1.00 Lecture 3

Operators, Control

Reading for next time: Big Java: sections 6.1-6.4
Skip all the advanced topics, hints, etc.

Main()

- **About main():**
  - **In each Java program there is a just a single main() method, no matter how many classes there are.**
    - **The main() method is often in a class that has no other methods, by convention. It can be in any class, though some choices would seem unnatural.**
  - **main() tells Java where to start the program; it's just a naming convention**
    - **It could easily have been called "StartHere"**
  - **In early examples we have only one class, so it will seem there's a main() method in each class. Nope.**
  - **Main() at a later point will be minimalist:**
    - **Main() does the least possible work to get the program running and then hands off all the remaining work to objects and their methods.**
    - **For now, since we haven't covered classes and objects, we'll do everything in main() for a little while longer.**

Logical Operators

- Produce results of type boolean
- Comparisons use 9 operators:

Equal	==	Not equal	!=
Less than	<	Less than or equal	<=
Greater than	>	Greater than or equal	>=
Logical and	&&	Logical or	||
Not	!		

```
// Example
int c= 0, b= 3;
if (c != 0 && b/c > 5) System.out.println("Buy this stock");
// Short circuit evaluation: quit after answer determined
boolean buy= true;
if (!buy || c == 0) System.out.println("Nah, don't buy");
```

Assignment Operators

- Assignment is not the same as equality
  - = is not the same as ==
- Assignments are expressions:

```
int x, y;
x= y= 5;            // Same as x = (y= 5); assoc from R to L
```

- Shortcut forms exist:

```
int x= 5, y= 3;
x += y;             // Same as x= x + y;
// This means take current value of x (5), add y (3), and
// set x to a new value of 8
```

- Shortcut forms include +=, -=, *=, /=, %= :

```
x /= y;             // Same as x= x / y;
x %= y;             // Same as x= x % y;
```

- Other shortcut forms are ++ and -- :

```
x++;                // Same as x= x + 1;
y= --x;             // Same as x= x-1; y = x; (More later)
```

Operator exercise

- **Create a new class VelocityTest**
  - Your main program will compute train velocities from Boston to New York with various improvements
  - On the very first line of your program write:
    ```
    import javax.swing.*;          // Allow GUI input
    ```
  - Accept an int input from the user, in main():
    ```
    String input= JOptionPane.showInputDialog("Enter time");
    int time= Integer.parseInt(input);    // Enter 4 (hrs)
    ```
  - Define double d= 225; // miles
  - Decrease d by 25
  - Compute velocity v
  - Print whether v > 60: System.out.println(logical expr);
  - <u>If you have time:</u>
  - Decrement time by 1
  - Print whether v > 60 and d < 225
  - Print whether v > 70 or d < 175 or time <= 3

Control Structures: Branch

General form	Example

if (boolean)
  statement;
	if (psgrs == seats)
	  carFull= true;
	if (psgrs >= seats) {
	  carFull= true;
	  excess= psgrs - seats; }

if (boolean)
  statement1;
else
  statement2;
	if (psgrs >= seats) {
	  carFull= true;
	  excess= psgrs - seats; }
	else
	  carFull= false;

if (boolean1)
  statement1;
…
else if (booleanN)
  statementN;
else
  statement;
	if (psgrs < seats)
	  carFull= false;
	else if (psgrs == seats) {
	  carFull= true;
	  excess= 0; }
	else {
	  carFull= true;
	  excess= psgrs - seats; }

Control exercise

- Create a class ControlTest
- Write in main():
  - Declare and initialize five double variables d, s, p, a and b
    - d= 100
    - s= 50
    - p = 10
    - a= .1
    - b= .2
  - Then write code so that:
    - If demand d > supply s, raise price p by a(d-s)
    - If demand == supply, do nothing
    - If demand d < supply s, lower price p by b(d-s)
  - If you have extra time, read s from a JOptionPane

Control structure: Iteration

General form	Example

while (boolean)
  statement;
	while (balance < richEnough) {
	  years++;
	  balance *= (1+ interestRate);
	}

do
  statement;
while (boolean);
// Always executes stmt at least once
	do {
	  years++;
	  balance *= (1+ interestRate);
	} while (balance < richEnough)

for (start_expr; end_bool; cont_expr)
  statement;
	for (years= 0; years< 20; years++) {
	  balance += (1+ interestRate);
	  if (balance >= richEnough) break;
	}

For loops

for (start_expr; end_bool; cont_expr)
  statement;
	for (yrs= 0; yrs < 20; yrs++)
	  balance *= (1 + rate);

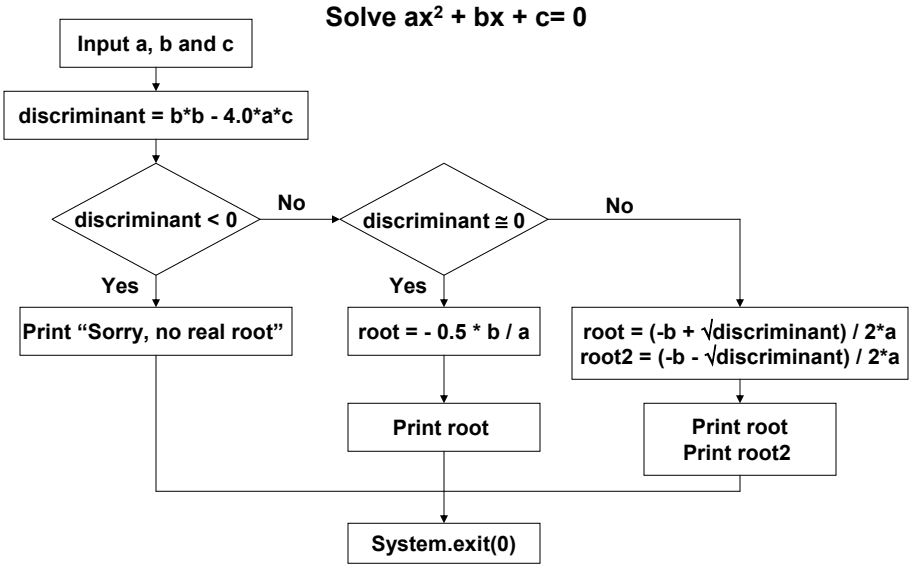
is equivalent to:	

start_expr;
while (end_bool) {
  statement;
  cont_expr;
}
	yrs= 0;
	while (yrs < 20) {
	  balance *= (1+rate);
	  yrs++;
	}

Iteration Exercises

- **Create a class IterationTest**
  - **Exercise 1: Write code in main() that prints out every third number between 11 and 47, including 11 and 47.**
  - **Exercise 2: Also print out whether each number output is odd or even.**
  - **Remember to declare the variables you use in your loops before you loop (e.g., `int i;`)**
- **If you finish, look at the next example**
  - **Find the bug**

Control example



Control example

```
import javax.swing.*;                  // To support simple input
public class Control {                 // Quadratic formula
    public static void main(String[] args) {
        final double TOL= 1E-15;       // Constant (use 'final')
        String input= JOptionPane.showInputDialog("Enter a");
        double a= Double.parseDouble(input);
        input= JOptionPane.showInputDialog("Enter b");
        double b= Double.parseDouble(input);
        input= JOptionPane.showInputDialog("Enter c");
        double c= Double.parseDouble(input);
        double discriminant= b*b - 4.0*a*c;
        if ( discriminant < 0)
            System.out.println("Sorry, no real root");
        else if (Math.abs(discriminant) <= TOL) {
            double root= -0.5 * b / a;
            System.out.println("Root is " + root); }
        else {      // Redefine 'root'; blocks have own scopes
            double root=(-b + Math.sqrt(discriminant))/ (2.0*a);
            double root2=(-b- Math.sqrt(discriminant))/ (2.0*a);
            System.out.println("Roots: " + root + " , " + root2);  }
        System.exit(0);  } }
```

Control example

- The previous program has a deliberate, subtle bug
  - Can you see it?
  - Is it likely that you'd find it by testing?
  - Is it likely you'd find it by using the debugger and reading the code?
- Fix the error by rearranging the order of the if-else clauses
- By the way, this is a terrible way to solve a quadratic equation—see Numerical Recipes, section 5.6

Example Method-Computing ln(x)

- The natural logarithm of any number x can be approximated by the formula

$$\ln(x) = (x-1) - (x-1)^2/2 + (x-1)^3/3 - (x-1)^4/4 + (x-1)^5/5 + \ldots\ldots$$

- The next two examples show computing this series with a for loop and a do loop
  - This is also a terrible way to compute a logarithm!

Iteration Example 1: Ln x

```
import javax.swing.*;

public class Iteration {
    public static void main(String[] args) {
        String input= JOptionPane.showInputDialog("Enter x (0-2)");
        double x= Double.parseDouble(input);
        // Compute 20 terms of
        // ln x= (x-1) - (x-1)^2/2 + (x-1)^3/3 - ...
        final int ITERATIONS= 20;     // Fixed no of iterations
        double logx= 0.0;
        double x1= x-1;
        for (int i= 1; i <= ITERATIONS; i++) {
            if (i % 2 == 0)                  // i even
                logx -= Math.pow(x1, i)/i;
            else
                logx += Math.pow(x1, i)/i;  }
        System.out.println("Ln x= " + logx);  }  }
```

Iteration Example 2: Ln x

```
import javax.swing.*;              // Same series as example 1
public class Iteration2 {
    public static void main(String[] args) {
        String input= JOptionPane.showInputDialog("Enter x (0-2)");
        double x= Double.parseDouble(input);
        final double TOLERANCE= 0.00001;  // Tol sets no of terms
        double logx= 0.0;
        double x1= x-1;
        int i= 1;
        double term= 0.0;               // Define outside do {}
        do {
            term= Math.pow(x1, i)/i;
            if (i % 2 == 0)                     // i even
                logx -= term;
            else
                logx += term;
            i++;
        } while (Math.abs(term) > TOLERANCE);
        System.out.println("Ln x= " + logx);
        System.out.println("Found in " + i + " iterations"); }  }
```