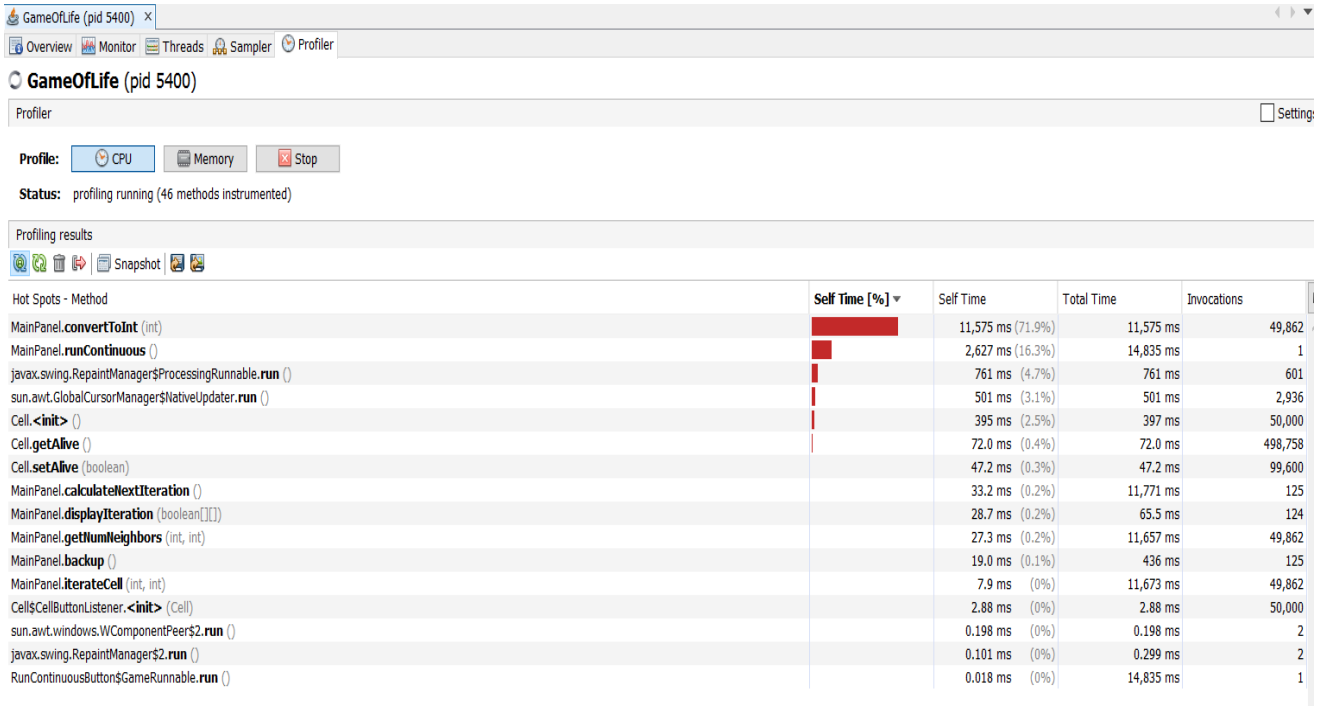
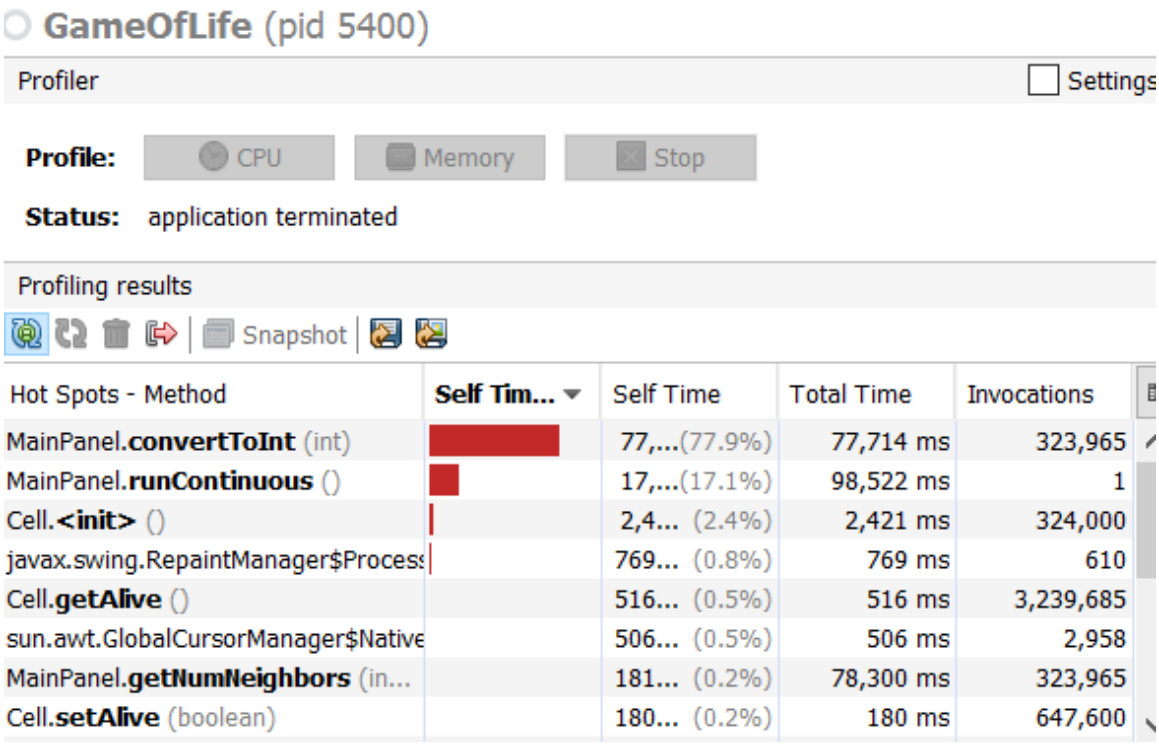
In this program ,we use the Visual VM profiler-cpu to judge the performance of the GameOfLife program.

This is the performance of the original program.





From the screenshot above, we can see that the convertToInt method takes most of the time. By improving the code in this method , we can improve the performance of the program. Inside the convertToInt method:

```
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;
     }
```

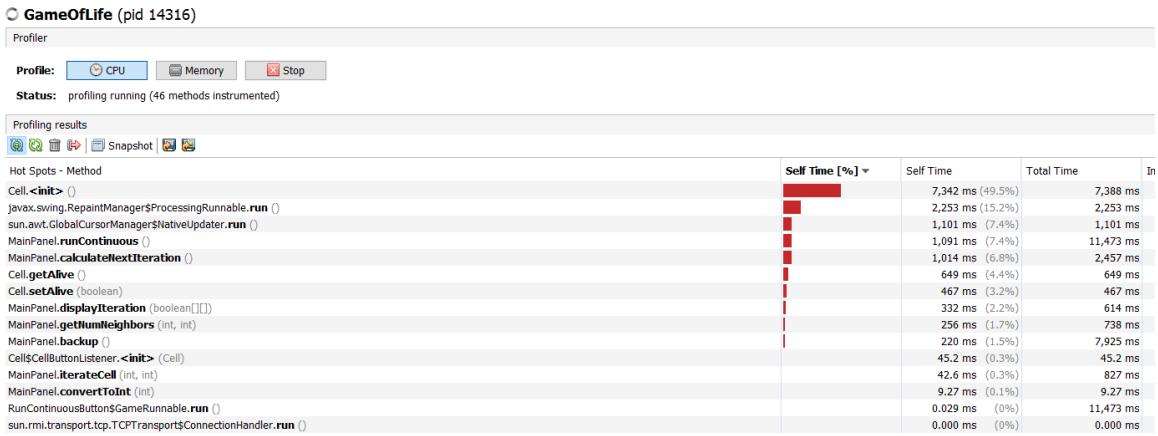
This is the performance after we revise the convertToInt method:

This time, 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. In this method, the for loop is unnecessary and can be deleted.

```
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 ".";

      }
```

The performance after improvement is shown below:

Hot Spots - Method	Self Time [%]	Self Time	Total Time	Invocations
javax.swing.RepaintManager$ProcessingRunnable.**run** ()		3,415 ms (42.6%)	3,415 ms	1,327
gameoflife.Cell.**<init>** ()		3,392 ms (42.3%)	3,403 ms	217,125
gameoflife.Cell.**setAlive** (boolean)		355 ms (4.4%)	355 ms	434,487
gameoflife.MainPanel.**getNumNeighbors** (int, int)		134 ms (1.7%)	221 ms	217,125
gameoflife.MainPanel.**runContinuous** ()		124 ms (1.6%)	4,427 ms	1
sun.awt.GlobalCursorManager$NativeUpdater.**run** ()		108 ms (1.3%)	108 ms	479
gameoflife.Cell.**getAlive** ()		107 ms (1.3%)	107 ms	2,171,250
gameoflife.MainPanel.**calculateNextIteration** ()		106 ms (1.3%)	719 ms	965
gameoflife.MainPanel.**backup** ()		84.8 ms (1.1%)	3,583 ms	965
gameoflife.MainPanel.**displayIteration** (boolean[][])		76.2 ms (1%)	353 ms	965
javax.swing.JComponent$2.**run** ()		67.9 ms (0.8%)	67.9 ms	15,248
gameoflife.MainPanel.**iterateCell** (int, int)		31.7 ms (0.4%)	259 ms	217,125
gameoflife.Cell$CellButtonListener.**<init>** (gameoflife.Cell)		10.8 ms (0.1%)	10.8 ms	217,125
gameoflife.MainPanel.**convertToInt** (int)		6.11 ms (0.1%)	6.11 ms	217,125
sun.awt.SunToolkit$1.**run** ()		0.846 ms (0%)	0.846 ms	1
gameoflife.Cell.**reset** ()		0.545 ms (0%)	2.46 ms	225
gameoflife.MainPanel.**clear** ()		0.276 ms (0%)	2.74 ms	1
gameoflife.Cell.**resetBeenAlive** ()		0.150 ms (0%)	0.149 ms	237
gameoflife.RunContinuousButton$RunContinuousButtonListener.**actionPerformed** (java.awt.event.ActionEvent)		0.102 ms (0%)	0.111 ms	1
gameoflife.Cell$CellButtonListener.**actionPerformed** (java.awt.event.ActionEvent)		0.087 ms (0%)	1.2 ms	12
gameoflife.StopButton$StopButtonListener.**actionPerformed** (java.awt.event.ActionEvent)		0.035 ms (0%)	0.035 ms	1
gameoflife.ClearButton$ClearButtonListener.**actionPerformed** (java.awt.event.ActionEvent)		0.025 ms (0%)	2.76 ms	1
gameoflife.RunContinuousButton$GameRunnable.**run** ()		0.016 ms (0%)	4,427 ms	1
gameoflife.RunContinuousButton$GameRunnable.**<init>** (gameoflife.RunContinuousButton)		0.009 ms (0%)	0.009 ms	1
gameoflife.MainPanel.**stop** ()		0.000 ms (0%)	0.000 ms	1