

Aula 13 – Paradigmas de Programação e Orientação a Objetos

Norton Trevisan Roman

23 de abril de 2018

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:
 - Primeiro faça isso, depois aquilo

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:
 - Primeiro faça isso, depois aquilo
 - Trabalhamos com comandos dados em uma ordem específica

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:
 - Primeiro faça isso, depois aquilo
 - Trabalhamos com comandos dados em uma ordem específica
- A abstração natural é a subrotina

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:
 - Primeiro faça isso, depois aquilo
 - Trabalhamos com comandos dados em uma ordem específica
- A abstração natural é a subrotina
 - Similar a rotinas do dia-a-dia: receitas, instruções para se fazer algo

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:
 - Primeiro faça isso, depois aquilo
 - Trabalhamos com comandos dados em uma ordem específica
- A abstração natural é a subrotina
 - Similar a rotinas do dia-a-dia: receitas, instruções para se fazer algo
 - Dados podem estar separados das subrotinas

Paradigma Imperativo

- Até agora trabalhamos com um modo de abordar problemas:
 - Primeiro faça isso, depois aquilo
 - Trabalhamos com comandos dados em uma ordem específica
- A abstração natural é a subrotina
 - Similar a rotinas do dia-a-dia: receitas, instruções para se fazer algo
 - Dados podem estar separados das subrotinas
 - Agrupados conforme o domínio ou necessidade computacional

Paradigma Imperativo

- Esse é o paradigma imperativo

Paradigma Imperativo

- Esse é o paradigma imperativo
 - Também conhecido como procedimental (ou procedural)

Paradigma Imperativo

- Esse é o paradigma imperativo
 - Também conhecido como procedimental (ou procedural)
 - Vê o problema como um conjunto de passos a serem resolvidos da forma “Primeiro faça isso, depois aquilo”

Paradigma Imperativo

- Esse é o paradigma imperativo
 - Também conhecido como procedimental (ou procedural)
 - Vê o problema como um conjunto de passos a serem resolvidos da forma “Primeiro faça isso, depois aquilo”
 - Baseado totalmente na máquina de Von Neumann

Paradigma Imperativo

- Esse é o paradigma imperativo
 - Também conhecido como procedimental (ou procedural)
 - Vê o problema como um conjunto de passos a serem resolvidos da forma “Primeiro faça isso, depois aquilo”
 - Baseado totalmente na máquina de Von Neumann
- Ex: C, Pascal.

Paradigma Imperativo

- Esse é o paradigma imperativo
 - Também conhecido como procedimental (ou procedural)
 - Vê o problema como um conjunto de passos a serem resolvidos da forma “Primeiro faça isso, depois aquilo”
 - Baseado totalmente na máquina de Von Neumann
- Ex: C, Pascal.
 - `int c = a + 2;`

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função
 - Abstrai uma expressão simples como uma função que pode ser avaliada como uma expressão

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função
 - Abstrai uma expressão simples como uma função que pode ser avaliada como uma expressão
 - Todos os cálculos são feitos pela aplicação de funções

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função
 - Abstrai uma expressão simples como uma função que pode ser avaliada como uma expressão
 - Todos os cálculos são feitos pela aplicação de funções
 - Não há variáveis globais, apenas parâmetros e variáveis locais

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função
 - Abstrai uma expressão simples como uma função que pode ser avaliada como uma expressão
 - Todos os cálculos são feitos pela aplicação de funções
 - Não há variáveis globais, apenas parâmetros e variáveis locais
- Baseado na matemática e na teoria das funções

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função
 - Abstrai uma expressão simples como uma função que pode ser avaliada como uma expressão
 - Todos os cálculos são feitos pela aplicação de funções
 - Não há variáveis globais, apenas parâmetros e variáveis locais
- Baseado na matemática e na teoria das funções
- Ex: Lisp (LISt Processing), Scheme

Paradigma Funcional

- Avalie uma expressão e use o valor resultante para algo
- A abstração natural é a função
 - Abstrai uma expressão simples como uma função que pode ser avaliada como uma expressão
 - Todos os cálculos são feitos pela aplicação de funções
 - Não há variáveis globais, apenas parâmetros e variáveis locais
- Baseado na matemática e na teoria das funções
- Ex: Lisp (LISt Processing), Scheme
 - `(let ((a 3) (c (+ a 2))))`

Paradigma Lógico

- Responda uma pergunta através da busca pela solução

Paradigma Lógico

- Responda uma pergunta através da busca pela solução
- Baseado em axiomas, regras de inferências e buscas

Paradigma Lógico

- Responda uma pergunta através da busca pela solução
- Baseado em axiomas, regras de inferências e buscas
- Muito usado em domínios que lidam com a extração de conhecimento a partir de fatos e relações entre eles

Paradigma Lógico

- Responda uma pergunta através da busca pela solução
- Baseado em axiomas, regras de inferências e buscas
- Muito usado em domínios que lidam com a extração de conhecimento a partir de fatos e relações entre eles
- A execução do problema torna-se uma busca sistemática em um conjunto de fatos

Paradigma Lógico

- Responda uma pergunta através da busca pela solução
- Baseado em axiomas, regras de inferências e buscas
- Muito usado em domínios que lidam com a extração de conhecimento a partir de fatos e relações entre eles
 - A execução do problema torna-se uma busca sistemática em um conjunto de fatos
 - Usa, para isso, um conjunto de regras de inferência

Paradigma Lógico

- Responda uma pergunta através da busca pela solução
- Baseado em axiomas, regras de inferências e buscas
- Muito usado em domínios que lidam com a extração de conhecimento a partir de fatos e relações entre eles
 - A execução do problema torna-se uma busca sistemática em um conjunto de fatos
 - Usa, para isso, um conjunto de regras de inferência
- Ex: Prolog, Mercury

Paradigma Lógico

- Responda uma pergunta através da busca pela solução
- Baseado em axiomas, regras de inferências e buscas
- Muito usado em domínios que lidam com a extração de conhecimento a partir de fatos e relações entre eles
 - A execução do problema torna-se uma busca sistemática em um conjunto de fatos
 - Usa, para isso, um conjunto de regras de inferência
- Ex: Prolog, Mercury
 - C is $A + 2$.

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse
 - Technicalidades da programação ficam em segundo plano

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse
 - Technicalidades da programação ficam em segundo plano
 - Muito importante quando os programas ficam grandes

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse
 - Technicalidades da programação ficam em segundo plano
 - Muito importante quando os programas ficam grandes
- Operações e dados são encapsulados em classes

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse
 - Technicalidades da programação ficam em segundo plano
 - Muito importante quando os programas ficam grandes
- Operações e dados são encapsulados em classes
 - Esconde informação para proteger propriedades internas da classe

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse
 - Technicalidades da programação ficam em segundo plano
 - Muito importante quando os programas ficam grandes
- Operações e dados são encapsulados em classes
 - Esconde informação para proteger propriedades internas da classe
- Classes são organizadas hierarquicamente, por meio de herança

Paradigma Orientado a Objetos

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
 - Conforme conceitos importantes para o domínio de interesse
 - Technicalidades da programação ficam em segundo plano
 - Muito importante quando os programas ficam grandes
- Operações e dados são encapsulados em classes
 - Esconde informação para proteger propriedades internas da classe
- Classes são organizadas hierarquicamente, por meio de herança
 - Permite a extensão ou especialização de classes

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com
imperativo)

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com
imperativo)
 - `int c = a + 2;`

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com
imperativo)
- `int c = a + 2;`
- `Integer c = new
Integer(a.intValue() + 2);`

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com imperativo)
 - `int c = a + 2;`
 - `Integer c = new Integer(a.intValue() + 2);`
- Ruby, Smalltalk
(orientação a objetos “pura” – tudo é objeto, inclusive literais)

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com imperativo)
 - `int c = a + 2;`
 - `Integer c = new Integer(a.intValue() + 2);`
- Ruby, Smalltalk
(orientação a objetos “pura” – tudo é objeto, inclusive literais)
 - `c = a + 2`

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com imperativo)
 - `int c = a + 2;`
 - `Integer c = new Integer(a.intValue() + 2);`
- Ruby, Smalltalk
(orientação a objetos “pura” – tudo é objeto, inclusive literais)
 - `c = a + 2`
 - `c = a.+(2)`

Paradigma Orientado a Objetos

- Java, C++
(compatíveis com imperativo)
 - `int c = a + 2;`
 - `Integer c = new Integer(a.intValue() + 2);`
- Ruby, Smalltalk
(orientação a objetos “pura” – tudo é objeto, inclusive literais)
 - `c = a + 2`
 - `c = a.+(2)`
 - `c = 2.+(a)`

Classes e Objetos

Classes:

- Agrupamento de entidades que possuem alguns atributos e métodos em comum
- Representam o modelo por trás do problema – a ideia

Classes e Objetos

Classes:

- Agrupamento de entidades que possuem alguns atributos e métodos em comum
- Representam o modelo por trás do problema – a ideia

Objetos:

- Representantes das classes
- Constituem as entidades computacionais que representam as classes
 - São a implementação física da classe

Encapsulamento e Subclasses

Encapsulamento

- Ato de agrupar entidades que contenham atributos e métodos em comum
- Atributos e métodos locais a um grupo de entidades ficam assim escondidos do resto do problema

Encapsulamento e Subclasses

Encapsulamento

- Ato de agrupar entidades que contenham atributos e métodos em comum
- Atributos e métodos locais a um grupo de entidades ficam assim escondidos do resto do problema

Subclasses

- São sub-agrupamentos de entidades distintas, mas que possuem alguns atributos em comum

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?
 - São “herdados” pela subclasse

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?
 - São “herdados” pela subclasse
 - Nem todos... mais tarde veremos

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?
 - São “herdados” pela subclasse
 - Nem todos... mais tarde veremos
 - Como se a subclasse possuísse também aqueles atributos e métodos

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?
 - São “herdados” pela subclasse
 - Nem todos... mais tarde veremos
 - Como se a subclasse possuísse também aqueles atributos e métodos
 - Não há a necessidade de repetir código

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?
 - São “herdados” pela subclasse
 - Nem todos... mais tarde veremos
 - Como se a subclasse possuísse também aqueles atributos e métodos
 - Não há a necessidade de repetir código
- Como separamos o problema em classes e subclasses?

Herança

- Quando criamos uma subclasse, o que acontece com os atributos e métodos da superclasse?
 - São “herdados” pela subclasse
 - Nem todos... mais tarde veremos
 - Como se a subclasse possuísse também aqueles atributos e métodos
 - Não há a necessidade de repetir código
- Como separamos o problema em classes e subclasses?
 - Dependerá das características do problema e da sua interpretação dele

Pondo os Pés no Chão

● Voltando à casa...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...

/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                           int material)

/* Calcula o valor total da construção */
static double valorCasa(double area)

/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)

/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```


Pondo os Pés no Chão

• Voltando à casa...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...

/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                             int material)

/* Calcula o valor total da construção */
static double valorCasa(double area)

/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)

/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

• Existiria uma divisão natural aqui?

Pondo os Pés no Chão

- Voltando à casa...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...
```

```
/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                           int material)
```

```
/* Calcula o valor total da construção */
static double valorCasa(double area)
```

```
/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)
```

```
/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

- Existiria uma divisão natural aqui? Métodos e atributos relativos à casa...

Pondo os Pés no Chão

- Voltando à casa...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...
```

```
/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                             int material)
```

```
/* Calcula o valor total da construção */
static double valorCasa(double area)
```

```
/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)
```

```
/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

- Existiria uma divisão natural aqui? Métodos e atributos relativos à casa... E os relativos à piscina

Então...

Casa

Piscina

Então...

Casa

- Valor do metro quadrado

Piscina

Então...

Casa

- Valor do metro quadrado

Piscina

- Tipos de materiais

Então...

Casa

- Valor do metro quadrado

Piscina

- Tipos de materiais
- Preços do m² dos materiais

Então...

Casa

- Valor do metro quadrado

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais

Então...

Casa

- Valor do metro quadrado
- Métodos:

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais

Então...

Casa

- Valor do metro quadrado
- Métodos:
 - Cálculo da área

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais

Então...

Casa

- Valor do metro quadrado
- Métodos:
 - Cálculo da área

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais
- Métodos:

Então...

Casa

- Valor do metro quadrado
- Métodos:
 - Cálculo da área

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais
- Métodos:
 - Cálculo da área

Então...

Casa

- Valor do metro quadrado
- Métodos:
 - Cálculo da área
 - Cálculo do valor total

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais
- Métodos:
 - Cálculo da área

Então...

Casa

- Valor do metro quadrado
- Métodos:
 - Cálculo da área
 - Cálculo do valor total

Piscina

- Tipos de materiais
- Preços do m² dos materiais
- Nomes dos materiais
- Métodos:
 - Cálculo da área
 - Cálculo do valor total, e métodos auxiliares

Então...

- Podemos simplesmente criar duas classes para acomodar essas diferenças

Então...

- Podemos simplesmente criar duas classes para acomodar essas diferenças
- E como ficaria o cálculo envolvendo casa e piscina?

Então...

- Podemos simplesmente criar duas classes para acomodar essas diferenças
- E como ficaria o cálculo envolvendo casa e piscina?
- Poderia ficar a cargo de uma terceira classe

Então...

- Podemos simplesmente criar duas classes para acomodar essas diferenças
- E como ficaria o cálculo envolvendo casa e piscina?
- Poderia ficar a cargo de uma terceira classe
- Ou poderia ficar dentro de Casa

Então...

- Podemos simplesmente criar duas classes para acomodar essas diferenças
- E como ficaria o cálculo envolvendo casa e piscina?
- Poderia ficar a cargo de uma terceira classe
- Ou poderia ficar dentro de Casa
 - Nesse caso, cada casa teria um representante da classe Piscina dentro dela

Classes em Java

- Como criamos uma classe em java?

Classes em Java

- Como criamos uma classe em java?

```
class Nome_da_classe {  
  
}
```

Classes em Java

- Como criamos uma classe em java?

```
class Nome_da_classe {  
  
}
```
- Como vínhamos fazendo – o programa nada mais era que uma classe

Classes em Java

- Como criamos uma classe em java?

```
class Nome_da_classe {  
  
}
```
- Como vínhamos fazendo – o programa nada mais era que uma classe
- Como tal, deveria agrupar coisas que julgamos ser semelhantes

Classes em Java

- Como criamos uma classe em java?

```
class Nome_da_classe {  
  
}
```
- Como vínhamos fazendo – o programa nada mais era que uma classe
- Como tal, deveria agrupar coisas que julgamos ser semelhantes
- Por convenção, o nome de uma classe inicia em maiúscula

Classes em Java

Componentes da classe:

```
class Nome_da_classe {  
  
  
  
  
  
}
```

Classes em Java

Componentes da classe:

- Atributos (ou campos)

```
class Nome_da_classe {  
    static int x = 2;  
  
}
```

Classes em Java

Componentes da classe:

- Atributos (ou campos)
- Métodos

```
class Nome_da_classe {  
    static int x = 2;  
  
    static void faz(int y) {  
        System.out.println(y);  
    }  
}
```

Classes em Java

Componentes da classe:

- Atributos (ou campos)
- Métodos
- Todos têm que ser static?

```
class Nome_da_classe {  
    static int x = 2;  
  
    static void faz(int y) {  
        System.out.println(y);  
    }  
}
```

Classes em Java

Componentes da classe:

- Atributos (ou campos)

```
class Nome_da_classe {  
    static int x = 2;
```

- Métodos

- Todos têm que ser static?

```
    static void faz(int y) {  
        System.out.println(y);  
    }
```

- Não. Veremos mais adiante...

```
}
```

Classes em Java

• Voltando à casa...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                           'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i',
                           'c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...

/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                           int material)

/* Calcula o valor total da construção */
static double valorCasa(double area)

/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)

/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

Classes em Java

• Voltando à casa... vamos reagrupar ...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                           'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...

/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                           int material)

/* Calcula o valor total da construção */
static double valorCasa(double area)

/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)

/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

Classes em Java

- Voltando à casa... vamos reagrupar ...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...
```

```
/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                           int material)
```

```
/* Calcula o valor total da construção */
static double valorCasa(double area)
```

```
/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)
```

```
/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

- Atributos e métodos relativos à área da casa

Classes em Java

- Voltando à casa... vamos reagrupar ...

```
/* valor do metro quadrado da casa */
static double valorM2 = 1500;
/* materiais da piscina */
static final int ALVENARIA = 0;
static final int VINIL = 1;
static final int FIBRA = 2;
static final int PLASTICO = 3;
/* preços dos materiais */
static double[] precos = {1500, 1100, 750, 500};
/* nomes dos materiais */
static char[][] nomes = {{'A','l','v','e','n','a',
                          'r','i','a'},
                          {'V','i','n','i','l'},
                          {'F','i','b','r','a'},
                          {'P','l','á','s','t','i','c','o'}};

/* Calcula a área da casa */
static void areaCasa(float lateral, float cquarto)
```

```
/* Calcula a área da piscina */
static double areaPiscina(double raio) ...
```

```
/* Calcula o valor da construção da piscina */
static double valorPiscina(double area,
                             int material)
```

```
/* Calcula o valor total da construção */
static double valorCasa(double area)
```

```
/* Carrega os valores das piscinas na matriz de
   área X material */
public static void carregaVal(double[][] m)
```

```
/* Retorna matriz com os preços finais. */
public static double[][] calculaFinal(
    double[][] val,
    double[][] desc)
```

- Atributos e métodos relativos à área da casa
- Atributos e métodos relativos à área da piscina

Classes em Java

● Métodos relativos à área da casa

```
/* Programa para calcular a área de uma casa
   (e seus cômodos) de 3 cômodos: uma sala de
   10X10m, um banheiro e um quarto de 5X7m cada.*/
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    static double valorM2 = 1500;

    /* Calcula a área da casa */
    static void area(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        if (lateral>=0 && cquarto>=0) {
            System.out.println("Cálculo da área da casa");
            areas = lateral*lateral;
            System.out.println("A área da sala é "+areas);
            areaq = cquarto*(lateral/2);
            System.out.println("A área do banheiro é "+areaq);
            System.out.println("A área do quarto é "+areaq);
            areat = areas + 2*areaq;
            System.out.println("A área total é " + areat);
        }
        else
            System.out.println("Erro: parâmetro < 0");
    }
}
```

```
/* Calcula o valor total da casa */
static double valor(double area) {
    if (area >= 0) return(valorM2*area);
    return(-1);
}

public static void main(String[] args) {
}
}
```

Classes em Java

● Métodos relativos à área da casa

```
/* Programa para calcular a área de uma casa
   (e seus cômodos) de 3 cômodos: uma sala de
   10X10m, um banheiro e um quarto de 5X7m cada.*/
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    static double valorM2 = 1500;

    /* Calcula a área da casa */
    static void area(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        if (lateral>=0 && cquarto>=0) {
            System.out.println("Cálculo da área da casa");
            areas = lateral*lateral;
            System.out.println("A área da sala é "+areas);
            areaq = cquarto*(lateral/2);
            System.out.println("A área do banheiro é "+areaq);
            System.out.println("A área do quarto é "+areaq);
            areat = areas + 2*areaq;
            System.out.println("A área total é " + areat);
        }
        else
            System.out.println("Erro: parâmetro < 0");
    }
}
```

```
/* Calcula o valor total da casa */
static double valor(double area) {
    if (area >= 0) return(valorM2*area);
    return(-1);
}

public static void main(String[] args) {
}
}
```

● Arquivo: AreaCasa.java

Classes em Java

- Métodos relativos à área da casa

```
/* Programa para calcular a área de uma casa
   (e seus cômodos) de 3 cômodos: uma sala de
   10X10m, um banheiro e um quarto de 5X7m cada.*/
class AreaCasa {
    /* valor do metro quadrado da casa */
    static double valorM2 = 1500;

    /* Calcula a área da casa */
    static void area(float lateral, float cquarto) {
        float areaq; // área do quarto
        float areas; // área da sala
        float areat; // área total

        if (lateral>=0 && cquarto>=0) {
            System.out.println("Cálculo da área da casa");
            areas = lateral*lateral;
            System.out.println("A área da sala é "+areas);
            areaq = cquarto*(lateral/2);
            System.out.println("A área do banheiro é "+areaq);
            System.out.println("A área do quarto é "+areaq);
            areat = areas + 2*areaq;
            System.out.println("A área total é " + areat);
        }
        else
            System.out.println("Erro: parâmetro < 0");
    }
}
```

```
/* Calcula o valor total da casa */
static double valor(double area) {
    if (area >= 0) return(valorM2*area);
    return(-1);
}

public static void main(String[] args) {
}
}
```

- Arquivo:
AreaCasa.java
- Ela possui seu próprio
main

Classes em Java

● Métodos relativos à área da piscina

```
class AreaPiscina {
    static final int ALVENARIA = 0;
    static final int VINIL = 1;
    static final int FIBRA = 2;
    static final int PLASTICO = 3;
    static double[] precos = {1500, 1100, 750, 500};
    static char[][] nomes =
        {{'A','l','v','e','n','a','r','i','a'},
         {'V','i','n','i','l'}, {'F','i','b','r','a'},
         {'P','l','á','s','t','i','c','o'}};

    static double area(double raio) {
        return((raio >= 0) ? Math.PI * Math.pow(raio,2) : -1);
    }

    static double valor(double area, int material) {
        if (material<ALVENARIA || material>PLASTICO || area<0)
            return(-1);
        return(area*precos[material]);
    }

    public static void carregaVal(double[][] m) {
        for (int i=0; i<m.length; i++) {
            for (int j=50; j<=200; j+=50) {
                m[i][j / 50 - 1] = precos[i] * j;
            }
        }
    }

    public static double[][] calculaFinal(
        double[][] val,double[][] desc) {
        double[][] saida = new double[val.length]
                                [val[0].length];

        for (int i=0; i<saida.length; i++) {
            for (int j=0; j<saida[0].length; j++)
                saida[i][j] = val[i][j] * (1-desc[i][j]);
        }
        return(saida);
    }

    public static void main(String[] args) {
    }
}
```

Classes em Java

● Métodos relativos à área da piscina

```
class AreaPiscina {
    static final int ALVENARIA = 0;
    static final int VINIL = 1;
    static final int FIBRA = 2;
    static final int PLASTICO = 3;
    static double[] precos = {1500, 1100, 750, 500};
    static char[][] nomes =
        {{ 'A', 'l', 'v', 'e', 'n', 'a', 'r', 'i', 'a' },
          { 'V', 'i', 'n', 'i', 'l' }, { 'F', 'i', 'b', 'r', 'a' },
          { 'P', 'l', 'á', 's', 't', 'i', 'c', 'o' } };

    static double area(double raio) {
        return((raio >= 0) ? Math.PI * Math.pow(raio,2) : -1);
    }

    static double valor(double area, int material) {
        if (material<ALVENARIA || material>PLASTICO || area<0)
            return(-1);
        return(area*precos[material]);
    }

    public static void carregaVal(double[][] m) {
        for (int i=0; i<m.length; i++) {
            for (int j=50; j<=200; j+=50) {
                m[i][j / 50 - 1] = precos[i] * j;
            }
        }
    }
}
```

```
public static double[][] calculaFinal(
    double[][] val, double[][] desc) {
    double[][] saida = new double[val.length]
        [val[0].length];

    for (int i=0; i<saida.length; i++) {
        for (int j=0; j<saida[0].length; j++)
            saida[i][j] = val[i][j] * (1-desc[i][j]);
    }
    return(saida);
}

public static void main(String[] args) {
}
}
```

● AreaPiscina.java

Classes em Java

- Métodos relativos à área da piscina

```
class AreaPiscina {
    static final int ALVENARIA = 0;
    static final int VINIL = 1;
    static final int FIBRA = 2;
    static final int PLASTICO = 3;
    static double[] precos = {1500, 1100, 750, 500};
    static char[][] nomes =
        {{'A','l','v','e','n','a','r','i','a'},
         {'V','i','n','i','l'}, {'F','i','b','r','a'},
         {'P','l','á','s','t','i','c','o'}};

    static double area(double raio) {
        return((raio >= 0) ? Math.PI * Math.pow(raio,2) : -1);
    }

    static double valor(double area, int material) {
        if (material<ALVENARIA || material>PLASTICO || area<0)
            return(-1);
        return(area*precos[material]);
    }

    public static void carregaVal(double[][] m) {
        for (int i=0; i<m.length; i++) {
            for (int j=50; j<=200; j+=50) {
                m[i][j / 50 - 1] = precos[i] * j;
            }
        }
    }
}
```

```
public static double[][] calculaFinal(
    double[][] val,double[][] desc) {
    double[][] saida = new double[val.length]
        [val[0].length];

    for (int i=0; i<saida.length; i++) {
        for (int j=0; j<saida[0].length; j++)
            saida[i][j] = val[i][j] * (1-desc[i][j]);
    }
    return(saida);
}

public static void main(String[] args) {
}
}
```

- AreaPiscina.java

- Também possui seu próprio main

Classes em Java

- Uma classe pode ter vários métodos e atributos

Classes em Java

- Uma classe pode ter vários métodos e atributos
- Costumamos colocar em uma mesma classe somente os métodos e atributos afins

Classes em Java

- Uma classe pode ter vários métodos e atributos
- Costumamos colocar em uma mesma classe somente os métodos e atributos afins
 - Atributos e métodos relativos ao cálculo da área da casa em `AreaCasa`

Classes em Java

- Uma classe pode ter vários métodos e atributos
- Costumamos colocar em uma mesma classe somente os métodos e atributos afins
 - Atributos e métodos relativos ao cálculo da área da casa em `AreaCasa`
 - Atributos e métodos relativos ao cálculo da área da piscina em `AreaPiscina`

Classes em Java

- Classes diferentes podem inclusive ter métodos e atributos homônimos

Classes em Java

- Classes diferentes podem inclusive ter métodos e atributos homônimos

```
class AreaCasa {
```

```
}
```

```
class AreaPiscina {
```

```
}
```

Classes em Java

- Classes diferentes podem inclusive ter métodos e atributos homônimos

```
class AreaCasa {  
    static void area(float lateral,  
                    float cquarto)  
    ...  
}
```

```
class AreaPiscina {  
    static double area(double raio)  
    ...  
}
```

Classes em Java

- Classes diferentes podem inclusive ter métodos e atributos homônimos

```
class AreaCasa {  
    static void area(float lateral,  
                    float cquarto)  
    ...  
  
    static double valor(double area)  
    ...  
  
}
```

```
class AreaPiscina {  
    static double area(double raio)  
    ...  
  
    static double valor(double area,  
                        int material)  
    ...  
  
}
```

Classes em Java

- Classes diferentes podem inclusive ter métodos e atributos homônimos

```
class AreaCasa {  
    static void area(float lateral,  
                    float cquarto)  
    ...  
  
    static double valor(double area)  
    ...  
  
    public static void main(String[]  
                           args)  
    ...  
}
```

```
class AreaPiscina {  
    static double area(double raio)  
    ...  
  
    static double valor(double area,  
                        int material)  
    ...  
  
    public static void main(String[]  
                           args)  
    ...  
}
```


Classes em Java

- Classes diferentes podem inclusive ter métodos e atributos homônimos

```
class AreaCasa {  
    static void area(float lateral,  
                    float cquarto)  
    ...  
  
    static double valor(double area)  
    ...  
  
    public static void main(String[]  
                            args)  
    ...  
}
```

```
class AreaPiscina {  
    static double area(double raio)  
    ...  
  
    static double valor(double area,  
                        int material)  
    ...  
  
    public static void main(String[]  
                            args)  
    ...  
}
```

- A diferença se dá em que area de AreaCasa tem significado diferente de area de AreaPiscina

Classes em Java

- E como usamos essas classes?

Classes em Java

- E como usamos essas classes?
- Fazemos antes uma mudança em AreaCasa:

```
class AreaCasa {  
    static double valorM2 = 1500;  
  
    static double area(double lateral,  
                        double cquarto){  
        double areat=-1;  
        if (lateral>=0 && cquarto>=0) {  
            areat = lateral*lateral;  
            areat += cquarto*lateral;  
        }  
        return(areat);  
    }  
  
    static double valor(double area) {  
        if (area >= 0) return(valorM2*area);  
        return(-1);  
    }  
}
```

Classes em Java

- E como usamos essas classes?
- Fazemos antes uma mudança em AreaCasa:
 - Fazemos retornar a área

```
class AreaCasa {  
    static double valorM2 = 1500;  
  
    static double area(double lateral,  
                        double cquarto){  
        double areat=-1;  
        if (lateral>=0 && cquarto>=0) {  
            areat = lateral*lateral;  
            areat += cquarto*lateral;  
        }  
        return(areat);  
    }  
  
    static double valor(double area) {  
        if (area >= 0) return(valorM2*area);  
        return(-1);  
    }  
}
```

Classes em Java

- Considere agora uma classe para criação de um projeto arquitetônico:

Classes em Java

- Considere agora uma classe para criação de um projeto arquitetônico:

```
class Projeto {  
    /* Calcula a área de uma casa com piscina */  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```

Classes em Java

- Considere agora uma classe para criação de um projeto arquitetônico:
- Note que ela usa métodos definidos em outras classes (outros arquivos)

```
class Projeto {  
    /* Calcula a área de uma casa com piscina */  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```

Classes em Java

- Considere agora uma classe para criação de um projeto arquitetônico:

```
class Projeto {  
    /* Calcula a área de uma casa com piscina */  
    static double area(double lateral,  
                       double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```

- Note que ela usa métodos definidos em outras classes (outros arquivos)
- Em java podemos chamar métodos de outras classes de dentro de uma classe

Classes em Java

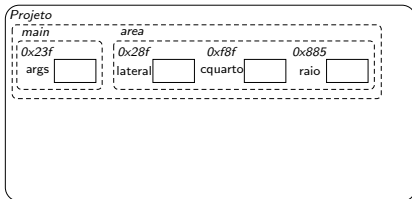
- O que acontece na memória quando executamos esse código?

```
class Projeto {  
    static double area(double lateral, double cquarto,  
                        double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```

Classes em Java

- Espaço é reservado para as variáveis locais e parâmetros de area

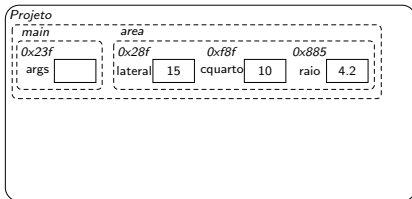
```
class Projeto {  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Classes em Java

- Espaço é reservado para as variáveis locais e parâmetros de area
- Os valores dos parâmetros são copiados para lá

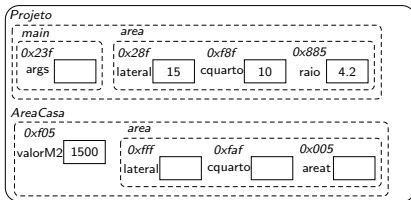
```
class Projeto {  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Classes em Java

- Ao chamar `AreaCasa.area`, espaço é reservado

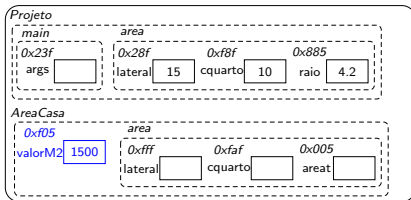
```
class Projeto {  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Classes em Java

- Ao chamar `AreaCasa.area`, espaço é reservado
- Inclusive para atributos da classe → podem ser úteis para algum método dela

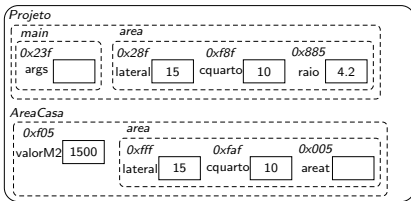
```
class Projeto {  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return (AreaCasa.area(lateral, cquarto) +  
                AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15, 10, 4.2));  
    }  
}
```



Classes em Java

- Ao chamar `AreaCasa.area`, espaço é reservado
- Inclusive para atributos da classe → podem ser úteis para algum método dela
- Os valores são armazenados nos parâmetros

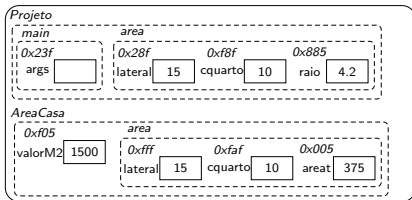
```
class Projeto {  
    static double area(double lateral,  
                        double quarto, double raio) {  
        return (AreaCasa.area(lateral, quarto) +  
                AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Classes em Java

- Ao chamar `AreaCasa.area`, espaço é reservado
- Inclusive para atributos da classe → podem ser úteis para algum método dela
- Os valores são armazenados nos parâmetros
- O cálculo da área é feito

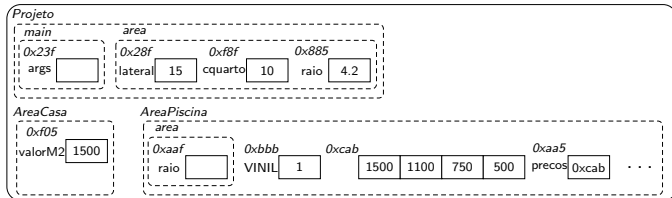
```
class Projeto {  
    static double area(double lateral,  
                        double quarto, double raio) {  
        return(AreaCasa.area(lateral,quarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Classes em Java

- O mesmo acontece com `AreaPiscina.area`

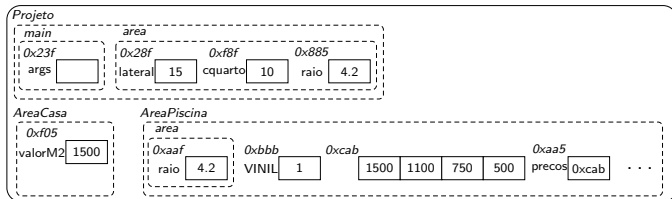
```
class Projeto {  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Classes em Java

- O mesmo acontece com `AreaPiscina.area`

```
class Projeto {  
    static double area(double lateral,  
                        double cquarto, double raio) {  
        return(AreaCasa.area(lateral,cquarto) +  
               AreaPiscina.area(raio));  
    }  
  
    public static void main(String[] args) {  
        System.out.println(area(15,10,4.2));  
    }  
}
```



Referências

- http://www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html
- Brookshear, J.G.: Computer Science: An Overview. 9ed. Addison-Wesley:New York. 2007.
- Reynolds, C.; Tymann, P.: Principles of Computer Science. 1ed. McGraw-Hill:New York. 2008.
- Goldman, A.; Kon, F.; Silva, P.J.S.: Introdução à Ciência da Computação com Java e Orientação a Objetos. 1ed. IME-USP:São Paulo. 2006.

<https://www.youtube.com/watch?v=RMIrqhSivso>

(cobre parcialmente o conteúdo)