

#### Goals

2/8/2002

- Survey of major Java language and library features
- Orientation not comprehensive
  - No way anyone actually understands all of the libraries(!)
  - Part of the job of learning a new language/ environment is to learn how to find information

Introduction to Java - CSE 413 Wi02

2

Ask lots of questions!

# Dverview A bit of history Casses and objects Core Java language Collection classes Calsas relationships – inheritance and interfaces Packages & scope Suception handling Cul basics (AWT & Swing) Threads

#### References (1)

- Way too many to count. Here are a couple of useful places to start (i.e., I've found them useful)
- From Sun

2/8/2002

- Java SDK and documentation (java.sun.com)
  The Java Tutorial (A-W). Online at
- http://java.sun.com/docs/books/tutorial/index.html (Good "how to do it" topic orientation)
- The Java Programming Language by Arnold, Gosling, and Holmes (A-W, 3<sup>rd</sup> edition) (Language and container classes primarily)

Introduction to Java - CSE 413 Wi02

#### 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)

2/8/2002

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

6

- 1998 (October) Java 1.2 final (Java2!)
- 2000 (April) Java 1.3 final
- early 2002 Java 1.4 final (assert)
- 2002-2003 Java 1.5 (parameterized types?)

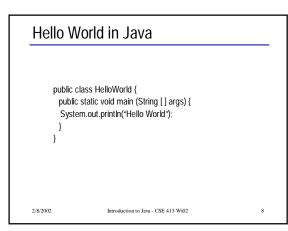
Introduction to Java - CSE 413 Wi02

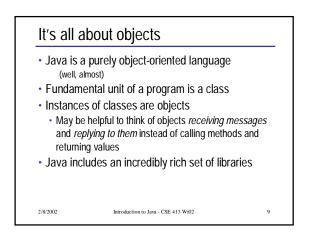
#### **Design Goals**

- 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



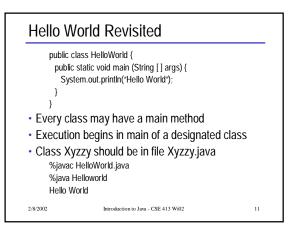


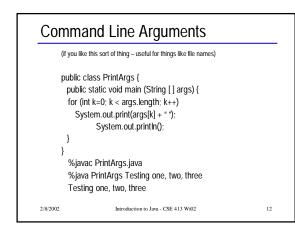
#### Classes

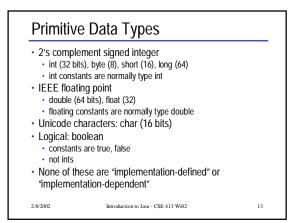
- Everything in Java is a member of some class • No external (global) functions or variables
- Classes may contain methods and data members
   Class members may be
- Class members may be
  - non-static: one copy for each instance of the class (one copy per object)
  - static: single copy associated with the class, not with any specific instances.

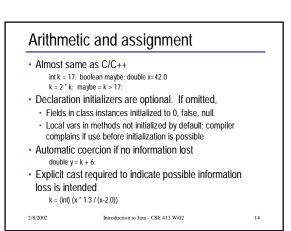
Introduction to Java - CSE 413 Wi02

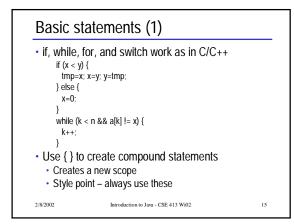
10









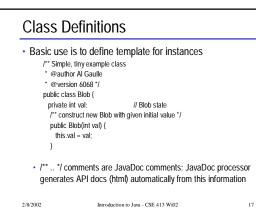


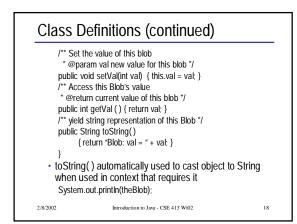
#### Basic statements (2)

2/8/2002

- · Logical && and || are short-circuit
- switch requires explicit break if fall-through to next case is not desired; if default case is not provided and no case label matches, execution silently proceeds with next statement.

Introduction to Java - CSE 413 Wi02





#### Constructors

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

20

21

2/8/2002

#### Instance Creation and References

• Except for primitive types (int, double, boolean, char, etc) all variables are references. Objects are only created by explicit allocation on the heap (with new).

Blob bob; // no blob allocated yet bob = new Blob(); // Blob allocated here bob.setVal(42); int k = bob.getVal((); System.out.println("bob is " + bob);

2/8/2002

Introduction to Java - CSE 413 Wi02

#### 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).

Introduction to Java - CSE 413 Wi02

#### **Object References**

- 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 or method 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).

Introduction to Java - CSE 413 Wi02

2/8/2002

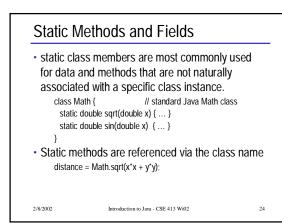
22

#### Visibility

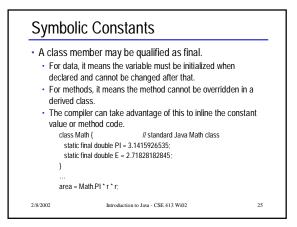
- Class members can be preceded by a qualifier to indicate accessibility
- public accessible anywhere the class can be accessed
- · private only accessible inside the class
- If nothing is specified, the field can be referenced anywhere in the same package (more later).
- protected same as package visibility,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

2/8/2002 Introduction to Java - CSE 413 Wi02

23



Introduction to Java



#### Arrays • Arrays are dynamically allocated. Declaring an array variable only creates a reference variable; it does not actually allocate the array. $\begin{array}{c} double[\ ] a; \\ a = new double[6] \\ for (int k = 0; k < 6; k++) \\ a[k] = 2^*k; \end{array}$

2/8/2002

Introduction to Java - CSE 413 Wi02

26

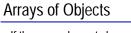
27

## Array Notes Arrays 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

 The brackets indicating an array type may also appear after the variable name, as in C/C++ int a[] = new int[100];

Introduction to Java - CSE 413 Wi02

A 2-D array is really a 1-D array of references to 1-D array rows. The allocation double[][] matrix = new double[10][20];
is really shorthand for double [][] matrix = new double[10][]; for (int k = 0; k < 10; k++) matrix[k] = new double[20];
Array elements are accessed in the usual way for (int r = 0; r < 10; r++) for (int r = 0; r < 10; r++) matrix[r][c] = 0.0;



If the array elements have an object type, the objects must be created individually.
 Blob [] list;
 list = new Blob[10];
 for (int k = 0; k < 10; k++)
 list[k] = new Blob();
 </li>

2/8/2002

Introduction to Java - CSE 413 Wi02

29

#### Strings

- A character string "abc" is an instance of class String, and is a read-only constant.
- Strings are objects; they are not arrays of chars.
- There is no visible '\0' byte at the end
- If s is a string, s.length() is its length, and s.charAt(k) is the character in position k.
- Class String includes many useful string processing functions (search, substring, ...).

Introduction to Java - CSE 413 Wi02

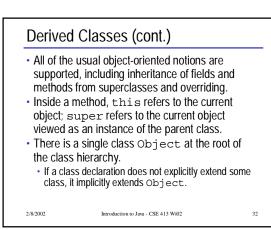
+ concatenates strings ("hello" + " there")

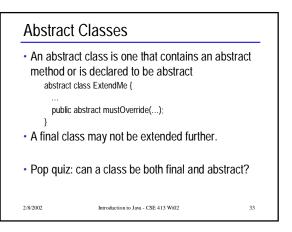
30

```
Derived Classes
• A class definition may extend (be derived from) a
single parent class (single inheritance).
class Point {
    private int h, v; // instance vars
    public Point(int x, int y) { h = x; v = y; } // constructor
    }
    class ColorPoint extends Point {
        private Color c; // additional instance var
        public ColorPoint(int x, int y, Color c) // constructor
        { super(x, y); this.c = c; }
    }
```

Introduction to Java - CSE 413 Wi02

31





#### Wrapper Classes for Basic Types

- For each basic type (int, double, etc.) there is a corresponding class (Integer, Double, etc.) that is an object version of that type.
- Integer(17) is an object representation of the int 17.
- Particularly useful with container classes that can only hold objects (ArrayList, HashTable, etc.)
- Wrapper classes also contain many useful utility functions and constants.
  - if (k < (Integer.MAX\_VALUE/10)) ...
  - if (Character.isLowerCase(ch)) ...
  - Introduction to Java CSE 413 Wi02

#### Interfaces

2/8/2002

- Interfaces allow specification of constants and methods independently of the class hierarchy.
- Interfaces may extend other interfaces, but since they are pure specification, no implementation is inherited.
  - interface AbsType {
     static final int one = 1;
     static final int two = 2;
     void f(int a, int b);

```
double g();
}
```

2/8/2002 Introduction to Java - CSE 413 Wi02

35

34

#### Interfaces (cont)

- · A class may implement as many interfaces as desired.
- Full implementation of all methods in the interface must be provided by the class or inherited from a parent class. Nothing is inherited from the interface.
- Gives most of the useful effects of multiple inheritance
   Allows otherwise unrelated classes to implement common behavior
- Some interfaces are "markers" identify classes that can be used in certain contexts
  - Widely used for event handling in the Java user interface (MouseMotionListener, ActionListener, many others)

36

#### Interfaces and Abstract Types

- Both define a new type
- 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
- 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

38

#### **Container Classes**

2/8/2002

- The Java container classes are a good example of the use of interfaces and classes
- Example: interface List ordered list of objects
   operations: add(obj), size(), get(k), set(k,obj),
- many, many more
- Implementations
- ArrayList ordered list with O(1) access to elements
   LinkedList ordered list implemented with doublylinked list
- Other kinds of collections: set, map (table), etc.
- 2/8/2002 Introduction to Java CSE 413 Wi02

#### Iterators

for (int Collection yields ar	neralizes the notion of k = 0; k < a.size; k++) { process a[k] } pros provide an iterator() method, which n object that provides element-by-eleme o items in the collection	ent
// co Iterator while (i Object	st theList = new ArrayList(); de to fill theList omitted it = theList.iterator(); t.hasNext()) { ct o = it.next(); sss o (may need to cast to specific element type)	
2/8/2002	Introduction to Java - CSE 413 Wi02	39

#### **Object Compare and Copy**

 Default assignment and comparison only copies or compares references (shallow operations) Blob b = new Blob(); Blob c = new Blob(); if (b==c) { System.out.println("Something wrong"); } c = b;

Introduction to Java - CSE 413 Wi02

40

41

```
b.setVal(100);
System.out.println( c.getval( ) );
```

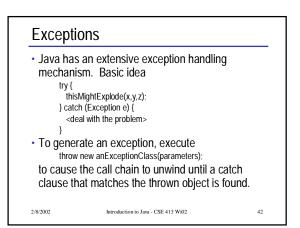
2/8/2002

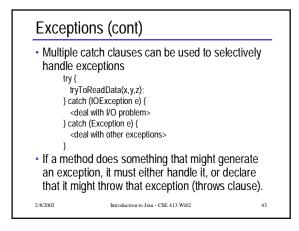
#### Defining Compare and Copy Intended meaning of a.equals(b) is that a and b are

- "equal" in sense appropriate for the class of a and b. • Tricky semantics if class is extended and fields are added/overridden
- b.clone() should create a new "copy" of b and return a reference to it.
- All classes inherit equals and clone from Object
   Default versions do a shallow compare/copy
  - Default versions do a snallow compare/copy
     Override if a different 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)

2/8/2002 Introdu

Introduction to Java - CSE 413 Wi02



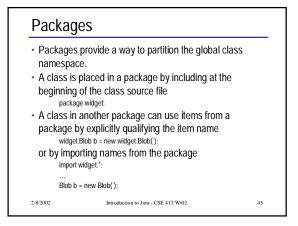


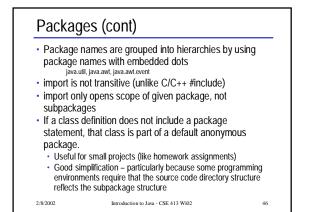
#### Exceptions (cont)

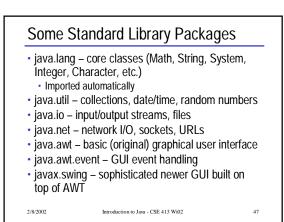
- · Classes of exceptions
  - Checked: things like IOException that result if an operation does not complete successfully
  - Unchecked: things that indicate programming errors or system failure (IndexOutOfBoundsException, NullPointerException)
- If a method does something that might generate a checked exception, it must either handle it, or declare that it might throw that exception (throws clause).

2/8/2002

Introduction to Java - CSE 413 Wi02







#### 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 streams to handle Unicode properly

Introduction to Java - CSE 413 Wi02

48

#### Stream Abstract Classes

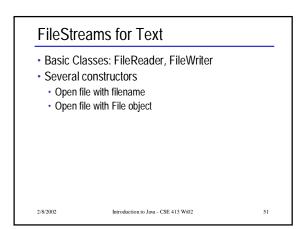
- Byte streams: InputStream, OutputStream
- Character streams: Reader, Writer
- All Java stream classes are extensions of these (directly or indirectly)
- There are wrapper classes to convert between these
- Historical note: console I/O streams (System.in, System.out, System.err) existed in Java 1.0,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

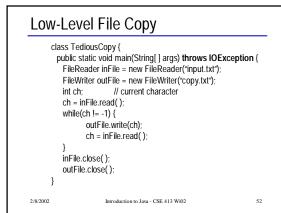
Introduction to Java - CSE 413 Wi02

49

2/8/2002

basic Reader/Writer Operations
e and the second second

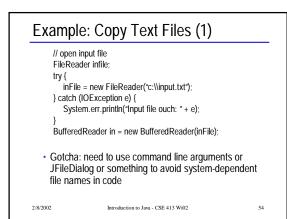


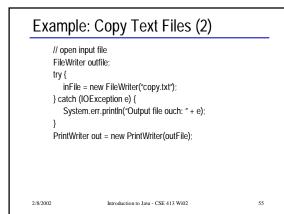


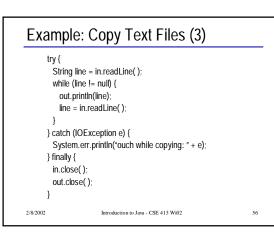


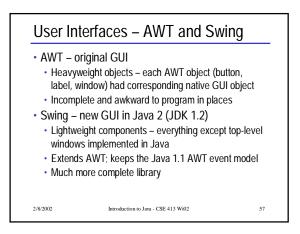
- Wrapper classes data read from or written to basic source/sink stream objects; the wrapper objects transform the stream
- Classes available to handle newlines transparently
- BufferedReader method ReadLine()
   Returns string with next line of input, or null if EOF
- PrintWriter methods print and println
   Overloaded for primitive types and String
- println emits end-of-line appropriate for host system after data written

2/8/2002 Introduction to Java - CSE 413 Wi02









#### Components & Containers

- Every AWT/Swing class ultimately extends Component
- Contains dozens of basic methods

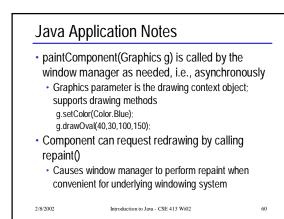
2/8/2002

- Some components are containers can contain other (sub-)components
- Top-level containers: JFrame, JDialog, JApplet
- Mid-level containers: JPanel, scroll panes, tool bars, ...
- Basic components: JButton, JLabel, text fields, check boxes, lists, file choosers, ...

Introduction to Java - CSE 413 Wi02

58

<text><code-block></code>



Introduction to Java

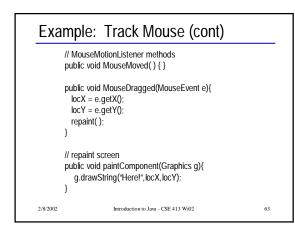
#### **Event Handling**

- 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
- Need import java.awt.event.\*;

2/8/2002

Introduction to Java - CSE 413 Wi02

Exam	ple: Track Mouse	
1 45	lic class TrackMouse extends JFrame implements MouseMotionListen	er {
in	Instance variables       t locX = 100;     // last mouse location       t locY = 100;	
// f	constructor - register this object to receive mouse move events ublic TrackMouse() {	
a } 	addMouseMotionListener(this);	
2/8/2002	Introduction to Java - CSE 413 Wi02	62



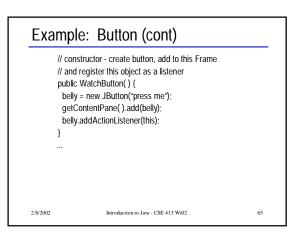
#### Example: Button

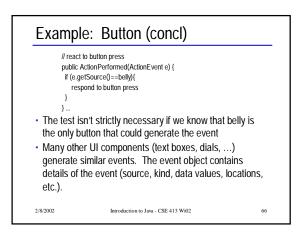
2/8/2002

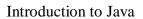
 Most user-interface components need to be allocated, added to an appropriate container, and interested objects need to register to receive events.

Public class WatchButton extends JFrame implements ActionListener { // instance variables JButton belly; // the button ...

Introduction to Java - CSE 413 Wi02









- A Layout Manager is associated with every Container. The layout manager is responsible for positioning components in the container when the container is redrawn.
- Basic layout manager classes
  - FlowLayout arranges components from left to right, top to bottom. Nothing Fancy
- GridLayout regularly spaced rows and columns
   BorderLayout Components can be placed in the Center, North, South, East, or West.
   Useful trick: to place several controls in one of these places, create a Panel containing the controls, then place the Panel in one of the 5 BorderLayout locations.
- GridBagLayout General constraint layout.
- 2/8/2002 Introduction to Java CSE 413 Wi02
- Description of the product of t

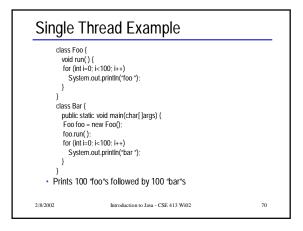
#### Threads

- Thread = Execution of one sequence of instructions (including function/method calls, conditionals, loops).
- Normal Java program executes in a thread created for main (application) or borrowed from the browser (applets).
- Class Thread can be used to create additional threads that execute concurrently.
- Each new thread is associated with (controlled by) a Thread object.

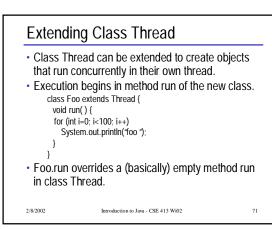
Introduction to Java - CSE 413 Wi02

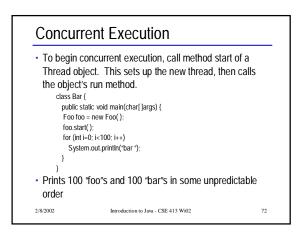
69

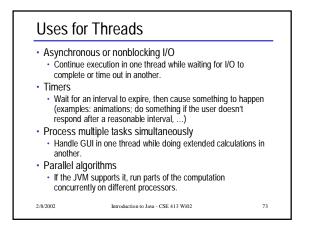
67

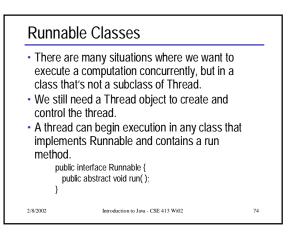


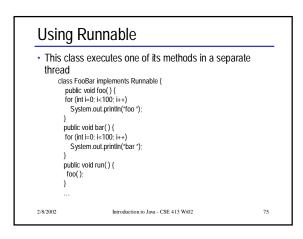


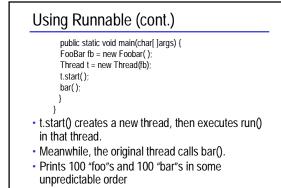












Introduction to Java - CSE 413 Wi02

76

2/8/2002

Synchronization Since threads may interleave execution in any order, we may need to control access to objects to ensure only one thread at a time can update related variables. class C { int x,y; public void setXY(int x, int x) { this.x = x; this.y = y; public int sumXY() { return x+y; } } · What happens if one thread executes sumXY while another thread is halfway through executing setXY on the same object? 2/8/2002 Introduction to Java - CSE 413 Wi02 77

#### synchronized methods

- · Every object has an associated lock
- We can require threads to acquire the lock before executing one of the object's methods by declaring the method to be synchronized.
- A synchronized method automatically acquires the object's lock when it is called. Other threads are blocked until the lock is released automatically when the synchronized method terminates.

Introduction to Java - CSE 413 Wi02

2/8/2002

