Linkia FP

Formación Profesional Oficial a Distancia

Linkia FP

DAW – M03 – Clase 05

Estructuras de datos avanzadas

Linkia FP

Contenidos

- ArrayList
- HashMap
- equals y hashCode

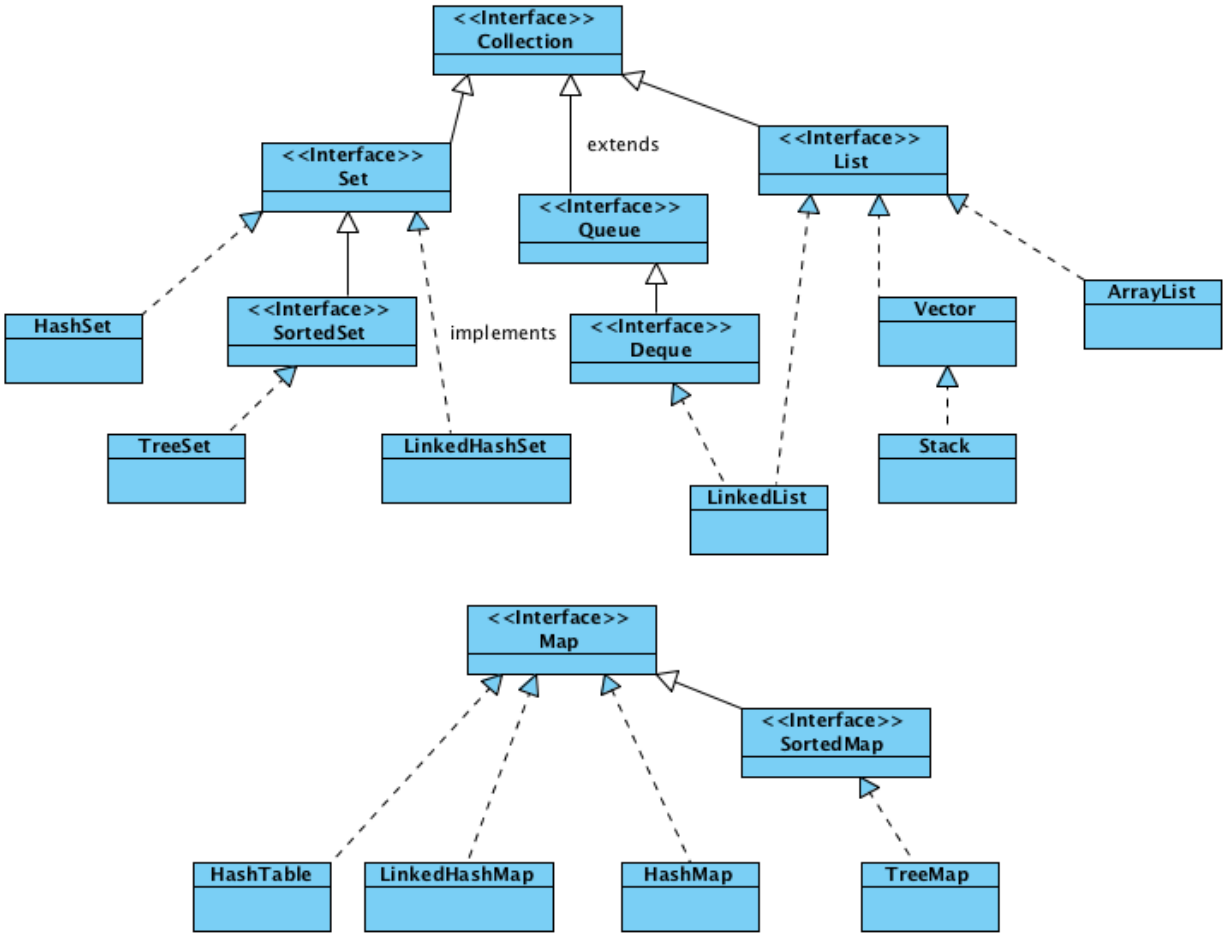
Linkia FP

Introducción

- El API de Java nos proporciona el framework de las colecciones, que nos permite utilizar diferentes estructuras de datos para almacenar y recuperar objetos de cualquier clase. Dichas colecciones no forman parte del lenguaje, sino que son clases definidas en el paquete java.util.

Linkia FP

Introducción



Linkia FP

Introducción

Enlace interesante (chuleta):

http://zeroturnaround.com/rebellabs/Java-Collections-Cheat-Sheet/

Linkia FP

ArrayList

- La principal desventaja de los arrays estáticos que hemos visto hasta ahora es que su tamaño debe ser definido previamente antes de ser usado.
- Eso puede ser un gasto innecesario de memoria o quedarse corto al momento de añadir un nuevo elemento a la colección dada.
- Java soluciona los inconvenientes de los arrays tradicionales con el uso de arrays dinámicos.

Linkia FP

ArrayList

- De forma general un ArrayList en Java se crea de la siguiente forma:

```
ArrayList nombreArray = new ArrayList();
```

- Esta instrucción crea el ArrayList nombreArray vacío.
- Un ArrayList declarado así puede contener objetos de cualquier tipo.
- Para poder utilizar un array dinámico es necesario importar:

```
import java.util.ArrayList;
```

Linkia FP

Ejemplo ArrayList

```
ArrayList a = new ArrayList();
a.add("Hola");
a.add(3);
a.add('a');
a.add(23.5);
```

- Los elementos del ArrayList a son:

"Hola" 2 'a' 23.5

Linkia FP

ArrayList

- Es decir, un ArrayList puede contener objetos de tipos distintos.
- En este ejemplo, el primer objeto que se añade es el String “Hola”.
- El resto no son objetos. Son datos de tipos básicos pero el compilador los convierte automáticamente en objetos de su clase contenedora antes de añadirlos al array.

Linkia FP

ArrayList

- Un array al que se le pueden asignar elementos de distinto tipo puede tener alguna complicación a la hora de trabajar con él.
- Por eso, una alternativa a esta declaración es indicar el tipo de objetos que contiene.
- En este caso, el array solo podrá contener objetos de ese tipo.

Linkia FP

ArrayList

- De forma general:

```
ArrayList<tipo> nombreArray = new ArrayList<tipo>();
```

- tipo debe ser una clase. Indica el tipo de objetos que contendrá el array.
- No se pueden usar tipos primitivos. Para un tipo primitivo se debe utilizar su clase contenedora.
- Por ejemplo:

```
ArrayList<Integer> numeros = new ArrayList<Integer>();
```

- Crea el array numeros de enteros.

Linkia FP

Agregar un elemneto

- Esta es la forma general

```
nombreDelArray.add(elemento);
```

- Ejemplos:

```
arrayTexto.add("Hola");
arrayTexto.add("¿Cómo estás?”);
arrayEnteros.add(10);
arrayDecimales.add(34.9);
String nombre = “Juan”;
arrayTexto.add(nombre);
```

Linkia FP

Tamaño del ArrayList

- Esta es la forma general

```
nombreDelArray.size();
```

- retorna un entero con el número de elementos que tiene el array almacenados.
- Ejemplos:

```
arrayTexto.size();
arrayEnteros.size();
arrayDecimales.size();
```

Linkia FP

Acceder a una posición

```
nombreDelArray.get(posicion);
```

- Ejemplos:

```
int a = arrayEnteros.get(0);
//almacena en a lo mismo que tiene el array en
la posición cero.
double p = arrayDecimales.get(0)+
arrayDecimales.get(1);
//almacena en p la suma de los valores de las
posiciones cero y uno del array
System.out.println(arrayTexto.get(0));
//imprime la posición cero del array de textos
dados
```

Linkia FP

Principales métodos

Método	Descripción
size()	Devuelve el número de elementos (int)
add(x)	Añade el objeto x al final. Devuelve true.
add(posicion, x)	Inserta el objeto x en la posición indicada.
get(posicion)	Devuelve el elemento que está en la posición indicada.
remove(posicion)	Elimina el elemento que se encuentra en la posición indicada. Devuelve el elemento eliminado.
remove(x)	Elimina la primera ocurrencia del objeto x. Devuelve true si el elemento está en la lista.
clear()	Elimina todos los elementos.
set(posición, x)	Sustituye el elemento que se encuentra en la posición indicada por el objeto x. Devuelve el elemento sustituido.
contains(x)	Comprueba si la colección contiene al objeto x. Devuelve true o false.
indexOf(x)	Devuelve la posición del objeto X. Si no existe devuelve -1

Linkia FP

Documentación

- Puedes consultar todos los métodos en:

https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Linkia FP

Recorrer un ArrayList

1. Podemos recorrerlo de forma clásica con un bucle for:

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

Linkia FP

Recorrer un ArrayList

2. Con un bucle foreach:

Si suponemos el array de enteros llamado numeros:

```
for (Integer i : numeros) {
     System.out.println(i);
}
```

Linkia FP

Recorrer un ArrayList

2b. Con un bucle foreach:

Si el array contiene objetos de tipos distintos o desconocemos el tipo:

```
for (Object o : nombreArray) {
     System.out.println(o);
}
```

Linkia FP

Recorrer un ArrayList

3. Utilizando un objeto Iterator. – Ejemplo:

```
ArrayList<Integer> numeros = new
ArrayList<Integer>();
…..
// se insertan elementos
.....
Iterator it = numeros.iterator();
//se crea el iterador it para el array numeros
while(it.hasNext())
//mientras queden elementos
System.out.println(it.next());
//se obtienen y se muestran
```

Linkia FP

Recorrer un ArrayList

3. Utilizando un objeto Iterator. – Ejemplo:

```
ArrayList<Integer> numeros = new
ArrayList<Integer>();
…..
// se insertan elementos
.....
Iterator it = numeros.iterator();
//se crea el iterador it para el array numeros
while(it.hasNext())
//mientras queden elementos
System.out.println(it.next());
//se obtienen y se muestran
```

Linkia FP

Recorrer un ArrayList

4. Usando stream y lambda (Java 8)

```
numeros.stream().forEach((i) -> {
            System.out.println(i);
        });
```

Linkia FP

HashMap

- Un HashMap es una implementación de la interfaz Map de Java.
- Es una colección de objetos.
- El HashMap nos permite almacenar pares clave/valor, de tal manera que una clave sólo puede tener un valor, es decir, las claves no se repiten.

Linkia FP

HashMap

- Si añadimos un elemento cuya clave ya existe, no se generará un nuevo elemento en el HashMap, sino que se reescribe el valor que pertenece a la clave existente.
- Hay que tener en cuenta que el HasMap no admite valores nulos.

Linkia FP

HashMap

- La particularidad de los HashMap radica en cómo se ordenan sus elementos.
- Se utiliza un algoritmo que, a partir de la clave, genera otra clave “hash”, que será la que determine el orden en que se almacenarán en memoria los elementos. (hashCode()).

Linkia FP

HashMap

- Esta forma de ordenar los objetos hace que el tiempo de ejecución de operaciones como coger un elemento o añadir uno nuevo permanezca constante independientemente del tamaño del HashMap.

Linkia FP

HashMap

- Para usar HashMap en nuestro programa, tendremos que importar la librería java.util.HashMap.
- La declaración del HashMap es la siguiente:

```
HashMap<TipoClave, TipoDato> map = new HashMap<>()
```

- Podemos declarar un HashMap sin especificar el tipo de datos de las claves y los valores a almacenar. Esto nos permite añadir cualquier tipo de objeto, lo cual no es recomendable, pues para mostrar los datos del HashMap podemos tener problemas de conversión de tipos de datos.

Linkia FP

Agregar un elemento

- Esta es la forma general

```
nombreHashMap.put(clave, elemento);
```

- Ejemplo:

```
HashMap<String, Integer> notas = new
HashMap<>();
notas.put("Juan", 8);
notas.put("Isabel", 5);
```

Linkia FP

Recorrer un HashMap

```
 Iterator i = notas.keySet().iterator();
 while (i.hasNext()) {
      String clave = (String) i.next();
      System.out.println("Clave: "+clave+" Valor:
"+notas.get(clave));
 }
```

Linkia FP

Recorrer un HashMap

```
 for (String clave : notas.keySet()) {
     System.out.println("Clave: "+clave+"
Valor: "+notas.get(clave));
  }
```

Linkia FP

Recorrer un HashMap

```
for (Map.Entry<String, Integer> entidad :
notas.entrySet()) {
      System.out.println("Clave:
"+entidad.getKey()+" Valor "+entidad.getValue());
}
```

Linkia FP

Recorrer un HashMap

Java 8

```
notas.forEach((clave, valor) ->
System.out.println("Clave "+clave+" Valor "+valor));
```

Linkia FP

Principales métodos

Método	Descripción
clear()	Vacía el HashMap
containsKey(key)	Indica si el HashMap contiene un elemento con la clave key.
containsValue(value)	Indica si el HashMap contiene un elemento con el valor value.
entrySet()	Devuelve una copia del HashMap en forma de colección. Cualquier cambio en la colección afecta directamente al HashMap.
get(key)	Devuelve el valor asociado a la clave key, o nulo si no existe la clave.
isEmpty()	Indica si el HashMap está vacío.
keySet()	Devuelve una colección con las claves del HashMap.

Linkia FP

Principales métodos

Método	Descripción
put(key, value)	Añade un elemento al HashMap con la clave key y el valor value. Si ya existe un elemento con clave key, sustituye su valor a value.
putAll(map)	Copia todos los elementos de map en el HashMap que invoca el método.
remove(key)	Elimina el elemento con clave key del HashMap si existe.
size()	Devuelve un número entero que es el número de elementos que tiene el HashMap.
values()	Devuelve una colección con los valores que contiene el HashMap.

Linkia FP

Documentación

- Puedes consultar todos los métodos en:

https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html

Linkia FP

Equals()

- El método equals nos devuelve un booleano que indica si dos elementos son iguales o no.
- Por defecto, se considera que dos elementos son iguales si tienen la misma clave.
- Podemos ir más allá y hacer que, por ejemplo, para que se consideren iguales dos elementos si tienen el mismo valor.
- Por ejemplo, consideramos dos personas iguales si tienen el mismo nombre y apellidos

Linkia FP

hashCode()

- Con el método hashCode se calcula el código hash que determinará el orden de los elementos en el HashMap.
- Podemos sobrescribirlo si queremos que este código se calcule de una forma concreta.
- Por ejemplo, podemos ordenar personas por la longitud de su nombre.

Linkia FP

Formación Profesional Oficial a Distancia