Repaso Java

//Parte 1

IT BOARDING

BOOTCAMP

Índice

IT BOARDING BOOTCAMP

1 | JVM y JRE

IT BOARDING

BOOTCAMP

Java Virtual Machine - JVM
Java Runtime Environment - JRE

- Todos los programas Java corren dentro de una Máquina Virtual.
- JVM es una especificación, un contrato de reglas.

- JRE es una implementación particular de una JVM.
- Existe una JRE para cada plataforma (Linux, Windows, Mac, AS-400)

Java Virtual Machine - JVM
Java Runtime Environment - JRE


Windows


Mac OS


Linux


AS - 400

2 | Tipos primitivos y Wrappers

IT BOARDING

BOOTCAMP

Tipos primitivos vs. Wrappers

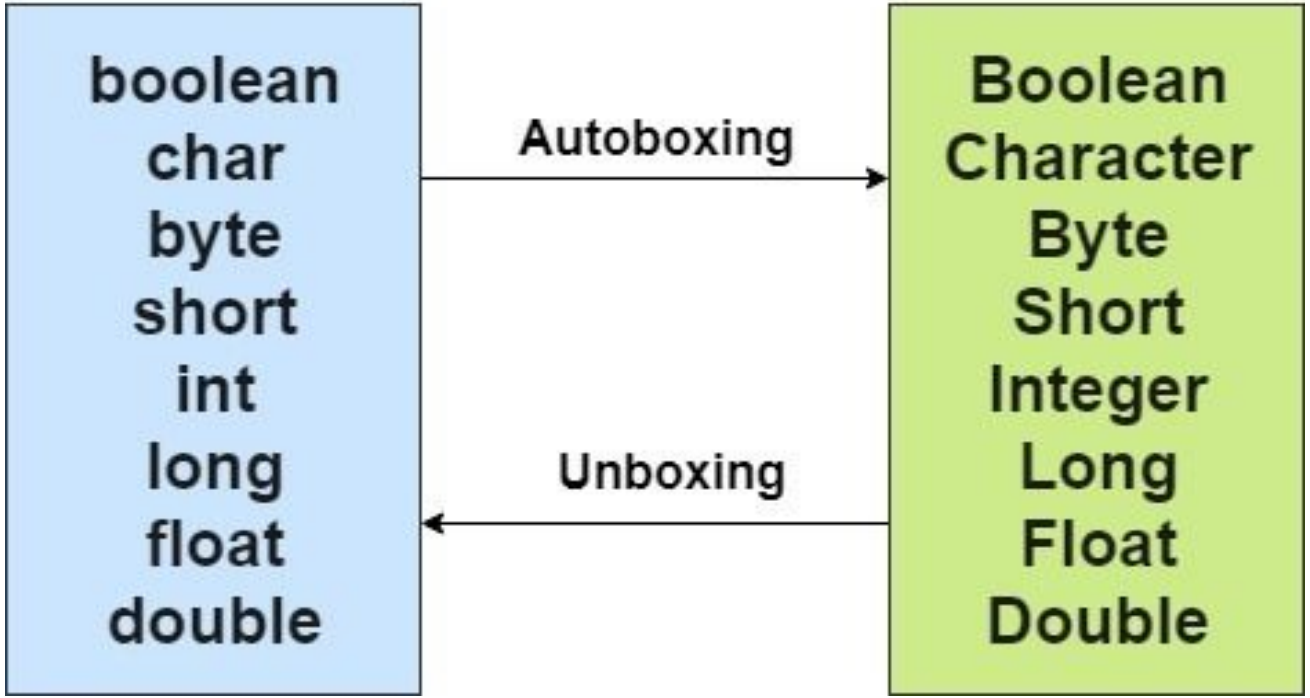
Tipos primitivos

- No son objetos.
- Son más eficientes en términos de memoria.
- Tienen un valor por defecto. (int **0**, boolean **false**, double **0.0d**)

Wrappers

- Cada tipo primitivo tiene su representación en Objeto (Wrapper).
- Los wrappers son inmutables, y ofrecen funcionalidades extra.
- No tienen un valor por defecto. Si no se instancian, son ***null***.

Wrapper Classes



3 | Colecciones

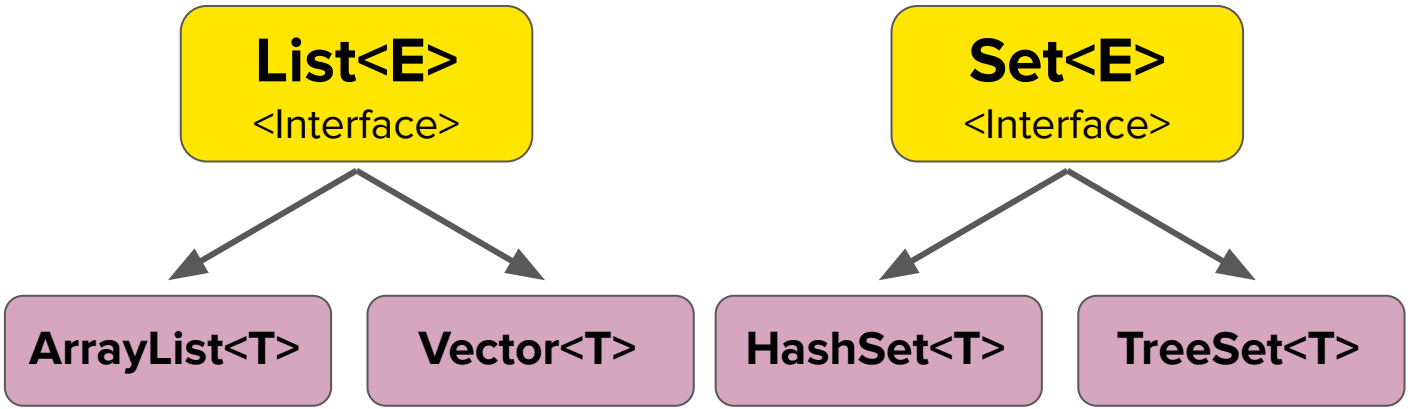
IT BOARDING

BOOTCAMP

Interface Collection<E>

Como cualquier interfaz, es un contrato. Cualquier implementación de Collection, deberá proveer de los siguientes métodos:

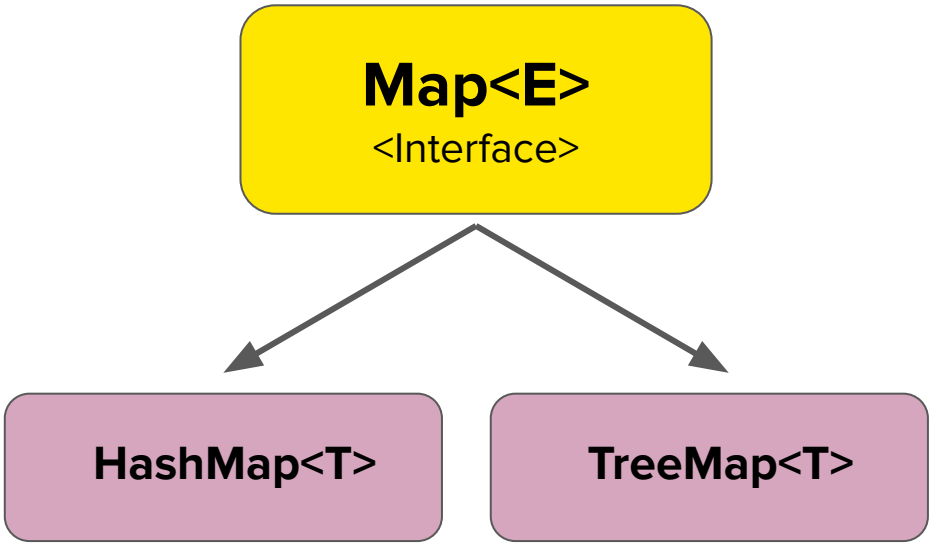
- **size**
- **isEmpty**
- **contains**
- **add**
- **remove**
- **clear**



Interface Map<E>

Como cualquier interfaz, es un contrato. Cualquier implementación de Collection, deberá proveer de los siguientes métodos:

- **size**
- **put**
- **get**
- **remove**
- **putAll**
- **isEmpty**
- **containsKey**
- **containsValue**
- **entrySet**
- **keySet**
- **clear**



Recorriendo Colecciones

Todas las colecciones (Collection) extienden de Iterable, por lo tanto, tienen la posibilidad de ser iteradas a través de un Iterador (Iterator).

FOR enriquecido

```
List<String> list = new ArrayList<String>();

for( String element : list ){
   System.out.println( element );
}
```

FOR EACH

```
List<String> list = new ArrayList<String>();

list.forEach(
(element) -> { System.out.println(element); });
```

Interface Iterable<E> e Iterator<E>

Otra forma de recorrer una colección es obteniendo un Iterador, a través de su Interfaz Iterable.

Interfaz Iterable<E>

- **iterator()**
  Obtiene un **Iterator<E>** para cualquier Collection.

Interfaz Iterator<E>

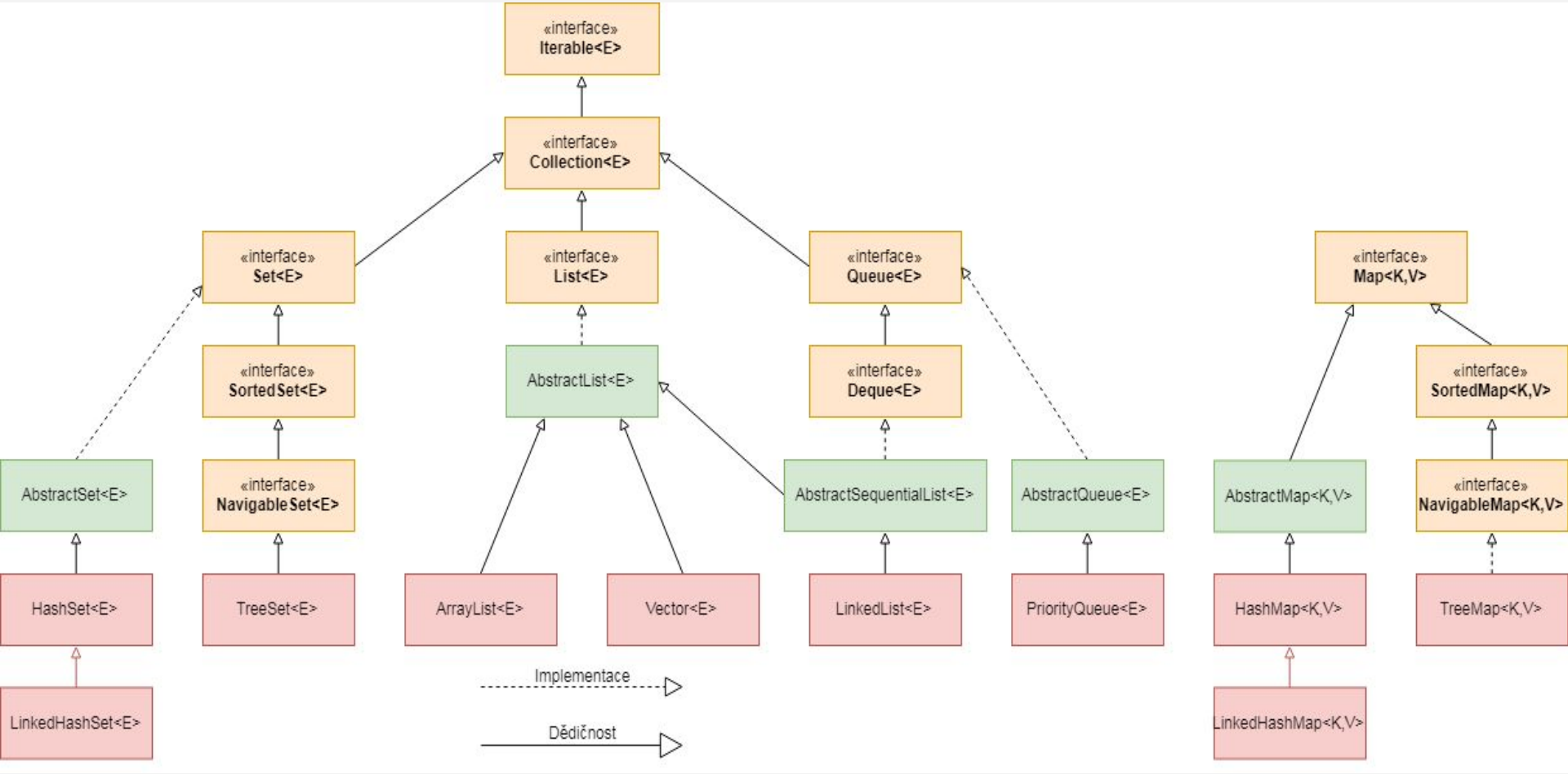
- **hasNext()**
- **next()**

Iterator<E>

```
List<String> list = new ArrayList<String>();

Iterator<String> iterator = list.iterator();

while (iterator.hasNext()) {
   String element = iterator.next();
   System.out.println(element);
}
```


«interface»
Iterable<E>
«interface»
Collection<E>
«interface»
Set<E>
«interface»
List<E>
«interface»
Queue<E>
«interface»
Map<K,V>
«interface»
SortedSet<E>
AbstractList<E>
«interface»
Deque<E>
«interface»
SortedMap<K,V>
AbstractSet<E>
«interface»
NavigableSet<E>
AbstractSequentialList<E>
AbstractQueue<E>
AbstractMap<K,V>
«interface»
NavigableMap<K,V>
HashSet<E>
TreeSet<E>
ArrayList<E>
Vector<E>
LinkedList<E>
PriorityQueue<E>
HashMap<K,V>
TreeMap<K,V>
Implementace
LinkedHashSet<E>
Dědičnost
LinkedHashMap<K,V>

4 Clases y objetos

IT BOARDING

BOOTCAMP

Clases y Objetos

- Una clase es un molde, una plantilla a través de la cual se crearán objetos.
- Un objeto, por el contrario es una instancia, generada siguiendo una definición dada por la clase.
- Los objetos tienen “vida”. Las clases, no.
- Los objetos interactúan entre ellos a través de “mensajes”.
- En un lenguaje 100% OO, todo es un objeto.

- Los objetos nos permiten modelar problemas de la vida real, a través de abstracciones.

Encapsulamiento

Las clases encapsulan los algoritmos, y proveen una ***interface* amigable** para el usuario (que será otro programador).

Por ejemplo:

- El control remoto de una televisión
- La llave que pone en marcha el auto
- Los botones que inician y detienen el horno a microondas

¿Dudas?
¿Preguntas?

IT BOARDING
BOOTCAMP

Gracias.

IT BOARDING

BOOTCAMP