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. Not so.
- main() at a later point in the term 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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal</td>
<td>==</td>
<td>Not equal</td>
</tr>
</tbody>
</table>
| Less than| <      | Greater than| >
|          | <=     |              |
|          | >=     |              |
| Logical and | &&    | Logical or   |
| Not      | !      |              |

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

Assignment operators

- Assignment is not the same as equality
  - = is not the same as ==
  - Assignment places right hand side into left hand side
- Assignments are expressions:
  int x, y;
  x = y = 5;        // Same as x = (y = 5); associate 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; % gives remainder
- Other shortcut forms are ++ and --:
  x++;   // Same as x = x + 1;
  y = --x;   // Same as x = x-1; y = x;
Operator exercise

- Create a new project Lecture3
- Create a new class VelocityTest with a main method
  - We will compute train velocities from Boston to New York (which are 225 miles apart) with various improvements
  - On the very first line of your program write:
    ```java
    import javax.swing.*;  // Allow GUI input
    ```
  - Accept an int input from the user, in main():
    ```java
    String input= JOptionPane.showInputDialog("Enter time");
    int time= Integer.parseInt(input);  // Enter 4 (hrs)
    ```
  - Define double d= 225;  // Miles
  - Decrease d by 25  // Shorten route thru realignment
  - Compute velocity v
  - Print whether v > 60:  System.out.println( v>60? +______);
- If you have time to do these steps (no ifs required):
  - Decrement time by 1 and recompute v  // Faster trains
  - Print whether v > 60 and d < 225
  - Print whether v > 70 or d < 175 or time <= 3

Control structures: branch

<table>
<thead>
<tr>
<th>General form</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (boolean) statement;</td>
<td>if ( psgrs == seats)</td>
</tr>
<tr>
<td></td>
<td>carFull= true;</td>
</tr>
<tr>
<td></td>
<td>if (psgrs &gt;= seats) {</td>
</tr>
<tr>
<td></td>
<td>carFull= true;</td>
</tr>
<tr>
<td></td>
<td>excess= psgrs - seats;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>carFull= false;</td>
</tr>
<tr>
<td>if (boolean) statement1; else statement2;</td>
<td>if ( psgrs &gt;= seats ) {</td>
</tr>
<tr>
<td></td>
<td>carFull= true;</td>
</tr>
<tr>
<td></td>
<td>excess= psgrs - seats;</td>
</tr>
<tr>
<td></td>
<td>} else</td>
</tr>
<tr>
<td></td>
<td>carFull= false;</td>
</tr>
<tr>
<td>if (boolean1) statement1; ... else if (booleanN)</td>
<td>if ( psgrs &lt; seats)</td>
</tr>
<tr>
<td>statementN; else statement;</td>
<td>carFull= false;</td>
</tr>
<tr>
<td></td>
<td>else if (psgrs == seats) {</td>
</tr>
<tr>
<td></td>
<td>carFull= true;</td>
</tr>
<tr>
<td></td>
<td>excess= 0;</td>
</tr>
<tr>
<td></td>
<td>} else</td>
</tr>
<tr>
<td></td>
<td>carFull= true;</td>
</tr>
<tr>
<td></td>
<td>excess= psgrs - seats;</td>
</tr>
</tbody>
</table>

There are no semicolons after if or else clauses
Control exercise

- Create a class ControlTest with a main method
- 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*(s-d)
  - Use the debugger to step through your program:
    - Set breakpoint at first executable line in main()
    - Run-> Debug As-> Java Application
  - If you have extra time, read s from a JOptionPane

Control structure: iteration

<table>
<thead>
<tr>
<th>General form</th>
<th>Example</th>
</tr>
</thead>
</table>
| while (boolean)                 | while (balance < richEnough) {
|       statement;                |     years++;                |
|                                 |     balance *= (1+ interestRate); |
|                                 | }                           |
| do                              | do {
| statement;                     |     years++;                |
| while (boolean);                |     balance *= (1+ interestRate); |
| // Always executes stmt at least once | } while (balance < richEnough); |
| for (start_expr; end_bool; cont_expr) | for (years= 0; balance < richEnough;
| statement;                     |     years++) {
|                                 |     balance *= (1+ interestRate); |
|                                 | }                               |

There are no semicolons after while, do or for clauses
### for loops

<table>
<thead>
<tr>
<th>Start expression</th>
<th>End condition</th>
<th>Cont expression</th>
<th>Code Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>for (start_expr; end_bool; cont_expr)</td>
<td>yrs = 0; yrs &lt; 20; yrs++</td>
<td>balance *= (1 + rate);</td>
<td>yrs = 0; yrs &lt; 20; yrs++</td>
</tr>
<tr>
<td>statement;</td>
<td>balance *= (1 + rate);</td>
<td>yrs++;</td>
<td></td>
</tr>
</tbody>
</table>

### 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.
    - Use the remainder (%) operator. If remainder is 0 after dividing by 2, number is even; otherwise it’s odd.
    - Remember to declare the variables you use in your loops before you loop (e.g., int i;)
- **If you finish, look at the control example that follows**
  - Find the bug
Control example

Solve $ax^2 + bx + c = 0$

- Input $a$, $b$ and $c$
- Discriminant $= b^2 - 4.0a^c$
  - No
    - Discriminant $< 0$
      - Print "Sorry, no real root"
    - Discriminant $= 0$
      - Yes
        - Root $= -0.5 * b / a$
        - Print root
  - Yes
    - Root $= (-b + \sqrt{\text{discriminant}}) / 2*a$
    - Root2 $= (-b - \sqrt{\text{discriminant}}) / 2*a$
    - Print root
    - Print root2
- End program

Control example

```java
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
• A note on format: we compress code examples to fit on slides, by putting multiple `}}` on one line, for example. Don't do this in your code; use Eclipse to indent and format well. (ctrl-A, ctrl-I)