بسمه تعالی

آزمایشگاه مهندسی نرم افزار

دانشگاه صنعتی شریف دانشکده مهندسی کامپیوتر

عنوان آزمایش: آشنایی با نحوه پروفایل برنامه (Profiling)

پرشان تیموری ۹۶۱۰۴۸۸۱

مهسا شیخی ۹۶۱۰۵۸۸۶

اهداف:

هدف این آزمایش آشنایی با نحوه پروفایل کردن برنامه یا profiling است. ریپوی پروژه در آدرس زیر قرار دارد:

https://github.com/mahsaShv/AZ_Narm_HW4.git

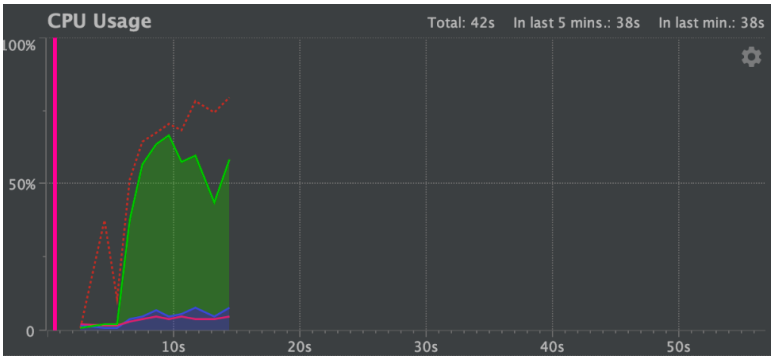
پیش زمینه:

فانکشن main() را اجرا میکنیم و با Yourkit وضعیت ریسورس های اختصاص داده شده برنامه و خروجی برنامه بصورت زیر مشاهده میشود:

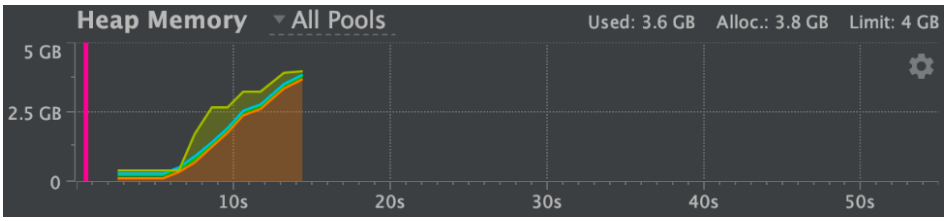
```
Press number1:
1
Press number2:
2
Press number3:
3
java.lang.OutOfMemoryError: Java heap space
Dumping heap to java_pid36781.hprof ...
Heap dump file created [5841435823 bytes in 75.568 secs]
Exception in thread "main" java.lang.OutOfMemoryError Create breakpoint : Java heap space
    at java.base/java.util.Arrays.copyOf(Arrays.java:3688)
    at java.base/java.util.ArrayList.grow(ArrayList.java:237)
    at java.base/java.util.ArrayList.grow(ArrayList.java:242)
    at java.base/java.util.ArrayList.add(ArrayList.java:467)
    at java.base/java.util.ArrayList.add(ArrayList.java:480)
    at JavaCup.temp(JavaCup.java:30)
    at JavaCup.main(JavaCup.java:14)

Process finished with exit code 1
```

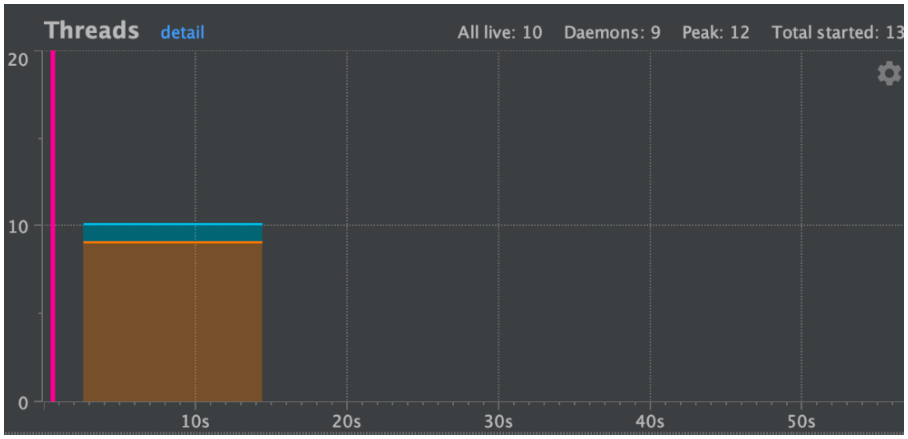
خروجی برنامه اولیه



وضعیت استفاده از CPU برنامه اولیه



وضعیت حافظه هیپ برنامه اولیه



وضعیت رشته های استفاده شده برنامه اولیه

همان طور که از خروجی برنامه مشخص است تابع temp() در main() بیشترین منابع بخصوص حافظه را مشغول کرده است.

زمان اجرا توابع برنامه اولیه

```java
public static void temp() {
    ArrayList a = new ArrayList();
    for (int i = 0; i < 10000; i++)
    {
        for (int j = 0; j < 20000; j++) {
            a.add(i + j);
        }
    }
}
```

کد تابع temp() اولیه

اجرای آزمایش:

پیاده سازی تابع temp() را عوض میکنیم و دوباره وضعیت منابع استفاده شده را بررسی میکنیم.

```java
public static void temp() {

    int counter = 0;
    int[] a = new int[200000000];

    for (int i = 0; i < 10000; i++)
        for (int j = 0; j < 20000; j++) {
            a[counter] = i + j;
            counter += 1;
        }

}
```

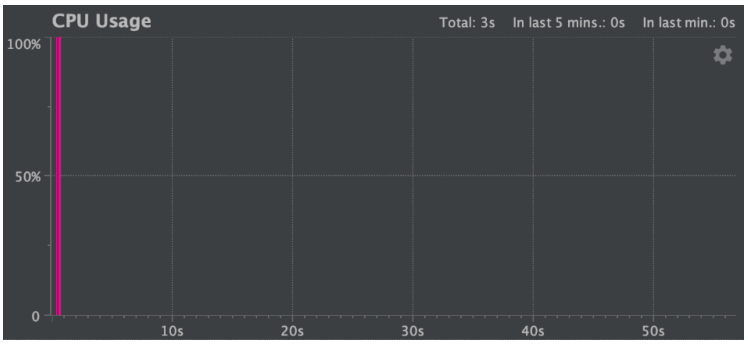
کد تابع temp() نهایی

دوباره برنامه را اجرا میکنیم.

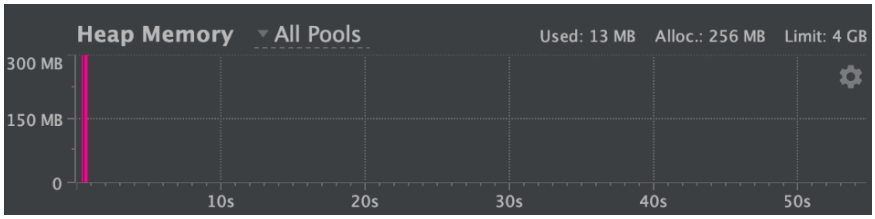
```
Press number1:
1
Press number2:
2
Press number3:
3
NO

Process finished with exit code 0
```

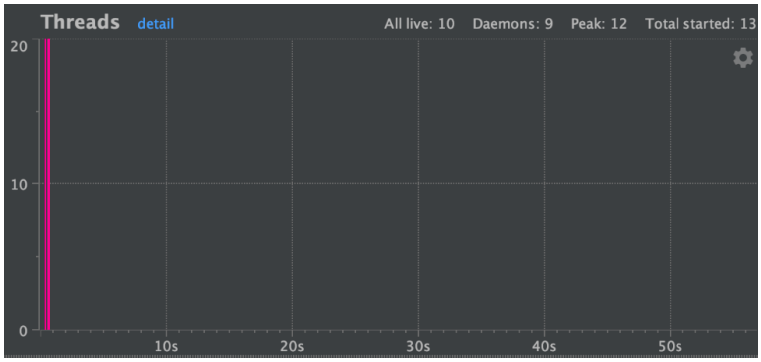
خروجی برنامه نهایی



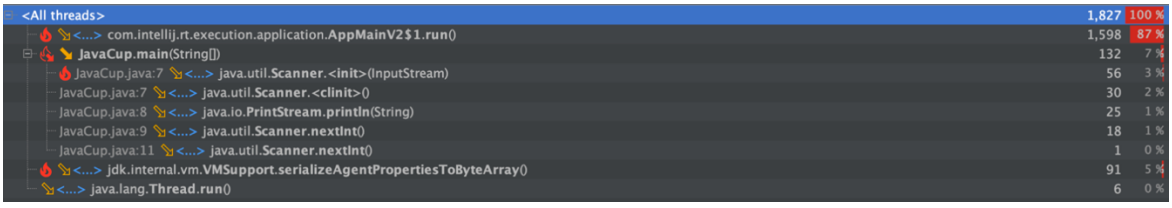
وضعیت استفاده از CPU برنامه نهایی



وضعیت حافظه هیپ برنامه نهایی



وضعیت رشته های استفاده شده برنامه نهایی



زمان اجرای توابع برنامه نهایی (که میبینیم کم شده است)

علت این درگیر شدن شدید حافظه و CPU استفاده از ()Arraylist بود که نوعی داده ساختار داینامیک است و با هربار پر شدن فضا دوبرابر فضا اختصاص میدهد که باعث استفاده بسیار زیادی از ریسورس ها میشود. اما در این جا چون میدانیم چه تعداد بلوک اختصاص بدهیم کافی است از اول تعداد را مشخص میکنیم تا این مشکل دیگر پیش نیاید.