
Newton
Quem se prepara, não para.

Programação Orienta a Objetos

3º período

Professora: Michelle Hanne

Sumário

- ✓ Membros Estáticos
- ✓ Relacionamento de Composição
- ✓ Classes Internas
- ✓ Comentários de Documentação e Gerar JavaDOC

Membros Estáticos

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*:

```java
public class Empregado {
  public static int proximoEmpregadoId = 0;
  public static final double TAXA_AUMENTO_SALARIO = 1.05;

  private int empregadoId;
  private String nome;
  private double salario;

  static { System.out.println("Bloco de inicial. estatico 1"); }
  static { System.out.println("Bloco de inicial. estatico 2"); }
  static { System.out.println("Bloco de inicial. estatico 3"); }

  public Empregado(String nome, double salario) {
    proximoEmpregadoId++;

    empregadoId = proximoEmpregadoId;
    this.nome = nome;
    this.salario = salario;
  }

  public int getIdentificacao() { return empregadoId; }
  ...
}
```

Membros Estáticos

Newton
Quem se prepara, não para.

Arquivo *EmpregadoPrograma.java*:

```
1  public class EmpregadoPrograma {
2    public static void main(String[] args) {
3      Empregado gerente, supervisor;
4      Empregado assistente;
5
6      gerente = new Empregado("Maria", 3000);
7      supervisor = new Empregado("Astolfo", 2500);
8      assistente = new Empregado("Anastacio", 2000);
9
10     gerente.imprimeDados();
11     supervisor.imprimeDados();
12     assistente.imprimeDados();
13
14     //acesso a variavel e constante estatica
15     System.out.printf("%d %.2f\n",
16     Empregado.proximoEmpregadoId,
17     Empregado.TAXA_AUMENTO_SALARIO);
18
19     assistente.aumentaSalario();
20     assistente.imprimeDados();
21   }
22 }
```

Membros Estáticos

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));
  }
}
```

Membros Estáticos

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));
  }
}
```

Composição de Objetos

Composição de Objetos

- Uma classe pode possuir um membro de dado que seja um objeto de outra classe.

- Esse tipo de relacionamento é chamado **composição**.

- Exemplos:
  - Um Círculo possui um Ponto como centro.
  - Uma Conta Corrente é de um Cliente.
  - Uma Turma tem um Professor.
  - Uma Turma tem muitos Alunos.

Composição de Objetos – Exemplo

```
public class Ponto {
    private float x, y;

    public Ponto(float x, float y){
      this.x = x;
      this.y = y;
    }

    public void alterarX(float x){
       this.x = x;
    }

    public void alterarY(float y){
       this.y = y;
    }

    public float obterX(){
      return x;
    }

    public float obterY(){
      return y;
    }
}
```

Composição de Objetos – Exemplo

```
public class Circulo {

    private float raio;
    private Ponto centro;

    public Circulo(float x, float y, float r){
       raio = r;
       centro = new Ponto(x,y);
    }

    public void alterarCentro(float x, float y){
       centro.alterarX(x);
       centro.alterarY(y);
    }

    public void alterarRaio(float r){
        raio = r;
    }

    public float obterCentroX(){
      return centro.obterX();
    }
```

Composição de Objetos – Exemplo

```
public float obterCentroY(){
      return centro.obterY();
    }

    public float obterRaio(){
      return raio;
    }

}
```

Composição de Objetos – Exemplo

```
public class Aplicacao {

    public static void main(String[] args){
          float x, y, raio;
          Circulo circ;

          Scanner in = new Scanner (System.in);

          System.out.println("Digite as coordenadas do centro: ");
          x = in.nextFloat();
          y = in.nextFloat();
          System.out.println("Digite o raio: ");
          raio = in.nextFloat();

          circ = new Circulo (x, y, raio);
          System.out.println("Circulo criado: ");
          System.out.println("Raio: " + circ.obterRaio());
          System.out.println("Centro: (" + circ.obterCentroX() + ", " +
                              circ.obterCentroY() + ").");
          System.out.println("");
    }
}
```

Composição de Objetos – Exemplo

- Considerando as classes Ponto e Circulo mostradas anteriormente, o que a classe Aplicação a seguir gera como saída para entradas 1, 1 e 2?

Composição de Objetos – Exemplo

```
public class Aplicacao {

    public static void alteraCirculo(Circulo c){
        c.alterarCentro(10, 20);
        c.alterarRaio(5);
        System.out.println("\n\n**Dados do circulo dentro do método: **");
        System.out.println("Raio: " + c.obterRaio());
        System.out.println("Centro: (" + c.obterCentroX() + ", " +
                           c.obterCentroY() + ").");

        c = new Circulo(3,3,9);
        System.out.println("\n\n**Dados do novo circulo dentro do método:
                          **");
        System.out.println("Raio: " + c.obterRaio());
        System.out.println("Centro: (" + c.obterCentroX() + ", " +
                           c.obterCentroY() + ").");

    }
```

Composição de Objetos – Exemplo

```
public static void main(String[] args){
        float x, y, raio;
        Circulo circ;

        Scanner in = new Scanner (System.in);

        System.out.println("Digite as coordenadas do centro: ");
        x = in.nextFloat();
        y = in.nextFloat();
        System.out.println("Digite o raio: ");
        raio = in.nextFloat();

        circ = new Circulo (x, y, raio);
        System.out.println("Circulo criado: ");
        System.out.println("Raio: " + circ.obterRaio());
        System.out.println("Centro: (" + circ.obterCentroX() + ", " +
                           circ.obterCentroY() + ").");

        //Passando objeto circ como parâmetro
        alteraCirculo(circ);
```

Composição de Objetos – Exemplo

```
 System.out.println("\n**Circulo após execução do método: **");
          System.out.println("Raio: " + circ.obterRaio());
          System.out.println("Centro: (" + circ.obterCentroX() + ", " +
                             circ.obterCentroY() + ").");
          System.out.println("");
    }
}
```

Circulo criado:
Raio: 2.0
Centro: (1.0, 1.0).

**Dados do circulo dentro do método: **
Raio: 5.0
Centro: (10.0, 20.0).

**Dados do novo circulo dentro do método: **
Raio: 9.0
Centro: (3.0, 3.0).

**Circulo após execução do método: **
Raio: 5.0
Centro: (10.0, 20.0).

Composição - Exemplo

Newton
Quem se prepara, não para.

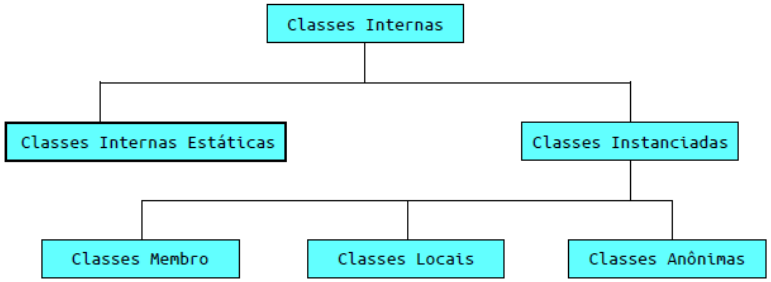
- Propriedades que são referências de objetos de outras classes.
- **Composição** ou **relacionamento tem um**.

```java
public class Tempo {
  private int hora, minuto;

  public Tempo(int hora, int minuto) {...}
  public Tempo(Tempo o) {...}
  public void imprime() {...}
  public boolean eIgual(Tempo o) {...}
  public void aumentaMinuto() {...}
}
```

```java
public class Alarme {
  private Tempo relogio, horario;
  private String musica;

  public Alarme(Tempo relogio, Tempo horario, String musica) {...}
  public boolean vaiDespertar() {...}
  public void play() {...}
  public void passaTempo() {...}
}
```

Classes Internas

- Implementação de classes dentro de outra.
- Restringe visibilidade e aumenta encapsulamento.
- Evita nomes genéricos de classe no pacote (`Elemento`, `No`, `Item`...).
- Melhora legibilidade por agrupamento de similaridades.
- Pode-se ocultar com modificadores `private` ou `protected`.
- Classe interna acessa membros da externa que a definiu.
- Estão no mesmo `.java`, mas geram 1 `.class` por classe:
  - `Externa.class` e `Externa$Interna.class`.
- Interna não pode ter o mesmo nome da externa.



Classes Interna Estática

Newton
Quem se prepara, não para.

- Também conhecida como classe aninhada, é definida como estática.
- Não relaciona com membros de instância da classe externa.
- Possui acesso apenas aos membros estáticos da classe externa:
  - Referência via `Externa.Interna`.

```
1  class ExemploClasse {
2    static int v1 = 10;
3    int v2 = 20;
4
5    void msg() {
6      System.out.println(v1);
7      System.out.println(v2);
8      //System.out.println(Interna.v3); //nao e possivel, pois v3 nao e estatica
9    }
10
11   static class Interna {
12     int v3 = 30;
13
14     void msgInterna() {
15       System.out.println(v1);
16       //System.out.println(v2); //nao e possivel, pois v2 nao e estatica
17       System.out.println(v3);
18     }
19   }
20 }
```

Classes Instanciadas

Newton
Quem se prepara, não para.

- Não estática interna que manipula estado do objeto de classe externa.
- Necessário um objeto da classe externa que referencie a interna.
- Tipos:
  - Classe Membro:
    - Não estática, manipula qualquer variável de instância da classe externa.
    - Definida no escopo da classe externa.
    - Referência: `Externa.Interna`
    - Instanciação: `ObjetoClasseExterna.new Interna()`
  - Classe Local:
    - Pertence ao escopo de um bloco (método, por exemplo).
    - Sem especificador de acesso.
    - Acessa (não altera) parâmetros e variáveis (já declaradas) do bloco.
  - Classe Anônima:
    - Sem identificador e construtor.
    - Cria apenas uma instância.
    - Criada a partir de outra classe ou interface.
    - Sintaxe estranha, mas facilita escrita de códigos comuns.

Classes Instanciadas

- Exemplo de Classe Membro:

```
1  public class CIMembro {
2    public static void main(String[] args) {
3      ClasseMembro o1 = new ClasseMembro(1);
4      ClasseMembro o2 = new ClasseMembro(2);
5
6      ClasseMembro.ClasseMembroInterna i1 = o1.new ClasseMembroInterna();
7      ClasseMembro.ClasseMembroInterna i2 = o2.new ClasseMembroInterna();
8
9      i1.imprimir();
10     i2.imprimir();
11   }
12 }
13
14 class ClasseMembro {
15   private int valor;
16
17   ClasseMembro(int v) {
18     valor = v;
19   }
20
21   class ClasseMembroInterna {
22     void imprimir() {
23       System.out.println(valor);
24     }
25   }
26 }
```

Classes Instanciadas

Newton
Quem se prepara, não para.

- Exemplo de Classe Local:

```
1  public class CILocal {
2    static void funcaoMembro(final int a, int b) {
3      final String sim = "ok";
4      int c = 10;
5
6      class ClasseLocal {
7        void imprimir() {
8          System.out.println(a);
9          System.out.println(sim);
10         System.out.println(b);
11         System.out.println(c);
12       }
13     }
14
15     System.out.println(b);
16
17     ClasseLocal local = new ClasseLocal();
18     local.imprimir();
19   }
20
21   public static void main(String[] args) {
22     funcaoMembro(10, 20);
23   }
24 }
```

Classes Instanciadas

Newton
em se prepara, não para.

- Exemplo de Classe Anônima:

```
1  abstract class Pessoa {
2    abstract void comer();
3  }
4
5  public class CIAnonima {
6    public static void main(String args[]) {
7      Pessoa p = new Pessoa() {
8        int i = 1;
9
10       void comer() {
11         System.out.println(i + " bolo... Nhaac!");
12       }
13     };
14
15     p.comer();
16   }
17 }
```

Comentários de Documentação

Newton
Quem se prepara, não para.

- `/** ... */`: Comentário de documentação e `javadoc` geram HTML.
  - Classes públicas, interfaces, construtores e métodos públicos, variáveis públicas e protegidas, pacotes e módulos.
  - Pode ter texto, tags (`@author`, `@param`, ...) e modificadores HTML.
- Comentário de classe:

```
1 /**
2 * An <code>Invoice</code> object represents an invoice with line items for each part of the
3 * @author Fred Flintstone
4 * @author Barney Rubble
5 * @version 1.1
6 */
7 public class Invoice {
8   ...
9 }
```

Project-> Generate Java DOC

- Comentário de método:

```
1  /**
2  * Raises the salary of an employee.
3  * @param byPercent the percentage by which to raise the salary (e.g., 10 means 10%)
4  * @return the amount of the raise
5  */
6  public double raiseSalary(double byPercent) {
7    double raise = salary * byPercent / 100;
8    salary += raise;
9    return raise;
10 }
```

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.