Lecture 2

Variables & Assignments

iClickers

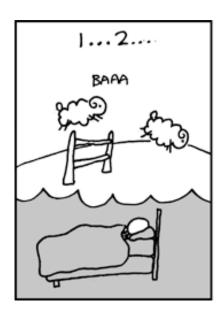
- Have you registered your iclicker?
- If not, visit
 - atcsupport.cit.cornell.edu/pollsrvc/
- Instructions on iclickers can be found here:
 - atc.cit.cornell.edu/course/polling/clickers.cfm
- Find these links on the course webpage
 - Click "Texts"
 - Scroll down on the page that opens.

Warm-Up: Demographic Question

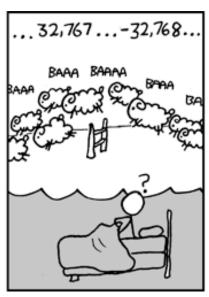
- What programming experience have you had?
 (Answers go from most experience to least)
 - A. I have seen objects and classes before
 - B. I have seen loops and conditionals, but not objects
 - C. I have seen variables and assignment statements, but that is all I remember
 - D. I know how to make a webpage in HTML, but I do not know any programming languages
 - E. I don't know anything listed above

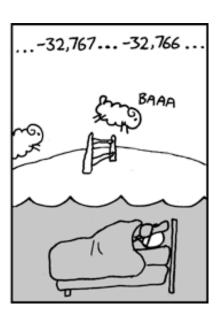
Readings for this Lecture and Previous

- Sections 1.1, 1.2, 1.3. Lab 1 will give you practice with concepts and details of 1.2, 1.3.
- PLive: Lesson 0, Lesson page 1.3, Activity 1-4.1.
- Quiz 1 in class, Tuesday 6 Sept. (more about it later)









Previously on CS 1110...

Casting: Converting Value Types

- Basic form: (type)value
 - (double) 2 casts 2 to type double. Value is 2.0 Widening cast. Java does it automatically if needed
 - (int) 2.56 casts 2.56 to type int. Value is 2 Narrowing cast. Java never does it automatically because it might lose information.
- Narrow to wide: $int \Rightarrow long \Rightarrow float \Rightarrow double$
- Other examples:
 - **double**)(int) 2.56 Value is 2.0
 - (**double**) 2.56 Value is 2.56

Type: Set of values and the operations on them

- Type boolean:
 - values: true, false
 - operations: ! (not) && (and) || (or)
 - !btrue if b is false and false if b is true
 - b && c read "b and c" true if both b and c are true, false otherwise
 - b || c,is true if b is true or c is true, false otherwise
 - i < j i == j i >= j i > j i != jevaluate to true or false ==, not =

Cannot cast to or

from int, double

read "not b"

Type: Set of values and the operations on them

- Type **String**:
 - values: any sequence of characters
 - operation(s): + (catenation, or concatenation)
- String literal: sequence of chars in double quotes
 - " abcex3\$g<&" or "Hello World!"</p>
 - String catenation: "bc" + "fg"
- + is **overloaded**: Outcome of x + y depends on type
 - If one operand (x or y) is a String, the other is converted to a String (if necessary) and catenation is done.
 - Otherwise, if one operand is a **double**, the other is cast to a double (if necessary) and a **double** addition is done.
 - Otherwise, both operands are ints and an int addition is done.

Cannot cast to or

from the other types

Operator Precedence

- What is the difference between the following?
 - **2***(1+3)

add, then multiply

2*1 + 3

multiply, then add

- Operations are performed in a set order
 - Parentheses make the order explicit
 - What happens when there aren't parentheses?
- Operator Precedence: The *fixed* order that Java processes operators in *absence* of parentheses

Precedence of Java Operators (p. 23)

- Unary operators: + -!
- Binary arithmetic: * / %
- Binary arithmetic: + -
- Arithmetic comparisons: < > <= >=
- Equality relations: == !=
- Logical and: &&
- Logical or: ||

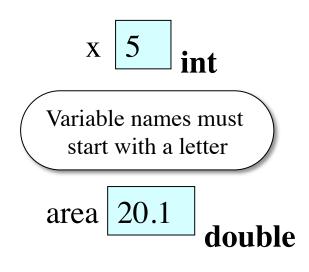
You will practice all of these in Lab 1.

Variables (p. 26)

- A variable is
 - a named memory location (box),
 - a value (in the box), and
 - a **type** (limiting what can be put in box)

Memorize definition!

Write it down several times.



Here is variable **x**, with value 5. It can contain an **int** value.

Here is variable **area**, with value 20.1. It can contain a **double** value.

• Draw variable x on piece of page B: I drew another box named x

A: I did it correctly!

C: I did something elseD: I did nothing –just watched

- Step 1: evaluate the expression x + 2
 - For x, use the value in variable x
 - Write the expression somewhere on your paper
- Step 2: Store the value of the expression in x
 - Cross off the old value in the box
 - Write the new value in the box for x
- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

You have this:

A: I did it correctly!

B: I drew another box named x

C: I did something else

D: I did nothing –just watched

Execute this command:

• Step 1: Evaluate the expression 3 * x + 1

• Step 2: **Store** its value in x

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

You now have this:

- The command:
 - Step 1: Evaluate the expression 3 * x + 1
 - Step 2: Store its value in x
- This command is called an assignment statement.
 - Tells YOU or the computer to DO something.
 - Performing it is called executing the command.
 - Command requires both evaluate AND store to be correct

You now have this:

- The command:
 - Step 1: Evaluate the expression 3 * x + 1
 - Step 2: **Store** its value in x
- This **assignment statement** is written in Java like this:

$$x = 3 * x + 1;$$
 the expression the variable

• When given an assignment statement, first **evaluate** the expression and *then* **store** its new value in the variable

• Put another variable y on your paper to get this:

• Execute this assignment:

$$y= x/y;$$

• Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

A: I did it correctly!B: I drew another box named y

C: I stored the value in x D: I did something else

Variables & As E: I did nothing – just watched

Variable Declaration (p. 26)

Memorize both these definitions!

• A *declaration of a variable* gives the **name** of the variable and the **type** of value it can contain

Write them down several times.

int x; Here is a declaration of x, indicating that it

contain an int value.

double area; Here is a declaration of area, indicating that

it can contain a double value.

Assignment Statement (p. 27)

• Execution of an assignment statement stores a value in a variable

To execute the assignment

<var>= <expr>;

evaluate expression <expr> and store its value in variable <var>

x = x + 1; Evaluate expression x+1 and store its value in variable x.

Initialization: Declaration+Assignment

Can combine declaration and assignment

int x = 3;

Here is a declaration of x, indicating that it

contain an int value.

It starts with a value of 3.

double area = 2.3;

Here is a declaration of area, indicating that

it can contain a **double** value.

It starts with a value of 2.3.

- This is called **initializing** the variable.
 - As a rule it is good to initialize all declarations.
 - Will see what happens if you do not, later.

Quiz(es) Next Week

- Click on "quizzes" on webpage for information
- Quiz 0. Complete by Sunday, January 29.
- Quiz 1. In class, Tuesday, January 31. You need to know
 - 1. Definition of "type" (see p. 7 of text)
 - 2. How to execute the assignment statement (p. 28)