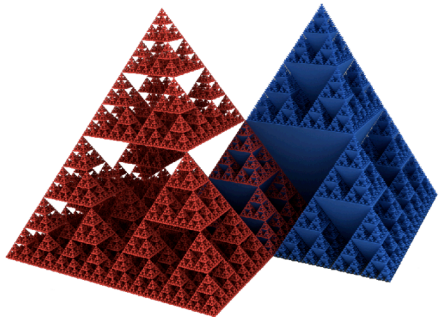
COMPSCI 101
Principles of Programming

Lecture 11

Boolean expressions (cont'd)

If statements (chapter 12)



Relational Operators

Relational Operators

- Evaluate to true or false
- Used to compare any two *primitive* values

Operator	Meaning
==	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
!=	Not equal to

~~10 == 15~~
false

~~8 < 9~~
true

~~true != false~~
true

Exercise: relational operators

(a) What is the output of the following code?

```
int a = 7;
int b = 7;
System.out.println( a != b );
System.out.println( b >= a );
```

```
false
true
```

(b) What is the output of the following code?

```
boolean b = (3 < 4);
boolean c = (3 != 4);
boolean d = (3 > 4);
System.out.println( (b && c) || (c && d) );
```

```
true
```

(c) What is the output of the following code?

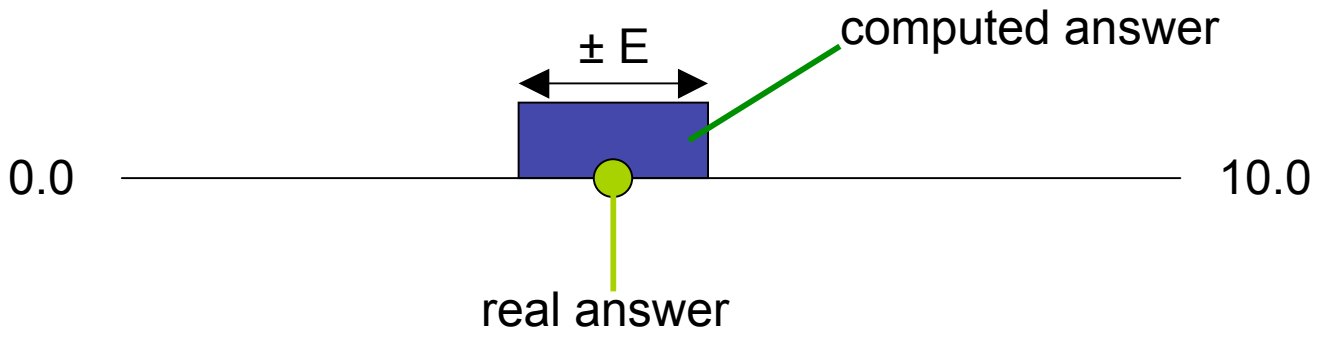
```
System.out.println( 3 == (1+2) );
System.out.println( (0.1 + 0.1 + 0.1) == 0.3 );
```

```
true
false
```

Warning: Floating point numbers

Never use the equality test to compare floating point numbers

- Floating point numbers are always stored approximately
- Use a value to represent how accurate you require the answer
- Accept all values within acceptable error limit



Operator Precedence

Operations with higher order of precedence are applied first

Higher order of precedence

Operation	Symbol
Grouping operators	`( )`
Unary operators	`+, -, !`
Multiplicative arithmetic	*, /, %
Additive arithmetic	`+, -`
Relational ordering	`<, >, <=, >=`
Relational equality	`==, !=`
Logical and	`&&`
Logical or	`||`
Assignment	`=, +=, -=, *=, /=, %=`

Example: Operator precedence

Evaluate the boolean expression given below:

```
int a = 1;
int b = 4;

boolean eval01 = a > 3 && b + 3 < 4 || true;
```

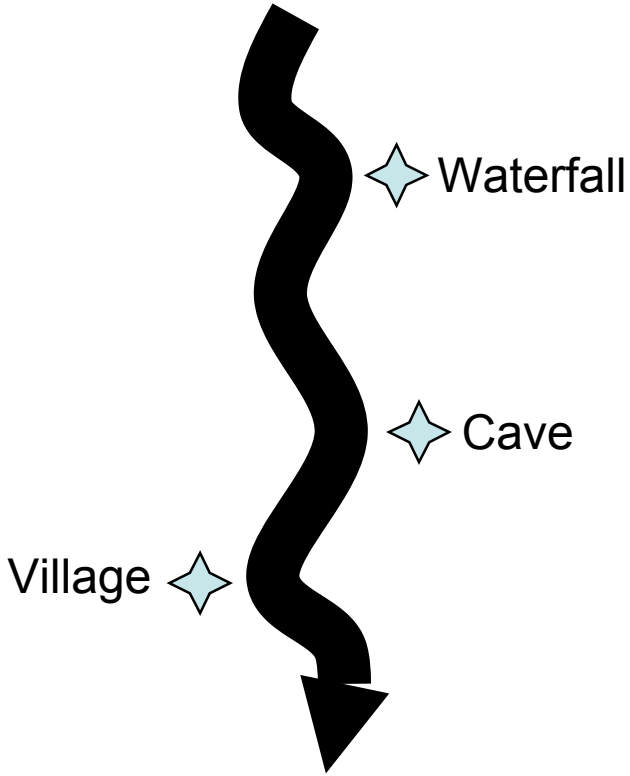
The order of precedence is arithmetic, then relational, then logical

- Arithmetic: Evaluate b+3 first (result is 7)
- Relational: Compare a > 3 (result false)
- Relational: Compare 7 < 4 (result false)
- Logical and: Combine false && false (result false)
- Logical or: Combine false || true (result true)

Statement Execution

Each statement is executed in sequential order

- If every statement is executed, then no decisions possible



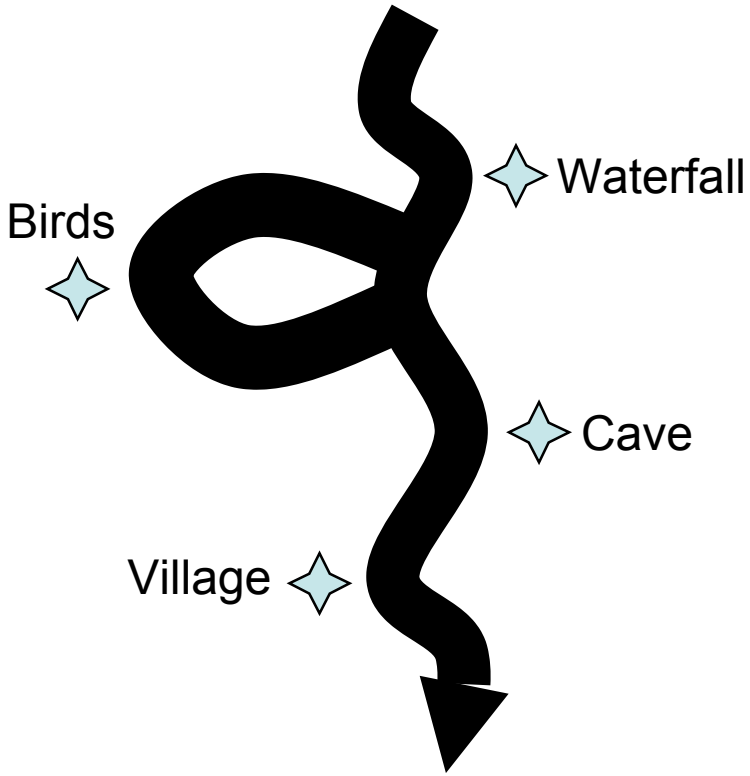

```
Visit the Waterfall

Visit the Cave

Visit the Village
```

Making a Decision

Must be able to choose which path to take




```
Visit the Waterfall

if you are interested in birds
   then Visit the Birds

Visit the Cave


Visit the Village
```

Conditional Statement

Allows us to make a decision in our programs

Formal Syntax:

```
if ( condition ) {
    statement 1;
    ...
    statement n;
}
```

Description:

- The condition must be a boolean expression (**`true`** or **`false`**)
- If the condition is true, then all statements in the block will be executed
- If the condition is false, then none of the statements in the block will be executed

Style: Testing boolean variables

Always use the boolean value directly in an if statement

- Do not compare the boolean value with **`true`** or **`false`**

Poor Style:

```
boolean b = (amount < 3) && (amount > 0);
if (b == true) {
   System.out.println("amount is either 1 or
   2");
}
```

Better Style:

```
boolean isSmallAmount = (amount < 3) && (amount > 0);
if (isSmallAmount) {
   System.out.println("amount is either 1 or 2");
}
```

Short-circuit evaluation

Evaluation of a boolean expression stops when we already know the answer (lazy evaluation)

Example: What value is stored in **`result`**?

```
boolean a = true;
boolean b = false;

boolean result = (a && !b) || ((a&&!b)&&(b||!a)||((!b||a)&&(b||!b)))
```

Short-circuit evaluation

We cannot divide a value by zero:

```
if (totalMark / numberOfLabs > 50) {
   System.out.println("You passed");
}
```

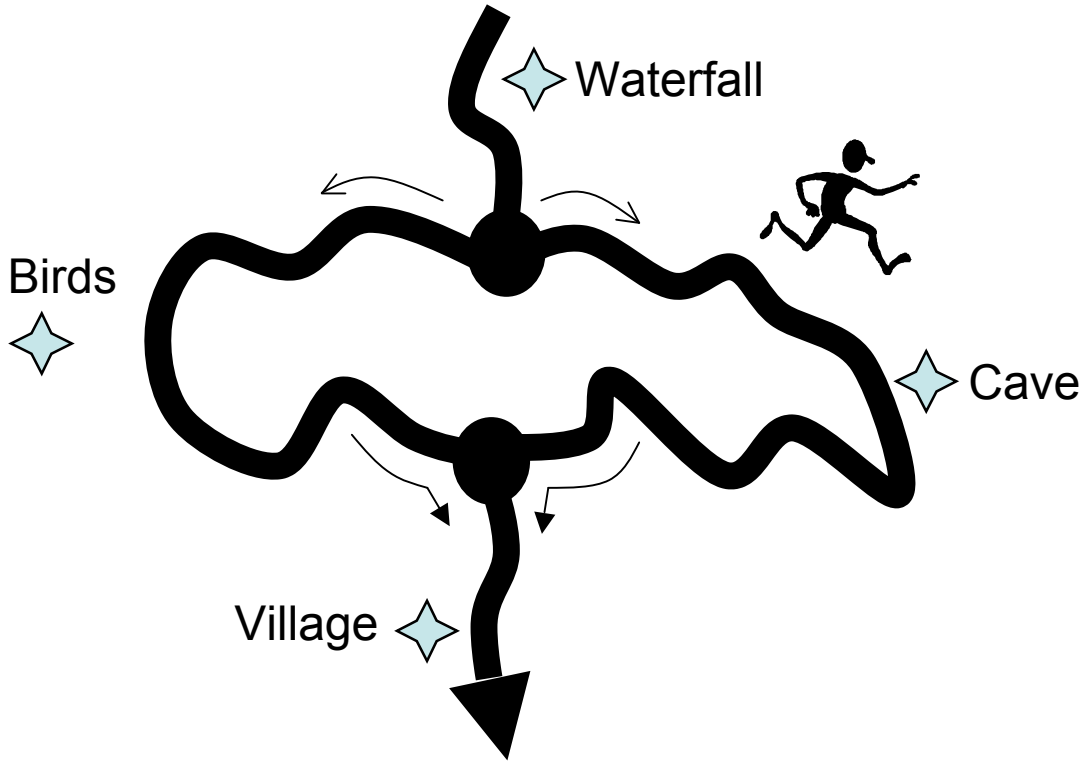
Causes an error if the number of labs is zero

Use lazy evaluation to check the value is non-zero before it is used:

```
if ((numberOfLabs > 0) && (totalMark / numberOfLabs > 50)) {
   System.out.println("You passed");
}
```

Choosing between 2 options

Choose one option or the other, but not both.



`if-else` statements

Discussed so far:

- If “something” is true then do “this”

Using an optional else clause

- If “something” is true then do “this” otherwise do “that”

```
public void start() {
        System.out.println( “Waterfall” );

        if (likesBirds) {
                System.out.println( “Birds” );
        } else {
                System.out.println( “Cave” );
        }
        System.out.println( “Village” );
}
```

Syntax for `if-else`

All if statements have an *optional* else

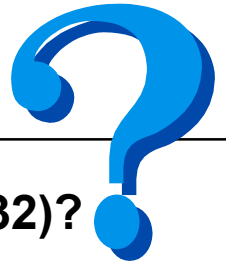
Formal Syntax:

```
if ( condition ) {
  statement_A1;
  ...
  statement_An;
} else {
  statement_B1;
  ...
  statement_Bn;
}
```

*If the condition is **true**, then statements in block A will be executed*

*If the condition is **false**, then statements in block B will be executed*

Exercise: `if-else`



What is the output of the following code (assuming the user enters 32)?

```
public class Exercise{
     public void start() {
          int number = Integer.parseInt( Keyboard.readInput() );
          if (number > 32) {
               System.out.println( "First line" );
               System.out.println("Second line");
          }
          else {
               System.out.println( "Third line" );
               System.out.println("Fourth line");
          }
          if (number < 32) {
               System.out.println( "Fifth line" );
          } else {
               System.out.println( "Sixth line" );
          }
     }
}
```

Third line
Fourth line
Sixth line

Nested ifs

The statements inside the block of an if statement can be any Java statements

- including *another* if statement
- these are known as *nested ifs*

```
int firstAge = 10;
int secondAge = 11;
int thirdAge = 12;

if (firstAge > secondAge) {
    if (firstAge > thirdAge) {
        System.out.println("The oldest age
                                        is: " + firstAge);
    }
}
```

if-else-if

Comparing a single variable with many different values

- In each case, different code executed
- Indentation causes code to shift right

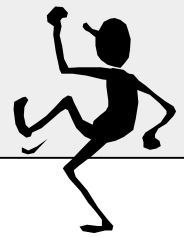
```
int year = ... ;
if (year < 1960) {
     System.out.println( "Born in the 1950s or even earlier");
} else {
     if (year < 1970) {
          System.out.println( "Born in the 1960s");
     } else {
          if (year < 1980) {
               System.out.println( "Born in the 1970s");
          } else {
               if (year < 1990) {
                    System.out.println( "Born in the 1980s");
               } else {
                    System.out.println("Born in the 1990s or later");
               }
          }
     }
}
```

Style rule: `if-else-if`

Much easier to read

- Only one of the statements is executed
- If all the conditions are false, then the last statement is executed

```
int year = … ;
if (year < 1930) {
    System.out.println( "Born in the 1920s or even earlier");
} else if (year < 1940) {
    System.out.println( "Born in the 1930s");
} else if (year < 1950) {
    System.out.println( "Born in the 1940s");
} else if (year < 1960) {
    System.out.println( "Born in the 1950s");
} else {
    System.out.println( "Born in the 1960s or later");
}
```



Using if statements to debug

Normally use System.out.println() to help us debug code

- Trace the path of execution
- Print contents of our variables

Don't want output being cluttered all the time

- Use a boolean variable to decide if we want debugging information

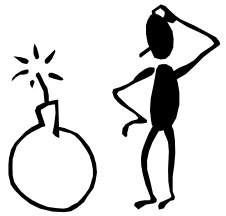
```
public void start() {
        boolean debug = true;

        if (debug) {  //********** DEBUGGING CODE **********
                System.out.println("----- Starting the main method ----");
        }

        int x = 3;
        int y = 7;
        int hyp = Math.sqrt(x * x + y * y);
        if (debug) {//********** DEBUGGING CODE **********
                System.out.println("hyp = " + hyp);
        }
        ... //code continues
}
```

Warning: Common Mistakes

Mathematics shortcuts do not work in Java.



```
if (10 < B < 20) {
   doSomething();
}
```

Not valid in Java

```
if (10 < B && B < 20) {
   doSomething();
}
```

Use this version instead

Remember that the equality test uses double equals ==

```
if (a = b) {
   doSomething();
}
```

Tries to do assignment