ECM251 - Linguagens I

Teoria - Java Collections - Parte 2
Prof. Murilo Zanini de Carvalho

Antes de começar!

Clone seu repositório do Github

- Lembre-se sempre antes de iniciar uma aula, clonar seu repositório remoto e realizar as atividades nele.
- Para cada atividade desenvolvida, criar um novo diretório.

Retirado de (https://miro.medium.com/max/4000/0*MZMI76wKo2FQLqG0.png), em 07/03/2021

Retomando - Pilares da Orientação a Objetos

Abstração

- Capacidade de representar elementos (físicos com ou não) por suas características e comportamentos.

Encapsulamento

- Isolar a forma como a abstração é realizada, fornecendo uma interface de acesso a ela.

Herança

- Transmissão de características de um elemento geral para elementos mais específicos.

Polimorfismo

- Possibilidade de executar o método correto de acordo com a situação/referência utilizada.

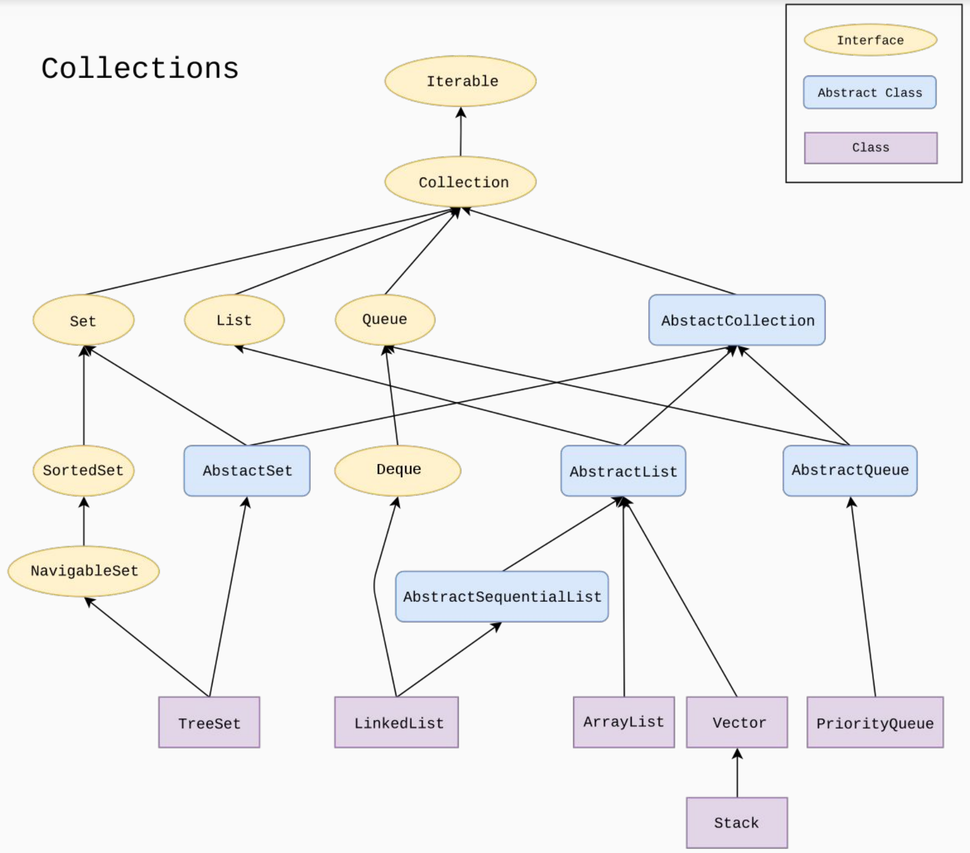
ATENÇÃO PARA AULA DE HOJE!!

Implemente os códigos da aula, isso vai ser importante!!

Retirado de (https://www.testbytes.net/wp-content/uploads/2019/06/Untitled-63.png), em 22/05/2021

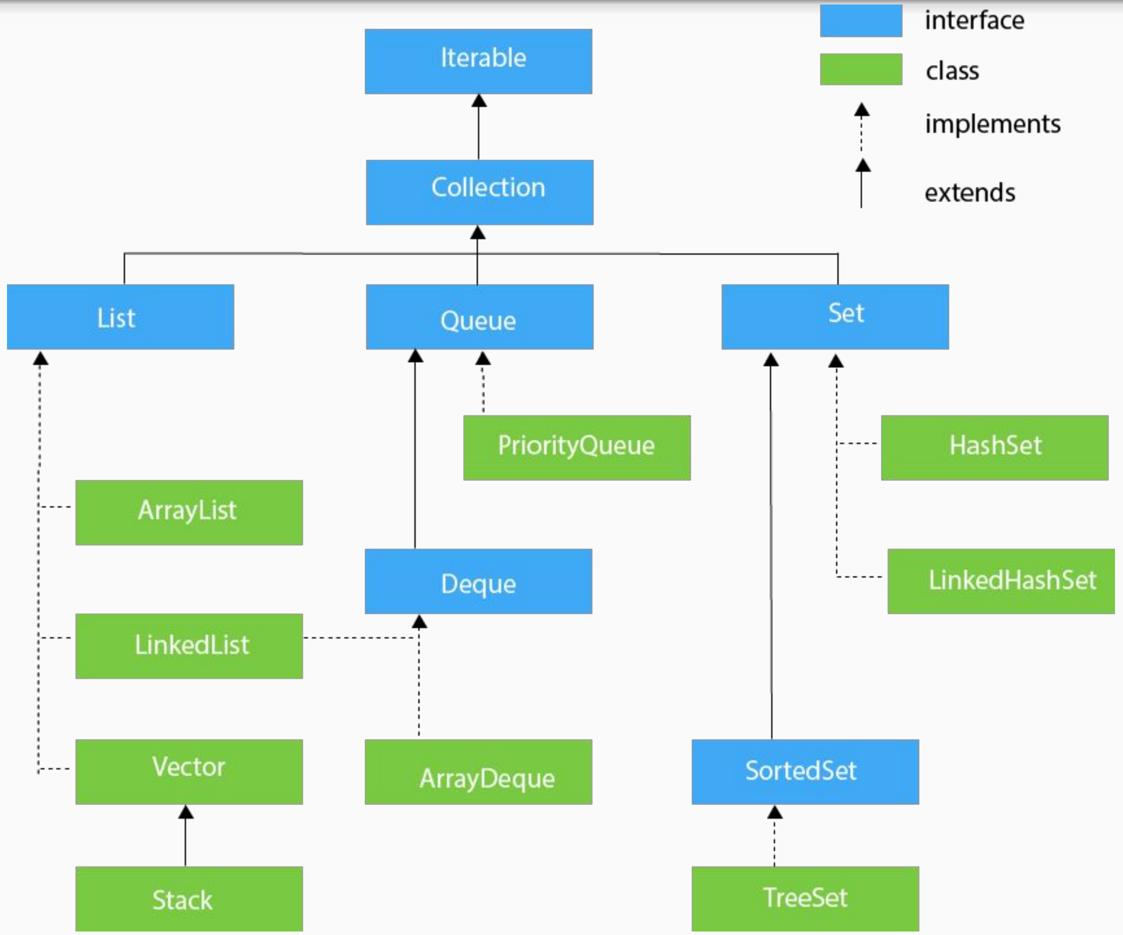
Java Collections
A continuação da saga!!

Java Collections



Retirado de (https://upload.wikimedia.org/wikipedia/commons/thumb/a/ab/Java.util.Collection_hierarchy.svg/1200px-Java.util.Collection_hierarchy.svg.png), em 09/05/2020.

Java Collections



Retirado de (https://static.javatpoint.com/images/java-collection-hierarchy.png), em 09/05/2020.

Sets

Interface Set

Diferente da interface List, a interface Set não permite objetos duplicados armazenados nela. Todos os elementos que implementam a interface Set possuem os mesmos métodos básicos, alterando a forma como implementam as estruturas.

Classes que implementam a interface:

- HashSet
- LinkedHashSet
- TreeSet

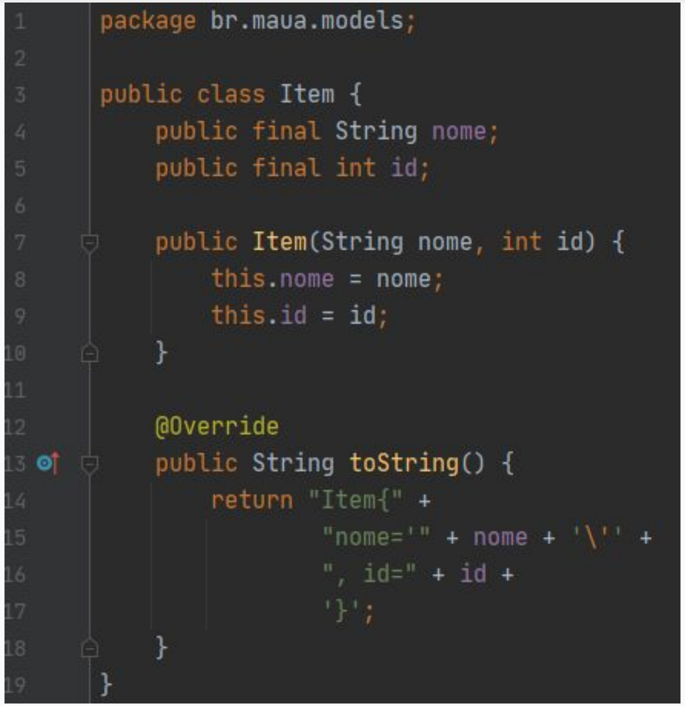
HashSet

Implementam:

- It stores unique elements and permits nulls
- It's backed by a HashMap
- It doesn't maintain insertion order
- It's not thread-safe

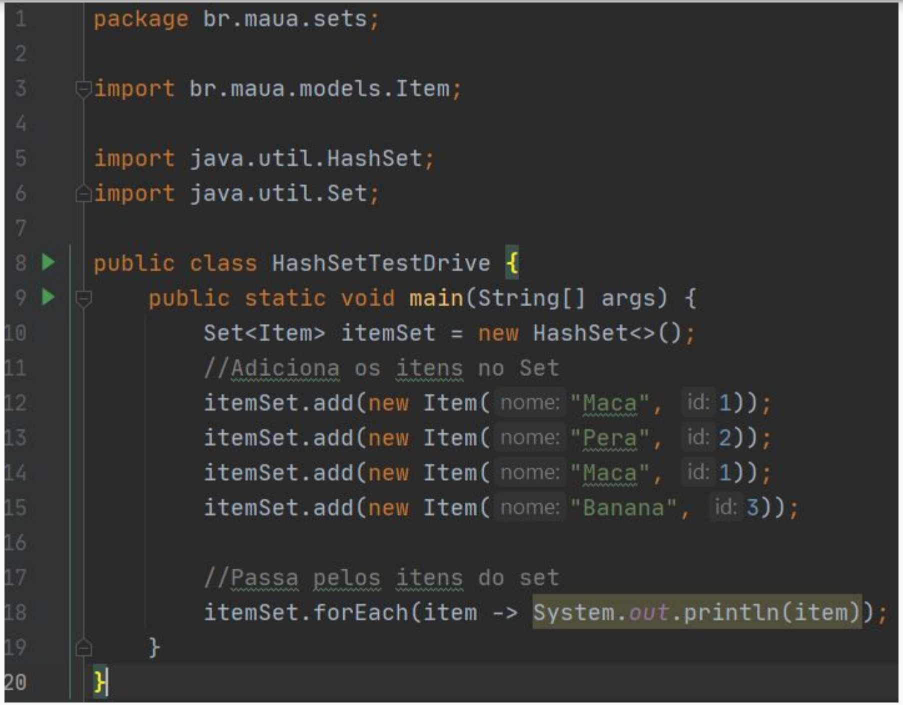
Retirado de (https://www.baeldung.com/java-hashset)

HashSet

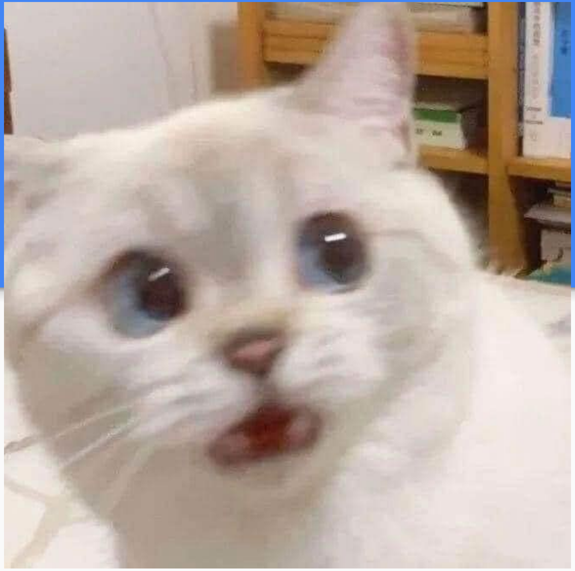


```java
package br.maua.models;

public class Item {
    public final String nome;
    public final int id;

    public Item(String nome, int id) {
        this.nome = nome;
        this.id = id;
    }

    @Override
    public String toString() {
        return "Item{" +
                "nome='" + nome + '\'' +
                ", id=" + id +
                '}';
    }
}
```

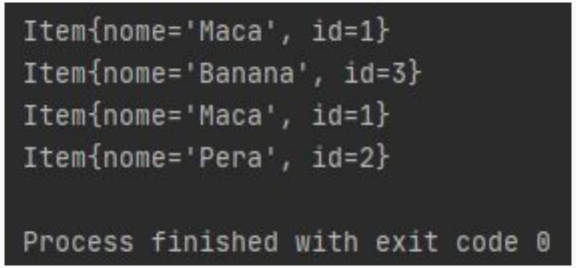
HashSet



```java
package br.maua.sets;

import br.maua.models.Item;

import java.util.HashSet;
import java.util.Set;

public class HashSetTestDrive {
    public static void main(String[] args) {
        Set<Item> itemSet = new HashSet<>();
        //Adiciona os itens no Set
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Pera",  id: 2));
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do set
        itemSet.forEach(item -> System.out.println(item));
    }
}
```



Retirado de (https://s.keepmeme.com/files/en_posts/20200908/blurred-surprised-cat-meme-5b734a45210ef3b6657bcbe2831715fa.jpg), em 28/05/2021



```
Item{nome='Maca', id=1}
Item{nome='Banana', id=3}
Item{nome='Maca', id=1}
Item{nome='Pera', id=2}

Process finished with exit code 0
```

HashSet

ESPERA!!

Os HashSet não guardam apenas valores únicos??

O que está acontecendo??

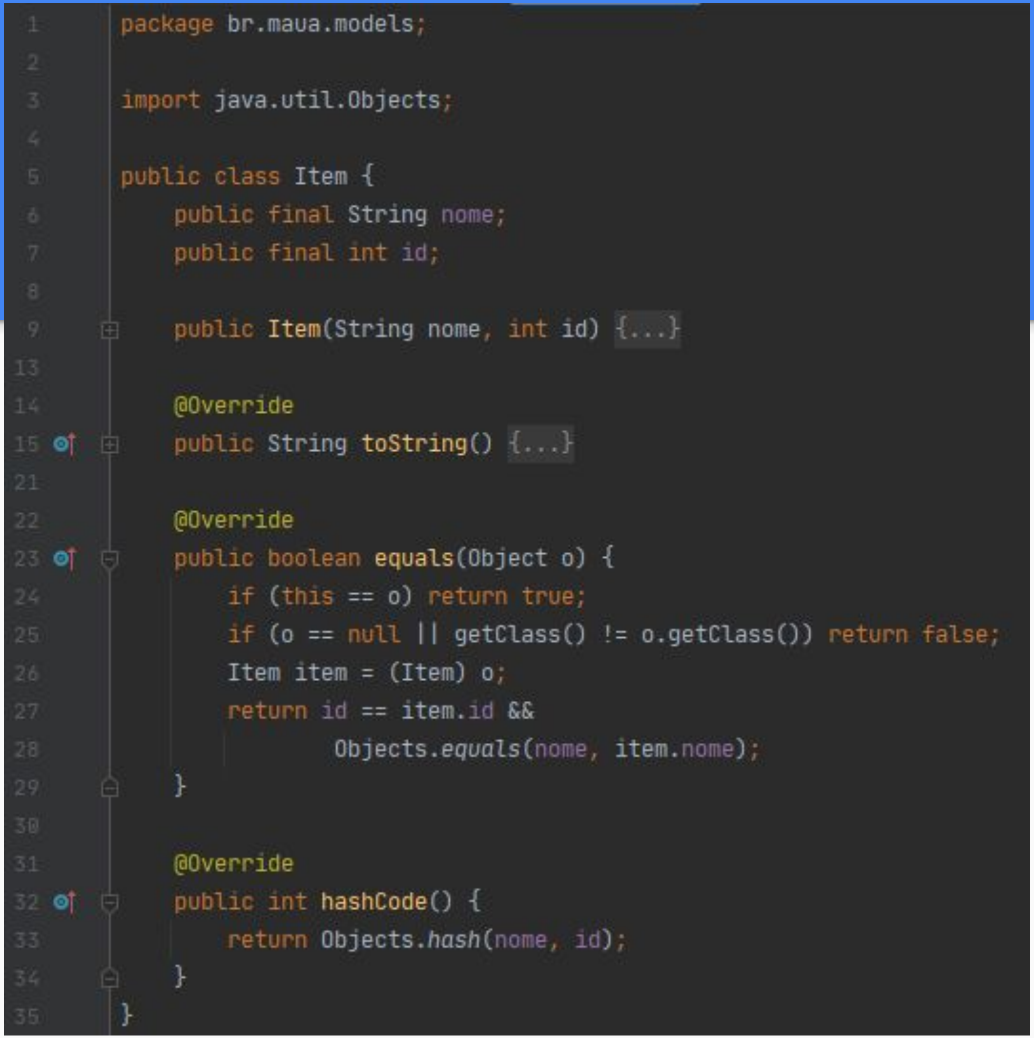
HashSet

Os HashSet utilizam o valor gerado pelo método hashCode() da classe.

Dois objetos podem não necessáriamente serem iguais se os seus hashCodes forem diferentes.

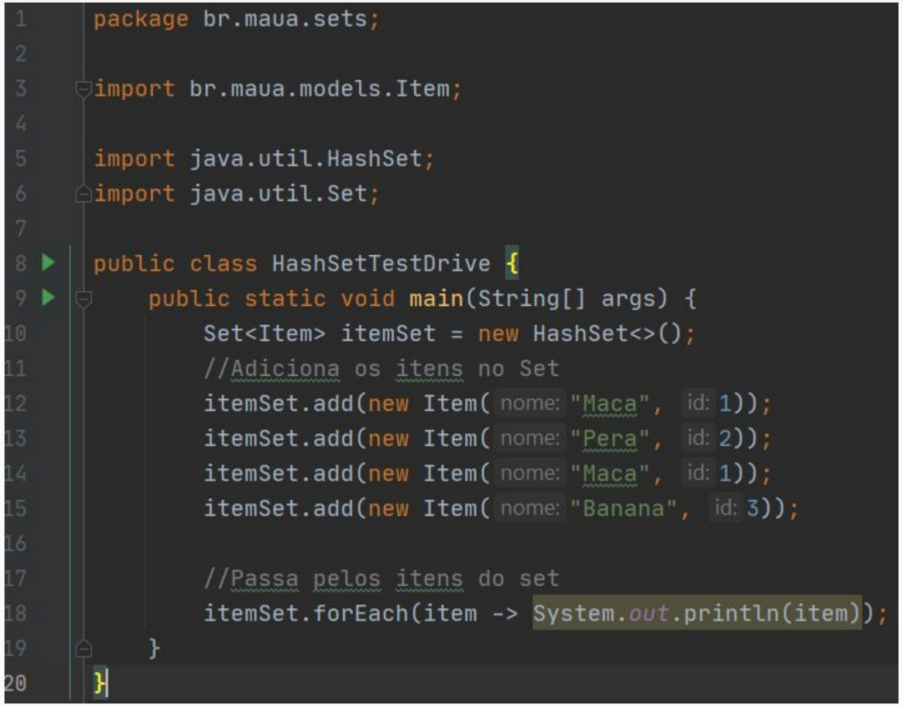
Retirado de (https://media.makeameme.org/created/ahhh-finally-relaxed.jpg), em 28/05/2021

HashSet

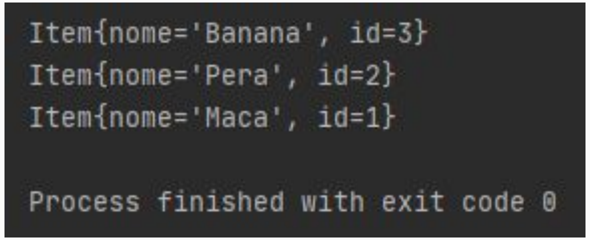


```
package br.maua.models;

import java.util.Objects;

public class Item {
    public final String nome;
    public final int id;

    public Item(String nome, int id) {...}

    @Override
    public String toString() {...}

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Item item = (Item) o;
        return id == item.id &&
                Objects.equals(nome, item.nome);
    }

    @Override
    public int hashCode() {
        return Objects.hash(nome, id);
    }
}
```

HashSet



```java
package br.maua.sets;

import br.maua.models.Item;

import java.util.HashSet;
import java.util.Set;

public class HashSetTestDrive {
    public static void main(String[] args) {
        Set<Item> itemSet = new HashSet<>();
        //Adiciona os itens no Set
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Pera",  id: 2));
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do set
        itemSet.forEach(item -> System.out.println(item));
    }
}
```

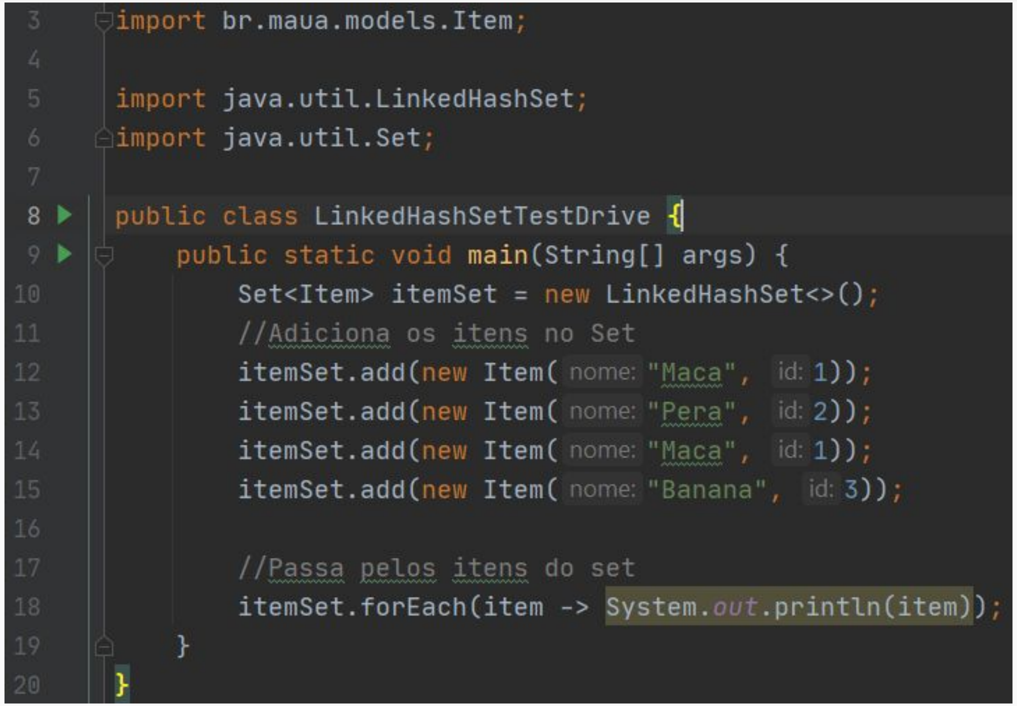


```
Item{nome='Banana', id=3}
Item{nome='Pera', id=2}
Item{nome='Maca', id=1}

Process finished with exit code 0
```

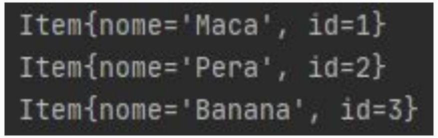
LinkedHashSet

The LinkedHashSet is an ordered version of HashSet that maintains a doubly-linked List across all elements. When the iteration order is needed to be maintained this class is used. When iterating through a HashSet the order is unpredictable, while a LinkedHashSet lets us iterate through the elements in the order in which they were inserted. When cycling through LinkedHashSet using an iterator, the elements will be returned in the order in which they were inserted. (Retirado de https://www.geeksforgeeks.org/linkedhashset-in-java-with-examples/)

LinkedHashSet



```java
import br.maua.models.Item;

import java.util.LinkedHashSet;
import java.util.Set;

public class LinkedHashSetTestDrive {
    public static void main(String[] args) {
        Set<Item> itemSet = new LinkedHashSet<>();
        //Adiciona os itens no Set
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Pera",  id: 2));
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do set
        itemSet.forEach(item -> System.out.println(item));
    }
}
```



```
Item{nome='Maca', id=1}
Item{nome='Pera', id=2}
Item{nome='Banana', id=3}
```

TreeSet

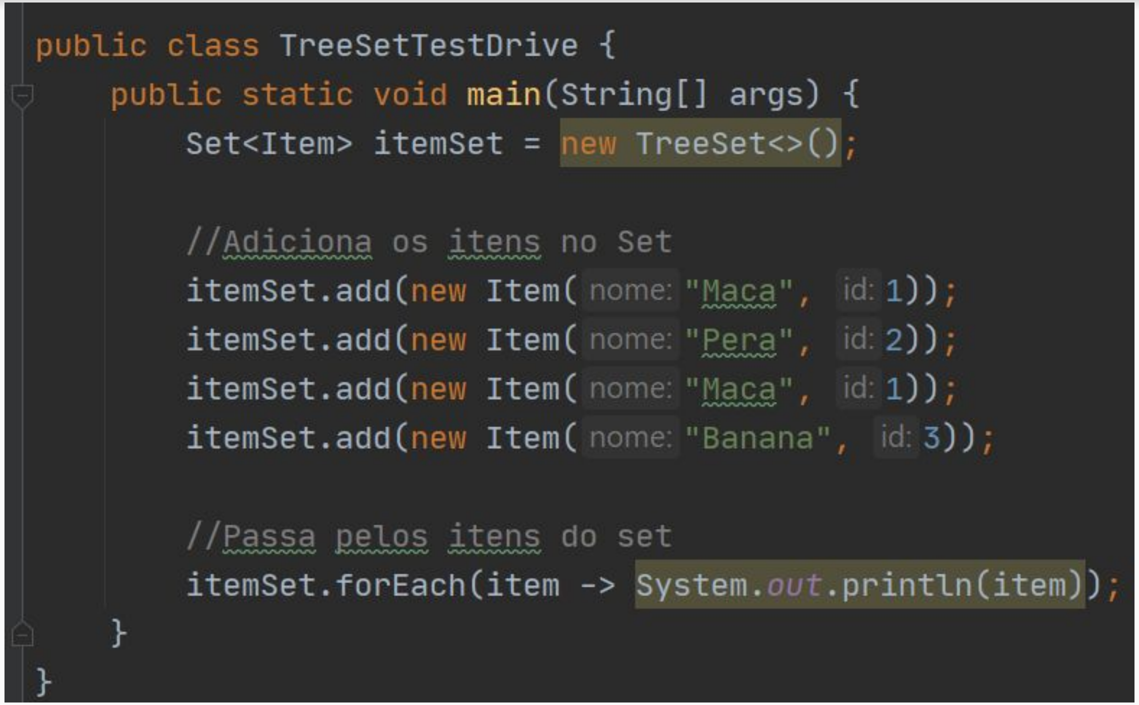
Ele implementa:

- It stores unique elements
- It doesn't preserve the insertion order of the elements
- It sorts the elements in ascending order
- It's not thread-safe

In this implementation, objects are sorted and stored in ascending order according to their natural order. The TreeSet uses a self-balancing binary search tree, more specifically a Red-Black tree.

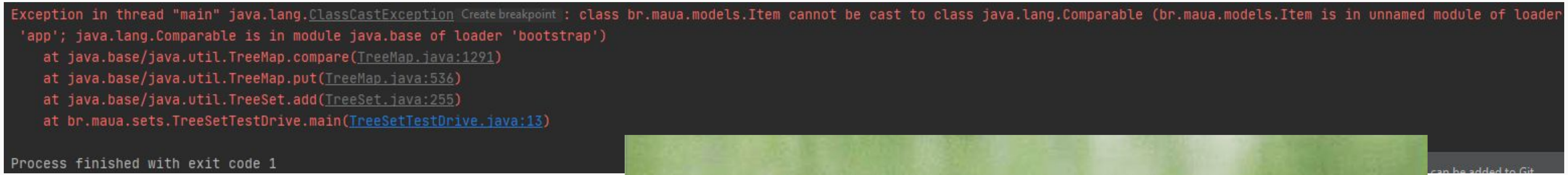
(Retirado de https://www.baeldung.com/java-tree-set)

TreeSet



```java
public class TreeSetTestDrive {
    public static void main(String[] args) {
        Set<Item> itemSet = new TreeSet<>();

        //Adiciona os itens no Set
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Pera",  id: 2));
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do set
        itemSet.forEach(item -> System.out.println(item));
    }
}
```

TreeSet



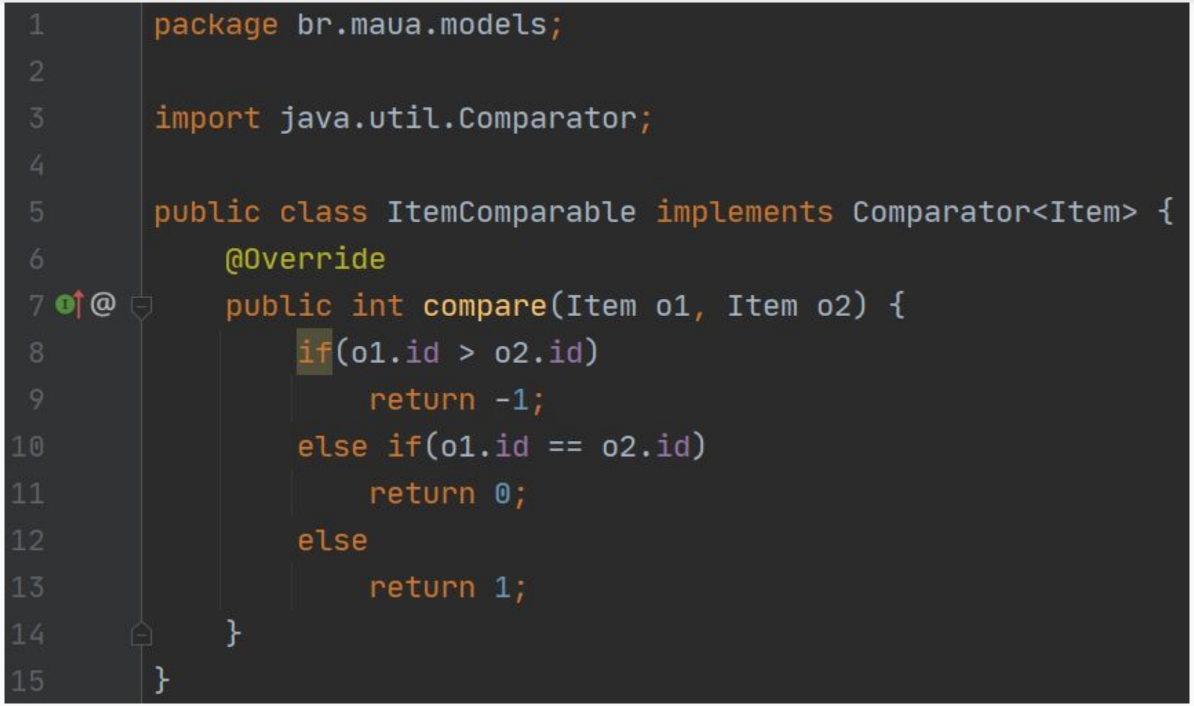
```
Exception in thread "main" java.lang.ClassCastException Create breakpoint : class br.maua.models.Item cannot be cast to class java.lang.Comparable (br.maua.models.Item is in unnamed module of loader
 'app'; java.lang.Comparable is in module java.base of loader 'bootstrap')
    at java.base/java.util.TreeMap.compare(TreeMap.java:1291)
    at java.base/java.util.TreeMap.put(TreeMap.java:536)
    at java.base/java.util.TreeSet.add(TreeSet.java:255)
    at br.maua.sets.TreeSetTestDrive.main(TreeSetTestDrive.java:13)

Process finished with exit code 1
```

Retirado de (https://i.imgflip.com/1p7u08.jpg), em 28/05/2021

TreeSet

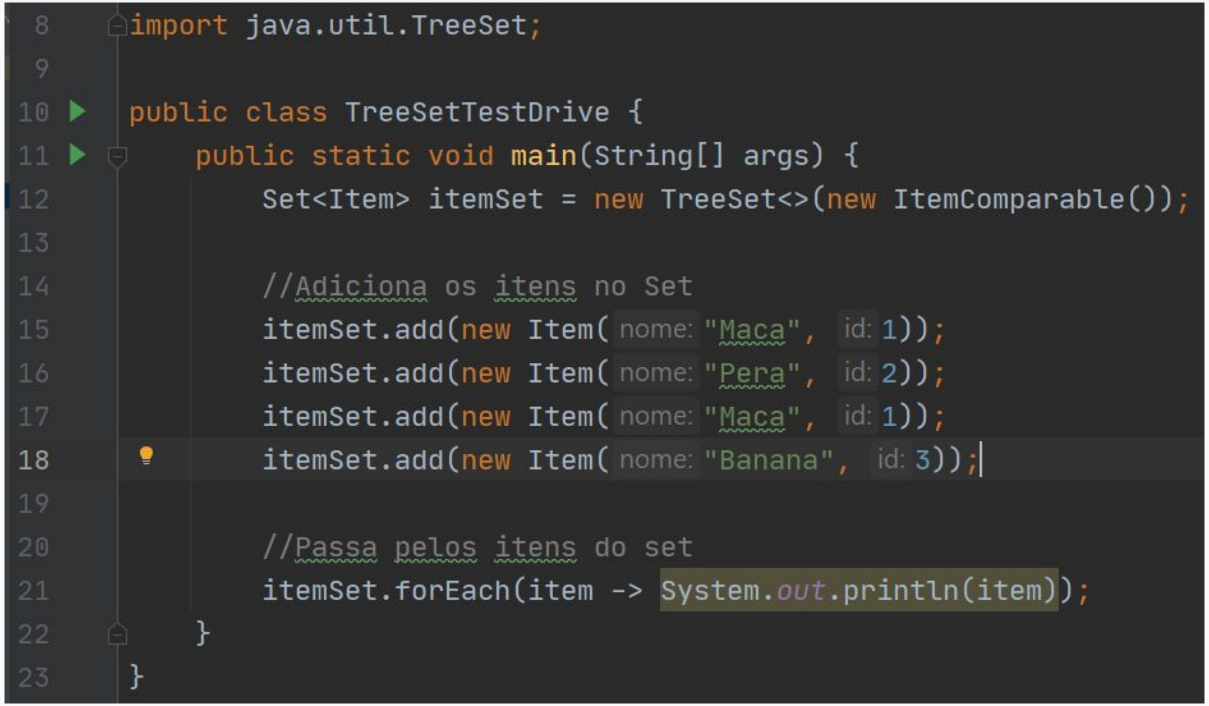
Para que os objetos possam ser inseridos em ordem, é preciso enviar ao construtor do TreeSet, uma classe que implemente a regra de comparação dos elementos.

TreeSet

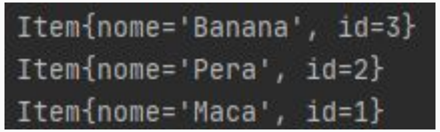


```java
package br.maua.models;

import java.util.Comparator;

public class ItemComparable implements Comparator<Item> {
    @Override
    public int compare(Item o1, Item o2) {
        if(o1.id > o2.id)
            return -1;
        else if(o1.id == o2.id)
            return 0;
        else
            return 1;
    }
}
```

TreeSet



```java
import java.util.TreeSet;

public class TreeSetTestDrive {
    public static void main(String[] args) {
        Set<Item> itemSet = new TreeSet<>(new ItemComparable());

        //Adiciona os itens no Set
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Pera",  id: 2));
        itemSet.add(new Item( nome: "Maca",  id: 1));
        itemSet.add(new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do set
        itemSet.forEach(item -> System.out.println(item));
    }
}
```



```
Item{nome='Banana', id=3}
Item{nome='Pera', id=2}
Item{nome='Maca', id=1}
```

Maps

Interface Map

A interface Map possibilita recuperar os objetos utilizando uma chave identificadora.

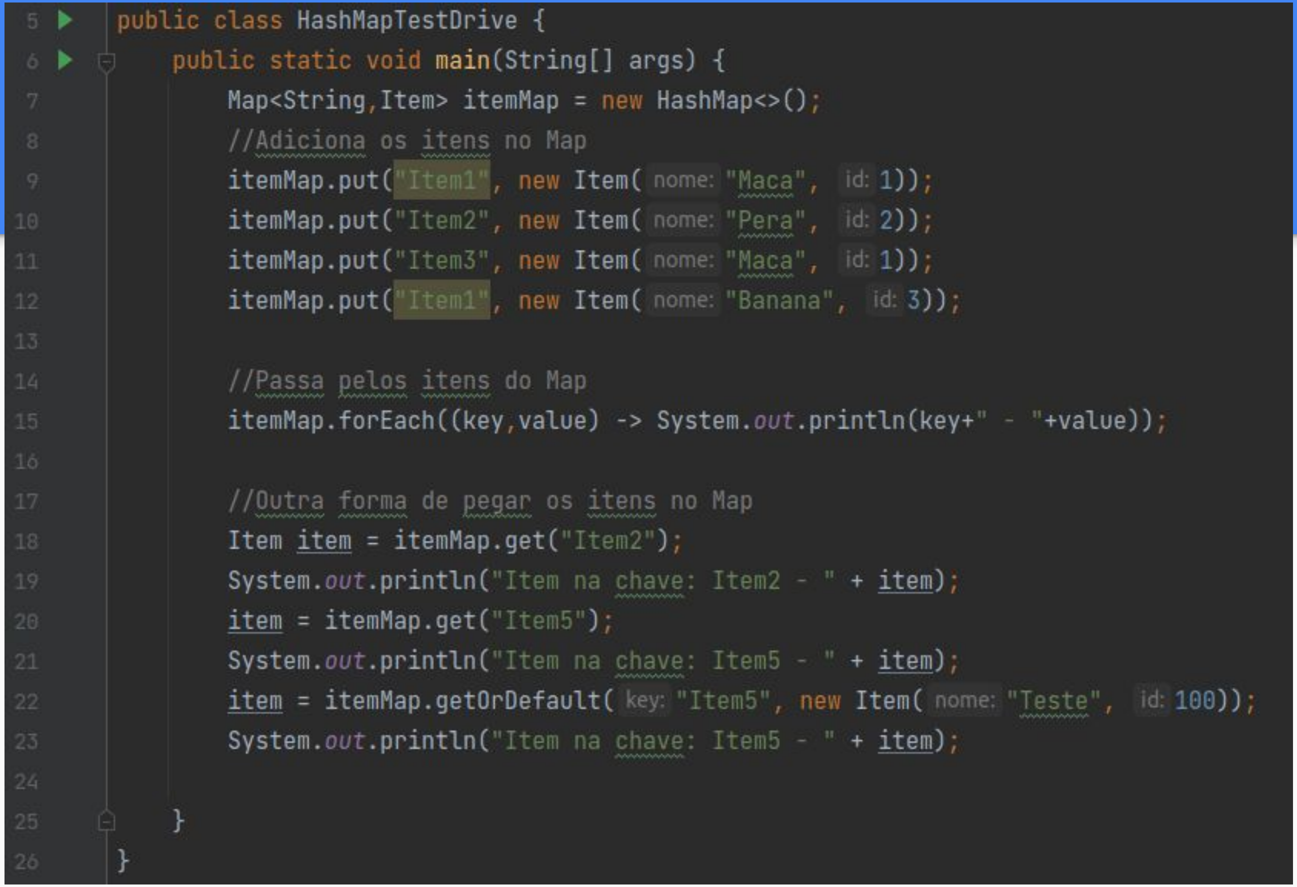
Implementam essa interface:

- HashMap
- TreeMap
- LinkedHashMap

HashMap

In the ArrayList chapter, you learned that Arrays store items as an ordered collection, and you have to access them with an index number (int type). A HashMap however, store items in "key/value" pairs, and you can access them by an index of another type (e.g. a String). One object is used as a key (index) to another object (value). It can store different types: String keys and Integer values, or the same type, like: String keys and String values. (Retirado de https://www.w3schools.com/java/java_hashmap.asp, em 30/05/2021)

HashMap

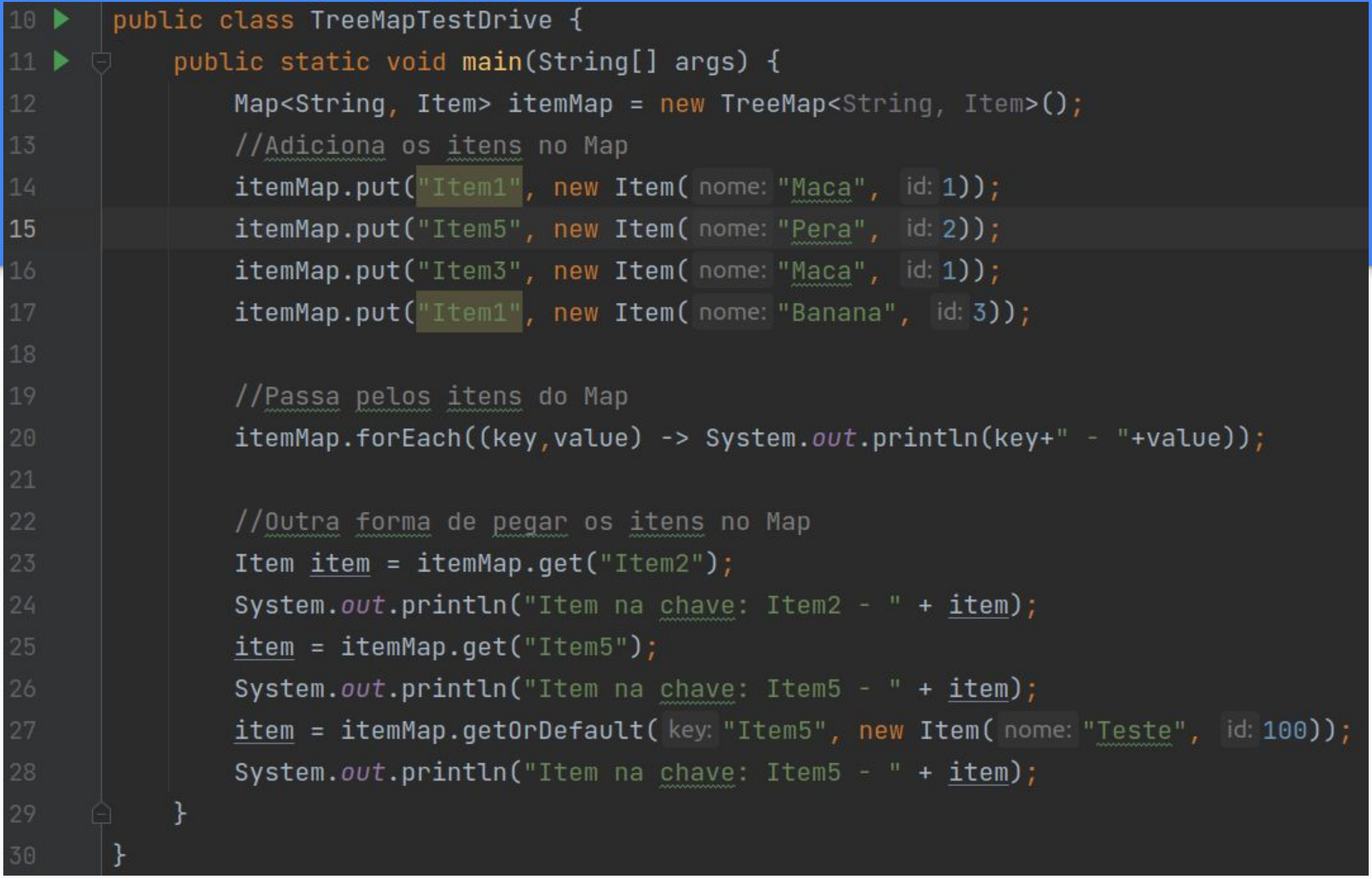


```java
public class HashMapTestDrive {
    public static void main(String[] args) {
        Map<String,Item> itemMap = new HashMap<>();
        //Adiciona os itens no Map
        itemMap.put("Item1", new Item( nome: "Maca",  id: 1));
        itemMap.put("Item2", new Item( nome: "Pera",  id: 2));
        itemMap.put("Item3", new Item( nome: "Maca",  id: 1));
        itemMap.put("Item1", new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do Map
        itemMap.forEach((key,value) -> System.out.println(key+" - "+value));

        //Outra forma de pegar os itens no Map
        Item item = itemMap.get("Item2");
        System.out.println("Item na chave: Item2 - " + item);
        item = itemMap.get("Item5");
        System.out.println("Item na chave: Item5 - " + item);
        item = itemMap.getOrDefault( key: "Item5", new Item( nome: "Teste",  id: 100));
        System.out.println("Item na chave: Item5 - " + item);

    }
}
```

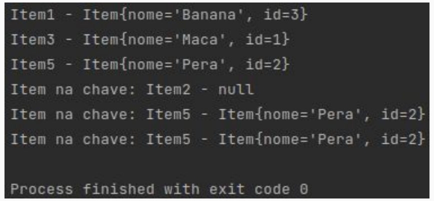
TreeMap

TreeMap is a map implementation that keeps its entries sorted according to the natural ordering of its keys or better still using a comparator if provided by the user at construction time. (Retirado de https://www.baeldung.com/java-treemap, em 30/05/2021)

TreeMap



```java
public class TreeMapTestDrive {
    public static void main(String[] args) {
        Map<String, Item> itemMap = new TreeMap<String, Item>();
        //Adiciona os itens no Map
        itemMap.put("Item1", new Item( nome: "Maca",  id: 1));
        itemMap.put("Item5", new Item( nome: "Pera",  id: 2));
        itemMap.put("Item3", new Item( nome: "Maca",  id: 1));
        itemMap.put("Item1", new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do Map
        itemMap.forEach((key,value) -> System.out.println(key+" - "+value));

        //Outra forma de pegar os itens no Map
        Item item = itemMap.get("Item2");
        System.out.println("Item na chave: Item2 - " + item);
        item = itemMap.get("Item5");
        System.out.println("Item na chave: Item5 - " + item);
        item = itemMap.getOrDefault( key: "Item5", new Item( nome: "Teste",  id: 100));
        System.out.println("Item na chave: Item5 - " + item);
    }
}
```

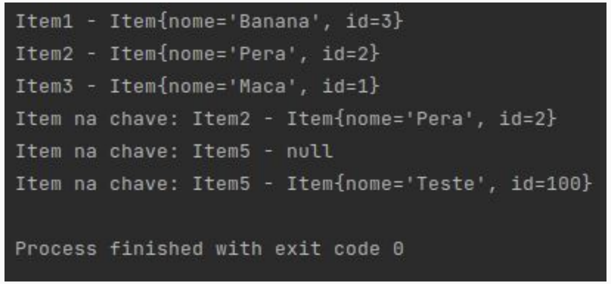


```
Item1 - Item{nome='Banana', id=3}
Item3 - Item{nome='Maca', id=1}
Item5 - Item{nome='Pera', id=2}
Item na chave: Item2 - null
Item na chave: Item5 - Item{nome='Pera', id=2}
Item na chave: Item5 - Item{nome='Pera', id=2}

Process finished with exit code 0
```

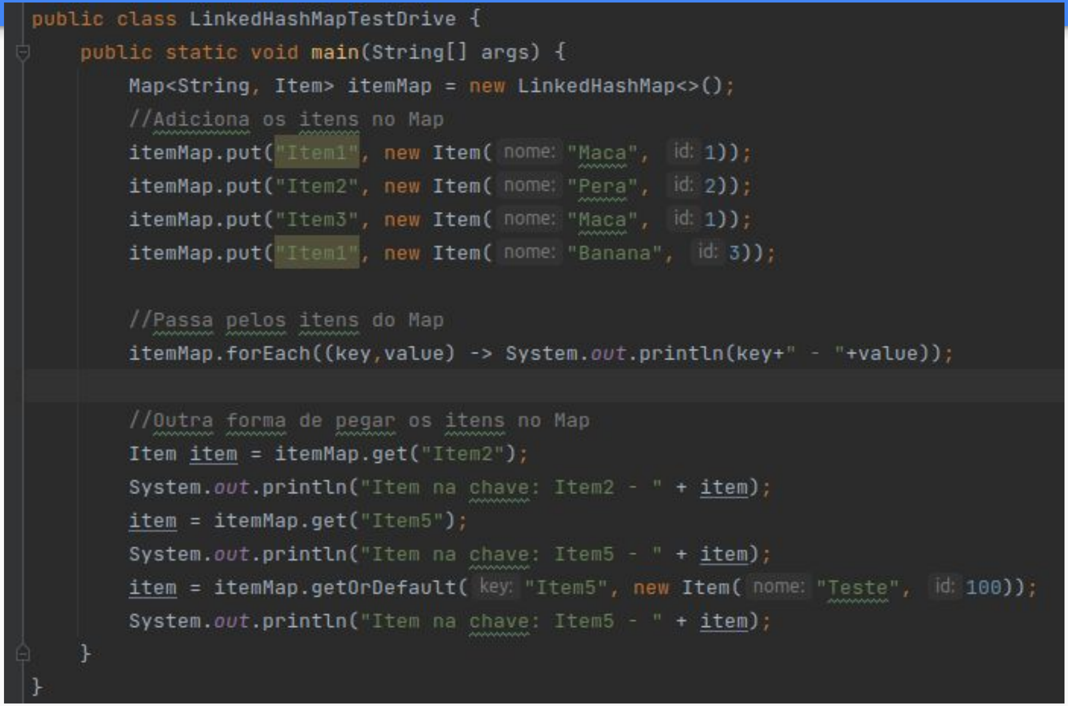
LinkedHashMap

The LinkedHashMap class is very similar to HashMap in most aspects. However, the linked hash map is based on both hash table and linked list to enhance the functionality of hash map. (Retirado de https://www.baeldung.com/java-linked-hashmap, em 30/05/2021)

LinkedHashMap



```
Item1 - Item{nome='Banana', id=3}
Item2 - Item{nome='Pera', id=2}
Item3 - Item{nome='Maca', id=1}
Item na chave: Item2 - Item{nome='Pera', id=2}
Item na chave: Item5 - null
Item na chave: Item5 - Item{nome='Teste', id=100}

Process finished with exit code 0
```



```java
public class LinkedHashMapTestDrive {
    public static void main(String[] args) {
        Map<String, Item> itemMap = new LinkedHashMap<>();
        //Adiciona os itens no Map
        itemMap.put("Item1", new Item( nome: "Maca",  id: 1));
        itemMap.put("Item2", new Item( nome: "Pera",  id: 2));
        itemMap.put("Item3", new Item( nome: "Maca",  id: 1));
        itemMap.put("Item1", new Item( nome: "Banana",  id: 3));

        //Passa pelos itens do Map
        itemMap.forEach((key,value) -> System.out.println(key+" - "+value));

        //Outra forma de pegar os itens no Map
        Item item = itemMap.get("Item2");
        System.out.println("Item na chave: Item2 - " + item);
        item = itemMap.get("Item5");
        System.out.println("Item na chave: Item5 - " + item);
        item = itemMap.getOrDefault( key: "Item5", new Item( nome: "Teste",  id: 100));
        System.out.println("Item na chave: Item5 - " + item);
    }
}
```

Comparação entre as Interfaces Map

Having looked at HashMap and LinkedHashMap implementations previously and now TreeMap, it is important to make a brief comparison between the three to guide us on which one fits where. A hash map is good as a general-purpose map implementation that provides rapid storage and retrieval operations. However, it falls short because of its chaotic and unorderly arrangement of entries.

This causes it to perform poorly in scenarios where there is a lot of iteration as the entire capacity of the underlying array affects traversal other than just the number of entries. A linked hash map possesses the good attributes of hash maps and adds order to the entries. It performs better where there is a lot of iteration because only the number of entries is taken into account regardless of capacity. A tree map takes ordering to the next level by providing complete control over how the keys should be sorted. On the flip side, it offers worse general performance than the other two alternatives. We could say a linked hash map reduces the chaos in the ordering of a hash map without incurring the performance penalty of a tree map. (Retirado de https://www.baeldung.com/java-treemap, em 30/05/2021)

Sugestões de Leitura

- https://www.baeldung.com/java-collections
- https://www.devmedia.com.br/java-collections-como-utilizar-collections/18450
- https://www.geeksforgeeks.org/collections-in-java-2/
- https://www.tutorialspoint.com/java/java_collections.htm

Perguntas?

Retirado de (https://cdn-icons-png.flaticon.com/512/1268/1268705.png), em 02/03/2022