آزمایشگاه نرم افزار

آزمایش اول

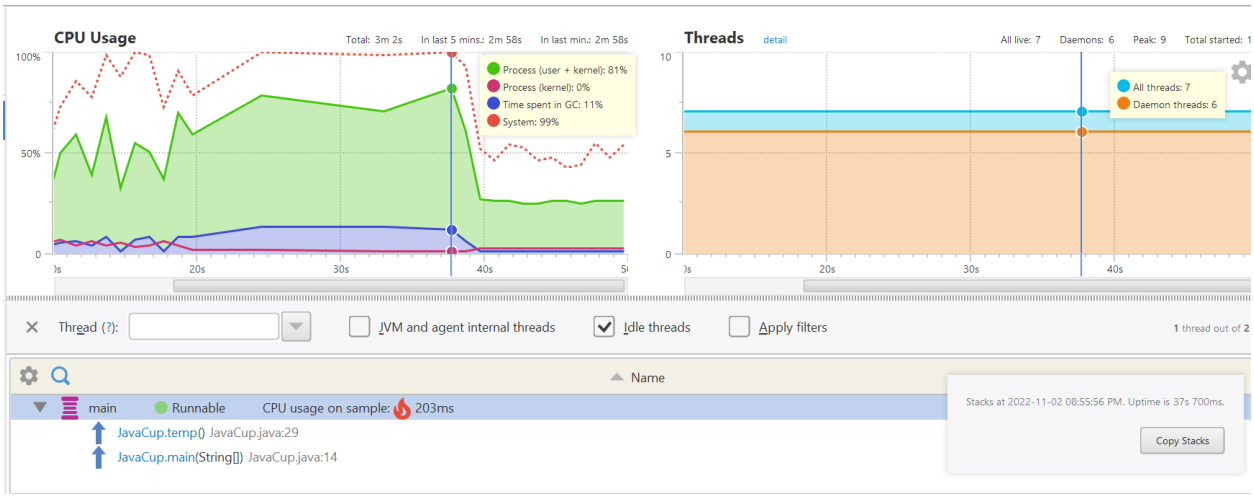
کیمیا نوربخش 97107256

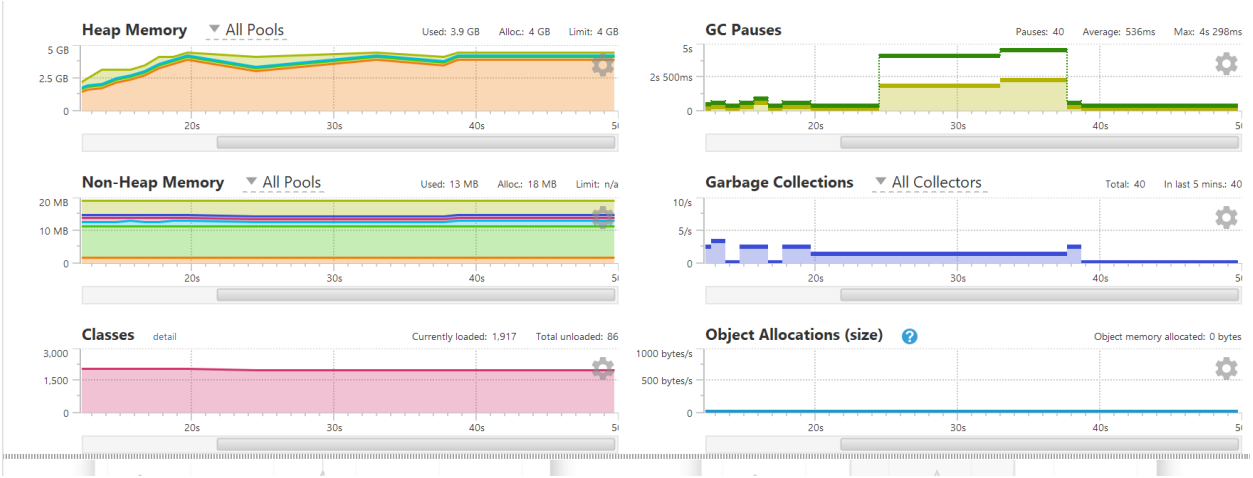
مبینا پورنعمت 97105833

Git: https://github.com/Mobinapournemat/Software_Eng_Lab

تمرین

سوال اول) در نرم افزار YourKit هم مشاهده میکنیم که بیشترین مصرف منابع مربوط به تابع temp بوده است که یک for با 20000 تکرار درون یک for با تکرار 10000 بار دارد و در هر تکرار یک مقدار را به یک ArraList اضافه میکند. این سربار بالایی برای cpu ایجاد میکند.
تصاویر مربوط به اجرای برنامه در intelij و YourKit را در زیر میبینیم.

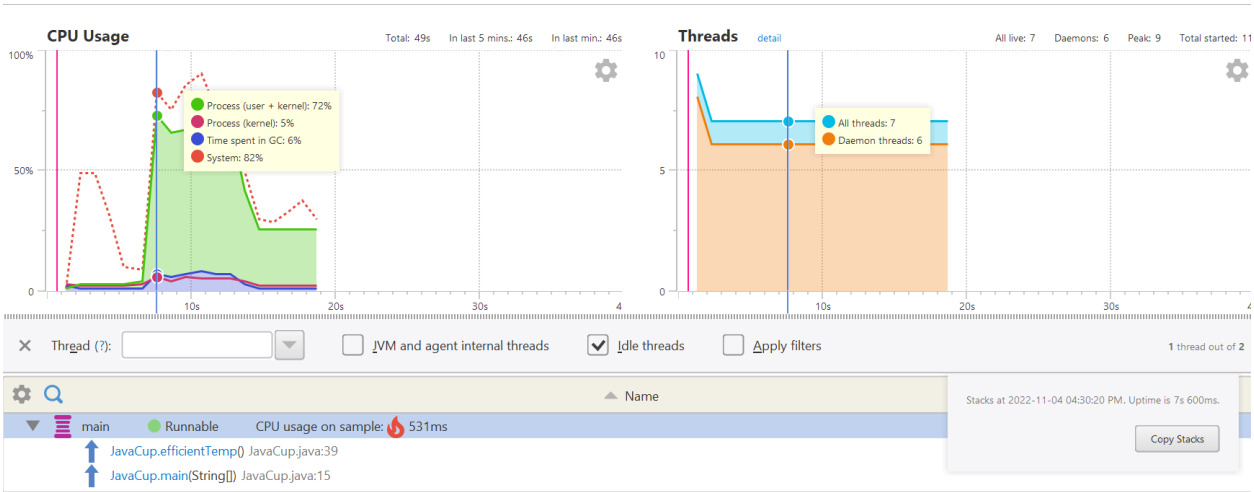




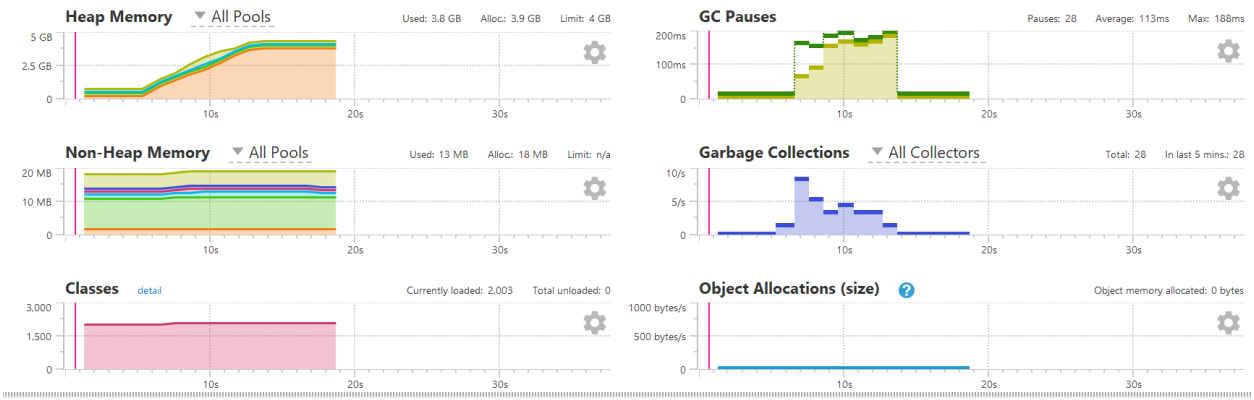
همان طور که در شکل بالا مشخص است، مصرف حافظه Heap در طول زمان افزایش می یابد (که این البته به علت بزرگی حجم آرایه نهایی است.) و در نهایت برنامه با پر شدن این حافظه خاتمه پیدا میکند. هم چنین از نمودار های دیگر می توان نتیجه گرفت که مصرف حافظه غیر Heap ای در طول زمان تغییری نمی کند.

راه اول برای بهبود:

حال برای بهبود عملکرد این تابع temp، تابع efficientTemp را پیاده سازی میکنیم که همان کار تابع temp را میکند، با این تفاوت که سایز آرایه a را از اول مشخص میکنیم. بدین ترتیب از کپی کردن های internal برای تایین سایز آرایه جلوگیری میکند و همانطور که در زیر میبینیم بیشترین میزان CPU usage از 81 درصد به 71 درصد کاهش یافته است.



البته مصرف مموری همچنان مشابه قبل است، چون در نهایت هدف تابع پر کردن چنین لیست بزرگی است.



راه دوم برای بهبود:

این راه صرفا عملکرد خاص تابع temp را بهتر میکند و هم مصرف CPU و هم مصرف heap را بسیار بهبود میبخشد. کلاس EffivientList را به صورت زیر تعریف میکنیم:

```
public class EfficientList extends ArrayList<Integer> {
   private final int outerForInt;
   private final int innerForInt;

   public EfficientList(int innerForInt, int outerForInt) {
      this.innerForInt = innerForInt;
      this.outerForInt = outerForInt;
   }

   @Override
   public Integer get(int index) {
      return (index / innerForInt) + (index % innerForInt);
   }

   @Override
   public int size() {
      return innerForInt * outerForInt;
   }
}
```

حال تابع efficientTempTwo را در کلاس JavaCup به صورت زیر مینویسیم:

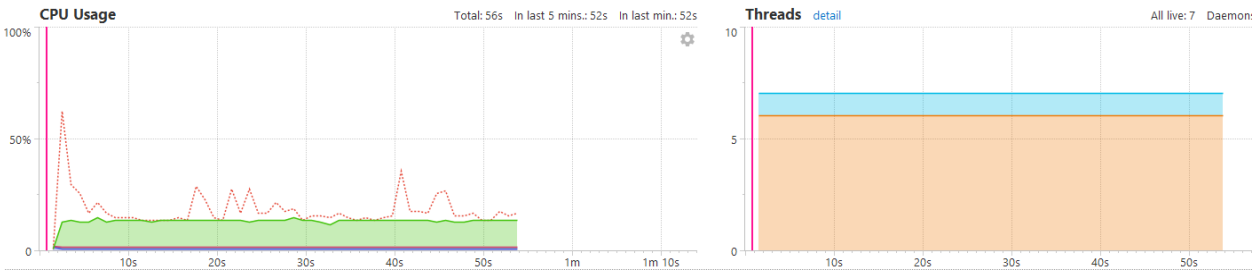
```
public static void efficientTempTwo() {
  ArrayList a = new EfficientList(20000, 10000);
}
```

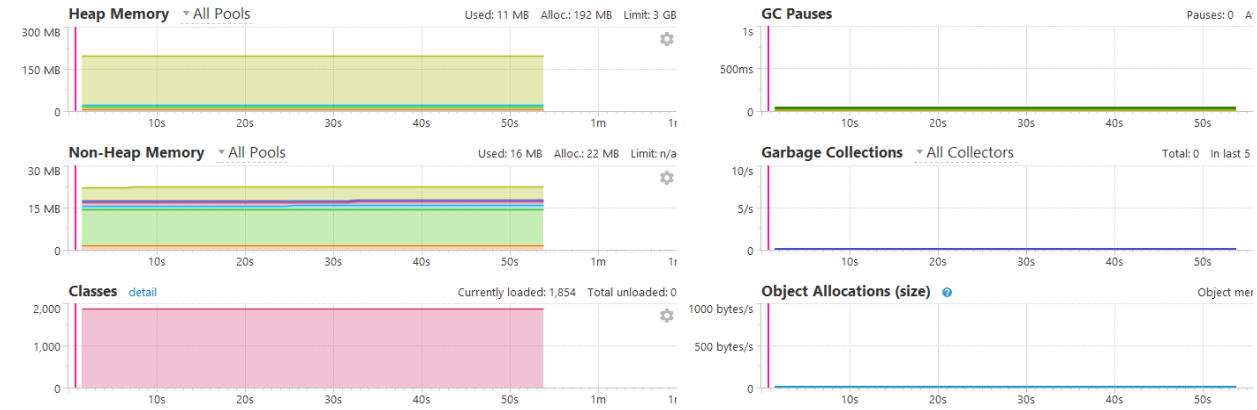
که همان مقادیر مورد نظر تابع temp را در arraylist به نام a میریزد. حال برنامه را با yourkit اجرا میکنیم:

و به این صورت مشکل Heap نیز برطرف شده است.

سوال دوم)

در ابتدا از الگوریتم Selection sort استفاده می کنیم که پیچیدگی زمانی آن O(N2) است. ابتدا آرایه ای با 200000 عضو رندوم پر می کنیم و سپس یه الگوریتم مرتب سازی می دهیم و در همین حین، چگونگی استفاده از منابع را به وسیله YourKit بررسی می کنیم.





سپس الگوریتم مرتب سازی را به merge sort تغییر میدهیم که پیچیدگی زمانی آن O(Nlogn) است و مشاهده می کنیم که همان آرایه از اعداد با سرعت خیلی بیشتر و کمترین استفاده از منابع مرتب می شوند.