Java Programming Languages Team

Mathias Ricken
mgricken@rice.edu

Robert "Corky" Cartwright
cork@rice.edu

Other collaborators: Walid Taha, Dung Nguyen, Stephen Wong, Edwin Westbrook, Jun Inoue

Testing Concurrent Programs

- **Concurrent programs becoming more important**
  - → **Computers become faster by adding more processor cores**
  - → **To benefit from new hardware, programs have to concurrently use more than one processor core**
- **Unit testing is effective for single-threaded programs**
  - → **Current approaches fail for concurrent programs**
- **Thread switching is non-deterministic and machine-specific**
  - → **Success of a unit test does not imply correct behavior under all possible schedules and on all machines**
  - → **Most programs are concurrent:**
    - → **GUI: separate thread for display**
    - → **Multi-core: programs must be concurrent to benefit**
  - → **Current tools not effective or easy to use on large projects**
- **Concutest: A Framework for Testing Concurrent Programs**
  - → **Concurrency-aware extension of JUnit (ConcJUnit)**
  - → **Lightweight checking of concurrency invariants (ThreadCheck)**
  - → **Logging of method execution to simplify and decouple unit tests for reactive programs**
  - → **Execution with short delays inserted at critical places to test different execution schedules**
- **DrJava Case Study for ConcJUnit and ThreadCheck**
  - → **900 unit tests in DrJava code base**
  - → **20 previously unknown problems detected by Concutest**
  - → **1% slowdown**
- **Website: www.concutest.org**

DrJava

- **Integrated development environment for Java**
  - → **Lightweight, cross-platform program**
  - → **Well-suited for beginning programmers and students**
  - → **Read-evaluate-print loop**
  - → **Develop DrJava in DrJava**
- **DrJava first released in January 2002**
  - → **Began working on DrJava in 2006, now one of two principal developers**
  - → **230,000 downloads in first 5 years, 870,000 downloads in 5 years since**
  - → **Recently surpassed a million downloads**
- **Implemented many useful features**
  - → **Predictive input dialog ("Go to File", "Complete Word under Cursor")**
  - → **Clipboard history**
  - → **Multiple underlined searches ("Find All")**
  - → **Detachable tabbed panes and debugger window**
  - → **Persistent breakpoints and bookmarks**
- **Use DrJava as tool to make research accessible to students**
  - → **Integrated Concutest**
  - → **Integrated JavaMint**
  - → **Integrated other Rice research projects (NextGen, Habanero Java)**
- **Website: www.drjava.org**

Multi-Stage Programming

- **Program abstractions (e.g. recursion) without performance overhead**
  - → **Abstractions make programs easier to understand**
  - → **Staging moves abstractions out of the runtime into a code generation step**
  - → **Result:** **Code written using abstractions (e.g. `power`) is optimized for special cases (e.g. `square`)**
  - → **Killer example:** **Interpreters become compilers**

```
// unstaged power function in Java
double power(double x, int n) {
  if (n==0) return 1.0;           // overhead: if and comparison
  else return x * power(x, n-1); // overhead: function call
}
```

```
// staged power function in Java Mint, runs 9x faster than unstaged
Code<Double> power(Code<Double> x, int n) {
  if (n==0) return <| 1.0 |>;
  else return <| `x * `power(x, n-1) |>;
}
// overhead of abstractions removed in generated code:
power(<| 2 |>, 5) returns <| 2 * 2 * 2 * 2 * 2 |>
```

Benchmark	speedup	unstaged μs	staged μs
`power`	9.2	0.060	0.0065
`fib`	8.8	0.058	0.0065
`mmult`	4.7	13	2.7
`eval-fact`	20	0.83	0.042
`eval-fib`	24	18	0.73
`av-mmult`	65	20	0.30
`av-mtrans`	14	1.0	0.071
`serialize`	26	1.5	0.057

- **Provably safe at compile time**
  - → **No compiler errors in generated code.**
- **Java Mint: Multi-Stage Programming in Java**
  - → **Expressive: Imperative programs and most side effects allowed**
  - → **Accessible: Java is widely used, many libraries are available**
- **Website: www.javamint.org**

Computer Science Education

- **Designed assignments and class projects for programming classes**
  - → **Marine Biology Simulation (OOP: object-oriented programming)**
  - → **Design Patterns for Parsing (OOP)**
  - → **Programming for Change (OOP, agile development)**
  - → **Bounded Buffer, Readers/Writers Locking (concurrent programming)**
  - → **Working on tutorial for Java Mint...**
- **Developed syllabus as instructor**
  - → **Principles of Object-Oriented Programming II (COMP 202)**
  - → **Production Programming: Concurrent Programming and DrJava (COMP 402)**