = y;
```

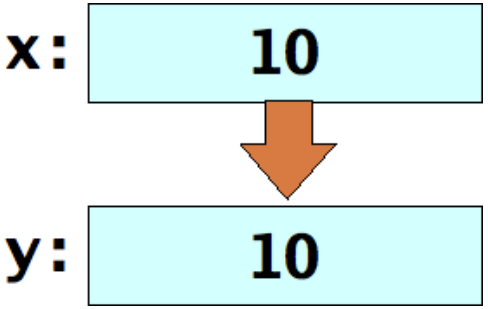
- Java faz a cópia do valor da variável da direita para a variável da esquerda;
  - *Para tipos primitivos, isso significa que alterações em x não implicam alterações em y;*
  - *Para objetos, como o que é copiado é a referência para o mesmo objeto, alterações no objeto que x referencia altera o objeto que y referencia, pois é o mesmo objeto!*

Atribuição de valores primitivos

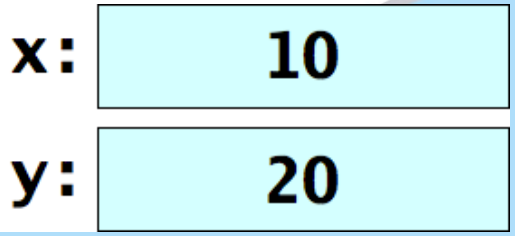
```
int x = 10;
```




```
int y = x;
```

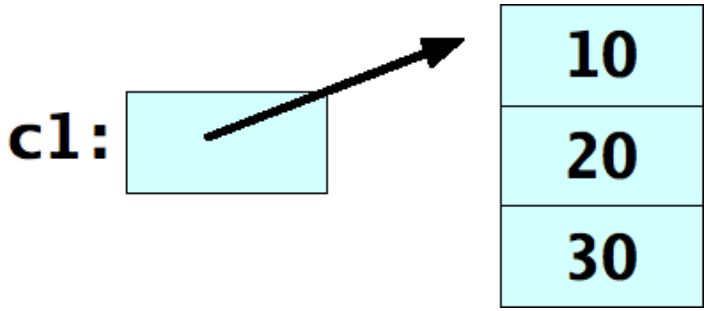



```
y = 20;
```

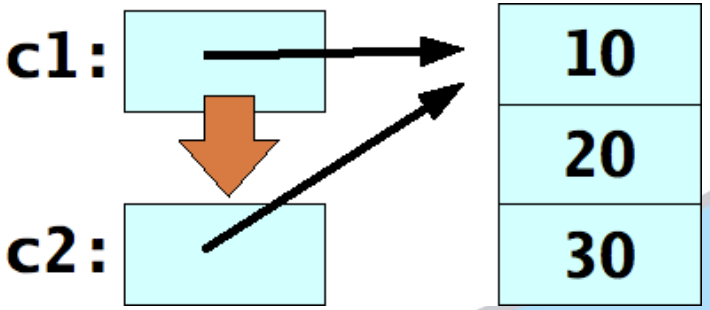


Atribuição de objetos

```
Coordenada c1;
c1 = new Coordenada();
c1.x = 10;
c1.y = 20;
c1.z = 30;
```

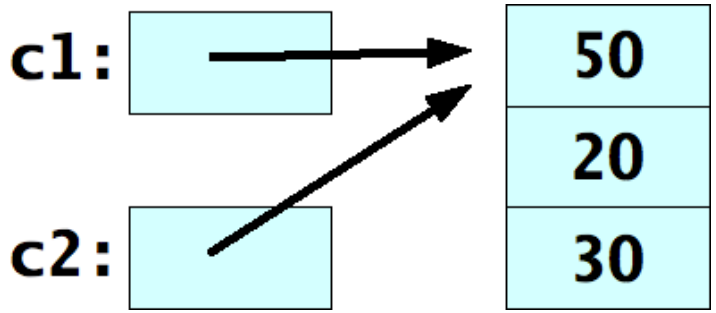



```
Coordenada c2;

// Erro comum:
// c2 = new Coordenada();

c2 = c1;
```



Atribuição de objetos

```
c2.x = 50;
```



Tenha sempre em mente a diferença entre um tipo primitivo e um objeto (referência).

”É parecido com um ponteiro, porém você não pode manipulá-lo como um número e nem utilizá-lo para aritmética, ela é tipada.” (Caelum FJ-11)

Comparações entre objetos

```
public class Comparacoes {
  public static void main(String[] args) {
    Coordenadas c1 = new Coordenadas();
    c1.x = 10; c1.y = 15; c1.z = 20;

    Coordenadas c2 = new Coordenadas();
    c2.x = 10; c2.y = 15; c2.z = 20;

    // O que imprime?
    System.out.println(c1 == c2);
  }
}
```

false

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Comparações entre objetos

```
public class Comparacoes {
  public static void main(String[] args) {
    Coordenadas c1 = new Coordenadas();
    c1.x = 10; c1.y = 15; c1.z = 20;

    Coordenadas c2 = c1;
    c2.x = 11; c2.y = 16; c2.z = 21;

    // O que imprime?
    System.out.println(c1 == c2);
  }
}
```

true

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

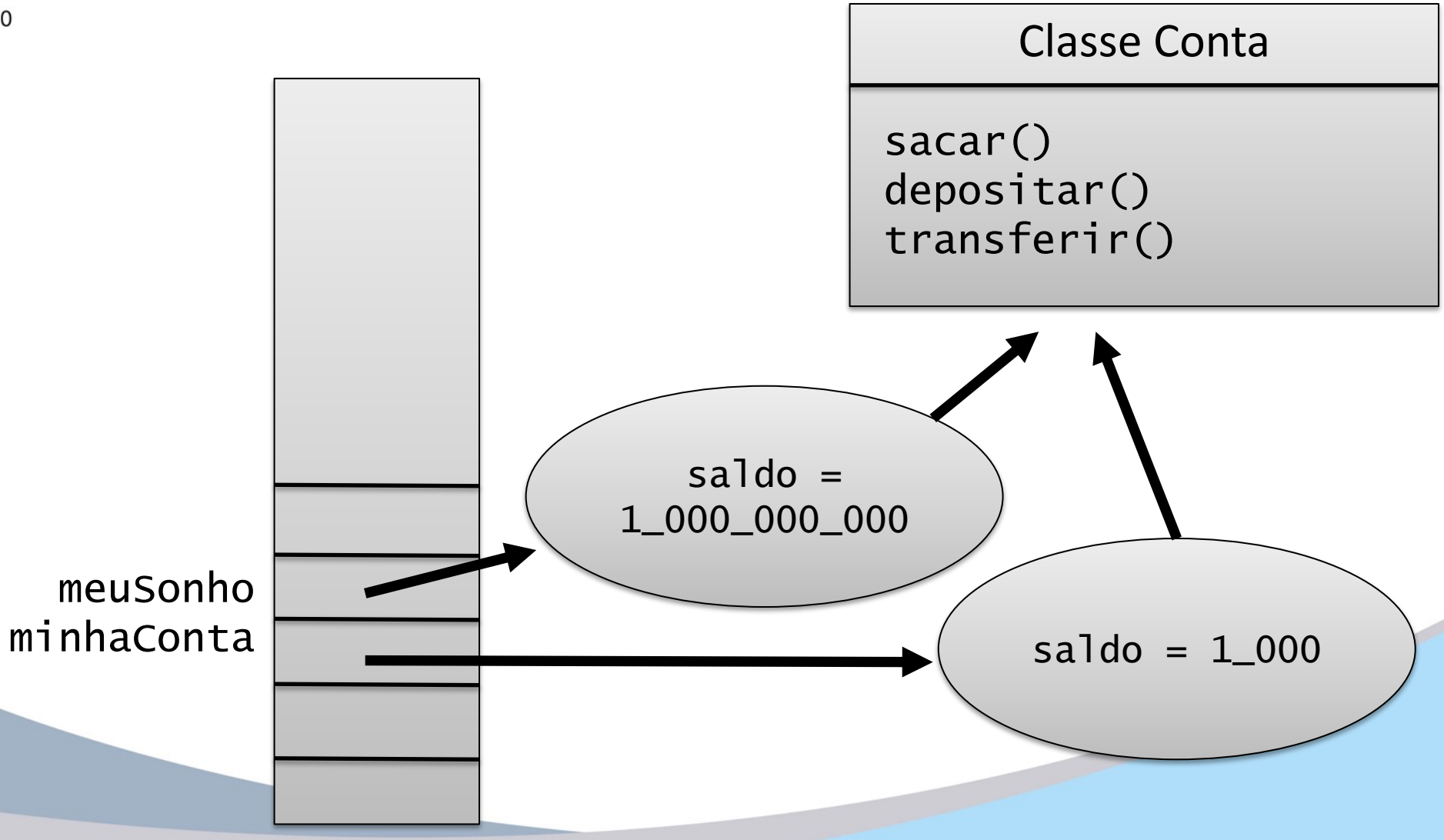
Métodos == funções de classe

- Métodos são funções que executam no contexto de uma classe:

```
class Conta { //...
  boolean transferir(Conta destino, double qtd) {
    if (! this.sacar(qtd)) return false;
    destino.depositar(qtd);
    return true;
  }

  public static void main(String[] args) {
    // Copie aqui minhaConta e meuSonho do slide 15.
    meuSonho.transferir(minhaConta, 1_000_000);
  }
}
```

O destino é minhaConta, mas e a origem?

Armazenamento em memória



UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Chamando métodos em objetos

- O código compilado dos métodos fica na área de memória da classe;
- Sendo assim, como Java sabe em qual objeto estou chamando um determinado método?

```
class Num {
  int i = 5;
  void somar(int j) { i += j; }
}
public class Teste
  public static void main(String[] args) {
    Num m = new Num(), n = new Num();
    m.somar(10); n.somar(5);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Chamando métodos em objetos

- Internamente é como se o método fosse:

```
// Imagina uma "função global", como em C:
void somar(Num this, int j) {
  this.i += j;
}
```

- E a chamada fosse:

```
// Como chamaríamos a função em programação
estruturada.
somar(m, 10);
somar(n, 5);
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

A palavra reservada `this`

- Java faz esta transformação para você, de forma que o objeto que "recebeu a mensagem" está disponível pela palavra-chave `this`:

```
class Num {
  int i = 5;
  void somar(int j) { this.i += j; }
}
```

- Não é necessário usar `this` quando acessamos membros do objeto de dentro do mesmo (como no exemplo acima).

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

A palavra reservada `this`

- `this` pode ser usado para diferenciar um atributo do objeto de um parâmetro do método:

```
class Num {
  int i = 5;
  void somar(int i) { this.i += i; }
}
```

- Neste caso, o `this` é necessário!

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

A palavra reservada this

```
class Num {
  int i = 5;
  Num somar(int j) {
    i += j;
    return this;  // Aqui, this é útil!
  }
}
public class Teste
  public static void main(String[] args) {
    Num m = new Num();
    m.somar(10).somar(5).somar(1);
    System.out.println(m.i); // 21
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Parâmetros variáveis

- A partir do Java 5 é possível definir métodos com um número variável de argumentos (*varargs*):

```
public class Teste {
  void print(boolean msg, String ... objs) {
    if (msg) System.out.println("Args:");
    for (int i = 0; i < objs.length; i++)
      System.out.println(objs[i]);
  }

  public static void main(String[] args) {
    Teste t = new Teste();
    t.print(true, "Java", "Sun", "JCP");
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Parâmetros variáveis

- Só pode haver uma lista de parâmetros variáveis na declaração do método;
- Deve ser a última a ser declarada;
- Funciona como um vetor do tipo declarado (no exemplo, vetor de `String`);
- Não há limite para o número de parâmetros;
- Também aceita zero parâmetros.

```
t.print(false, "A", "B", "C", "D", "E");
t.print(true, "Um", "Dois");
t.print(false);
```

Valores *default* para atributos

- Um atributo pode ser inicializado:

```
class Conta {
  int numero;   // 0
  String dono;  // null
  double saldo; // 0.0
  double limite = 1000.0;
}
```

- Quando não inicializamos explicitamente, um valor *default* é atribuído a ele:

Tipo	Valor
boolean	false
char	'\u0000'
byte	(byte) 0
short	(short) 0
int	0
long	0l
float	0.0f
double	0.0

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Implementando associações entre classes

- Atributos podem ser referências para objetos de outras classes (ou da mesma classe):

```
class Conta {
  int numero;
  double saldo;
  double limite = 1000.0;
  Cliente titular;
}
```

```
class Cliente {
  String nome;
  String sobrenome;
  String cpf;
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Implementando associações entre classes

- Atributos podem ser referências para objetos de outras classes (ou da mesma classe):

```
public class Programa
  public static void main(String[] args) {
    Cliente larry = new Cliente();
    larry.nome = "Larry";
    larry.sobrenome = "Ellison";
    Conta conta = new Conta();
    conta.saldo = 50_400_000_000.0;
    conta.titular = larry;

    // Navegando no grafo de objetos...
    System.out.println(conta.titular.nome);
  }
}
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

O valor null

- O valor *default* para referências (objetos) é `null`;
- É uma referência que não aponta para nenhum objeto;
- Usar uma referência nula como se ela apontasse para um objeto causa `NullPointerException`.

```
public class Programa
  public static void main(String[] args) {
    Conta conta = new Conta();
    conta.saldo = 50_400_000_000.0;
    System.out.println(conta.titular.nome);
  }
}
// Exception in thread "main"
// java.lang.NullPointerException
```

Valores *default* para variáveis locais

- Variáveis locais não são "zeradas" automaticamente e geram erros de compilação se utilizadas sem valor:

```
public class Teste
  public static void main(String[] args) {
    Cliente larry;
    Conta conta = new Conta();
    conta.saldo = 50_400_000_000.0;
    conta.titular = larry;
    System.out.println(conta.titular.nome);
  }
}

// error: variable larry might not have been initialized
//    conta.titular = larry;
//                    ^
```

UNIVERSIDADE FEDERAL
DO ESPÍRITO SANTO

Exercitar é fundamental

- Apostila FJ-11 da Caelum:
  - *Seção 4.12, página 51 (class Funcionario);*
  - *Seção 4.13, página 55 (recursividade / Fibonacci);*
  - *Seção 4.14, página 56 (fixando o conhecimento).*