Documentatie WebServer

Descriere generala

Aplicatia reprezinta un webserver ce permite clientilor sa vizualizeze intr-un browser paginile html, precum si css-urile aferente acestora. WebServer-ul se poate afla in una din cele 3 stari: stopped, running, maintenance. In functie de starea in care se afla, acesta va raspunde clientului cu pagini html specifice. WebServer-ul este controlat de catre administrator prin intermediul unei interfete grafice.

Functionare

WebServer

Atunci cand webserver-ul este in starea stopped, acesta va raspunde cu pagina stop.html indiferent de request-ul clientului. Prin intermendiul acestei pagini clientul este anuntat ca webserver-ul este oprit.

In starea running, webserver-ul va raspunde cu pagina ceruta de catre client (din web root directory), daca este gasita. Daca acesta lipseste, va raspunde cu pagina 404.html prin care acesta este anuntat ca pagina nu a fost gasita. Daca clientul nu a specificat nicio cale, este trimis la pagina index.html (pagina default).

In starea maintenance, webserver-ul va raspunde cu pagina index.html si in functie de datele introduse, il va trimite la pagina pentru maintenance (din folder-ul specific pentu maintenance) a admin-ului ( daca numele introdus este admin) sau a clientului (orice alt nume).

GUI

Prin intermediul interfetei grafice, webserver-ul poate sa fie configurat. Interfata grafica afiseaza informatii despre statusul serverului, adresa si portul.

Atunci cand webserver-ul este in starea stopped, webserver-ului i se pot configura: port-ul ,web root directory si maintenance directory. Acesta poate trece in stare running prin intermediul butonului Start server.

In starea running nu se mai pot configura port-ul sau web root directory, doar maintenance directory. Din aceasta stare, webserver-ul poate trece in stare stopped( prin butonul Stop server) sau in starea maintenance (prin bifarea checkbox-ului Switch to maintenance mode).

In starea maintenance se poate configura doar web root directory. Din maintenance putem trece server-ul in starea running (prin debifarea checkbox-ului) sau il putem opri (Stop server).

Detalii de implementare

WebServer

Clasa webserver are un ServerSocket prin intermediul caruia sa face comunicarea cu clientul. Metoda setPort seteaza port-ul de care va asculta server-ul si apeleaza metoda startServer. Aceasta accepta conexiunile clientilor, pentru fiecare creeaza un nou thread. Comunicarea cu clientul se face cu ajutorul clasei WebClientHandler. Clasa WebServer mai are metode pentru a trece server-ul dintr-o stare in alta, pentru verificarea starii in care se afla si pentru a schimba folderele.

Metoda run din clasa WebClientHandler este folosita pentru a raspunde client-ului cu o pagina web, in functie de starea in care se afla server-ul si de pagina ceruta de client.

GUI

Clasa GUI creeaza interfata grafica pentru configurarea webserver-ului. Elementelor din interfata grafica le sunt adaugate ActionListener pentru a raspunde comenzilor efectuate din interfata: start server, stop server, set port ..etc. Metoda change este folosita pentru a modifica elemente din interfata grafica atunci cand se face trecerea de la o stare la alta.

Testare

Testare WebServer(100 %methods, 93% lines covered)

Am creat 34 de teste Junit ce verifica crearea unei instante a clasei Webserver, port-ul, adresa,starile webserver-ului, raspunsul acestuia la request-urile clientului in functie de starea in care se afla,

inainte si dupa schimbarea portului, web root directory, maintenance directory.

Am creat 3 teste folosind Mockito pentru a verifica conexiunea, pentru a verifica daca webserver-ul primeste request si daca trimite raspuns.

Testare GUI

Am creat 3 teste pentru interfata grafica folosind Marathon:

Test 1: TestStates - Scenariu:

* am pornit serverul (Start server)
* am trecut in starea de maintenance
* am trecut inapoi in starea running
* am oprit server-ul

Test 2: TestPortAndRoot - Scenariu:

* am configurat webserver-ul cu un port gresit
* am configurat webserver-ul cu o valoare negativa
* am modificat web root directory
* am pornit webserver-ul
* am oprit server-ul

Test 3: TestMaintenance – Scenariu:

* am modificat maintenance directory
* am pornit webserver-ul
* am modificat iar maintenance directory
* am trecut webserver-ul in maintenance
* am trecut server-ul in running
* am oprit server-ul

Testare web

Am creat 4 teste folosind Selenium:

Test 1:RunningStateTest –Scenariu:

* webserver-ul este in starea running. Se acceseaza <http://localhost:8080/>, se introduce numele utilizatorului si se apasa butonul de submit si se deschide o noua pagina.

Test 2:FileNotFoundTest-Scenariu:

* webserver-ul este in starea running. Se acceseaza o pagina ce nu exista si se raspunde cu pagina 404.html
* ne intoarcem la pagina home

Test 4:MaintenanceTest-Scenariu:

* webserver-ul este in starea maintenance. Se acceseaza <http://localhost:8080/> si ne duce la pagina default
* se introduce nume de utilizator si dupa submit se deschide pagina MaintenanceUser
* ne intoarcem la pagina default
* se introduce numele “admin” si dupa submit se deschide pagina MaintenanceAdmin
* ne intoarcem la pagina default

Test 4:StopStateTest – Scenariu:

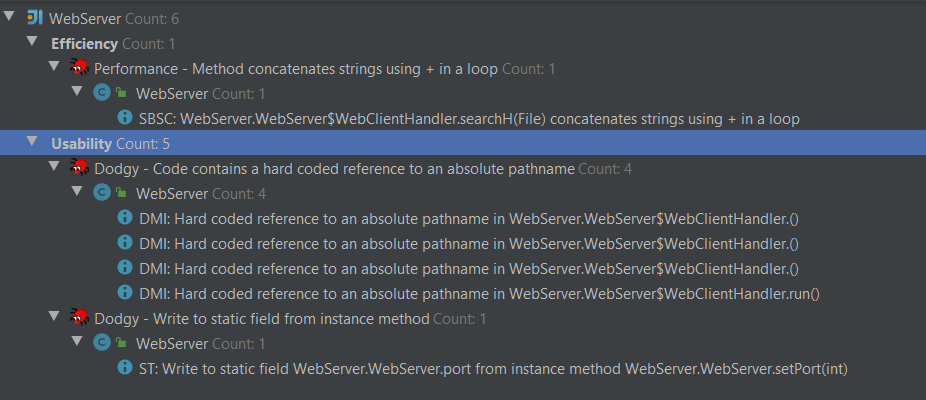
* webserver-ul este in stare stopped si se face un request. Acesta raspunde cu pagina stop.html. Se verifica titlul si mesajul afisat pe pagina.

Analiza

Analiza Statica

Am folosit:

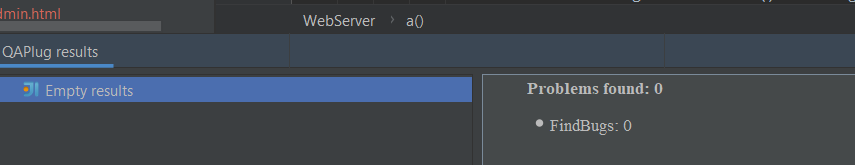
* FindBugs - este un tool open source folosit pentru a efectua analize statice pe cod Java.



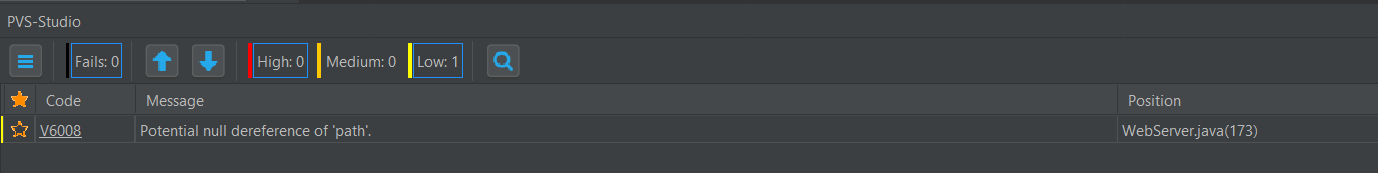
Rezolvari:

1. Am inlocuit + cu metoda concat()
2. Am inlocuit caile absolute cu caile relative
3. Am extras codul ce genera bug-ul intr-o metoda statica

Rezultat:

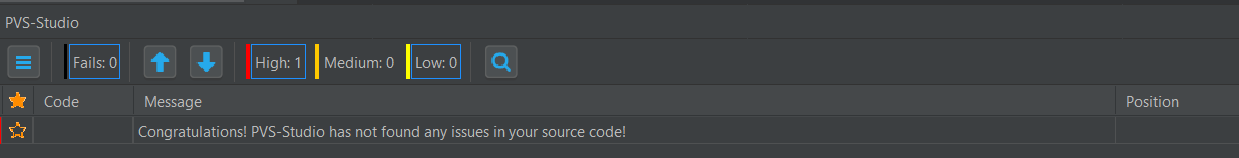


* PVS-Studio-este un tool pentru detectare bug-urilor in codul sursa al programelor.



Rezolvare: Am inlocuit initializarea cu null a lui path, cu sirul gol(“”);

Rezultat:

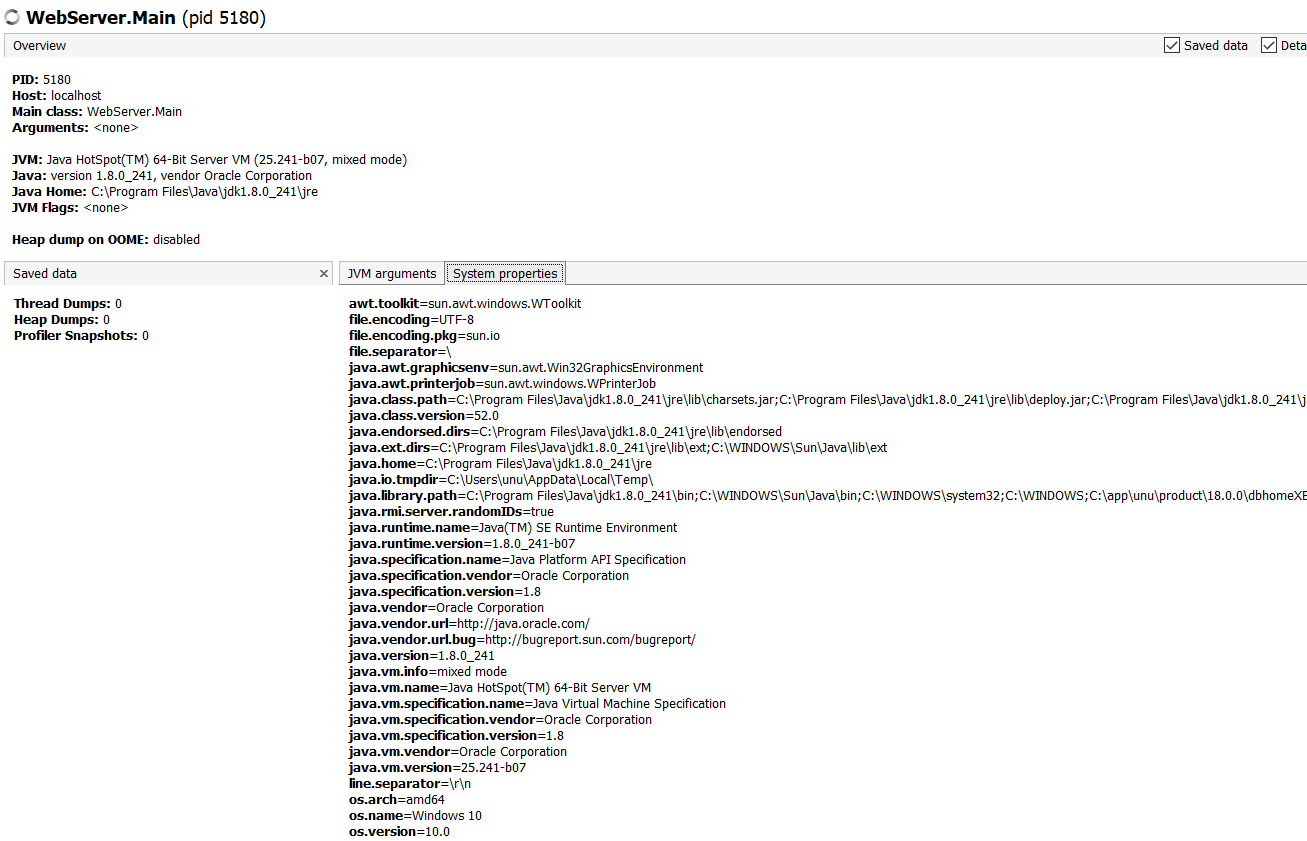


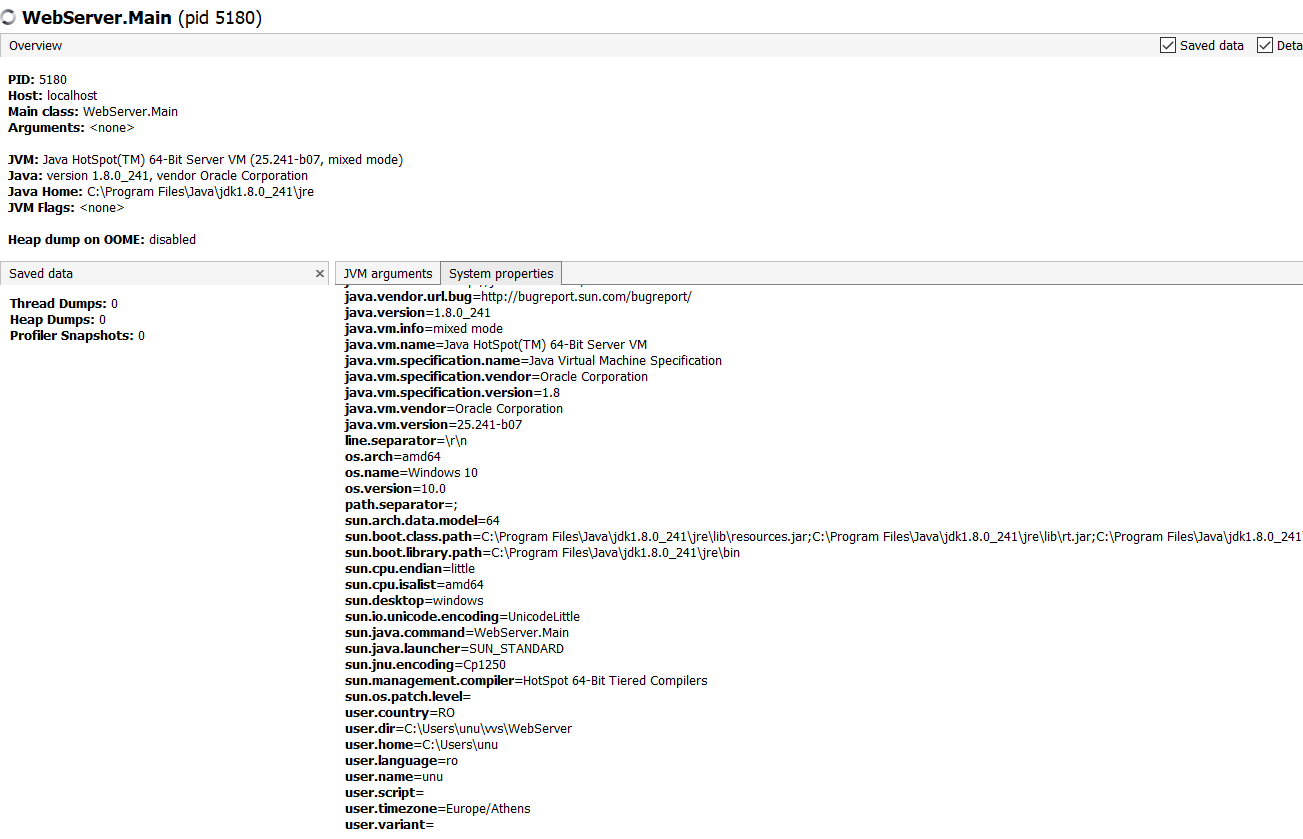
**Analiza dinamica**

Am folosit VisualVm. Acesta este un instrument ce ofera o interfata vizuala destinata vizualizarii informatiilor despre aplicatiile Java in timp ce ruleaza pe JVM.

Configuratie la runtime

Proprietatile sistemului, cum ar fi: java version, numele sistemului de operare, versiunea; tara utilizatorului, calea directorului, timezone, etc..​





Monitorizeaza consumul de memorie a aplicației la runtime:

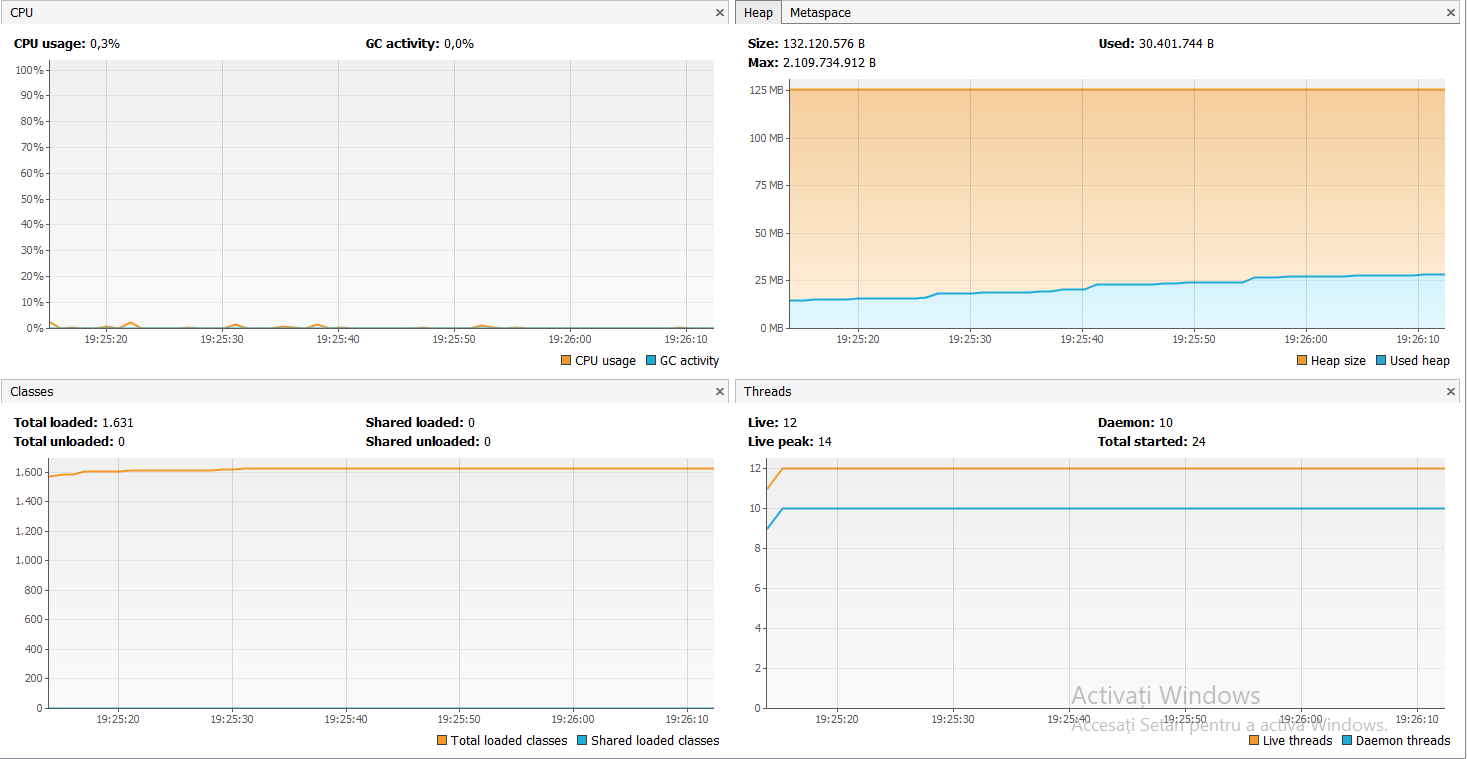
Cuprinde informatii despre:

• utilizarea CPU: 0.3%, cata memorie se consuma.

• Memoria creste atunci cand clientul face un request.

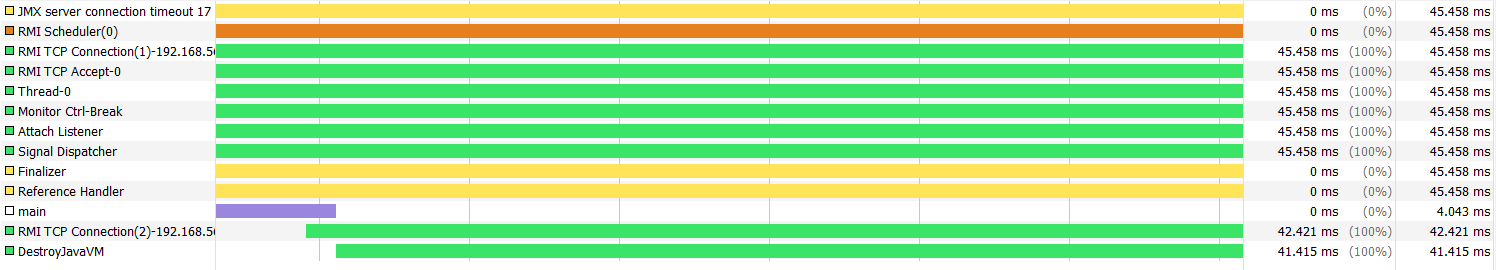
• Ruleaza 10 daemon thread-uri.

• Se folosesc 1631 clase.



Monitorizarea threadurilor

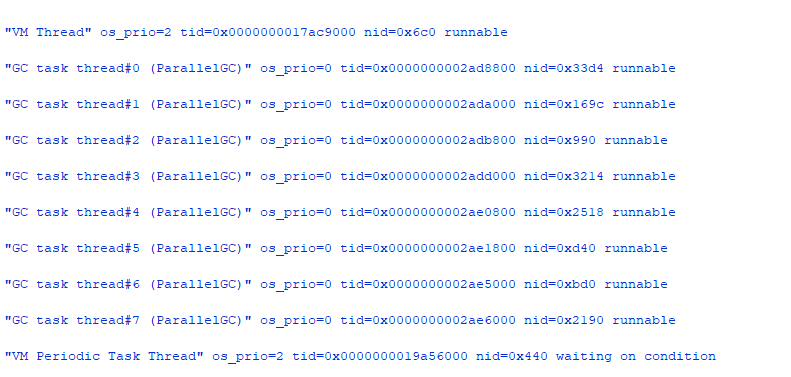
* Thread-urile cu verde sunt in starea running, cele cu galben in wait, iar cu mov sleeping (main – din cauza apelului din main la Thread.sleep(..))



Obținerea și afișarea thread dumbs​

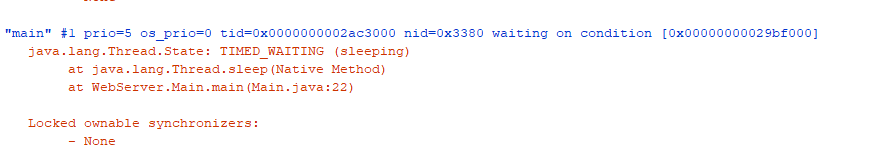
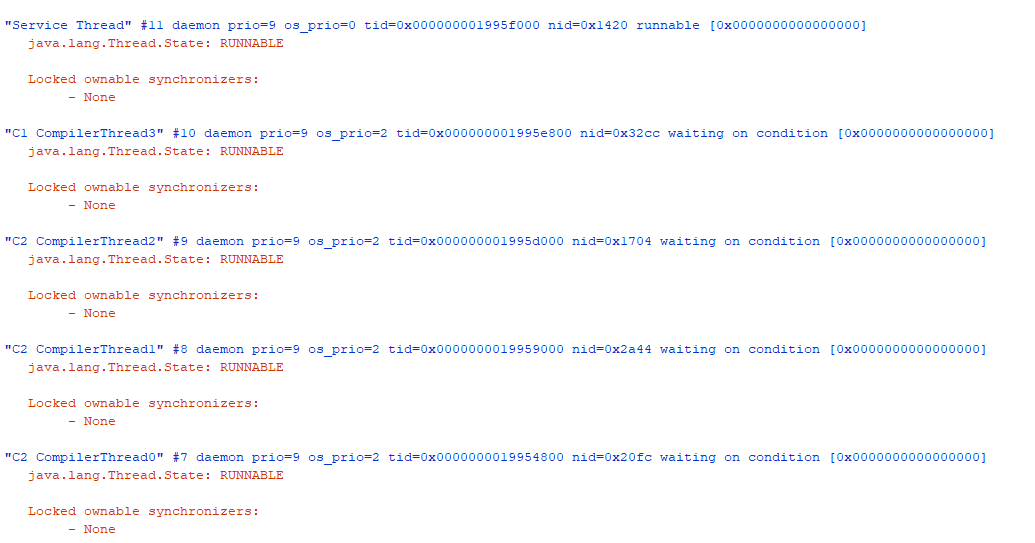
Acestea sunt fire interne ale JVM, ce sunt legate de sistemul de operare. Ele nu au un id unic si au statusul “runnable”.

Aceste fire sunt de obicei compuse din fire GC si alte fire utilizate de JVM pentru a rula si intretine o aplicatie Java.

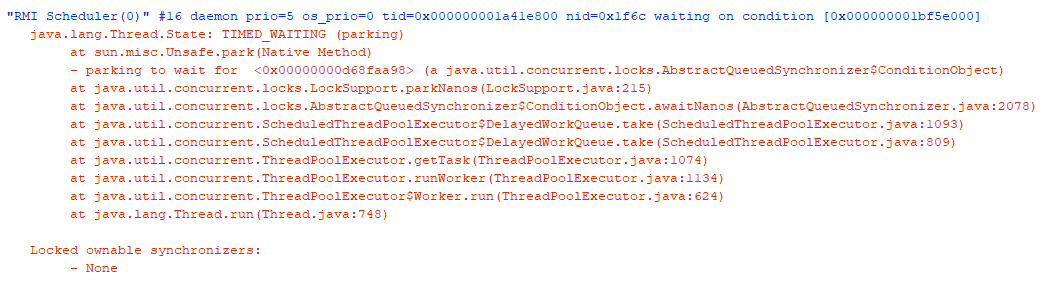


Numele thread-ului este “main”, #1 reprezinta id-ul unic al thread-ului, prio este 5, iar statusul “waiting on condition”.

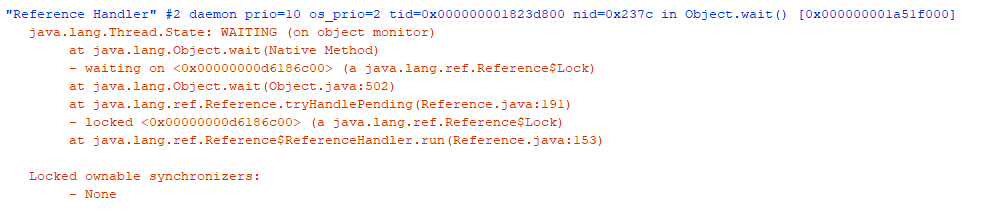
Thread-ul se afla in starea TIMED\_WAITING din cauza apelului Thread.sleep(…) din main.

Sunt daemon thread-uri si se afla in starea RUNNABLE. Unele au statusul “runnable”, iar altele “waiting for a condition” 

Thread-ul are id-ul 16, este un daemon thread cu prioritatea 5, are statusul waiting on condition. Thread-ul are starea TIMED\_WAITING din cauza metodei park (asteapta dupa un obiect cu adresa 0x00000000d68faa98 si tipul java.util.concurrent.locks.AbstractQueuedSynchronizer).



Thread-ul are numele “Reference Handler”, id-ul 2, prioritatea 10. Este un daemon thread si are statusul “in Object.wait()”. Se afla in starea WAITING din cauza metodei wait.



La WAITING firul asteapta folosind metodele wait, join, park, iar la TIMED\_WAITING folosind sleep, wait,join sau park. Diferenta dintre TIMED\_WAITING si WAITING este ca TIMED\_WAITING include un timeout.

Thread-ul RMI TCP Connection(1) se afla in starea RUNNABLE, are id-ul 15, prioriatatea 5 si statusul:runnable. Acesta asteapta incontinuu sa citeasca socket-ul.



Obținerea și afișarea heap dumbs

* Arata cate instante s-au folosit pentru fiecare clasa:
* Pentru char[] s-au folosit 7548
* Pentru String 7499

