

Comparison between Windows CE and J2ME Applications Based On Memory Consumption and Performance

Nauman. M. Malik, MS candidate in Computer Science at CSU, Chico
(email:mmahmood@mail.csuchico.edu)

Abstract – The choice of the programming language for mobile phone application development depends on the performance limitations of these languages. This paper discuss and compare J2ME and Windows CE as programming languages used for mobile phone application development based on performance and memor consumption.

Index Terms—J2ME, Windows CE, .NET, Mobile application performance.

I. INTRODUCTION

For many people mobile phones has become an essential part of the life. It gives us facilities to play with it, call family and friends, send emails and text messages, and manage schedules. The involvement of mobile phones in our daily life has increased the market for mobile phone based applications. There are three main players in mobile phone application development i.e. Java ME/J2ME, Symbian C++ and Windows CE [4]. Symbian OS has the market monopoly and it is clearly dominating the mobile phone market [1].

Java-based application comes under the category of third party development tools. For that Sun has come up with a trimmed down version of Java as Java ME, formally known as J2ME. Microsoft has come up with the operating system for mobile phones and named it Windows CE, where CE stands for Compact Edition [4]. The development kit for J2ME is freely available online and the development environment used is NetBeans.

Nauman, MALIK

Similarly, for Windows CE the development platform is Microsoft Visual Studio which includes all the libraries and emulator to test your application in.

The application development process for both J2ME and Windows CE is very robust and fast but still for developers it is a hard decision to choose between these two. In this paper I have tried to compare J2ME and Windows CE based applications on the same matrix. A program has been written for both platforms with same functionalities. Later this program is tested on both platforms and is judged on the bases of time and memory consumption.

II. RELATED WORK

Mobile application development needs different techniques that are not familiar to the general PC developer. It is mostly due to the limitations of the mobile platforms. In PC environment it is very easy to upgrade the hardware to cater more complex applications. But this is not the same with mobile environment, where it is developer's responsibility to produce optimized software with minimal overhead to produce the desired results. The work done by Fadi is about the generic paradigms related to memory and code optimizations for Symbian C++ applications on mobile phones. It also describes the actual test results of these techniques on Symbian OS based devices and their importance for mobile application developers [1].

Nowadays, smart phones have a built-in camera and a good processing capability. The captured images can be processed and used in new applications on the fly. The paper

Nauman, MALIK

written by Celeste presents the performance evaluation of image processing applications written in Symbian C++ and J2ME. It also talks about the possibility of developing different applications in these languages and Smart phone's support to extend these functionalities [2].

The next generation mobile phones are fully equipped with online access of information and services. With the enhanced performance ability of these phones it is very critical to choose the best programming language that can support these features. This paper written by Rashid evaluates Symbian C++ and J2ME as potential development language to develop online services based applications. This paper discusses about different pros and cons of both the languages and evaluates the performance and memory issues [3].

III. TECHNICAL BACKGROUND

a. Algorithm Used

The logical design for this application is very simple and straightforward. This application takes a number as an input and calculate Fibonacci number. The results are calculated for a set of number from 5 to 35 and are then compared on the bases of time consumption and memory usage. After the execution, application shows the actual result of the Fibonacci number and time consumed in milliseconds. This software takes only one input and shows the desired results in an output box. The menu has two options i.e. Calculate Fibonacci & Clear. While the Clear option resets the values in both text-boxes and make the software ready to be used again.

b. J2ME base application

This J2ME base application is built using NetBeans 6.1 as development environment and J2ME SDK 2.2 is used as the development library. The developed application is a Java MIDlet and deployable to any J2ME based mobile phone or handheld device.

c. Windows CE base application

This Windows CE base application is built using Microsoft Visual Studio 2005, and the programming language used is C#. The Emulator used is Smartphone 2003 and the application is a Smartphone application for Windows CE. This software is deployable to any Windows CE based mobile phone and handheld device.

d. Hardware used

This experimentation is not conducted on real mobile phones. For both J2ME and Windows CE emulators are used. Mobile phone Emulators are software programs that take the same input and give the same output as any regular mobile phone. It also mimics the same hardware specifications as any handheld device. The computer used for this experiment is Dell Vostro 1000 with 2 GHz AMD 64 Bit process and 1 GB RAM.

IV. EXPERIMENTS

a. Experimentation with J2ME

The screenshot in Figure 1 shows the Emulator input, output, controls and interface. The first field on the top which says “Enter Number” takes the number that we want to calculate the Fibonacci of. By choosing option “2 Fibonacci”, we instruct this MIDlet to calculate Fibonacci number and display the results in “Result” box. The first output is the Fibonacci of number 35 and second output is the number of milliseconds it took to calculate it, which in this case are 7937 ms.

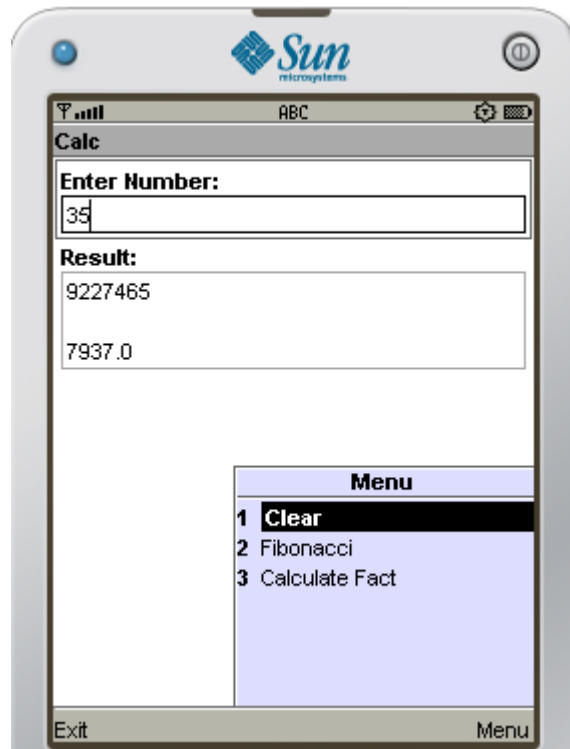


Figure 1-J2ME Emulator with controls

The snippet of the final code written in JAVA for this experiment is shown in Table 1. It shows the Fibonacci number is calculated from the input by the user. The result is output to the display. At the end it calculates the total time taken and displays

Table 1
J2ME Code to calculate Fibonacci

```
Date t1 = new Date();
int iFibResult = 0;
int iConvert = Integer.parseInt(txtInput.getString());
iFibResult = Fibonacci(iConvert);
txtResult.setString(String.valueOf(iFibResult));
Date t2 = new Date();
double result = t2.getTime() - t1.getTime();
txtResult.setString(txtResult.getString()+ "\n\n"+result);
```

Nauman, MALIK

it on the screen.

b. Experimentation with Windows CE

The screenshot in Figure 2 is showing the input, output and controls for Smartphone Emulator. The first field on the top is to input the number that we want the Fibonacci number of and the blue color box below shows the output. As it shows in the picture the input provided is '35'. '1 Fibonacci' is the control that instructs the software to start calculating the Fibonacci number. The output area shows the final result and the total time taken. In this case it took 37000 milliseconds to calculate Fibonacci number for integer 35.



Figure 2-Windows CE Emulator with controls

The snippet of the final code written in C# for this experiment is shown in Table 2. It calculates the Fibonacci number and the result is output to the display after final loop. At the end of this loop it calculates the total time taken and displays it on the screen.

Table 2
C# code for Windows CE

```
DateTime dt1 = DateTime.Now;
int iFabResult = 0;
int iConvert = Convert.ToInt32(txtInput.Text.Trim());
iFabResult = Fibonacci (iConvert);
txtResult.Text = iFabResult.ToString();
DateTime dt2 = DateTime.Now;
TimeSpan ts = dt2 - dt1;
double result = ts.TotalMilliseconds;
txtResult.Text += "\r\n" + result;
```

c. Results

Table 3
Experimentation Performance Results

	J2ME	Windows CE
Binary file size:	2.86 KB (2,929 bytes)	8.00 KB (8,192 bytes)
CPU usage:	33%	50%
Available Memory:	299208 K	250004 K

Table 4
Experimentation Timing Results

Input Number	Windows CE Time (ms)	J2ME Time (ms)	Fibonacci Result
5	1000	0	5
10	0	0	55
15	0	0	610
20	0	15	6765
25	1000	63	75025
30	3000	734	832040
35	35000	8109	9227465

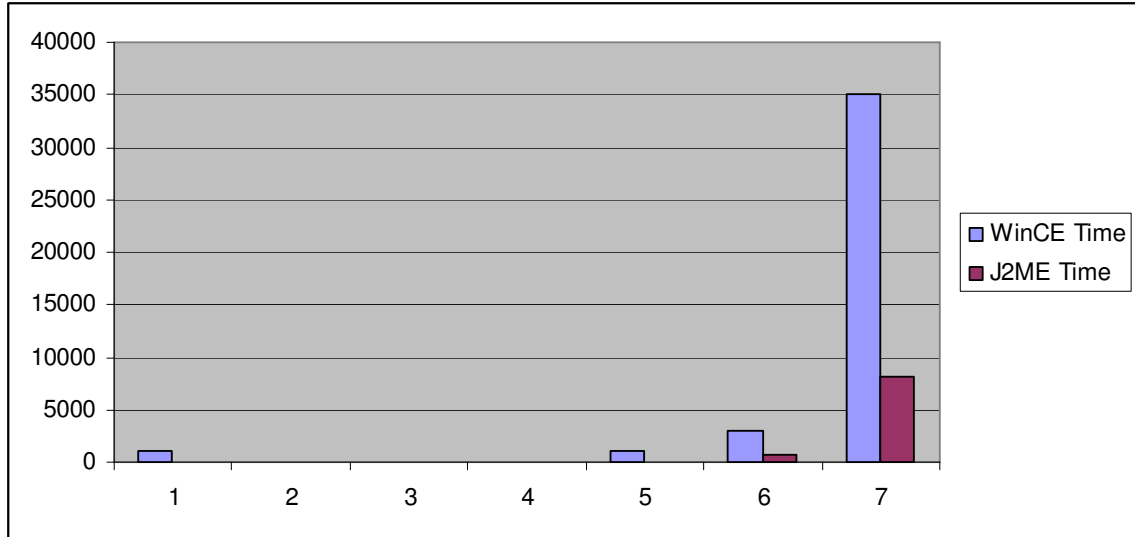


Figure 3- Bar Graph between Windows CE and J2ME time performance

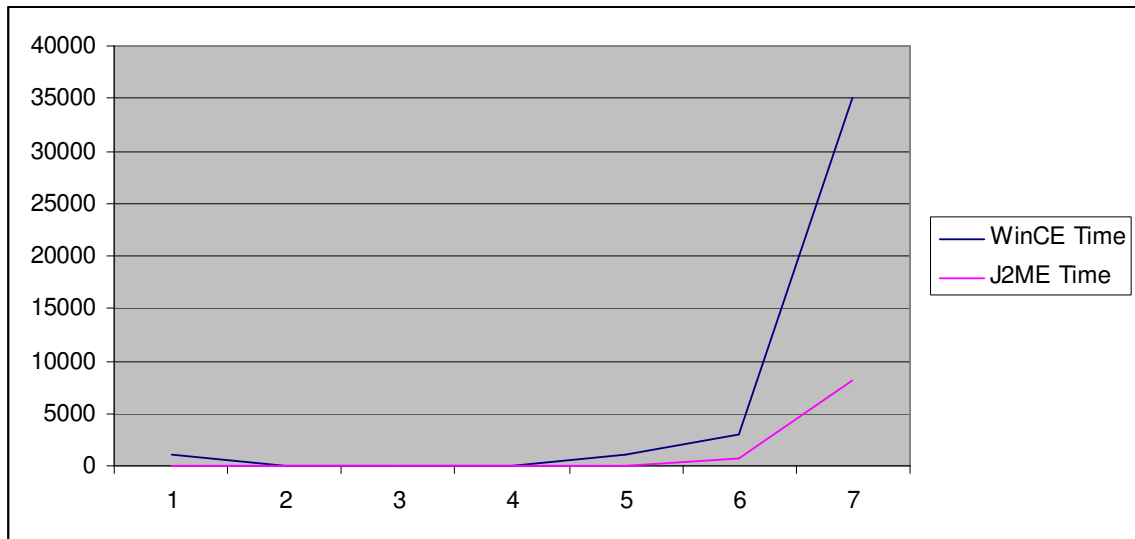


Figure 4- Line Graph between Windows CE and J2ME time performance

V. PERFORMANCE ANALYSIS

The above given results show that overall performance by J2ME based application is far better than Windows CE based application. The experiments are run for a set of number from 5 to 35 as shows in table 4. The memory consumption by Windows CE application is about four times more then the application written in J2ME. Similarly the CPU usage in Windows CE was about 17% more than the CPU usage for J2ME

Nauman, MALIK

applications. As figure 4 shows that initially both J2ME and Windows CE applications performed equally, but as the value of number started increasing, windows CE performance started decreasing.

There are many aspects of this situation that should be analyzed before any final decision. The very first constraint about this experimentation was the unavailability of real phones. In both the cases an Emulator was used and it is not clear if Emulator is using the full processing bandwidth or only using that it should be using while mimicking the desired phone. The same situation is with memory usage as it is different in a real phone than a computer.

Overall bad performance by Windows CE is a major issue and it might have to do something with the final code generated. As the final binary file generated by Windows CE is about 5 KB bigger in size than J2ME and it means that number of instructions for Windows CE are more than J2ME. It is an interesting fact as both the languages compile to an intermediate language and runs on a framework i.e. Java Runtime & .NET framework.

VI. CONCLUSION

The processing power of J2ME base applications is far better than the Windows CE base applications. The initial performance of Windows CE was same as J2ME application, but later it started decreasing. The two reasons can be number of instructions generated by .NET framework were more than the ones generated by J2ME and the other

Nauman, MALIK

reason can be the garbage collection and its implementation for mobile phones. More investigation is required to find the real cause of this scenario and how to overcome this deficiency. The future experiments should be conducted on real mobile phones with more complex algorithms.

REFERENCES

- [1] Chehimi, "C++ optimizations for mobile applications," Consumer Electronics, IEEE Tenth International Symposium, March 2006, pp. 389—394.
- [2] Campo, "Performance evaluation of J2ME and Symbian applications in Smart camera phones," Digest of Technical Papers - IEEE International Conference on Consumer Electronics, Jan. 2007, pp. 668—747.
- [3] Rashid, O., "A comparative study of mobile application development in Symbian and J2ME using example of a live football results service operating over GPRS," 2004 IEEE International Symposium on Consumer Electronics – Proceedings, Sep. 2004, pp. 203—207.
- [4] Hoske, "Windows CE embeds itself in automation, control, and instrumentation," Control Engineering, Nov 1998, pp 103
- [5] Fang and He, "A Pocket PC based field information fast collection system," Computers and Electronics in agriculture, Nov. 2007, pp 254 – 260
- [6] Kortenkamp and Materlik, "Geometry teaching in wireless classroom environments using Java and J2ME," Science of Computer Programming, June 2004, pp 71—85