Chapter 3: Program Statements
Lab Exercises

Rock, Paper, Scissors

Program `Rock.java` contains a skeleton for the game Rock, Paper, Scissors.. Add statements to the program as indicated by the comments so that the program asks the user to enter a play, generates a random play for the computer, compares them and announces the winner (and why). For example, one run of your program might look like this:

```
Enter your play: R, P, or S
r
Computer play is S
Rock crushes scissors, you win!
```

Note that the user should be able to enter either upper or lower case `r, p`, and `s`. The user's play is stored as a string to make it easy to convert whatever is entered to upper case. Use a switch statement to convert the randomly generated integer for the computer's play to a string.

```
// ************************************************************
//     Rock.java
//
//   Play Rock, Paper, Scissors with the user
//
// ************************************************************
import java.util.Scanner;

public class Rock
{
    public static void main(String[] args)
    {
      String personPlay;    //User's play -- "R", "P", or "S"
      String computerPlay = "U";  //Computer's play -- "R", "P", or "S"
      int computerInt;      //Randomly generated number for computer play

      Scanner scan = new Scanner(System.in);

      //Get player's play -- note that this is stored as a string

      //Make player's play uppercase for ease of comparison

      //Generate computer's play (0,1,2). Use the Math.random() method

      //Translate computer's randomly generated play to string
      if (computerInt == 0)
          computerPlay = "R";
      else if (computerInt == 1)

            //...  Fill in rest of code

      //Print computer's play

      //See who won.  Use nested ifs instead of &&.
      if (personPlay.equals(computerPlay))
          System.out.println("It's a tie!");
      else if (personPlay.equals("R"))
               if (computerPlay.equals("S"))
                   System.out.println("Rock crushes scissors.  You win!!");
               else

            //...  Fill in rest of code
    }
}
```

Pizza Order

In this lab, we will be editing a pizza ordering program. It creates a Pizza order to the specifications that the user desires. It walks the user through ordering, giving the user choices, which the program then uses to decide how to make the pizza and how much the cost of the pizza will be. The user will also receive a $2.00 owner discount if his/her name is Mike or Diane.

Task #1

1. Open the file `PizzaOrder.java` contained in the Chapter 3 Lab zip file.
2. Compile and run `PizzaOrder.java`. You will be able to make selections, but at this point, you will always get a Hand-tossed pizza at a base cost of $12.99 no matter what you select, but you will be able to choose only pepperoni topping, and it should add into the price correctly. You will also notice that the output does not look like money. So we need to edit `PizzaOrder.java` to complete the program so that it works correctly.
3. Construct a simple `if` statement. The condition will compare the `String` input by the user as his/her first name with the first names of the owners, Mike and Diane. Be sure that the comparison is not case sensitive.
4. If the user has either first name, set the discount flag to `true`. This will not affect the price at this point yet.

Task #2

1. Write an `if-else-if` statement that lets the computer choose which statements to execute by the user input size (10, 12, 14, or 16). For each option, the cost needs to be set to the appropriate amount.
2. The default `else` of the above `if-else-if` statement should print a statement that the user input was not one of the choices, so a 12 inch pizza will be made. It should also set the size to 12 and the cost to 12.99.
3. Set user's crust choices using an `if-else-if` statement so it shows the correct choice in the final output.
4. Write additional statements to capture all the different topping choices and the output should reflect all the choices selected and the total should be updated as well.
5. Compile, debug, and run. You should now be able to get correct output for size and price (the output won't look like money, and no discount will be applied yet). Run your program multiple times ordering a 10, 12, 14, 16, and 17 inch pizza.

Task #3

1. Write an `if` statement that uses the flag as the condition. Remember that the flag is a `Boolean` variable, therefore is `true` or `false`. It does not have to be compared to anything.
2. The body of the `if` statement should contain two statements:
   a) A statement that prints a message indicating that the user is eligible for a $2.00 discount.
   b) A statement that reduces the variable cost by 2.
3. Compile, debug, and run. Test your program using the owners' names (both capitalized and not) as well as a different name. The discount should be correctly applied at this time.

Task #4

1. The `NumberFormat` class is used to format information so that it looks right when printed or displayed. Read pages 99-101 of your text book to learn about the `NumberFormat` class. Look at and use as an example the program on page 100 to see how money output is formatted. Modifiy your program to format money output correctly.

Counting and Looping

The program in `LoveCS.java` prints "`I love Computer Science!!`" 10 times. Compile and run it to see how it works. Then modify it as follows:

```
// ************************************************************
//     LoveCS.java
//
//     Use a while loop to print many messages declaring your
//     passion for computer science
// ************************************************************

public class LoveCS
{
    public static void main(String[] args)
    {
      final int LIMIT = 10;

      int count = 1;

      while (count <= LIMIT){
          System.out.println("I love Computer Science!!");
          count++;
      }
    }
}
```

1. Instead of using constant `LIMIT`, ask the user how many times the message should be printed. You will need to declare a variable to store the user's response and use that variable to control the loop. (Remember that all caps is used only for constants!)
2. Number each line in the output, and add a message at the end of the loop that says how many times the message was printed. So if the user enters 3, your program should print this:

```
1 I love Computer Science!!
2 I love Computer Science!!
3 I love Computer Science!!
  Printed this message 3 times.
```

3. If the message is printed N times, compute and print the sum of the numbers from 1 to N. So for the example above, the last line would now read:

   ```
   Printed this message 3 times.  The sum of the numbers from 1 to 3 is 6.
   ```

   Note that you will need to add a variable to hold the sum.

Characters

1. Write a program `Chars.java` that asks the user for a string of characters and prints it one character per line.

   For example:

```
Please enter a string of characters:  This is fun!
T
h
i
s

i
s

f
u
n
!
```

Odd and Even

1. Write a program `OddEven.java` that asks the user for a positive integer and prints the number of odd, even, and zero digits.

   For example:

   ```
   Please enter a positive integer:  223049

   There are 3 even digits.
   There are 2 odd digits.
   There is 1 zero digit.
   ```

Be sure to use proper grammer in your output (`is/are` and `digit/digits`)!

Powers of 2

File `PowersOf2.java` contains a skeleton of a program to read in an integer from the user and print out that many powers of 2, starting with `2^0` or 2^0.

1. Using the comments as a guide, complete the program so that it prints out the number of powers of 2 that the user requests. **Do not use** `Math.pow` **to compute the powers of 2!** Instead, compute each power from the previous one. (`nextPowerOf2 *= 2;`) For example, if the user enters 4, your program should print this:

   ```
   Here are the first 4 powers of 2:
   1
   2
   4
   8
   ```

2. Modify the program so that instead of just printing the powers, you print which power each is, e.g.:

   ```
   Here are the first 4 powers of 2:
   2^0 = 1
   2^1 = 2
   2^2 = 4
   2^3 = 8
   ```

```
// ****************************************************************
//     PowersOf2.java
//
//     Print out as many powers of 2 as the user requests
//
// ****************************************************************
import java.util.Scanner;

public class PowersOf2
{
    public static void main(String[] args)
    {
        int numPowersOf2;        //How many powers of 2 to compute
        int nextPowerOf2 = 1;    //Current power of  2
        int exponent;            //Exponent for current power of 2 -- this
                                 //also serves as a counter for the loop
        Scanner scan = new Scanner(System.in);

        System.out.println("How many powers of 2 would you like printed?");
        numPowersOf2 = scan.nextInt();

        //print a message saying how many powers of 2 will be printed
        //initialize exponent -- the first thing printed is 2 to the what?

        while (  )
        {
            //print out current power of 2

            //find next power of 2 -- how do you get this from the last one?


            //increment exponent

        }
    }
}
```

A Guessing Game

File `Guess.java` contains a skeleton for a program to play a guessing game with the user. The program should randomly generate an integer between 1 and 10, then ask the user to try to guess the number. If the user guesses incorrectly, the program should ask them to try again until the guess is correct; when the guess is correct, the program should print a congratulatory message.

1. Using the comments as a guide, complete the program so that it plays the game as described above.
2. Modify the program so that if the guess is wrong, the program says whether it is too high or too low. You will need an `if` statement (inside your loop) to do this.
3. Now add code to count how many guesses it takes the user to get the number, and print this number at the end with the congratulatory message.
4. Finally, count how many of the guesses are too high and how many are too low. Print these values, along with the total number of guesses, when the user finally guesses correctly.

```
// *************************************************************
//     Guess.java
//
//     Play a game where the user guesses a number from 1 to 10
//
// *************************************************************
import java.util.Scanner;
import java.util.Random;

public class Guess
{
    public static void main(String[] args)
    {
      int numToGuess;       //Number the user tries to guess
      int guess;            //The user's guess

      Scanner scan = new Scanner(System.in);
      Random generator = new Random();

      //randomly generate the  number to guess

      //print message asking user to enter a guess

      //read in guess

      while (  )  //keep going as long as the guess is wrong
        {
          //print message saying guess is wrong
          //get another guess from the user
      }


      //print message saying guess is right
    }
}
```

Sample Output:

```
I'm thinking of a number!!
Enter your guess: 5
Too high, guess again: 4
Too high, guess again: 3
Too high, guess again: 1
You got it, after only 4 guesses!!
3 guesses were too high.
0                guesses                were                too                low.
```

Factorials

The *factorial* of n (written n!) is the product of the integers between 1 and n. Thus 4! = 1*2*3*4 = 24. By definition, 0! = 1. Factorial is not defined for negative numbers.

1. Write a program `Factorial.java` that asks the user for a non-negative integer and computes and prints the factorial of that integer. You'll need a while loop to do most of the work—this is a lot like computing a sum, but it's a product instead. And you'll need to think about what should happen if the user enters 0.

2. Now modify your program so that it checks to see if the user entered a negative number. If so, the program should print a message saying that a nonnegative number is required and ask the user the enter another number. The program should keep doing this until the user enters a nonnegative number, after which it should compute the factorial of that number. **Hint:** you will need another while loop **before** the loop that computes the factorial. You should not need to change any of the code that computes the factorial!

Divide GUI

The goal or this laboratory is to introduce you to the flow of control statements that are common to every programming language using Java's GUI components.

Compile and run "`DivideTwoGui.java`" and observe the output. The program prompts the user for two integers that represent the numerator and the denominator of a fraction. JOptionPane.showlnputDialog returns a string representing the user input and immediately parses into an integer format to be used in the calculation.

Modify the program to include an `If/Else` statement to check for division by zero. If the denominator is not equal to zero, display the result of the division, otherwise display a message to the user that division by zero is not allowed.

```
import javax.swing.*;

public class DivideTwoGui
{
    public static void main(String[] args)
    {
      String input;
      JOptionPane.showMessageDialog(null, "This program divides two numbers.");
      input = JOptionPane.showInputDialog("Enter the numerator:");
      int numerator = Integer.parseInt(input);
      input = JOptionPane.showInputDialog("Enter the denominator:");
      int denominator = Integer.parseInt(input);
      JOptionPane.showMessageDialog(null, numerator + "/" + denominator + " = " +
                                    (double) numerator/denominator);
      System.exit(0);
    }
```