Programming Exercise 4:
Decisions Using Conditional Statements

Purpose: Practice the various ways Java programs can make decisions.

Background readings from textbook: Liang, section 3.1-3.6, 3.9

Due date for section 001: Monday, February 8 by 10 am
Due date for section 002: Wednesday, February 10 by 10 am

Overview

Many people think that computers are smart. In fact, they are not. What they do is process data very quickly. In order to make a computer appear smart, we have to write programs that can make decisions. How does a program make a decision? It uses a condition and based on the result of the condition, does one of a couple of things. What is a condition? A condition is a test that results in a true or false value. We use the condition to control what should be done. There are two types of statements that use conditions, selection statements (which we cover here) and loops (which we cover later in a couple weeks). A condition typically tests a variable against a value (or another variable). For instance, we might test to see if x is 0, if name is "Frank" or if y does not equal z. We use *relational* operators (equal to, not equal to, greater than, etc) to test numbers and we use some built-in methods in Java to test if Strings are equal or not, or if one String is greater than another. In this assignment, we will look at some basic conditions and use them in selection statements. In the next lab, we look at more complex forms of conditions.

Part 1: Exploring Conditions and Selections in Java

In Java, all conditions are placed in `( )`. We use <, >, <=, >=, ==, != as the 6 relational operators (less than, greater than, less than or equal to, greater than or equal to, equal, not equal). Notice for equal, we use == (two equal signs) because the single equal sign (=) is used for assignment. For instance, `x > 0` means "is x greater than zero", `x==0` means "is x equal to 0" and `x!=0` means "is x not equal to 0". Every condition evaluates to true or false.

Selection statements generally come in three flavors: if, if-else and nested if-else. The if statement tests a condition and if true, the computer executes the statement's "body". If false, the "body" is skipped. In either event, the instruction after the if is then executed. In the if-else, if the condition is true, the body of the "if" statement (called the "if clause") is executed and if not, the body after the if statement (called the "else clause" is executed. This is known as a 2-way selection because one of two things will happen. The if clause and/or the else clause can itself contain an if or if-else statement. This leads to a nested if-else statement. We typically use a nested if-else to support multiple conditions to give us a 3-way selection, or a 4-way selection, or more. There is no limit to the number of nested statements that we can use.

The one-way selection (the if statement): if the condition is true, the statement following it is executed.

```
if( age < 18 )
   System.out.println( "You are a minor." );
```

If there are multiple instructions in the if-clause, you need to enclose them all in a block. A block is denoted using { }. As a good habit, you should always use { } around your clause no matter if its one instruction or multiple instructions. We rewrite the above statement as follows.

```
if( age < 18 )
{
   System.out.println( "You are a minor." );
}
```

Two-way selection (the if-else statement): if the condition is true, the statement(s) following the condition is executed, otherwise the statement(s) following the word else (the else-clause) is executed.

```
if( age < 18 )
{
   System.out.println( "You are a minor." );
}
else
{
   System.out.println( "You are an adult." );
}
```

Three-way selection (nested if-else): if the condition is true, the statement(s) following it is executed, otherwise the second condition is tested and if it is true, the statement(s) following it is executed, otherwise the statement(s) after the else is executed.

```
if( age < 18 )
{
   System.out.println( "You are a minor." );
}
else if( age < 120 )
{
   System.out.println( "You are an adult." );
}
else
{
   System.out.println( "You are not human!" );
}
```

Note: Notice how we indent the code inside a block. The Java compiler does not care, but it makes the structure of the code clearer for human readers. Also, there is no semicolon after a closing brace! We will see that we will have many blocks in our code. Again, you can omit the { } if there is only a single instruction.

Part 2: Common Pitfalls

1.
```
if( age >= 18 );
{
   System.out.println( "You are an adult." );
}
```

Logic error!

There should be no semicolon. The compiler will give no error message since it thinks you mean
"if age ≥ 18, do nothing."
When your program runs, it will print "You are an adult" regardless of ***age****.*

2.
```
if( age = 18 )
{
   System.out.println( "You are 18 years old." );
}
```

Logic error!

The use of = is for assignment, not equality. Here, = has a strange effect on the program. First, age is set equal to 18 and then the condition is considered true so the output statement will always execute.

3.

```
if( age < 18 )
{
   System.out.println( "You are a minor." );
}
if( age < 120 )
{
   System.out.println( "You are an adult." );
}
else
{
   System.out.println( "You are not human!" );
```

Logic error!

*There is an **else missing before the second if statement**. The compiler will not give an error message. If **age** is 14, the program will first print "You are a minor." then also print "You are an adult."*

4.

```
if( age < 18 )
   System.out.println( "You are a minor." );
   System.out.println( "but you will grow up!" );
```

Logic error!

*You forgot the braces { }. The compiler will not give you an error message. Without braces, it will think there is only one statement under the **if**. It ignores indentation. The second println statement executes no matter what **age** is.*

5.

```
if( age < 18 )
{
   System.out.println( "You are a minor." );
}
else
{
   System.out.println( "You are an adult." );
}
else
{
   System.out.println( "You are not human!" );
}
```

Syntax error!

This is a situation known as an "else without if". The compiler found an else statement that does not appear to be connected to an if. The problem here is that the second else should really be "else if" with another condition. The second else is where the error arises.

Part 3: Problem

King's Island needs a program for its admission booths. When visitors to the park come up to the booth to purchase their tickets, the worker uses this program to figure out how much to charge them. You will write this program.

In the first version of the program, there is one ticket price of $30.00. Senior citizens (age ≥ 65) are given a 50% discount. Write this program as follows. Import your Scanner. Declare the needed variables (the person's age, the base price of a ticket ($30) and the price you will charge). Input the user's age, compute the price of the ticket and output the result in a formatted way (that is, using a $). You do not need to use DecimalFormat for this part of the program but you will as you enhance it so you might want to set this up now. Save, compile and run your program a few times, asking the user for different ages such as 10, 50, 65, 80 and 0.

Part 4: Program Modifications

The park wants to add further alterations to ticket costs. Children under 5 (less than 5, not less than or equal to) are free. Ticket prices are now $40 unless the person is from Warren County in which case the tickets are $30. Senior citizens still receive a 50% discount regardless of their county of residence. There are two input parameters now, one for age and one for county. For the county, input this as a String. See below about how to compare Strings. Calculate and output the ticket price. Ticket prices should be either 0, $15, $20, $30 or $40 depending on the person's age and location.

To compare Strings, we cannot use ==. Instead, we use a String message called `equals`. The format is string.equals(string2). For instance, if your input variable is called county, then to see if county is Warren, you would use `county.equals("Warren")`. There is also `equalsIgnoreCase` if you do not want to force the user to use proper capitalization. Save, compile and run your program using a variety of inputs to make sure it works correctly.

We have two additional changes to make to the program. The first is that we want to ensure that the user's age is valid. A valid age is anything > 0. Use an if statement that will test to see if `age <= 0` and if so, output an error message that it is an invalid age. Otherwise, continue with the remainder of your code. You will have to place the remainder of your code in a block.

Second, we want to give discounts to people from other counties as follows. Children under 14 from Clermont County get an 18% discount (over the base price of $40) and senior citizens (>= 65) from Campbell County get an additional 7.5% discount over their senior citizen discount. There are two ways you can do multiple tests. The first is to use two nested if statements. For instance, if you want to test to see if someone is male ('M') and age is greater than 20, you could do:

```
if(sex=='M')
      if(age>20)
```

Or, you can combine two tests in one conditional by ANDing the tests together. In Java, AND is indicated using `&&`. The above code can be rewritten as `if(sex=='M'&&age>20)`. Use whichever approach you prefer for the two new possibilities (Clermont children, Campbell seniors). Revise your program accordingly. Compile, run and test your program. At this point, make sure you are formatting your output to be of the form $xx.yy. Note: a 7.5% discount can be computed in two ways. You can do `amount=amount-amount*.075;` (that is, subtract from amount 7.5% of amount) or `amount=amount*.925;` (make amount be 92.5% of its current value).

As this is a challenging set of logic, you might want to write down on paper all the conditions, how to determine each one and how to compute the cost. Also, make sure you have commented your code well. If needed, ask your instructor for help. When done, run the final version of your program on the following inputs, copy the output to a text file and print out the source code and the text file (output) and hand it in to your instructor, or email both files to your instructor.

Run #	County	Age
1	Hamilton	12
2	Hamilton	72
3	Kenton	2
4	Warren	0
5	Clermont	35
6	Butler	4
7	Warren	24
8	Campbell	65
9	Clermont	10
10	Campbell	21
11	Hamilton	-15
12	Kenton	13