

## References (2)

- Overview of Object-Oriented Programming
  - Understanding Object-Oriented Programming with Java by Tim Budd (Addison-Wesley)
- · Longer tutorial on language and libraries
  - Learning Java by Niemeyer & Knudsen (O'Reilly)
- Look-it-up references
- · Java in a Nutshell (core language and libraries)
- Java Foundation Classes in a Nutshell (AWT, Swing)
- Java Examples in a Nutshell
- all by David Flanagan (O'Reilly)

Introduction to Java - CSE 413 Wi02

### Some History • 1993 Oak project at Sun • 1995 Oak becomes Java; web happens • 1996 Java 1.0 available

- 1997 (March) Java 1.1 some language changes, much larger library, new event handling model
- 1997 (September) Java 1.2 beta huge increase in libraries including Swing, new collection classes, J2EE

Introduction to Java - CSE 413 Wi02

- 1998 (October) Java 1.2 final (Java2!)
- 2000 (April) Java 1.3 final

2/8/2002

- early 2002 Java 1.4 final (assert)
- 2002-2003 Java 1.5 (parameterized types?)

Introduction to Java

2/8/2002

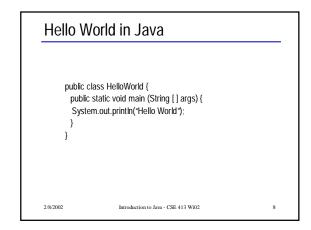
### **Design Goals**

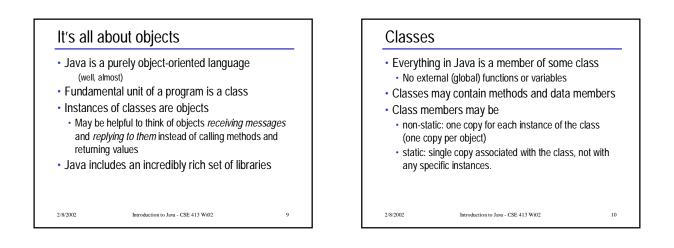
- Support secure, high-performance, robust applications running as-is on multiple platforms and over networks
- "Architecture-neutral", portable, allow dynamic updates and adapt to new environments
- Look enough like C++ for programmer comfort

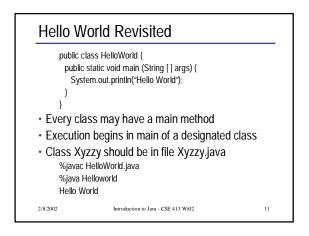
Introduction to Java - CSE 413 Wi02

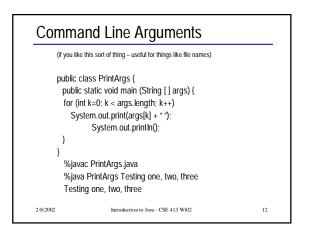
- Support object-oriented programming
- Support concurrency (multithreading)
- Simplicity

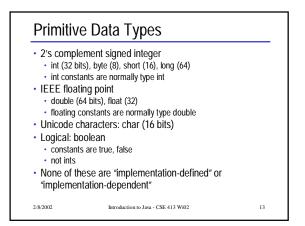
2/8/2002

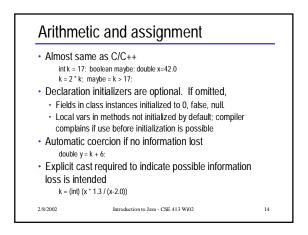


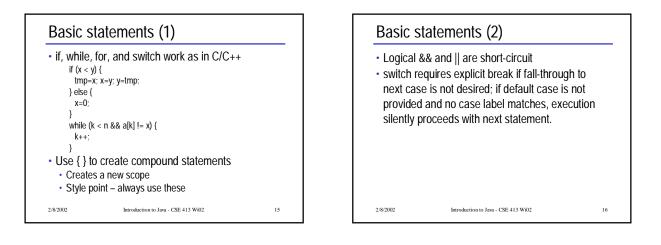


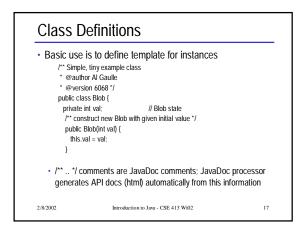


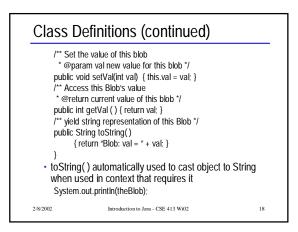














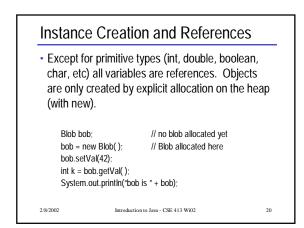
2/8/2002

Constructor(s) can be provided to initialize objects when they are created. Constructors can be overloaded and can delegate to other constructors.
 class Blob {
 private int val;
 /\*\* construct Blob with given initial value \*/
 Blob (int initial) { val = initial; }
 /\*\* construct Blob with default initial value \*/
 Blob () { this(17); }
 ...

Introduction to Java - CSE 413 Wi02

19

21

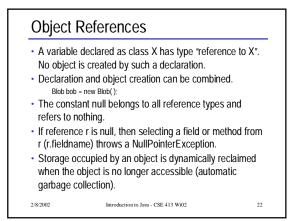


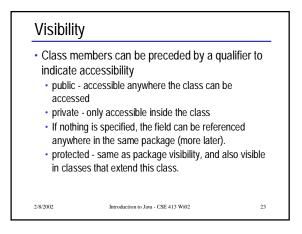
## References and Methods Dot notation is used to select methods and fields; implicit dereference (no -> as in C/C++). No pointer arithmetic; no & operator to generate the address of arbitrary variable; can't create pointers from random bits. "Java has no pointers" All method parameters are call-by-value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

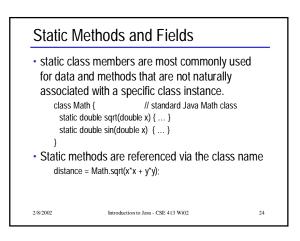
parameters).

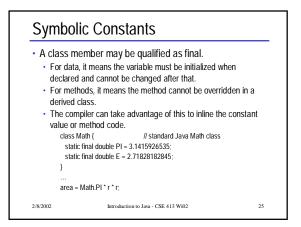
2/8/2002

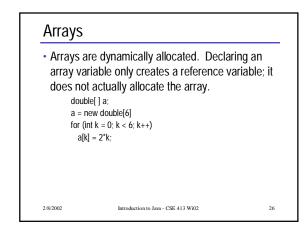
Introduction to Java - CSE 413 Wi02

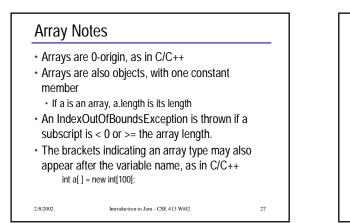


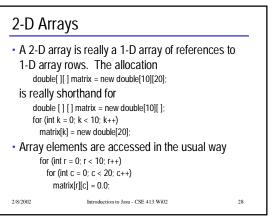


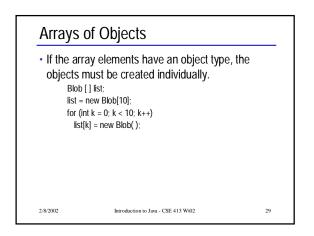


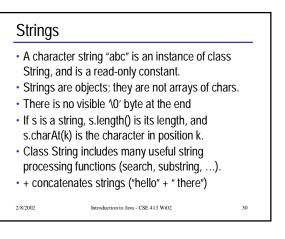


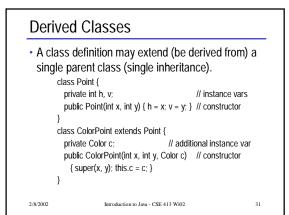


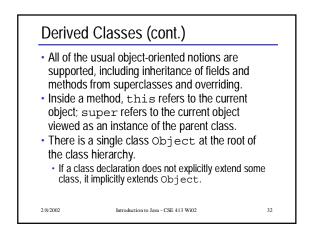


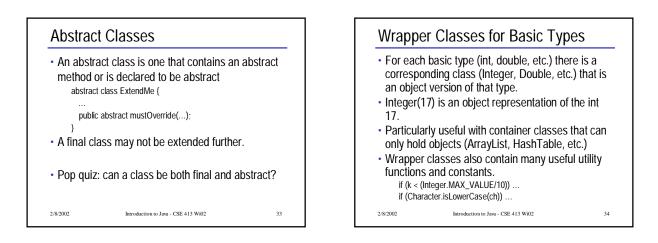


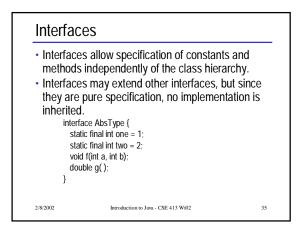


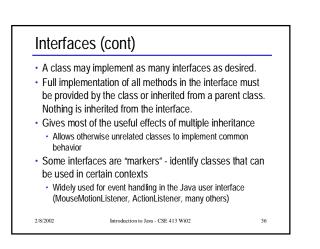












### Interfaces and Abstract Types

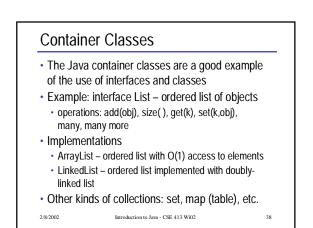
• Both define a new type

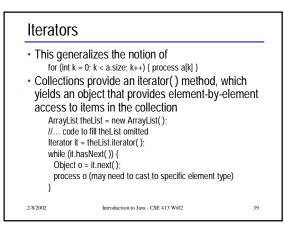
2/8/2002

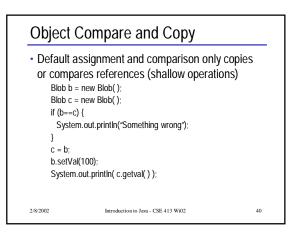
- In real systems, any important type should be defined by an interface
- Specifies the type without tying to an implementation
  Often, should provide a model implementation of
- the interface in an abstract or concrete class
- Programmer has choice of implementing the interface or using (maybe extending) the abstract class

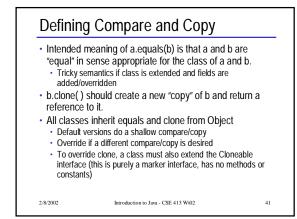
Introduction to Java - CSE 413 Wi02

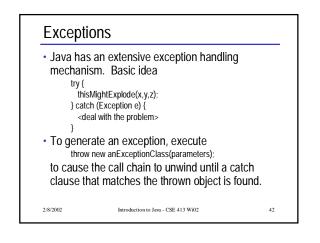
37

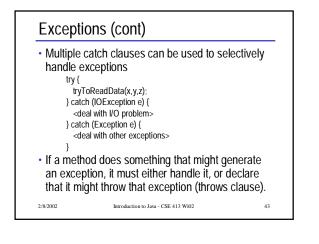


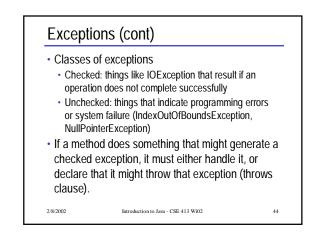












### Packages

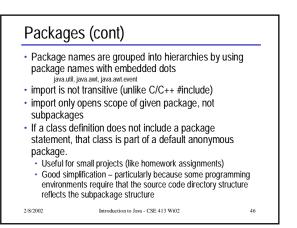
- Packages provide a way to partition the global class namespace.
- A class is placed in a package by including at the beginning of the class source file package widget:
- A class in another package can use items from a package by explicitly qualifying the item name widget.Blob b = new widget.Blob();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

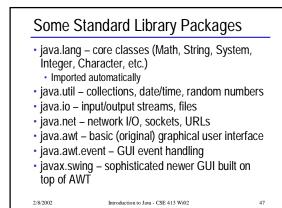
Blob b = new Blob();

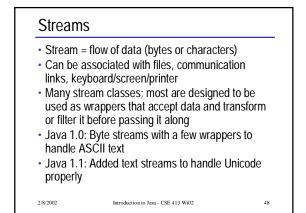
2/8/2002

Introduction to Java - CSE 413 Wi02

45





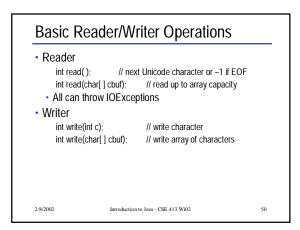


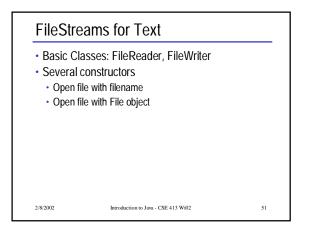
### Stream Abstract Classes

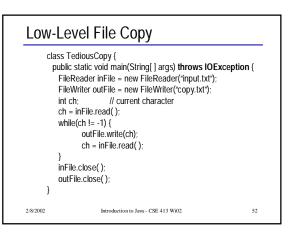
- Byte streams: InputStream, OutputStream
- · Character streams: Reader, Writer
- All Java stream classes are extensions of these (directly or indirectly)
- There are wrapper classes to convert between these
- Historical note: console I/O streams (System.in, System.out, System.err) existed in Java 1.0, so these are InputStreams and OutputStreams, even though they really should be Readers and Writers

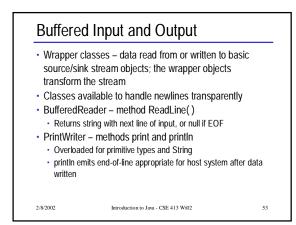
Introduction to Java - CSE 413 Wi02

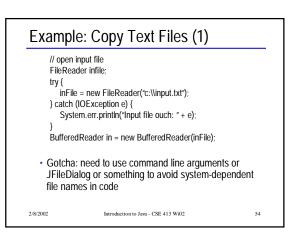
2/8/2002

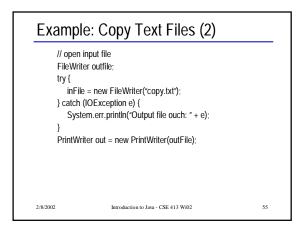


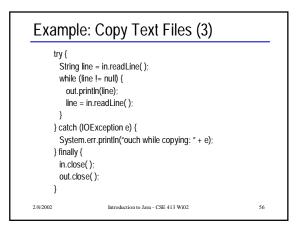


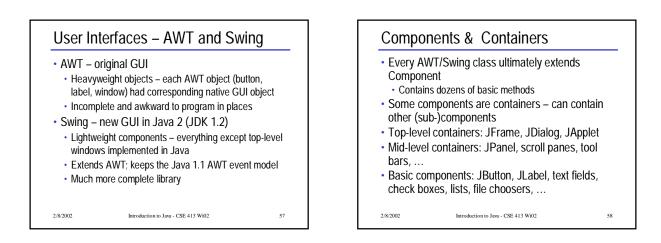


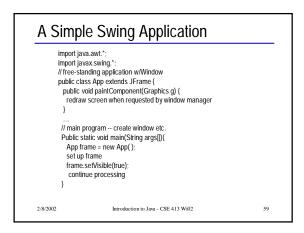


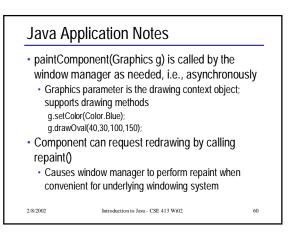












### **Event Handling**

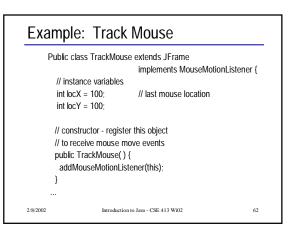
- User interface components generate events
- Objects (often other components) can register themselves to receive events of interest
- When an event happens, an appropriate method is called in all listeners (all registered objects)
- A listener object must implement the interface corresponding to the events, which means implementing all methods declared in the interface

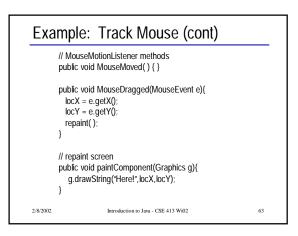
Introduction to Java - CSE 413 Wi02

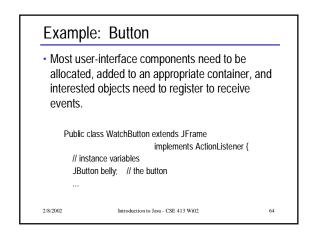
- Need import java.awt.event.\*;
- 2/8/2002

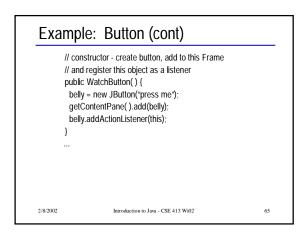


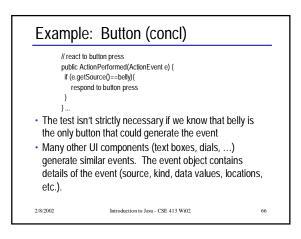
61

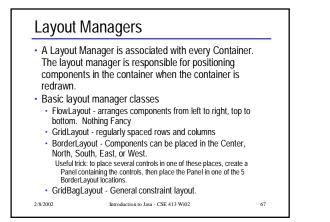


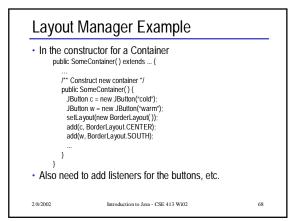












# Dhreads A fixed a fixed fixed

