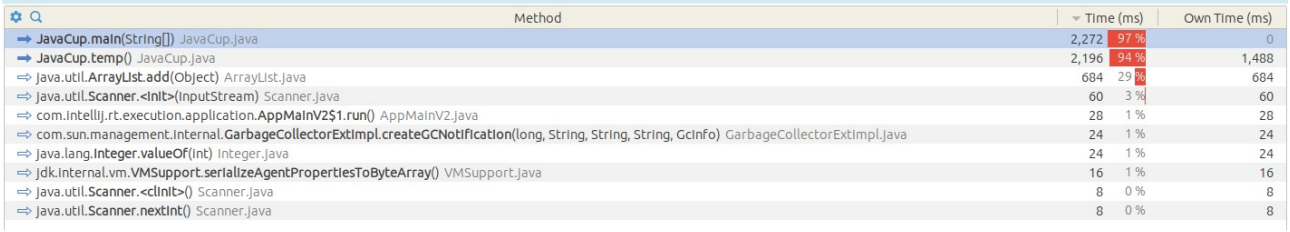
Profiling Homework

.این پروژه برای تمرین پروفایلینگ درس آزمایشگاه مهندسی نرم افزار است

تمرین ۱

متوجه میشویم که بیشترین تابعی که برای برنامه ما مشکل ایجاد میکند تابع yourkit پس از بررسی کد ها با ابزار .است javaCup واقع در کلاس temp



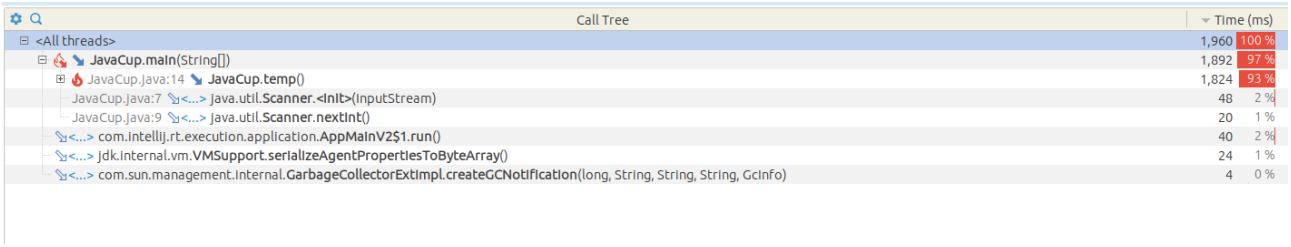
Method	Time (ms)	Own Time (ms)
JavaCup.main(String[]) JavaCup.java	2,272 97 %	0
JavaCup.temp() JavaCup.java	2,196 94 %	1,488
java.util.ArrayList.add(Object) ArrayList.java	684 29 %	684
java.util.Scanner.<init>(InputStream) Scanner.java	60 3 %	60
com.intellij.rt.execution.application.AppMainV2$1.run() AppMainV2.java	28 1 %	28
com.sun.management.internal.GarbageCollectorExtImpl.createGCNotification(long, String, String, String, GcInfo) GarbageCollectorExtImpl.java	24 1 %	24
java.lang.Integer.valueOf(int) Integer.java	24 1 %	24
jdk.internal.vm.VMSupport.serializeAgentPropertiesToByteArray() VMSupport.java	16 1 %	16
java.util.Scanner.<clinit>() Scanner.java	8 0 %	8
java.util.Scanner.nextInt() Scanner.java	8 0 %	8

حال با بررسی این تابع سعی در برطرف کردن مشکل ان داریم. در این تابع ما بدون مشخص کردن یک ظرفیت اولیه برای یک ارایه در مموری تلاش میکنیم مقدار زیادی درایه را در ان قرار دهیم. با مشخص کردن یک ظرفیت اولیه برای دیگر زمانی صرف اختصاص دوباره حافظه به ان نمیشود و کد سریعتر و بدون مشکل انجام میشود a ارایه

.با دانستن این موضوع کد را به صورت زیر تبدیل میکنیم

```
1 usage   raadled
public static void temp() {
    ArrayList<Integer> a = new ArrayList<>( initialCapacity: 10000 * 20000);
    for (int i = 0; i < 10000; i++)
    {
        for (int j = 0; j < 20000; j++) {
            a.add(i + j);
        }
    }
}
```

:و پس از اجرا داریم



Call Tree	Time (ms)
<All threads>	1,960 100 %
JavaCup.main(String[])	1,892 97 %
JavaCup.java:14 JavaCup.temp()	1,824 93 %
JavaCup.java:7 <...> java.util.Scanner.<init>(InputStream)	48 2 %
JavaCup.java:9 <...> java.util.Scanner.nextInt()	20 1 %
<...> com.intellij.rt.execution.application.AppMainV2$1.run()	40 2 %
<...> jdk.internal.vm.VMSupport.serializeAgentPropertiesToByteArray()	24 1 %
<...> com.sun.management.internal.GarbageCollectorExtImpl.createGCNotification(long, String, String, String, GcInfo)	4 0 %

تمرین ۲

برای سادگی ما یک برنامه مینویسیم که با شروع از کمترین عدد صحیح ممکن تا بیشترین عدد صحیح بررسی میکند که ایا عدد داده شده در این بازه قرار دارد یا نه.

```java
public class Calculator {
    public static void main(String[] args) {
        boolean mult = isInrange( n: Integer.MAX_VALUE - 10);

        System.out.println("Is value in range of ints: " + mult);
    }

    static boolean isInrange(int n) {
        for (int i = Integer.MIN_VALUE; i < Integer.MAX_VALUE; i++) {
            if (i == n)
                return true;
        }
        return false;
    }
}
```

نتیجه اجرای این پیاده سازی به صورت زیر است

Call Tree	Time (ms)		Samples
<All threads>	884	100 %	113
Calculator.main(String[])	828	94 %	42
Calculator.java:3 Calculator.isInrange(int)	828	94 %	42
Calculator.java:9	828	94 %	42
<...> java.lang.Thread.exit()	24	3 %	1
<...> com.intellij.rt.execution.application.AppMainV2$1.run()	16	2 %	46
<...> jdk.internal.vm.VMSupport.serializePropertiesToByteArray()	16	2 %	1
<...> java.lang.Thread.run()	0	0 %	22
<...> jdk.internal.vm.VMSupport.serializeAgentPropertiesToByteArray()	0	0 %	1

همانطور که مشاهده میشود این پیاده سازی بسیار نابهینه و اشتباه است.

حال ما یک پیاده سازی منطقی از این تابع را انجام میدهیم.

```java
raadted *
public class Calculator {
    raadted
    public static void main(String[] args) {
        boolean mult = isInrange( n: Integer.MAX_VALUE - 10);

        System.out.println("Is value in range of ints: " + mult);
    }

    1 usage  new *
    static boolean isInrange(double n) {
        return n >= Integer.MIN_VALUE && n <= Integer.MAX_VALUE;
    }
}
```

که نتایج این پیاده سازی به صورت زیر است.

Call Tree	Time (ms)		Samples
<All threads>	24	100 %	23
<...> com.intellij.rt.execution.application.AppMainV2$1.run()	16	67 %	21
Calculator.main(String[])	8	33 %	2
Calculator.java:5 <...> java.io.PrintStream.println(String)	8	33 %	1
Calculator.java:5 <...> java.lang.invoke.MethodHandleNatives.linkMethodHandleConstant(Class, int, Class, String, Object)	0	0 %	1

همانطور که مشاهده میشود در این پیاده سازی اصلا تابع ما زمانی نمیگیرد و قابل تشخیص نیست و زمان پیاده سازی برنامه هم به مقدار قابل توجهی کاهش یافته است.

ادرس ریپو این آزمایش
https://github.com/aliraad79/Software-engineering-profiling