

# Profiling

Software Engineering



Chair of Programming Methodology

**ETH**

Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

# Agenda for today

1. Profiling
2. General Guidelines
3. Optimizing
4. NetBeans Profiler
5. Interactive Exercise

# Profiling

- The activity of analyzing performance in your application
- The collection of run-time measurements to determine the meaning of the analysis
- The tool instrumentation to collect the data
- The insight given when analysis is finished

## Goal

- The optimization of the program leading to faster and more efficient implementation

# 80/20 Rule

- 80% of run-time is spent in 20% of the code
- Focus on the 20%, since optimization of the 80% will not provide overall benefit
- Programs are too complex, do not spend time optimizing the thousands of code paths that are not relevant
- Use a tool! You do not know the relevant 20%!

# General Procedures

- Any operation that is perceptibly slow to the user.
- Focus on the functionality used most often.
- If large percentage of program spent in  
small portion of code, look for low-level optimizations
- If no obvious hotspots, performance may be improved through  
high-level improvements (i.e. architectural changes)

~~high-level improvements (i.e. architectural changes)~~

# Improving Performance

- Now that you have found the hotspots, how do you optimize your code?
- Several ways
  - Change algorithms
  - Cache values (especially strings!)
  - For graphic applications, reduce the amount of drawing per frame

-

U

- Create your database after your

# NetBeans Profiler

- Uses the JFluid technology
  - Research project from Sun Microsystems
  - Dynamic bytecode instrumentation
- Gives accurate results of program analysis
- Uses data-mining techniques to deal with the large data sets
- Included with NetBeans
- And, it is easy to use

# Profiler – Options

- Monitor Application
  - High-level information about the target JVM, thread activity and memory usage.
  - Helpful for program and JVM analysis
- Analyze Performance
  - Detailed data on program performance
  - Analyze entire application, part of application, or startup performance
  - Useful for profiling the different life-cycles of your program



# Profiler – Options

- Analyze Code Fragment Performance
  - Extremely detailed data on a piece of code
  - Useful for analyzing low-level code and checking different optimizations
- Analyze Memory Usage
  - Data on object allocation and garbage collection
  - Useful for analyzing memory usage
- Run Custom Profiling
  - Profiling collects an overwhelming amount of data
  - Filter the data and what to analyze depending on what you want

# Interactive Exercise

- Today we will profile your programs
  - Discover where your hotspots are
  - See how much memory your application consumes
- Figure out where you may need optimization
- Analyze the performance of your program and determine the hotspots
- Determine a possible optimization