>>>>>>>> 15/05/2022 <<<<<<<<
Trabalhando com Collections Java
Profa. Camila Cavalcante

##

1) Introdução ao Curso
- Apresentação do Curso
List, Set, Map, Stream API

- Visão geral: Collections Framework
Pré requisitos - Java JDK11, IDE IntelliJ, Git, Github
Collection = objeto que agrupa multiplos elementos
(var. primitivas ou objetos) em unidade única.
Antes das Collections, havia o objeto array, mas este
não é tão simples de ser manipulado (deve ser dito logo
no inicio quantos elementos terá, pra aumentar tamanho
é preciso mudar no começo do codigo, complicado excluir
elemento, deve escrever um codigo maior pra descobrir
quais posições estao ocupadas).
Composição de collections (interfaces, implementação ou
classes, algoritmos).
Ex. de interfaces de collections: Set, List, Map
Ex. de implementações de interfaces:
(Set) --> HashSet e LinkedHashSet
(List) --> ArrayList e LinkedList
(Map) --> HashMap e LinkedHashMap
Interface Map não extende da interface Collection, mas
está dentro do universo do framework.
Ex. de metodos de implementações: TreeSet, TreeMap (têm
por baixo a estrutura de árvore binária).
Ex. de objetos: arrays e collections (serão usados vá-
rios métodos estáticos dessas implementações no curso).

Tendo: List<Character>consoantesMeuNome =
Array.asList('c','m','l','c','v','l','c','n','t');
qual saída terá a chamada System.out.println(consoantesMeuNome); ?
Saída: [c,m,l,c,v,l,c,n,t]

[Java Collections: Como utilizar
Collections](https://www.devmedia.com.br/java-collections-como-utilizar-collecti
ons/18450)
[Tutorial de Java Collections Framework - site
Oracle/Sun](https://docs.oracle.com/javase/tutorial/collections/index.html)
[Atalhos do Intellij
Idea](http://www.basef.com.br/index.php/Atalhos_do_IntelliJ_Idea)

Códigos devmedia (estudo complementar)
Teste.java - execução 1 com Tempo total = 379; execução
2 com Tempo total = 526; depois diminuíram mais.
ListaAluno.java - cria lista, add elementos, imprime
Aluno.java - classe criada para modificar ListaAluno,
depois Aluno foi modificada, implementação de interface
Comparable() para possibilitar execução do metodo sort()

em ListaAluno.
ComparaAluno.java - classe pra implementar a interface
Comparator
Plugins do intelliJ = Atom, Git tool box, Key promoter,
Nyan, One dark theme, Rainbow brackets. OK!

- Slides - Introdução
baixei

- Slides - Visão geral sobre Collections
baixei

2) Lists
- Coleções com iterações ordenadas: listas
Caracteristicas, como e quando usar implementações
ArrayList e LinkedList, metodos.
List - java.util; elementos duplicados, ordem de inserção.
Implementação Vector está relacionada a Threads, não quer
dizer que não se sincronizem outras implementações.
Implementação ArrayList só implementa interface List,deve
ser usada onde é necessário mais pesquisa; usa um array
pra armazenar elementos; manipulação mais custosa.
Implementação LinkedList implementa interfaces List,Queue;
deve ser usada onde precisa mais de inserção e exclusão;
usa lista duplamente ligada pra armazenar elementos; leva
menos tempo na manipulação.

- Conhecendo os métodos List - parte 1
Codigo ExemploList.java

- Conhecendo os métodos List - parte 2
Finalizado código ExemploList.java
Consegui fazer o exercício final com implementação de uma
LinkedList, consultei o GeeksForGeeks, W3Schools
[Class
LinkedList<E>](https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.ht
ml)
[LinkedList in
Java](https://www.geeksforgeeks.org/linked-list-in-java/?ref=gcse)
[Java LinkedList](https://www.w3schools.com/java/java_linkedlist.asp)
Código criado ExemploList2.java

- Ordenação de elementos em uma coleção List - parte 1
Código ExemploOrdenacaoList.java
[Get e Set - Métodos Acessores em
Java](https://www.devmedia.com.br/get-e-set-metodos-acessores-em-java/29241)
[Como comparar objetos - Classe abstrata Comparable e o método
compareTo](https://www.javaprogressivo.net/2012/11/Comparando-objetos-Classe-abs
trata-Comparable-metodo-compareTo.html#:~:text=A%20classe%20abstrata%20Comparabl
e%20e,usada%20como%20padr%C3%A3o%20de%20compara%C3%A7%C3%A3o.)
[Interface
Comparable<T>](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.ht
ml)

- Ordenação de elementos em uma coleção List - parte 2
Código ExemploOrdenacaoList.java

- Exercícios Propostos - List
[Iterator Interface In
Java](https://www.geeksforgeeks.org/iterator-interface-in-java/?ref=gcse)
[Java forEach y sus
opciones](https://www.arquitecturajava.com/java-foreach-y-sus-opciones/)
Codigos ExercicioProposto01 e ExercicioProposto02

- Slides
baixei

3) Set
- Coleções com singularidade: Set
Interface Set (pacote java,util)
implementações HashSet, LinkedHashSet, TreeSet
Não permite elementos duplicados, não tem índice.
Precisa bastante da implementação TreeSet para ordenação;
usa TreeMap pra armazenar, mantem ordem natural por padrao,
menor performance que HashSet e LinkedHashSet, não permite
elemento nulo.
HashSet implementa interface Set; usa HashMap pra armazenar
elementos, nao mantem ordem, melhor performance que TreeSet
e LinkedHashSet; permite só 1 elemento nulo.
LinkedHashSet usa LinkedHashMap pra armazenar, mantem ordem
de inserção, performance ente HashSet e TreeSet, permite só
1 elemento nulo.
Ordem de performance das implementações da interface Set:
HashSet>LinkedHashSet>TreeSet

- Conhecendo os métodos Set
Código ExemploSet.java

- Ordenação de elementos em uma coleção Set
Código ExemploOrdenacaoSet.java

- Exercícios Propostos Set
Codigo ExercicioProposto01
[O que é LinkedHashSet em
Java?](https://comozed.com/o-que-%C3%A9-linkedhashset-em-java)
Codigo ExercicioProposto02
[Como usar o toString?](https://blog.cod3r.com.br/como-usar-o-tostring/)

- Slides
baixei!

4) Map
- Coleções de pares: Map
caracteristicas da interface; implementações LinkedHashMap,
HashMap, TreeMap; metodos; dentro do pacote java.util.

Map não estende da interface Collections, mas está dentro do desse framework; é um objeto onde podemos colocar multiplos elementos únicos (key) pra cada value, que podem ser outro objetos e var. primitivas; as chaves são únicas e os valores podem se repetir. Implementações:
Hashtable é thread-safe, acesso sincronizado, outras + modernas
As 3 implementações nao permitem sincronização, mas podemos usar a classe Collections.synchronizedMap(). Trabalho com Map requer sobrescrição de metodos equals() e hashCode().

TreeMap implementa interface NavigableMap, estende de SortedMap e de Map; usado em compração/ordenação do Map; itera elementos em ordem natural; nao permite chave nula; implementa ainda Map e SortedMap; estrutura de dados é arvore vermelha-preta;

LinkedHashMap estende da interface HashMap, ambas implementam a interface Map; itera elementos por ordem de inserção; nao permite chave nula; implementa Map; a estrutura de dados é doubly linked list of buckets;

HashMap itera elementos aleatoriamente; nao permite chave nula; implementa Map; estrutura de dados é list of buckets
Tempo de performance medida por Big O notation. LinkedHashMap e HashMap são melhores que TreeMap (quanto a metodos Get, Put, ContainsKey, Remove).

Operação map() tem a função de converter cada elemento recebido em um outro objeto, de acordo coma função passada.
É possível ordenar um Map pelo valor.

- Conhecendo os métodos Map - parte 1
Código ExemploMap.java

- Conhecendo os métodos Map - parte 2
Código ExemploMap.java
A partir da modelo menos eficiente

- Ordenação - Map
Codigo ExemploOrdenacaoMap.java
Consegui corrigir e rodar o ComparatorPaginas

- Exercícios Propostos - Map
Código ExercicioProposto01
Código ExercicioProposto02

- Slides
baixei!

5) Stream
- Visão geral: Stream API - parte 1
Stream API facilita manipulação de Collections.
Classe anônima nao recebe nome, declarada e instanciada em única vez. Ex.: classe simples é declarada dentro do

```
argumento de um método, nao tem atributos, construtor etc.
meusGatos.sort(new Comparator<Gato>() {

        public int compare(Gato g1, Gato g2)
                return Integer.compare(g1.getIdade(), g2.getIdade());
        }
});

#Functional Interface é um Single Abstract Method (SAM)
com implementação tratada como expressão lambda.
Ex.: publico ibterface Comparator <T> {
                int compare(T var1, T var2);

Comparator tem compare como único método abstrato.
Ex.: ActionListener tem actionPerformed como unico metodo
abstrato, mas sem anotação visivel de Functionaç Interface
Functional Interfaces = Comparator, Consumer, Function,
Predicate.
# Lambda é função sem declaração, não tem nome, tipo de
retorno e modificador de acesso (get); metodo declarado
em mesmo lugar de uso. Sintaxe: (argumento) -> (corpo)
Parece com classe anônima.
Ex.: meusGatos.sort(Comparator.comparing((Gato gato) -> gato.getNome()));
Lambda é como uma simplificação de classe anônima.

- Visão geral: Stream API - parte 2
Reference Method faz referência a método ou construtor de
classe (forma funcional), indica que deve ser usado em
ponto específico do código que fica + simples e legível.
Uso: informar classe ou referencia :: metodo s/parênteses
no final. É forma de simplificar expressao lambda.
Ex.: meusGatos.sort(Comparator.comparing(Gato::getNome));

#Streams API manipula coleções em Java pelos princípios
da programação funcional; combina-se com as expressões
lambda, lida com conjuntos, tem escrita simples e concisa
de código que facilita manutenção e paralelização. Tem a
estrutura: Source (fonte,collection)-> Pipeline (operaçao
intermediaria -> Terminal (operação terminal)

Tendo: List<String> numeros = List.of("1","2","5","3"),
qual linha de comando exibiria a média?
System.out.print(numeros.stream().mapToInt(Integer::parseInt).average());

- Principais operações Stream API - parte 1
## Codigo RefatoracaoOrdenacaoMap.java

- Principais operações Stream API - parte 2
Operações intermediarias, retornam um stream, podem ser
encadeadas.
Operações terminais, só pode usar uma, retornam um objeto
ou um valor.
## Código ExercicioStreamAPI.java
Atentar para map e collect!
```

- Principais operações Stream API - parte 3
Código ExercicioStreamAPI.java
Continuou a partir do exercicio de nros pares em lista.

- Conclusão do curso
Sugeridos 3 cursos da DIO (2 de Java e 1 de estrutura de dados e algoritmos)

- Slides
baixei!