CSCE 155 - Java

Lab 03 - Conditionals

Dr. Chris Bourke

Prior to Lab

Before attending this lab:

1. Read and familiarize yourself with this handout.
2. Review the lecture notes on conditionals, or the following free textbook resources:
   - Controlling Execution", pages 93-106 in *Thinking in Java* 4th Edition online textbook (available on Blackboard)
   - http://www.toves.org/books/java/ch07-if/index.html

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

1 Lab Objectives & Topics

At the end of this lab you should be familiar with the following

- How basic control flow works
- When and how to use if, if-then-else, and if-else-if conditional statements in a program

2 Background

The default control flow in a typical program is sequential. That is, each statement is executed one after the other. However, we often have to make decisions based on program state or variable values. This is known as conditional control flow. Java, like many programming languages, provides several control structures to alter the flow of control in a program based on the truth-value of some conditional statement.

An if-statement can be used to conditionally execute a block of code. If the condition in an if-statement evaluates to true the code block is executed, otherwise it is not.

```
1 if(x > 0) {
2   System.out.println("x is positive!\n");
3   System.out.println("the value of sqrt(x) = " + Math.sqrt(x) + "\n");
4 }
```

An if-then-else statement can be used to execute one of two mutually exclusive code blocks. If the condition evaluates to true, then the first block is executed. Otherwise, if the condition is false, the second code block is executed.

```
1 if(x > 0) {
2   System.out.println("x is positive!\n");
3   System.out.println("the value of sqrt(x) = " + Math.sqrt(x) + "\n");
4 } else {
5   System.out.println("x is not positive, cannot compute its square root\n");
6 }
```

An if-else-if statement can be used to execute one of several mutually exclusive statements. It also requires more than one conditional statement to determine which code block should be executed. The control flow of an if-else-if statement is such that the first

condition that evaluates to true is executed. All subsequent statements are skipped. If no condition holds true, then the else code block is executed. However, much like the difference between an if-statement and an if-then-else statement, the final else block is optional.

```
if(x > 0) {
  System.out.println("x is positive!\n");
  System.out.println("the value of sqrt(x) = " + Math.sqrt(x) + "\n");
} else if(x == 0) {
  System.out.println("x is zero, its square root is zero\n");
} else {
  System.out.println("x is not positive, its square root is complex:\n");
  System.out.println(Math.sqrt(x*-1.0));
}
```

Yet another type of conditional control statement is a switch statement. In a switch statement, a single variable's value is used to determine which, among several provided cases gets executed.

```
switch(class_level) {
  case 1:
    System.out.println("Freshman");
    break;
  case 2:
    System.out.println("Sophomore");
    break;
  case 3:
    System.out.println("Junior");
    break;
  case 4:
    System.out.println("Senior");
    break;
  default:
    System.out.println("ERROR: Unknown value!");
    break;
}
```

In Java, the switch-statement will work with the `byte`, `short`, `char`, and `int` primitive data types. It can also work with the `String` class and enumerated types. The most common form of error using switch statements is forgetting to include the break statement after each case. The break statement tells Java when to exit the switch selection structure.

In this lab you will write several programs that utilize control statements.

Operator	Java Syntax	Example
Negation	`!`	`!(x > 0)`
Logical And	`&&`	`(x != 0) && (y < 10)`
Logical	`||`	`(x < 0) || (x >= 10)`

Table 1: Logical Operators in Java

2.1 Compound Statements

Many logical statements require the use of logical operators that can be used to form more complex, compound statements. The three logical operators that we'll focus on are as follows.

- Negation: this is a unary operator that flips the truth value of the statement or variable that it is applied to so that true becomes false and vice versa.
- Logical And: this is a binary operator that is applied to two logical expressions and evaluates to true if and only if both expressions are true
- Logical Or: this is a binary operator that is applied to two logical expressions and evaluates to true if at least one (or both) of the expressions are true

Logical operators are necessary to do more complex statements such as checking for ranges of variable values. For example, to check if a variable `x` lies in the range $[0, 10]$, one would need to use a logical and operator:

```
if( x >= 0 && x <= 10 ) {
  //code
}
```

As another example, consider deciding whether or not the current year is a leap year. The Gregorian calendar defines a year as a leap year if it is divisible by 4. However, every year that is divisible by 100 is not a leap year unless it is divisible by 400. This logic can be modeled with the following expression.

```
if( year % 400 == 0 || ( year % 4 == 0 && year % 100 != 0 ) ) {
  printf("leap year!\n");
}
```

3 Activities

We have provided partially completed programs for each of the following activities. You will need to clone the Lab 03 project from Github using the URL: https://github.com/cbourke/CSCE155-Java-Lab03. Refer to previous labs for a step-by-step process.

If the AGI is over-	But not over-	The tax is:	Of the amount over-
$0	$17,000	10%	$0
$17,000	$69,000	$1,700 + 15%	$17,000
$69,000	$139,350	$9,500 + 25%	$69,000
$139,350	$212,300	$27,087.50 + 28%	$139,350
$212,300	$379,150	$47,513.50 + 33%	$212,300
$379,150	–	$102,574.00 + 35%	$379,150

Table 2: 2012 Tax Brackets

AGI	Number of Kids	Taxes
$4,000	1	$0
$20,000	0	$2,150
$120,000	3	$19,250
$150,000	4	$26,069.50
$250,000	0	$59954.50
$500,000	5	$139,871.50

Table 3: Several example inputs/outputs.

3.1 Tax Program

The federal income tax for a married couple filing jointly for 2012 is determined by the rules indicated in Table 2. In addition, the total tax is reduced by $1000 for each child that a couple has.

We have provided a partially completed program that reads in a user's Adjusted Gross Income (AGI) and prompts whether or not they have any children. You need to complete the program by writing code to do the following:

- If the user indicates that they do have children, prompt them to enter how many and compute the total tax credit
- Compute the user's tax based on the user's AGI and the total tax after the child tax credit

Some example input/output results can be found in Table 3.

Answer the questions on your worksheet.

3.2 Calculator

In this activity, you will implement a simple menu-based command line calculator. A partially completed program has been provided to you, `Calculator.java`. The program prompts the user for two operands and one of several different choices for an

operation to be performed on them. Your program should process the input and display the result of the chosen operation. Take care with the following possibilities:

- For division, you should check if $b = 0$ (division by zero is not defined). If it is, an appropriate error message should be output instead.
- For the logarithm operation, you should use the math library's `Math.log` function, which is the natural logarithm (base e). To change to another base, use the formula:

$$\log_a(b) = \frac{\ln(b)}{\ln(a)}$$

In addition, you should check that both operands are positive, if not then output an appropriate error message.

Complete the program and answer the questions on your worksheet.

3.3 Triangles

A triangle can be characterized in terms of the length of its three sides. In particular, an equilateral triangle is a triangle with all three sides being equal. A triangle such that two sides have the same length is isosceles and a triangle with all three sides having a different length is scalene. Examples of each can be found in Figure 1.

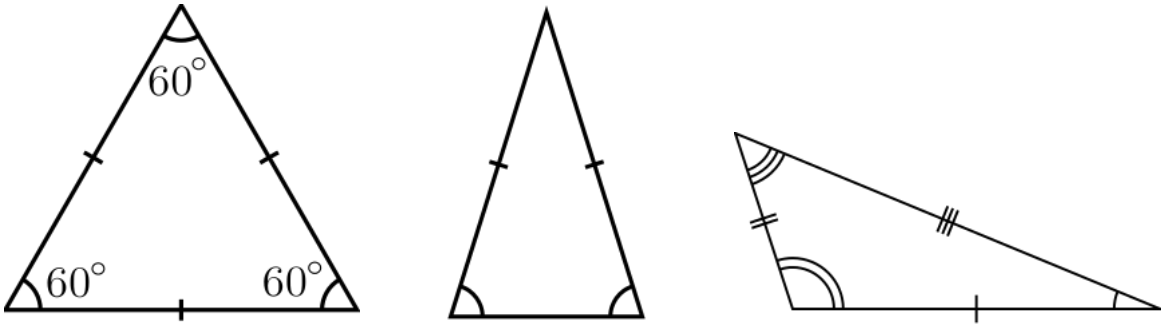

Figure 1: Examples of Equilateral, Isosceles, and Scalene triangles

In addition, the three sides of a triangle are valid only if the sum of the length of any two sides is strictly greater than the length of the third side. In this exercise you will complete a program that determines if 3 values form a valid triangle. If valid, the program should output whether or not the triangle is equilateral, isosceles or scalene. The program has been started for you (see `Triangles.java`), which reads in the three sides of a triangle via command line arguments. Complete the program by implementing the logic to determine if the sides form a valid triangle and if so, what type.

After you have completed the program and thoroughly tested it, hand it in using web-handin and grade yourself using webgrader. Refer to the previous lab for instructions on

how to do this if you've forgotten. Demonstrate your graded program to a lab instructor and have them sign your worksheet.

4 Advanced Activity (Optional)

1. Another conditional operator is the ternary if-then-else operator. It is often used to choose between two values. For example:

   `int min = ( (a < b) ? a : b );` The syntax, `X ? Y : Z` is as follows: `X` is any conditional statement; if it evaluates to true, then the expression takes on the value `Y`; otherwise it takes on the value `Z`. Modify your programs to use this ternary operator where appropriate.

2. Change the calculator program as follows: add a menu option so that the user has the option to quit; then add a loop so that the program continues to print the menu. As long as the user performs an operation, it should continue until the user selects the quit option.