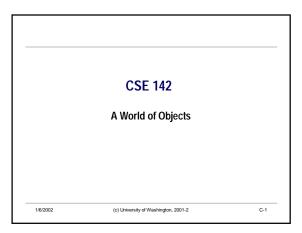
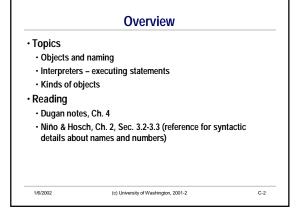
CSE142 A World of Objects





Introduction

- Java lets us build *simulations* of the world. The *things* in this simulation are called *values* or *objects*.
- Objects are just like what we think of as objects.
- · Chairs, apples, people, desks, bank accounts, cars, planes, ...
- · Objects can have parts, which are just other objects.
- $\boldsymbol{\cdot}$ The human body, a car, this room
- Objects are animated. They can respond to messages that we send them.
- Objects can be referred to by *name*. Several names can be given to the same object.

1/6/2002

(c) University of Washington, 2001-2

C-3

Number Objects

C-4

· Numbers are a simple kind of object in Java:

3 7.5

-234.657

- · Java numbers come in two main types:
- Integers
- · Rational numbers (written with a decimal point)
- · We can do arithmetic on numbers:

3 + 4 / 2

200 + 30.5 * (10 - 3.1415927)

1/6/2002 (c) University of Washington, 2001-2

Jeva: A Java Interpreter

- · Our first tool is called an interpreter
- An interpreter is similar to a human language interpreter who will translate your conversation with a speaker of another language
- · It does the following (forever):
 - · Reads what you type.
- · Translates it and executes or evaluates it.
- $\boldsymbol{\cdot}$ Prints the result for you.
- · Let's try some examples.

1/6/2002

(c) University of Washington, 2001-2

Programmer
Interpreter A Machine

(c) University of Washington, 2001-2

C-6

CSE142 Wi02 C-1

Naming Numbers

- · Sometimes we want to give names to the numbers we calculate.
- In Java we name something using this pattern (a declaration): <the type of thing> <the name> = <the thing we're naming>
- · Can then refer to the something just by using the name.
- · Types of numbers we'll be using:
 - · int, for integers
 - · double, for rational numbers
- · Examples of this declaration pattern:

int radius = 10; double pi = 3.1415927; double area = pi * radius * radius;

· Draw pictures.

(c) University of Washington, 2001-2

Shape Objects

(left x, top y, width, height)

- · Many graphics-oriented programs manipulate shapes.
- · Let's create some shapes and windows:

new Triangle() new Rectangle(200, 50, 100, 10)

new GWindow()

· We use the following patterns for creating new objects:

new <type of object>(<optional list of parts or attributes>)

Some objects, like numbers, are just written down directly, not created fresh: $3\ vs.\ new$ int()

· We usually should give newly created objects a name:

GWindow w = new GWindow();

Rectangle kaneHall = new Rectangle(50, 150, 250, 200, Color.red, true); Oval sun = new Oval(200, 50, 35, 35, Color.yellow, true);

(x, y, w, h, color, filled?)

(x, y, w, h, color, filled?)

(c) University of Washington, 2001-2

Sending Messages

- · We get objects to do things, or answer questions, or calculate results for us, by sending them messages
- · Also called invoking a method or (in other languages) calling a function
- · We use the following pattern for sending a message:

<object name> . < message name> (< optional list of parameters>)

· Examples:

sun.getX() sun . addTo (w) sun . moveBy (30, -20)

1/6/2002

(c) University of Washington, 2001-2

C-9

Drawing a Scene

· To draw a nice picture, first create a window:

GWindow w = new GWindow();

• Then create a shape object, and add it to the window:

Line horizon = new Line(50, 200, 200, 200, Color.green); (x1, y1, x2, y2, color) horizon.addTo(w);

· Create and add more shapes:

Oval sun = new Oval(100, 175, 35, 25, Color.orange, true); (x, y, w, h, c, f?)

sun.addTo(w);

Rectangle deadTree = new Rectangle(150, 150, 10, 50); (x, v, w, h)

deadTree.addTo(w);

Rectangle tallBuilding = deadTree;

1/6/2002 (c) University of Washington, 2001-2 C-10

The Inspector

- · We can peek inside of objects by using the inspector.
- · The inspector is just a Java object that knows how to look inside of other objects.
- Example:

OBrowser . inspect (sun); sun . moveBy (10, -10); OBrowser . inspect (sun);

(c) University of Washington, 2001-2

Text Objects

- · Many programs need to manipulate text, so Java provides us with Strings for this purpose.
- Examples:

String myName = "Bill Shakespeare";

String myBook = "As You Like It, or As You Wish It Were (I think?)";

myName.length()

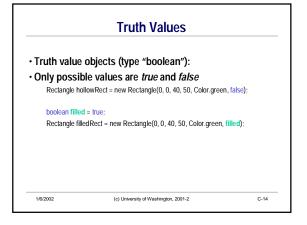
mvName.charAt(2)

myName + " wrote " + myBook

(c) University of Washington, 2001-2

CSE142 Wi02 C-2

Character Values Character objects (type "char"): char someChar = 'A'; char anotherChar = 'b'; String name = "Billy"; String anotherName = name.replace('B', 'W');



Collection Objects Many programs need to represent collections of objects. Suppose I want to build a list of students in Java. Here's one way: ArrayList students = new ArrayList(); students.add("Bob"); students.add("Jil"); int classSize = students.size(); 18/2002 (c) University of Washington, 2001-2 C-15

Java type	Used to represent	Example:
int	integers	int x = 34;
double	rational numbers	double y = 34.0;
Oval etc.	shapes	Oval sun = new Oval();
char	individual characters	char letter = 'x';
boolean	truth values	boolean filled = true;
String	text	String name = "Bill";
ArrayList	collections of things	ArrayList list = new ArrayList()

CSE142 Wi02 C-3