Java Chapter 3 Part 2

* Decisions and Scope, switch, and Number Formatting
* CIS 255 • Shelby-Hoover Campus

Declaration and Scope

* Many programmers prefer to declare all variables and constants at the top of the method
  + Easy to find identifiers, types
  + Scope: remainder of method
* A variable that is not used until a specific set of statements may be declared just before those statements
* Variables may be declared within any block
  + Inside an if, else if, or else compound statement
  + Inside a loop body (chapter 4)
  + Scope: from the declaration statement to the end of the containing block (closing curly bracket)

The Conditional Operator

* The conditional operator is a ternary operator (three operands) that is a shorthand version of the if-else statement
* Form #1: Standalone Statement  
  (Boolean\_expression) ? stmt\_if\_true : stmt\_if\_false;  
  (month == 11) ? discount = 0.20 : discount = 0;
* Form #2: Assignment Statement  
  variable = (Boolean\_expression) ? value\_if\_true : value\_if\_false;  
  discount = (month == 11) ? 0.20 : 0;

The switch Statement

* An alternative to using a long if-else-if structure for multiple values of the same variable is the switch statement
* Each value to which the variable or expression is to be compared is written in a case within the brackets
* Basic Form:  
    
  switch(testExpression)  
  {  
   case value1:  
   statements\_for\_value1;  
   break;  
   case value2: // as many cases as needed  
   statements\_for\_value2;  
   break;  
   default:  
   statements\_for\_no\_match;  
  }

Notes about switch Syntax

* The expression in the parentheses is not a complete Boolean expression; it should be a variable or arithmetic expression to be compared to each of the case values and must be of the type char, byte, short, or int (Java 7 also supports test expressions of type String)
* Once Java finds a match between the expression and a case value, it executes the statements below it until it hits a break statement or the closing curly bracket
  + Multiple case statements can be stacked to execute the same code for different values
  + If you forget a break statement, Java will continue executing any remaining statements until it encounters a break or the closing curly bracket of the switch
* To execute statements when there is no match (similar to the trailing else), place a default case at the end

switch Examples

* Translating gender abbreviations:  
    
  String input;  
  char gender;  
    
  input = JOptionPane.showInputDialog("Enter gender (M/F):");  
  gender = input.charAt(0);  
    
  switch(gender) {  
   case 'M':  
   case 'm':  
   JOptionPane.showMessageDialog(null, "You are male!");  
   break;  
   case 'F':  
   case 'f':  
   JOptionPane.showMessageDialog(null, "You are female!");  
   break;  
   default:  
   JOptionPane.showMessageDialog(null, "Consult a physician.");  
  }
* Determining the cost of pet food by grade: PetFood.java (Code Listing 3-15)
* Translating seasons from English to Spanish: Seasons.java (Code Listing 3-16)
* The impact of missing break statements: NoBreaks.java (Code Listing 3-14)

Console Formats using printf

* The console object System.out supports a third output method, printf, that applies formatting to values displayed in the console
* Instead of separating the string literals from the variables using the + operator, printf uses format specifiers within a single string literal as placeholders for the value to be displayed, with the values to be inserted in the placeholders listed after the string literal
* Each format specifier begins with a percent symbol and ends with a conversion character representing the type of value to be formatted
  + %f – floating-point (float, double)
  + %d – decimal integer (byte, short, int, long)
  + %s – string (String)
* Basic Example:  
    
  System.out.printf("Hello %s. Your age is %d and your income is $%f\n",   
   name, age, income);

Format Specifier Options

* Between % and the conversion character, various formatting options can be added
* The general format (items are marked with square brackets to show that they are optional):   
    
  %[flags][width][.precision]conversion  
  + The width is a minimum field width, or the minimum number of characters to be set aside for this value (usually spaces are added to the left to right-justify the value within the field width)
  + A period followed by a precision value specifies the number of decimal places to which a floating-point number should be rounded
  + The flags include such formatting as comma separators (,), padding with leading zeros to fill the field width (0), and left-justifying the value within the field width (-)

printf Examples

* Aligning floating-point values in columns with a consistent precision: Columns.java   
  (Code Listing 3-17)
* Adding commas to a currency amount: CurrencyFormat.java (Code Listing 3-18)
* Padding to fill the field width with leading zeros instead of spaces: LeadingZeros.java   
  (Code Listing 3-19)
* Left-justifying numbers in two columns: LeftJustified.java (Code Listing 3-20)
* Note that printf does not start a new line automatically: use the escape sequence \n to indicate where to start a new line
* The Java APIs (and other programmer reference sites) may describe other options for format specifiers in printf

The DecimalFormat Class

* The method printf cannot be used with dialog boxes
* Objects of the Java class DecimalFormat apply formatting to decimal values in any output
* Requires: import java.text.DecimalFormat;
* Declare an object similar to the Scanner type:  
    
  DecimalFormat objectName = new DecimalFormat("options");
* The String literal in quotation marks lets Java know how to format numbers using that object

The Formatting String

* Characters in the String literal of a new DecimalFormat object indicate how many digits are required, how many are optional, and what punctuation is added
  + # - optional digit
  + 0 – required digit
  + . – decimal point
  + , - comma separator (only 1 needed)
  + % - multiply value by 100, add % sign to end
* If a number to be formatted has more digits to the right of the decimal point than allowed, the displayed value is rounded (but digits to the left of the decimal point are not constrained)
* Examples:
  + "#0.00" – 10s digits optional, 1s digit required, 2 digits after decimal required
  + "#,##0.00" – 1s digit required, 2 digits after decimal required, comma added between groups of 3 digits

Using a DecimalFormat Object

* Apply the method format to the object name, placing the value you want to format inside the parentheses:  
    
  formatObject.format(myDouble)
* Place this call inside an output statement (console or dialog) to display the formatted result:  
    
  double agi;  
  DecimalFormat money = new DecimalFormat("#,##0.00");  
  // statements to calculate AGI here  
  System.out.println("Your AGI is $" + money.format(agi));
* Examples
  + Two digits after decimal point: Format1.java (Code Listing 3-21)
  + Two digits before decimal point: Format2.java (Code Listing 3-22)
  + Commas where needed: Format3.java (Code Listing 3-23)
  + Percent format (multiplied by 100): Format4.java (Code Listing 3-24)

Reminders for Chapter 3

* = and == are not the same operation
* The Boolean expression for if and else if headers must be placed in parentheses
* The if, else if, else, and switch headers do not end in a semicolon
* To execute more than one statement for a particular Boolean result in an if-else statement, group the statements in curly brackets
* The trailing else can be useful in a chain of if-else-if statements to execute particular statements when none of the Boolean expressions are true
* When using && and ||, make sure each simple Boolean expression is complete
* && has precedence over || unless parentheses are used

More Reminders

* String objects must be compared using methods
* The String methods for comparison can be case sensitive or case insensitive
* The conditional operator’s elements are separated by ? and :
* The expression that controls a switch (in parentheses) must be an int, short, byte, or char (Java 7 also supports String)
* case values must be either literals or named constants (final)
* Each case statement ends in a colon
* To prevent execution to the end of the switch structure, use a break statement at the end of the statements for each case
* To execute particular statements when none of the case values matches the switch expression, add a default case
* With printf, format specifiers for each value are written inside the string literal; the values to be formatted are listed after the literal
* The DecimalFormat object must have a formatting String value in parentheses when it is created