Chapter 3: Conditionals and Loops
Lab Exercises – Part 2

Topics	**Lab Exercises**
The **if** statement	Rock, Paper, Scissors
The **while** statement	Counting and Looping Powers of 2 Factorials A Guessing Game Rock, Paper, Scissors - revisited

Rock, Paper, Scissors

Program *Rock.java* contains a skeleton for the game Rock, Paper, Scissors. Open it and save it to your directory. 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:

```
$ java Rock
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 if statements 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;  //Computer's play -- "R", "P", or "S"
      int computerInt;      //Randomly generated number used to determine
                            //computer's play

      Scanner scan = new Scanner(System.in);

      // Add the code for each action beneath the comment.
      // more than one line may be needed in some cases

      //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 Math.random()

      //Translate computer's randomly generated play to string
      // using mutually exclusive conditional statements

      //Print computer's play

      //See who won.  Use nested ifs instead of &&.

      }
}
```

Counting and Looping

The program in *LoveCS.java* prints "I love Computer Science!!" 10 times. Copy it to your directory and 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.

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 20.

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 (how do you get 2^n from 2^{n-1}?). 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

        }
    }
}
```

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 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!

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 100, 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. Include the necessary “Hey Dummy!” loops to make sure the initial guess is between 1 and 100, and when told too high that the next guess is actually lower, and when told too low that the next guess is actually higher.
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
    }
}
```

Rock, Paper, Scissors - revisited

Copy your Rock.java program to Rock2.java (be sure to change the class name, as well)

People hardly ever play Rock, Paper, Scissors one game at a time. They play best 2 out of 3, or 3 out of 5, etc

Change your program to prompt the user for the odd number of games they want to play and use a “Hey Dummy” loop to make sure it is an odd number.

Then, add a loop to your program to play the game multiple times. Keep track of who wins, and make sure the loop stops when either the computer or the user has accumulated enough wins.

Also, add “Hey Dummy” loops to make sure the user only enters an R, P, or S.

Extra Credit Challenge (will be used as Formative Replacement Points)

Copy your program to Rock3.java.

Modify your program to play Rock, Scissors, Paper, Lizard, Spock.   It must use the correct outcomes and the correct language for why one objects beats another.