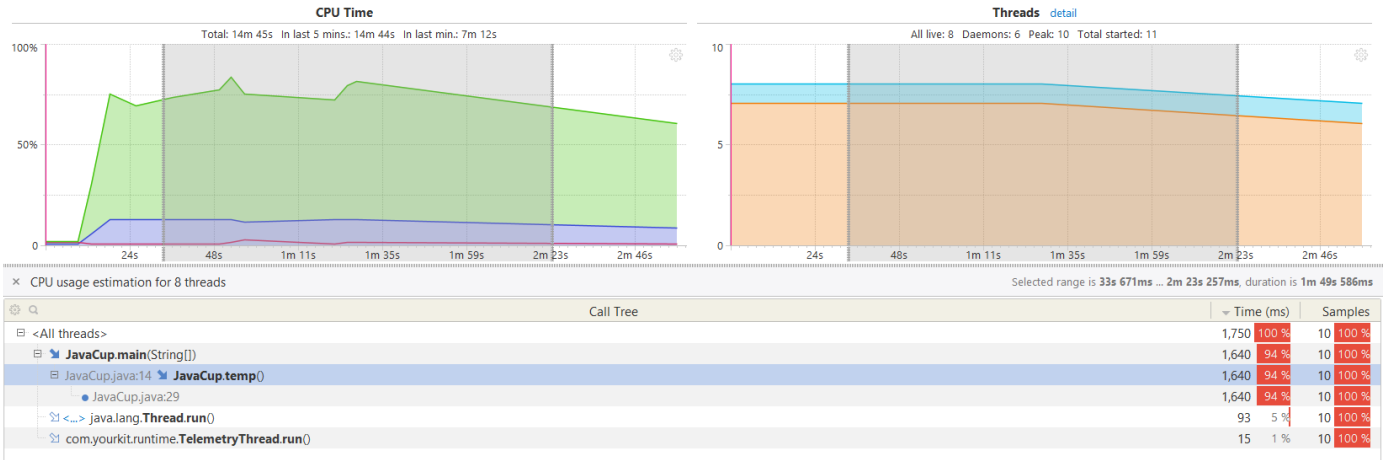
آشنایی با نحوه پروفایل برنامه

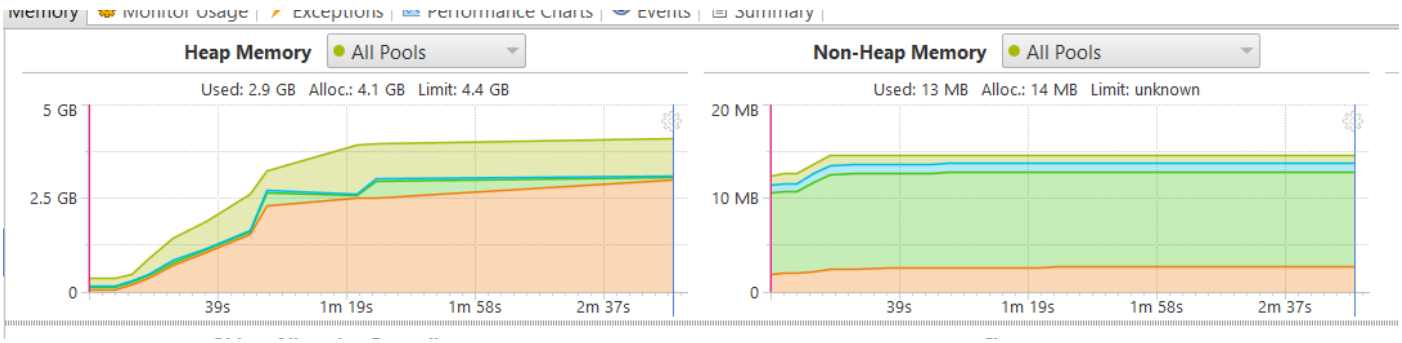
عرفان فراوانی 97102174

تمرین ۱

از پروژه ProfilingTest ، عملیات Profiling را با استفاده از Yourkit بر روی کلاس JavaCup اجرا نموده و تابعی از پروژه که بیشترین مصرف منابع را دارد شناسایی کنید و آن را در گزارش توضیح دهید . سپس نحوه پیاده سازی آن تابع را به گونه ای تغییر دهید که مصرف منابع نسبت به قبل بهتر شود.

اجرای برنامه JavaCup:





بعد از اجرای برنامه ی اولیه با ارور

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

مواجه می‌شویم و پس از بررسی پروفایلر مصرف cpu و heap memory از تابع temp می باشد.
این تابع شامل دو for تودرتو است که اولی ۱۰۰۰۰ و دومی ۲۰۰۰۰ مرحله دارد و در یک ارایه جمع را ذخیره می کنند

خروجی تابع در نهایت مانند زیر است:

0 1 ... 19999 1 2 ... 20000 29998

برای بهینه کردن تابع ابتدا ظرفیت Arraylist اولیه را به 20000*10000 تغییر می دهیم زیرا نحوه کار Arraylist هنگام اضافه کردن به این شکل است که در صورت نبود ظرفیت لیستی جدید ساخته و به آن اضافه میکند.

در مرحله بعد از ارایه دیگری به نام temp استفاده میکنیم برای این که for تودرتو نداشته باشیم و نحوه کار آن به این شکل است که ظرفیت را ۲۰۰۰۰ می گذاریم و سپس یک بار از ۰ تا ۱۹۹۹۹ آن را مقدار دهی می کنیم

در for بعدی که ۱۰۰۰۰ تایی است هر بار با استفاده از دستور addall همه آرایه temp را به a اضافه میکنیم و برای مرحله بعدی خانه اول temp را پاک کرده و به آخر آن یکی اضافه میکنیم.

اجرای برنامه JavaCup پس از بهینه سازی:

تمرین ۲

قطعه کدی به زبان جاوا جهت مرتب سازی مجموعه ای از اعداد بنویسید و سپس عملیات Profiling را با استفاده از Yourkit بر روی آن اجرا و خروجیها را به صورت تصویری گزارش نمایید. الگوریتم مرتب سازی را بهینه کرده و مجدداً عملیات Profiling را بر روی کد جدید اِعمال کنید و خروجیها را در گزارش نمایش دهید. در واقع گزارش باید حاوی تصاویری از وضعیت مصرف کلیه منابع در هر دو پیاده سازی باشد.

در این آزمایش از bubble sort استفاده شده است.

اجرای برنامه Bubble sort قبل از بهینه سازی:

Start time:	November 4, 2022 08:31:04 PM
Uptime:	1m 4s
CPU time:	1m 4s
Command line:	C:\Users\erfan\.jdks\corretto-1.8.0_352\bin\java.exe -agentpath:C:\ProgramData\YourKit\2019.1.117.822574B9\64\yjpagent.dll=sampling,_ide_project
VM arguments:	-agentpath:C:\ProgramData\YourKit\2019.1.117.822574B9\64\yjpagent.dll=sampling,_ide_project_name=ProfilingTest,sessionname=BubbleSort,profil
Class path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\charsets.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\ext\access-bridge-64.jar;C:\Users\erfan\.jdks\c
Boot class path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\resources.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\rt.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\
Library path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\system32;C:\Windows;C:\Windov
System properties:	...
Agent version:	YourKit Java Profiler 2019.1-b117
Agent mode:	Loaded on start

Heap Memory

Used:	24 MB
Allocated:	305 MB
Limit:	4.4 GB

Non-Heap Memory

Used:	10 MB
Allocated:	11 MB
Limit:	unknown

Classes

Currently loaded:	1,198
Total unloaded:	0

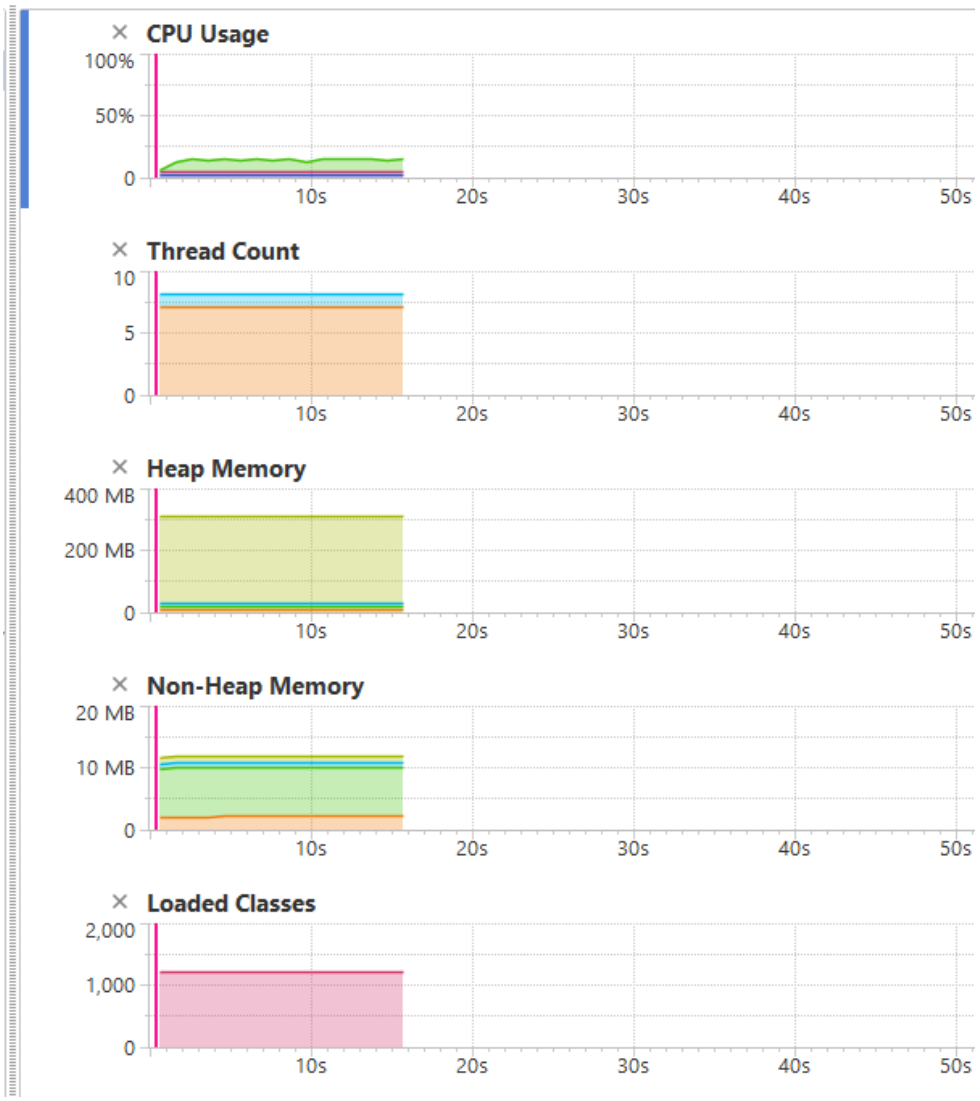
Threads

Currently live:	8
Currently live daemons:	7
Peak:	10
Total started:	11

اجرای برنامه Bubble sort بعد از بهینه سازی:

در برنامه از کلاس Integer استفاده شده برای ذخیره مجموعه اعداد ولی به جای کلاس می توان از int استفاده کرد که primitive است و سرعت بیشتری دارد .

در منطق الگوریتم زمان اجرا همیشه از o(N^2) است حتی اگر ارایه مرتب باشد برای همین برای حلقه داخلی شرط می گذاریم که اگر فقط اگر نیاز بود swap را انجام دهد.



Uptime:	16s
CPU time:	15s
Command line:	C:\Users\erfan\.jdks\corretto-1.8.0_352\bin\java.exe -agentpath:C:\ProgramData\YourKit\2019.1.117.822574B9\64\yjpagent.dll=sampling,_ide_projec
VM arguments:	-agentpath:C:\ProgramData\YourKit\2019.1.117.822574B9\64\yjpagent.dll=sampling,_ide_project_name=ProfilingTest,sessionname=OptimizedBubb
Class path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\charsets.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\ext\access-bridge-64.jar;C:\Users\erfan\.jdks\
Boot class path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\resources.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\rt.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352
Library path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\system32;C:\Windows;C:\Windo
System properties:	...
Agent version:	YourKit Java Profiler 2019.1-b117
Agent mode:	Loaded on start

Heap Memory

Used:	21 MB
Allocated:	305 MB
Limit:	4.4 GB

Non-Heap Memory

Used:	10 MB
Allocated:	11 MB
Limit:	unknown

Classes

Currently loaded:	1,193
Total unloaded:	0

Threads

Currently live:	8
Currently live daemons:	7
Peak:	10
Total started:	11

Automatic Deobfuscator

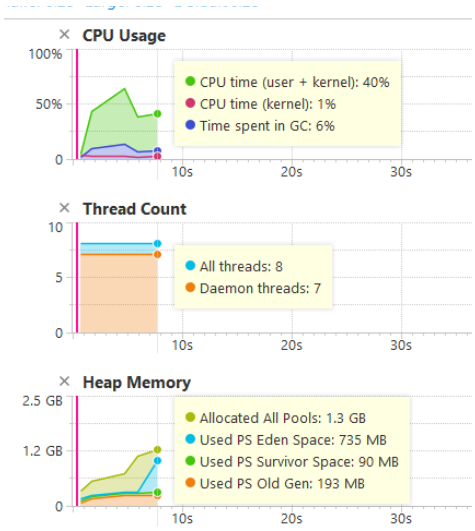
مقایسه:

با توجه به اینکه منطق برنامه در هر دو تقریبا یکی است میزان مموری مصرف شده در هر دو تقریبا برابر است ولی با توجه به استفاده از int میزان ۳ heap memory مگابایت کاهش داشته است.

با توجه به یکسان بودن thread ها مصرف cpu نیز یکی است اما به دلیل چک کردن تمام حالت ها در کد اولی زمان بسیار بیشتری برای برنامه مصرف شده است.

اجرای برنامه MergeSort :

در اینجا به جای بهینه سازی الگوریتم را تغییر دادیم و تعداد داده را نیز ۱۰۰ برابر کردیم و به نتایج زیر رسیدیم. اردر از nlogn است و چون بازگشتی است استفاده از cpu بیشتر شده است و حجم بیشتری را نیز اشغال میکند اما سرعت بسیار بالا میرود.



Start time:	November 4, 2022 09:48:52 PM
Uptime:	7s
CPU time:	29s
Command line:	C:\Users\erfan\.jdks\corretto-1.8.0_352\bin\java.exe -agentpath:C:\ProgramData\YourKit\2019.1.117.822574B9\64\yjpagent.dll=sampling,_ide_prc
VM arguments:	-agentpath:C:\ProgramData\YourKit\2019.1.117.822574B9\64\yjpagent.dll=sampling,_ide_project_name=ProfilingTest,sessionname=MergeSort,p
Class path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\charsets.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\ext\access-bridge-64.jar;C:\Users\erfan\.j
Boot class path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\resources.jar;C:\Users\erfan\.jdks\corretto-1.8.0_352\jre\lib\rt.jar;C:\Users\erfan\.jdks\corretto-1.8.0_
Library path:	C:\Users\erfan\.jdks\corretto-1.8.0_352\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\system32;C:\Windows;C:\Wi
System properties:	...
Agent version:	YourKit Java Profiler 2019.1-b117
Agent mode:	Loaded on start

Heap Memory

Used:	1019 MB
Allocated:	1.3 GB
Limit:	4.4 GB

Non-Heap Memory

Used:	12 MB
Allocated:	13 MB
Limit:	unknown

Classes

Currently loaded:	1,396
Total unloaded:	0

Threads

Currently live:	8
Currently live daemons:	7
Peak:	10
Total started:	11