Java

Tema 3: Colecciones

Clases

ArrayList - HashSet

Contenido

Introducción

- Una colección es un objeto contenedor que almacena y organiza objetos de una misma clase, para manipularlos de forma individual y/o como grupo.

- El paquete java.util.* contiene clases e interfaces de colecciones.

- **Interfaces** de colecciones:

List	Una lista ordenada.
Map	Una lista formada por parejas de clave y valor.
Set	Un conjunto
Queue **Deque**	Estructuras FIFO-cola (First In First Out) Estructuras LIFO-pila (Last In First Out)

3

Introducción

- Creacion genérica:

```
Coleccion<Clase>  objeto = new Coleccion<>();
```

- Ejemplo:

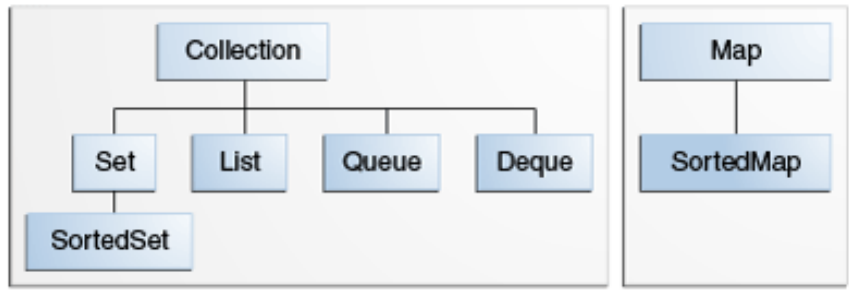
```
ArrayList<String>  objeto = new ArrayList<>();
```

- Las **clases** más usadas y que trabajeremos:

CLASE	USO	INTERFACE QUE IMPLEMENTA
ArrayList	Permite crear listas de objetos	List
HashSet	Permite crear conjunto de objetos	Set
ArrayDeque	Permite crear cola (y pila)	Queue Deque
HashMap	Permite crear listas de datos formados con parejas de elementos clave-valor.	Map

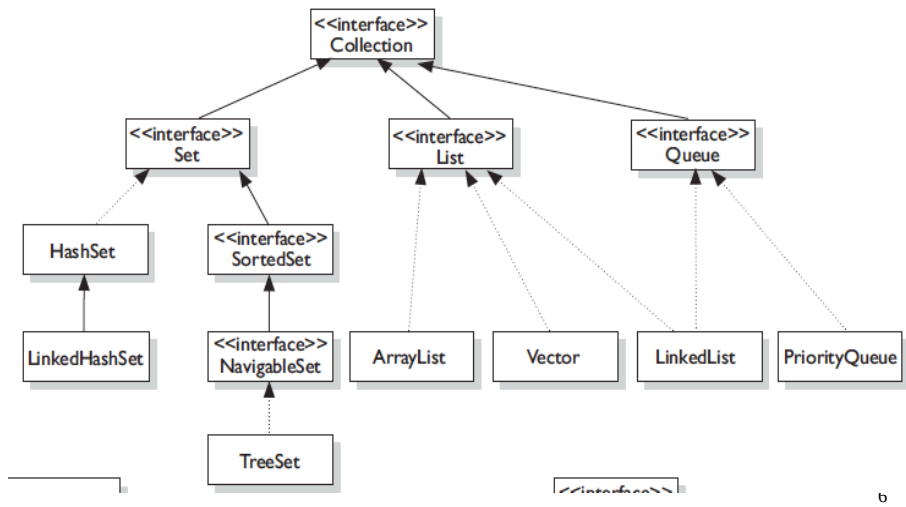
4

Introducción

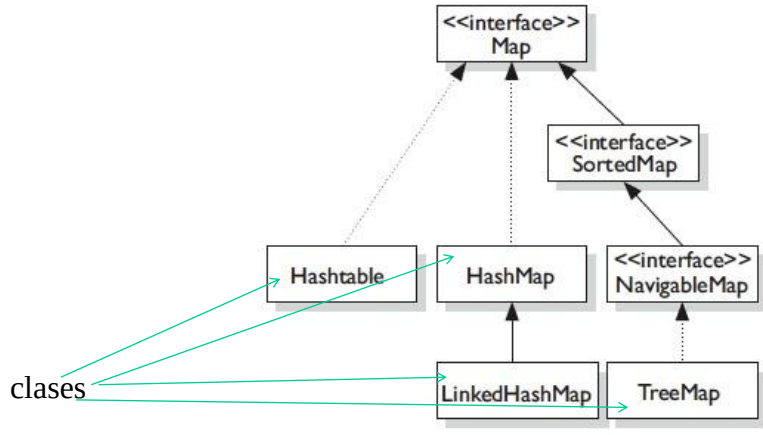


Interfaces que forman las colecciones de java.

Jerarquia de Collection



Jerarquia de Map



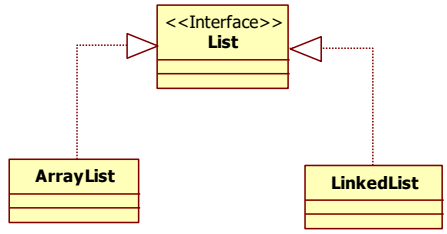
7

Interfaces List, Map, Set, Queue y Deque

8

List

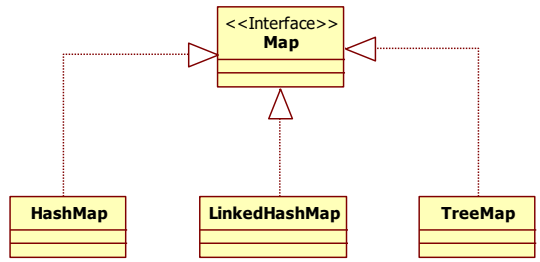
- ¿Cuál es la diferencia entre List y ArrayList?.
- List es algo genérico, hay muchos tipos de List.
- List es un **Interface**, NO una clase.
- ArrayList SI es una clase



9

Map

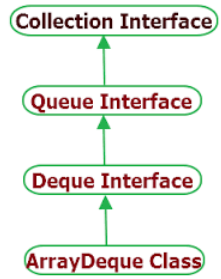
- ¿Cuál es la diferencia entre Map y HashMap?
- Map es algo genérico, hay muchos tipos de Map.
- Map es un **Interface**, NO una clase.
- HashMap SI es una clase



10

Queue y Deque

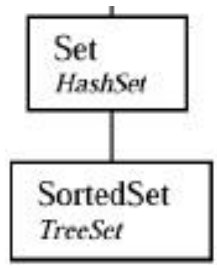
- ¿Cuál es la diferencia entre Deque y ArrayDeque?
- Deque es algo genérico, hay muchos tipos de Deque.
- Deque es un **Interface**, NO una clase.
- ArrayDeque SI es una clase



11

Set

- ¿Cuál es la diferencia entre Set y HashSet?
- Set es algo genérico, hay muchos tipos de Set.
- Set es un **Interface**, NO una clase.
- HashSet SI es una clase



12

Como declarar objetos coleccion

- Siempre se deben declarar los **HashMap** como Map
- Siempre se deben declarar los **ArrayList** como List
- Siempre se deben declarar los **HashSet** como Set
- Siempre se deben declarar los **ArrayDeque** como Deque
- De esa forma no nos atamos a una implementación concreta.

```
Map<Integer,String> variable = new HashMap <>();

List<String> lista=new ArrayList<>();

Set<String> conjunto = new HashSet <>();

Deque<String> pila = new ArrayDeque <>();
```

13

clase **ArrayList**

14

La clase ArrayList

- Representa una lista de elementos ordenados.
- Puede contener elementos repetidos
- Es como un array pero sin limitación de tamaño.
- Accedemos a cada elemento por su posición.
- Es un tipo de "List"

Ana
Carlos
María
Luís
Carlos
Marcos

15

Declaración

List<*clase*> *variable* = new
ArrayList<[*clase*]>( );

Los elementos que almacena deben ser objetos de una misma clase.

Para datos primitivos se deben usar: Integer, Double, Float, Short, Long, Boolean, Character

Ejemplos:

mejor usar List

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

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

16

Métodos principales

void add(Object o)	Añade el elemento especificado al final de la lista
int size()	Devuelve el número de elementos de la lista
Object get(int indice)	Devuelve el elemento de la posición especificada por *indice*
void clear()	Elimina todos los elementos de la lista.
Object remove(int i)	Elimina el elemento de la posición índice i.

17

Método add

- Para cargar datos o elementos a un ArrayList se utiliza el método add()

Ejemplo:

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

   lista.add("Pedro");
   lista.add("Juan");
   lista.add("Antonio");
    lista.add("Pedro");
```

18

Método size

- Para obtener el número de elementos del ArrayList se utiliza el método size()

Ejemplo:

```
System.out.println(lista.size());
```


Método get

- Para obtener el elemento de la posicion i del ArrayList se utiliza el método get(int i).
- El índice va **desde 0 a size()-1**

Ejemplo:

```
System.out.println(lista.get(0));
System.out.println(lista.get(1));
System.out.println(lista.get(2));
```

Método sort

- Ordena los elementos de la lista.
- Con parámetro `null`, le indicamos que ordene de forma ascendente.

Ejemplo:

```
System.out.println(lista);
lista.sort(null);
System.out.println(lista);
```

21

Método toString

- Devuelve como String los valores almacenados separados por , entre []
- Dentro de un println podemos no escribir el nombre del método toString() porque Java lo invoca automáticamente.

Ejemplo:

```
System.out.println(lista.toString());
System.out.println(lista);
```

```
Output - Tema3 (run) ×
run:
[Pedro, Juan, Antonio, Pedro]
[Pedro, Juan, Antonio, Pedro]
BUILD SUCCESSFUL (total time: 0 seconds)
```

22

Recorrerlo

- Para recorrer un arrayList se usa un bucle for.

```
for(int i = 0; i < lista.size(); i++){
   System.out.println(lista.get(i));
}
```

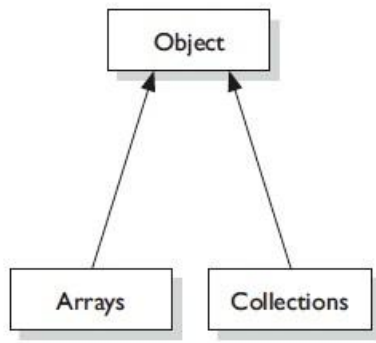
- O el bucle for/each.

```
for(String elemento : lista) {
      System.out.println(elemento);
}
```

23

Clase Collections

La clase Collections tiene métodos para trabajar con colecciones



24

Ordenar datos: métodos sort y reverse

- Se puede usar el método estático sort(List) de la clase Collections

```
Collections.sort(lista);
```

- Invertir el orden de los elementos

```
Collections.reverse(lista);
```

25

ArrayList: ejercicio

- Crea un ArrayList para almacenar los nombres de tus mejores amigos.
- Recorre la lista y muestra por pantalla los nombres introducidos.
- Ordenalos alfabeticamente y muestralos
- Recoge por pantalla un nuevo nombre y añadelo. Muestra la lista por pantalla.

26

clase **HashSet**

La clase HashSet

- La clase HashSet crea conjunto de datos sin ningún orden entre ellos
- **NO** permite tener elementos duplicados.
- Se recorre mas rápido que un ArrayList

valor
Juan López
Sara Baras
Pablo González
Luisa Pérez

Declaración

Set<clase> variable
= new **HashSet**<[clase]>( );

Ejemplo:

```
Set<String> alumnos = new HashSet <>();
HashSet<String> alumnos = new HashSet <>();
```

Carlos
Juan
Luisa
Ana

Métodos principales

void add(Object elemento)	Añade el elemento
int size()	Devuelve el número de elementos
boolean remove(Object elemento)	Elimina el elemento boolean borrado = alumnos.remove("Juan");
void clear()	Elimina todos los elementos alumnos.clear();
boolean contains(Object elemento)	Comprueba la existencia del elemento boolean existe = alumnos.contains("Juan");
boolean isEmpty()	Devuelve true si NO hay elementos en el conjunto boolean esvacio = alumnos.isEmpty();

Método add

- Para cargar datos o elementos a un HashSet se utiliza el método add(Object elemento)

Ejemplo:

Carlos
Juan
Luisa
Ana

```
alumnos.add("Carlos");
alumnos.add("Juan”);
alumnos.add(“Luisa");
alumnos.add(“Ana");
```


Método size

- Para obtener el número de elementos del HashSet se utiliza el método size()

Ejemplo:

```
System.out.println(alumnos.size());
```

Método contains

- Para saber si un elemento existe en el HashSet se utiliza el método contains(Object elemento).
- Nos devuelve true si existe

Ejemplo:

```
boolean existe = alumnos.contains("Carlos");
System.out.println( existe );

System.out.println(alumnos.contains("Pepe"));
```


Recorrerlo

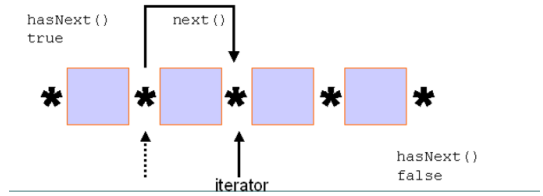
- bucle for-each

```
for(String alum: alumnos){
   System.out.println(alum);
}
```

Recorrerlo usando Iterator

```
String nomAlumno;
Iterator<String> it = alumnos.iterator();
while( it.hasNext()) {
      nomAlumno = it.next(); //devuelve el elemento
      System.out.println(nomAlumno);
    }
```

Se puede pensar a Iterator como un cursor ubicado entre los elementos de la colección
Permite iterar sólo hacia delante



35

Ejemplo con HashSet

```
import java.util.Set;
import java.util.HashSet;
    String pelicula;
   Set<String> peliculas = new HashSet <String>();

    peliculas.add("El señor de los anillos");
    peliculas.add("Titanic");
    peliculas.add("Avatar");

   Iterator<String> it = películas.iterator();
   while(it.hasNext()) {
       pelicula = it.next();
       System.out.println( pelicula );
   }

// otra forma de recorrer
    for (String  pel: peliculas)
            System.out.println("Titulo: " + pel);
  }
}
```

36

HashSet: ejercicio

- Crea un HashSet misAsignaturas para almacenar nombres de las asignaturas que estas matriculado en ciclo DAW. Rellénala
- Recorre el HashSet y muestra por pantalla las asignaturas introducidas. (usa un for-each)
- Recoge una asignatura y comprueba si está almacenada.

37

Operaciones entre colecciones

- Aparecen en List y Set, por herencia de clase Collection

boolean containsAll(c2)	Devuelve true si los elementos de la colección c2 están presentes en el conjunto c1. Es decir c2 es una subcoleccion de c1 boolean estan = alumnos.**containsAll**(alumnos_nuevos);
boolean addAll(c2)	Operación UNION Añade los elementos de la colección c2 a c1 y devuelve true alumnos.**addAll**(alumnos_nuevos);
boolean retainAll(c2)	Operación INTERSECCION Elimina los elementos de la colección c1 que no estan en la colección c2 y devuelve true si hace modificaciones. Deja en c1 los elementos presentes en c1 y c2 alumnos.**retainAll(**alumnos_aprobados);
boolean removeAll(c2)	Operación DIFERENCIA Elimina los elementos de la colección c1 que están en la colección c2 y devuelve true si hace modificaciones boolean cambiados1 = alumnos.**removeAll**(alumnos_becados);

38

HashSet: ejercicio

- Crea otro HashSet convalidar, con las asignaturas que has solicitado su convalidacion de DAW.
- Elimina de misAsignaturas las que aparecen en convalidar
- Muestra contenido de misAsignaturas