Habanero Extreme Scale Software Research Project
Comp215: Evolution of Java

Zoran Budimlić (Rice University)
The Beginnings

Sun Microsystems

1990 - Create a language for delivering programs on small electronic devices

1991 - 1993 - James Gosling and Patrick Naughton form a “Green Team”, developing a new language called Oak for this purpose

1993 - Interest in set-top boxes, interactive TVs and PDAs started to fade

NCSA creates Mosaic, the first web browser. Start of the Internet boom

The Green Team renames itself into a project “First Person”, and refocuses the efforts on a language that would allow programs (“applets”) to run within a web browser

They call it Java
Welcome to NCSA Mosaic, an Internet information browser. Mosaic was developed at the National Center for Supercomputing Applications of the University of Illinois in Urbana-Champaign and is supported by the Board of Trustees of the University of Illinois.
Java Language Principles

Familiar syntax - very similar to C and C++
Simplify - throw away everything from C++ that wasn’t needed
No pointers - everything is accessed through a “reference”
No user operator overloading
No preprocessor
No explicit memory allocation/deallocation - built-in garbage collector instead
No global functions - everything is in a class
Single inheritance. Multiple inheritance only through interfaces
  Avoid the C++ multiple inheritance nightmare
“Java Is C++ Without The Guns, Knives, And Clubs”
- James Gosling
Two-stage Approach

Java programs are compiled into a portable, machine-independent “bytecode”

- Stack machine approach
- Standardized types (32-bit int, 64-bit int, IEEE floats, e.t.c)
- Portable - Write Once Run Anywhere (WORA)

Bytecode is executed on a Java Virtual Machine

- Bytecode is first verified for malicious behavior
- All objects are allocated on the heap, not on stack
- Initially only interpreted
What Java left out from C++

No stack objects, only heap objects
No destructors, only finalize() method
No pointers, everything is a reference
No delete, garbage collector instead
No const, only final (methods, fields, classes)
No templates, no preprocessor
No operator overloading
No multiple inheritance of classes
No enumerations or typedefs
What Java added that C++ didn’t have

Garbage collector
Object-rooted, rich class hierarchy (well, almost)
Strings, first-class arrays with bounds checking
Package system with import
interface, implements, extends, abstract
finally blocks, static/instance initializers
Secure and portable JavaVM, threads
Dynamic reflection capabilities
JavaDoc code documentation system
1996

JDK (Java Development Kit) 1.0 was released
javac - Java compiler with a Recursive Descent Parser
jvm - A virtual machine with a bytecode interpreter

Compiler and runtime for Solaris, Linux, Windows and Mac OS

Java becomes an instant hit with programmers tired of C++, yearning for a “true” object-oriented language

Sun creates JavaSoft, a company dedicated solely to Java development
About 30-40 employees in the summer of 1996
1997

JDK (Java Development Kit) 1.1 was released
  AWT with event-driven programming model
  inner classes
  reflection (introspection only)
  Just In Time (JIT) Compilation (outsourced to Symantec)
JavaSoft forms the HotSpot group to work on the in-house JIT
  Significant Rice presence
JavaSoft grows to about 300 employees in the Summer of 1997
Microsoft releases the Microsoft JVM
  Faster than Sun’s implementation
  Sun sues Microsoft in October for “incompletely implementing” Java
Java is the #2 programming language in the world, with ~.5M developers
1997

HotJava browser

- A web browser fully implemented in Java
- Support for executing Java applets
- AWT GUI
- Not really competitive with the browsers of the day
Java Inner Classes

An outer class can contain an inner class:

```java
public class OuterClass {
    ...
    class InnerClass {
        ...
    }
}
```

Inner classes can be declared inside of a method

Inner classes can be anonymous

```java
public void sayHello() {
    HelloWorld frenchGreeting = new HelloWorld() {
        String name = "tout le monde";
        public void greet() {
            greetSomeone("tout le monde");
        }
        public void greetSomeone(String someone) {
            name = someone;
            System.out.println("Salut " + name);
        }
    };
```
Java Grande

An academic forum aimed at using Java to address “Grande” challenges

Commercial: Seismic processing, oil reservoir simulation, vehicle simulations

Government: Satellite image processing, climate and weather, nuclear weapon simulations

Academic: Particle physics, biochemistry, earthquake prediction

Cryptography

Addressed Java shortcomings in concurrency and numerics

Reproducibility of Java executions

IEEE standard floating-point behavior

Relaxed with Java 1.2 standard
1998

Java 2 (version 1.2) released by Sun
   New event handling
   JIT plug-in support

IBM introduces the Jikes RVM Java virtual machine
   written completely in Java!
   still beats both Microsoft and Sun's implementations of Java

Symantec Visual Cafe (an IDE for Java) is the leading platform for
developing Java
2000

Java 1.3 (Kestrel) released by Sun
  HotSpot JVM
  Plug-in debugging now supported
Microsoft releases C#, which “borrows” a lot of features from Java

“C# is sort of Java with reliability, productivity and security deleted.”

- James Gosling
2002

Java 1.4 (Merlin) released by Sun

- Assertions
- Regular expressions
- XML support
- Cryptography and SSL
- Logging
2004

Java 1.5 (Tiger) released by Sun

- Generics!
- Autoboxing/unboxing of primitive types
- Static imports
- Annotations
- JVM libraries shared among JVM instances
Java Generics

So much more intuitive

```java
IList<String> emptyList = List.Empty.create();
IList<String> list = emptyList.add("Alice").add("Bob").add("Charlie");
```

Java uses “type erasure” at compile time to get rid of generic type parameters

The parametric types are converted to a super type
Casts are inserted
Both `emptyList` and `list` above will be lists of Object
Problems with type erasure

// create an array of objects
Object[] objects = new Object[10];

// cast it to an array of strings
String[] strings = (String[])objects;

The second line will compile, but it will throw a runtime exception, even if all the elements of the objects array are instances of the String class.
Problems with type erasure

// create an array of strings
String[] strings = new String[10];

// cast it to an array of objects
Object[] objects = strings;

// insert an object into the array
objects[0] = new Object();

Similar code with generic Vectors:

// create a vector of strings
Vector<String> strings = new Vector<String>(10);

// cast it to a vector of objects
Vector<Object> objects = (Vector<Object>)strings;

// insert an object into the vector
objects.add(new Object());
2006

Java 1.6 (Mustang, Java SE 6) released by Sun

- Web services
- Mixing with JavaScript
- Compiler access (enables tools like IntelliJ)
- Pluggable Annotations
- Security improvements
Started in 2002 at Yahoo
Now Open Source as part of the Apache Software Foundation
Google MapReduce programming model
   Decompose your programs into “maps” and “folds”
   Sound familiar?
Java virtual machines on all nodes
Hadoop Distributed file system (HDFS)
Yahoo! and Facebook
More than half of Fortune 50 companies use Hadoop
Remember these guys?
2011

Java 1.7 (Dolphin, Java SE 7) released by Oracle

- Support for dynamic languages
- Mixing with JavaScript
- Compiler access (enables tools like IntelliJ)
- Pluggable Annotations
- Security improvements
2014

Java 1.8 (Java SE 8) released by Oracle

- Lambdas! Yay!
  - Use **invokedynamic** introduced in Java 7
- Type inference for generics
- Functional interfaces
- Type annotations
- Parallel operations
- Java + JavaScript = ❤
- New Date/Time API
- Concurrent accumulators
Languages targeting JVM

ColdFusion - web development
JRuby and Jython - scripting languages based on Ruby and Python
Groovy - scripting language based on Java
Scala - functional and object-oriented programming language
JGNAT and AppletMagic - compile from Ada to Java bytecode
C - using C to Java byte code compilers
Clojure - functional language based on Lisp
MIDletPascal - ah, Pascal
Kotlin - statically typed, object-oriented