Aula 14 – Paradigmas de Programação e Orientação a Objetos

Norton Trevisan Roman

14 de abril de 2011

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

- 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
- Esse é o paradigma imperativo

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

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
- 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

- 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
- 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

- 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
- 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

- 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
- 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
- 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
- Baseado na matemática e na teoria das funções
- 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
- Baseado na matemática e na teoria das funções
- 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
- Baseado na matemática e na teoria das funções
- Todos os cálculos são feitos pela aplicação de funções
  - Não há variáveis globais, apenas parâmetros e variáveis locais
- 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
- Baseado na matemática e na teoria das funções
- Todos os cálculos são feitos pela aplicação de funções
  - Não há variáveis globais, apenas parâmetros e variáveis locais
- 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
  - Tecnicalidades 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
  - Tecnicalidades 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
  - Tecnicalidades 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
  - Tecnicalidades 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
  - Tecnicalidades 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
  - Tecnicalidades 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

- Baseia-se no agrupamento de aspectos do programa, separando-os do resto
  - Conforme conceitos importantes para o domínio de interesse
  - Tecnicalidades 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
- Java, C++ (compatíveis com imperativo)

Paradigma Orientado a Objetos

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

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

Paradigma Orientado a Objetos

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

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

Paradigma Orientado a Objetos

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

- 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

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

- 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

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

- 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

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

- 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

- Imagine uma partida de futebol

Classes e Objetos

- Imagine uma partida de futebol

Papel	*Número de Atores*
Técnico	2
Árbitro	
Juiz	1
Auxiliar de Arbitragem	2
Jogador	
Goleiro	2
Meio	20

Classes e Objetos

- Imagine uma partida de futebol

Papel	*Número de Atores*
Técnico	2
Árbitro	
Juiz	1
Auxiliar de Arbitragem	2
Jogador	
Goleiro	2
Meio	20

- Todos com papéis, ações e atributos

Classes e Objetos

- Papel: Técnico

Classes e Objetos

- Papel: Técnico
  - Atributos: Time

Classes e Objetos

- Papel: Técnico
  - Atributos: Time
  - Ações:

Classes e Objetos

- Papel: Técnico
  - Atributos: Time
  - Ações:
    - VenceuOTimeX:
      se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      caso contrário, diga que o seu time foi prejudicado pela arbitragem e que futebol é uma caixinha de surpresas.

Classes e Objetos

- Papel: Técnico
  - Atributos: Time
  - Ações:
    - VenceuOTimeX:
      se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      caso contrário, diga que o seu time foi prejudicado pela arbitragem e que futebol é uma caixinha de surpresas.
    - Pitaco: Grite desesperadamente instruções ao seu time

Classes e Objetos

- Papel: Árbitro

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz
  - Ações:

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz
  - Ações:
    - ★ Irregularidade Identificada:
      Se identificou uma irregularidade, páre o jogo

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz
  - Ações:
    - ★ Irregularidade Identificada:
      Se identificou uma irregularidade, páre o jogo
    - ★ Irregularidade Apontada:
      Se um auxiliar apontou irregularidade, páre o jogo

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz
  - Ações:
    - ★ Irregularidade Identificada:
      Se identificou uma irregularidade, páre o jogo
    - ★ Irregularidade Apontada:
      Se um auxiliar apontou irregularidade, páre o jogo
- Sub-Papel: Auxiliar de Arbitragem

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz
  - Ações:
    - ★ Irregularidade Identificada:
      Se identificou uma irregularidade, páre o jogo
    - ★ Irregularidade Apontada:
      Se um auxiliar apontou irregularidade, páre o jogo
- Sub-Papel: Auxiliar de Arbitragem
  - Ações:

Classes e Objetos

- Papel: Árbitro
  - Atributos: N/D
- Sub-papel: Juiz
  - Ações:
    - ★ Irregularidade Identificada:
      Se identificou uma irregularidade, páre o jogo
    - ★ Irregularidade Apontada:
      Se um auxiliar apontou irregularidade, páre o jogo
- Sub-Papel: Auxiliar de Arbitragem
  - Ações:
    - ★ Irregularidade Identificada:
      Se identificou uma irregularidade, sinalize ao juiz

Classes e Objetos

- Papel: Jogador

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - ★ VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro
  - Ações:

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro
  - Ações:
    - Defender: Vindo a bola para o goleiro, pegue

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro
  - Ações:
    - Defender: Vindo a bola para o goleiro, pegue
- Sub-papel: Meio

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro
  - Ações:
    - Defender: Vindo a bola para o goleiro, pegue
- Sub-papel: Meio
  - Ações:

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro
  - Ações:
    - Defender: Vindo a bola para o goleiro, pegue
- Sub-papel: Meio
  - Ações:
    - Atacar: Leve a bola até dentro do gol

Classes e Objetos

- Papel: Jogador
  - Atributos: Time, Número da camisa
  - Ações:
    - VenceuOTimeX:
      Se TimeX é igual ao atributo Time no seu cartão de identificação, comemore;
      Caso contrário, diga que não era esperado e que o time está de parabéns
- Sub-papel: Goleiro
  - Ações:
    - Defender: Vindo a bola para o goleiro, pegue
- Sub-papel: Meio
  - Ações:
    - Atacar: Leve a bola até dentro do gol
    - Recuperar: Estando a bola com alguém do outro time, recupere-a

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

- Classes:

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

- Classes:
  - Agrupamento de entidades que possuem alguns atributos e métodos em comum

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

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

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

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

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

- Classes:
  - Agrupamento de entidades que possuem alguns atributos e métodos em comum
  - Representam o modelo por trás do problema – a idéia
- Objetos:
  - Representantes das classes

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

- Classes:
  - Agrupamento de entidades que possuem alguns atributos e métodos em comum
  - Representam o modelo por trás do problema – a idéia
- Objetos:
  - Representantes das classes
  - Constituem as entidades computacionais que representam as classes

Classes e Objetos

Futebol	*Orientação a Objetos*
Papel	Classe
Subpapel	Subclasse
Ator	Objeto
Ação	Método
Atributo	Atributo

- Classes:
  - Agrupamento de entidades que possuem alguns atributos e métodos em comum
  - Representam o modelo por trás do problema – a idéia
- Objetos:
  - Representantes das classes
  - Constituem as entidades computacionais que representam as classes
    - ★ São a implementação física da classe

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - Atributos e métodos locais a um grupo de entidades ficam assim escondidos do resto do problema

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - Atributos e métodos locais a um grupo de entidades ficam assim escondidos do resto do problema
- Subclasses

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - 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

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - 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
- Ex: Juiz e Auxiliar de arbitragem possuem atributos e métodos em comum:

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - 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
- Ex: Juiz e Auxiliar de arbitragem possuem atributos e métodos em comum:
  - Objetivos no jogo

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - 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
- Ex: Juiz e Auxiliar de arbitragem possuem atributos e métodos em comum:
  - Objetivos no jogo
  - Ações a serem tomadas em algumas circunstâncias

Encapsulamento e Subclasses

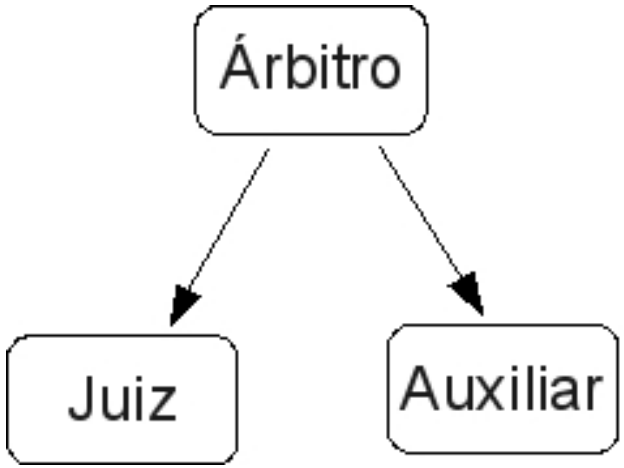
- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - Atributos e métodos locais a um grupo de entidades ficam assim escondidos do resto do problema
- Subclasses
  - São sub-agrupamentos de entidades distintas, mas que possuem alguns atributos em comum
- Ex: Juiz e Auxiliar de arbitragem possuem atributos e métodos em comum:
  - Objetivos no jogo
  - Ações a serem tomadas em algumas circunstâncias
    - ★ Como o protocolo no início e fim do jogo

Encapsulamento e Subclasses

- Ao ato de agrupar entidades que contenham atributos e métodos em comum damos o nome de encapsulamento
  - 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
- Ex: Juiz e Auxiliar de arbitragem possuem atributos e métodos em comum:
  - Objetivos no jogo
  - Ações a serem tomadas em algumas circunstâncias
    - ★ Como o protocolo no início e fim do jogo
- Como também possuem características próprias, cada um possui um grupo (classe), sendo esses grupos agrupados em uma outro grupo (superclasse)

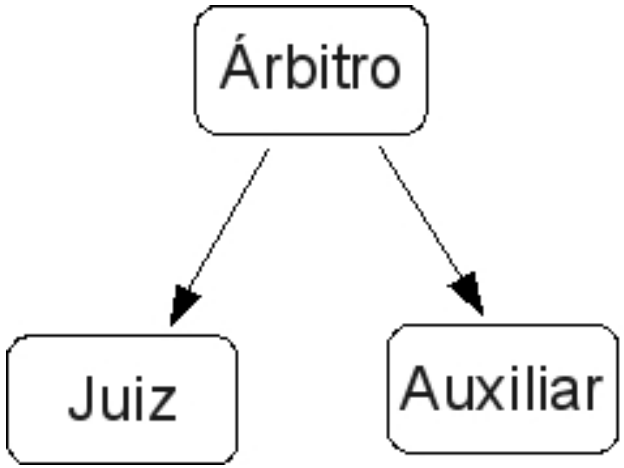
Classes e Subclasses

- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos



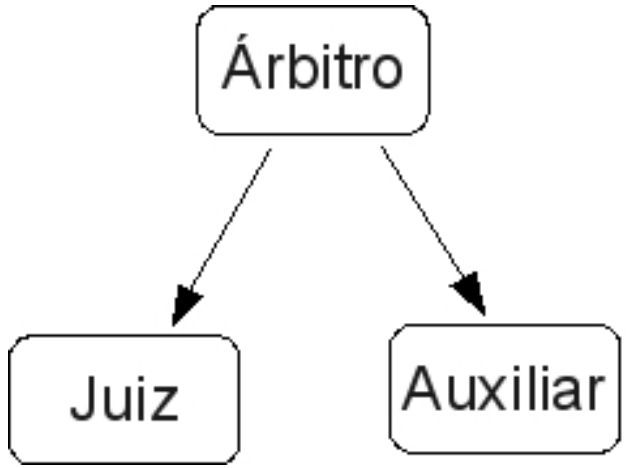
Classes e Subclasses

- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos
  - Por isso possuem uma classe própria



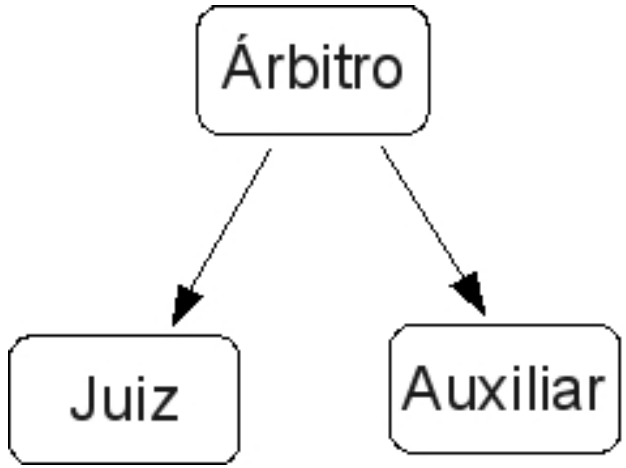
Classes e Subclasses

- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos
  - Por isso possuem uma classe própria
- Contudo, também possuem muito em comum



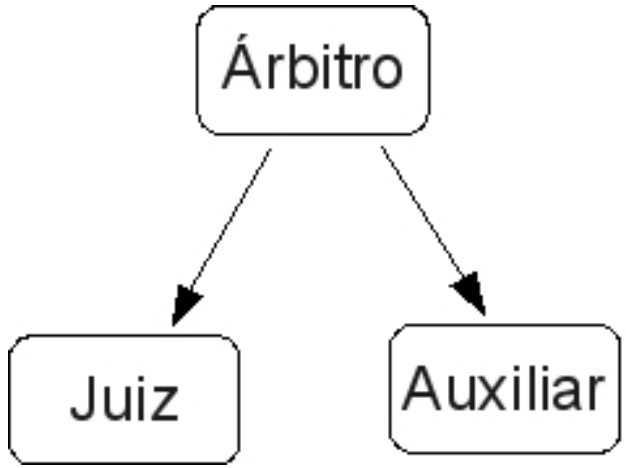
Classes e Subclasses

- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos
  - Por isso possuem uma classe própria
- Contudo, também possuem muito em comum
  - São parte de uma classe maior – Árbitro



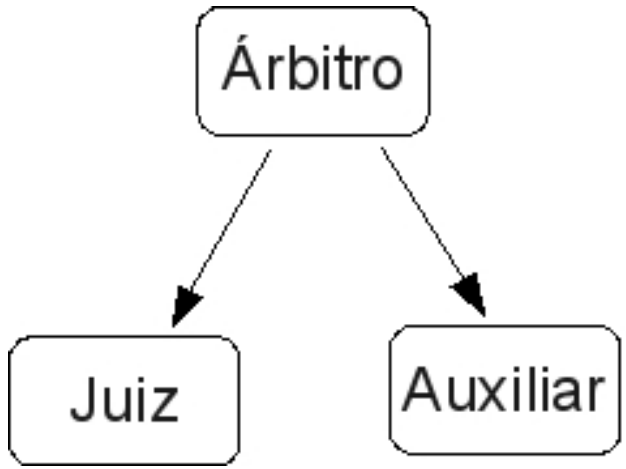
Classes e Subclasses

- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos
  - Por isso possuem uma classe própria
- Contudo, também possuem muito em comum
  - São parte de uma classe maior – Árbitro
- Juiz e Auxiliar são subclasses de Árbitro



Classes e Subclasses

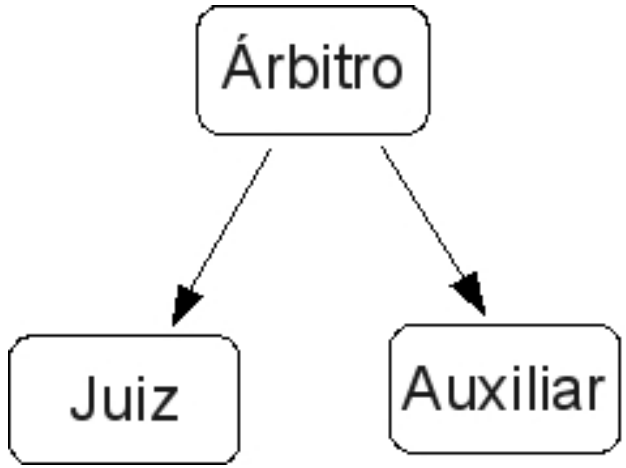
- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos
  - Por isso possuem uma classe própria
- Contudo, também possuem muito em comum
  - São parte de uma classe maior – Árbitro



- Juiz e Auxiliar são subclasses de Árbitro
- Árbitro é superclasse de juiz e Auxiliar

Classes e Subclasses

- Tanto Juiz quanto Auxiliar possuem atributos e métodos distintos
  - Por isso possuem uma classe própria
- Contudo, também possuem muito em comum
  - São parte de uma classe maior – Árbitro



- Juiz e Auxiliar são subclasses de Árbitro
- Árbitro é superclasse de juiz e Auxiliar
- A mesma relação se dá entre Jogador e Goleiro e Meio

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 possuisse 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 possuisse 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 possuisse 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 possuisse 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

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo
  - Irregularidade Identificada:

    Se identificou uma irregularidade, sinalize ao juiz

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo
- Como se dá essa sinalização?
  - Irregularidade Identificada:

    Se identificou uma irregularidade, sinalize ao juiz

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo
  - Irregularidade Identificada:

    Se identificou uma irregularidade, sinalize ao juiz
- Como se dá essa sinalização?
  - Em seu método "Irregularidade Identificada" o Auxiliar faz uma chamada ao método "Irregularidade Apontada" do Juiz

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo
  - Irregularidade Identificada:

    Se identificou uma irregularidade, sinalize ao juiz
- Como se dá essa sinalização?
  - Em seu método "Irregularidade Identificada" o Auxiliar faz uma chamada ao método "Irregularidade Apontada" do Juiz
  - Como se passasse uma mensagem – métodos também são encarados como mensagens

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo
  - Irregularidade Identificada:

    Se identificou uma irregularidade, sinalize ao juiz
- Como se dá essa sinalização?
  - Em seu método "Irregularidade Identificada" o Auxiliar faz uma chamada ao método "Irregularidade Apontada" do Juiz
  - Como se passasse uma mensagem – métodos também são encarados como mensagens
  - O comportamente frente às mesmas mensagens não necessariamente é igual:

Mensagens

- Considere as ações executadas por Auxiliar e Juiz:
  - Irregularidade Identificada:

    Se identificou uma irregularidade, páre o jogo
  - Irregularidade Apontada:

    Se um auxiliar apontou irregularidade, páre o jogo
  - Irregularidade Identificada:

    Se identificou uma irregularidade, sinalize ao juiz
- Como se dá essa sinalização?
  - Em seu método "Irregularidade Identificada" o Auxiliar faz uma chamada ao método "Irregularidade Apontada" do Juiz
  - Como se passasse uma mensagem – métodos também são encarados como mensagens
  - O comportamente frente às mesmas mensagens não necessariamente é igual:
    - ★ "Irregularidade Identificada" corresponde a diferentes ações, dependendo se a classe é Juiz ou Auxiliar

Pondo os Pés no Chão

- Voltando à área... Considere os métodos desenvolvidos até agora:

```
/* 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 metragem,
                           int material)

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

/* 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 área... Considere os métodos desenvolvidos até agora:

```
/* 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 metragem,
                           int material)

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

/* 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 entre eles?

Pondo os Pés no Chão

- Voltando à área... Considere os métodos desenvolvidos até agora:

```
/* 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 metragem,
                           int material)

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

/* 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 entre eles?
  - Métodos relativos à casa...

Pondo os Pés no Chão

- Voltando à área... Considere os métodos desenvolvidos até agora:

```
/* 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 metragem,
                           int material)

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

/* 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 entre eles?
  - Métodos 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 m^2 dos materiais

Então...

- Casa
  - Valor do metro quadrado
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais
  - Métodos:

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
- Piscina
  - Tipos de materiais
  - Preços do m^2 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 m^2 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 m^2 dos materiais
  - Nomes dos materiais
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total, e métodos auxiliares

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total, e métodos auxiliares
- Podemos simplesmente criar duas classes para acomodar essas diferenças

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total, e métodos auxiliares
- Podemos simplesmente criar duas classes para acomodar essas diferenças
- E como ficaria o cálculo envolvendo casa e piscina?

Então...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - Cálculo da área
    - Cálculo do valor total
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais
  - Métodos:
    - Cálculo da área
    - Cálculo do valor total, e métodos auxiliares
- 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...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total, e métodos auxiliares
- 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...

- Casa
  - Valor do metro quadrado
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total
- Piscina
  - Tipos de materiais
  - Preços do m^2 dos materiais
  - Nomes dos materiais
  - Métodos:
    - ★ Cálculo da área
    - ★ Cálculo do valor total, e métodos auxiliares
- 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 Piscina dentro dela

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.