

#### Some History

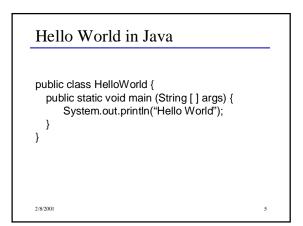
- 1993 Oak project at Sun
- 1995 Oak becomes Java; every major web player announces support
- 1996 Java 1.0 available
- 1997 (March) Java 1.1 some language changes and much larger library, including new event handling GUI model (AWT)
- 1997 (September) Java 1.2 beta including Swing GUI package
- 1998 (October) Java 1.2 final
- 2000 (April) Java 1.3 final

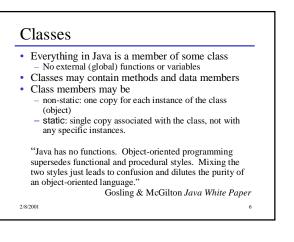
2/8/2001

## Design Goals

- Support secure, high-performance, robust applications running as-is on multiple platforms and over networks
- "Architecture-neutral", portable, allow dynamic updates and adapt to new environments
- Look enough like C++ for programmer comfort
- Support object-oriented programming
- Support concurrency (multithreading)
- Simplicity

2/8/2001





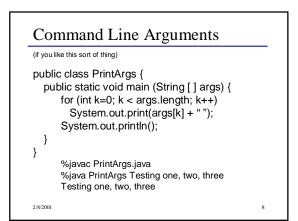
#### Hello World Revisited

public class HelloWorld {
 public static void main (String [] args) {
 System.out.println("Hello World");
 }
}

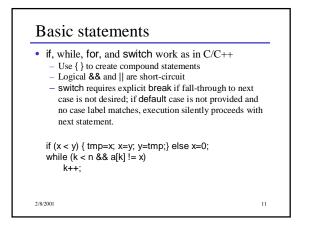
- }
- Every class may have a main method
- Execution begins in main of a designated class
- Class Xyzzy should be in file Xyzzy.java

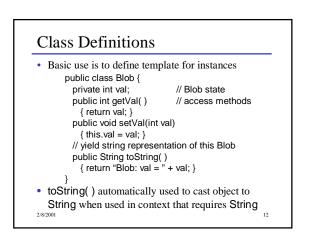
%javac HelloWorld.java %java Helloworld Hello World





#### Primitive Data Types Vars, Expressions & Assignment • 2's complement signed integer • Almost same as C/C++ int (32 bits), byte (8), short (16), long (64) int k = 17; boolean maybe; double x=42.0 constants are normally type int k = 2 \* k; maybe = k > 17; • IEEE floating point · Declaration initializers are optional. If omitted, - double (64 bits), float (32) - Fields in class instances initialized to 0, false, null. floating constants are normally type double - Local vars in methods not initialized by default; • Unicode characters: char (16 bits) compiler complains if use before initilize is possible Logical: boolean · Assignment does coercion if no information lost constants are true, false double y = (k+6)/2;not interchangeable with int · Assignment that could lose information requires · None of these are "implementation-defined" or explicit cast "implementation-dependent" k = (int) x \* 1.3 / (x-2)2/8/2001 2/8/200 10

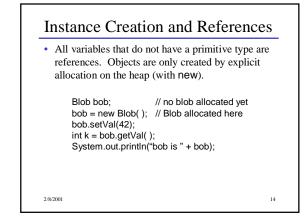




#### Visibility

- Class members can be preceded by a qualifier to indicate accessibility
  - public accessible anywhere the class can be accessed
     private only accessible inside the class
  - private only accessible inside the class
     If nothing is specified, the field can be referenced
  - anywhere in the same package (more later).
  - protected same as package visibility, and also visible in classes that extend this class.

2/8/2001



#### References and Methods

- Dot notation is used to select methods and fields; implicit dereference (no -> as in C/C++).
- No pointer arithmetic; no & operator to generate the address of arbitrary variable; can't create pointers from random bits.
   "Java has no pointers"
- All method parameters are call-by-value (copy of primitive value or object reference)
- Methods can be overloaded (different methods with same name but different number or types of parameters).

2/8/2001

### **Object Allocation**

2/8/2001

- A variable declared as class X has type "reference to X". No object is created by such a declaration.
- Declaration and object creation can be combined.
   Blob bob = new Blob();
- The constant null belongs to all reference types and refers to nothing.
- If reference r is null, then selecting a field from r (r.fieldname) throws a NullPointerException.
- Storage occupied by an object is dynamically reclaimed when the object is no longer accessible (automatic garbage collection).

16

Constructors

• Constructor(s) can be provided to initialize objects
when they are created. Constructors can be
overloaded and can call other constructors.
class Blob {
 int val;
 // constructors

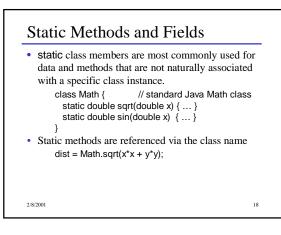
```
Blob (int initial) { val = initial; }
Blob () { this(17); }
```

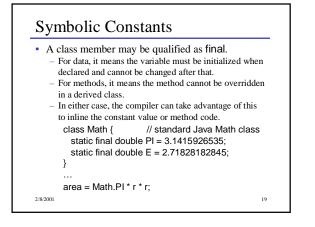
2/8/2001

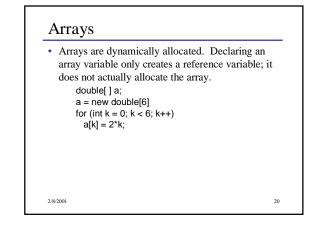
17

13

15





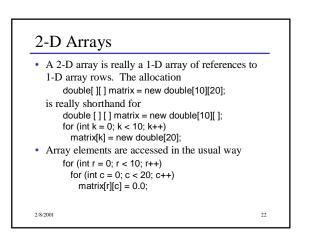


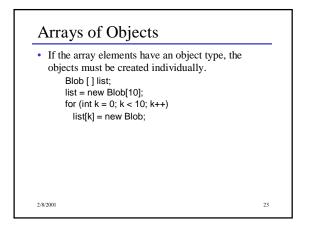
# Array NotessArrays are 0-origin, as in C/C++

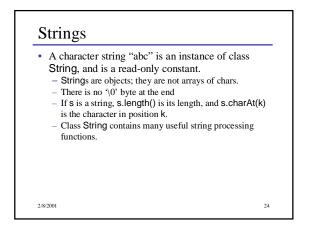
- Arrays are also objects, with one constant member
- If a is an array, a.length is its length
- An IndexOutOfBoundsException is thrown if a subscript is < 0 or >= the array length.
- The brackets indicating an array type may also appear after the variable name, as in C/C++ int a[] = new int[100];

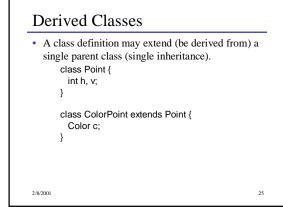
2/8/2001

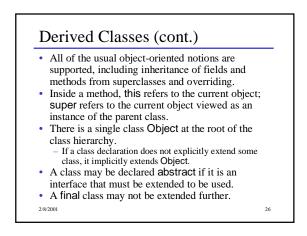
21

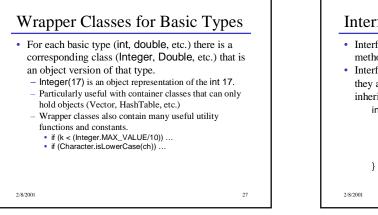


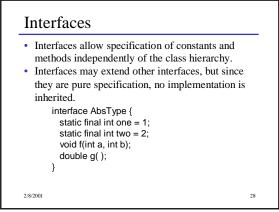


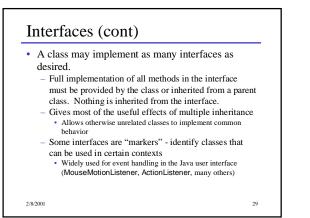


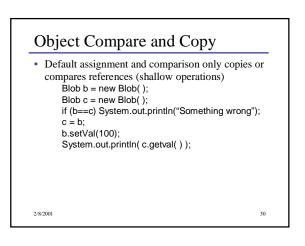












#### Deep Compare and Copy

- · All classes inherit equals and clone from Object
  - Default versions do a shallow compare/copy
  - Override if a deep compare/copy is desired
  - To override clone, a class must also extend the Cloneable interface (this is purely a marker interface, has no methods or constants)
- Intended meaning of a.equals(b) is that a and b are "equal" in whatever sense is appropriate for the class of a and b.
- b.clone should create a new "copy" of b and return a reference to it.

2/8/2001

#### Exceptions • Java has an extensive exception handling mechanism. Basic idea try { thisMightExplode(x,y,z); } catch (Exception e) { <deal with the problems } • If an exception happens, a throw new anExceptionClass(parameters); statement will cause the call chain to unwind until a catch clause that matches the thrown object is found. 28200

#### Exceptions (cont) • Multiple catch clauses can be used to selectively handle exceptions try { tryToReadData(x,y,z); } catch (IOException e) { <deal with I/O problem> } catch (Exception e) { <deal with other exceptions> } • If a method does something that might generate an exception, it must either handle it, or declare that it might throw that exception (throws clause). 2/8/2001 2/8/2001 33

31

#### Packages

- Packages provide a way to partition the global class namespace.
- A class is placed in a package by including at the beginning of class source file package widget;
- A class in another package can use items from a package by explicitly qualifying the item name widget.Blob b = new widget.Blob();

or by importing names from the package import widget.\*;

Blob b = new Blob( );

34

36

# Packages (cont)

- Package names are grouped into hierarchies by using package names with embedded dots

   java.util, java.awt, java.awt.image
- Parts of a package hierarchy can be selectively imported.
- import is not transitive (unlike C/C++ #include)
- If a class definition does not include a package statement, that class is part of a default anonymous package.
  - Useful for small projects

2/8/2001

35

#### Streams

- Stream = flow of data (bytes or characters)
- Can be associated with files, communication links, keyboard/screen/printer
- Many stream classes; most are designed to be used as wrappers that accept data and transform or filter it before passing it along
- Java 1.0: Byte streams with a few wrappers to handle ASCII text
- Java 1.1: Added text stream classes to handle Unicode text properly

2/8/2001

#### Stream Classes (1)

- InputStream/OutputStream abstract classes defining basic raw byte stream operations
- Reader/Writer abstract classes defining basic text stream operations
- All Java stream classes are built on top of these
- InputStreamReader/OutputStreamWriter basic conversion between bytes and characters (in both directions)

2/8/2001

# Stream Classes (2) BufferedInputStream/ BufferedOutputStream BufferedReader/BufferedWriter versions of streams that add buffering and additional input/output methods PrintWriter - Text stream with methods for printing Strings and primitive types as text output.

38

# Stream Classes (3) DataInputStream/DataOutputStream -Filter streams that can read/write simple types including String and primitive numeric types as binary byte streams. FileInputStream/FileOutputStream FileReader/FileWriter - byte and text streams that read and write from/to the local file system.

2/8/2001

39

37

2/8/2001

# Ex: Read a byte from Keyboard • System.in is an InputStream. At the lowest level, we can read bytes. As in C, the basic read() operation returns an int, with -1 indicating end of stream. try { int nibble = System.in.read(); catch (IOException e) { ... }

