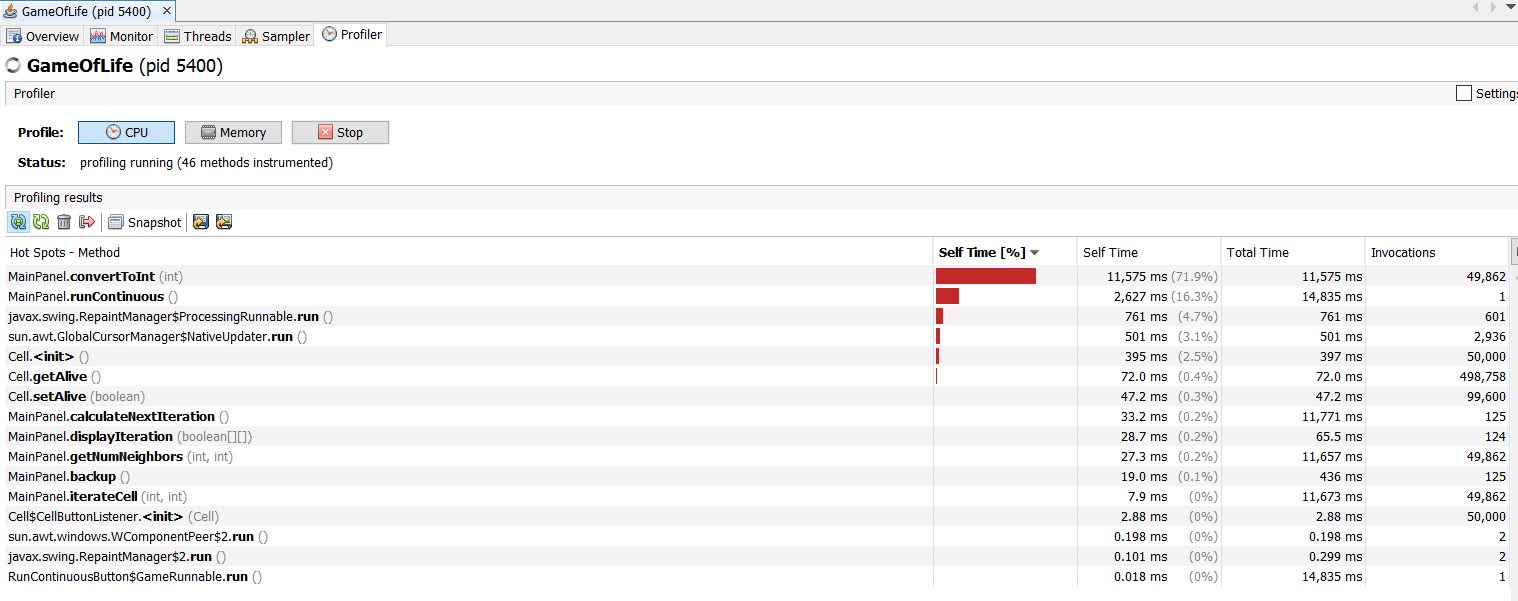
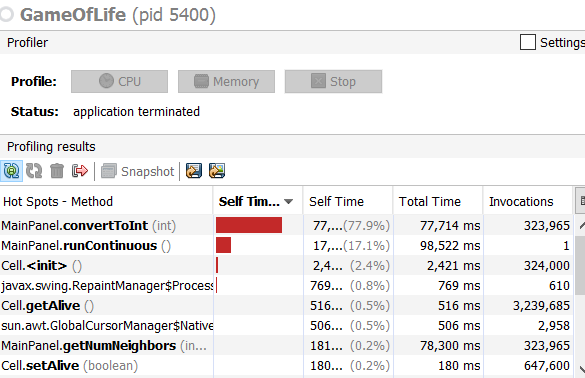
In this program ,we use the profiler-cpu to judge the performance of the original GameOfLife program.



We can see that the convertToInt method takes most of the time. So we can improve the code in this method to improve the performance of the program.



private int convertToInt**(**int x**)** **{**

int c **=** 0**;**

String padding **=** "0"**;**

**while** **(**c **<** \_r**)** **{**

String l **=** **new** String**(**"0"**);**

padding **+=** l**;**

c**++;**

**}**

String n **=** padding **+** String**.**valueOf**(**x**);**

int q **=** Integer**.**parseInt**(**n**);**

**return** q**;**

**}**

We may not need this method at all. This is just change the given integer x to a same integer and return. The while loop (1000 times) just add 1000 “0” before the string. And it changes the integer x to string and add those 1000 “0” before it. But “0”s before valid numbers are not calculated when “ParsetoInt”. The method is useless.

private int convertToInt**(**int x**)** **{**

**return** x**;**

**}**

This is the performance after we revise the convertToInt method:



This time,we can see that the runContinious method takes most of the time.



runContinuous method:

public void runContinuous**()** **{**

\_running **=** **true;**

**while** **(**\_running**)** **{**

System**.**out**.**println**(**"Running..."**);**

int origR **=** \_r**;**//1000

**try** **{**

Thread**.**sleep**(**20**);**

**}** **catch** **(**InterruptedException iex**)** **{** **}**

**for** **(**int j**=**0**;** j **<** \_maxCount**;** j**++)** **{**

\_r **+=** **(**j **%** \_size**)** **%** \_maxCount**;**

\_r **+=** \_maxCount**;**

**}**

\_r **=** origR**;**

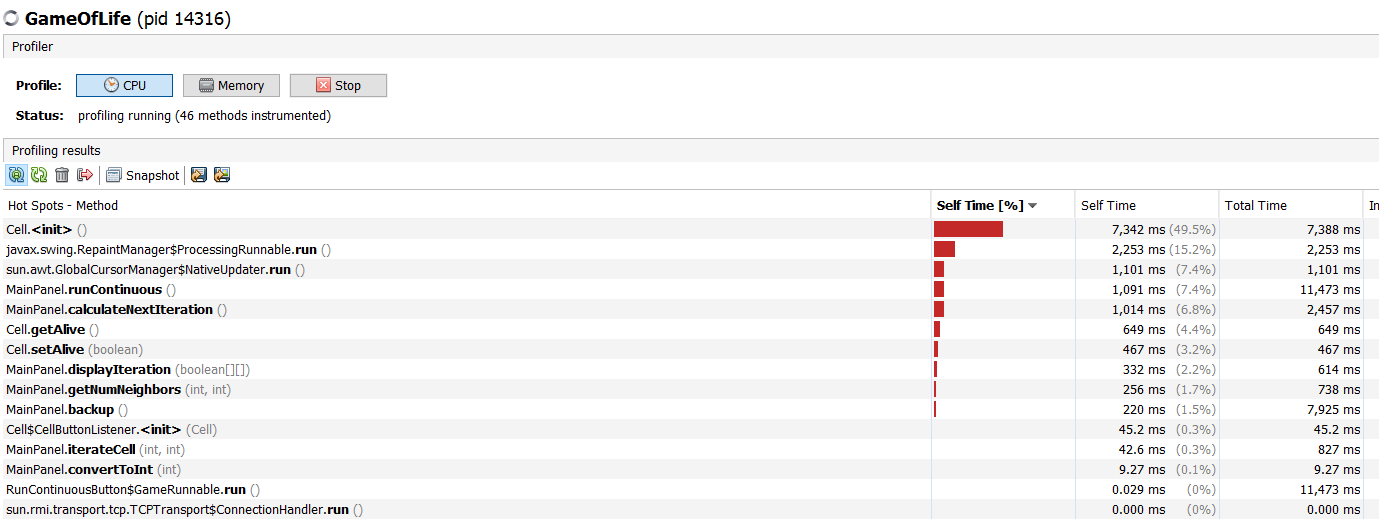
backup**();**

calculateNextIteration**();**

**}**

**}**

The thread sleep method is useless. It increases the time cost and is useless to the functionality. We can delete it and watch the program’s performance.



**This time we can see that the the Cell init takes most of the time. This is because of the ToString method in the cell class.**

**public String toString() {**

**String toReturn = new String("");**

**String currentState = getText();**

**/\*\*for (int j = 0; j < \_maxSize; j++) {**

**toReturn += currentState;**

**}**

**if (toReturn.substring(0,1).equals("X")) {**

**return toReturn.substring(0,1);**

**} else {**

**return ".";**

**}\*\*/**

**if(currentState.equals("X"))**

**return currentState;**

**else**

**return ".";**

**}**

**We can delete the useless for loops. And the performance after is shown below:**

