

Boolean Expressions & the 'if' Statement

September 24, 2007

ComS 207: Programming I (in Java)
Iowa State University, FALL 2007
Instructor: Alexander Stoytchev

© 2004 Pearson Addison-Wesley. All rights reserved

Midterm Results

- Average: 91.6
- Median: 94
- Standard Deviation: 18.80
- Maximum: 129 (out of 130)

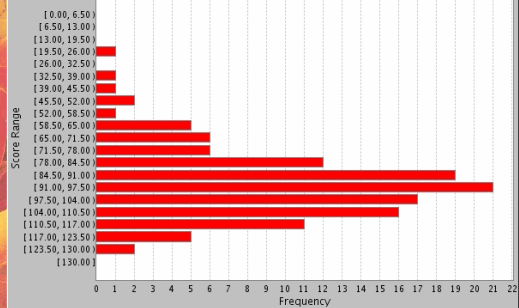
© 2004 Pearson Addison-Wesley. All rights reserved

Midterm Results (Fall 2006)

- Average: 86.1
- Median: 97
- Standard Deviation: 24.18
- Maximum: 127 (out of 130)

© 2004 Pearson Addison-Wesley. All rights reserved

Grade Histogram



© 2004 Pearson Addison-Wesley. All rights reserved

Top Scores

- Hrishank Jhildiyal 129
- Alaettin Mete 125
- Xu Feng 121
- Daniel Pratt 120
- Kian Chen 120
- Alex Cole 119
- Paul Wolf 117
- Michael Steffen 116
- Alexander Reifert 116
- Eun Kim 116
- Keith Johnson 116
- Austin Green 115

© 2004 Pearson Addison-Wesley. All rights reserved

HW4 is out

- Due this Friday
- It is shorter than normal (programs only)
- Electronic submission only

© 2004 Pearson Addison-Wesley. All rights reserved

Chapter 5
Sections 5.1 – 5.2

PEARSON Addison-Wesley
© 2004 Pearson Addison-Wesley. All rights reserved.

Flow of Control

- Unless specified otherwise, the order of statement execution through a method is linear: one statement after another in sequence
- Some programming statements allow us to:
 - decide whether or not to execute a particular statement
 - execute a statement over and over, repetitively
- These decisions are based on *boolean expressions* (or *conditions*) that evaluate to true or false
- The order of statement execution is called the *flow of control*

© 2004 Pearson Addison-Wesley. All rights reserved.

Method Control Flow

- If the called method is in the same class, only the method name is needed

© 2004 Pearson Addison-Wesley. All rights reserved.

Method Control Flow

- The called method is often part of another class or object

© 2004 Pearson Addison-Wesley. All rights reserved.

Encapsulation

- An encapsulated object can be thought of as a *black box* -- its inner workings are hidden from the client
- The client invokes the interface methods of the object, which manages the instance data

© 2004 Pearson Addison-Wesley. All rights reserved.

Conditional Statements

- A *conditional statement* lets us choose which statement will be executed next
- Therefore they are sometimes called *selection statements*
- Conditional statements give us the power to make basic decisions
- The Java conditional statements are the:
 - *if statement*
 - *if-else statement*
 - *switch statement*

© 2004 Pearson Addison-Wesley. All rights reserved.

The if Statement

- The *if* statement has the following syntax:

`if` is a Java reserved word

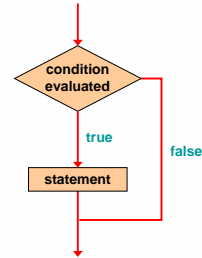
The *condition* must be a boolean expression. It must evaluate to either true or false.

```
if ( condition )
    statement;
```

If the *condition* is true, the *statement* is executed. If it is false, the *statement* is skipped.

© 2004 Pearson Addison-Wesley. All rights reserved

Logic of an if statement



© 2004 Pearson Addison-Wesley. All rights reserved

Boolean Expressions

- A condition often uses one of Java's *equality operators* or *relational operators*, which all return boolean results:

<code>==</code>	equal to
<code>!=</code>	not equal to
<code><</code>	less than
<code>></code>	greater than
<code><=</code>	less than or equal to
<code>>=</code>	greater than or equal to

- Note the difference between the equality operator (`==`) and the assignment operator (`=`)

© 2004 Pearson Addison-Wesley. All rights reserved

The if Statement

- An example of an *if* statement:

```
if (sum > MAX)
    delta = sum - MAX;
System.out.println ("The sum is " + sum);
```

- First the condition is evaluated -- the value of `sum` is either greater than the value of `MAX`, or it is not
- If the condition is true, the assignment statement is executed -- if it isn't, it is skipped.
- Either way, the call to `println` is executed next

© 2004 Pearson Addison-Wesley. All rights reserved

Example: [Age.java](#) (page 208)

© 2004 Pearson Addison-Wesley. All rights reserved

Indentation

- The statement controlled by the *if* statement is indented to indicate that relationship
- The use of a consistent indentation style makes a program easier to read and understand
- Although it makes no difference to the compiler, proper indentation is crucial

"Always code as if the person who ends up maintaining your code will be a violent psychopath who knows where you live."

-- Martin Golding

© 2004 Pearson Addison-Wesley. All rights reserved

The if Statement

- What do the following statements do?

```
if (top >= MAXIMUM)
    top = 0;
```

Sets `top` to zero if the current value of `top` is greater than or equal to the value of `MAXIMUM`

```
if (total != stock + warehouse)
    inventoryError = true;
```

Sets a flag to true if the value of `total` is not equal to the sum of `stock` and `warehouse`

- The precedence of the arithmetic operators is higher than the precedence of the equality and relational operators

© 2004 Pearson Addison-Wesley. All rights reserved

Logical Operators

- Boolean expressions can also use the following *logical operators*:

```
!      Logical NOT
&&    Logical AND
||    Logical OR
```

- They all take boolean operands and produce boolean results
- Logical NOT is a unary operator (it operates on one operand)
- Logical AND and logical OR are binary operators (each operates on two operands)

© 2004 Pearson Addison-Wesley. All rights reserved

Logical NOT

- The *logical NOT* operation is also called *logical negation* or *logical complement*
- If some boolean condition `a` is true, then `!a` is false; if `a` is false, then `!a` is true
- Logical expressions can be shown using a *truth table*

a	!a
true	false
false	true

© 2004 Pearson Addison-Wesley. All rights reserved

Logical AND and Logical OR

- The *logical AND* expression

```
a && b
```

is true if both `a` and `b` are true, and false otherwise

- The *logical OR* expression

```
a || b
```

is true if `a` or `b` or both are true, and false otherwise

© 2004 Pearson Addison-Wesley. All rights reserved

Logical Operators

- Expressions that use logical operators can form complex conditions

```
if (total < MAX+5 && !found)
    System.out.println ("Processing...");
```

- All logical operators have lower precedence than the relational operators
- Logical NOT has higher precedence than logical AND and logical OR

© 2004 Pearson Addison-Wesley. All rights reserved

Logical Operators

- A truth table shows all possible true-false combinations of the terms
- Since `&&` and `||` each have two operands, there are four possible combinations of conditions `a` and `b`

a	b	a && b	a b
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

© 2004 Pearson Addison-Wesley. All rights reserved

Boolean Expressions

- Specific expressions can be evaluated using truth tables

total < MAX	found	!found	total < MAX && !found
false	false	true	false
false	true	false	false
true	false	true	true
true	true	false	false

© 2004 Pearson Addison-Wesley. All rights reserved

Short-Circuited Operators

- The processing of logical AND and logical OR is "short-circuited"
- If the left operand is sufficient to determine the result, the right operand is not evaluated

```
if (count != 0 && total/count > MAX)
    System.out.println ("Testing...");
```

- This type of processing must be used carefully

© 2004 Pearson Addison-Wesley. All rights reserved

The if-else Statement

- An *else clause* can be added to an *if* statement to make an *if-else statement*

```
if ( condition )
    statement1;
else
    statement2;
```

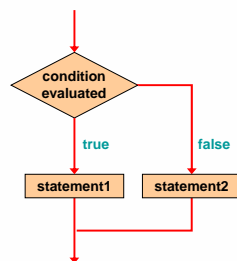
- If the *condition* is true, *statement1* is executed; if the condition is false, *statement2* is executed
- One or the other will be executed, but not both

© 2004 Pearson Addison-Wesley. All rights reserved

Example: [Wages.java](#) (page 211)

© 2004 Pearson Addison-Wesley. All rights reserved

Logic of an if-else statement



© 2004 Pearson Addison-Wesley. All rights reserved

THE END

© 2004 Pearson Addison-Wesley. All rights reserved