INSTITUT THOS I CODINA

M03
Programació

CFGS Desenvolupament d'Aplicacions Web

INSTITUT THOS I CODINA

Col.leccions

INSTITUT THOS I CODINA

Definició

Un objecte "collection", també anomenat contenidor, és un objecte que agrupa múltiples elements en una sola unitat.

- Les col·leccions s'usen per a emmagatzemar, recuperar, manipular i comunicar dades agregades.
- Típicament, representen dades que formen un grup natural.

INSTITUT THOS I CODINA

Definició

Una estructura collection és una arquitectura unificada per a representar i manipular col·leccions

· Totes les estructures de col·leccions contenen el següent:

- Interfícies
- Implementacions
- Algorismes

INSTITUT THOS I CODINA

Objectius

- Redueix l'esforç de programació
- Incrementa la velocitat i qualitat d'un programa
- Permet interoperabilitat entre APIs no relacionades
- Redueix l'esforç per a aprendre i usar noves APIs
- Redueix l'esforç per a dissenyar noves APIs
- Fomenta la reutilització de software

INSTITUT THOS I CODINA

Interfaces

Les interfícies Collection són tipus de dades abstractes que representen col·leccions.

Les interfícies permeten a les col·leccions ser manipulades independentment dels detalls de representació.

– Comportament polimòrfic: formen una jerarquia.

– Un selecciona la que compleix les necessitats com a tipus.

INSTITUT THOS I CODINA

Implementacions

Són les implementacions concretes de les interfícies, que són estructures de dades reusables. Hi ha diversos tipus d'implementacions:

– Implementacions de propòsit general.

– Implementacions de propòsit especial

– Implementacions concurrents

– Implementacions envolupants

– Implementacions de conveniència

– Implementacions abstractes

Implementacions

Interfaces	Implementacions				
	Hash Table	Resizable Array	Tree	Linked List	Hash Table + Linked List
SET	HashSet		TreeSet		LinkedHashSet
LIST		ArrayList		LinkedList	
QUEUE					
MAP	HashMap		TreeMap		LinkedHashMap

INSTITUT THOS I CODINA

Algorismes

Són mètodes polimòrfics que realitzen tractaments útils en objectes i implementen la interfície ***collection***.

Els algorismes més comuns són:

- Ordenament (sorting)
- Remenar (shuffling)
- Manipulacions de rutina (invertir, emplenar, copiar, intercanviar, afegir)
- Cerca (searching)
- Composició (frequency, disjoint)
- Valors extrems (min, max, etc.)

INSTITUT THOS I CODINA

Interface Collection

Totes les col·leccions són genèriques.

public interface Collection<E>...

on E és el tipus d'objecte contingut en la col·lecció.

La interface *Collection* proporciona la funcionalitat bàsica comuna a les col·leccions, com ara mètodes *add* i *remove*.

La interface *Map* crea una correspondència les claus i els valors

INSTITUT THOS I CODINA

Interface Collection

```
public interface Collection<E> extends Iterable<E> {
  // Basic operations
  public int size();
  public boolean isEmpty();
  public boolean contains(E element);
  public boolean add(E element);                    // optional
  public boolean remove(E element);          // optional
  public Iterator<E> iterator();

  // Bulk operations
  public boolean containsAll(Collection<?> c);
  public boolean addAll(Collection<? extends E> c); //optional
  public boolean removeAll(Collection<?> c);      // optional
  public boolean retainAll(Collection<?> c);      // optional
  public void clear(); // optional

// Array operations
  public Object[] toArray();
  public <E>[] toArray(E[] a);
}
```

Collection a Oracle

INSTITUT THOS I CODINA

Interface Collection

for-each

Sintaxis similar a *for-since* accés a l'element recorregut

```
for (Object o : collection)
        System.out.println(o);
```

INSTITUT THOS I CODINA

Interface Collection

És un objecte que permet recorrer una col.lecció i eliminar elements de forma selectiva.

```
public interface Iterator {
  public boolean hasNext();
  public Object next();
  public void remove(); //optional
}
```

```
for (Iterator it =col.iterator(); it.hasNext();) {
  if (!cond(it.next())) it.remove();
}
```

INSTITUT THOS I CODINA

Interface Collection

Operacions massives

Són operacions que realitzen una operació sobre tota la col·lecció

- containsAll()
- addAll()
- removeAll()
- retainAll() elimina de la Collection objectiu tots els elements que no estan continguts en la Collection especificada
- clear()

INSTITUT THOS I CODINA

Interface Collection

Operacions array

Els mètodes toArray són un pont entre collections i APIs antigues que esperen arrays com a entrada.
Les operacions array permeten que el contingut d'una Collection sigui convertit a un array.
Per exemple, si c és una Collection.
Fragment de codi bolca el contingut de c en un array de Object

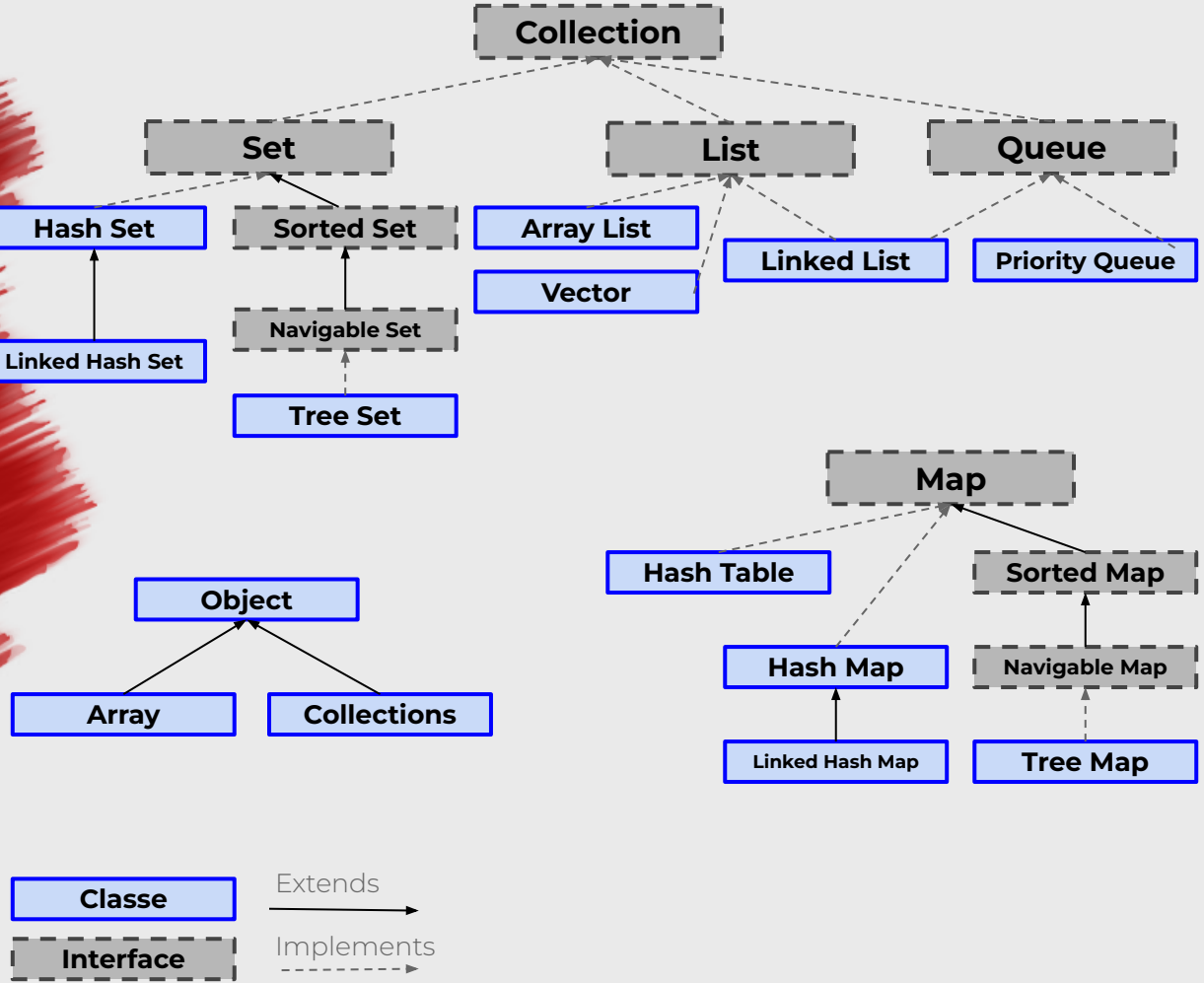
```
Object[ ] a = c.toArray();
```

Si c conté strings (pe Collection<String>).

```
String[ ] a = c.toArray(new String[0]);
```

INSTITUT THOS I CODINA

java.util



INSTITUT THOS I CODINA

interface List

La interfície *List* està implementada per les classes *ArrayList*, *Vector* i *LinkedList*.
Hi ha un autoboxing quan s'agreguen tipus primitius a objectes d'aquestes classes, perquè ells només emmagatzemen referències a objectes.

Característiques:

- Es basa en els índexs. És el màxim diferencial respecte a altres col·leccions.
- Estan permesos els elements duplicats en les llistes.
- Contenen un conjunt de mètodes que ens permeten:
  - Accés posicional
  - Cerca
  - Iteració
  - Manipular un rang d'elements específic.

INSTITUT THOS I CODINA

interface List

Implementacions:

- ArrayList
  - No sincronitzat
  - Ràpida iteració sobre els seus elements
  - Bo per a l'accés posicional
- Vector
  - Sincronitzat
  - Implementa un accés aleatori als elements del vector.
- LinkedList
  - Semblant a ArrayList, amb llista doblement enllaçada.
  - És bo per a implementar cues i piles.
  - És més ràpid per a eliminar i agregar elements.

INSTITUT THOS I CODINA

interface Set

Un *Set* s'ocupa que els elements en la col·lecció siguin únics (no permet duplicats).

El mètode *equals* és el que determina si dos objectes són idèntics (en tal cas només es podrà agregar un a la col·lecció).

INSTITUT THOS I CODINA

interface Set

Implementacions:

- HashSet
  - Unsorted i unordered
  - Fa servir el *hash code* per a inserir i recuperar elements.
- LinkedHashSet
  - És una versió ordenada de HashSet.
  - Bona quan importa l'ordre d'iteració.
- TreeSet
  - És sorted.
  - Emmagatzema elements en un arbre.

INSTITUT THOS I CODINA

interface Map

Als Map es fonamenta en que els identificadors únics.
Quan s'utilitza alguna implementació de *Map* es *mapeja* (relaciona) una clau única (id) amb un valor específic, i per descomptat, d'objectes.
És una estructura de dades que utilitza el parell **clau/valor**.
Les implementacions d'aquesta interfície permeten realitzar cerques basades en un id o buscar sobre la base dels valors.
Els Map necessiten en el mètode *equals()* per a determinar si dos objectes són iguals l'un de l'altre, és a dir, si generen el mateix *hash code*.

INSTITUT THOS I CODINA

interface Map

Implementacions:

- HashMap
  - És unsorted i unordered.
  - Permet agregar a una col·lecció claus i valors nuls.
- Hashtable
  - És sincronitzat.
  - No permet nuls en les claus ni en els valors.
- LinkedHashMap
  - Manté l'ordre d'inserció (ordered).
  - Itera més ràpid que el HashMap.
  - És més lent per a agregar i treure elements que el HashMap.
- TreeMap
  - És sorted.

INSTITUT THOS I CODINA

interface Queue

Aquesta interfície aquesta dissenyada per a contenir una llista d'objectes que seran processats en alguna forma específica (una estructura de dades tipus cua).
Són dissenyades com **FIFO** (first in, first out).
Les cues suporten tots els mètodes estàndard de *Collection*, i agreguen mètodes per a obtenir i veure elements d'una cua.

Aquesta interfície té una implementació, que és la *PriorityQueue*.

INSTITUT THOS I CODINA

velocitats

Llistes i conjunts

Estructura	get	add	remove	contains
ArrayList	O(1)	O(1)	O(n)	O(n)
LinkedList	O(n)	O(1)	O(1)	O(n)
HashSet	O(1)	O(1)	O(1)	O(1)
LinkedHash Set	O(1)	O(1)	O(1)	O(1)
TreeSet	O(log n)	O(log n)	O(log n)	O(log n)

Mapes

Estructura	get	put	remove	containsKey
HashMap	O(1)	O(1)	O(1)	O(1)
LinkedHashMap	O(1)	O(1)	O(1)	O(1)
TreeMap	O(log n)	O(log n)	O(log n)	O(log n)