ACH 2003 - Computação Orientada a Objetos

Coleções Java

Prof. Flávio Luiz Coutinho
flcoutinho@usp.br

Coleções Java (*Java Collections Framework*)

Conjunto de classes que implementam diversas estruturas de dados clássicas (listas, filas, pilhas, árvores, tabelas de espalhamento, etc), organizadas em uma arquitetura unificada.

- Não é preciso “reinventar a roda” (processo sujeito a falhas).
- Estruturas implementadas da forma mais eficiente possível, levando em conta as características de cada uma.
- A arquitetura unificada permite a troca de uma estrutura por outra, causando o mínimo de impacto ao código.

Coleções Java (*Java Collections Framework*)

Todas as estruturas implementadas no *framework* são dinâmicas (isto é, crescem de acordo com a demanda de armazenamento), e permitem (ao menos):

- Adicionar elementos à estrutura.
- Remover elementos da estrutura.
- Verificar a existência de elementos na estrutura.
- Percorrer os elementos armazenados na estrutura (iteração).

Além disso, todas as implementações são genéricas, de modo que elas são capazes de armazenar elementos de qualquer tipo, e há verificação de segurança de tipo durante a compilação.

Coleções Java (*Java Collections Framework*)

O *framework* é composto por um conjunto de classes e interfaces.

Principais interfaces:

- Collection — comportamentos básicos que toda estrutura deve ter
- List — armazenam os elementos de forma linear
- Set — não aceitam elementos repetidos / não definem ordem
- SortedSet — elementos organizados de acordo com seus valores

- Map — estruturas que guardam “valores” associados a “chaves”

Coleções Java (*Java Collections Framework*)

O *framework* é composto por um conjunto de classes e interfaces.

Principais classes:

- ArrayList — lista linear baseada em vetor (*array*)
- LinkedList — lista ligada
- HashSet — tabela de espalhamento (tabela de *hashing*)
- TreeSet — árvore de busca binária balanceada

- HashMap — chaves organizadas em uma tabela de espalhamento
- TreeMap — chaves organizadas em uma árvore de busca binária.

Coleções Java (*Java Collections Framework*)

Interface **Iterator**:

- Oferece um mecanismo uniforme de iteração, que independe do tipo da coleção (e da estrutura de dado implementada).
- Quando queremos iterar pelos elementos de uma coleção, obtemos a partir desta um objeto iterador (através do método **iterator**).
- Este objeto será o responsável por gerenciar o processo de iteração entre os elementos armazenados na coleção. Oculta dos usuários do *framework* a organização interna das estruturas de dados.
- O processo de iteração ocorre pelo uso coordenado de dois métodos do objeto iterador: **hasNext** e **next**.

Coleções Java (*Java Collections Framework*)

Exemplos:

- Operações básicas: adição, iteração, verificação, remoção.
- Implementações: ArrayList, LinkedList, HashSet, TreeSet.

- Coleção de inteiros
- Coleção de strings
- Coleção de alunos*

- Quais os efeitos percebidos ao se trocar o tipo de uma coleção?

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.
- afinal, quem decide se um objeto mantido na coleção e outro passado como parâmetro ao **contains** são iguais ou diferentes?

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.
- afinal, quem decide se um objeto mantido na coleção e outro passado como parâmetro ao **contains** são iguais ou diferentes?
    - Seria a implementação da estrutura? Faz sentido?

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.
- afinal, quem decide se um objeto mantido na coleção e outro passado como parâmetro ao **contains** são iguais ou diferentes?
    - Seria a implementação da estrutura? Faz sentido?
    - Ou esta decisão deve ser “terceirizada”? Por quem?

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.
- afinal, quem decide se um objeto mantido na coleção e outro passado como parâmetro ao **contains** são iguais ou diferentes?
  - Seria a implementação da estrutura? Faz sentido?
  - Ou esta decisão deve ser “terceirizada”? Por quem?

- Método equals()

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.
- afinal, quem decide se um objeto mantido na coleção e outro passado como parâmetro ao **contains** são iguais ou diferentes?
  - Seria a implementação da estrutura? Faz sentido?
  - Ou esta decisão deve ser “terceirizada”? Por quem?

- Método equals(), hashCode()

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- por que as chamadas a **contains** não funcionam como esperado?
- instâncias diferentes com mesmo conteúdo são consideradas diferentes.
- afinal, quem decide se um objeto mantido na coleção e outro passado como parâmetro ao **contains** são iguais ou diferentes?
  - Seria a implementação da estrutura? Faz sentido?
  - Ou esta decisão deve ser “terceirizada”? Por quem?

- Método equals(), hashCode(), compareTo()

Coleções Java (*Java Collections Framework*)

Exemplos: coleção de alunos

- Rastreio das chamadas dos métodos equal(), hashCode(), compareTo()
- Confirma o funcionamento esperado de cada tipo de coleção.
- Apesar de todas as coleções implementarem o mesmo conjunto básico de operações, a eficiência de cada operação irá variar conforme o tipo da coleção.
- A coleção ideal dependerá do problema que queremos resolver.
- Mesmo que não venhamos a implementar “na mão” estruturas de dados no dia-a-dia, conhecimento pleno sobre como funcionam e a complexidade assintótica de cada operação é fundamental (IAA, AED1, AED2).