Implementation of Garbage Collection Java Application on Sun Java Real Time Operating System Mobile Embedded Application

Amandeep Kaur[1], Balpreet Kaur[2], Gurpreet Kaur[3]

[1,2,3]*Assistant Professor, Dept. of CSE, BBSBEC, FGS, Punjab, India*

Abstract- *Java possesses many advantages for embedded system development, including fast product deployment, portability, security, and a small memory footprint. As Java makes inroads into the market for embedded systems, much effort is being invested in designing real-time garbage collectors. Memory allocation can be done in constant time and sweeping can be performed in parallel by multiple modules. In this paper, garbage collection java application has been implemented on real time system mobile embedded application.*

Key Words— Security, garbage Collection, Embedded Systems.

1. INTRODUCTION

The need of automated garbage collection, or automated memory management, in terms of time and memory is oblivious. If used properly it will cut development time in projects, the bigger and more complex project, the more time and time is money. The ultimate garbage collector, or automated memory management scheme, should allocate the exact amount of memory needed when it is needed. It should also reclaim memory the moment it becomes useless to the running program.

2. COMPARISON OF GARBAGE COLLECTION TECHNIQUES WORST CASE EXECUTION TIMES

Garbage collection performances vary when we use reference counting technique. Generational garbage collection worst case allocation characteristics are different from reference counting.

Table 1: Comparison

	Worst Case Allocation	Worst Case Recycling	Generality
Malloc/Free	Walk free list	Constant	high
Garbage Collection	Constant	Size of memory (typically)	high
Reference Counting	Walk free list	Size of memory	Med(cycles)
Pool Analysis	Constant	Constant	low

3. IMPLEMENTATION

The java application has been executed on sun java real time mobile embedded application. The application is made run on the simulator of default color phone. The operating system on which the application is working is Symbian real time operating system.

Fig -1: Opening a project on real time java mobile embedded system

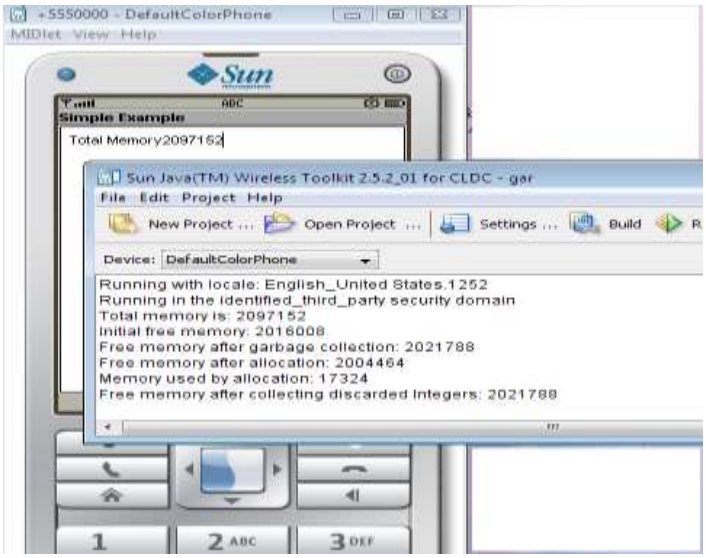
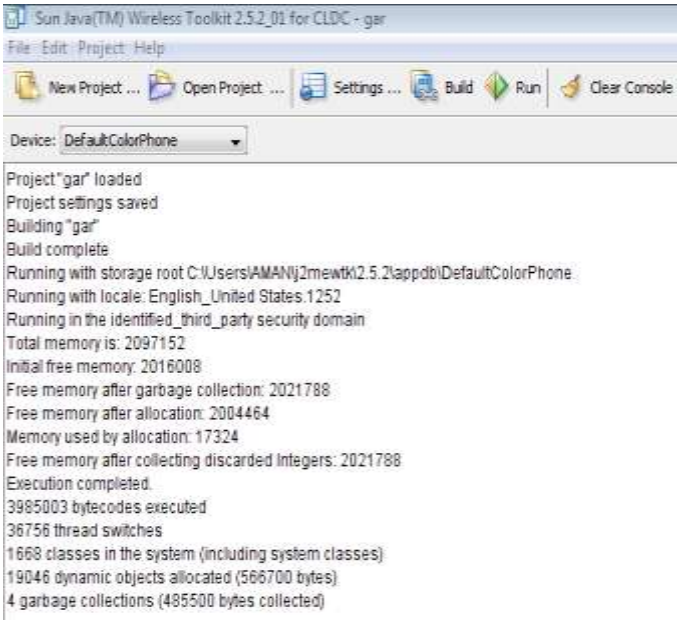


Fig -2: Executing java application on real time mobile embedded application

4. RESULTS

The screenshot of the garbage collection java application implementing on mobile embedded symbian real time operating system shows the free memory before and after garbage collection. During the process, 48550 bytes have been collected and memory freed from garbage data.



5. CONCLUSION

The use of the real time garbage collection together with the extensions defined in the real time specifications for java makes it possible to provide a more straightforward and simpler development of real time code using java. Even systems that do not require dynamic memory management within real time code becomes simpler, such that higher productivity and higher software quality can be expected. Such a system provides the advantage that made java so successful to the developer of real time systems.

REFERENCES

[1] Adrienne Bloss. Update analysis and the efficient implementation of functional aggregates, pages 26-38.ACM Press, 1990.

[2] Paul R. Wilson. Uniprocessor garbage collection techniques. Submitted to ACM Computing surveys 1994.

[3] Yoo C.Chung and Soo-Mook Moon.Memory allocation with lazy fits.

[4] Jaques Cohonu and Alexendru Nikolau.Comaprison of compaction algorithm for garbage collection.ACM transactions on programming languages and systems, 5(4):532-553, October 1983.

[5] Ben Cranston and rick Thomas. A simplified recombination scheme for the fibonacci buddy systems, pages 331-332.ACM Press, june 1975.

[6] David Detlefs. Automatic inference of reference-count invariants.

[7] David M.Harland.REKURSIV:Object-oriented computer arcitecture. Ellis Horwood Ltd.,1998.

[8] Roger Henriksson. Scheduling garbage collection in embedded systems. Phd thesis,Lund Institute of Technology,1998.

[9] Daniel S.Hirschberg. A class of dynamic memory allocation algorithams,pages 615-618. ACM Press, October 1973.

[10] Mark S.johnstone and Paul R. Wilson. The memory fragmentation problem: Solved? October 18, 1997