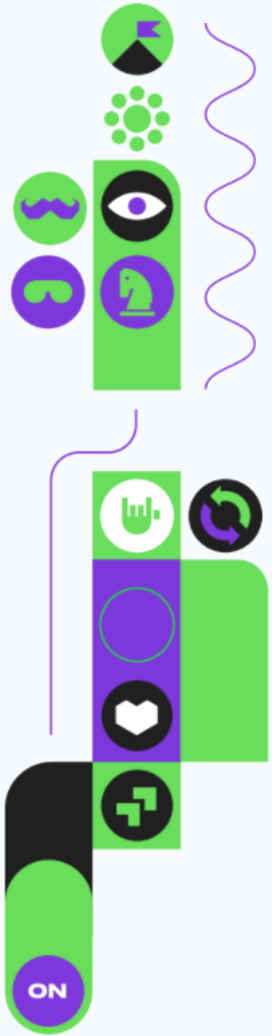

Coleções

GAMA ACADEMY

Classes empacotadoras de tipo

Wrappers

Todo tipo primitivo tem uma classe empacotadora de tipo correspondente (no pacote java.lang).

Essas classes chamam-se **Boolean, Byte, Character, Double, Float, Integer, Long e Short**.

Elas permitem manipular valores de tipo primitivo como objetos. Isso é importante porque as estruturas de dados Set, Queue e List (e suas classes derivadas) manipulam e compartilham objetos elas não podem manipular variáveis de tipos primitivos. Mas podem manipular objetos das classes empacotadoras de tipo(Wrappers), porque cada classe em última análise deriva de Object.

Autoboxing e auto-unboxing

O boxing converte um valor primitivo em um objeto da classe empacotadora de tipo correspondente. O unboxing converte um objeto empacotador de tipo no valor primitivo correspondente.

- O Java executa automaticamente conversão boxing e unboxing.

Autoboxing
```
Integer i = 1;
System.out.println(i instanceof Integer); // true
System.out.println(i.getClass().getSimpleName()); // Integer
```

Equals

O método equals é utilizado para comparações.

Classes por referência sobrescrevem equals() para garantir que dois objetos analisados, com o mesmo conteúdo, possam ser considerados iguais. E, quando a classe a qual os objetos em questão pertencem não sobrescreve o método Equals(), o método Object.Equals() será chamado.

Quando comparamos objetos, é considerada uma boa prática utilizarmos o método Equals() para fazer a comparação de igualdade.

Collections

A Java API fornece várias estruturas de dados pré definidas, chamadas **coleções**, usadas para armazenar grupos de objetos relacionados na memória.

Essas classes fornecem métodos eficientes que organizam, armazenam e recuperam seus dados sem a necessidade de conhecer como os dados são armazenados. Isso reduz o tempo de desenvolvimento de aplicativos.

SET	Não ordenado (por padrão) Não indexado Não aceita obj. duplicados Pode ser heterogêneo Pode Ser homogêneo
LIST	Ordenada Indexada Aceita obj. duplicados Pode ser heterogêneo Pode Ser Homogêneo
QUEUE (Fila)	Implementa fila Fist in/ Fist out(FIFO)

MAP	Chave/Valor Chave não aceita repetição Valor aceita repetição
STACK (pilha)	Implementa pilha(stack) Last in/ Fist out(LIFO)

Interface	Descrição
Collection	A interface-raiz na hierarquia de coleções a partir da qual as interfaces Set, Queue e List são derivadas.
Set	Uma coleção que não contém duplicatas.
List	Uma coleção ordenada que pode conter elementos duplicados.
Queue(fila)	Em geral, uma coleção primeiro a entrar, primeiro a sair que modela uma fila de espera; outras ordens podem ser especificadas.
Map	Uma coleção que associa chaves a valores e que não pode conter chaves duplicadas. Map não deriva de Collection.

Escolhendo uma coleção

A documentação para cada coleção discute os requisitos de memória e as características de desempenho dos métodos para operações como adição e remoção de elementos, pesquisa de elementos, classificação de elementos etc.

Antes de escolher uma coleção, revise a documentação on-line para a categoria da coleção que você está considerando (Set, List, Map, Queue etc.), então selecione a implementação que melhor atende às necessidades de seu aplicativo.

documentação (acesso em 04/2022)

SET

Características - Set

- Velocidade na pesquisa de dados, sendo mais rápida que um objeto do tipo List;
- A inserção de dados é mais lenta;
- Permite trabalhar com conjuntos e pode ser implementado como instâncias das classes HashSet ou TreeSet;
- Não precisa especificar a posição para adicionar um elemento;
- Não aceita valores duplicados. Se caso inserir um registro que já tenha no Set não será adicionado.
- Podem ser implementados como instâncias das classes HashSet ou TreeSet;

Declaração - Set

Quando está construindo objetos na classe Set é necessário informar que tipo de coleção será implementada.

Sintaxe: **Set set = new Type()**;

E - é o objeto declarado, podendo ser classes Wrappers ou tipo de coleção.
Type - é o tipo de objeto da coleção a ser usado;

LIST

Características – List

Uma coleção ordenada (também conhecida como *sequência*). O usuário desta interface tem controle preciso sobre onde na lista cada elemento é inserido.
O usuário pode acessar os elementos por seu índice inteiro (posição na lista) e pesquisar os elementos na lista.
A interface List coloca estipulações adicionais, além das especificadas na interface Collection , nos contratos dos métodos iterador , add , remove , equals e hashCode .

Declaração - List

```
ArrayList<Usuario> lista = new ArrayList<>();
```

QUEUE

A interface Java Queue ordena o elemento de maneira FIFO (First In First Out). No FIFO, o primeiro elemento é removido primeiro e o último elemento é removido por último.

Lembre-se de que uma fila é uma coleção que representa uma fila de espera — normalmente, inserções são feitas na parte de trás de uma fila e exclusões são feitas a partir da frente.

Declaração - Queue

```
Queue<String> fila = new LinkedList<>();
```

MAP

Mapeia chaves para valores. Cada elemento tem na verdade dois objetos: **uma chave e um valor**. Valores podem ser duplicados, mas chaves não. SortedMap é uma interface que estende Map, e permite classificação ascendente das chaves. As chaves em um Map **devem ser únicas**, mas os valores associados **não precisam ser**.

Três das várias classes que implementam a interface Map são Hashtable, HashMap e TreeMap. Hashtables e HashMaps armazenam elementos em tabelas de hash e TreeMaps armazenam elementos em árvores.

Declaração - Map

```
// cria HashMap para armazenar chaves de Strings e valores Integer
Map<String, Integer> mapa = new HashMap<>();
```

Links úteis

Interface de fila em Java
https://www.tutorialcup.com/pt/Java/interface-de-fila-em-java.htm

Obrigada

GAMA ACADEMY