Lecture 7: Chapter 3 Review

• CS 170, Section 000, Fall 2009
• 17 September 2009
Lecture Plan

• Logistics (important announcements)
• Chapter 2 “quiz”
• HW1 Questions?
• Chapter 3 review
2.13: Scanner class

- System.out.print("Enter a double value: ");
- Scanner input = new Scanner(System.in);
- double d = input.nextDouble();
Chapter 2 Quiz

• What is the result of \( 17 \mod 4 \)?
  \[ 1 \]

• What is the result of \( 21.5 \mod 4 \)?
  \[ 1 \ ? \ 1.5 \]

  – What is the value of \( n \) and \( m \) after each statement?

  - \( \text{int } n = 1; \)
  - \( \text{int } m = ++n; \)  \( n = 2 \)
  - \( n += 3; \)  \( n = 5 \)
  - \( n += m++; \)  \( m = 1 \)
  - \( n += m + 7!; \)  \( n = 7 \)  \( m = 2 \)
Chapter 3 Selections

CS170
Introduction to Computer Science
## Objectives

- Declare and use boolean types
- Compare values using relational operators
- Write Boolean expressions using logic operators
- Implement selection control using if statements
- Implement selection control using switch statements
The Boolean Expressions

• Use a boolean variable
  
  ```java
  boolean var; // declare a boolean variable
  ```

• Assign a boolean value
  
  ```java
  var = expression; // assign a boolean value
  ```

• Declare and assign
  
  ```java
  boolean var = expression; // declare and assign
  ```

• A Boolean expression is an expression that evaluates to a Boolean value

• Comparison operators: compare a pair of values (numbers, characters, boolean values)
  
  ```java
  boolean validInput = radius > 0;
  ```

• Boolean operators: perform logic operations
# Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater 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>
# Boolean Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>exclusive or</td>
</tr>
</tbody>
</table>
Truth Table for Operator !

<table>
<thead>
<tr>
<th>p</th>
<th>!p</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

Example

!(1 > 2) is true, because (1 > 2) is false.
!(1 > 0) is false, because (1 > 0) is true.
### Truth Table for Operator &&

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>p1 &amp;&amp; p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

**Example**

(3 > 2) && (5 >= 5) is true, because (3 > 2) and (5 >= 5) are both true.

(3 > 2) && (5 > 5) is false, because (5 > 5) is false.
## Truth Table for Operator $\lor$ $\lor$

<table>
<thead>
<tr>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_1 \lor p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

### Example

$(2 > 3) \lor (5 > 5)$ is false, because $(2 > 3)$ and $(5 > 5)$ are both false.

$(3 > 2) \lor (5 > 5)$ is true, because $(3 > 2)$ is true.
## Truth Table for Operator ^

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>p1 ^ p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

### Example

(2 > 3) ^ (5 > 1) is true, because (2 > 3) is false and (5 > 1) is true.

(3 > 2) ^ (5 > 1) is false, because both (3 > 2) and (5 > 1) are true.
Example – TestBoolean

```java
Scanner input = new Scanner(System.in);
int number = input.nextInt();

System.out.println("Is " + number + " divisible by 2 and 3? " +
                    (number % 2 == 0) && (number % 3 == 0));
System.out.println("Is " + number + " divisible by 2 or 3? " +
                    (number % 2 != 0) || (number % 3 != 0));
System.out.println("Is " + number + " divisible by 2 or 3, but not both? " +
                    || ^ || );
```

/home/cs170000/inclass/sept17/TestBoolean.java
• Java uses shortcut evaluation - evaluation of boolean expression stops as soon as the result is known, which makes code execute faster than if all boolean operands were evaluated.

```java
int x = 0;
if (x > 0 && x++ < 5) x = x - 1;
```
Problem: Determining Leap Year?

• Write a program that determines if given year is a leap year

• Read input year

• Compute whether input year is a leap year - a year is a leap year if it is divisible by 4 but not by 100 or if it is divisible by 400

• Display the result

/home/cs170000/inclass/sept17/LeapYear.java
Exercise: A Simple Math Learning Tool

• Write a program to let a first grader practice additions
• The program should randomly generate two single-digit integers number1 and number2 and display a question such as “What is 7 + 9?” to the student
• After the student types the answer, the program displays a message to indicate whether the answer is true or false.

/home/cs170000/inclass/sept17/AdditionQuiz.java
Selection Statements

• if statement
• switch statement
• Conditional operators
Simple if Statements

- if (booleanExpression) {
  - statement(s);
- }

- if (radius >= 0) {
  - area = radius * radius * PI;
  - System.out.println("The area
    + " for the circle of radius " + radius + " is " + area);
- }

(A)  

(B)  

Diagram:

- Boolean Expression
  - false
  - true
  - Statement(s)

- (radius >= 0)
  - false
  - true
  - Compute area and print results
Flow Chart

• A flowchart is a schematic representation of an algorithm or a process.
• Start and end symbols, represented as rounded rectangles
  – usually containing the word "Start" or "End“
• Arrows, showing "flow of control".
• Processing steps, represented as rectangles.
• Input/Output, represented as a parallelogram.
• Conditional (or decision), represented as a diamond.
Example

- Testing whether a number is even or odd
  
  ```java
  if (number % 2 == 0) {
      System.out.println(number + " is even");
  }
  if (number % 2 != 0) {
      System.out.println(number + " is odd");
  }
  ```
if (i > 0) && (i < 10) {
    int a = 10;
    a = a / 3;
    System.out.println("i is between 0 and 10");
}

if ((i > 0) && (i < 10)) {
    int a = 10;
    a = a / 3;
    System.out.println("i is between 0 and 10");
}
Note – Block statement

```java
if ((i > 0) && (i < 10)) {
    System.out.println("i is between 0 and 10");
}
```

```java
if ((i > 0) && (i < 10))
    System.out.println("i is between 0 and 10");
```
Caution – Block statement

```java
if ((i > 0) && (i < 10)) {
    int a = 10;
    a = a / 3;
    System.out.println("i is between 0 and 10");
}
```

```java
if ((i > 0) && (i < 10)) {
    int a = 10;
    a = a / 3;
    System.out.println("i is between 0 and 10");
}
```
Caution – Extra semicolon

• Adding a semicolon at the end of an if clause is a mistake.

Wrong

• if (radius >= 0);
• {

  area = radius*radius*PI;
  }

  System.out.println("The area for the circle of radius "+ radius + " is " + area);

• }
The if...else Statement

- if (booleanExpression) {
  - statement(s)-for-the-true-case;
- }
- else {
  - statement(s)-for-the-false-case;
- }

![Diagram showing the flow of the if...else statement]
if...else Example

```java
• if (radius >= 0) {
  • area = radius * radius * 3.14159;
  • System.out.println("The area for the circle " +
    •     "of radius " + radius + " is " + area);
  • }
• else {
  • System.out.println("Negative input");
  • }
```
Multiple Alternative if Statements

```java
if (score >= 90.0)
    grade = 'A';
else
    if (score >= 80.0)
        grade = 'B';
    else
        if (score >= 70.0)
            grade = 'C';
        else
            if (score >= 60.0)
                grade = 'D';
            else
                grade = 'F';
else
    grade = 'F';
```

Equivalent

```java
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
```
Trace if-else statement

Suppose score is 70.0

```java
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
```

The condition is false
Trace if-else statement

Suppose score is 70.0

```java
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
```

The condition is false
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
Trace if-else statement

Suppose score is 70.0

```java
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
```

grade is C
Trace if-else statement

Suppose score is 70.0

```java
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
```

Exit the if statement
The else clause matches the most recent if clause in the same block.

What does it print?

(a) int i = 1;
   int j = 2;
   int k = 3;
   if (i > j) {
      if (i > k) {
         System.out.println("A");
      } else {
         System.out.println("B");
      }
   } else {
      System.out.println("B");
   }

(b) int i = 1;
    int j = 2;
    int k = 3;
    if (i > j) {
      if (i > k) {
         System.out.println("A");
      } else {
         System.out.println("B");
      }
    } else {
      System.out.println("B");
    }
To force the else clause to match the first if clause, you must add a pair of braces.

```java
int i = 1;
int j = 2;
int k = 3;
if (i > j) {
    if (i > k)
        System.out.println("A");
} else 
    
```
TIP – Assigning value to a boolean variable

boolean even;
if (number % 2 == 0)
    even = true;
else
    even = false;

boolean even = number % 2 == 0;
TIP – testing a boolean variable

(a) if (even == true)
   System.out.println("It is even.");

Equivalent

(b) if (even)
   System.out.println("It is even.");
Conditional Operator

```java
if (booleanExp)
    var = exp1;
else
    var = exp2;

if (x > 0)
    y = 1
else
    y = -1;
```

```java
var = (booleanExp) ? exp1 : exp2
y = (x > 0) ? 1 : -1;
```
Conditional Operator

```java
if (num % 2 == 0)
    System.out.println(num + "is even");
else
    System.out.println(num + "is odd");

System.out.println((num % 2 == 0)? (num + " is even") : (num + " is odd"));

if (amount/100 <= available100 )
    n100 = amount/100;
else
    n100 = available100;

n100 = (amount/100 <= available100) ? amount/100 : available100
```
switch Statement

```java
switch (switch-expression) {
    case value1: statement(s)1;
        break;
    case value2: statement(s)2;
        break;
    ...
    case valueN: statement(s)N;
        break;
    default: statement(s)-for-default;
}
```
switch Statement

• switch (dayOfWeek) {
  • case 0: System.out.println("Sunday");
  •   break;
  • case 1: System.out.println("Monday");
  •   break;
  • case 2: System.out.println("Tuesday");
  •   break;
  • case 3: System.out.println("Wednesday");
  •   break;
}
switch Statement Flow Chart

- Day is 0
  - Day is Sunday
  - break
- Day is 1
  - Day is Monday
  - break
- Day is 2
  - Day is Tuesday
  - break
- Day is 3
  - Day is Wednesday
  - break
- default
  - Default actions

Next Statement
Caution: switch statement

Don’t forget the break statement!

Once a case is matched, the statements are executed until a break statement or the end of switch statement is reached

Suppose ch is 'a':

```java
switch (ch) {
    case 'a': System.out.println(ch);
    case 'b': System.out.println(ch);
    case 'c': System.out.println(ch);
}
```
Problem: display floating point number with two digits after decimal point

Previous solution:
```java
double x = 12.2223345322;
System.out.println("Number is " + (int) (x * 100) / 100.0);
```

Good solution: use printf and specify a format
```java
double x = 12.2223345322;
System.out.printf("Number is %.2f", x);
```

```
12.22
```

Formatting Output
Formatting Output

**Syntax:**

```java
System.out.printf(format, item1, item2, ... , itemk);
```

**Usage:**

Format consists of substrings and format specifiers which specifies how the items should be displayed respectively.

**Examples:**

```java
int n = 100;
int i = 0;
... Change value of i ...
System.out.printf("You completed \%d out of \%d tasks", i, n);

String name = "Steve";
double balance;
... Change value of balance ...
System.out.printf("Hello \%s, your balance is \%2f", name, balance);
```
Frequently-Used Format Specifiers

A format specifier consists of a percent sign and a conversion code

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Output</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%b</td>
<td>a boolean value</td>
<td>true or false</td>
</tr>
<tr>
<td>%c</td>
<td>a character</td>
<td>'a'</td>
</tr>
<tr>
<td>%d</td>
<td>a decimal integer</td>
<td>200</td>
</tr>
<tr>
<td>%f</td>
<td>a floating-point number</td>
<td>45.460000</td>
</tr>
<tr>
<td>%e</td>
<td>a number in standard scientific notation</td>
<td>4.556000e+01</td>
</tr>
<tr>
<td>%s</td>
<td>a string</td>
<td>&quot;Java is cool&quot;</td>
</tr>
</tbody>
</table>
Operator Precedence (operators order)

- **What is the value of the expression:**
  \[3 + 4 * 4 > 5 * (4 + 3) - 1\]

- The expression in parenthesis is evaluated first
  \[(2 \% 13 + 4) > 3 + 2\]
  
  Evaluate subexpressions in parenthesis

- When expression does not have parenthesis, certain rules are applied during evaluation
Operator Precedence

- `var++`, `var--`
- `+`, `-` (Unary plus and minus), `++var`, `--var`
- `(type)` Casting
- `!` (Not)
- `*`, `/`, `%` (Multiplication, division, and remainder)
- `+`, `-` (Binary addition and subtraction)
- `<`, `<=`, `>`, `>=` (Comparison)
- `==`, `!=`; (Equality)
- `^` (Exclusive OR)
- `&&` (Conditional AND) Short-circuit AND
- `||` (Conditional OR) Short-circuit OR
- `=`, `+=`, `-=`, `*=`, `/=`, `%=` (Assignment operator)
Operator Associativity

- When two operators with the same precedence are evaluated, the associativity of the operators determines the order of evaluation.

- All binary operators except assignment operators are left-associative.
  
  \[(a - b) + c - d\]  \[a - b + c - d\]

- Assignment operators are right-associative.
  
  \[a = (b += (c = 5))\]  \[a = b += c = 5\]
Example

\[
3 + 4 \times 4 > 5 \times (4 + 3) - 1
\]

(1) inside parentheses first

\[
3 + 4 \times 4 > 5 \times 7 - 1
\]

(2) multiplication

\[
3 + 16 > 5 \times 7 - 1
\]

(3) multiplication

\[
3 + 16 > 35 - 1
\]

(4) addition

\[
19 > 35 - 1
\]

(5) subtraction

\[
19 > 34
\]

(6) greater than

false
HINT

• If not sure what gets evaluated first, ALWAYS use parenthesis to make the expression to be evaluated as you want

```java
double value = i++ + x * 4 % n;
```

VS

```java
double value = (i ++) + (x * 4) % n;
```
Problem: Lottery

• Write a program that implements a lottery
• Generate a random number between 0-99
• Prompt the user to enter a guess between 0-99
• Check the guess and give out prize using the following rule:
  – If user’s number has the same digits as the generated number, the prize is $3,000
  – If user’s number has one digit that appears in the generated number, the prize is $1,000
  – Otherwise there is no prize