Boolean Expressions
Conditional Statements

Chris La Pilla
CompSci 6
Summer 2007

Boolean Expressions

• Recall that there is a primitive type boolean
• true and false are the two Boolean literals
• We can have boolean variables just like ints or doubles:
  boolean done = false;

Boolean Variables

Review: Mathematical Expressions

• Recall the mathematical operators (+, -, *, etc.)
• We use them to construct mathematical expressions by applying an operator to one or two operands:
  5 + 3.0
  -num
• All expressions have a type and a value

Review: Mathematical Expressions

• Relational operators are used to compare values:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal</td>
</tr>
<tr>
<td>==</td>
<td>Equal</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal</td>
</tr>
</tbody>
</table>

Relational Operators

Boolean Expressions

• Relational operators are used in Boolean Expressions, e.g.
  num < 5
• These work just like math expressions: they have a type (boolean) and a value (true/false)
• Can assign the result to variables, e.g.
  boolean isSmaller = num < 5;

Boolean Expressions

• Relational operators always have two operands
• Is this expression valid?
  num1 < num2 < num3
Boolean Operators

- Can chain Boolean expressions together with Boolean or Logical Operators
- ! (not) : flips the value of the operand (unary) (true → false, false → true)
- && (and) : true only if both operands are true, false otherwise
- || (or) : true if either or both operands are true, false otherwise
- Only work with boolean operands

Examples

- boolean small = false;
  boolean big = !small; //big is true

- boolean between = num2 > num1 && num2 < num3;

- boolean pass = grade == 'A' || grade == 'B' || grade == 'C' || grade == 'D';

Common mistake

- This is a syntax error:
  boolean test = num1 == 2 || 3;
- Why?

Order of operations

- Unary operators: - (minus), ! (not)
- Casts
- *, /, % (from left to right)
- +, - (from left to right)
- <, <=, >, >=
- ==, !=
- &&
- ||
- Can always group with parentheses

Floating-Point Equality

- Doubles only hold a finite number of digits
- Need to be careful when comparing doubles because of round-off error; e.g.
  double a = Math.sqrt(2);
  double aSquared = a * a;
  boolean equal = aSquared == 2;
  System.out.println(equal); //prints false!

- Instead, check to see if the numbers are really close together:
  final EPS = 1e-14;
  boolean equal = Math.abs(aSquared - 2) < EPS;
  System.out.println(equal); //prints true
String Comparisons

- `==` and `!=` will not work properly with Strings (or other objects)
- We'll discuss the reasons why next week
- Instead, use `equals` or `equalsIgnoreCase` methods:
  ```java
  String str1 = "hello";
  String str2 = "HELLO";
  boolean same = str1.equals(str2); //false
  same = str1.equalsIgnoreCase(str2); //true
  ```
- For now, only use `==` or `!=` with primitive types

Examples

- if (num1 < 10)
  System.out.println("Less than 10.");
- if (!done)  //done is boolean
  System.out.println("Not done.");
- if (str1.equals(str2))  //strings
  System.out.println("Same strings.");

Examples

- if (num1 < 10)
  System.out.println("< 10.");
  else
  System.out.println(">= 10");
- if (done)  //done is boolean
  System.out.println("Done.");
  else
  System.out.println("Not done.");

if statement

- Used to make decisions
- Syntax:
  ```java
  if (condition) 
  statement;
  ```
- Condition is a boolean variable or expression
- Statement is executed only if condition evaluates to true

if-else statement

- Syntax:
  ```java
  if (condition) 
  statement1;
  else
  statement2;
  ```
  - Executes statement1 if condition is true, otherwise executes statement 2
  - This is NOT the same as
    ```java
    if (!condition) 
    statement2;
    ```
    (why not?)

Block statement

- If we want to execute multiple statements, enclose them in curly braces `{ }`, e.g.
  ```java
  if (!validInput){
    System.out.println("Invalid input");
    System.out.print("Enter an int: ");
    value = scanner.nextInt();
  } else{
    System.out.println("Valid input");
  }
  ```
Sequences of if-else

• If-else statements can be chained together
• Example:
  
  ```java
  if (grade == 'a'){
  } else if (grade == 'b'){
  } else if (grade == 'c'){
  
  etc.
  ```

Nested if statements

• If or if-else statements can be nested:
  ```java
  if (gameOver){
    if (win){
      System.out.println("You win!");
    } else {
      System.out.println("You lose.");
    }
  }
  ```
• Always use curly braces when nesting if statements