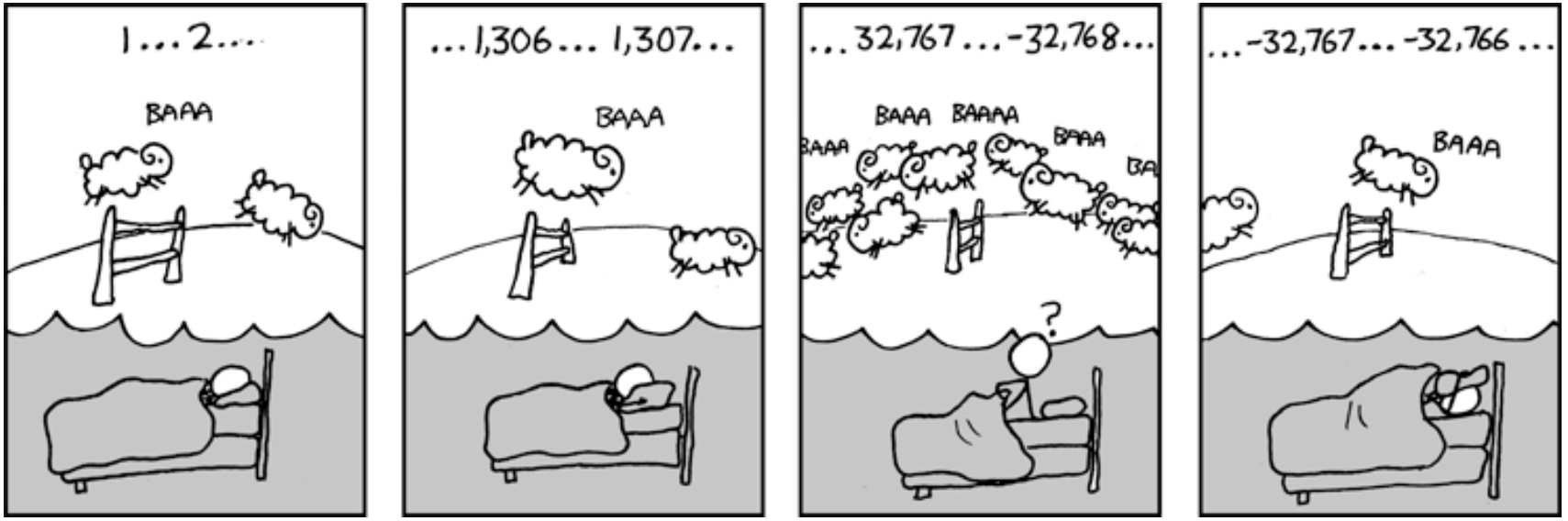
Readings for this Lecture and Previous

- 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)



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 ⇒ long ⇒ float ⇒ 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)

Cannot cast to or from **int**, **double**

  - !b read "not b"

    true 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, read "b or c"

    is true if b is true or c is true, false otherwise
  - i < j i <= j i == j i >= j i > j i != j

    evaluate to true or false

==, not =

Type: Set of values and the operations on them

Cannot cast to or from the other types

- 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!”
  - 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 **int**s and an **int** addition is done.

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**: ||

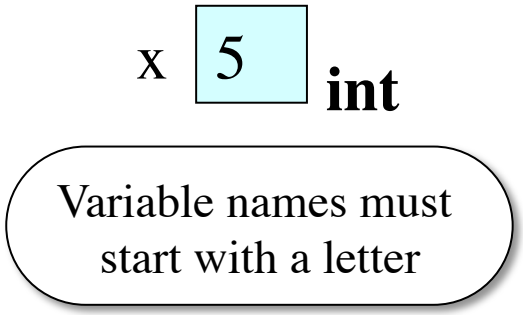
You will practice all of these in Lab 1.

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.

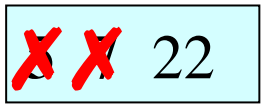


Here is variable **x**, with value 5.
It can contain an **int** value.

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

Exercise: Understanding Assignment

- You now have this:

  x  **int**

- 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

Exercise: Understanding Assignment

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

x 22 **int** y 7 **int**

- 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
E: I did nothing – just watched

Variable Declaration (p. 26)

Memorize both these definitions!

Write them down several times.

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

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)