Effective Java Programming

efficient software development

Structure

efficient software development

- what is efficiency?
- development process
- profiling during development
- what determines the performance of applications in Java

Motto

"More computing sins are committed in the name of efficiency (without necessarily achieving it) than for any other single reason - including blind stupidity"

William Allan Wulf

What is efficiency?

- speed = efficiency?
- what influences the performance evaluation:
  - performance
  - memory requirements (RAM footprint)
  - run time
  - scalability
  - subjective perception of performance
- different priorities on client and server side

Computational performance

- this "performance" to most users
- key aspects:
  - choice of algorithm
  - choice of data structures
  - number of instructions to perform
- truly high-performing application needs to take other factors besides calculations into account

Memory requirements

- amount of memory consumed can significantly affect performance
- important factors
  - whether the application forces the OS to paging (drastic drop in performance)
  - amount of memory in target environment
  - efficient absorption of resources (your application is not alone)

Program start time

- affect the subjective assessment of performance
- most significant for client applications
  - packaging (JAR applets)
  - architecture with lazy loading plugins (eg Eclipse)
  - control class loading
- differences in the modes of the virtual machine
  - client - quick start
  - server - code optimization

Scalability

- determines how the system works when stressed
- in a poorly scalable system as the load increases, performance decreases dramatically
- truly scalable systems are more resistant to increased load
  - eg text editor when opening 5 or 5000 pages
- performance should be measured at the planned load of the system
- scalability examples

Subjective performance impression

- users do not measure time with a stopwatch - they feel
- more important is how fast a program seems to be rather then it actually is
- critically important to GUI
- reaction time and response time
- reduction of response time
  - screens
  - change the cursor
- tricks
  - progress bars
  - rapidly changing subtitles

Efficient software development process

- work on productivity is not limited to a single step
- must be part of the whole production plan
- can not be confined only to the coding
- decent analysis and design are the key
- without a good analysis you do not create a good design
- without good design you will not achieve good performance
- code optimizations alone are not enough

Efficient software development process

- basic OOSD process has four main phases:
  - analysis
  - designing
  - coding
  - testing
- capacity planning requires one more:
  - profiling
- for use in any OOSD process

Analysis

- defines what and how the system is doing
- abstracts from low-level issues as:
  - programming language, syntax
  - data Structures
  - classes, methods
- It includes:
  - requirements analysis
  - definition of system boundaries
  - creation of a use case model

Analysis vs. performance

- system requirements that affect performance:
  - minimum configuration (RAM, CPU)
  - recommended configuration (RAM, CPU)
  - link speed (network)
  - other applications running on the system
- performance requirements:
  - response times at a given load
  - boot-time
  - reaction time
  - specific (ie, the number of frames per second during teleconference)

Analysis vs. performance

- system boundaries
  - knowledge of what the system will not do can open many possibilities of optimization
- use case model
  - defines the prototypes that should be created to measure performance
- omission of performance in the analysis phase
  - how do you know that the system is fast enough?

Designing

- object-oriented design has a number of factors affecting the quality of the solutions
- encapsulation is critical in terms of performance
  - worsens performance (stack grows), but ...
  - facilitates testing of many algorithms, the choice of the best and the amendment of the existing ones
  - improves maintenance

Coding

- code has an obvious impact on performance
- very similar codes can often have huge differences in performance
- many attempts to optimize the code are bad
  - for the compiler
  - not where it is needed

Testing

- checks whether the system meets the quality and performance requirements
- performance testing is based on:
  - checking whether the requirements are met
  - comparing alternative solutions
- tests can be performed on individual modules
  - quickly determine the limits of performance
  - system will not run faster than its parts

Profiling

- often the system does not meet performance requirements
- you should make changes
- a common mistake is to focus on performance irrelevant parts
- reason - poor conditions
- solution - profiling

Profiling

- where are the bottlenecks in your application?
- profiling is to identify the components consuming the most resources
- specialized tools are designed for this task
- allows you to identify parts of the system needing most changes
- know what changes will bring the greatest benefits

What determines efficiency in Java

- The overall performance of applications in Java depends on:
  - application project
  - execution speed of Java code
  - speed of the native libraries
  - speed of hardware and operating system
- JVM is responsible only for execution speed and has code optimization mechanisms
  - JIT (Just In Time) - compilation of code before executing
  - HotSpot - continuous code compilation at runtime
- You can not speed up native calls
  - I/O, graphic generation, GUI, DB

Conclusions

- what affects the evaluation of performance?
- how to make the system appear to be faster?
- in which the phases of software development performance should be taken into account?
- what is the relationship between analysis and performance?
- what are the advantages of profiling?