**Computer Science Notes**

## Chapter 3: Selections

These notes are meant to accompany Introduction to Java Programming: Brief Version, seventh edition by Y. Daniel Lang.

**Programming Skills in a Nutshell:**

At the end of this chapter you should have the following programming skills:

1. Translate logical conditions into Boolean expressions.
2. Use simple **if … else** statements to control program flow.
3. Use nested if statements for multiple outcomes and conditions.
4. Use a switch statement for multiple outcomes and conditions that are based on a positive integer.
5. Here is a template that uses the key programming skills you should have at this point:

|  |
| --- |
| **import** java.util.Scanner;  //The import statement tells the linker which additional classes that will be used.  //In this case, the Scanner class will be used.  //The Scanner class reads in and stores keystrokes from the keyboard.  **public** **class** Chap03Basics  {  /\*\* Prompts the user to enter a tax year, filing status, and income,  \* then computes the income tax due according to those values,  \* and prints the results.  \* **@param** args is not used.  \*/  **public** **static** **void** main(String[] args)  {  **final** **int** FIRST\_TAX\_YEAR = 2007;  **final** **int** LAST\_TAX\_YEAR = 2008;  **final** **int** SINGLE = 1; //filing status of single  **final** **int** JOINT = 2; //filing status of married filing jointly  **final** **int** SEPARATE = 3; //filing status of married filing separately  **final** **int** HEAD = 4; //filing status of Head of Household  **final** String SINGLE\_TEXT = "Single";  **final** String JOINT\_TEXT ="Married Filing Jointly";  **final** String SEPARATE\_TEXT ="Married Filing Separately";  **final** String HEAD\_TEXT = "Head of Household";  //Tell the user what the program does  String programPurpose = "This program performs U.S. federal tax " +  "calculations for any year between " +  FIRST\_TAX\_YEAR + " and " + LAST\_TAX\_YEAR;  System.*out*.println(programPurpose);  //Create a Scanner object for reading in the user's input  Scanner keyboard = **new** Scanner(System.*in*);  //Read in the tax year, and make sure it is in the specified range  String prompt = "\nPlease enter the tax year (" + FIRST\_TAX\_YEAR +  " through " + LAST\_TAX\_YEAR + "):";  System.*out*.println(prompt);  **int** taxYear = keyboard.nextInt();    **if** (taxYear < FIRST\_TAX\_YEAR || taxYear > LAST\_TAX\_YEAR)  {  System.*out*.println("Invalid tax year. Program is now terminating.");  System.*exit*(0);  }  //Read in the income, and make sure it is positive.  prompt = "\nPlease enter your income:";  System.*out*.println(prompt);  **double** income = keyboard.nextDouble();  **if** (income < 0)  {  System.*out*.println("Invalid (negative) income. " +  "Program is now terminating.");  System.*exit*(0);  }  //Read in the filing status, and make sure it is valid.  System.*out*.println("\nUse one of the menu choices below to enter " +  "your filing status.");  System.*out*.println(SINGLE + ") " + SINGLE\_TEXT);  System.*out*.println(JOINT + ") " + JOINT\_TEXT);  System.*out*.println(SEPARATE + ") " + SEPARATE\_TEXT);  System.*out*.println(HEAD + ") " + HEAD\_TEXT);  prompt = "\nPlease enter your filing status: ";  System.*out*.println(prompt);  **int** filingStatus = keyboard.nextInt();  **if** (!(filingStatus == SINGLE || filingStatus == JOINT ||  filingStatus == SEPARATE || filingStatus == HEAD))  {  System.*out*.println("Invalid filing status. " +  "Program is now terminating.");  System.*exit*(0);  }  //Declare variables to store the tax rates for each tax bracket.  //Initialize them with the most common values used from 2003 - 2008.  //They can be set to new values, if needed, when the income cutoffs get set.  //See http://www.moneychimp.com/features/tax\_brackets.htm for all the info.  **double** taxRate1 = 0.10;  **double** taxRate2 = 0.15;  **double** taxRate3 = 0.25;  **double** taxRate4 = 0.28;  **double** taxRate5 = 0.33;  **double** taxRate6 = 0.35;  //Declare variables to store the upper income cutoffs for each tax bracket.  //Initialize them to zero in case something is wrong  //with the taxYear value or the filingStatus value.  **double** cutoff1 = 0.00;  **double** cutoff2 = 0.00;  **double** cutoff3 = 0.00;  **double** cutoff4 = 0.00;  **double** cutoff5 = 0.00;  //Set the income cutoffs for each tax year and filing status  **switch** (taxYear)  {  **case** 2007:  //Set the income cutoffs for each bracket and filing status.  **if** (filingStatus == SINGLE)  {  cutoff1 = 7825.00;  cutoff2 = 31850.00;  cutoff3 = 77100.00;  cutoff4 = 160850.00;  cutoff5 = 349700.00;  }  **else** **if** (filingStatus == JOINT)  {  cutoff1 = 15650.00;  cutoff2 = 63700.00;  cutoff3 = 128500.00;  cutoff4 = 195850.00;  cutoff5 = 349700.00;  }  **else** **if** (filingStatus == SEPARATE)  {  cutoff1 = 7825.00;  cutoff2 = 31850.00;  cutoff3 = 64250.00;  cutoff4 = 97925.00;  cutoff5 = 174850.00;  }  **else** **if** (filingStatus == HEAD)  {  cutoff1 = 15650.00;  cutoff2 = 63700.00;  cutoff3 = 128500.00;  cutoff4 = 195850.00;  cutoff5 = 349700.00;  }  **break**;  **case** 2008:  //Set the income cutoffs for each bracket and filing status.  **if** (filingStatus == SINGLE)  {  cutoff1 = 8025.00;  cutoff2 = 32550.00;  cutoff3 = 78850.00;  cutoff4 = 164550.00;  cutoff5 = 357700.00;  }  **else** **if** (filingStatus == JOINT)  {  cutoff1 = 16050.00;  cutoff2 = 65100.00;  cutoff3 = 131450.00;  cutoff4 = 200300.00;  cutoff5 = 357700.00;  }  **else** **if** (filingStatus == SEPARATE)  {  cutoff1 = 8025.00;  cutoff2 = 32550.00;  cutoff3 = 65725.00;  cutoff4 = 100150.00;  cutoff5 = 178850.00;  }  **else** **if** (filingStatus == HEAD)  {  cutoff1 = 11450.00;  cutoff2 = 43650.00;  cutoff3 = 112650.00;  cutoff4 = 182400.00;  cutoff5 = 357700.00;  }  **break**;  **default**:  System.*out*.println("Invalid tax year. Program is now terminating.");  System.*exit*(0);  }//end of switch statement for setting the income cutoffs based on taxYear  //Compute the tax due.  **double** tax = 0.0;  **int** taxBracket = 0;  **double** taxBracketRate = 0.0;  **if** (income <= cutoff1)  {  tax = income \* taxRate1;  taxBracket = 1;  taxBracketRate = taxRate1;  }  **else** **if** (income <= cutoff2)  {  tax = cutoff1 \* taxRate1  + (income - cutoff1) \* taxRate2;  taxBracket = 2;  taxBracketRate = taxRate2;  }  **else** **if** (income <= cutoff3)  {  tax = cutoff1 \* taxRate1  + (cutoff2 - cutoff1) \* taxRate2  + (income - cutoff2) \* taxRate3;  taxBracket = 3;  taxBracketRate = taxRate3;  }  **else** **if** (income <= cutoff4)  {  tax = cutoff1 \* taxRate1  + (cutoff2 - cutoff1) \* taxRate2  + (cutoff3 - cutoff2) \* taxRate3  + (income - cutoff3) \* taxRate4;  taxBracket = 4;  taxBracketRate = taxRate4;  }  **else** **if** (income <= cutoff5)  {  tax = cutoff1 \* taxRate1  + (cutoff2 - cutoff1) \* taxRate2  + (cutoff3 - cutoff2) \* taxRate3  + (cutoff4 - cutoff3) \* taxRate4  + (income - cutoff4) \* taxRate5;  taxBracket = 5;  taxBracketRate = taxRate5;  }    **else**  {  tax = cutoff1 \* taxRate1  + (cutoff2 - cutoff1) \* taxRate2  + (cutoff3 - cutoff2) \* taxRate3  + (cutoff4 - cutoff3) \* taxRate4  + (cutoff5 - cutoff4) \* taxRate5  + (income - cutoff5) \* taxRate6;  taxBracket = 6;  taxBracketRate = taxRate6;  }    System.*out*.println("\nHere are your results:");  System.*out*.println("filing year = " + taxYear);    String filingStatusEcho = "filing status = ";  **if** (filingStatus == SINGLE)  filingStatusEcho += SINGLE\_TEXT;  **else** **if** (filingStatus == JOINT)  filingStatusEcho += JOINT\_TEXT;  **else** **if** (filingStatus == SEPARATE)  filingStatusEcho += SEPARATE\_TEXT;  **else** **if** (filingStatus == HEAD)  filingStatusEcho += HEAD\_TEXT;  System.*out*.println(filingStatusEcho);    System.*out*.println("income = $"  + String.*format*("%,12.2f", income) );  System.*out*.println("income tax owed = $" + String.*format*("%,12.2f", tax));  System.*out*.println("tax bracket = " + taxBracket);  System.*out*.println("tax bracket rate = "  + String.*format*("%4.1f", taxBracketRate\*100) + "%");    System.*out*.println("\nProgram is now terminating.");  }//end method main(String[])  }//end of class Chap03Basics |

|  |
| --- |
| This program performs U.S. federal tax calculations for any year between 2007 and 2008  Please enter the tax year (2007 through 2008):  2007  Please enter your income:  50000  Use one of the menu choices below to enter your filing status.  1) Single  2) Married Filing Jointly  3) Married Filing Separately  4) Head of Household  Please enter your filing status:  1  Here are your results:  filing year = 2007  filing status = Single  income = $ 50,000.00  income tax owed = $ 8,923.75  tax bracket = 3  tax bracket rate = 25.0%  Program is now terminating. |

|  |
| --- |
| This program performs U.S. federal tax calculations for any year between 2007 and 2008  Please enter the tax year (2007 through 2008):  2010  Invalid tax year. Program is now terminating. |

**Book’s Statement of Skills:**

1. To declare **boolean** type and use Boolean values true and false. (3.2)
2. To apply relational operators **(<, <=, ==, !=, >, >=)**and logic operators **(!, &&, ||, ^)** to write Boolean expressions. (3.2)
3. To use Boolean expressions to control selection statements. (3.3 – 3.5)
4. To implement selection control using **if** and nested **if** statements. (3.3)
5. To implement selection control using **switch** statements. (3.4)
6. To write expressions using the conditional operator. (3.5)
7. To display formatted output using the **System.out.printf** method and to format strings using the **String.format** method. (3.6)
8. To examine the rules governing operator precedence associativity. (3.7)
9. (GUI) To get user confirmation using confirmation dialogs. (3.8)

**Section 3.1: Introduction**

This chapter will teach you how to use **if** and **switch** statements, which are used to determine which lines of code a program executes based on certain conditions like user input or the values stored in variables.

**Example:**

**Problem to solve:** If the user enters a negative value for a radius of a circle, then display an error message; otherwise compute and display the circumference.

↓ **Pseudocode**..

Get the user’s input for a circle radius

If (the radius is negative)

Then print an error message

Otherwise

Compute and print the circumference

↓ **Real code**

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** CircleCircumference  {  **public** **static** **void** main(String[] args)  {  System.*out*.println("This program will compute the circumference of " +  "a circle given its radius.");    Scanner keyboard = **new** Scanner(System.*in*);    // Get the user’s input for a circle radius  System.*out*.println("Enter the radius of a circle:");  **double** radius = keyboard.nextDouble();  **if** (radius < 0.0)  System.*out*.println("Bad entry; you entered r = " + radius +  ", and the radius of a circle should NOT be negative.");  **else**  System.*out*.println("For a circle of radius " + radius +  ", the circumference is: " + (2\*Math.*PI*\*radius) + ".");  }  } |

|  |
| --- |
| This program will compute the circumference of a circle given its radius.  Enter the radius of a circle:  -15  Bad entry; you entered r = -15.0, and the radius of a circle should NOT be negative. |

|  |
| --- |
| This program will compute the circumference of a circle given its radius.  Enter the radius of a circle:  12  For a circle of radius 12.0, the circumference is: 75.39822368615503. |

**Section 3.2: boolean Data Type and Operations**

* The **boolean** data type contains one of two values: **true** or **false**.
* What can be true or false??? 🡺 the answer to a comparison of numbers…
* The comparison operators are: **<, <=, >, >=, = =, !=**
* Comparison operators are used to compare numeric values.
* The result of a comparison calculation is a Boolean value of **true** or **false**.
* **true** and **false** Are Boolean literals and reserved keywords
* Boolean variables hold Boolean values.
* The Boolean operators are: **!, &&. ||, and ^**
* Boolean operators relate boolean expressions to determine the overall truth or falsehood of the statement.
* Where does the adjective “Boolean” come from? It is in honor of British mathematician George Boole, who laid the groundwork for modern logic theory in his book [*An Investigation of the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities*](http://en.wikipedia.org/wiki/The_Laws_of_Thought) (1854).

**Comparison operators: used to compare the value of two numerical expressions.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| **<** | Less than | **1 < 2** | **true** |
| **<=** | Less than or equal to | **1 <= 2** | **true** |
| **>** | Greater than | **1 > 2** | **false** |
| **>=** | Greater than or equal to | **1 >= 1** | **true** |
| **==** | Equal to | **1 = = 2** | **false** |
| **!=** | Not equal to | **1 != 2** | **true** |

**Boolean operators: used to determine the truth of a combination of expressions.**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Description** |
| ! | Not | Logical negation |
| && | And | Logical conjunction |
| || | Or | Logical disjunction |
| ^ | Exclusive or | Logical exclusion |

**Practice:**

Fill in the following truth tables:

|  |  |  |  |
| --- | --- | --- | --- |
| **Truth Table for && (AND)** | | | |
| Example | Value of **expr1** | Value of **expr2** | Value of **exprJoint** |
| boolean expr1 = 3 > 2;  boolean expr2 = 1 < 5;  boolean exprJoint = expr1 && expr2; | true | True | True |
| boolean expr1 = 3 > 2;  boolean expr2 = 1 > 5;  boolean exprJoint = expr1 && expr2; | True | False | False |
| boolean expr1 = 3 < 2;  boolean expr2 = 1 < 5;  boolean exprJoint = expr1 && expr2; | False | True | False |
| boolean expr1 = 3 < 2;  boolean expr2 = 1 > 5;  boolean exprJoint = expr1 && expr2; | False | False | False |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Truth Table for || (OR)** | | | |
| Example | Value of **expr1** | Value of **expr2** | Value of **exprJoint** |
| boolean expr1 = 3 > 2;  boolean expr2 = 1 < 5;  boolean exprJoint = expr1 || expr2; | True | True | True |
| boolean expr1 = 3 > 2;  boolean expr2 = 1 > 5;  boolean exprJoint = expr1 ||expr2; | True | False | True |
| boolean expr1 = 3 < 2;  boolean expr2 = 1 < 5;  boolean exprJoint = expr1 ||expr2; | False | True | True |
| boolean expr1 = 3 < 2;  boolean expr2 = 1 > 5;  boolean exprJoint = expr1 ||expr2; | False | False | False |

|  |  |  |  |
| --- | --- | --- | --- |
| **Truth Table for ^ (EXCLUSIVE OR)** | | | |
| Example | Value of **expr1** | Value of **expr2** | Value of **exprJoint** |
| boolean expr1 = 3 > 2;  boolean expr2 = 1 < 5;  boolean exprJoint = expr1 ^ expr2; | True | True | False |
| boolean expr1 = 3 > 2;  boolean expr2 = 1 > 5;  boolean exprJoint = expr1 ^expr2; | True | False | True |
| boolean expr1 = 3 < 2;  boolean expr2 = 1 < 5;  boolean exprJoint = expr1 ^expr2; | False | True | True |
| boolean expr1 = 3 < 2;  boolean expr2 = 1 > 5;  boolean exprJoint = expr1 ^expr2; | False | False | False |

|  |  |  |  |
| --- | --- | --- | --- |
| **Truth Table for ! (NOT)** | | | |
| Example | Value of **expr1** | Value of **expr2** | Value of **exprJoint** |
| boolean expr1 = 3 > 2;  boolean exprJoint = !expr1; |  |  |  |
| boolean expr1 = 3 < 2;  boolean exprJoint = !expr1; |  |  |  |

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** BooleanExample  {  **public** **static** **void** main(String[] args)  {  Scanner keyboard = **new** Scanner(System.*in*);    System.*out*.println("This program will determine where you were born " +  "based on your answers to two questions.");  **boolean** lovePackers = **true**;  **boolean** loveWisconsin = **true**;  System.*out*.println("Type true if you love the Packers, ");  System.*out*.println("or type false if you dislike the Packers.");  lovePackers = keyboard.nextBoolean();  System.*out*.println("Type true if you love Wisconsin, ");  System.*out*.println("or type false if you dislike Wisconsin.");  loveWisconsin = keyboard.nextBoolean();  **if** (loveWisconsin == **true** && lovePackers == **true**)  System.*out*.println("You must be a native Wisconsinite.");  **if** (loveWisconsin == **true** && lovePackers == **false**)  System.*out*.println("You must be a Wisconsin immigrant.");  **if** (loveWisconsin == **false** && lovePackers == **true**)  System.*out*.println("You must be from Minnesota.");  **if** (loveWisconsin == **false** && lovePackers == **false**)  System.*out*.println("You must be from Illinois.");  //Exclusive or:  //i.e., you love Wisconsin or you love the Packers, but not both.  **if** (loveWisconsin ^ lovePackers)  System.*out*.println("That's good enough for me.");  //Longhand exclusive or:  **if** ((loveWisconsin || lovePackers) && !(loveWisconsin && lovePackers))  System.*out*.println("That's good enough for me.");  }  } |
| This program will determine where you were born based on your answers to two questions.  Type true if you love the Packers,  or type false if you dislike the Packers.  true  Type true if you love Wisconsin,  or type false if you dislike Wisconsin.  true  You must be a native Wisconsinite. |

|  |
| --- |
| This program will determine where you were born based on your answers to two questions.  Type true if you love the Packers,  or type false if you dislike the Packers.  true  Type true if you love Wisconsin,  or type false if you dislike Wisconsin.  false  You must be from Minnesota.  That's good enough for me.  That's good enough for me. |

|  |
| --- |
| This program will determine where you were born based on your answers to two questions.  Type true if you love the Packers,  or type false if you dislike the Packers.  false  Type true if you love Wisconsin,  or type false if you dislike Wisconsin.  true  You must be a Wisconsin immigrant.  That's good enough for me.  That's good enough for me. |

|  |
| --- |
| This program will determine where you were born based on your answers to two questions.  Type true if you love the Packers,  or type false if you dislike the Packers.  false  Type true if you love Wisconsin,  or type false if you dislike Wisconsin.  false  You must be from Illinois. |

**More Boolean notes:**

* To test if a variable is within a range of numbers:

|  |  |
| --- | --- |
| WRONG WAY TO TEST A NUMERICAL RANGE | CORRECT WAY TO TEST A NUMERICAL RANGE |
| **int** numberOfDaysInAMonth = 32;  **if** (1 <= numberOfDaysInAMonth <= 31)  ... | **int** numberOfDaysInAMonth = 32;  **if** (1 <= numberOfDaysInAMonth &&  numberOfDaysInAMonth <= 31)  ... |

* A Boolean data type can not be cast into other data types.
* DeMorgan’s Law:

!(condition1 && condition2) **is the same thing as** !condition1 || !condition2

!(condition1 || condition2) **is the same thing as** !condition1 && !condition2

**Example:**

!(1 <= numberOfDaysInAMonth && numberOfDaysInAMonth <= 31)

**is the same thing as**

!(1 <= numberOfDaysInAMonth) || !(numberOfDaysInAMonth <= 31)

* && is a conditional, or short-circuit operator because if the first expression evaluates to false, then the overall expression is false, and so the second expression’s value is not computed because there is no need to.
* || is also a short-circuit operator because if the first expression evaluates to true, then the overall expression is true, and so second expression’s value is not computed because there is no need to.
* & is an unconditional operator because both expressions’ values are computed before the overall value of the expression is determined.

**Section 3.3: Problem: A Simple Math Learning Tool**

* See [www.cs.armstrong.edu/liang/intro8e/book/AdditionQuiz.java](http://www.cs.armstrong.edu/liang/intro8e/book/AdditionQuiz.java)

**Section 3.4: if Statements**

An **if** statement is used to force a program to execute certain lines of code based on the truth of a given condition.

**Section 3.4.1: One-Way if Statements**

The **if** statement by itself forces a program to execute a statement only if a given condition is true.

**Syntax of a simple** if **statement:**

if (*condition*)

{

*conditionTrueStatement(s)*;

}

*nextStatement;*

* When the condition is false, the program executes *nextStatement* right away, thus skipping *conditionTrueStatement(s)*,
* but when the condition is true, the program executes *conditionTrueStatement(s)* first, and then *nextStatement*.

**Example:**

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** CircleCircumference2  {  **public** **static** **void** main(String[] args)  {  System.*out*.println("This program will compute the circumference of " +  "a circle given its radius.");    Scanner keyboard = **new** Scanner(System.*in*);    // Get the user’s input for a circle radius  System.*out*.println("Enter the radius of a circle:");  **double** radius = keyboard.nextDouble();  **if** (radius >= 0.0)  System.*out*.println("For a circle of radius " + radius +  ", the circumference is: " + (2\*Math.*PI*\*radius) + ".");  System.*out*.println("This program is done now.");  }  } |

**Section 3.5: Problem: Guessing Birthdays**

* See [www.cs.armstrong.edu/liang/intro8e/book/GuessingBirthdays.java](http://www.cs.armstrong.edu/liang/intro8e/book/GuessingBirthdays.java)

**Section 3.6: Two-Way if Statements**

The **if .. else** statement forces a program to execute one statement only if a given condition is true, and a different statement if the condition is false.

**Syntax of a simple** if **statement:**

if (*condition*)

{

*conditionTrueStatement(s)*;

}

else

{

*conditionFalseStatement(s)*;

}

*nextStatement;*

* When the condition is true, the program executes *conditionTrueStatement(s)* first, and then *nextStatement*.
* When the condition is false, the program executes *conditionFalseStatement* right away, and then *nextStatement*.

**Example:**

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** CircleCircumference  {  **public** **static** **void** main(String[] args)  {  System.*out*.println("This program will compute the circumference of " +  "a circle given its radius.");    Scanner keyboard = **new** Scanner(System.*in*);    // Get the user’s input for a circle radius  System.*out*.println("Enter the radius of a circle:");  **double** radius = keyboard.nextDouble();  **if** (radius < 0.0)  System.*out*.println("Bad entry; you entered r = " + radius +  ", and the radius of a circle should NOT be negative.");  **else**  System.*out*.println("For a circle of radius " + radius +  ", the circumference is: " + (2\*Math.*PI*\*radius) + ".");  System.*out*.println("This program is done now.");  }  } |

**Section 3.7: Nested if Statements**

Nested if statements are used when you want your program to execute one of many different statements based on one of several different conditions.

**Syntax of a nested** if **statement:** Use the syntax of an if … else statement, with the else statements containing additional if … else statements.

**Example:**

Compute the letter grade of a numerical score according to the following grading scale:

A 92 % - 100 %

AB 88 % - 91 %

B 82 % - 87 %

BC 78 % - 81 %

C 72 % - 77 %

D 65 % - 71 %

F 0 % - 64 %

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** NestedIfExample  {  **public** **static** **void** main(String[] args)  {  Scanner keyboard = **new** Scanner(System.*in*);    System.*out*.println("This program will convert a numerical score " +  "into a letter Grade");    System.*out*.println("Enter a numerical score.");  **double** score = keyboard.nextDouble();  String letterGrade;  **if** (score >= 91.5)  letterGrade = "A";  **else** **if** (score >= 87.5)  letterGrade = "AB";  **else** **if** (score >= 81.5)  letterGrade = "B";  **else** **if** (score >= 77.5)  letterGrade = "BC";  **else** **if** (score >= 71.5)  letterGrade = "C";  **else** **if** (score >= 64.5)  letterGrade = "D";  **else**  letterGrade = "F";  System.*out*.println("A score of " + score + " earns a letter grade of " +  letterGrade + ".");  }  } |

|  |
| --- |
| This program will convert a numerical score into a letter Grade  Enter a numerical score.  85.6  A score of 85.6 earns a letter grade of B. |

**Example:**

Write the Wisconsite/Packer messages using nested if statements instead of compound boolean expressions.

|  |  |
| --- | --- |
| **if** (loveWisconsin == **true** && lovePackers == **true**)  System.*out*.println("You must be a native Wisconsinite.");  **if** (loveWisconsin == **true** && lovePackers == **false**)  System.*out*.println("You must be a Wisconsin immigrant.");  **if** (loveWisconsin == **false** && lovePackers == **true**)  System.*out*.println("You must be from Minnesota.");  **if** (loveWisconsin == **false** && lovePackers == **false**)  System.*out*.println("You must be from Illinois.");  //Exclusive or:  //i.e., you love Wisconsin or you love the Packers, but not both.  **if** (loveWisconsin ^ lovePackers)  System.*out*.println("That's good enough for me."); | **if** (loveWisconsin == **true**)  **if** (lovePackers == **true**)  System.*out*.println("You must be a native Wisconsinite.");  **else**  System.*out*.println("You must be an immigrant.");  **else**  **if** (lovePackers == **true**)  System.*out*.println("You must be from Minnesota.");  **else**  System.*out*.println("You must be from Illinois.");  //Exclusive or:  **if** (loveWisconsin)  **if** (!lovePackers)  System.*out*.println("That's good enough for me.");  **else**;  **else**  **if** (lovePackers)  System.*out*.println("That's good enough for me."); |

**Section 3.8: Common Errors in Selection Statements**

* Forgetting necessary braces
* Semicolon at the end of the **if** line
* Redundant testing of boolean values
* Dangling **else** ambiguity – it doesn’t matter how you indent things, the else always goes with the most recent **if**.

**Section 3.9: Problem: An Improved Math Learning Tool**

* See [www.cs.armstrong.edu/liang/intro8e/book/SubtractionQuiz.java](http://www.cs.armstrong.edu/liang/intro8e/book/SubtractionQuiz.java)
* Notice the use of the **Math.random()** method.
* Note: the expression to generate a random integer between **min** (inclusive) and **max** (inclusive) is:

(**int**)((max - min + 1)\*Math.*random*()) + min

**Section 3.10: Problem: Computing Body Mass Index**

* See [www.cs.armstrong.edu/liang/intro8e/book/ComputeBMI.java](http://www.cs.armstrong.edu/liang/intro8e/book/ComputeBMI.java)

**Section 3.11: Problem: Computing Taxes**

* See [www.cs.armstrong.edu/liang/intro8e/book/ComputeTax.java](http://www.cs.armstrong.edu/liang/intro8e/book/ComputeTax.java)
* Notice the use of the System.exit(0) method.
* This is a good example of using incremental development and testing.

**Section 3.12: Logical Operators**

* + See notes in Section 3.2 …

**Section 3.13: Problem: Determining Leap Year**

* See [www.cs.armstrong.edu/liang/intro8e/book/LeapYear.java](http://www.cs.armstrong.edu/liang/intro8e/book/LeapYear.java)

**Section 3.14: Problem: Lottery**

* See [www.cs.armstrong.edu/liang/intro8e/book/Lottery.java](http://www.cs.armstrong.edu/liang/intro8e/book/Lottery.java)

**Section 3.15: switch Statements**

**switch** statements are used when you want to force the program to execute one of many different statements based on the integer values of a single integer expression.

**Syntax of a simple** switch **statement:**

**switch** (*integerExpression*)

{

**case** *value1*:

*statement(s)1*;

**break**;

**case** *value2*:

*statement(s)2*;

**break**;

…

**case** *valueN*:

*statement(s)N*;

**break**;

**default**:

*statement(s)Default*;

}

*nextStatement;*

* If the *integerExpression* evaluates to the number *value1*, the program executes *statement(s)1*, and then skips to *nextStatement* when the break statement is encountered. If the break statement were omitted by accident, the program would execute *statement(s)2* next.
* The same pattern follows for other values of *integerExpression*.
* If *integerExpression* doesn’t evaluate to any of the values, then the *statement(s)Default* get executed.
* Notice that parentheses for block statements are not needed after each case…

**Example:**

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** SwitchStatementExample  {  **public** **static** **void** main(String[] args)  {  Scanner keyboard = **new** Scanner(System.*in*);  System.*out*.println("This program will give you a one-problem arithmetic quiz.");  System.*out*.println("\nYour menu of problem types is:");  System.*out*.println("+: Addition problem.");  System.*out*.println("-: Subtraction problem.");  System.*out*.println("\*: Multiplication problem.");  System.*out*.println("/: Division problem.");  System.*out*.println("Enter the symbol for the type of problem you want.");  //Get the user's menu choice, then extract the first character.  String operationChoiceString = keyboard.next();  **char** operationChoice = operationChoiceString.charAt(0);  //Generate two random numbers between 0 and 9 for the quiz question.  //Use the fact that the Math.random() method  //generates random floating point numbers between 0 and 1  **int** number1 = (**int**) (Math.*random*() \* 10);  **int** number2 = (**int**) (Math.*random*() \* 10);    //Declare a variable to store the answer to the quiz question.  **int** correctAnswer = 0;    //The quiz will always be: number1 operationChoice number2.  //So, we need to adjust number1 and number2  //to always give nice answers for the chosen operationChoice.  **switch** (operationChoice)  {  **case** '+': //Compute the answer to the addition quiz question.  correctAnswer = number1 + number2;  **break**;    **case** '-': //Compute the answer to the subtraction quiz question.  //Swap numbers if necessary so number1 is greater than number 2  **if** (number2 > number1)  {  **int** temp = number1;  number1 = number2;  number2 = temp;  }  correctAnswer = number1 - number2;  **break**;    **case** '\*': //Compute the answer to the multiplication quiz question.  correctAnswer = number1 \* number2;  **break**;    **case** '/': //Compute the answer to the division quiz question.  //Make sure the divisor is not zero.  **if** (number2 == 0)  number2 = 3;  //Create a new dividend that's a multiple of the divisor.  number1 = (**int**) (Math.*random*() \* 10) \* number2;  correctAnswer = number1 / number2;  **break**;    **default**://Create the set-up for a simple addition quiz question,  //because the user isn't smart enough to enter a value for  //operationChoice correctly.  number1 = 1;  number2 = 1;  correctAnswer = number1 + number2;  operationChoice = '+';  }  //Print the quiz question.  System.*out*.println("What is " + number1 + " " + operationChoice +  " " + number2 + " ?");    //Read the user's answer.  **int** userAnswer = keyboard.nextInt();    //Tell the user is s/he is correct or not.  **if** (userAnswer == correctAnswer)  System.*out*.println("You are correct!");  **else**  System.*out*.println("Sorry, the correct answer is: " + correctAnswer);  }  } |

|  |
| --- |
| This program will give you a one-problem arithmetic quiz.  Your menu of problem types is:  +: Addition problem.  -: Subtraction problem.  \*: Multiplication problem.  /: Division problem.  Enter the symbol for the type of problem you want.  /  What is 9 / 9 ?  11  Sorry, the correct answer is: 1 |

|  |
| --- |
| This program will give you a one-problem arithmetic quiz.  Your menu of problem types is:  +: Addition problem.  -: Subtraction problem.  \*: Multiplication problem.  /: Division problem.  Enter the symbol for the type of problem you want.  B  What is 1 + 1 ?  2  You are correct! |

**Section 3.16: Conditional Expressions**

🡺 A shorthand way to assign one of two values to a variable based on a condition

🡺 **Syntax of a conditional expression:** *booleanExpression* ? *expression1* : *expression2*;

* If *booleanExpression* is true, then *expression1* is evaluated; otherwise, *expression2* is evaluated.

**Example: Computing the absolute value of a number.**

|  |
| --- |
| **if … else method for computing an absolute value:** |
| **import** java.util.Scanner;  **public** **class** AbsoluteValue  {  **public** **static** **void** main(String[] args)  {  Scanner keyboard = **new** Scanner(System.*in*);  System.*out*.println("This program will compute the absolute " +  "value of a given number.");  // Get the user’s input for the number  System.*out*.println("Enter a number:");  **double** x = keyboard.nextDouble();    **double** abs\_x;  **if** (x < 0.0)  abs\_x = -x;  **else**  abs\_x = x;    System.*out*.println("|" + x + "| = " + abs\_x);  }  } |

|  |
| --- |
| **Conditional expression method that does the same thing** |
| **import** java.util.Scanner;  **public** **class** AbsoluteValue  {  **public** **static** **void** main(String[] args)  {  Scanner keyboard = **new** Scanner(System.*in*);  System.*out*.println("This program will compute the absolute " +  "value of a given number.");  // Get the user’s input for the number  System.*out*.println("Enter a number:");  **double** x = keyboard.nextDouble();    **double** abs\_x;  abs\_x = (x < 0.0) ? -x: x;    System.*out*.println("|" + x + "| = " + abs\_x);  }  } |

**Section 3.17: Formatting Console Output**

The printf method of the PrintStream class (in the java.io package) allows you to display the values stored in variables in any of several different formats. The formats you can control include the number of columns used, the number of places after the decimal displayed, and how negative numbers are displayed. The way the printf method works is to replace the variable you want to display in a string with a format specifier (a code that starts with the % character and ends with a letter) that tells the JVM how to display the variable. The variable goes after the string that it is supposed to appear in. Format specifiers may include flags that give additional format information.

**Example:**

// the following line produces an output where the format specifier

// %6.2f will be replaced with the floating point number 1.5

// written using 6 spaces and 2 digits after the decimal point

// (and followed by a newline)

System.out.printf("Price is: $%6.2f\n", 1.517);

sample output:

Price is: $ 1.52

|  |  |  |
| --- | --- | --- |
| **Format Types** | | |
| **Code** | **Type** | **Example** |
| d | Decimal integer | System.out.printf("%5d", 123);  //output:  123 |
| x | Hexadecimal integer | System.out.printf("%5x", 123);  //output:  7B |
| o | Octal integer | System.out.printf("%5o", 123);  //output:  173 |
| f | Fixed floating point | System.out.printf("%6.2f", 123.);  //output:  123.00 |
| e | Exponential floating point | System.out.printf("%10.2e", 123.);  //output:  1.23e+1 |
| g | General floating point (printf decides whether to use f or e) | System.out.printf("%6.2g", 123);  //output:  123.00 |
| s | String | System.out.printf("%6s", "123");  //output:  123 |
| b | Boolean value | System.out.printf("%b", lovePackers);  //output:  true |
| c | Character value | System.out.printf("%c", menuChoice);  //output:  g |

|  |  |  |
| --- | --- | --- |
| **Format Flags** | | |
| **Flag** | **Meaning** | **Example** |
| - | Left justifiation | System.out.printf("%-5dhello", 123);  //output:  123 hello |
| 0 | Show leading zeros | System.out.printf("%06d", 123);  //output:  000123 |
| + | Show a + sign for positive numbers | System.out.printf("%+6d", 123);  //output:  +123 |
| ( | Enclose negative numbers in parentheses | System.out.printf("%(6d", -123);  //output:  (123) |
| , | Show decimal separators | System.out.printf("%,10d", 123000);  //output:  123,000 |

**Note:** to access this functionality use Java version 5.0 or higher.

The format method of the String class also recognizes format specifiers.

Example:

String outputString = String.format("%-5dhello", 123);

System.out.println(outputString);

//output:

23 hello

**Section 3.18: Operator Precedence and Associativity**

|  |  |
| --- | --- |
| **Operator Precedence Chart** | |
| **Precedence** | **Operator** |
| Highest Precedence | var++ and var-- (Postfix operators) |
|  | +, − (Unary operators)  ++var and --var (Prefix operators) |
|  | (*type*) (casting) |
|  | ! (Not) |
|  | \*, /, % |
|  | +, − (Binary addition and subtraction) |
|  | <, <=, >, >= (Comparison) |
|  | & (Unconditional and) |
|  | ^ (Exclusive or) |
|  | | (Unconditional or) |
|  | && (Conditional and) |
|  | || (Conditional or) |
| Lowest Precedence | =, +=, -=, \*=, /=, %= (Assignment operators) |

🡺 From left to right, according to the operator precedence

**Section 3.19: (GUI) Confirmation Dialogs**

**Review:**

* The showMessageDialog() method (of the javax.swing.JOptionPane class) displays a message in a dialog box.
* The showInputDialog() method (of the javax.swing.JOptionPane class) displays a dialog box that can be used to retrieve String input.

**New GUI method:**

* The showConfirmDialog() method (of the javax.swing.JOptionPane class) displays a dialog box with Yes, no, and Cancel buttons.
* This method returns a value of 0 (stored in the constant JOptionPane.YES\_OPTION), 1 (stored in the constant JOptionPane.NO\_OPTION), or 2 (stored in the constant JOptionPane.CANCEL\_OPTION).
* See [www.cs.armstrong.edu/liang/intro7e/book/GuessBirthDateUsingConfirmationDialog.java](http://www.cs.armstrong.edu/liang/intro7e/book/GuessBirthDateUsingConfirmationDialog.java)