UFMS DIGITAL

Programação Orientada a Objetos

Prof. Dr. Anderson V. de Araujo

UFMS DIGITAL

Módulo IV - Conceitos avançados

Unidade II - Conjuntos e Mapas

A Interface Set

- Representa um agrupamento de elementos (de qualquer tipo);
- Mesmas características do Set ADT:
  - Similar ao conceito de conjuntos finitos na matemática;
  - Não há noção de ordem, posição dos elementos;
  - Não aceita elementos repetidos.
- Existem duas principais implementações:
  - **HashSet**: os elementos não ficam ordenados (mais comum e mais rápido);
  - **TreeSet**: os elementos ficam ordenados, independente de como foram adicionados.

Exemplo Set - HashSet

```
public static void main(String[] args) {
   Set set = new HashSet();//Ou TreeSet, mesmo resultado
   set.add("B");
   set.add("1");
   set.add("C");
   set.add("E");
   set.add("3");
   set.add("A");
   set.add("A");

    for (Object obj : set){
      String str = (String) obj;//cast necessário
      System.out.print(str);
}}
```

Saída: 1ABC3E

Como visto, esta classe não permite duplicatas e não mantém a sequência de inserção dos elementos.

Teste com TreeSet!

A Interface Map

- Semelhante a um array, mas ao invés de índices numéricos, usa objetos como chaves;
- Mesmas características do Map ADT:
  - Cada entrada possui:
    - Uma chave/key (de qualquer tipo): serve para indexar os elementos, ou seja, serve para achar um elemento rapidamente. Deve ser comparável;
    - Um valor/value (de qualquer tipo): o valor armazenado na fila e associado a chave.
  - As chaves são únicas, porém os valores podem ser duplicados:
    - Caso haja uma repetição de chave, o elemento antigo é sobreposto pelo novo.
  - Também chamado de `dictionary`.

A Interface Map (2)

- Principais classes que implementam a interface:
  - `HashMap`: não sincronizado e não ordenado, aceita `null` como chave (mais comum);
  - `HashTable`: igual ao `HashMap`, porém sincronizado e não aceita `null` como chave;
  - `TreeMap`: não sincronizado mas é ordenado.

Exemplo Map (HashMap)

```
public static void main(String[] args) {
   Map nomes = new HashMap();
   nomes.put("João", 35);
   nomes.put("José", 22);
   nomes.put("Maria", 76);
   nomes.put("Bianca", 18);

   System.out.println("Quantidade: "+nomes.size());

   int idadeMaria = (int)nomes.get("Maria");//cast necessário
    System.out.print(idadeMaria);
}
```

- Inicialmente será impresso 4, que é o tamanho total do Mapa criado;
- Em seguida, será impresso: “76” que é o objeto (int) correspondente à chave “Maria”;
- Não dá pra usar o foreach direto (só se for na Entry).

Collections + Generics

- É possível definir uma coleção ou mapa restrito a um tipo específico de objeto:
  - A partir do Java 1.5.
- Pode ser chamado de polimorfismo paramétrico;
- Desta forma, fica inviável adicionar elementos à coleção que não sejam do tipo especificado;
- Com isso, temos:
  - Maior nível de segurança;
  - Evitar o *cast*.

Collections + Generics - Criação

- Ao ser criado, informamos o tipo do elemento que pode ser armazenado na coleção:
  - `Set<String> nomes = new HashSet<String>();`
  - `List<Cliente> clientes = new ArrayList<>();` (notação *diamond*)
- Isto porque a coleção é implementada com tipos genéricos, onde o tipo é determinado no momento da execução (*Runtime*).

Collections + Generics - Exemplo

```
List<ContaCorrente> contas = new ArrayList<ContaCorrente>();
contas.add(c1);
contas.add(c3);
contas.add(c2);
```

```
for(int i = 0; i < contas.size(); i++) {
   ContaCorrente cc = contas.get(i); // sem cast!
   System.out.println(cc.getSaldo());
}
```

Não há necessidade do cast, por exemplo:

```
ContaCorrente cc = (ContaCorrente) contas.get(i);
```

For-each + Generics

- Exemplo:
  - `for (String nome: listaNomes) { ... }`
    - Onde listaNomes é uma coleção do tipo String:

      `Collection<String> listaNomes`
- Não dá pra usar direto para Maps
  - Precisa pegar para as chaves (keySet) ou valores (values)

Exemplo - ForEach

```
public static void main(String[] args) {
  Collection<String> nomes = new ArrayList<>();//LinkedList,...
  System.out.println("Qtd elementos: "+nomes.size());

  for (String str: nomes) {
     System.out.println(str);
  }
}
```

- Será impresso o tamanho total da estrutura do conjunto criado
- Em seguida, por meio do foreach, serão impressos todos os elementos (Strings) que pertencem à coleção criada
  - Que pode ser qualquer coleção de objetos

Licenciamento