

- Importa en Netbeans o proxecto RendimientoColecciones

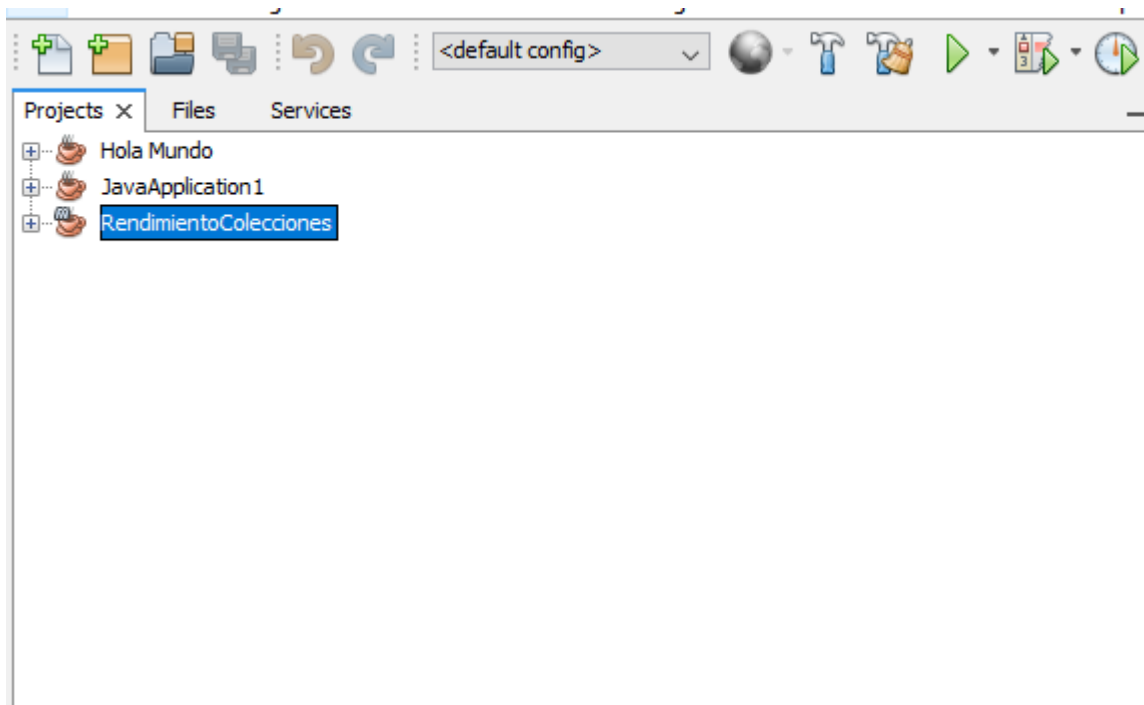


Figura 1: Importación de proyecto

- Executa o profile do mesmo, configurando a sesión seleccionando “Métodos” (ollo: se estás en Windows corríxe o bug que se explicou arriba no cadro azul). Fai unha captura de pantalla e comenta a diferenza de uso de CPU dos distintos método empregados.

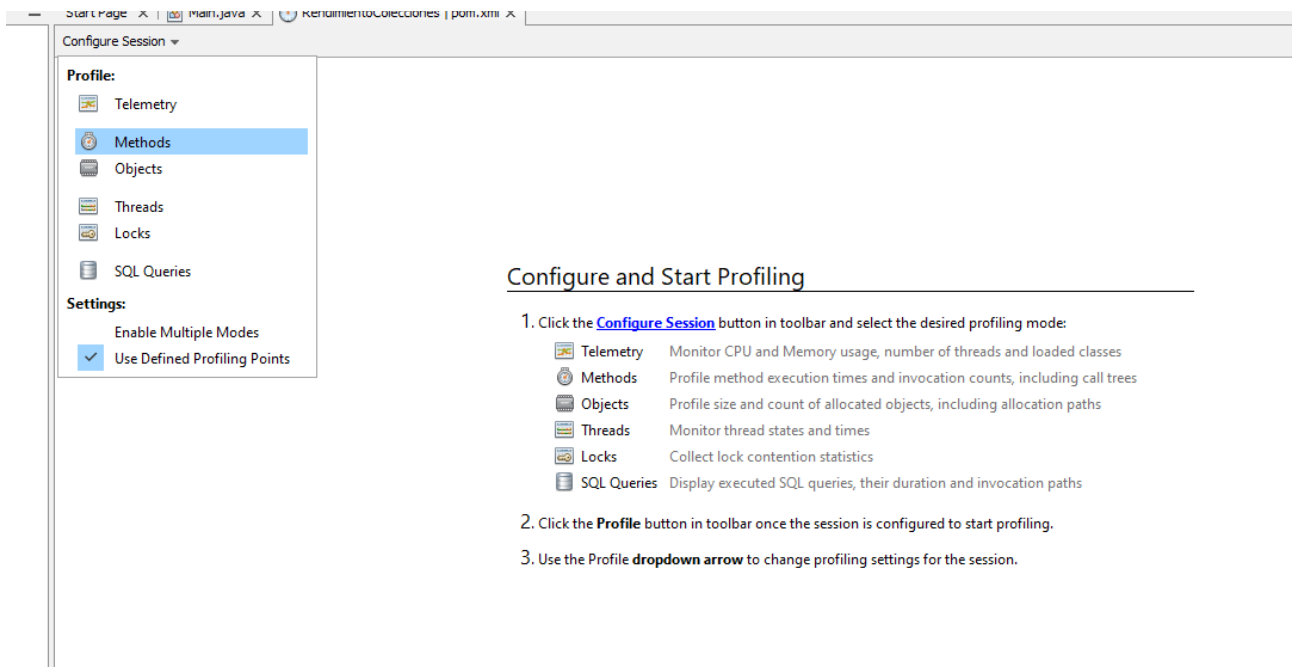


Figura 2: Profile: métodos

| Name              | Total Time       | Total Time (CPU) |
|-------------------|------------------|------------------|
| main              | 1,799 ms (100 %) | 1,628 ms (100 %) |
| Reference Handler | 0,0 ms (- %)     | 0,0 ms (- %)     |
| Finalizer         | 0,0 ms (- %)     | 0,0 ms (- %)     |
| Common-Cleaner    | 0,0 ms (- %)     | 0,0 ms (- %)     |
| DestroyJavaVM     | 0,0 ms (- %)     | 0,0 ms (- %)     |

Figura 3: Profile: métodos 2

Se puede ver como la clase principal “main” es en la única donde la CPU trabaja porque es la única que tiene métodos.

- Executa o profile opción Telemetría e fai unha captura da pantalla.

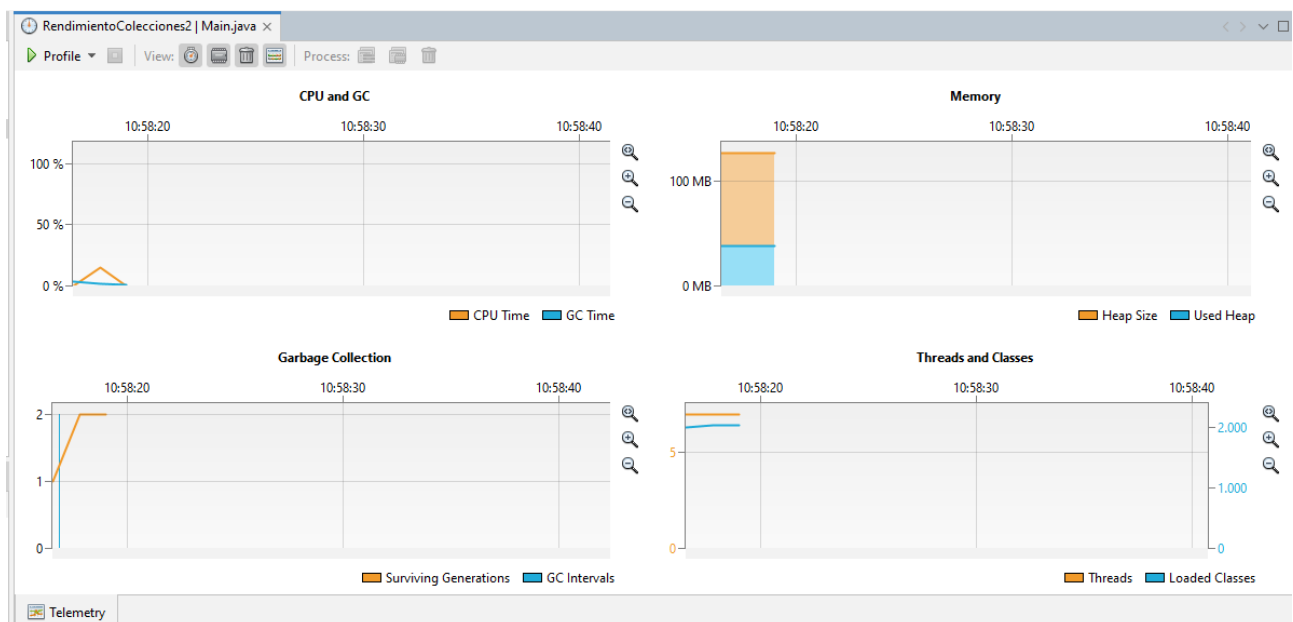
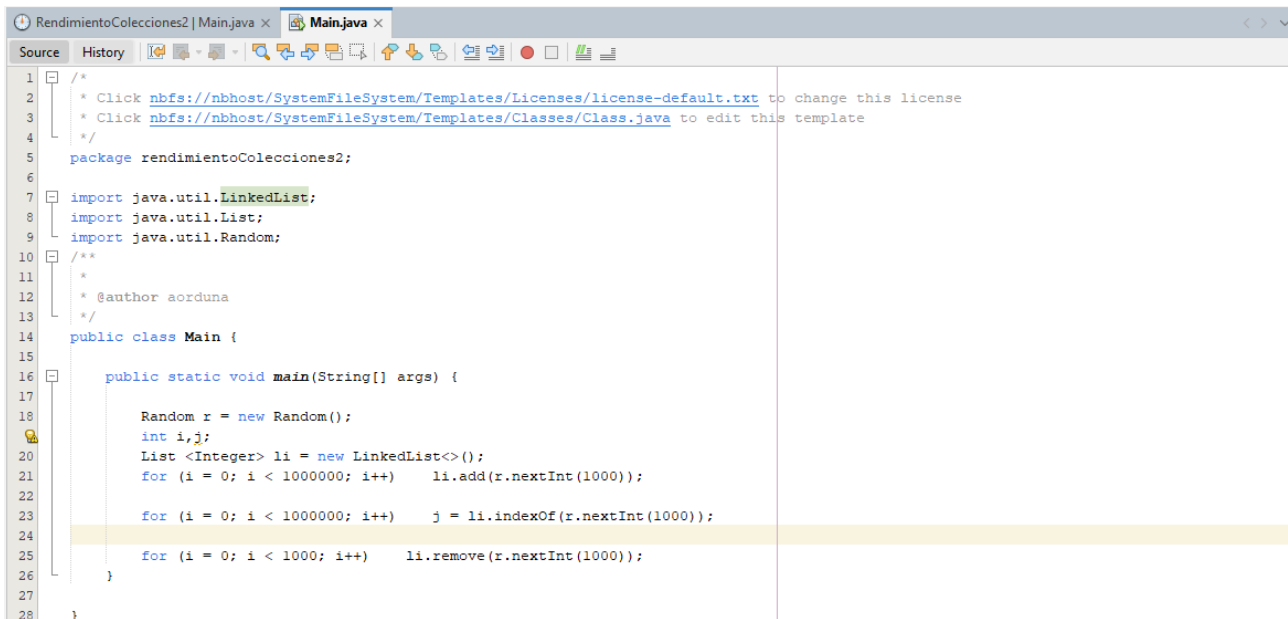


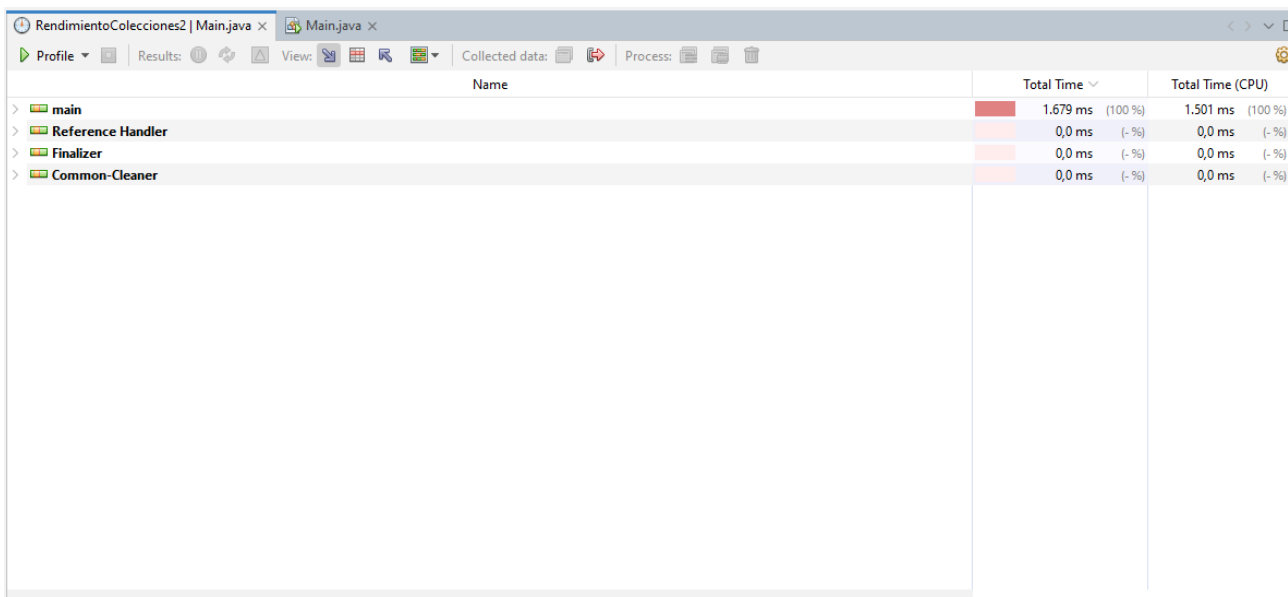
Figura 4: Profile: Telemetría

- Substitúe a ArrayList por LinkedList (logo fai botón dereito > Fix Imports) e repite a todos os pasos anteriores.



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package rendimientoColecciones2;
6
7  import java.util.LinkedList;
8  import java.util.List;
9  import java.util.Random;
10
11  /**
12   * @author aorduna
13   */
14  public class Main {
15
16      public static void main(String[] args) {
17
18          Random r = new Random();
19          int i, j;
20          List<Integer> li = new LinkedList<>();
21          for (i = 0; i < 1000000; i++)    li.add(r.nextInt(1000));
22
23          for (i = 0; i < 1000000; i++)    j = li.indexOf(r.nextInt(1000));
24
25          for (i = 0; i < 1000; i++)    li.remove(r.nextInt(1000));
26      }
27  }
28  }
```

Figura 5: Sustitución de ArrayList



| Name              | Total Time       | Total Time (CPU) |
|-------------------|------------------|------------------|
| main              | 1.679 ms (100 %) | 1.501 ms (100 %) |
| Reference Handler | 0,0 ms (- %)     | 0,0 ms (- %)     |
| Finalizer         | 0,0 ms (- %)     | 0,0 ms (- %)     |
| Common-Cleaner    | 0,0 ms (- %)     | 0,0 ms (- %)     |

Figura 6: Profile: Métodos con LinkedList

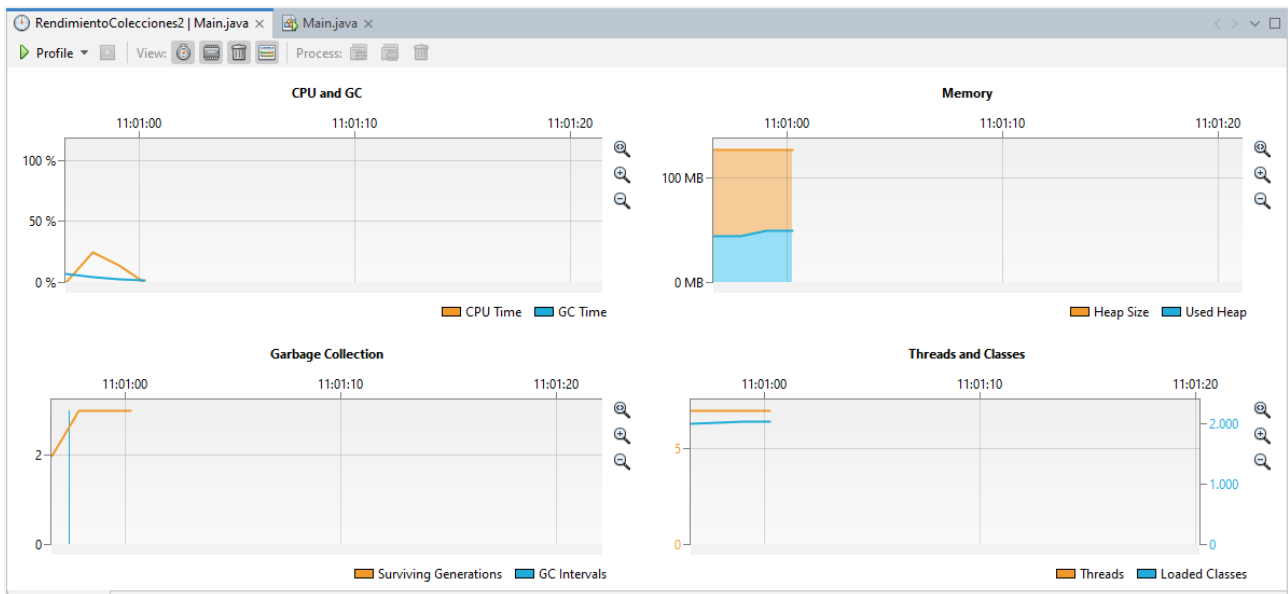


Figura 7: Profile: Telemetría con LinkedList

▪ Fai unha comparativa entón do rendemento de LinkedList vs. ArrayList.

En las capturas anteriores, se puede ver como usando ArrayList, tarda más en ejecutar pero consume menos memoria y CPU.

En cambio, usando LinkedList, tardará menos pero consumirá un poco más de memoria y de CPU