

Hoja de trabajo 5

C.

a.

HashMap			
5			
→	Menu.main(String[]) Menu.java	78	100 %
⇒	java.io.PrintStream.println(String) PrintStream	62	79 %
⇒	java.util.HashMap\$KeyIterator.next() HashMap	15	19 %
6			
⇒	java.io.PrintStream.println(String) PrintStream.java	78	100 %
⇒	Menu.main(String[]) Menu.java:295	46	59 %
⇒	Menu.main(String[]) Menu.java:299	15	19 %
⇒	Menu.main(String[]) Menu.java:303	15	19 %

b.

Linked			
5			
→	Menu.main(String[]) Menu.java	125	100 %
⇒	java.io.PrintStream.println(String) PrintStream	93	74 %
⇒	java.lang.invoke.Invokers\$Holder.linkToTargetMethod() Invokers\$Holder	31	25 %
6			
→	Menu.main(String[]) Menu.java	109	100 %
⇒	java.io.PrintStream.println(String) PrintStream	93	85 %
⇒	java.lang.invoke.Invokers\$Holder.linkToTargetMethod() Invokers\$Holder	15	14 %

c.

Tree			
5			
➔ Menu.main(String[]) Menu.java	93	100 %	
➔ java.io.PrintStream.println(String) PrintStrear	78	84 %	
➔ java.util.TreeMap.get(Object) TreeMap.java	15	16 %	
6			
➔ Menu.main(String[]) Menu.java	93	100 %	
➔ java.io.PrintStream.println(String) PrintStrear	78	84 %	
➔ java.util.TreeMap\$KeyIterator.next() TreeMa	15	16 %	

➔ Menu.main(String[]) Menu.java	78	100 %
➔ java.io.PrintStream.println(String) PrintStrear	62	79 %
D. ➔ java.util.HashMap\$KeyIterator.next() HashM	15	19 %

Para mostrar todas las cartas para la implementación HashMap, lo que toma tiempo es el for que eso tomo 15 ms, y luego el print de las cartas dentro del for que es el que tomó más tiempo siendo 62ms.

```
for (String key : cartas.keySet())
{
    System.out.println("Nombre: " + key + " | Tipo: " + cartas.get(key));
}
for (String key : coleccionUsuario.keySet())
{
    System.out.println("Nombre: " + key + " | Tipo: " + coleccionUsuario.get(key));
}
```