The Squawk JVM

Nik Shaylor
Senior Staff Engineer

Squawk is…

- Technology for constructing virtual machines using the Java language.
- A vehicle for virtual machine research.
- A VM suitable for small devices.

The Big Problem

- Virtual machines are complex, hard to understand, modify, maintain, and port
- Architectural weaknesses:
  - Interaction between compiled, interpreted, and native code
  - Inadequate modularisation
- C is not a good language for VM implementation
  - Too low level a language
  - No provision for garbage collection

Project Constraints

- JVM compatibility
- Highly modular design
- Favor simplicity over performance
- But also achieve reasonable performance
- Useable in small devices

Project Status

- Complete CLDC functionality
- Currently working with a slow interpreter
- Ported to x86, SPARC, Mac
- Recently also ported to 64 bit architectures

Squawk Team

- Bill Bush - Project PI
- Cristina Cifuentes - Compiler technology
- Nik Shaylor - Core VM technology
- Doug Simon - Core VM technology

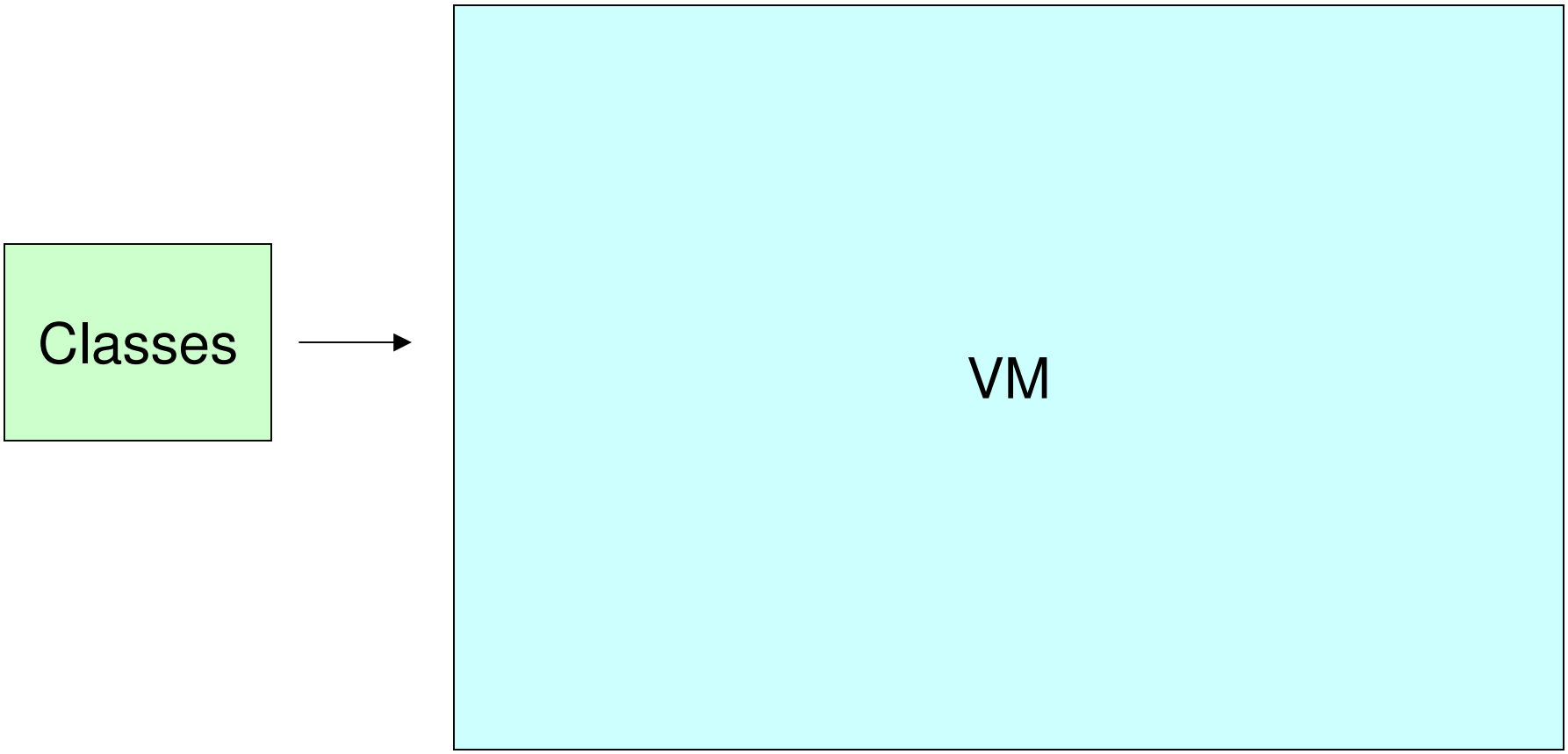
Agenda

- General Introduction to Squawk
- Isolates
- Suites
- Demo
- Q & A

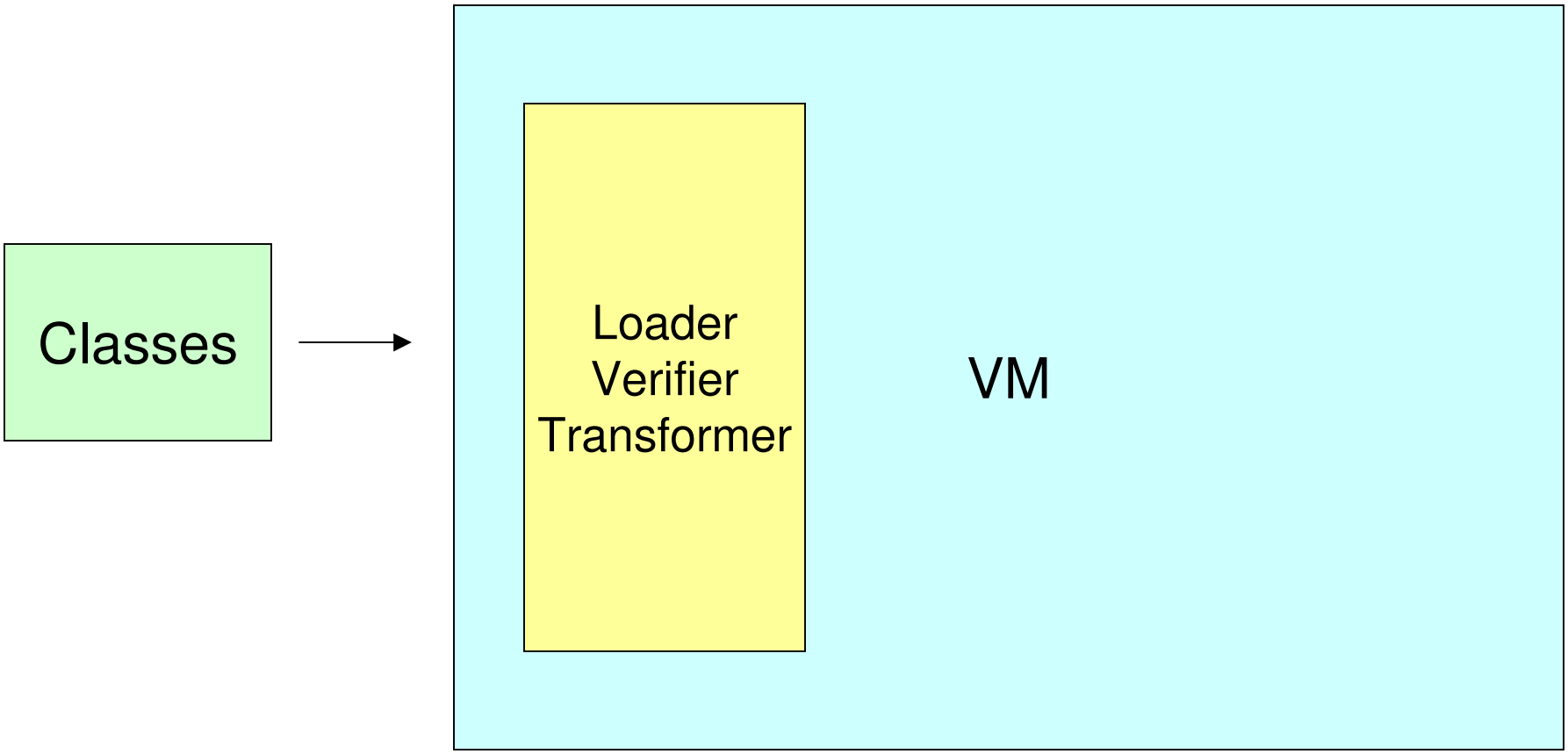
Technical Approach

- The system is designed around a single compiler that can be used dynamically and ahead-of-time
- Everything written in Java
- Bootstrapped using another JVM

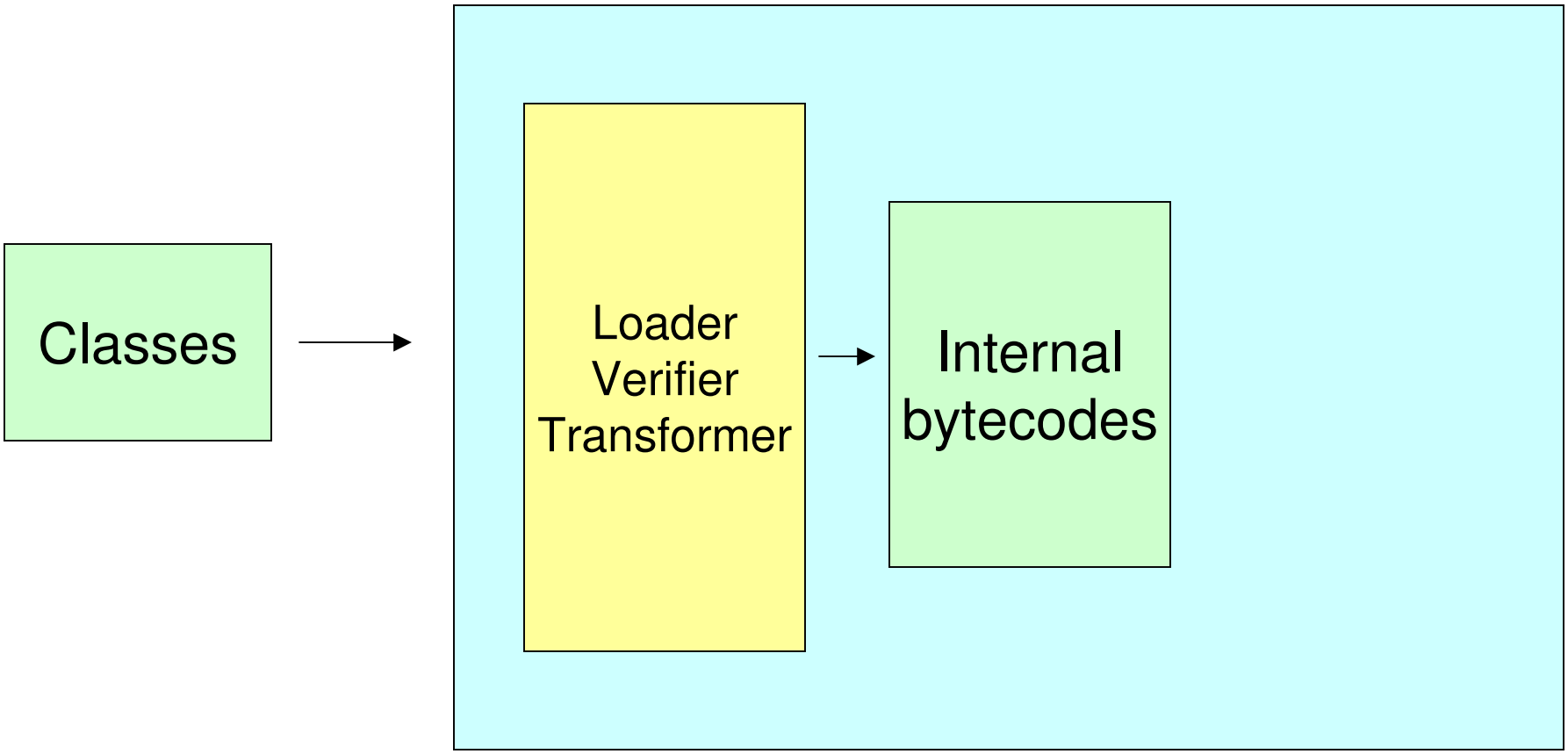
Technical Approach



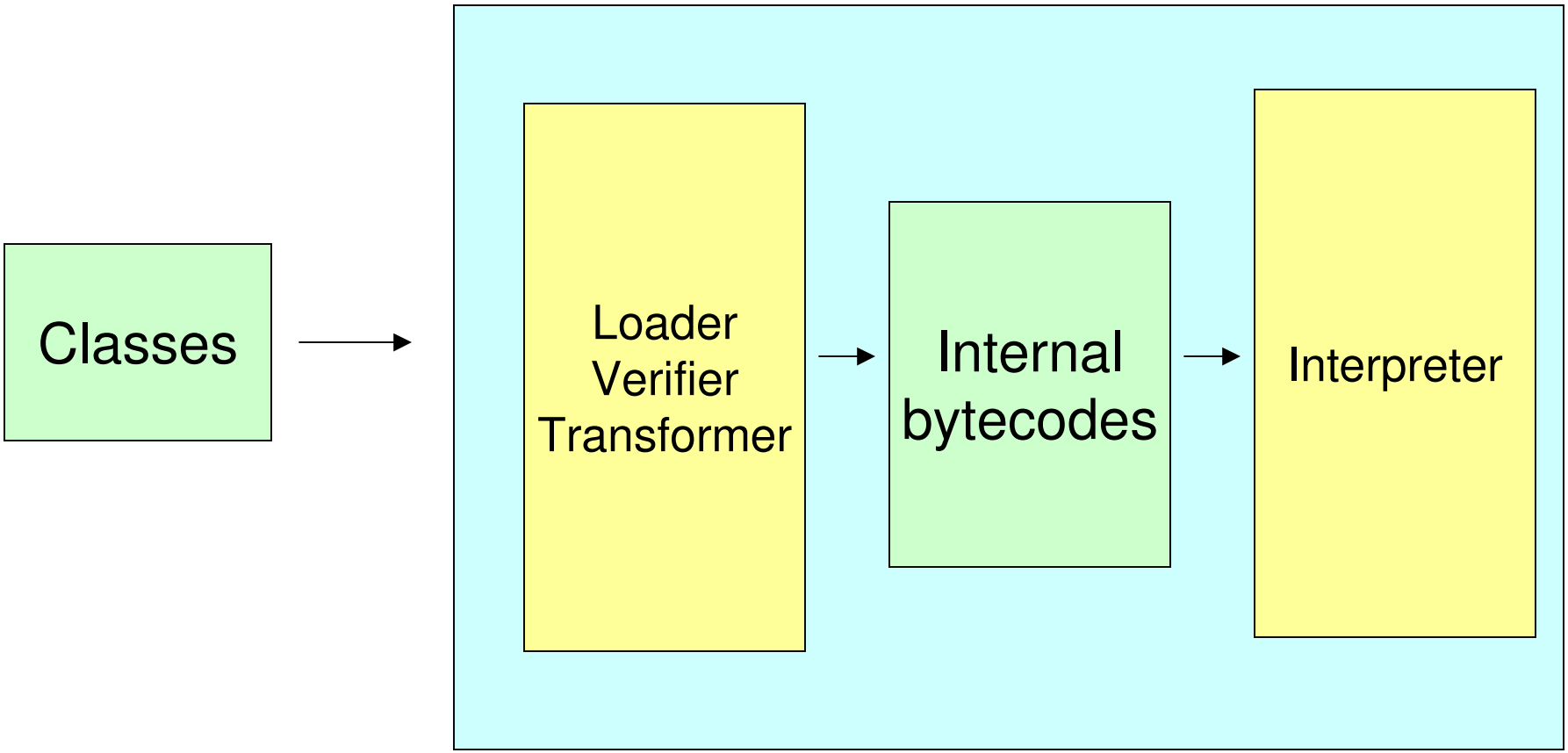
Technical Approach



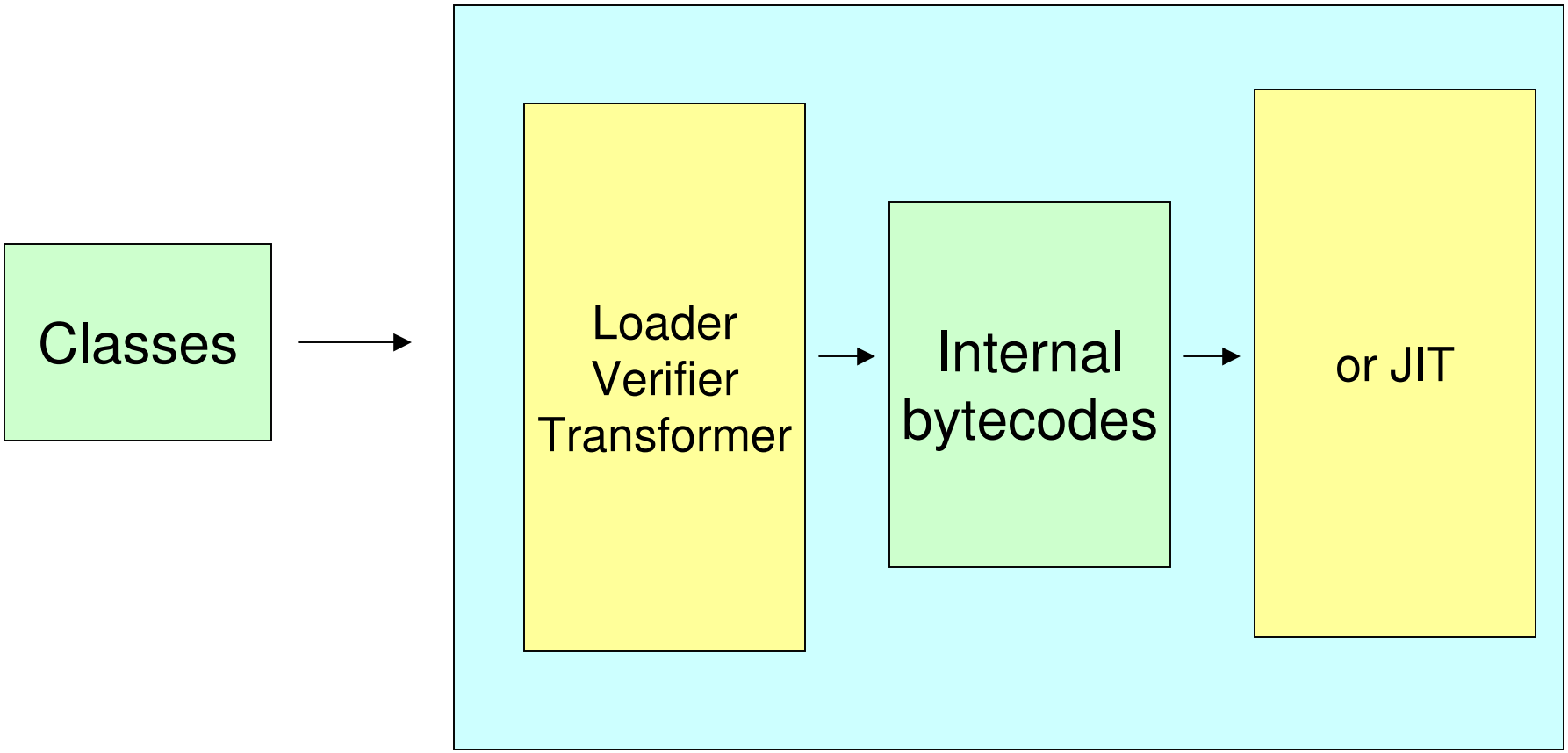
Technical Approach



Technical Approach



Technical Approach



Technical Approach

What is in a JVM?

- Class Loader
- Verifier
- Interpreter
- Garbage collector
- Thread scheduler
- Dynamic compiler
- Dynamic assembler
- Native code

Technical Approach

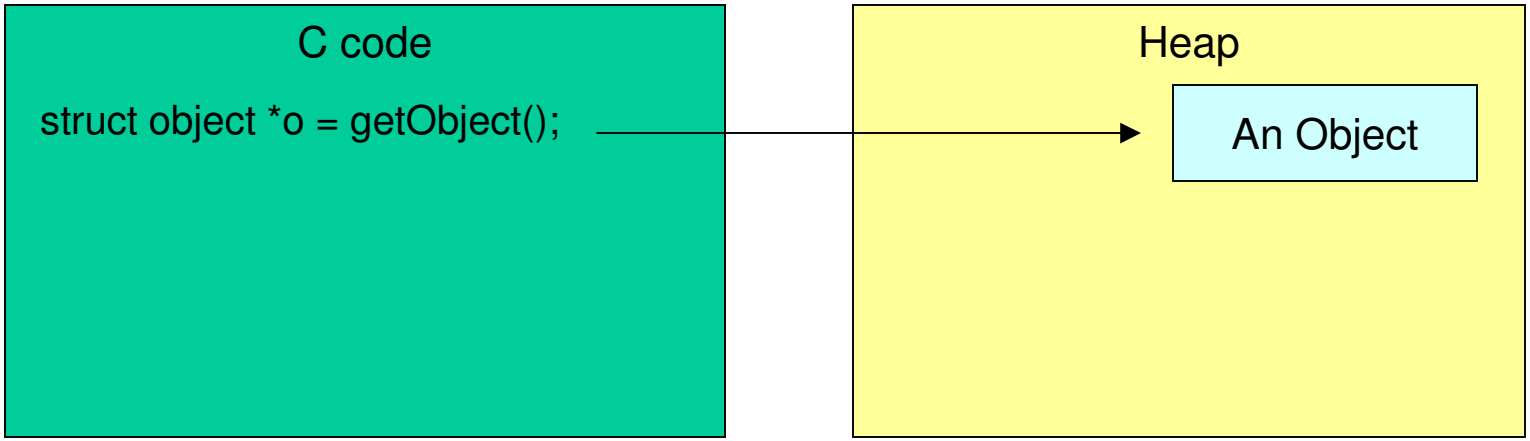
What is in a JVM?	For most JVMs
• Class Loader	• C/C++
• Verifier	• C/C++
• Interpreter	• C/C++
• Garbage collector	• C/C++
• Thread scheduler	• C/C++
• Dynamic compiler	• C/C++
• Dynamic assembler	• C/C++
• Native code	• C/C++

Technical Approach

What is in a JVM?	For most JVMs	Squawk
• Class Loader	• C/C++	• Java
• Verifier	• C/C++	• Java
• Interpreter	• C/C++	• Java
• Garbage collector	• C/C++	• Java
• Thread scheduler	• C/C++	• Java
• Dynamic compiler	• C/C++	• Java
• Dynamic assembler	• C/C++	• Java
• Native code	• C/C++	• C/C++

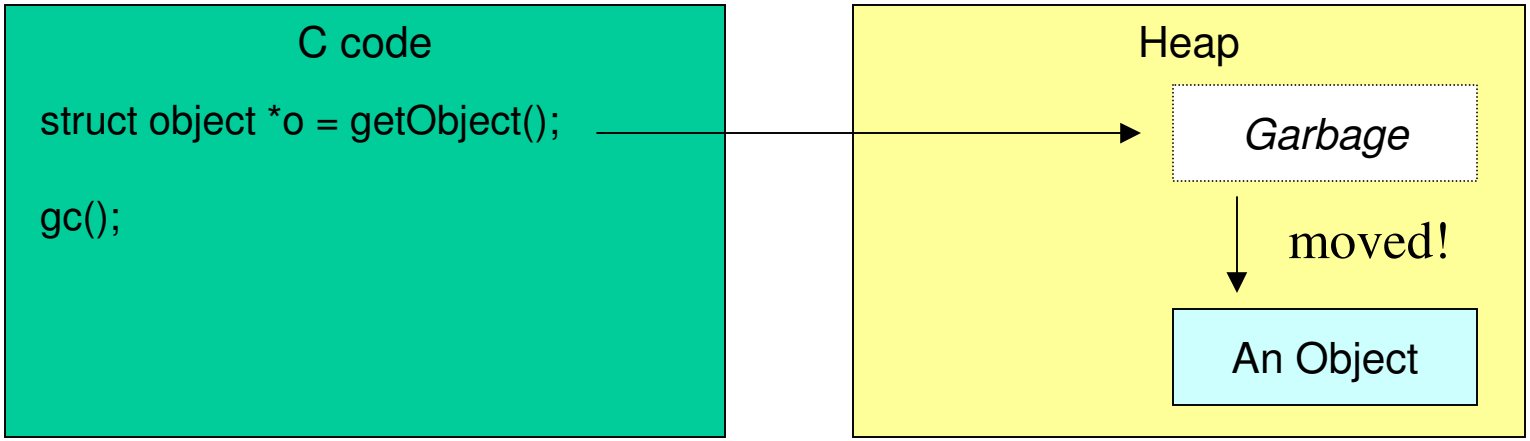
The Big Pointer Problem

There is no support in the C language for compacting garbage collectors.



The Big Pointer Problem

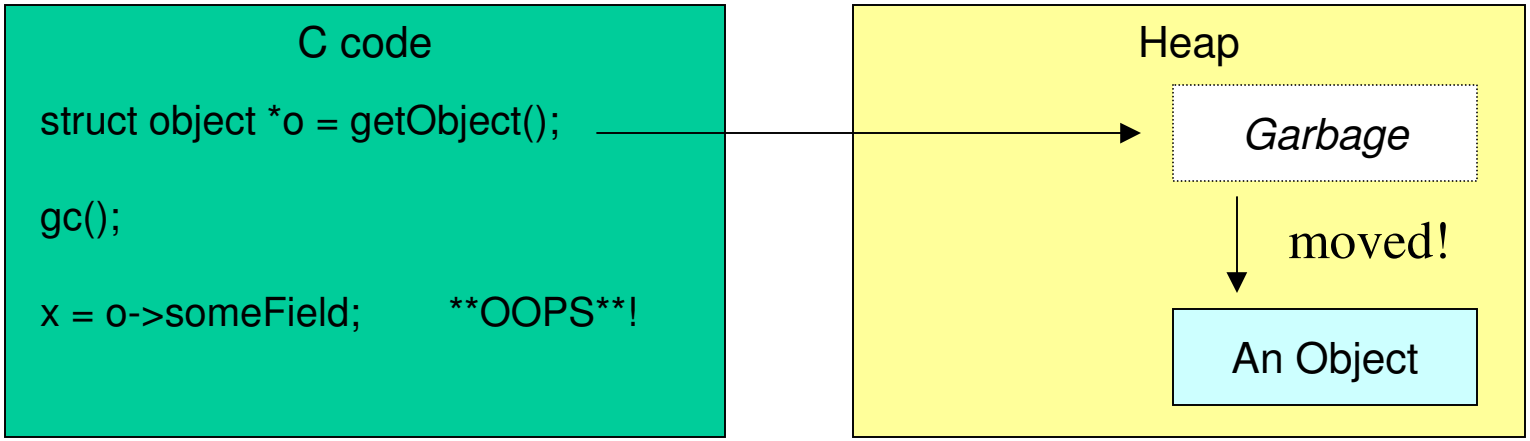
There is no support in the C language for compacting garbage collectors.



Calling the garbage collector moved the object and the local variable that pointed to it in the C code was not updated.

The Big Pointer Problem

There is no support in the C language for compacting garbage collectors.

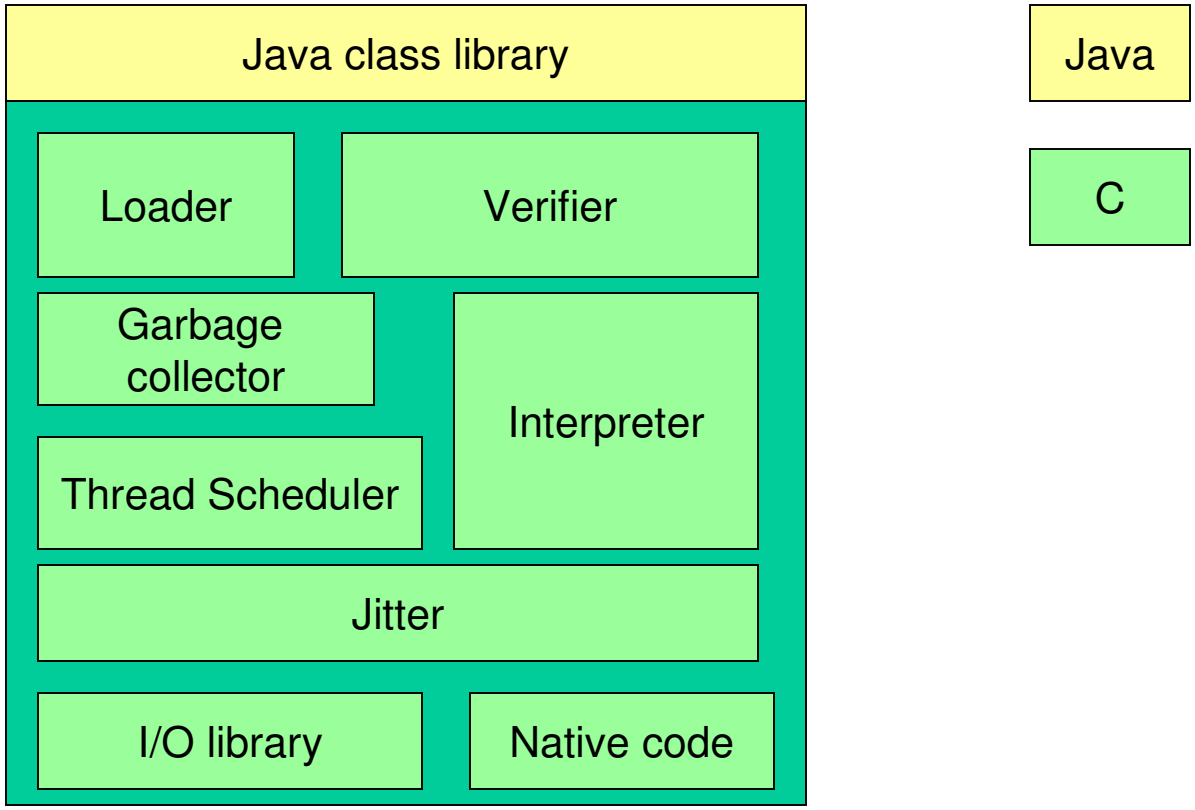


Conclusion:

C and compacting garbage collectors do not work well together.

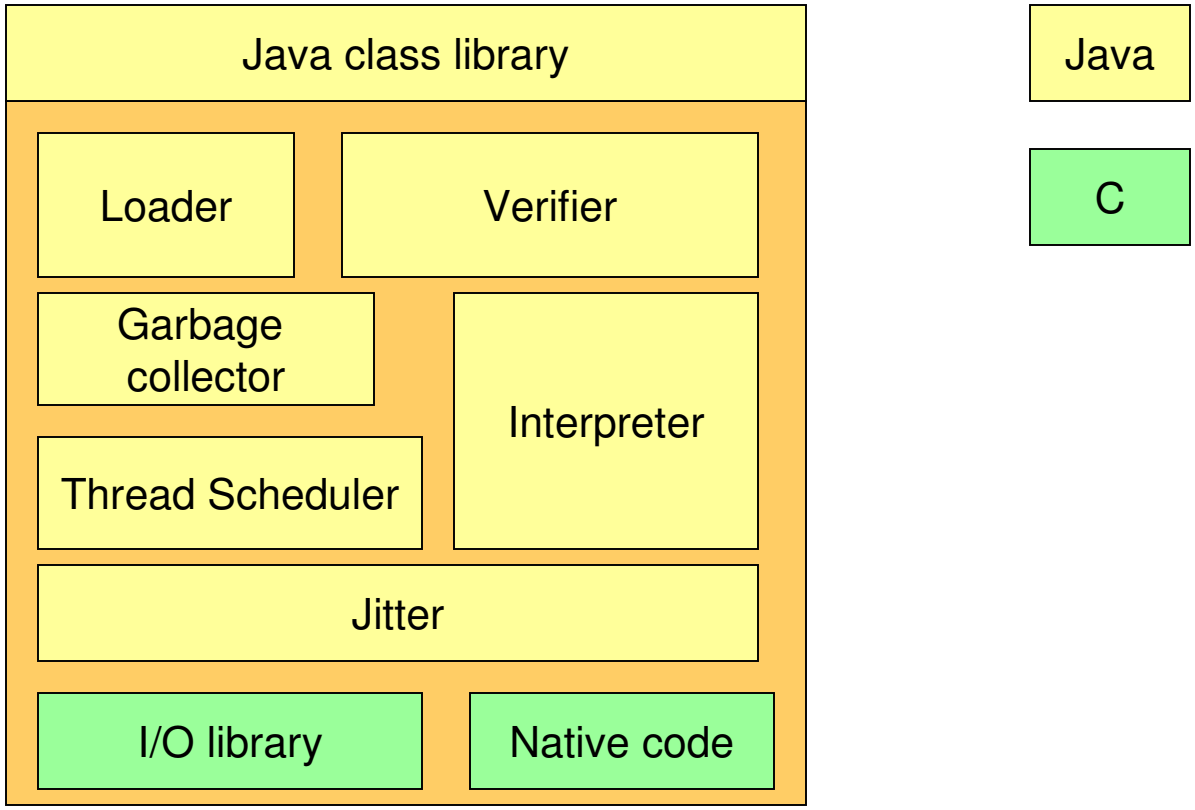
The Big Pointer Problem

A Standard JVM



The Big Pointer Problem

The Squawk JVM



The Squawk Solution

- Class Loader - A Java subroutine
- Verifier - A Java subroutine
- Interpreter - Machine code from compiler
- Garbage collector - Java compiled AOT
- Thread scheduler - Code in `java.lang.Thread`
- Dynamic compiler - A Java subroutine
- Dynamic assembler - A Java subroutine
- Native code - Native code

Suites

Suites are:

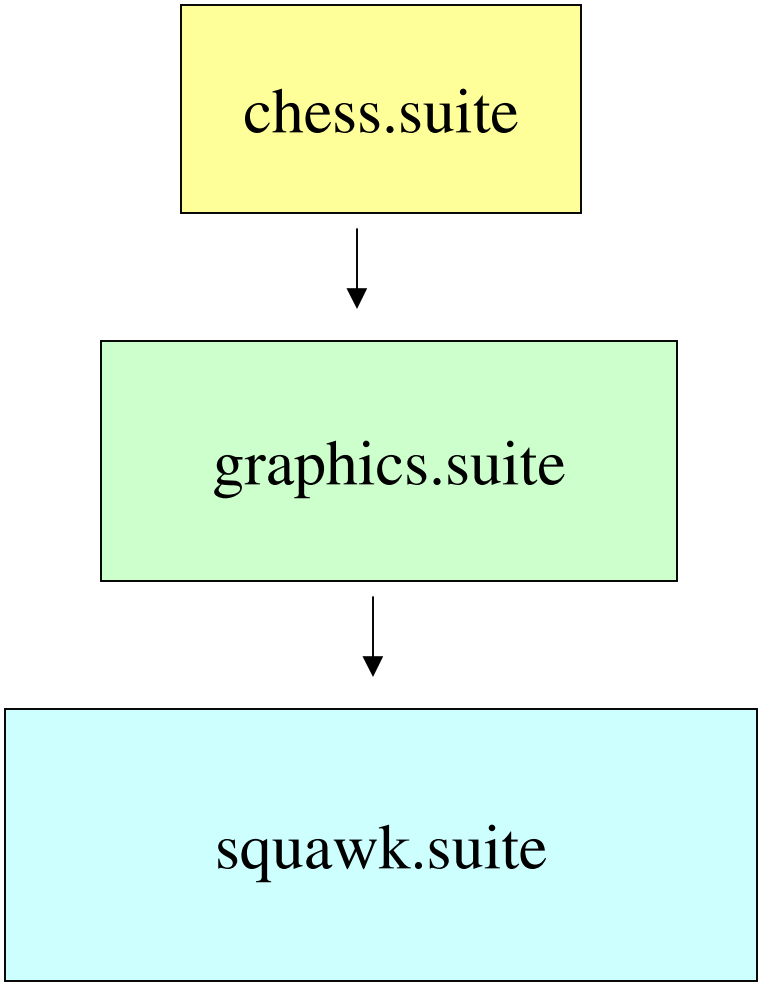
- Resolved collections of Java classes
- Squawk's internal object representation of Java software
- Are read-only data structures
- Enable extremely fast application start up
- Can be saved and restored

Suites

Application suite

Library suite

System suite



Isolates

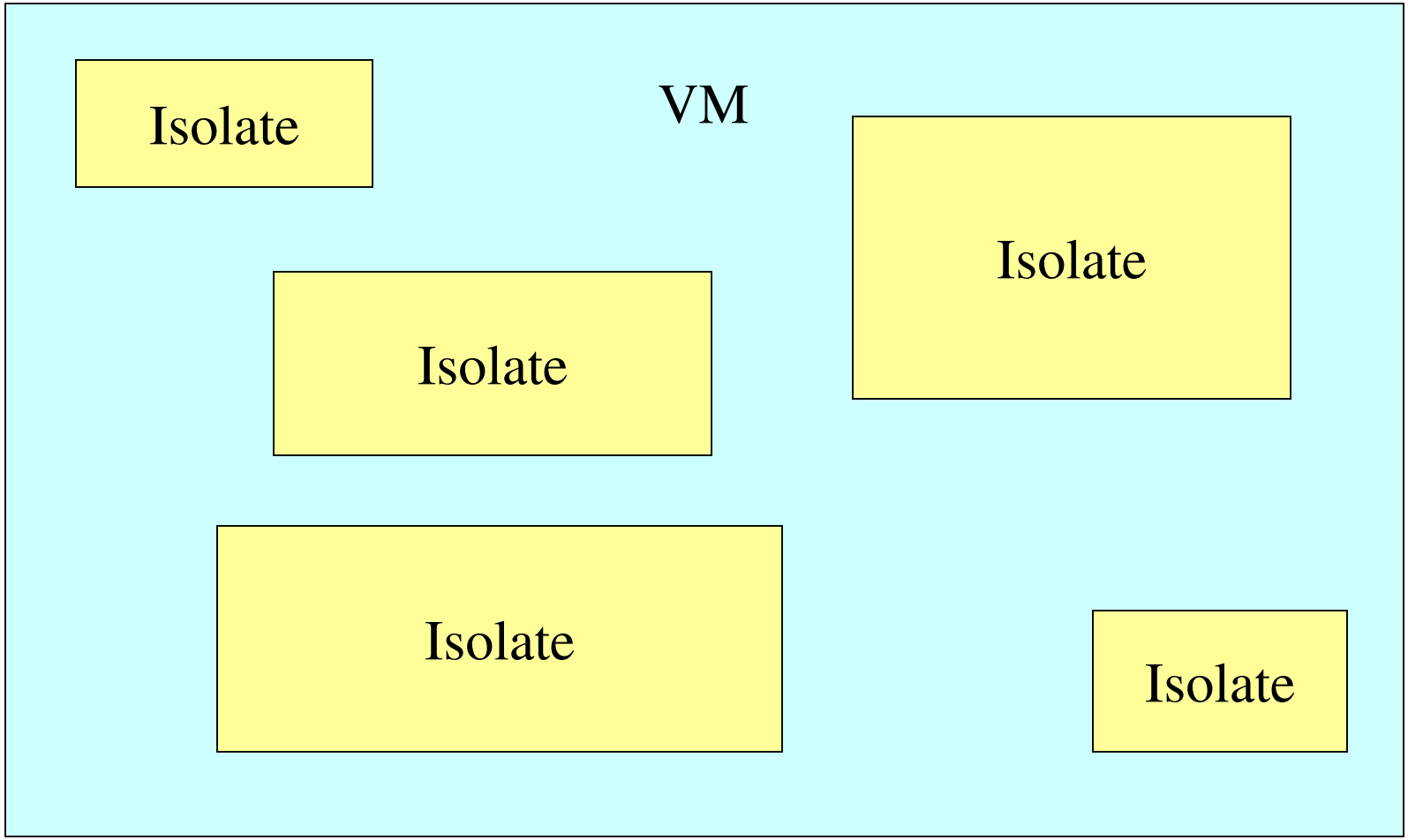
Are great because:

- They allow several Java applications to execute independently in a single JVM
- Isolate bugs
- Maximize sharing of VM resources
- Enable the VM to behave like an OS

Isolates in Squawk are:

- Implemented in Java
- Can be save and restored

Isolates



Demo

Q & A

nik.shaylor@sun.com