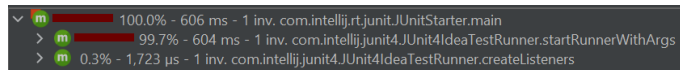


Juan Diego Ávila 20090  
Marco Pablo Orózco 20857  
Daniel González Carrillo 20293

## Hoja de trabajo 6

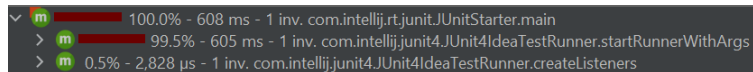
### Mostrar la información de todas las cartas en orden

#### 1. HashMap:



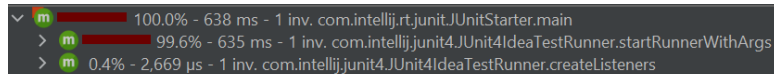
```
✓ m 100.0% - 606 ms - 1 inv. com.intellij.rt.junit.JUnitStarter.main
> m 99.7% - 604 ms - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.startRunnerWithArgs
> m 0.3% - 1,723 μs - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.createListeners
```

#### 2. LinkedHashMap:



```
✓ m 100.0% - 608 ms - 1 inv. com.intellij.rt.junit.JUnitStarter.main
> m 99.5% - 605 ms - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.startRunnerWithArgs
> m 0.5% - 2,828 μs - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.createListeners
```

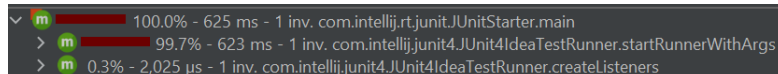
#### 3. TreeMap:



```
✓ m 100.0% - 638 ms - 1 inv. com.intellij.rt.junit.JUnitStarter.main
> m 99.6% - 635 ms - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.startRunnerWithArgs
> m 0.4% - 2,669 μs - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.createListeners
```

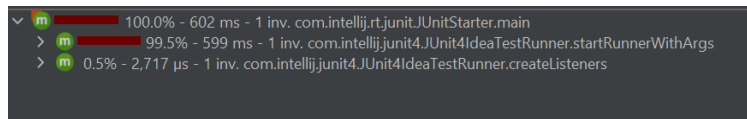
### Mostrar información de todas las cartas:

#### 1. HashMap:

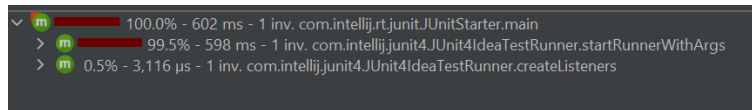


```
✓ m 100.0% - 625 ms - 1 inv. com.intellij.rt.junit.JUnitStarter.main
> m 99.7% - 623 ms - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.startRunnerWithArgs
> m 0.3% - 2,025 μs - 1 inv. com.intellij.junit4.JUnit4IdeaTestRunner.createListeners
```

## 2. LinkedHashMap:



## 3. TreeMap:



	Tiempo de cartas ordenadas (ms)	Tiempo de cartas desordenadas (ms)
HashMap	606	625
LinkedHashMap	608	602
TreeMap	635	602

## Complejidad de HashMap

```
//Se imprimen todas las cartas con sus tipos de forma desordenada.  
String res = "";  
int cont = 0;  
for(String keys: coleccion.keySet()){  
    res += keys + " " + coleccion.get(keys) + "\n";  
    cont++;  
}  
System.out.println(res);
```

Para complejidad promedio del método get de HashMap

1. Recorrer todas las llaves del hashmap con for:  $n$
2. Utilizar el método get:  $c1$  (Luka, 2021)
3. Imprimir los resultados:  $c2$

Donde  $n$  es la cantidad de elementos en el hashmap,  $c1$  y  $c2$  constantes.

$$Tiempo = n(c1) + c2$$

$$Complejidad = O(n)$$

Para el peor caso del método get de HashMap

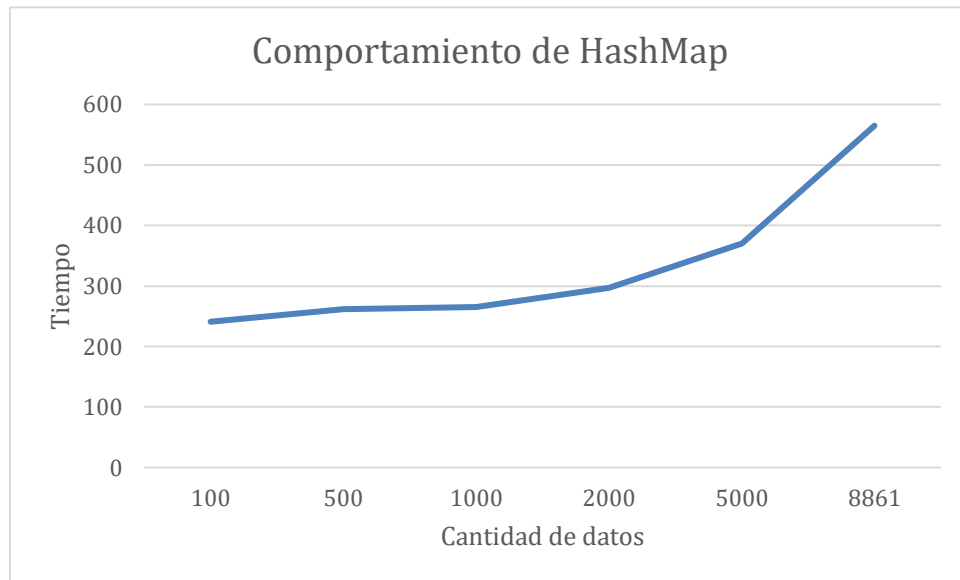
1. Recorrer todas las llaves del hashmap con for:  $n$
2. Utilizar el método get:  $n$  (Luka, 2021)
3. Imprimir los resultados:  $c1$

Donde  $n$  es la cantidad de elementos en el hashmap y  $c1$  constante

$$Tiempo = n(n) + c1$$

$$Complejidad = O(n^2)$$

Este último resultado se puede observar al hacer una gráfico de tiempo contra cantidad de datos mostrados. Donde se observa que, al aumentar la cantidad de datos por mostrar, estos empiezan a aumentar de una forma cuadrática.



## Referencias

Luka. (2021, March 1). *Data Structure: Hash Table & Big O Notation*. Medium.

<https://lukabaramishvili.medium.com/data-structure-hash-table-big-o-notation-a2ee869be861>