Software Concepts -- Introduction

- Now we can begin to examine the basic ideas behind writing programs

- Chapter 2 focuses on:
  - the structure of a Java application
  - basic program elements
  - preparing and executing a program
  - basic object-oriented programming concepts
  - helpful support for writing software
  - Java applets

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Java Program Structure

- See `Lincoln.java`
- A program is made up of one or more *classes*
- A class contains one or more *methods*
- A method contains program *statements*
- A Java application always executes the *main* method

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

White Space

- Spaces, blank lines, and tabs are collectively called *white space* and are used to separate words and symbols in a program
- Extra white space is ignored
- A valid Java program can be formatted many different ways
- See `Lincoln2.java` and `Lincoln3.java`
- Programs should be formatted to enhance readability, using consistent indentation

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Comments

- Comments in a program are also called *inline documentation*
- They should be included to explain the purpose of the program and describe processing steps
- Java comments can take two forms:

```
//  comment runs to the end of the line

/*  comment runs to terminating

    symbol, even across line breaks  */
```

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Identifiers

- *Identifiers* are the words a programmer uses in a program
- Most identifiers have no predefined meaning except as specified by the programmer
- An identifier can be made up of letters, digits, the underscore character (_), and the dollar sign
- They cannot begin with a digit
- Java is *case sensitive*, therefore `Total` and `total` are different identifiers

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Reserved Words

- Some identifiers, called *reserved words*, have specific meanings in Java and cannot be used in other ways

abstract
boolean
break
byte
byvalue
case
cast
catch
char
class
const
continue
default
do
double
else
extends
false
final
finally
float
for
future
generic
goto
if
implements
import
inner
instanceof
int
interface
long
native
new
null
operator
outer
package
private
protected
public
rest
return
short
static
super
switch
synchronized
this
throw
throws
transient
true
try
var
void
volatile
while

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Literals

- A *literal* is an explicit data value used in a program
- Integer literals:

```
25      69      -4288
```

- Floating point literals:

```
3.14159     42.075     -0.5
```

- String literals:

```
"The result is: "
"To thine own self be true."
```

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

The Java API

- The Java *Application Programmer Interface* (API) is a collection of classes that can be used as needed
- The `println` and `print` methods are part of the Java API; they are not part of the Java language itself
- Both methods print information to the screen; the difference is that `println` moves to the next line when done, but `print` does not
- See `Countdown.java`

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

String Concatenation and Addition

- The `+` operator serves two purposes
- When applied to two strings, they are combined into one (*string concatenation*)
- When applied to a string and some other value (like a number), that value is converted to a string and they are concatenated
- When applied to two numeric types, they are added together arithmetically
- See `Antarctica.java` and `Sum.java`

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Programming Languages

- There are four basic programming language levels:
  - machine language
  - assembly language
  - high-level language
  - fourth-generation language
- Each CPU has its own specific *machine language*
- The other levels were created to make programming easier

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Programming Languages

- A program must be translated into machine language before it can be executed on a particular type of CPU
- This can be accomplished in several ways
- A *compiler* is a software tool which translates source code into a specific target language
- Often, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

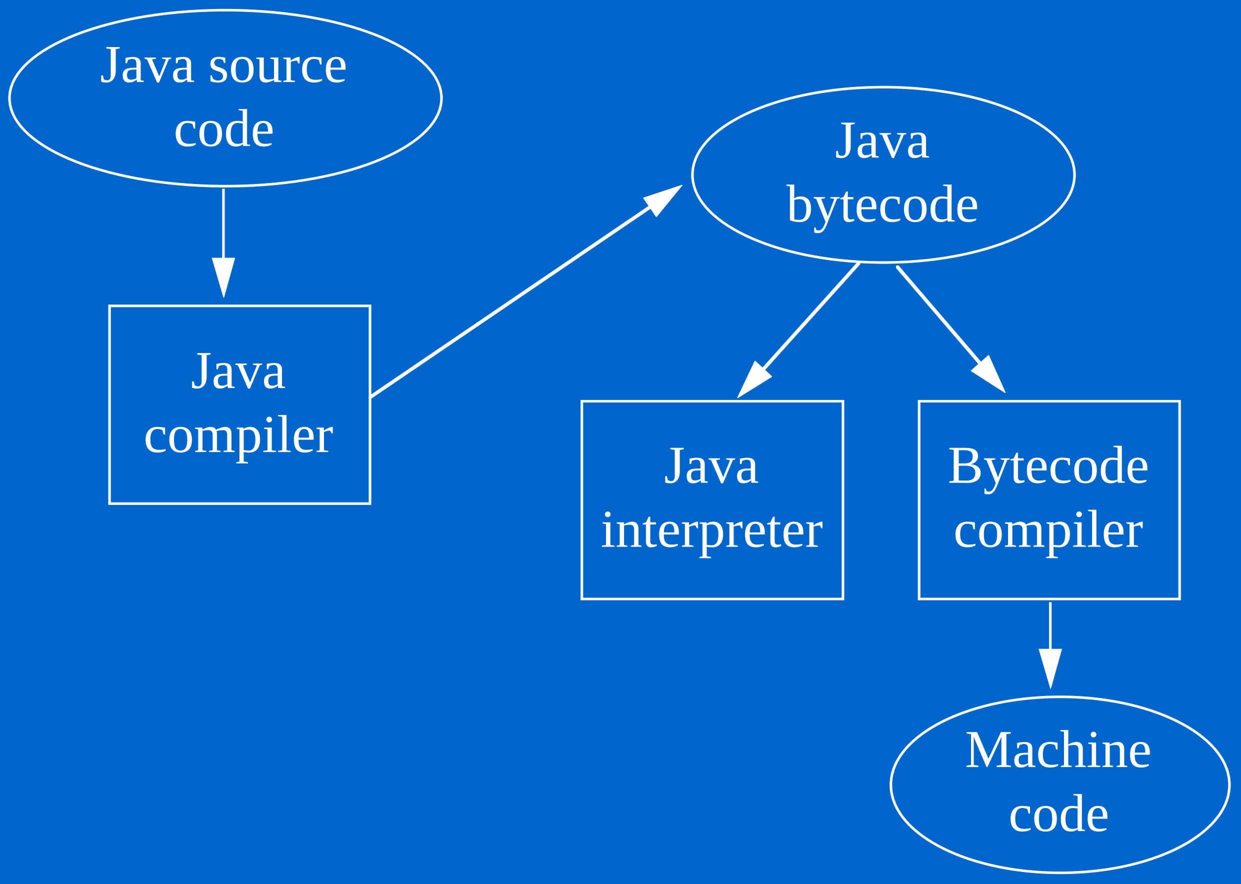
Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Java Translation and Execution

- The Java compiler translates Java source code into a special representation called *bytecode*
- Java bytecode is not the machine language for any traditional CPU
- Another software tool, called an *interpreter*, translates bytecode into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine
- Java is considered to be *architecture-neutral*

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Java Translation and Execution



Java source code

Java compiler

Java bytecode

Java interpreter

Bytecode compiler

Machine code

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Java Translation and Execution

- Executing the compiler in a command line environment:

```
>  javac Lincoln.java
```

- This creates a file called `Lincoln.class`, which is submitted to the interpreter to be executed:

```
>  java Lincoln
```

- The `.java` extension is used at compile time, but the `.class` extension is not used with the interpreter

- Other environments do this processing in a different way

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Syntax and Semantics

- The *syntax* of a language defines how you can put symbols, reserved words, and identifiers together to make a valid program
- The *semantics* of a language construct is the meaning of the construct; it defines its role in a program
- A syntactically correct program does not mean it is logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Errors

- A program can have three types of errors
- The compiler will find problems with syntax and other basic issues (*compile-time errors*)
  - If compile-time errors exist, an executable version of the program is not created
- A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
- A program may run, but produce incorrect results (*logical errors*)

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Command Line Arguments

- See `Name_Tag.java`
- The `main` method accepts extra information on the command line when a program is executed

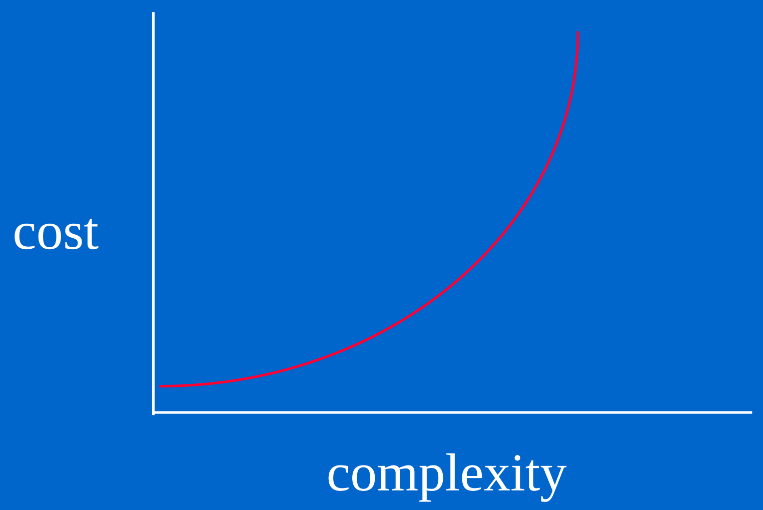
```
> java Name_Tag John
```

- Each extra value is called *command line argument*
- In Java, command line arguments are always read as a list of character strings

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Software Engineering

- We should always strive to engineer our software to make it reliable and maintainable
- As the complexity of a program increases, its cost to develop and revise grows exponentially

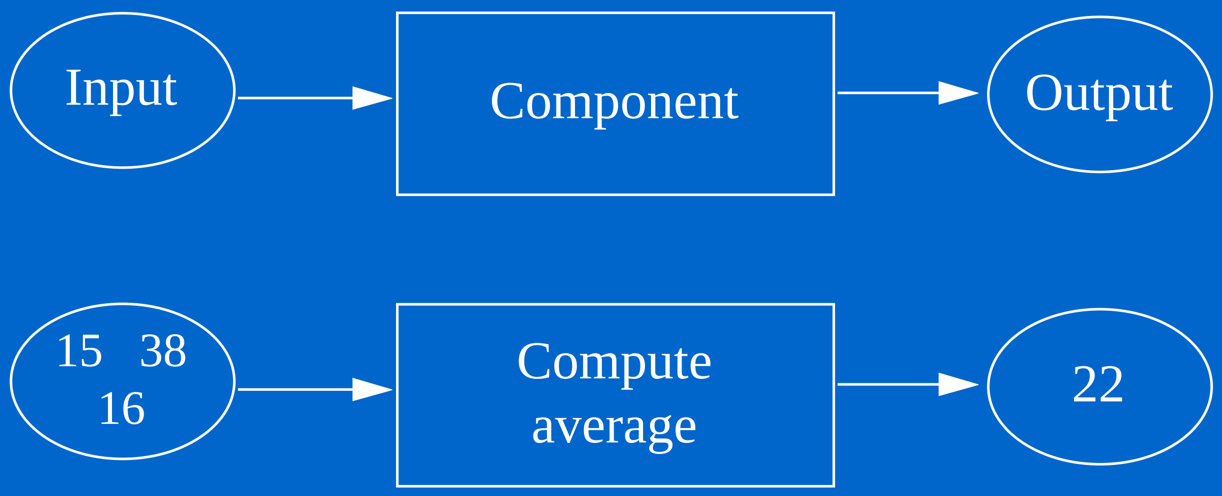


Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Software Components

- Programs are easier to construct and modify when they are made up of separate components

- A software component can be thought of as any program element that transforms input into output



Input

Component

Output

15 38 16

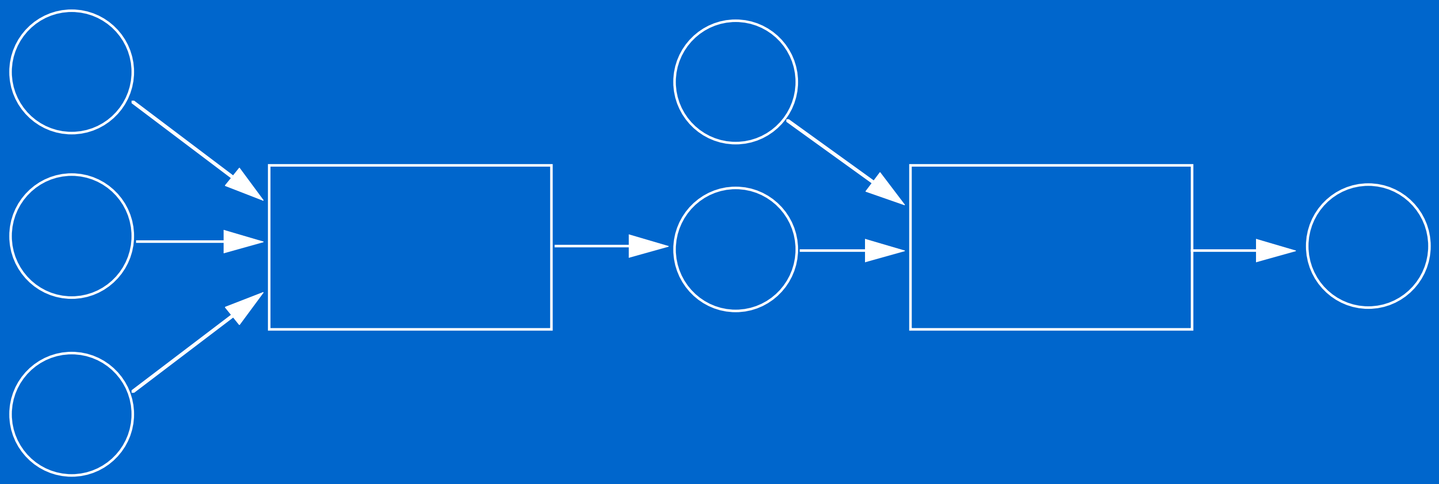
Compute average

22

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Software Components

- Components can be combined to make larger components



Copyright 1997 by John Lewis and William Loftus. All rights reserved.

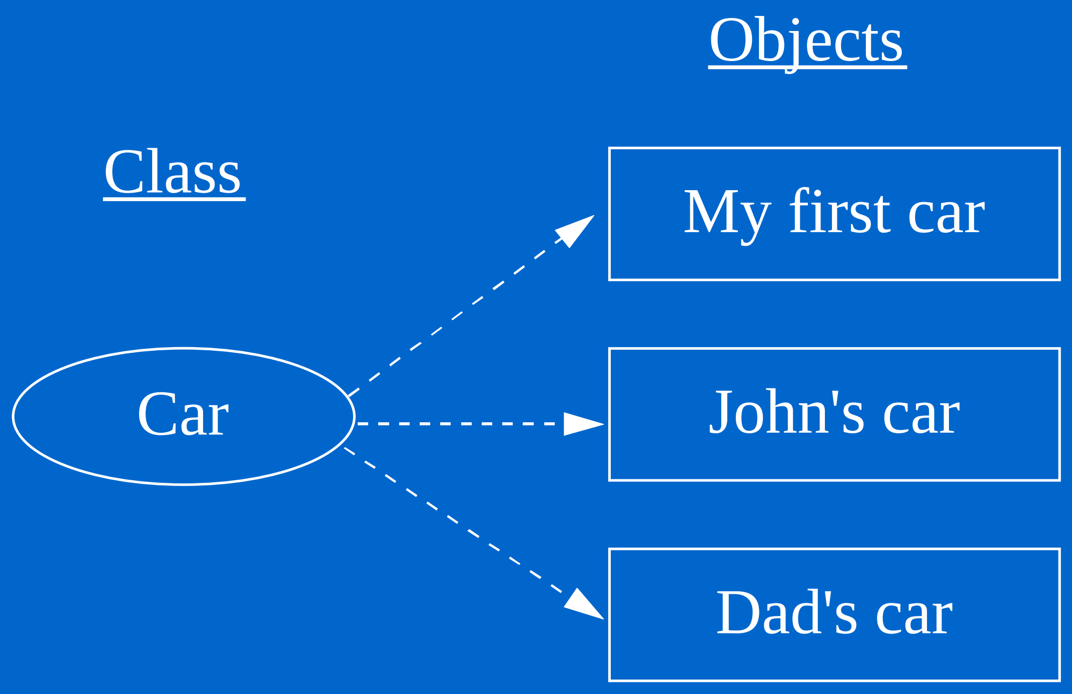
Object-Oriented Programming

- Java is *object-oriented language*
- Programs are made from software components called objects
- An *object* contains data and methods
- An object is defined by a *class*
- Multiple objects can be created from the same class

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Object-Oriented Programming

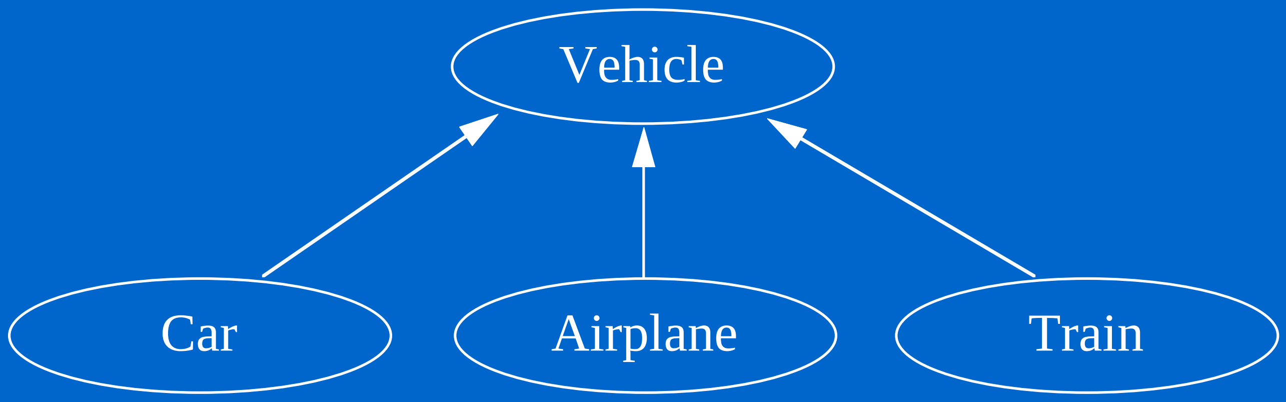
- A class represents a concept and an object represents the realization of that concept



Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Object-Oriented Programming

- Objects can also be derived from each other using a process called *inheritance*



- Objects, classes, and inheritance will be discussed in greater detail later

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Class Libraries

- The Java API is a *class library*, a group of classes that support program development
- Classes in a class hierarchy are often related by inheritance
- The classes in the Java API is separated into *packages*
- The `System` class, for example, is in package `java.lang`
- Each package contains a set of classes that relate in some way

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

The Java API Packages

- Some packages in the Java API:

java.applet
java.awt
java.beans
java.io
java.lang
java.math
java.net
java.rmi
java.security
java.sql
java.text
java.util

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Importing Packages

- Using a class from the Java API can be accomplished by using its fully qualified name:

```
java.lang.System.out.println ();
```

- Or, the package can be imported using an *import statement*, which has two forms:

```
import java.applet.*;

import java.util.Random;
```

- The `java.lang` package is automatically imported into every Java program

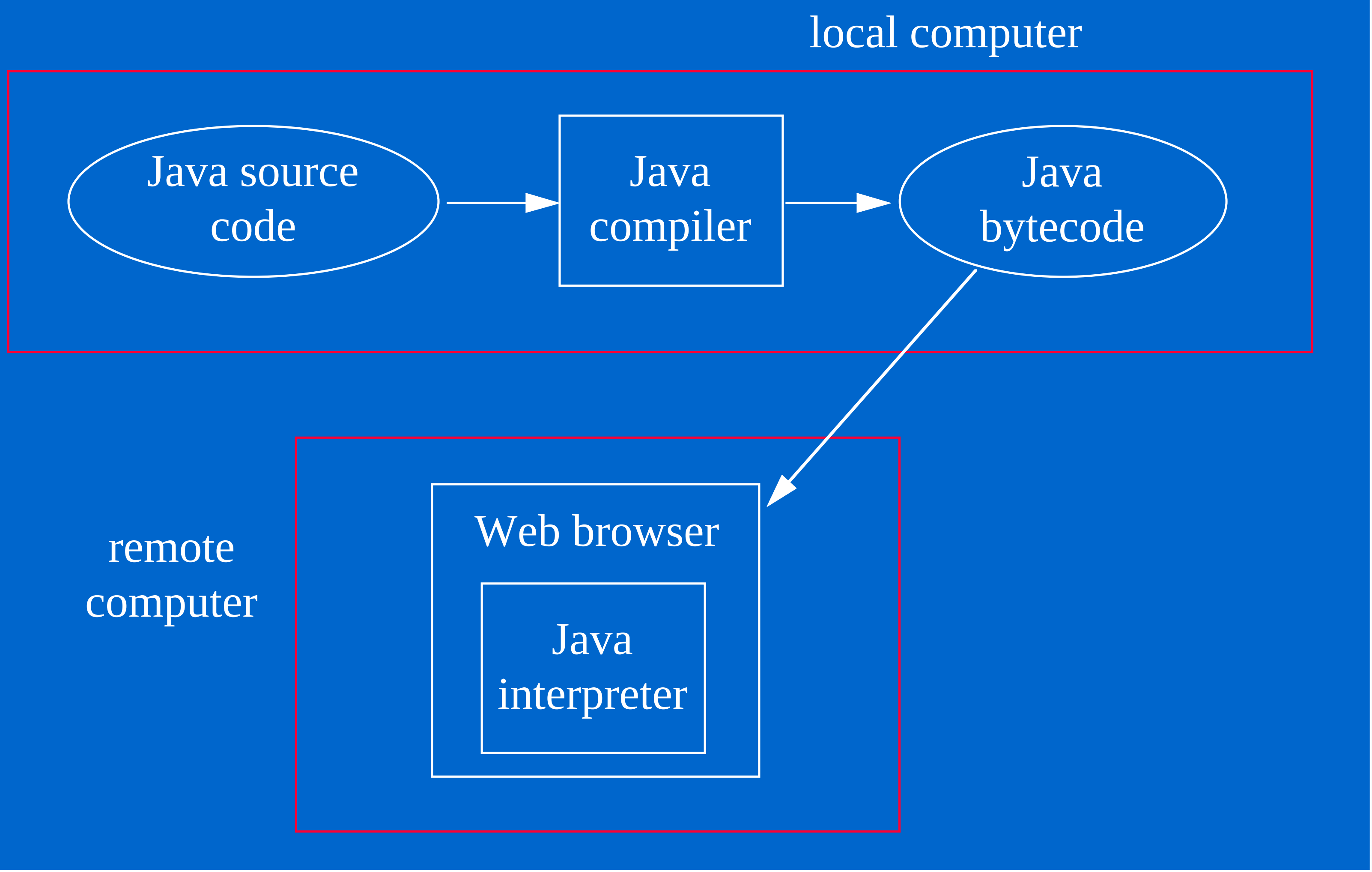
Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Java Applets

- A *Java applet* is a Java program that is intended to be sent across a network and executed using a Web browser
- A *Java application* is a stand alone program
- Applications have a `main` method, but applets do not
- Applets are derived from the `java.applet.Applet` class
- See `Confucius.java` and `No_Parking.java`
- Links to applets can be embedded in HTML documents

Copyright 1997 by John Lewis and William Loftus. All rights reserved.

Java Applets



Copyright 1997 by John Lewis and William Loftus. All rights reserved.