Introduction to Computers and Programming
Lecture 5
Boolean type; if statement
Chris Conway
New York University
Road Map

- Review Div / Mod
- Reverse digits of a number
- boolean type
- Algorithms
- Representing programs
  - Pseudocode
  - Flow charts
- Control structures
  - overview
  - introduction to if statement

Reading
- Liang 5: Chapter 2: 2.10; Chapter 3: 3.1 – 3.2..2
- Liang 6 & 7: Chapter 3: 3.1 – 3.3
review

- Why do we use constants in our programs instead of literal values?
- Is the following constant declaration good style?
  ```java
  final int total  = 5;
  ```
- Name 4 different numeric data types?
- Is there a difference between the way 4.0 (a double) and 4 (an int) are represented by the computer?
- What does it mean to cast a value?
- When must you explicitly cast a value?
- If you mix numeric data types on the right hand side of a gets operator, what happens?
• What happens if you go beyond a type’s upper bound?
  – For example:
  ```java
  int i = 2147483647;
  i = i + 1;
  ```
• What happens if you try to place a literal value in an integer type that is too big for the variable?
  – For example: `byte b = 483648;`
• What if you go beyond an integer type’s bounds using arithmetic?
• Which has higher precedence:
  – + (addition)
  – % (modulus)
Do you see a pattern?

1234 / 1000 = ?
1234 % 1000 = ?
234 / 100 = ?
234 % 100 = ?
34 / 10 = ?
34 % 10 = ?
What is the pattern?

1234 / 1000 = 1
(lost three right digits and ended up with the left (first) digit)

1234 % 1000 = 234
(lost the left most digit and ended up with the remaining 3 digits)

234 / 100 = 2
(lost the 2 right digits and ended up with the left (first) digit)

234 % 100 = 34
(lost the left most digit and ended up with the remaining 2 digits)

34 / 10 = 3
(lost right digit and ended up with the left (first) digit)

34 % 10 = 4
(lost the left most digit and ended up with the remaining right digit)
Boolean values

• Java provides a type just for true and false evaluation.

• Named after George Boole, the English mathematician who published “An investigation into the Laws of Thought” in 1854 which began Boolean logic.

• Any Boolean expression will evaluate to either true or false.
## Relational Operators

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

©2003 Prentice Hall, Inc. All rights reserved.
Example

```java
import javax.swing.JOptionPane;

public class BoolTest
{
    public static void main(String[] args)
    {
        boolean boolVar;

        boolVar = false;
        System.out.println("boolVar: "+boolVar);

        int a = 10;
        int b = 10;
        boolVar = (a == b);
        System.out.println("boolVar: "+boolVar);

        System.out.println(a == b);
        System.out.println(a != b);
        System.out.println(a < b);
        System.out.println(a <= b);
        System.out.println(a > b);
        System.out.println(a >= b);
    }
}
```
Equality v. Assignment

• Remember Gets not Equals!
  ( grade = 100 )
  
  Will not evaluate to true or false

• In this case, we are using a single = character.
  (We really want to use ==)
Introduction to Problem Solving with Computers

• Before writing a program:
  – Have a thorough understanding of the problem
  – Carefully plan an approach for solving it

• While writing a program:
  – Know what “building blocks” are available
  – Use good programming principles
Algorithms

• Computing problems
  – All can be solved by executing a series of actions in a specific order

• Algorithm: procedure in terms of
  – Actions to be executed
  – The order in which these actions are to be executed

• Program control
  – Specify order in which statements are to executed

• Examples of problems:
  – Determining the class average for a final exam
  – Sorting a list of names in alphabetical order
Pseudocode

• Pseudocode
  – Artificial, informal language that helps us develop algorithms
  – Similar to everyday English
  – Not actually executed on computers
  – Helps us “think out” a program before writing it
    • Easy to convert into a corresponding Java program
    • Consists only of executable statements
      – For example, declarations and import statements are not used in pseudocode.
What is a “Flow Chart?”

• A flow chart is a visual tool that helps you understand the flow of your program.
  – Graphical representation of program structure

• Different shapes have different meaning.
Flow Chart Basics 1

Rectangles represent statements of work. For example:
System.out.println();
Control Structures
Control Structures

• Control the flow of a program
• Normally, in Java, statements are executed in sequential order
• Control structures allow the programmer to specify that a statement other than the next be executed
  - i.e. programmer can control what’s executed in which order
Three Basic Control Structures

- All programs can be written with just these types of structures
  - **Sequence structure**
    - Statements run one after the other
  - **Selection structure**
    - Depending on a condition, do one thing; otherwise, do something else
    - Examples in Java: if, if else, and switch.
  - **Repetition structure**
    - Repeat some actions over and over
    - Examples in Java: for loops, while loops, and do/while loops.
The if structure

• Pseudocode:
  
  if some Boolean expression is true
   do this

• Example:
  
  if ( x == y ) {
    System.out.println(" x is equal to y!");
  }

• Every procedural / OO programming language has some form of an if statement.
Another if example

if ( temperature >= 85 ){
    System.out.println( "It is hot out!" );
}


if Flow Chart

- temperature >= 85
  - true: print "It is hot"
  - false: return to previous step

The flow chart illustrates a conditional statement where if the temperature is greater than or equal to 85, it prints "It is hot". If the condition is false, it goes back to the previous step.
if with a twist: if else

• Pseudocode:
if some Boolean expression is true
do this
otherwise
do something else

• Example:
if ( grade >= 65 )
    System.out.println( "You passed!" );
else
    System.out.println( "You failed!" );
if/else Flow Chart

grade >= 60

- True: Print "You passed"
- False: Print "You failed"
Blocks

• To run several lines of code together, you must include them within a block using curly braces

• For example:

```java
if ( grade >= 60 ) {
    System.out.println ( "You passed!" );
    System.out.println ( "Congratulations!" );
}
```
Indentation

• Everything within the block of code (even if it is an implicit block because we only use one statement) should be indented
  – helps you see the block at a quick glance.

• Avoid writing code like this:
  if (grade >= 65) {
    System.out.println("You passed!!!\n");
    System.out.println ("Congratulations!\n");
  }

• This is valid Java code, but it is not easy to view the block: *bad style*
Common error: misplaced semi-colon

• Remember, Java requires that you use a semicolon to terminate a statement.

• A complete if statement is formed as follows:

```java
if (boolean expression)
Statement or block of code;
```
Common error: misplaced semi-colon

• But a statement can be *empty*!
• If you place a semicolon after the conditional as in

```java
if (boolean expression);
    Statement or block of code;
```

The compiler will interpret the semicolon as a null statement. In other words, nothing will happen if the expression evaluates to true and the statement of block of code will be executed whether or not the boolean expression is true.
import javax.swing.JOptionPane;

public class PassFail
{
    public static void main(String[] args)
    {
        int grade;
        String gradeAsString;

        gradeAsString = JOptionPane.showInputDialog(null,"What is your grade?");
        grade = Integer.parseInt (gradeAsString);

        /* find out if grade is passing */
        if ( grade >= 65 )
        {
            System.out.println ( "You passed!!!" );
            System.out.println ( "Congratulations!" );
        }
        else
        {
            System.out.println ("You failed!");
        }
        System.exit (0);
    }
}