Linguagem Java

Continuação do estudo sobre coleções de dados

Set e HashSet

Map e HashMap

Exercício

- Continuação da resolução do exercício 13 da ficha

13. Pretende-se uma aplicação para gerir os livros de uma biblioteca. Os livros são identificados por um código (um número inteiro positivo que representa a ordem de criação do registo dos livros na biblioteca). O registo de um livro, para além do referido código, tem obrigatoriamente informação sobre o título e os autores.

...

d. Desenvolva uma nova versão da classe `Library` recorrendo a um objeto `HashSet` para gerir o conjunto de livros.
e. Desenvolva uma nova versão da classe `Library` recorrendo a um objeto `HashMap` para gerir o conjunto de livros.

- Para a versão com HashMap usar o código do livro como chave

Set<E>

- Um conjunto, `Set`, permite o armazenamento e gestão de uma série de elementos não repetidos
  - Dois itens, `i1` e `i2`, são considerados repetidos quando a comparação dos dois através do método `equals`, `i1.equals(i2)`, retorna o valor `true`
  - É imprescindível que os métodos `hashCode` estejam devidamente implementados para que as operações sobre um `Set` conduzam aos resultados esperados
    - Caso a indexação dos elementos não seja feita corretamente, as procuras de elementos iguais não funcionará e, consequentemente, o mecanismo que garante a não existência de elementos repetidos não funcionará

Set<E> e HashSet<E>

- A interface `Set<E>` define o protocolo associado a este tipo de coleção
  - `add, addAll, clear, contains, containsAll, isEmpty, iterator, remove, removeAll, size, toArray, ...`
- Não é garantida a manutenção da ordem de inserção dos elementos de um `Set`
- O acesso aos elementos é realizado através de um processo iterativo sobre os elementos
- Existem várias implementações desta interface sendo uma das mais usadas a `HashSet<E>`
  - Outros: `EnumSet, TreeSet, LinkedHashSet,...`

HashSet e hashCode

```
class Item {
    private static int count = 0;
    int i;

    public Item(int i) { this.i = i; }

    @Override
    public int hashCode() { return i; }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        Item other = (Item) obj;
        if (i != other.i) return false;
        return true;
    }
}

public class App {
    public static void main(String[] args) {
        HashSet<Item> set = new HashSet<>();
        System.out.println(set.add(new Item(1))); // true
        System.out.println(set.add(new Item(2))); // true
        System.out.println(set.add(new Item(1))); // false
        System.out.println(set.size()); // 2
    }
}
```

```
class Item {
    private static int count = 0;
    int i;

    public Item(int i) { this.i = i; }

    @Override
    public int hashCode() { return count++; }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        Item other = (Item) obj;
        if (i != other.i) return false;
        return true;
    }
}

public class App {
    public static void main(String[] args) {
        HashSet<Item> set = new HashSet<>();
        System.out.println(set.add(new Item(1))); // true
        System.out.println(set.add(new Item(2))); // true
        System.out.println(set.add(new Item(1))); // true
        System.out.println(set.size()); // 3
    }
}
```

Map<K,V>

- Um mapa, `Map`, permite o armazenamento e gestão de pares atributo-valor (*key-value*)
  - As chaves, K, e os valores, V, podem ser de qualquer tipo não primitivo
  - Não podem existir chaves repetidas, mas os valores podem ser repetidos

- A interface `Map<K,V>` define o protocolo associado a este tipo de coleção
  - `clear, containsKey, containsValue, get, getOrDefault, isEmpty, keySet, put, putAll, putIfAbsent, remove, replace, size, values, ...`

- Se foi indicado um novo valor para uma chave já existente (através do `put` ou `replace`) o valor antigo será substituído pelo novo (o valor anterior é retornado em ambas as funções indicadas)

Map<K,V> e HashMap<K,V>

- Existem várias implementações da interface Map sendo uma das mais usadas a HashMap<K,V>
  - Outros: Hashtable, EnumMap, TreeMap, LinkedHashMap,...

```
HashMap<String,Integer> colors = new HashMap<>();

colors.put("Orange",0xFF8C00);  // 0xFF8C00==16747520
colors.put("Yellow",0xFFFF00);  // 0xFFFF00==16776960
System.out.println(colors);     // Output: {Yellow=16776960, Orange=16747520}

colors.put("Orange",0xFF6500);  // 0xFF6500==16737536
System.out.println(colors);     // Output: {Yellow=16776960, Orange=16737536}

System.out.println(colors.keySet()); // Output: [Yellow, Orange]
System.out.println(colors.values()); // Output: [16776960, 16737536]
```