Java

Tema 3: Colecciones

Clases:

ArrayDeque - HashMap

Contenido

5.- Colas: clase **ArrayDeque**

6.- Mapas/diccionarios: clase **HashMap**

clase **ArrayDeque**

3

La clase ArrayDeque

- Representa una cola con acceso al primer y último elemento.
- Representa estructuras
  - LIFO: Last In First Out (pila)
  - FIFO: First In First Out (cola)
- Puede contener elementos repetidos
- Mejor rendimiento que las clases Stack y LinkedList
- Acceso aleatorio y recorrido muy lentos.
- Es un tipo de "Deque"

4

Declaración

Deque<*clase*> *variable* = new **ArrayDeque**<[*clase*]>( );

El tipo de datos debe ser una clase.

Ejemplos:

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

ArrayDeque<String> cola= new ArrayDeque<>();
```

5

Métodos manejo del final (end)

boolean add(Object o) **boolean offer(Object o)** **void addLast(Object o)** **boolean offerLast(Object o)**	Añade el elemento al final (cola)
Object getLast() **Object peekLast()**	Devuelve el último elemento del final (cola)
Object removeLast() **Object pollLast()**	Devuelve y borra el último elemento del final (cola)

6

Métodos manejo del principio (front)

- Si queremos una PILA sólo usaríamos estos métodos

void addFirst(Object o) **void offerFirst(Object o)** **void push(Object o)**	Añade el elemento al principio (cabeza)
Object getFirst() **Object peekFirst()** **Object element()** **Object peek()**	Devuelve el primer elemento del principio (cabeza)
Object removeFirst() **Object pollFirst()** **Object remove()** **Object poll()** **Object pop()**	Devuelve y borra el primer elemento del principio (cabeza)

7

Método size

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

Ejemplo:

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

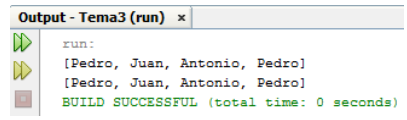
8

Método toString

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

Ejemplo:

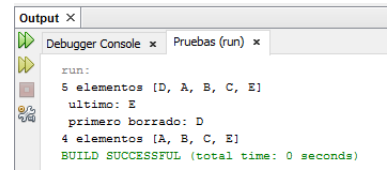
```
System.out.println(cola.toString());

System.out.println(cola);
```



9

Ejemplo

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

cola.add("A");
cola.add("B");
cola.add("C");
//añade elemento al principio
cola.addFirst("D");

//añade elemento al final
cola.addLast("E");

System.out.println(cola.size() + " elementos " + cola);

//recupera el ultimo elemento
System.out.println(" ultimo: " + cola.getLast());

// recupera y borra el primero
System.out.println(" primero borrado: " + cola.removeFirst());

System.out.println(cola.size() + " elementos " + cola);
```



10

Recorrerlo

- El rendimiento al recorrer es muy bajo.
  - O el bucle for/each.

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

  - Usando un iterator.

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

11

ArrayDeque: ejercicio

- Crea un ArrayDeque para almacenar una lista de espera para comprar el nuevo Iphone 8. Apunta a 5 personas
- Muestra la lista indicando cuantos hay y el orden de cada uno
- Saca de la lista a los 3 primeros y muestralos por pantalla

12

clase **HashMap**

La clase HashMap

- La clase HashMap crea listas de datos en los que se almacenan parejas de elementos clave - valor.
- Lo habitual es utilizar un String como clave para almacenar objetos de otra clase como valores.
- NO permite tener claves duplicadas. Pero si valores duplicados.

Clave	Valor
NIF	**NOMBRE**
19556432Z	Juan López
23789543A	Sara Baras
22569345X	Pablo González
18543123Y	Luisa Pérez

Declaración

Map<clave_clase, valor_clase> variable
= new **HashMap**<>( );

Ejemplo:

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

clave	valor
Carlos	963258741
Ana	963258741
primo	626987458
Trabajo	963569852

Métodos principales

int size()	Devuelve el número de parejas el Map
Object get(Object key)	Devuelve un objeto dada su clave-key
Object remove(Object key)	Elimina una pareja con una clave-key dada
void clear()	Elimina todas las parejas clave-valor agenda.clear();
boolean containsKey(Object key)	Comprueba la existencia de una clave. boolean existe = agenda.containsKey("Ana");
boolean containsValue(Object value)	Comprueba la existencia de un valor. boolean existe = agenda.containsValue(963258741);
Set keySet()	Devuelve las claves del Map en un Set (conjunto) Set claves = agenda.keySet();
Collection values()	Devuelve una colección con los valores del Map Collection valores = agenda.values();

Método put

- Para almacenar elementos en un HashMap se utiliza el método put(Object clave,Object valor)

Ejemplo:

Carlos	963258741
Ana	963258741
Primo	626987458
Trabajo	963569852

```
agenda.put("Carlos",963258741);
agenda.put("Ana",963258741);
agenda.put("Primo",626987458);
agenda.put("Trabajo",963569852);
```

17

Método size

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

Ejemplo:

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

18

Método get

- Para recuperar un valor del HashMap se utiliza el método get(Object clave)
- El valor siempre se recupera en función de la clave. Si no existe la clave devuelve null

Ejemplo:

```
System.out.println("tlf:" + agenda.get("Carlos") );
System.out.println("tlf:" + agenda.get("Trabajo") );
System.out.println("tlf:" + agenda.get("Sara") );
```


Recorrer

- Usando un bucle for-each

```
Integer tlf;
for(String  nombre: agenda.keySet() ){
    tlf= agenda.get(nombre);
    System.out.println(nombre + "-" + tlf);
}
```

- Para recorrer un HashMap se usa un bucle con iterador

```
String nombre;
Integer tlf;
Iterator<String> it = agenda.keySet().iterator()
while( it.hasNext()) {
     nombre= it.next(); //devuelve una clave
     tlf = agenda.get(nombre);
    System.out.println(nombre + "-" + tlf);
}
```

Ejemplo con HashMap

```
import java.util.*;

public class TestHashMap {
  public static void main(String[] args) {
    String cod_banco;
    Map <String,String> bancos = new HashMap <>();

     bancos.put("1827","BBVA");
     bancos.put("0049","Santander");
     bancos.put("2038","Bankia");
     bancos.put("2100","La Caixa");

    Iterator<String> it = bancos.keySet().iterator();
    while(it.hasNext()) {
      cod_banco = it.next();
      System.out.println(cod_banco + ": " + bancos.get(cod_banco));
    }
  }
}
```


HashMap: ejercicio

- Crea un HashMap para almacenar los nombres de los meses del año y la cantidad de días de cada uno. Rellénala
- Recorre el HashMap y muestra por pantalla los meses introducidos, indicando entre paréntesis el número de días de cada uno. (usa un bucle for-each)