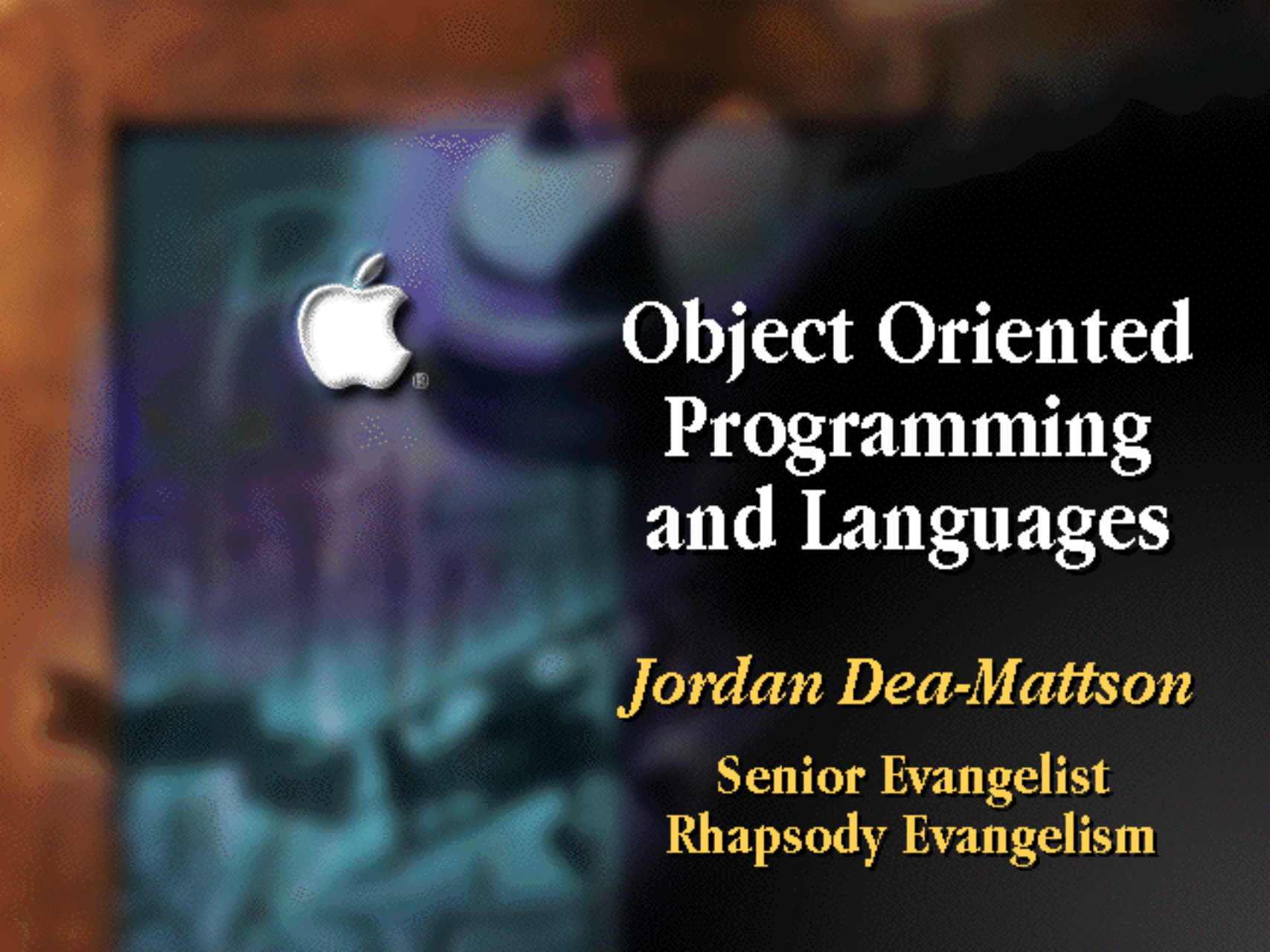
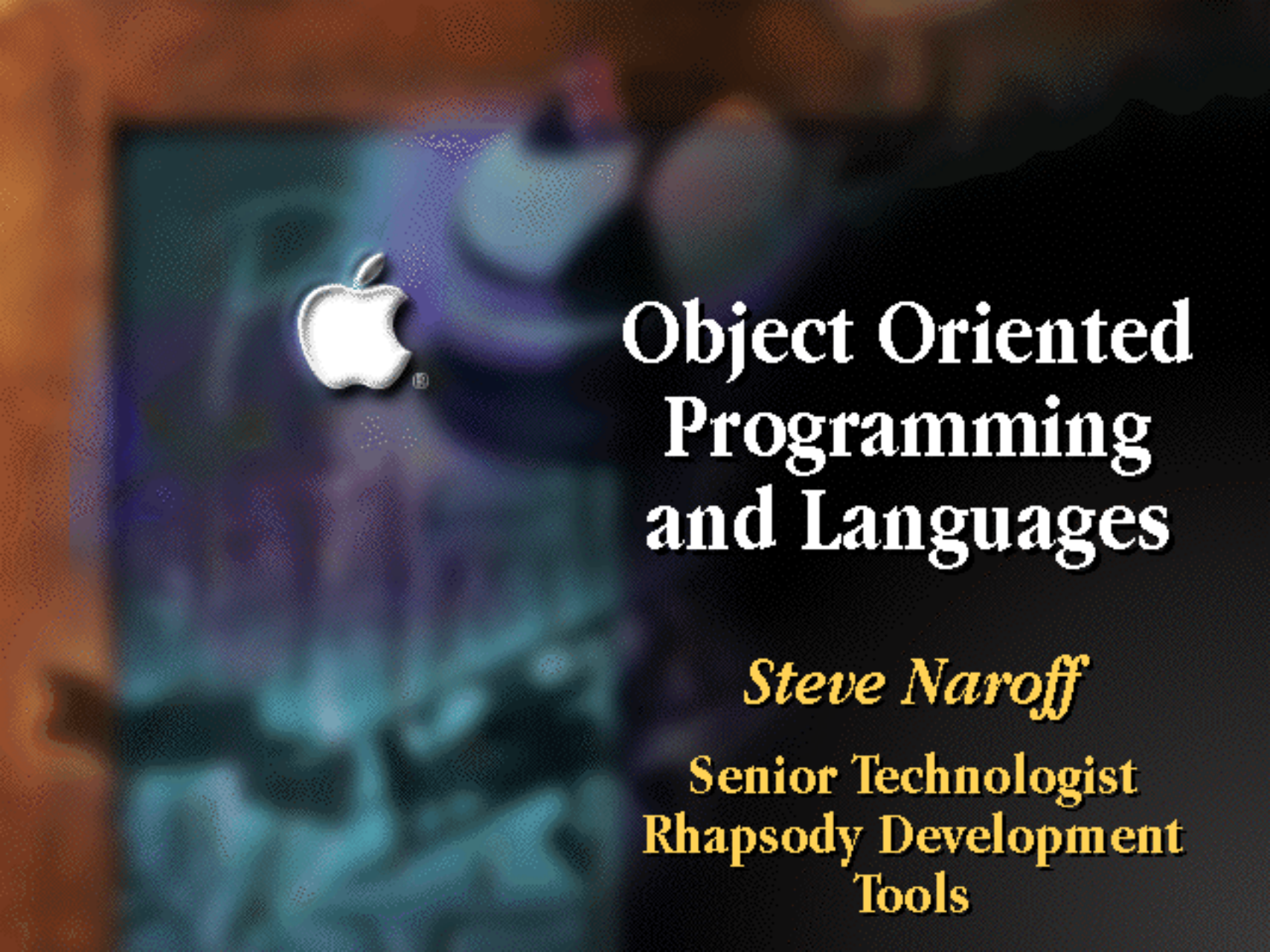

Worldwide
Developers
Conference


Object Oriented
Programming
and Languages
Jordan Dea-Mattson
Senior Evangelist
Rhapsody Evangelism



Object Oriented Programming and Languages

Steve Naroff

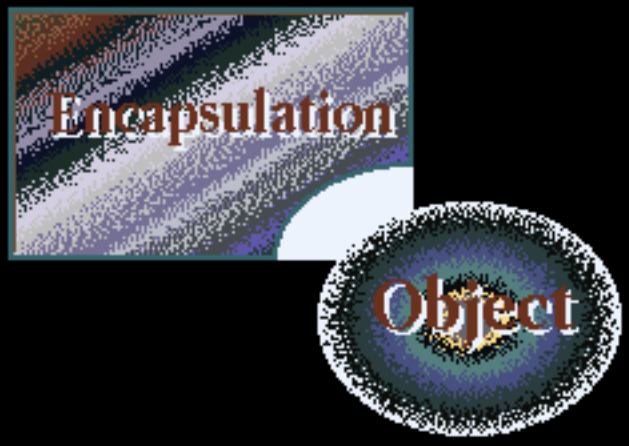
Senior Technologist
Rhapsody Development Tools

Session Topics

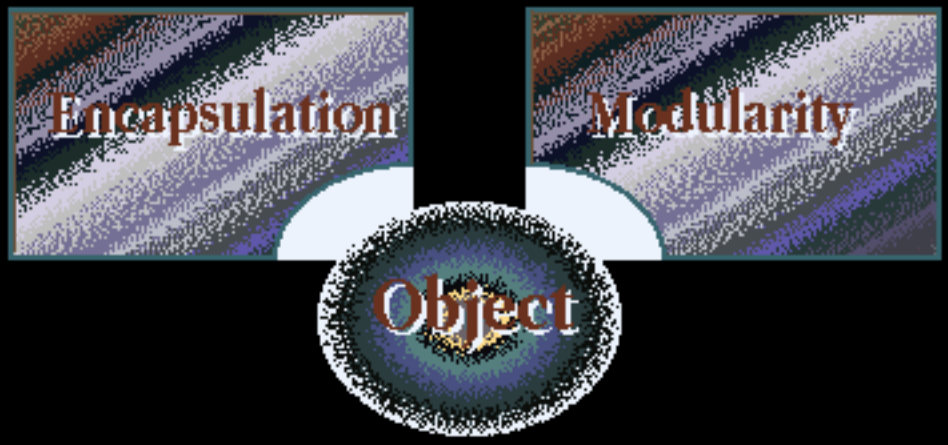
- Object Oriented Concepts
- Language Requirements
- Language Support
- Closing Thoughts

Object Oriented Concepts

Object Oriented Concepts

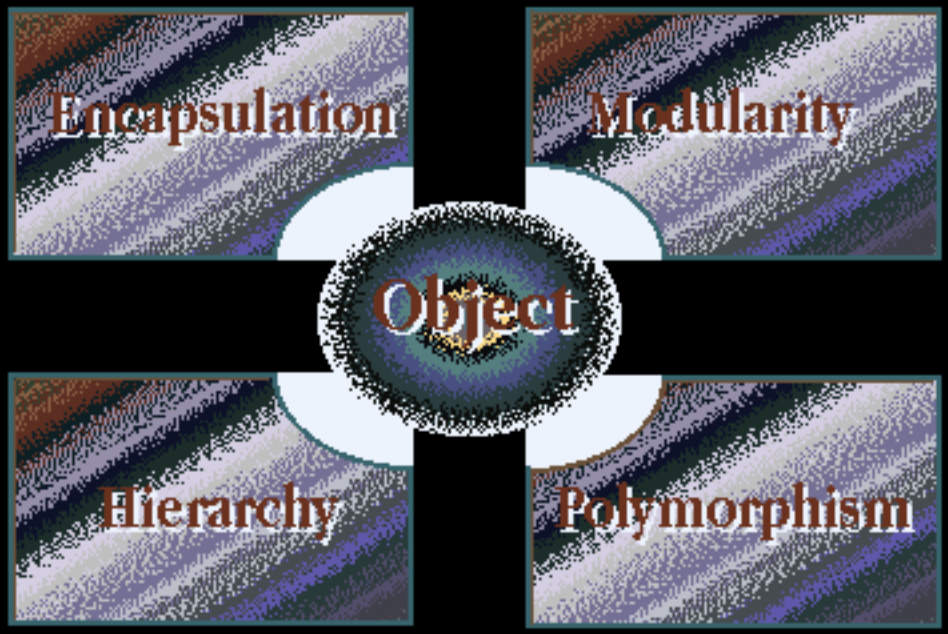


Object Oriented Concepts



Object Oriented Concepts

Object Oriented Concepts



Object Oriented Concepts

Language Requirements

Religion (circa '87)

- Dynamic Object Model
  - Serve application frameworks and tools

Language Requirements

Religion (circa '87)

- Dynamic Object Model
  - Serve application frameworks and tools
- Easy to Learn
  - "Less is more" principle

Language Requirements

Religion (circa '87)

- Dynamic Object Model
  - Serve application frameworks and tools
- Easy to Learn
  - "Less is more" principle
- Small footprint
  - Code and meta-data must be shared

Language Requirements

Religion (circa '87)

- Dynamic Object Model
  - Serve application frameworks and tools
- Easy to Learn
  - "Less is more" principle
- Small footprint
  - Code and meta-data must be shared
- Smooth integration with ANSI-C/C++
  - Embrace legacy code

Language Requirements

Religion (circa '87)

- Dynamic Object Model
  - Serve application frameworks and tools
- Easy to Learn
  - "Less is more" principle
- Small footprint
  - Code and meta-data must be shared
- Smooth integration with ANSI-C/C++
  - Embrace legacy code

Dynamic Object Model

Distinctive features

- Introspection
  - All objects are self-describing

Dynamic Object Model

Distinctive features

- Introspection
  - All objects are self-describing
- Release-to-release binary compatibility
  - No need to relink client applications

Dynamic Object Model

Distinctive features

- Introspection
  - All objects are self-describing
- Release-to-release binary compatibility
  - No need to relink client applications
- Message forwarding
  - Integration with other object models

Dynamic Object Model

Distinctive features

- Introspection
  - All objects are self-describing
- Release-to-release binary compatibility
  - No need to relink client applications
- Message forwarding
  - Integration with other object models
- C-based API/ABI fully specified
  - CodeWarrior modules link painlessly!

Dynamic Object Model

Distinctive features

- Introspection
  - All objects are self-describing
- Release-to-release binary compatibility
  - No need to relink client applications
- Message forwarding
  - Integration with other object models
- C-based API/ABI fully specified
  - CodeWarrior modules link painlessly!

Language Support

- Objective-C
- Objective-C++
- Java
- Dylan?
- Object Pascal?

Objective-C Type Declarations

Classes — implementation reuse

```
@interface NSBrowser : NSControl
{
   <instance varibles>
}

<methods>

@end
```

Objective-C Type Declarations

Protocols — design reuse

```
@protocol NSDraggingInfo

<methods>

@end
```

Objective-C Object Declarations

```
// untyped
id anyObject;

// typed by class
NSBrowser *myBrowser;

// typed by protocol
id <NSDraggingInfo> aDrag;
```

Objective-C Method Declarations

Classic syntax

```
// instance methods
- (void)setTitle:(NSString *)title
        ofColumn:(int)column;
- (void)setDelegate:anObject;
- (void)setAction:(SEL)action;

// class methods
+ (Class)cellClass;
+ (void)setCellClass:(Class)creator;
```

Objective-C Method Declarations

Modern syntax

```
// instance methods
void setTitleOfColumn(NSString *title,
                      int column);
void setDelegate(id anObject);
void setAction(SEL name);

// class methods
static Class cellClass();
static void setCellClass(Class creator);
```

Objective-C Method Categories

Binary extensions

```
@interface NSString(NSStringDrawing)

NSSize size()
void drawAtPoint(NSPoint p);
void drawInRect(NSRect r);

@end
```

Objective-C Message Expressions

```
// classic
[myBrowser setDelegate:obj];

// modern
myBrowser->setDelegate(obj);
myBrowser.setDelegate(obj);
```

Objective-C++

What role does it play in the Yellow Box?

- C++ access to Yellow Box APIs
  - Header files are "C++ aware"

Objective-C++

What role does it play in the Yellow Box?

- C++ access to Yellow Box APIs
  - Header files are "C++ aware"
- **Embrace the next generation(s) of C**
  - Legacy code

Objective-C++

What role does it play in the Yellow Box?

- C++ access to Yellow Box APIs
  - Header files are "C++ aware"
- Embrace the next generation(s) of C
  - Legacy code
- Objective-C able to access "C++ goodies"
  - Declarations as statements
  - Exceptions
  - Etc.

Objective-C++

What role does it play in the Yellow Box?

- C++ access to Yellow Box APIs
  - Header files are "C++ aware"
- Embrace the next generation(s) of C
  - Legacy code
- Objective-C able to access "C++ goodies"
  - Declarations as statements
  - Exceptions
  - Etc.

Objective-C++

Peaceful coexistence

- Both languages retain…
  - Native semantics
  - Native time/space characteristics

Objective-C++

Does CodeWarrior support it?

- YES!

Objective-C++ Example

Reuse via aggregation

```
@interface CalculatorInterface : NSObject
{
  id display;
  CalculatorEngine *engine;
}

id init();
id equalsKey(id sender);
id operationsKey(id sender);

@end
```

Objective-C++ Example

```
id init() // display is set automatically
{
  engine = new CalculatorEngine;
  return self;
}
id operationKeys(id sender)
{
  res = engine->computeResult(sender);
  display->setDoubleValue(res);
  return self;
}
```

Java Integration

Design points

- Exploit similarities with Objective-C

Java Integration

Design points

- Exploit similarities with Objective-C
- Work with standard Java language tools
  - No special language/compiler magic

Java Integration

Design points

- Exploit similarities with Objective-C
- Work with standard Java language tools
  - No special language/compiler magic
- Leverage "best of breed" JVM technology
  - Sun, Microsoft, Apple, Metrowerks, etc.

Java Integration

Design points

- Exploit similarities with Objective-C
- Work with standard Java language tools
  - No special language/compiler magic
- Leverage "best of breed" JVM technology
  - Sun, Microsoft, Apple, Metrowerks, etc.
- Allow hybrid interaction
  - Aggregation and subclassing

Java Integration

Design points

- Exploit similarities with Objective-C
- Work with standard Java language tools
  - No special language/compiler magic
- Leverage "best of breed" JVM technology
  - Sun, Microsoft, Apple, Metrowerks, etc.
- Allow hybrid interaction
  - Aggregation and subclassing

Java Integration

Full client access to Yellow Box APIs

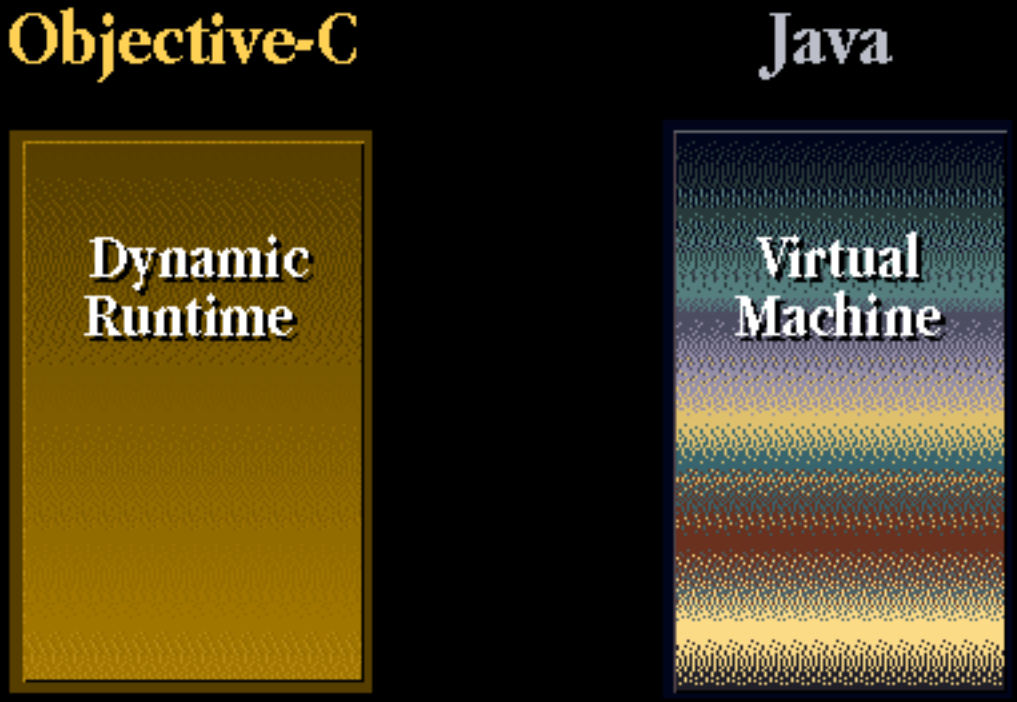
```
public class Browser extends Control
{
public Browser(Rect frameRect);

public native void setDelegate(Object o);
public native void setAction(Selector s);
public native boolean setPath(String p);
public native Array selectedCells();
}
```

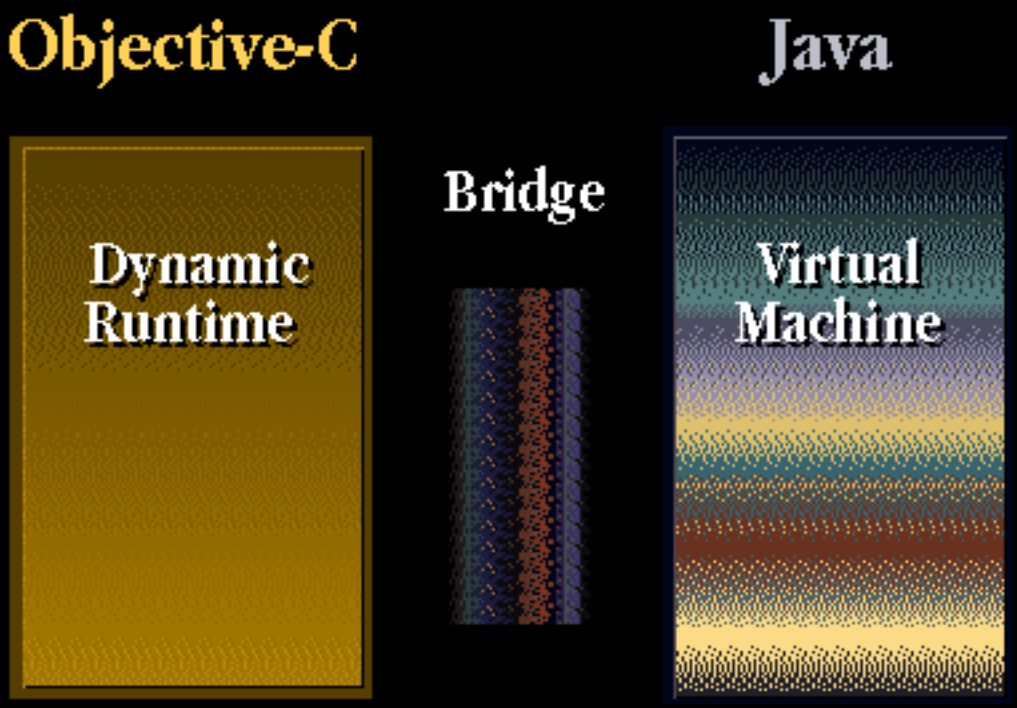
Java Integration

Objective-C

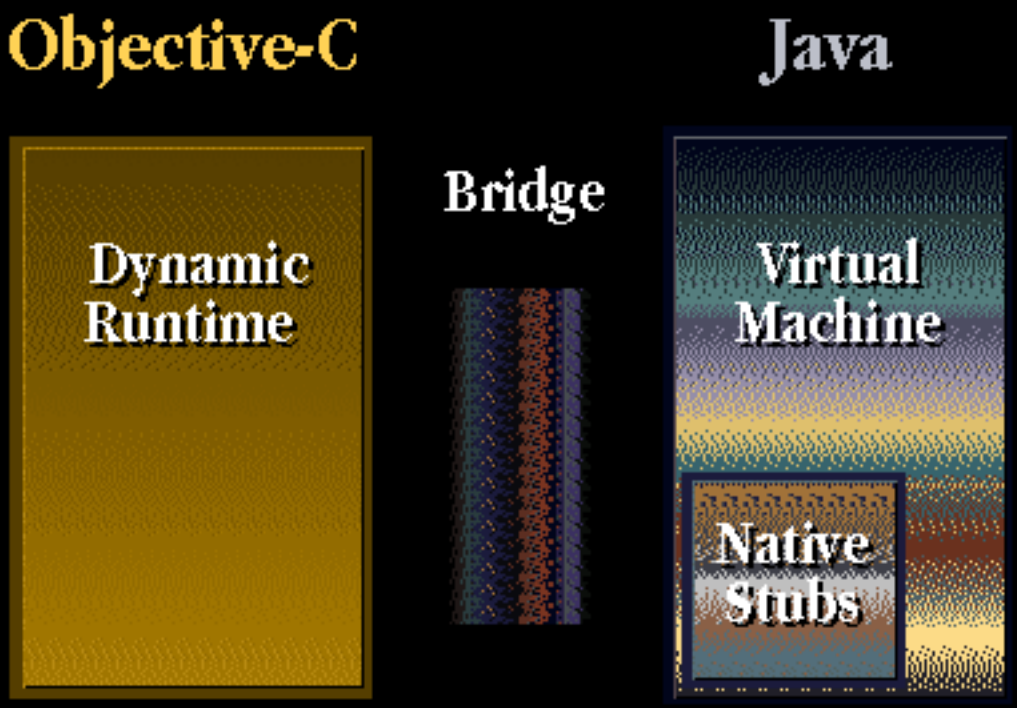
Java Integration



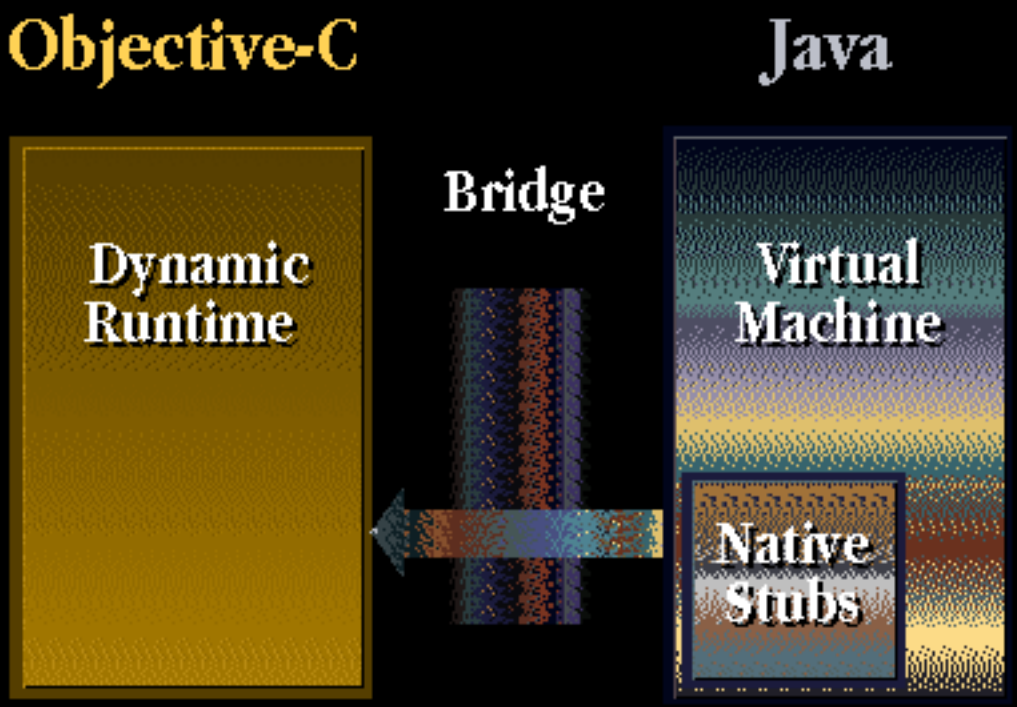
Java Integration



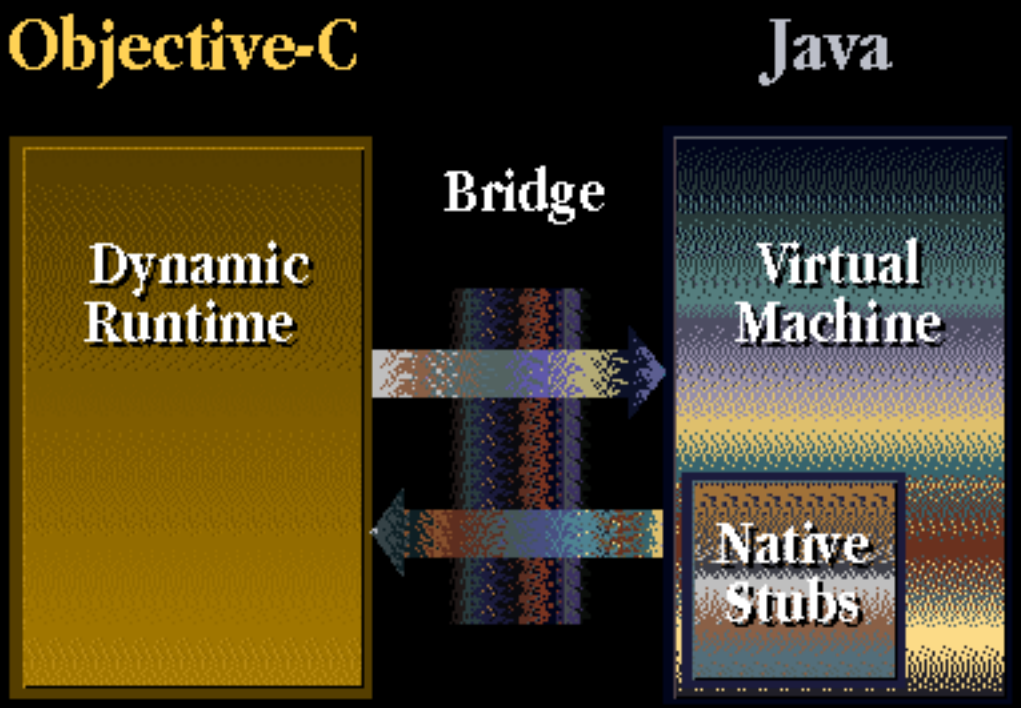
Java Integration



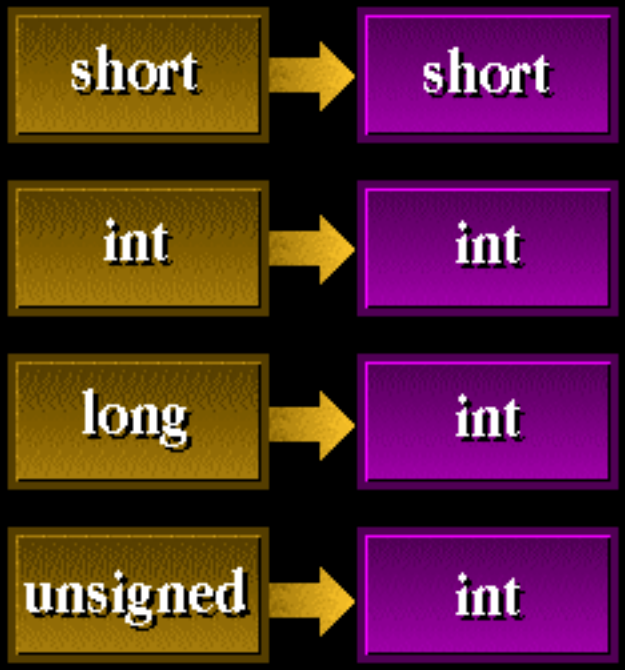
Java Integration



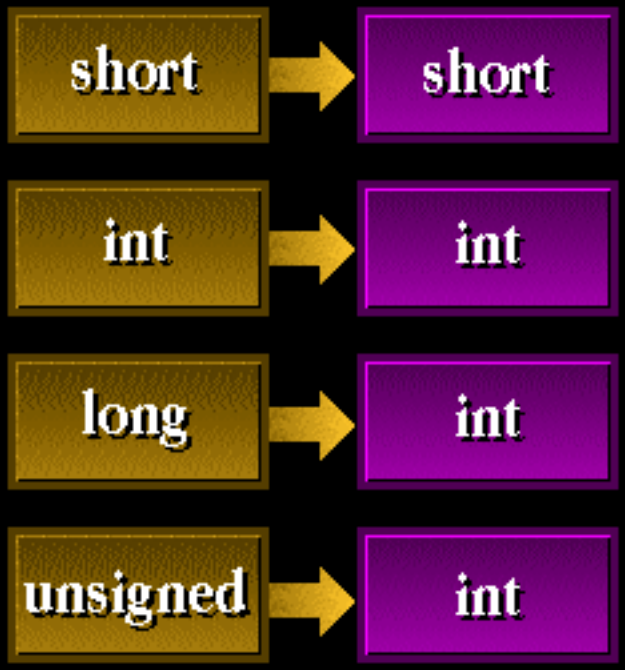
Java Integration

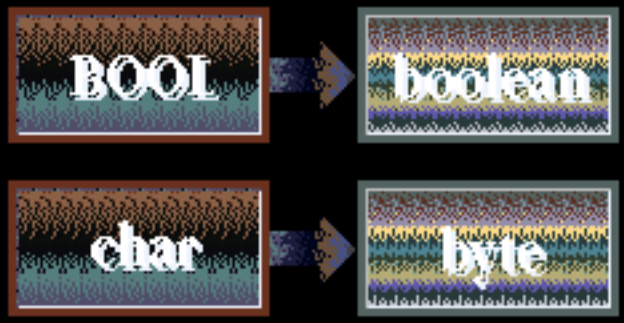


Java Mappings

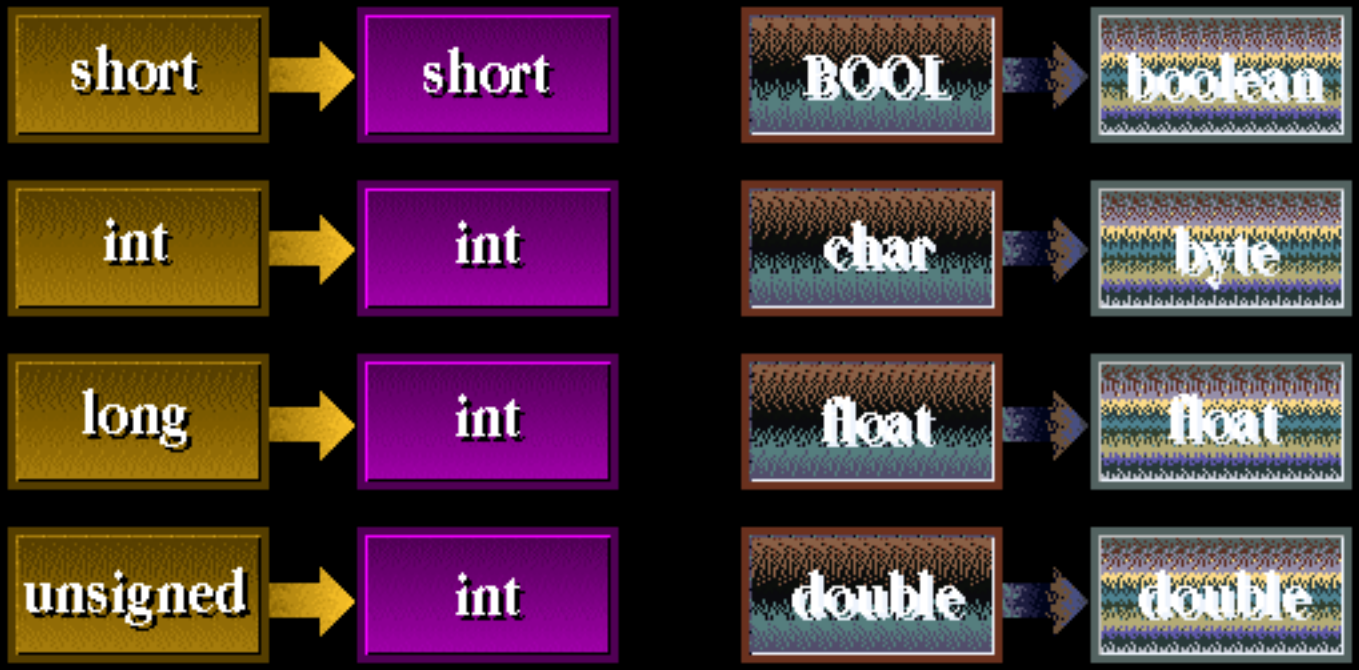


Java Mappings



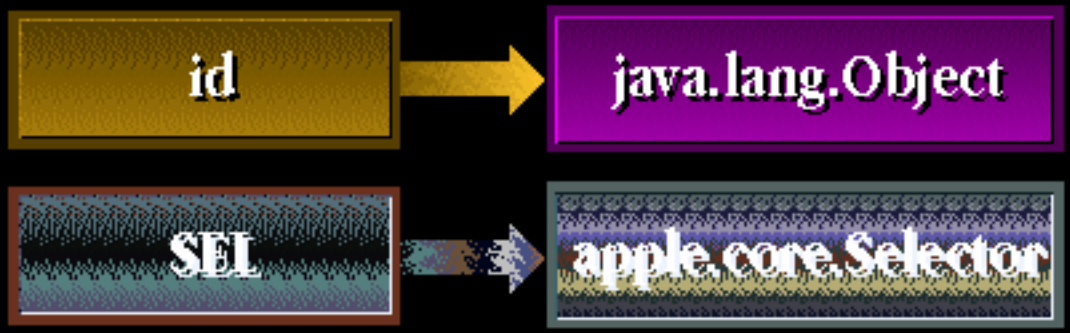


Java Mappings

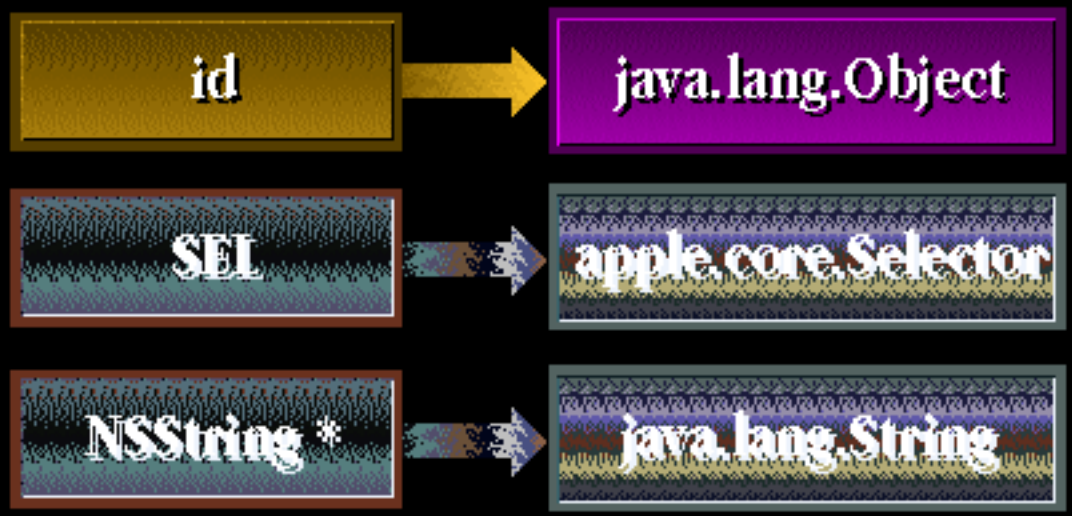


Java Mappings

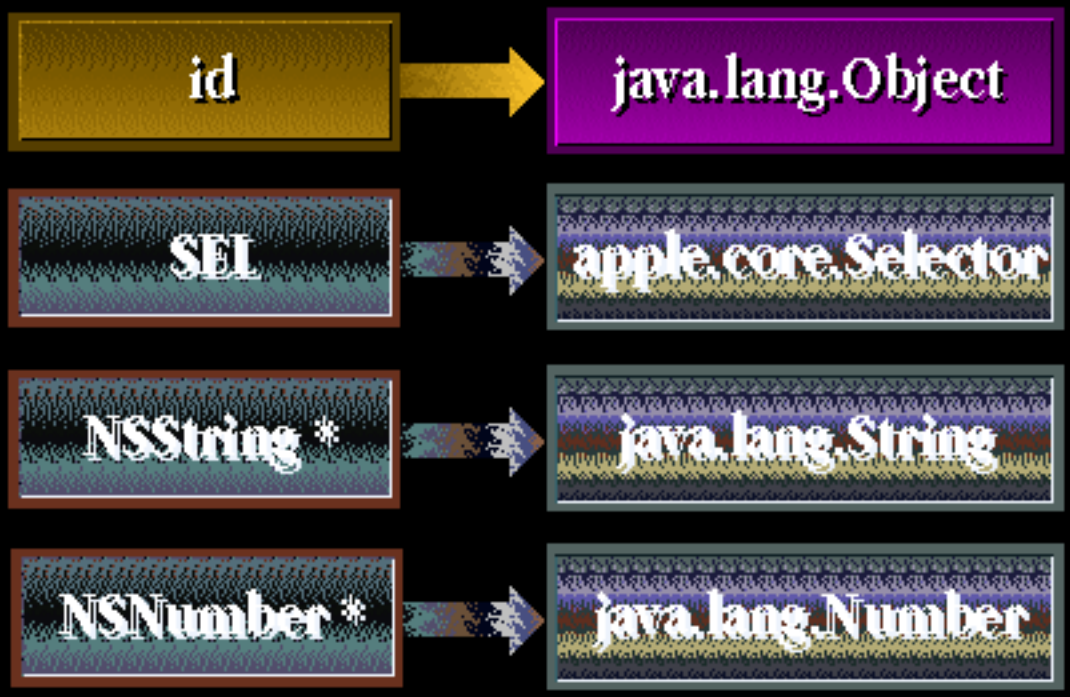
Java Mappings



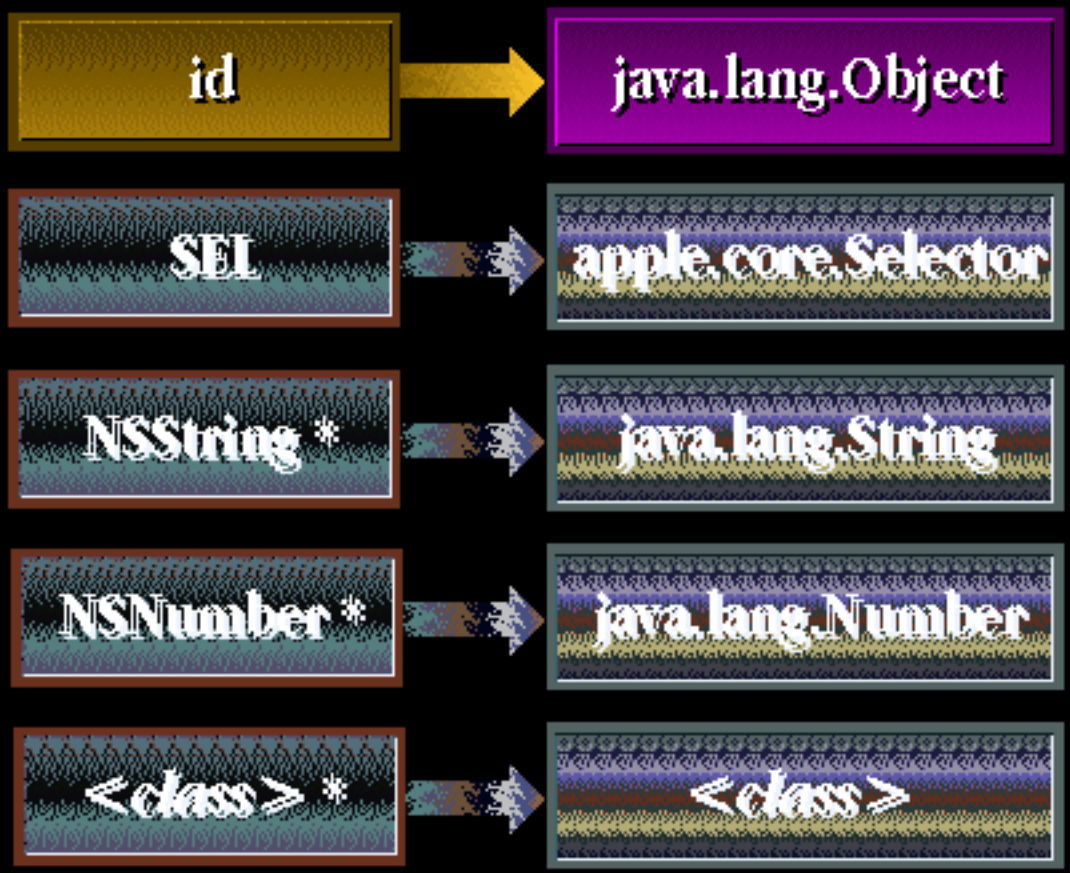
Java Mappings



Java Mappings



Java Mappings



Java Integration

How does it work?

- Types that don't map automatically
  - Non-object pointer usage
  - Structures, unions, and enums
  - Typedefs

Java Integration

How does it work?

- Types that don't map automatically
  - Non-object pointer usage
  - Structures, unions, and enums
  - Typedefs
- Java enabled Objective-C objects
  - Descendents of apple.core.id
  - Subclassing "just works"

Java Integration

How does it work?

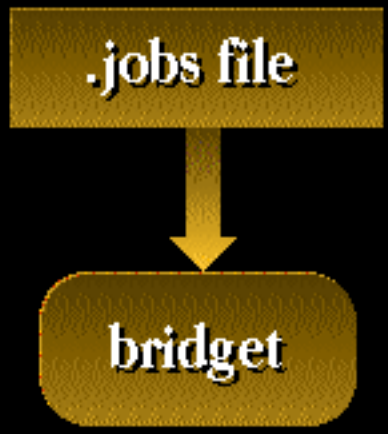
- Types that don't map automatically
  - Non-object pointer usage
  - Structures, unions, and enums
  - Typedefs
- Java enabled Objective-C objects
  - Descendents of apple.core.id
  - Subclassing "just works"

Java to ObjC Bridge Specification

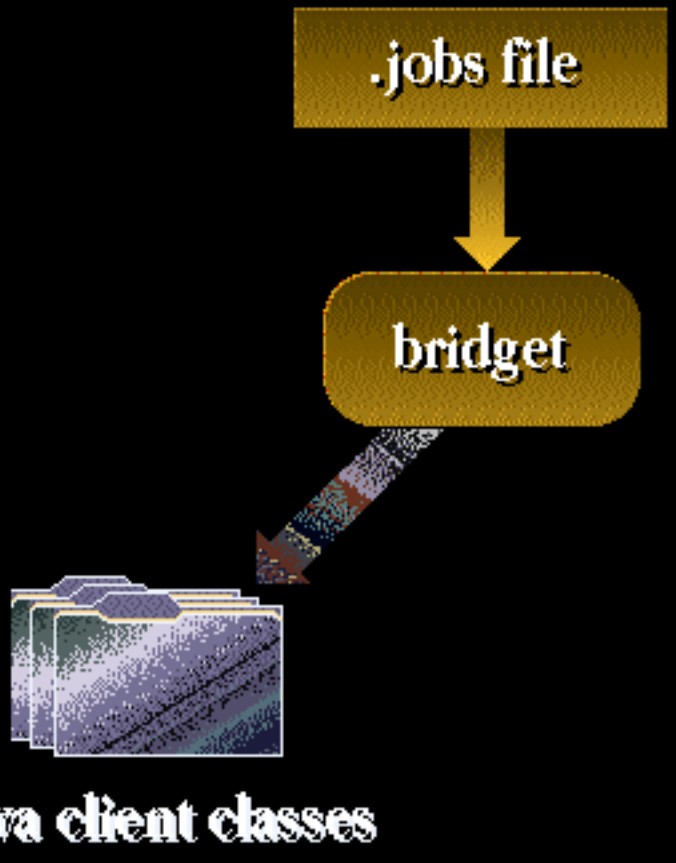
.jobs file

```
type
   BOOL = boolean
   NSRect = apple.core.Rect using
            NSConvertRectToJava
            NSConvertRectFromJava
selector
   -isEqual: = equals
   -hash = hashCode
   -copyWithZone: = clone
```

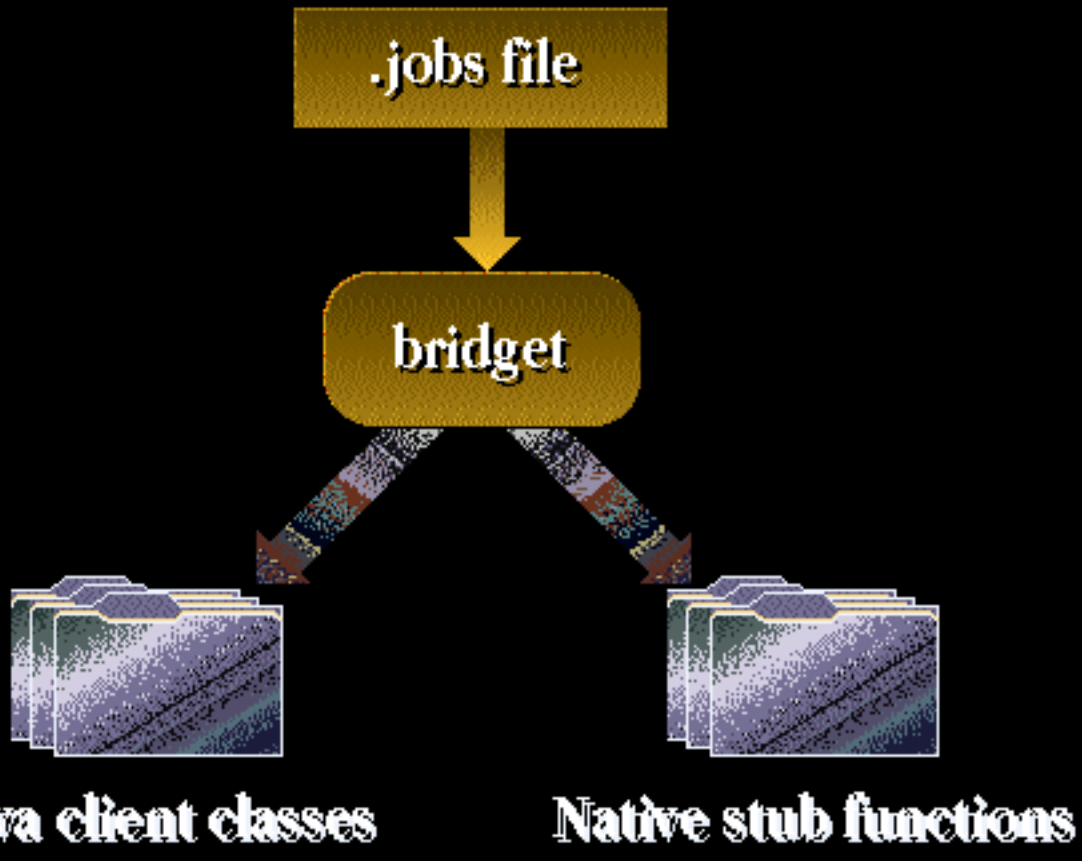
Java to ObjC Bridge Specification



Java to ObjC Bridge Specification



Java to ObjC Bridge Specification



Closing Thoughts

Language choices

- Dynamic
  - Over static

Closing Thoughts

Language choices

- Dynamic
  - Over static
- Diversity
  - Over purity

Closing Thoughts

Language choices

- Dynamic
  - Over static
- Diversity
  - Over purity
- Performance
  - Over purity

Closing Thoughts

Language choices

- Dynamic
  - Over static
- Diversity
  - Over purity
- Performance
  - Over purity
- Pragmatic solutions

Closing Thoughts

Language choices

- Dynamic
  - Over static
- Diversity
  - Over purity
- Performance
  - Over purity
- Pragmatic solutions


Q&A


Worldwide
Developers
Conference