Java Collections

¿Qué hay detrás de las estructuras de datos en Java?

Hola mundo!

Soy Sarina Bolaños

Estoy aquí porque me gusta Java y quiero compartirlo contigo.

Sarina20Lives

sarina20lives

Acerca de mi...

1. Actualmente estudio Ingeniería en Ciencias y Sistemas.
2. Laboré como auxiliar de cátedra en la Facultad de Ingeniería, USAC.
3. Trabajo como desarrolladora de software en Nabenik, utilizando Java, Kotlin y JavaScript..
4. Formo parte de StackOverLove

NABENIK

Herencia...

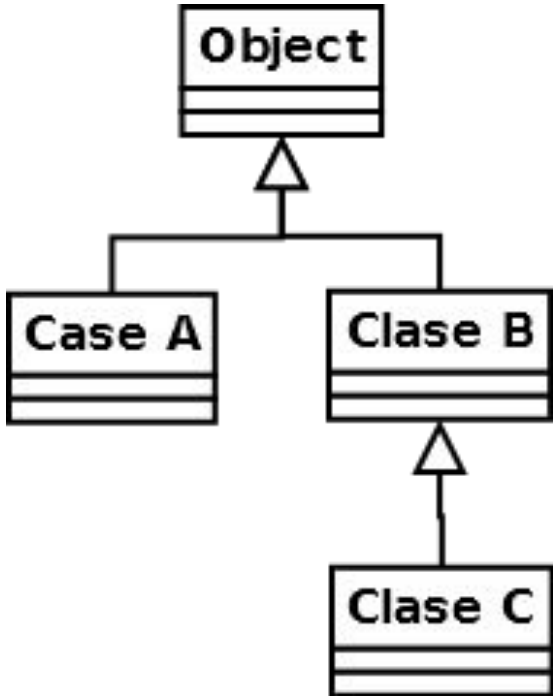
Herencia por defecto, declaración de herencia.

Herencia

Se conoce como herencia a la relación que existe entre dos clases, en donde una figura como **clase padre** y otra como **clase hija**, la clase hija obtiene de su padre todas sus propiedades y métodos que no sean privadas y que no posean limitaciones de acceso. Por definición la herencia podría ser “Single (simple)” o “Multiple (múltiple)”. **Java soporta herencia simple.**

Herencia

En Java por defecto, toda clase hereda de Object (java.lang.Object) y posee métodos como equals(), hashCode(), toString(), entre otros.



Sintaxis:

```
public class ClaseC extends ClaseB{

}
```

Interfaz...

Haciendo contratos

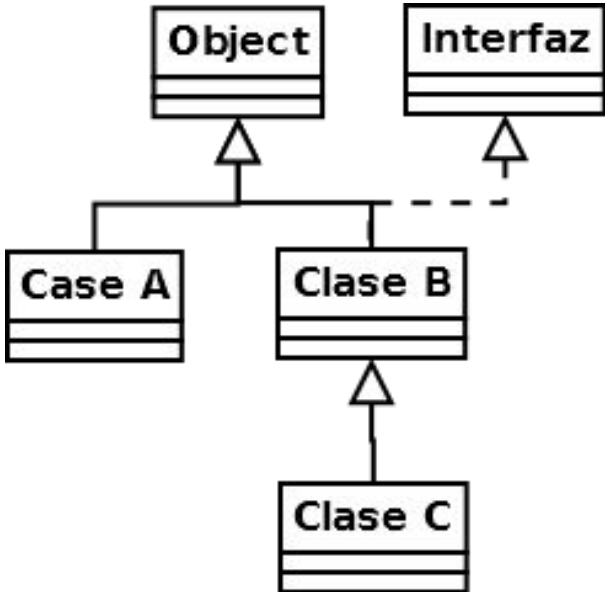
Interfaz

Es un contrato, un conjunto de métodos abstractos, los cuales son declarados pero no desarrollados, es decir que su comportamiento no está definido. Serán las clases concretas que implementen la interfaz quienes se comprometan a proveer el comportamiento para todos ellos.

A partir de Java 8 una interfaz puede contener variables y métodos default, pero eso, eso es otro tema de investigación.

Interfaz

Con el uso de interfaces, se consigue modelar una herencia múltiple en Java. Por defecto los métodos definidos en una interfaz son public y abstract. Java permite la implementación de n interfaces.



Sintaxis:

```
public class ClaseC extends ClaseB implements Interfaz{}
```

Interfaz y Herencia

1. Una interfaz no puede ser instanciada directamente.
2. Una clase abstracta puede implementar interfaces sin proveer comportamiento, el cual lo tendrán que proveer las clases concretas que hereden de la clase abstracta.
3. Una interfaz puede heredar de n interfaces.

Interfaz - Ejemplo

```
public interface Sociable{
    public abstract void iniciarConversación();
    public abstract void sonreir();
}

public class Persona implements Sociable{
    public void iniciarConversación(){
        System.out.println("Hola…");
    }
    public void sonreir(){
        System.out.println(":)");
    }
}
```

Java Collections

Collections, Map

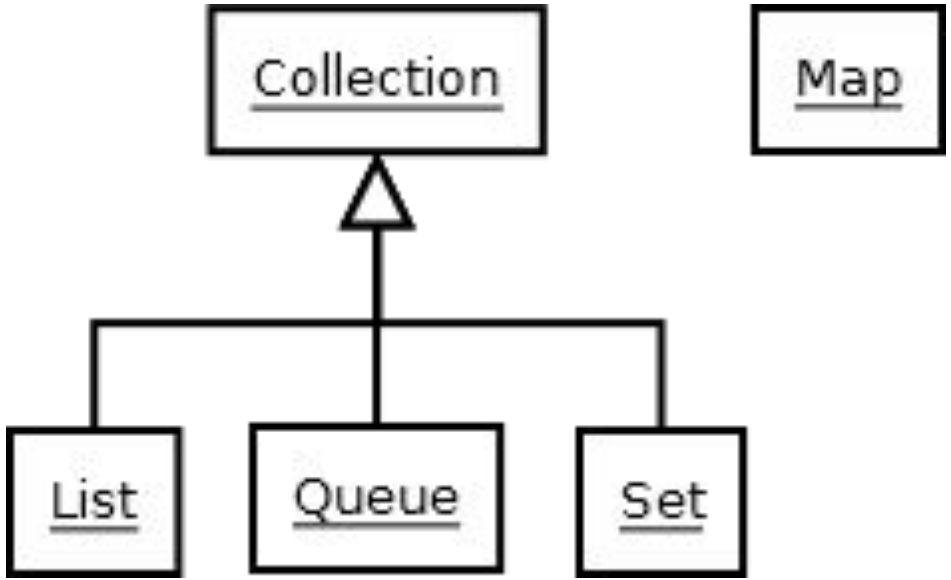
Java Collections Framework

Java Collections Framework es un conjunto de clases e interfaces del paquete java.util para la gestión de colecciones. Se Incluyen clases que implementan las interfaces List, Map, Queue y Set, entre otras. Collections Framework utiliza genéricos para el manejo de tipo de datos.

https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Collection interface

Es la raíz de todas las colecciones, excepto de Map.



Métodos comunes para la interfaz Collection

List, Set and Queue:

- `boolean add(E element)`
- `boolean remove(Object object)`
- `boolean isEmpty()`
- `int size()`
- `void clear()`
- `boolean contains()`

Iterable - Interfaz

Interfaz que permite hacer un recorrido óptimo de una colección.

Sintaxis:

```
public interface Collection<E> extends Iterable<E>
```

```
Iterator<String> iter = list.iterator();
while(iter.hashNext()){
    String string = iter.next();
    System.out.println(string);
}
```

List

Interface

List

Interfaz utilizada por colecciones de elementos que permiten entradas duplicadas. Los elementos son ordenables, accesibles por sus índices como en un array.

Clases (estructuras) que la implementan:

- ArrayList (Vector)

- LinkedList (también usa Queue)

Métodos comunes para la interfaz List

- `void add(E element)`
- `void add(int index, E element)`
- `E get(int index)`
- `int indexOf(Object o)`
- `void remove(int index)`
- `E set(int index, E e)`

ArrayList vs Array

Un ArrayList y un array son muy parecidos en su comportamiento y propósito, sin embargo difieren en lo siguiente:

1. Un Array puede contener objetos y tipos primitivos, un ArrayList solo puede contener objetos.
2. Un Array es fijo, no puede aumentar su tamaño, un ArrayList aumenta su tamaño según se requiera.

LinkedList

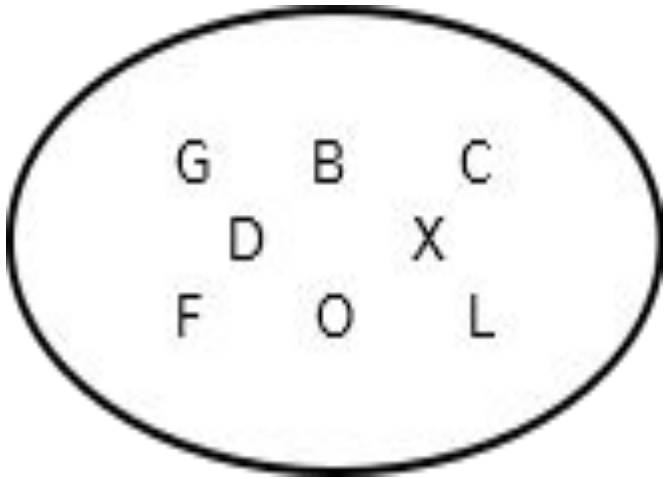
Es doblemente finalizada. Implementa la interfaz List pero también la interfaz Queue, lo cual le permite comportarse como pila o cola.

Set

Interface

Set

Interfaz utilizada por colecciones de elementos ordenables y que no permiten valores duplicados. La prioridad no es el ordenamiento sino que cada objeto sólo exista una única vez.



Clases (estructuras) que implementan la interfaz Set:

- HashSet:

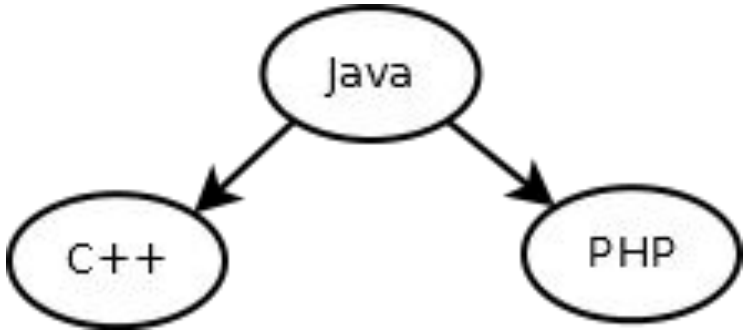
Almacena sus elementos en una hash table, por tanto usa el método hashCode() de Object, se pierde un poco el control del lugar en donde se insertan los objetos, pero su beneficio sería en el aumento de rapidez en las búsquedas.

- TreeSet:

Su principal beneficio es que inserta sus elementos de forma ordenada. TreeSet usa la interfaz Comparable y NavigableSet.

HashSet y TreeSet

-124384749	PHP
-102536945	Java
157895632	C++



Comparable - Interfaz

La interfaz comparable solo posee un método. Al implementarse el resultado se interpreta de la siguiente manera:

0 : El objeto actual es igual al objeto del argumento.
1 : El objeto actual es mayor al objeto del argumento.
-1 : El objeto actual es menor al objeto del argumento.

Sintaxis:

```
public interface Comparable<T>{
    public int compareTo(T o);
}
```

NavigableSet - Interfaz

```
public class TreeSet<E> extends AbstractSet<E>
implements NavigableSet<E>, Cloneable, Serializable
```

`E lower(E e)`

Retorna el elemento más grande que es < que **e** o null

`E floor(E e)`

Retorna el elemento más grande que es <= que **e** o null

`E ceiling(E e)`

Retorna el elemento más pequeño que es >= que **e** o null

`E higher (E e)`

Retorna el elemento más pequeño que es > que **e** o null

Map

Interface

Map

Es una colección por pares (llave y valor), no se permiten llaves repetidas. Map es considerada parte del Java Collections Framework aunque en realidad no implementa la interfaz Collection. La razón es que posee diferentes métodos para el acceso y control de sus valores.

Clases (estructuras) que implementan la interfaz Map:

- TreeMap
- HashMap

Toda estructura de datos permite nulos, excepto:

1. TreeMap (key)
2. Hashtable (key, value)
3. TreeSet(elements)
4. ArrayDeque(elements)

Métodos comunes para la interfaz Map

- `void clear()`
- `boolean isEmpty()`
- `int size()`
- `V get(Object key)`
- `V put(K key, V value)`
- `V remove(Object key)`
- `boolean containsKey(Object key)`
- `boolean containsValue(Object)`
- `Set<K> keySet()`
- `Collection<V> values()`

Gracias!