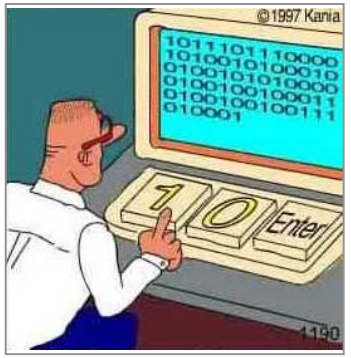
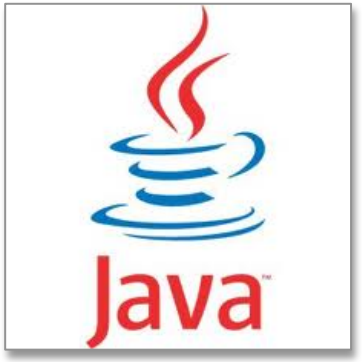
Lecture 1

# Course Overview
# Types & Expressions

# CS 1110 Spring 2012: Walker White

- **Outcomes:**
  - Basics of (Java) procedural programming
    - Usage of assignments, conditionals, and loops.
    - Ability to write recursive programs.
  - Basics of (Java) object-oriented programming
    - Ability to write programs using classes.
  - Fluency with Java and the Java API
    - Knowledge of base classes and specifications.




- **Website:**
  - www.cs.cornell.edu/courses/cs1110/2012sp/

# Intro Programming Classes Compared

## CS 1110: Java

- No prior programming experience
- No calculus
- Non-numerical problems
- Later assignments:
  - processing images
  - making games

## CS 1112: Matlab

- No prior programming experience
- One semester of calculus
- Math & engineering-type problems

CS 1110 is required for all later CS courses

# Related Courses

## CS 1130: Transition to OO (using Java)
## CS 1132: Transition to Matlab

- Require previous experience with programming
- Self-paced, 1-credit
  - (4 weeks), S/U
- Engineers take
  - CS1110–CS1132 or
  - CS1112–CS1130

## CS 2110: Computers & Programming

- Also in Java
- More advanced OO topics
- **Prerequisite**: CS 1110 or CS 1130.

Could move to CS 1130 if this course is too easy

# Helping You Succeed: Class Resources

- **Section/labs.** In the ACCEL Lab, Carpenter Library 2nd floor. Guided exercises on computer, with TA and consultants walking around, helping. **Mandatory**.
- **Quizzes.** Let you know what material is important for you to know at that point. You will know exactly what the quiz will cover. Everyone expected to get A on each quiz.
- **Lectures:** Not just slides. See interactive demos almost every lecture. We try to make it interesting.
- **Course text:** (Optional) CD at back of book has 250 2-4 minute lectures. CD missing? Using Mac OS X? See course website.
- **One-on-one sessions beginning 3rd week.** Work for 30 minutes with Walker White, TA, or consultant on the computer

# Helping You Succeed: Class Resources

- "**Interludes**." Discussion of some aspect of computing, internet, or CS to help you understand the computing world we live in.
- **AEW Workshops**. 1 credit, 2 hours. No homework. Small, collaborative classes parallel to course. No class first week. See link on course website, talk to advisors in Olin 167.
- **iClickers**. Everyone: get your own clicker. **By Thursday**. We use them to judge the sense of understanding of the class, to encourage staying alert. Part of your participation grade.
- **Piazza.** Our "town square", a place to ask and answer questions.

Course Management System. Visit cms.csuglab.cornell.edu/
Not listed? Email Betsy Appleton, ea86@cornell.edu, ask to add you to CS 1110 CMS. Include your Cornell netid in your email.

# Assignments

- Larger programming tasks (every two weeks)
- First assignment requires **mastery**
  - Submit, get feedback, resubmit, … until correct
  - Everyone eventually scores 10/10
- Later assignments designed to be fun
  - A4: Color models
  - A5: Graphics and drawing
  - A6: Image processing (jpeg, png…)
  - A7: Breakout game

# Other Course Issues

- Recitations (Labs) in the Engineering ACCEL LAB
  - **ACCEL Lab**: into (old) Engineering Library, walk straight until you come to a staircase on your left, go up the stairs.
  - Register for ANY section. But go to one you want.
  - Times of the recitation-labs: Attend ONE of them.
    - Tuesday: 12:20, 1:25, 2:30, 3:35
    - Wednesday: 12:20, 1:25, 2:30, 3:35

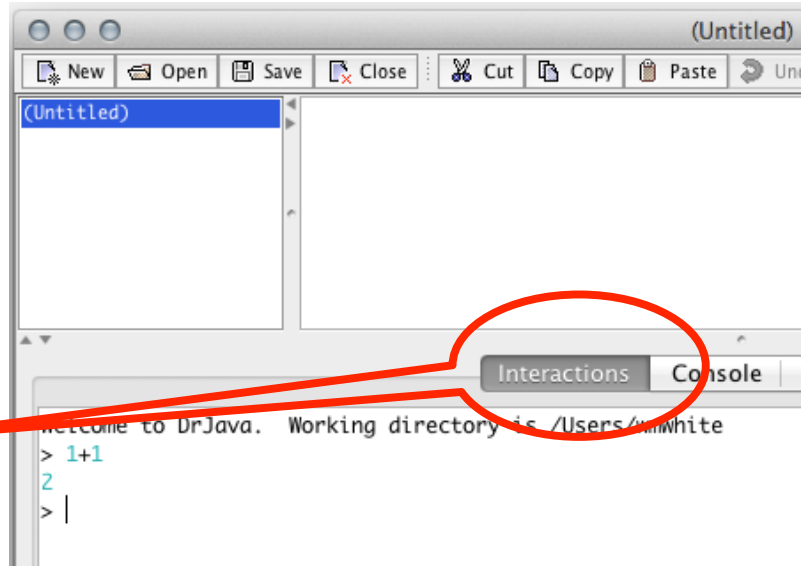
- **Academic Integrity**
  - We ask you not to cheat, in any way, shape, or form. In return, we try to be fair about the amount of work, in grading the work, and in giving you a course grade. See website for more information
  - **Do Quiz 0 on Course CMS.**

# Reading for This and Next Lecture

- **Reading:** Sections 1.1, 1.2, 1.3.
  - Lab 1 will give you practice with concepts in 1.2.
  - May not understand all the reading because a lot of new terms, but doing the reading will enhance next lecture.
- **Plive** (optional)**:** Lesson 0; also Page 1.3, Activity 1-4.1
- Summary of lectures all on course website
- **Today and Tuesday:**
  - Introduce our IDE, DrJava
  - Introduce types, expressions, and assignments
  - Learn to use DrJava as a "fancy calculator"

# DrJava: An IDE for Java

- **IDE:** Integrated Development Environment
  - Makes programming easier
  - Other IDEs: Eclipse, NetBeans
- Analogy: Web Design Tools
  - Could just write pure HTML
  - But design tools make easier
- **DrJava:** Interactions pane
  - Works like a calculator
  - Allows us to get started quickly
  - But you still have to understand **types**



# How Java Represents Data

- **Everything** on a computer reduces to numbers
  - ASCII codes convert letters to numbers
  - Pixel colors are just numbers (red, blue, green)
  - So how can Java tell all these numbers apart?

**Memorize this definition!**

**Write it down several times.**

- **Type:**
  **A set of values and the operations on them.**
  - Examples of operations: +, /, *
  - The meaning of these depends on the type

# Type: Set of values and the operations on them

- Type **integer**:
  - values: …, –3, –2, –1, 0, 1, 2, 3, 4, 5, …
  - operations: +, –, *, /, unary –
- Type **int**: A **FINITE** set of integers
  - values: –2147483648, –2147483647, …, –3, –2, –1, 0, 1, 2, 3, 4, 5, …, 2147483646, 2147483647
  - operations: +, –, *, % (7 % 3 = remainder when dividing 7 by 3), /, unary –
- Principal: These **int** operations must yield an **int**.
  - **Example:** 1 / 2 rounds toward 0
- **Bounds:** Integer.MIN_VALUE-1, Integer.MAX_VALUE+1

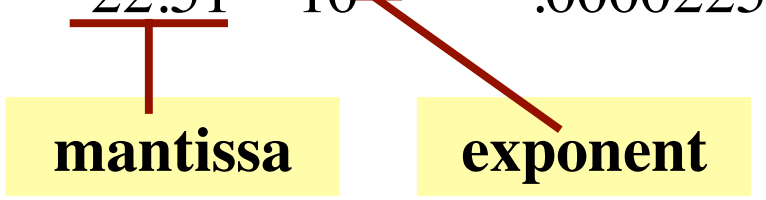
Why this restriction?

$-2^{31} .. 2^{31}-1$

# Type: Set of values and the operations on them

- Type **double**:
  - values: Numbers in scientific notation, e.g.

22.3 $-22.51 * 10^{6}$ .00002251 $22.51 * 10^{-6}$




- **Important points:**
  - Only approximations to real numbers; can't represent all of them.
  - We must distinguish between type **int** and **double**!
    **double** numbers always have period or exponent (e.g. 2.0).
  - Can't use notation $10^{6}$. Exponents have to be written differently.

# Type: Set of values and the operations on them

**mantissa** **exponent**

- Type **double**:
  - values: –22.51E6 equivalent to –22510000
    or $-22.51 * 10^6$
    22.51E–6 equivalent to .00002251
    or $22.51 * 10^{-6}$
  - An **approximation** to the real numbers.
  - operations: +, –, *, /, unary –
    - 1.0/2.0 is 0.5 not 0. Operation / behaves *differently* for **double**
- Double.MIN_VALUE 4.9E-324 — **Smallest POSITIVE value**
- Double.MAX_VALUE 1.7976931348623157E308

# Aside: Why Is It Called double?

- Historically, "real number" type was **float**
  - Short for "floating point number"
  - Increasing/decreasing the exponent "floats" the decimal place left or right
- Mantissas are only allowed so many digits
  - Floats have **very small** mantissas

    111,111.11 ✔ 1,111,111.11 ✗

  - Doubles allow for double the digits!

# 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)
  - !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

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

==, 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

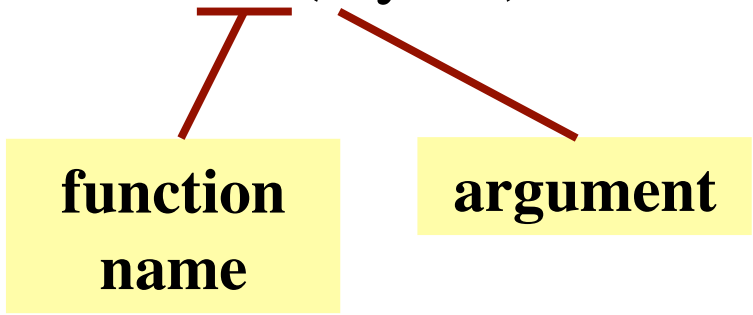
# Precedence of Java Operators (p. 23)

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

You will practice all of these in Lab 1.

# Function Calls

- Java supports functions as well
  - A function in an expression is *function call*
  - Will explain the meaning of this later
- Function expressions have the form **fun**(x,y,…)




- Basic Java math functions have prefix Math.
  - **Examples**: Math.sin(2.0)  Math.round(3.5)
  - Experiment with this during the first lab