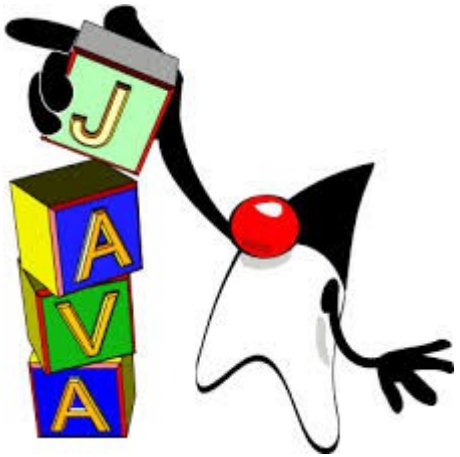
Advanced Programming
Introduction

Course Description

- The Goal
- The Motivation
- Lectures and Assignments
- Programming Platform
- Resources
- Evaluation



Lab: *problems, projects, essays* → easy

Exam: *written test* → hard

What exactly is “Java”?

- Programming Language
- Programming **Platform**
- 1995
- Sun Microsystems / **Oracle** (2010)
- James Gosling
- Duke

Why Java?


Java Setup - Progress
Java
ORACLE
Status: Registering Java Runtime Environment
3 Billion Devices Run Java
Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...
ORACLE

Java Programming Language

- **Simplicity**

  “as simple as possible, but not simpler”
- **Robustness**: ~~pointers~~, automatic memory management, garbage collection, strong typing
- Completely **object-oriented**
- **Secure** class loading and verification
- **Architecture Neutrality**
- **Portability**
- **Performance**

WORA
Write once, run anywhere

Java Platforms

- **Java SE (Standard Edition)**

  Desktop applications, applets, Java Web Start, JavaFX

- **Java EE (Enterprise Edition)**

  Complex, distributed, large scale, applications; server-side components, Web Services, etc.

- **Java ME (Micro Edition)**

  Programming embedded systems, mobile devices, TVs, GPSs, etc.

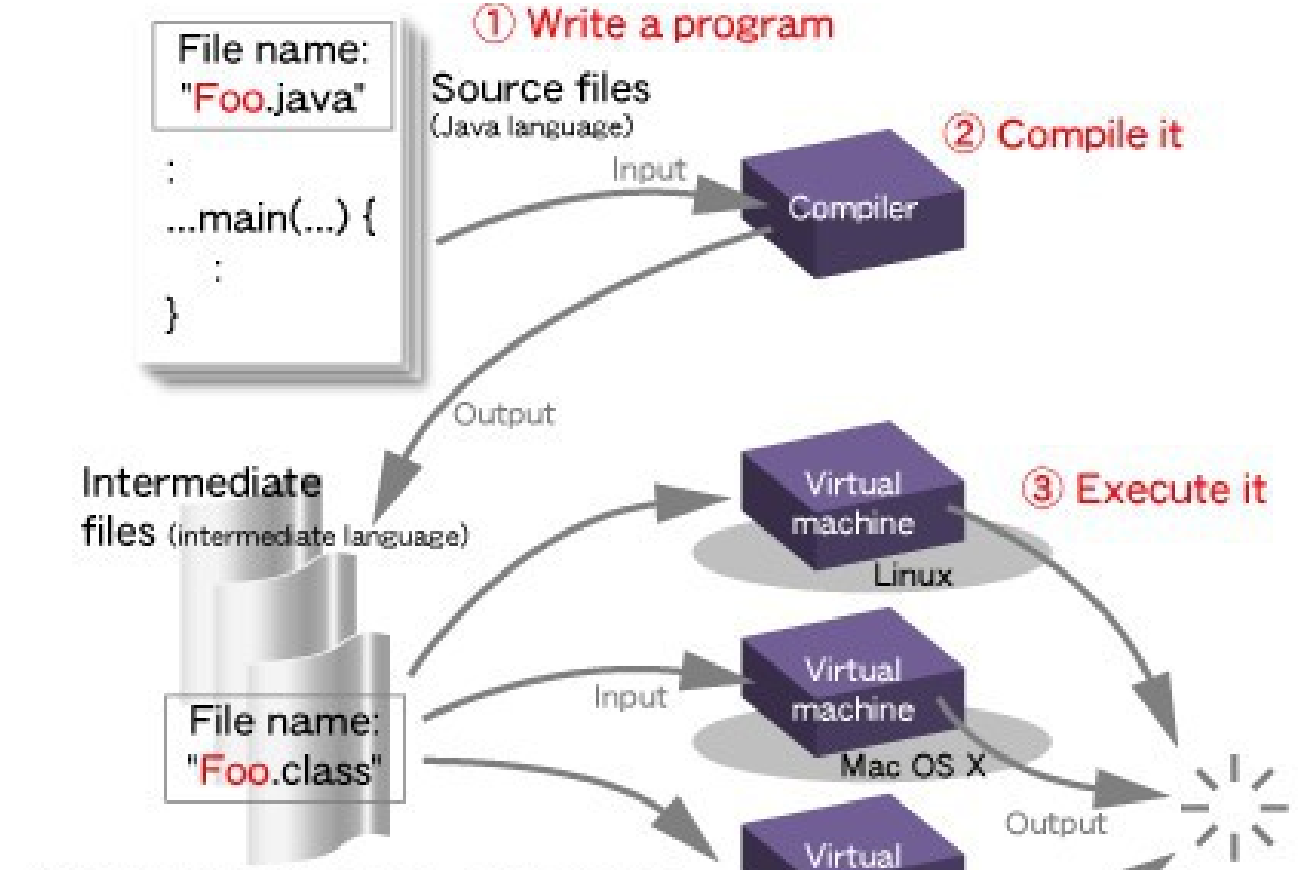
- **Java Card**

Compiled and Interpreted

- **Interpreted languages**
  - simplicity, portability
  - low execution speed
- **Compiled languages**
  - high execution speed
  - no portability
- **Java: compiled *and* interpreted**

  The Java compiler doesn't generate "machine code" (native hardware instructions). Rather, it generates **bytecodes**: a high-level, machine-independent code for a hypothetical machine that is implemented by the Java interpreter and run-time system.

Java Virtual Machine (JVM)



- **Actually in many cases, a program is distributed in the form of a JAR file, which stands for Java ARchive file.**

The First Program

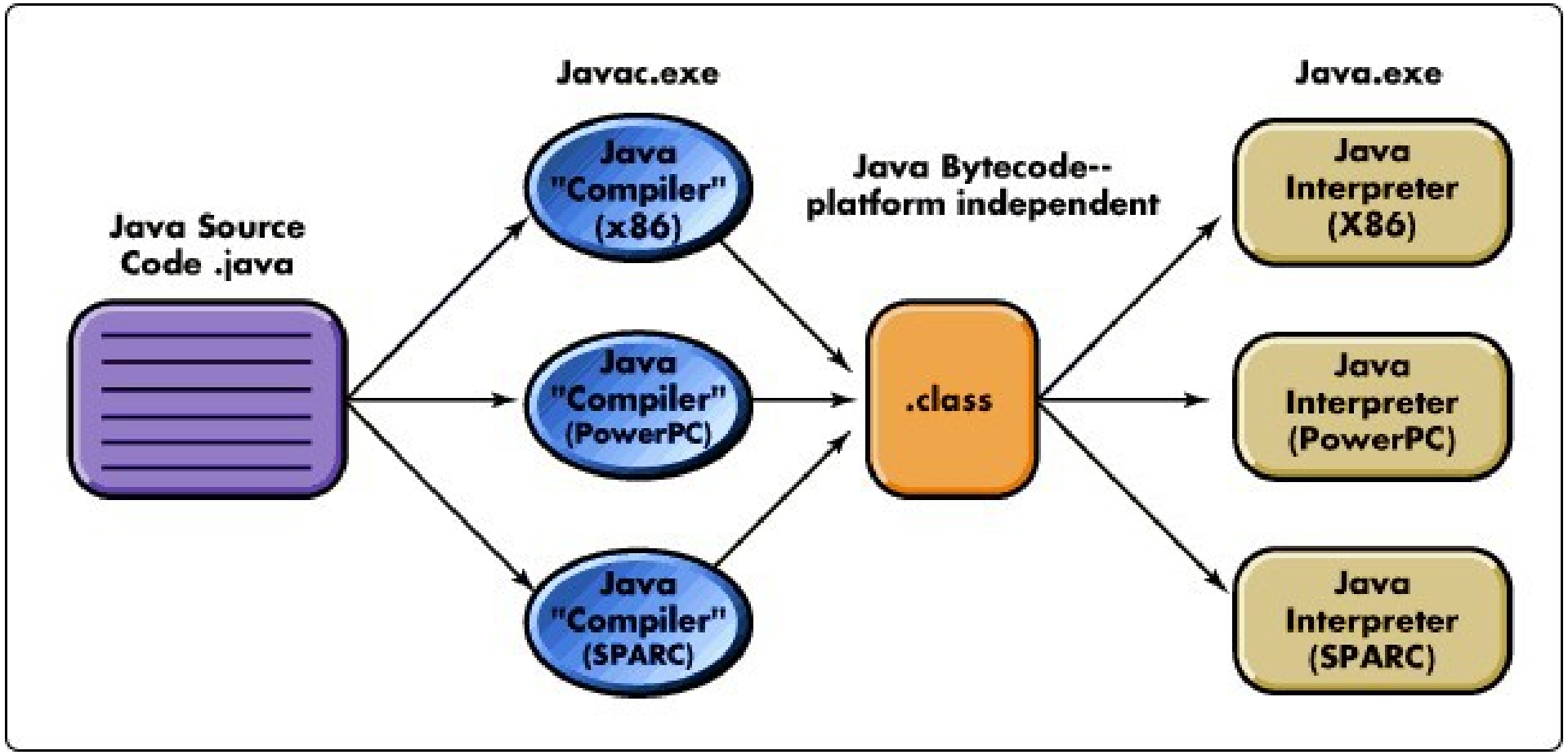
```
public class HelloWorld {

 public static void main(String args[]) {
   System.out.println("Hello world!");
 }
}
```

- *Source*: `HelloWorld.java`
- *Compile*

  `javac HelloWorld.java` → `HelloWorld.class`
- *Run*

  `java HelloWorld`

java, javac


Javac.exe
Java.exe
Java Source Code .java
Java "Compiler" (x86)
Java "Compiler" (PowerPC)
Java "Compiler" (SPARC)
Java Bytecode-- platform independent
.class
Java Interpreter (X86)
Java Interpreter (PowerPC)
Java Interpreter (SPARC)

javap

javap -c HelloWorld

```
Compiled from "HelloWorld.java"
public class HelloWorld extends java.lang.Object{
HelloWorld();
  Code:
   0: aload_0
   1: invokespecial #1;
      //Method java/lang/Object."<init>":()V
   4: return

public static void main(java.lang.String[]);
  Code:
   0: getstatic  #2;
      //Field java/lang/System.out:Ljava/io/PrintStream;
   3: ldc #3;
      //String Hello world!
   5: invokevirtual  #4;
      //Method java/io/PrintStream.println:(Ljava/lang/String;)V
   8: return
}
```

Obfuscation

UNICODE

“Without Unicode, Java wouldn’t be Java, and the Internet would have a harder time connecting the people of the world.”
James Gosling, Inventor of Java

- Character encoding system.
- It supports most of the written languages.
- Each character is represented using **2 bytes**
- **65536** symbols, **`\uxxxx`** `(\u03B1 → α)`
- ASCII compatible
- Structured in **blocks**: Basic Latin, Greek, Arabic, Gothic, Currency, Mathematical, Arrows, Musical, etc.
- public class **приветмир**{ }
- System.out.println(" 好世界 ");

Java Basic Syntax

- Similar to C++
- Keywords
- Literals: “Hello World”, 'J', 'a', 'v', 'a', 10, 010, 0xA, 0b11, 12.3, 12.3d, 12.3f, 12e3, 123L, true, false, null, 0722_123_456
- Separators: ( ) { } [ ] ; , .
- Operators

  (char)65 + “nna” + “has” + (8 >> 2) + “ apples”

  http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html

Comments

```
/* To change this template file, choose Tools | Templates
   and open the template in the editor. */
/**
 * Main class of the application
 * @author Duke
 */
public class HelloWorld {
    /**
     The execution of the application starts here.
     @param args the command line arguments
     */
    public static void main(String args[]) {
        // TODO code application logic here
        System.out.println("Hello World!"); // Done!
    }
}
```

javadoc – a tool for generating API documentation in HTML format from doc comments in source code

Data Types

Primitive types

- arithmetic: `byte (1), short (2), int (4), long (8)`
- floating point: `float (4), double (8)`
- character: `char (2)`
- logical: `boolean (?)`

Reference types

classes, interfaces, annotations, enumerations

~~pointer, struct, union~~

Variables

Declaration [+ Initialization]

```
byte a;
int value = 100;
final double PI = 3.14;
boolean isFebruary = true;
long numberOfElements = 12345678L;
String myFavouriteDrink = "water";
```

Java naming conventions

Variables (cont.)

```
class Example {
  int a; //class member

  public void someMethod(int b) { //method argument
    a = b;
    int c = 10;//local to a method
    for(int d=0; d < 10; d++) {
      //local to a block of code
      c --;
    }
    try {
      a = b/c;
    } catch(ArithmeticException e){
      //exception handler argument
      System.err.println(e.getMessage());
    }
  }
}
```

Control Flow Statements

- Decision-making

  if-else, switch-case
- Looping

  for, while, do-while
- Exception handling

  try-catch-finally, throw
- Branching

  break, continue, return, ~~goto~~, *label:*

Arrays

- Declaration

```
int[] a; byte b[];
```

100 elements of type char

- Instantiation

```
a = new int[10]; char c[] = new char[100];
```

- Initialization

```
String colors[] = {"Red", "Yellow"};

someMethod( new String[] {"Red", "Yellow"} );
```

- The size of an array

`a.length` **and not** ~~`a.length()`~~

Multi-dimensional Arrays

- Arrays of arrays

```
int[][] m2d = new int[10][20];

int[][][] m3d = new int[10][20][30];
```

- Copying arrays

```
System.arrayCopy

int a[]; int b[]; …  What about a = b?;
```

- Utility methods for arrays

```
java.util.Arrays

   - binarySearch, equals, fill, ...
```

Strings

- char[]

```
char data[] = {'a', 'b', 'c'};
```

- String Immutable Object

```
String s = "abc";  String s = "a" + "b" + "c";

String s = new String("abc");

String s = new String(data);
```

- StringBuilder, StringBuffer

```
StringBuilder sb = new StringBuilder("a");

sb.append("b").append("c");
```

Equality Testing

- Arrays

```
int a[] = {1, 2};
int b[] = {1, 2};
a == b / a.equals(b) / Arrays.equals(a,b)
```

- Strings

```
String s1 = new String("abc");
String s2 = new String("abc");
s1 == s2 / s1.equals(s2) / s1.compareTo(s2)
"abc" == "abc"
```

Example of Using Chars and Strings

```
/** Generates random words, using a given set of characters. */
public class Example {

  public static void main(String args[]) {
      Example app = new Example();
      int nbWords = 10; //how many words to generate
      final int alphabetSize = 26; //how many characters has the alphabet
      char[] latin = new char[alphabetSize]; //create the alphabet array
      for (int i = 0; i < latin.length; i++) {
          latin[i] = (char) ('a' + i); //a b c d ...
      }
      String words[] = app.generate(nbWords, latin);
    }

    public String[] generate(int n, char[] alphabet) {
      String[] words = new String[n];
      for (int i = 0; i < n; i++) {
          StringBuilder sb = new StringBuilder();
          while (true) {
              int pos = (int) (Math.random() * (alphabet.length + 1)) - 1;
              if (pos < 0) break;
              sb.append(alphabet[pos]);
          }
          words[i] = sb.toString();
      }
      return words;
  }
}
```

Command Line Arguments

```
public class Main {
  public static void main (String args[]) {
    if (args.length < 3) {
      System.out.println("Not enough arguments!");
      System.exit(-1);
    }
    String str = args[0];
    int a = Integer.parseInt(args[1]);
    double x = Double.parseDouble(args[2]);
  }
}
```

```
java Main "Hello World" 2016 1.8
```

Bibliography

- ***The Java Tutorials***

  http://docs.oracle.com/javase/tutorial/
- ***The Java Language Specification,*** James Gosling, Bill Joy, Guy Steele, Gilad Bracha
- ***The Java Virtual Machine Specification***

  Tim Lindholm, Frank Yellin
- **Curs practic de Java**, C. Frăsinaru
- http://profs.info.uaic.ro/~acf/java