Introduction to Computers and Programming

Lecture 7

Chris Conway
New York University
Midterm

• Our midterm will be held in class on Thursday, June 4th.

• I will tell you the exact material covered in the coming classes.

• I will also talk about the format of the exam.

• There will be no make up exam so your attendance is mandatory.
Road map

• `switch` statement
• `char` data type
• Assignment operators

• Reading
  – Chapter 2: 2.7.4; 2.9; Chapter 4.1, 4.2 (part)
Review

• True or False: An int variable can contain any integer.

• What is a magic number? How should you deal with a magic number? Why?

• Explain what the exclusive or (^) operator tests.
  \[(\text{exp } a) \wedge (\text{exp } b)\]

• Define the purpose of each part of this expression:
  \[(\text{part } a) \? (\text{part } b) : (\text{part } c)\]
• What is the output of the following code fragment?

```java
int a = 100, b = 50;
if ((a == 60) && (b <= 100))
    System.out.println("Yes");
```

• What is the output of the following code fragment?

```java
int a = 100, b = 50;
if ((a == 60) || (b <= 100))
    System.out.println("Yes");
```
switch Multiple-Selection Structure

• Used when testing a variable or expression for EQUALITY (i.e., no >, <, >=, <= tests) separately for each of the constant integral values it may assume.

• Preferred over if else in situations where you are testing the same expressions for equality with many different values.

• Allows you to perform different actions for each value.
switch (expression) {
  case value1:
    action(s);
    break;
  case value2:
    action(s);
    break;
  ...
  default:
    actions(s);
    break;
}

- The keyword `switch` marks the start of a switch statement.
- The expression inside the parentheses is evaluated.
- The `case` keyword is followed by an expression or value.
- Each case block contains actions that will be executed if the corresponding expression matches.
- The `break` keyword is used to exit the switch statement.
- The default case is executed if no other case matches.

- Expression can be a variable or a more complicated expression.
- Could use more than one case if the same actions are required.
- Actions within a single case do not need brackets.
- The default case will be executed in the event that no other case is matched.
The `switch` Multiple-Selection Structure

- Flowchart of the `switch` structure

```
true
• case a action(s) break
false
• case b action(s) break
false
• case z action(s) break
false
default action(s)
```

©2000 Prentice Hall, Inc.
All rights reserved.
beware of “fall through”

• If you forget to use the break keyword between cases, unexpected things may happen.

• Once a case tests true, all the statements following that case, will be executed until the next break.

• Experienced programmers may use this on purpose. For this class we will rarely use fall though.

• If you do you fall through, add a comment pointing it out

• (But don't use fall through.)
char data type
Java allows us to store "single" character values.

* `char character = 'a'; // not the same as "a";`
* `char character = '7';`
* `char newline = '\n';`
* `char tab = '\t';`
* `char space = ' ';`

The characters are actually stored as integers (ascii values).

See [http://asciitable.com/](http://asciitable.com/)

Note: `char` use single quotes. We have seen that Strings use double quotes.
### ASCII Table

**TABLE B.1** ASCII Character Set in the Decimal Index

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>nul</td>
<td>soh</td>
<td>stx</td>
<td>etx</td>
<td>eot</td>
<td>enq</td>
<td>ack</td>
<td>bel</td>
<td>bs</td>
</tr>
<tr>
<td>1</td>
<td>nl</td>
<td>vt</td>
<td>ff</td>
<td>cr</td>
<td>so</td>
<td>si</td>
<td>dle</td>
<td>dcl</td>
<td>dc2</td>
</tr>
<tr>
<td>2</td>
<td>dc4</td>
<td>nak</td>
<td>syn</td>
<td>etb</td>
<td>can</td>
<td>em</td>
<td>sub</td>
<td>esc</td>
<td>fs</td>
</tr>
<tr>
<td>3</td>
<td>rs</td>
<td>us</td>
<td>sp</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
</tr>
<tr>
<td>4</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>-</td>
<td>.</td>
<td>/</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:)</td>
</tr>
<tr>
<td>6</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
<td>`</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>10</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
</tr>
<tr>
<td>11</td>
<td>n</td>
<td>o</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
</tr>
<tr>
<td>12</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td></td>
<td>}</td>
<td>-</td>
<td>del</td>
</tr>
</tbody>
</table>

Source: Liang
• Given:
  ```java
  char letter = 'a';
  ```
• The following code:
  ```java
  letter = (char) (letter + 1);
  ```
• Would result in letter storing a 'b' character.
• If we add or subtract two char values, the result is an int value.
  – For example 'c' – 'a' would result in the value 2.
Reading **char** values from the user

- You should use `charAt(0)` to parse the character from user input.
- For example:
  ```java
  char c;
  String cAsString;
  cAsString = JOptionPane.showInputDialog(null, "Enter a character");
  c = cAsString.charAt(0);
  ```
- Will grab the first character from the user's input.
Casting between char and int values

• A char uses 16 bits of memory.
  – You can implicitly cast a char to an int
    ```
    char c = 'a';
    int i = c;
    ```
  – You must explicitly cast an int to a char
    ```
    int i = 65;
    char = (char) i;
    ```
• Note: Even though chars are equal to shorts in the amount of memory they use, they do not hold the same values (shorts can hold negative integers).
Warning about numeric digit characters

- The value of a single digit numeric characters are not equivalent to the values themselves. In fact the ASCII value of '0' is 48, '1' is 49, ..., '9' is 57.
- How do you think we could convert a numeric char to an int?
Unicode

• In Java, there are many more characters available than in the basic ASCII table.
• The ASCII table only has 128 characters in it.
• Java uses 2 bytes to store characters which allows it to hold 65536 unique characters.
• Java can store any Unicode (see: unicode.org) character in a char variable.
• That means you can print any character from any language on any platform.
• To print a Unicode character, use '\uxxxx' where xxxx is a hexadecimal representation of the Unicode for the desired character.
Assignment Operators

• Given the following:

```java
x = 2;
x = x + 1;
System.out.println("x: " + x);
```

• There are actually several ways to rewrite this more concisely.
Short Cut Operator

• One option is to use the += operator
  
x = 2;
x += 1;  // same as x = x + 1;
System.out.println ("x: " + x);

• There are similar operators for other operators:
  
  - x = x * 5;  is equivalent to x *= 5;
  - x = x - 5;  is equivalent to x -= 5;
  - x = x / 5;  is equivalent to x /= 5;
  - x = x % 5;  is equivalent to x %= 5;

• Good Practice: place a space before and after your short cut operators.
Increment Operator

• A second option is to use an increment operator:
  \[ x++ \]  Post-Increment Operator
  \[ ++x \]  Pre-Increment Operator

• Both operators will increment \( x \) by 1, but they do have subtle differences.
Pre v. Post Increment

• PostIncrement Operator (\(x++\)):
  – use the current value of \(x\) in the expression. Then, increment by 1.

• PreIncrement Operator (\(++x\)):
  – Increment \(x\) by 1. Then, use the new value of \(x\) in the expression.
How about a real example?

// Preincrementing v. PostIncrementing
public class PrePost {

    public static void main (String[] args) {
        int c = 5;
        System.out.println (c);
        System.out.println (c++);
        System.out.println (c);
        System.out.println();
        c = 5;
        System.out.println (c);
        System.out.println (++c);
        System.out.println (c);
    }
}

Output:
5
5
6
5
6

Pre v. Post Decrement

• PostDecrement Operator ($x--$):
  – use the current value of $x$ in the expression. Then, decrease by 1.

• PreDecrement Operator ($--x$):
  – Decrease $x$ by 1. Then, use the new value of $x$ in the expression.

• Good practice: Place unary operators directly next to their operands, with no intervening spaces.