ntroduction to the Java programming language S 4354 Summer II 2014 III Seaman	<ul> <li>Free Java textbook available online</li> <li>"Thinking in Java" by Bruce Eckel, 4th edition, 2006, ISBN 0131872486, Pearson Education</li> <li>The third edition is a free electronic book: http://www.mindview.net/Books/TIJ/</li> </ul>
A simple java program	Compilation
Welcome.java	<ul> <li>To compile the program enter at the prompt (Unix or Dos):</li> <li>javac Welcome.java</li> </ul>

# Execution

• To run the program enter at the prompt (Unix or Dos):

workspace jill\$ java Welcome
Welcome to Java!
workspace jill\$

- This runs the java bytecode on a Java Virtual Machine.
- The java tool launches a Java application. It does this by starting a Java runtime environment, loading a specified class, and invoking that class's main method.
- The main method must be declared public and static, it must not return any value, and it must accept a String array as a parameter.

5

# Editions of Java

- · Different editions of java target different application environments
  - +Java Card for smartcards.
  - ◆Java Platform, Micro Edition (Java ME) targeting environments with limited resources.
  - Java Platform, Standard Edition (Java SE) targeting workstation environments.
  - ✦Java Platform, Enterprise Edition (Java EE) targeting large distributed enterprise or Internet environments.
- Each edition offers slightly different libraries (APIs) suited for the given environment.
- API: Application Programming Interface: the specification of the interface.

### Java Platform

- a bundle of related programs that allow for developing and running programs written in the Java programming language
- two distributions:
  - Java Runtime Environment (JRE) contains the part of the Java platform required to run Java programs (the JVM)
  - Java Development Kit (JDK) is for developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

6

8

### Releases of Java

- · Different releases of Java
  - +JDK 1.0 (1996) Codename: Oak
  - ◆JDK 1.1 (1997)
  - ◆J2SE 1.2 (1998)
  - ◆J2SE 1.3 (2000)
  - ◆J2SE 1.4 (2002)
  - ◆J2SE 5.0 (2004) (1.5)
  - ◆Java SE 6 (2006) (1.6)
  - ◆Java SE 7 (2011) (1.7) (I have this one)
  - ◆Java SE 8 (2014)

# Principles

- There were five primary goals in the creation of the Java language:
  - ◆It should be "simple, object-oriented and familiar"
  - It should be "robust and secure"
  - It should be "architecture-neutral and portable"
  - It should execute with "high performance"
  - ◆It should be "interpreted, threaded, and dynamic"

# Characteristics of Pure object-oriented programming

• Everything is an object.

attributes + operations

- A program is a bunch of objects telling each other what to do by sending messages
  - ightarrow a message as a request to call a method that belongs to a particular object
- Each object has its own memory made up of other objects.
  - this is how to represent complex systems
- Every object has a type.
  - $\blacklozenge$  its type is a class, the class specifies the methods of the object
- All objects of a particular type can receive the same messages.
  - ◆Even the instances of the subclasses

# Features

- · Interesting features of Java
  - +Object-oriented: everything is an object
  - ◆Inheritance
  - ◆Polymorphism: can use a subclass object in place of the superclass
  - +Garbage collection (dynamic memory allocation)
  - Exception handling: built-in error handling
  - +Concurrency: built-in multi-threading
  - ◆Persistence: support for saving objects' state between executions
  - ◆Platform independence: supports web programming

All objects in Java are really references [TIJ ch 2]

- Everything is treated as an object, using a single consistent syntax.
- However, the identifier you manipulate is actually a "reference" to an object

#### String s; //this is just a ref, a pointer

Safer to initialize a reference when you create it:

### String s = "asdf";

• Usually you use "new" to create new objects:

### String s = new String("asdf");

• Note: references are on the run-time stack, objects are in heap.

11

9

# Special case: primitive types

- These are NOT references, not objects
- They are stored on the run-time stack
- Size is not machine-dependent, always the same

Primitive type	Size	Minimum	Maximum	Wrapper type			
boolean	—	_	—	Boolean	Wrapper: object that		
char	16-bit	Unicode 0	Unicode 2 <sup>16</sup> - 1	Character	contains the primitive		
byte	8-bit	-128	+127	Byte	char c = 'x'; Character C =		
short	16-bit	-2 <sup>15</sup>	+215-1	Short	new Character(c);		
int	32-bit	-2 <sup>31</sup>	+2 <sup>31</sup> -1	Integer			
long	64-bit	-2 <sup>63</sup>	+2 <sup>63</sup> -1	Long			
float	32-bit	IEEE754	IEEE754	Float			
double	64-bit	IEEE754	IEEE754	Double			
void	_	_	—	Void			

13

# Classes in Java, fields

- A Class defines a type with fields (data) and methods (operations)
- · Fields can be objects or primitives

```
class ClassA {
  int i;
  Weeble w;
}
```

· Can create an object of this class using new:

ClassA a = new ClassA();

· Fields are accessible using dot operator

```
a.i = 11;
a.w = new Weeble();
```

# Arrays in Java

- An array is ALWAYS initialized to default values (see slide 16) +cannot access uninitialized elements by mistake
- Arrays have bounds checking

+unable to access memory outside its block (using the array): runtime error

- This is to enforce safety (though it requires overhead)
- · Arrays are objects, contain primitives or references to objects

member length returns size of array

✦can access elements using [x]

```
Weeble[] c = new Weeble[4];
for(int i = 0; i < c.length; i++)
 if(c[i] == null) // test for null reference
    c[i] = new Weeble();
```

14

# Default values for fields

• If you provide no explicit initialization to instance variables, they will be assigned the following default initial values

Туре	Default Value
boolean	false
byte	(byte) 0
short	(short) 0
int	0
long	0L
char	\u0000
float	0.0f
double	0.0d
object reference	null

• These apply to fields (and array elements), not to local variables.

## Classes in Java, methods

- Methods in Java determine the messages an object can receive.
- They are functions that the object can execute on itself

```
• Syntax is very similar to C++
class ClassA {
    int i;
    Weeble w;
    int mult (int j) {
        return i*j;
    }
}
```

· Methods are accessible using dot operator

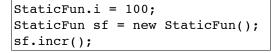
```
ClassA a = new ClassA();
a.i = 10;
int x = a.mult(4);
```

### static keyword

- When a field or method is declared static, it means that data or method is not tied to any particular object instance of that class
- · Instances of the class share the same static fields
- · Static methods may not access non-static fields

```
class StaticFun {
  static i = 11;
  static void incr () { i++; }
}
```

• Static fields and methods may be accessed without instantiating any objects by using the class name, or from an existing object.



# Accessing classes from libraries

- In Java libraries, elements are grouped into packages
- Packages have dotted path names (like internet domains)
- To use a class from a package, import the qualified class name:

### import java.util.ArrayList;

• Or import the entire package:

import java.util.\*;

### A Java program

```
// HelloDate.java
import java.util.*;
public class HelloDate {
   public static void main(String[] args) {
     System.out.println("Hello, it's: ");
     System.out.println(new Date());
   }
}
```

• Standalone program: one class must have same name as file. that class must have a main method with signature as above.

- args are for command line arguments.
- · public means method is available outside the file
- comments: /\* ... \*/ or //...to end of line

17

# Java library documentation

Online documentation for Java 1.7 API

#### http://docs.oracle.com/javase/7/docs/api/

- · java.lang is always implicitly loaded
  - System class, contains out field (a static PrintStream)
  - ♦PrintStream has overloaded println methods
- · Look for Date in the online documentation
  - ✦java.util.Date
  - +shows constructor and other methods in documentation

### Javadoc

- javadoc: a tool to extract comments embedded in source code and put them in a useful form:
  - +HTML files, viewable from a browser.

◆Can regenerate the HTML files whenever the comments/code change.

- Uses a special comment syntax to mark the documentation inside the source code
- javadoc also pulls out the class name or method name that adjoins the comment(s).
- html files are similar to the online Java API documentation.
- Purpose is to document the public **interface**: the class names and public methods.

21

### Javadoc syntax

• The javadoc commands occur only within / \*\* ... \*/ comments

◆Note the initial double asterisks.

• Each javadoc comment must precede the class definition, instance variable definition or method definition that it is documenting.

```
/** A class comment */
public class DocTest {
    /** A variable comment */
    public int i;
    /** A method comment */
    public void f() {}
}
```

- The javadoc comments may contain the following:
  - ◆embedded html code, especially for lists and formatting code snippets
  - "doc tags": special keywords that begin with @ that have special meaning to the javadoc tool.

# Javadoc tags

• This table summarizes the more commonly used tags.

TAG	USED WHERE	PURPOSE
@author <i>name</i>	Interface and Classes	Indicates the author of the code.
@since version	Interfaces and Classes	Indicates the version item was introduced.
<pre>@version description</pre>	Interfaces and Classes	Indicates the version of the source code.
@deprecated	Interfaces, Classes and Methods	Indicates a deprecated API item.
@param name description	Methods	Indicates the method's parameters.
@return description	Methods	Indicates the method's return value.
@throws name descripion	Methods	Indicates exceptions the method throws.
@see Classname	All	Indicates additional class to see.
@see Classname#member	All	Indicates additional member to see.

24

```
/**
 * A Container is an object that contains other objects.
 * @author Trevor Miller
 * @version 1.2
 * @since 0.3
 */
public abstract class Container {
    /**
     * Create an empty container.
     */
    protected Container() { }
    /**
     * Return the number of elements contained in this container.
     * @return The number of objects contained
     */
    public abstract int count();
    /**
     * Accept the given visitor to visit all objects contained.
     * Oparam visitor The visitor to accept
     */
    public abstract void accept(final Visitor visitor);
   /**
     * Determine whether this container is empty or not.
     * @return <CODE>true</CODE> if the container is empty:
     * <CODE>count == 0</CODE>, <CODE>false</CODE> otherwise
     */
    public boolean isEmpty() {
        return (this.count() == 0);
    3
}
```

# Javadoc: generating the html files

- Use the javadoc command (from the JDK) to produce the html files: javadoc -d api Container.java
- · The -d option indicates a target directory for the html files
- · Generates multiple .html files
- click on api/Container.html to see the result.
- For more details on javadoc, follow the javadoc links on the class website "readings" page:

#### http://cs.txstate.edu/~js236/cs4354/readings.html

### Operators in Java [TIJ ch 3]

• Mathematical operators, same as C++

+ = \* / % ++ --+= -= \*= /= %=

✦integer division truncates, like C++

· Relational operators yield boolean result (not int)

< > <= >= == !=

- ◆== over objects tests the value of the reference (the pointers)
- Logical operators
- String + is concatenation: "abc" + "def"

88 | !

this yields a new String object: "abcdef"

# Assignment in Java

• Assignment in Java is like in C++

+For primitive types, values are copied

int a; a = 10;

For objects, the reference is copied so both variables refer to the same object.

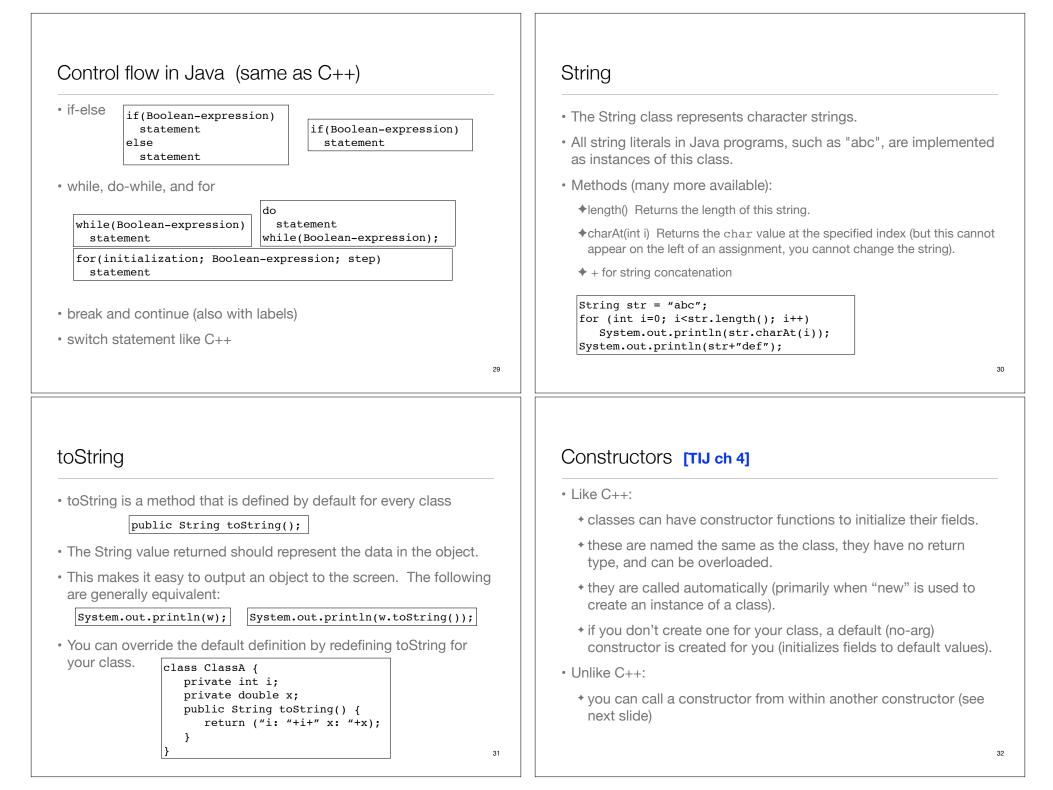
```
Weeble b = new Weeble();
Weeble a;
a = b; // a and b refer to same Weeble object
```

+changes to a will also affect b

· Objects are passed by reference by default

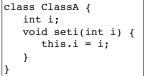
27

25



### this

- The this keyword—which can be used only inside a method produces a reference to the object the method has been called on.
  - + in Java it's a reference, not a pointer



ClassA x = new ClassA(); x.seti(10); { //inside seti, "this" is equal to x

• It can also be used to call a constructor from another constructor (Unlike C++):

```
class ClassA {
    int i;
    ClassA(int i)
    { this.i = i; }
    ClassA()
    { this(0); } // calls ClassA(0)
}
```

33

### Packages: example

- To put your classes in a package called xx.myPackage:
  - Declare the package on the first line of each java file

#### package xx.myPackage;

import ....

```
public class SmallBrain { ....
```

Put all the files in package xx.myPackage in the following directory: ...src/xx/myPackage

cd ...src

Make src the current directory:

✦To compile:

✦To run:

javac	xx/myPackage/*.java	
java	xx.myPackage.ClassA	

#### Assuming ClassA contains a main method

35

# Packages [TIJ ch 5]

· Classes can be grouped into packages:

package myPackage;

import ....

public class SmallBrain { ....

Declares these classes to belong to a package called "myPackage"

package statement must come first in the file.

- Other classes (outside of myPackage) wanting access to SmallBrain must import myPackage, or fully specify it: myPackage.SmallBrain.
- This is a mechanism to manage name spaces: this code will work with another package that has its own SmallBrain class.
- Anytime you create a package, you implicitly specify a directory structure: this file should be in a directory named "myPackage"

# Access specifiers

- · keywords that control access to the definitions they modify
  - ◆public: accessible to all other classes
  - protected: accessible to classes derived from (subclasses of) the class containing this definition as well as other classes in the same package.
  - **package** (unspecified, default): accessible only to other classes in the same package
  - **private**: accessible only from within the class in which it is defined

# The final keyword

- Java's final keyword has slightly different meanings depending on the context, but in general it says "This cannot be changed."
- Data
  - ◆To create named constants (primitive type):

#### public static final int VAL\_THREE = 39;

- ◆Use static so the class does not recreate it for each instance
- If you create an object that is final, it only means the reference cannot change, but the contents of the object itself could

private final Value v2 = new Value(22);

Cannot assign v2 to something else, but you could change its fields

v2.setValue(25);

### ArrayList class [New]

• Must specify the element types (base type) when declaring:

#### ArrayList<String> list = new ArrayList<String>(20);

◆20 is the initial capacity

The base type must be a class (NOT primitive type).

- Basic methods:
  - \*add(BaseType x) Appends the specified element to the end of this list. Starts at position 0, increases capacity as necessary.
  - ◆get(int i) Returns the element at the specified position in this list.
  - +size() Returns the number of elements in this list (not the capacity).
  - remove(int i) Removes the element at the specified position in this list, and closes the gap.

38

array vs. ArrayList

- array elements can be any type, ArrayList must contain objects.
- ArrayList can increase in size as needed (array size cannot be changed).
- ArrayList implements a "partially filled array" automatically. For an array, you must manage the size and implement "add" and "remove" operations yourself.
- ArrayList can be iterated over using a "for-each" loop:

ArrayList<String> list = new ArrayList<String>(20);
//Some code here to fill the list
for (String s : list)
 System.out.println(s); //does this for each String in list

✦General syntax is: for (BaseType var : arrayList) stmt