
Newton
Quem se prepara, não para.

# Programação Orienta a Objetos

3º período

Professora: Michelle Hanne

# Sumário

- ✓ Semântica de Referência
- ✓ Vetor e Matriz de Objetos
- ✓ Membros Estáticos

# Semântica de Referência

# Semântica de Referência

- Em Java não há ponteiros.  Java implementa semântica de referência.
- **Exemplo:** A declaração de um objeto de uma **classe *C*** cria uma referência para um objeto da **classe *C***.

*C obj;*

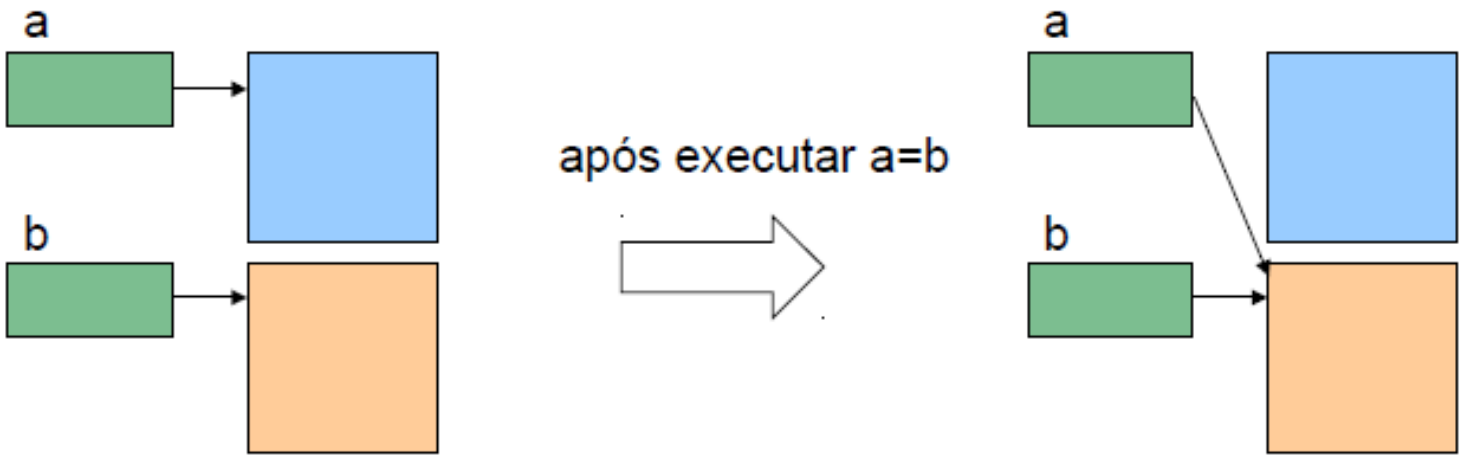
- Um objeto criado deve ser associado a uma referência.

*obj = new C();*

# Semântica de Referência

- A atribuição de uma referência *b* a outra *a*, resulta em *a* e *b* referenciando o mesmo objeto.

```
a = new C();
b = new C();
a = b;
```



# Semântica de Referência

- O que ocorre com o objeto para o qual não há mais referência?
- **Java possui o Coletor de Lixo (*Garbage Collector*) que elimina objetos pendentes na memória de tempos em tempos.**

# Semântica de Referência

- Analise o programa a seguir, escrito em Java e formado pelas classes *Principal* e *ClasseMaluca.*
- **O que ele imprime?**

# Semântica de Referência

```
package exemplosemantica;

/**
 *
 * @author Hanne
 */
public class ClasseMaluca {
private int dado1;
private int dado2;
public ClasseMaluca(int d1, int d2) {
dado1 = d1;
dado2 = d2;
}
public int getDado1(){
return (dado1);
}
public int getDado2(){
return (dado2);
}
}
```

# Semântica de Referência

```
 */
public class ExemploSemantica {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    ClasseMaluca objA, objB;
    objA = new ClasseMaluca(10,20);
    objB = new ClasseMaluca(50,100);
    System.out.println(objA.getDado1() + " " + objA.getDado2());
    System.out.println(objB.getDado1() + " " + objB.getDado2());
    objA = new ClasseMaluca(30,60);
    System.out.println(objA.getDado1() + " " + objA.getDado2());
    objB = objA;
    System.out.println(objB.getDado1() + " " + objB.getDado2());
    }

}
```

# Referência de Objeto

- Variável de classe armazena referência para um objeto.
- `a1` e `a2` apontam para o mesmo objeto:

```
1 AlunoV1 a1 = new AlunoV1(), a2 = new AlunoV1();
2
3 a2 = a1;
4 a1.setNome("Tico Souza");
5 a1.setNota(8.8);
6 System.out.printf("%s %.2f%n", a2.getNome(), a2.getNota());
7 a1 = null;
```

- Referências compartilhadas: cuidado com métodos mutantes.
- `a1` e `a2` apontam para o mesmo objeto (linha 3).
- `a1` aponta para nenhum objeto (linha 7).

# Modificadores de Métodos

- **abstract**: método abstrato, sem corpo.
- **final**: método não pode ser redefinido.
- **public**: método pode ser acessado por outras classes.
- **private**: método só pode ser acessado pela própria classe.
- **protected**: método pode ser acessado por classes dentro do mesmo pacote ou pelas subclasses.
- **static**: método compartilhado por todos os objetos da classe, com acesso a apenas campos estáticos.

# Vetor de Objetos

Newton
Quem se prepara, não para.

```java
import java.util.Scanner;

public class AlunoV1Vetor {
  public static void main(String[] args) {
    final int TOT_ALUNOS = 5;
    AlunoV1[] vet_alunos = new AlunoV1[TOT_ALUNOS];
    double med_notas = 0;

    Scanner in = new Scanner(System.in);
    for(int i = 0; i < TOT_ALUNOS; i++)
      vet_alunos[i] = new AlunoV1();
    for(int i = 0; i < TOT_ALUNOS; i++) {
      vet_alunos[i].setNome(in.nextLine());
      vet_alunos[i].setNota(in.nextDouble());
      in.nextLine(); //para "comer" a quebra de linha
      med_notas += vet_alunos[i].getNota();
    }
    med_notas /= TOT_ALUNOS;
    System.out.println(med_notas);
    in.close();
  }
}
```

# Matriz de Objetos

Newton
Quem se prepara, não para.

```
final int LINHAS = 2, COLUNAS = 3;
AlunoV1[][] alunos = new AlunoV1[LINHAS][COLUNAS];

Scanner in = new Scanner(System.in);

for(int i = 0; i < LINHAS; i++)
  for(int j = 0; j < COLUNAS; j++) {
    alunos[i][j] = new AlunoV1();
    alunos[i][j].setNome(in.nextLine());
    alunos[i][j].setNota(in.nextDouble());
    in.nextLine();
  }
for(int i = 0; i < LINHAS; i++) {
  int maior_col = 0;
  for(int j = 1; j < COLUNAS; j++) {
    if(alunos[i][j].getNota() > alunos[i][maior_col].getNota())
      maior_col = j;
  }
  System.out.printf("Maior nota linha %d: %s %.1f%n", i,
    alunos[i][maior_col].getNome(),
    alunos[i][maior_col].getNota());
}
```

# Membros Estáticos

Newton
Quem se prepara, não para.

- **Membro estático**: pertence a classe, não ao objeto.
  - Variáveis, Constantes, Blocos de Inicialização e Métodos Estáticos.
- Acesso, se público: `NomeClasse.Membro` ou `NomeClasse.Membro(...)`.[3]
- Variável/Constante estática (de classe):
  - Uma variável/constante estática para a classe.
  - Objetos compartilham uma única variável/contante entre todos.
- Blocos de inicialização estáticos:
  - Executado uma única vez, ao carregar a classe na primeira vez, na ordem implementada.
- Métodos estáticos:
  - Não opera em estado de objetos da classe.
  - Não precisa instanciar objetos da classe para usar métodos estático.
  - Pode manipular variáveis/constantes estáticas da classe.
  - Vide `main` e métodos da `Math`.

[3] Pode também ser via um objeto da classe, porém perde-se legibilidade.

# Membros Estáticos

Newton
Quem se prepara, não para.

Arquivo *Empregado.java*:

```
1  public class Empregado {
2    public static int proximoEmpregadoId = 0;
3    public static final double TAXA_AUMENTO_SALARIO = 1.05;
4
5    private int empregadoId;
6    private String nome;
7    private double salario;
8
9    static { System.out.println("Bloco de inicial. estatico 1"); }
10   static { System.out.println("Bloco de inicial. estatico 2"); }
11   static { System.out.println("Bloco de inicial. estatico 3"); }
12
13   public Empregado(String nome, double salario) {
14     proximoEmpregadoId++;
15
16     empregadoId = proximoEmpregadoId;
17     this.nome = nome;
18     this.salario = salario;
19   }
20
21   public int getIdentificacao() { return empregadoId; }
22   ...
23 }
```

# Membros Estáticos

Newton
m se prepara, não para.

Arquivo *EmpregadoPrograma.java*:

```java
public class EmpregadoPrograma {
  public static void main(String[] args) {
    Empregado gerente, supervisor;
    Empregado assistente;

    gerente = new Empregado("Maria", 3000);
    supervisor = new Empregado("Astolfo", 2500);
    assistente = new Empregado("Anastacio", 2000);

    gerente.imprimeDados();
    supervisor.imprimeDados();
    assistente.imprimeDados();

    //acesso a variavel e constante estatica
    System.out.printf("%d %.2f\n",
    Empregado.proximoEmpregadoId,
    Empregado.TAXA_AUMENTO_SALARIO);

    assistente.aumentaSalario();
    assistente.imprimeDados();
  }
}
```

# Membros Estáticos

Newton
Quem se prepara, não para.

Arquivo *Numeros.java* e *NumerosPrograma.java*:

```java
public class Numeros {
  public static int soma(int ...v) {
    if(v.length > 0) {
      int s = 0;
      for(int i = 0; i < v.length; i++) s += v[i];
      return s;
    }
    else return 0;
  }
  public static int multiplica(int ...v) {
    if(v.length > 0) {
      int m = 1;
      for(int i = 0; i < v.length; i++) m *= v[i];
      return m;
    }
    else return 0;
  }
}
```

```java
public class NumerosPrograma {
  public static void main(String[] args) {
    int[] v = {1, 3, 6, 8, 10};

    System.out.println(Numeros.soma(v) + " " + Numeros.soma(2, 4, 6));
    System.out.println(Numeros.multiplica(v) + " " + Numeros.multiplica(-2, 5, -3));
  }
}
```

# Referências


SILVA, Fabricio Machado da. **Paradigmas de programação**. SAGAH, 2019. ISBN digital: 9788533500426

Barnes, David e Kölling, M. **Programação Orientada a Objetos com Java**. São Paulo: Pearson Prentice Hall, 2004.

Deitel, H. M.; Deitel, P. J. **Java - Como Programar**. 6. ed. Prentice-Hall, 2005. Capítulo 4 e 5.

PRESSMAN, Roger S. **Engenharia de Software**. Rio de Janeiro: MacGraw Hill, 2002.