Introducción al Lenguaje Java

Programación Orientada a Objetos

San Salvador de Jujuy

UNJu – Facultad de Ingeniería
Ing. José Zapana

Contenido

- Strings
- Collections

Strings - Creación

- String cadena = “unaCadena”
- String cadena = “uno + “dos”
- String cadena = new String (“unaCadena”)

Strings - Operaciones

- Búsqueda de longitud
  - int length();
- Búsqueda de un carácter específico
  - char charAt(int index)
- Devolución de una subcadena
  - String substring(int inicio, int fin)

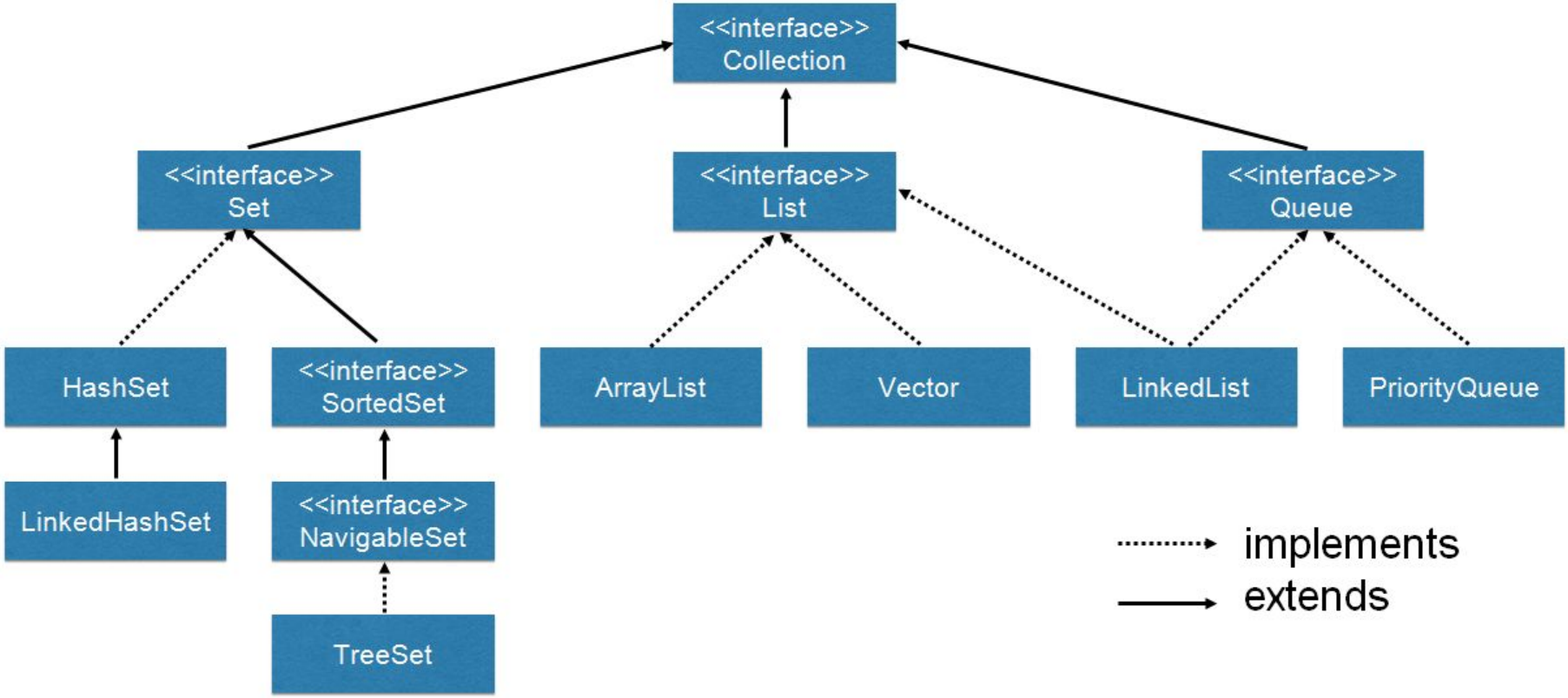
Operaciones adicionales

- Conversión en mayúsculas / Minúsculas
  - toUpperCase(), toLowerCase(),
- Recortes de espacio
  - trim()
- Búsqueda de una subcadena
  - indexOf(String str)
- Comparación de objetos
  - equals(str), equalsIgnoreCase(str)

Java Collection Frameworks

- Es una arquitectura de API para la gestión de un grupo de objetos que se pueden manipular independientemente de su implementación interna
  - Se encuentra en el paquete java.util
  - Se define mediante seis interfaces principales y algunas clases de implementación
    - Interfaz Collection
    - Interfaz Set
    - Interfaz List
    - Interfaz Map
    - SortedSet y SortedMap para Set y Map ordenados

Java Collections Framework



Interfaz Collection

-Collection

- `int` **`size`**`()`
- `boolean` **`empty`**`()`
- `boolean` **`contains`**`(Object elem)`
- `Iterator` **`iterator`**`()`
- `Object[]` **`toArray`**`(), Object[]` **`toArray`**`(Object dest[])`
- `boolean` **`add`**`(Object elem),`
- `boolean` **`remove`**`(Object elem)`
- `void` **`clear`**`()`

- **List** – Una colección cuyos elementos permanecen en un orden particular a menos que se modifique la lista

  - `void` **`add`**`(int index, Object element)`
  - `Object` **`remove`**`(int index)`
  - `Object` **`get`**`(int index)`
  - `Object` **`set`**`(int index, Object element)`
  - `int` **`indexOf`**`(Object o)`
  - `int` **`lastIndexOf`**`(Object o)`
  - `List` **`subList`**`(int min, int max)`

Interfaz Collection (cont.)

- **Set** Conjunto donde no puede haber elementos repetidos, y cuyos elementos no se almacenan necesariamente siguiendo un orden particular.
  - Mismos métodos que `Collection` con otro contrato.

- **SortedSet** – Conjunto con elementos ordenados.
  - `Object` **`first`**`()`
  - `Object` **`last`**`()`
  - `SortedSet` **`subSet`**`(Object fromElement, Object toElement)`
  - `SortedSet` **`headSet`**`(Object toElement)`
  - `SortedSet` **`tailSet`**`(Object fromElement)`

Interfaz Map

- **Map**
  - Un objeto que asocia claves con valores.
  - No puede tener claves duplicadas.
  - `Object` **`put`**`(Object key, Object value);`
    `Object` **`remove`**`(Object key);`
    `Object` **`get`**`(Object key);`
  - **`containsKey, containsValue, isEmpty, size`**
  - Proporciona tres vistas de colección: colección de claves (**`keySet`**), colección de valores (**`values`**), colección de asociaciones clave-valor (**`entrySet`**).
- **SortedMap:** Un mapa cuyas claves están ordenadas.
  - `Object` **`firstKey`**`(), Object` **`lastKey`**`(), SortedMap` **`subMap`**`(Object minKey, Object maxKey), SortedMap` **`headMap`**`(Object maxKey), SortedMap` **`tailMap`**`(Object minKey)`

Uso de ArrayList y HashTable

- **ArrayList:**
  - Es una interfaz redimensionable de la interfaz **List**
  - Permite la manipulación del tamaño de la matriz.
  - Tiene una capacidad que aumenta a medida que se agregan elementos a la lista
- **HashTable:**
  - Es una clases heredada similar a las implementaciones **Map**.
  - Se utiliza para almacenar objetos arbitrarios indexados por otro objeto arbitrario.
  - Se utiliza normalmente con **String** como clave para almacenar objetos como valores.

Declaración de colecciones

```
import java.util.*;

public class ColeccionSimple {
    public static void main(String args[]) {
        List<Integer> lista = new ArrayList();
        for( int i=0; i < 10; i++ )
            lista.add(new Integer(i));

        for (int i: lista)
           System.out.println(i);
    }
}
```

Ejemplo con **HashMap**

```
public class Principal{

        public static void main(String[]args){
             Map<Integer,String> map = new HashMap();
             map.put(1, "Argentina");
             map.put(2, "Brasil");
             …
      String nombrePais = (String)map.get(1);
          System.out.println(nombrePais);
    }
}
```

Colecciones en Java (comparativa rápida)

Colección	Características principales	Cuándo usarla (ejemplos)	Complejidad típica
List (ArrayList)	- Ordenada - Permite duplicados - Acceso por índice	- Recorrer en orden - Carrito compras - Mostrar resultados de DB	`get/set` O(1) `contains` O(n) `insert/remove medio` O(n)
Set (HashSet)	- No permite duplicados - No tiene índice - Orden no garantizado	- Emails únicos - Permisos de usuario - Evitar duplicados en DB	`add/remove/contains` O(1) promedio
Set (TreeSet)	- No duplicados - Orden natural/comparador	- Ranking - Lista ordenada de apellidos	`add/remove/contains` O(log n)
HashMap<K,V>	- Pares clave→valor - Claves únicas - No mantiene orden	- Agenda contactos (DNI→Persona) - Diccionario - Índice por ID	`get/put/remove` O(1) promedio