Lab 8: More Conditionals and Loops!

Let's get started!

Today we'd like you to:

1. **Open Eclipse**. Remember to keep all your projects in the same workspace. Think of a workspace as a shelf, and each project is a binder of stuff on the shelf. If you need some help organizing your workspace, we'd be happy to help!
2. **Create a new nonmodular Java project** Call it Comp1510Lab08*LastNameFirstInitial*
3. **Complete the following tasks**. Remember you can right-click the src file in your lab 8 project to quickly create a new Java class.
4. When you have completed the exercises, **show them to your lab instructor**. Be prepared to answer some questions about your code and the choices you made.

What will you DO in this lab?

In this lab, you will:

1. Repeat blocks of code using while, do-while, and the for loop
2. Make decisions using if, if-else, and switch
3. Create a guessing game and a rock, paper, scissors game
4. Validate a date (including leap year).

Table of Contents

1. Games

Now that we know how to create loops and make decisions, we can start making games. Let's start with a pair of easy, fun games that use the Random class and luck:

1. Create a new class called **Games** inside package ca.bcit.comp1510.lab08:
    i. Do not include a **main method** inside the class definition. Remember the main method is what gets executed first when we start our program. In this case, the Games class is not a complete program.
    ii. Include a **Javadoc** comment at the top of the class. The Javadoc comment should contain:
        1. The name of the class and a (very) short description
        2. An @author tag followed by your name
        3. An @version tag followed by the version number.
2. The games we play will need the following elements. Create private instance variables to store each of them:
    i. user's score (an int)
    ii. a Scanner object to get input from the user
    iii. a Random object to generate random numbers
3. Create a constructor that:
    i. initializes the user score to 0
    ii. instantiates the Scanner object
    iii. instantiate the Random object
4. There's no main method. How do we play? Create a **Driver** class that contains a main method inside the same package. Don't forget to give it comments. In the main method, declare and instantiate a Games object, like this:

```
/**
 * Drives the program.
 * @param args unused
 */
public static void main(String[] args) {
   Games myGame = new Games();
   myGame.play();

}
```

5. **The Games class needs a public play method. Let's create one now. Create a public method called `public void play()` inside Games:**
    i. This method must continuously print the following menu:

```
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
>
```

    ii. After printing the menu, use the Scanner to get the user's choice:
        1. If the user chooses 1, print their score
        2. If the user chooses 2 or 3, do nothing right now

3. If the user chooses 4, end the game (just return from the play method)

iii. If the user chooses anything else, tell them to make another choice.

iv. The following output is how your play() method should behave:

```
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 2
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 3
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 1
Your score is 0
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 5
That's not a valid choice!
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> -1
That's not a valid choice!
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 4
```

6. Let's make our first game. Modify the `play()` method so that when the user selects 2, a method called **`public void guessANumber()`** is called. The guessANumber game is easy. The users must guess a number between 0 and 100

(inclusive). The game tells the user if their guess is too high or too low. The user can keep guessing until they get the right number. If the user can guess in fewer than five guesses, add five points to their score.

i. Here is some sample output for the number guessing game. Remember: when the user finally guesses the number, if they have made the guess in five guesses or fewer, add five points to their score, otherwise there is no penalty:

```
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 2
I've picked a random number between 0 and 100
Can you guess it?
Guess the number!
50
Too high, guess again!
Guess the number!
25
Too low, guess again!
Guess the number!
38
Too high, guess again!
Guess the number!
31
Too high, guess again!
Guess the number!
28
RIGHT!
Five points!
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
>
```

7. Now it's time for our second (and final game): Rock, Paper, Scissors, a traditional North American child's game. Modify your play() method so that if the user selects 3, a method called **`public void rockPaperScissors()`** is called. This method plays one round of Rock, Paper, Scissors:

   i. Use the random number generator to generate a 0, 1, or a 2. The zero will represent a Rock, 1 will represent Paper, and 2 will represent Scissors.

   ii. Ask the user to enter their choice: Rock, Paper, or Scissors. Wrap this in a loop so that if the user enters something else, they are prompted to try again. Your program should behave like this:

```
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
```

```
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 3
I've picked one of ROCK, PAPER, and SCISSORS
Which one do you choose?
> tomato
That's not a valid choice! Try again!
pencil
That's not a valid choice! Try again!
scissor
That's not a valid choice! Try again!
scissors
```

iii. Once the user has made a valid choice, compare it to the computer's choice. The rules are:
   1. A rock smashes scissors, ties with another rock, but loses to paper.
   2. Paper wraps rock, ties with other paper, but loses to scissors.
   3. Scissors cut paper, tie with other scissors, but lose to rock.
   4. (optional) you can implement the Rock, Paper, Scissors, Lizard, Spock variant.

iv. If the user wins, tell them and add 5 points to their score

v. If the user loses, tell them and take 3 points off their score

8. When we put it all together, we end up with a Driver program that instantiates a Games object. The Games object contains a play method that keeps asking the user to play a game or display their score until the user quits. The user can select a number guessing game or a round of Rock, Paper, Scissors. Your game should behave like this:

```
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 1
Your score is 0
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 2
I've picked a random number between 0 and 100
Can you guess it?
Guess the number!
50
Too low, guess again!
Guess the number!
75
Too high, guess again!
```

```
Guess the number!
58
Too high, guess again!
Guess the number!
54
Too low, guess again!
Guess the number!
56
Too high, guess again!
Guess the number!
55
RIGHT!
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 1
Your score is 0
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 3
I've picked one of ROCK, PAPER, and SCISSORS
Which one do you choose?
> sleep
That's not a valid choice! Try again!
#imsotired
That's not a valid choice! Try again!
rocks
That's not a valid choice! Try again!
rock
Nope, I picked Paper
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 1
Your score is -3
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 3
I've picked one of ROCK, PAPER, and SCISSORS
Which one do you choose?
```

```
> paper
Nope, I picked Scissors
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 1
Your score is -6
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 3
I've picked one of ROCK, PAPER, and SCISSORS
Which one do you choose?
> iamawfulatthis
That's not a valid choice! Try again!
paper
Yes!  Paper wraps rock
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 1
Your score is -1
Welcome to the Games program!
Make your choice (enter a number):
1. See your score
2. Guess a number
3. Play Rock, Paper, Scissors
4. Quit
> 4
```

2. Finish with some Date Validation

Many calendars have been used by people over the years and across time. Among the calendars used in Canada is the Gregorian calendar which was introduced to correct a cumulative error in the Julian calendar used by the Romans in 24 February 1582 and adopted at different times by different countries, see https://en.wikipedia.org/wiki/Gregorian_calendar

In this exercise you write a program that checks to see if a date entered by the user is a valid date in the Gregorian calendar. This means it is not before 24 February 1582 and has a valid day, month, year. (A skeleton of the program is in Date.java (see file in learning hub). Open this program and save it to your project. As indicated by the comments in the program, code the

constructor and methods and use them in the main method with the following logic (some of which will be in methods, others in the main procedure):

1. Statements that prompt and read the input values.
2. An assignment statement that sets monthValid to true if the month entered is between 1 and 12, inclusive.
3. An assignment statement that sets yearValid to true if the year is between 1582 and 2999, inclusive.
4. An assignment statement that sets leap Year to true if the year is a leap year. Here is the leap year rule (there's more to it than you may have thought!):

If the year is divisible by 4, it's a leap year UNLESS it's divisible by 100, in which case it's not a leap year UNLESS it's divisible by 400, in which case it is a leap year. If the year is not divisible by 4, it's not a leap year.

Put another way, it's a leap year if a) it's divisible by 400, or b) it's divisible by 4 and it's not divisible by 100. So 1600 and 1592 are leap years, but 1700 and 1594 are not.

4. An if statement that determines the number of days in the month entered and stores that value in variable daysInMonth. If the month entered is not valid, daysInMonth should get 0. Note that to figure out the number of days in February you'll need to check if it's a leap year.
5. An assignment statement that sets dayValid to true if the day entered is legal for the given month and year.
6. If the month, day, and year entered are all valid and greater or equal to 24 February 1582, print "Date is valid" and indicate whether or not it is a leap year. If any of the items entered is not valid, just print "Date is not valid" without any comment on leap year.
7. When you create a Date object, the constructor will need to go through the above logic as well. How many of the above statements in the main method can be replaced just by using the Date constructor to create a Date object?

Write a JUnit test program, called DateTest for your Date class. Put it into the same package but in a test folder (i.e. not in the src folder). Remember to add the test folder in the source tab of the Java build path window in Eclipse. What things do you need to test?

3. You're done! Submit your work.

If your instructor wants you to submit your work in the learning hub, export it into a Zip file in the following manner:

1. Right click the project in the Package Explorer window and select export…
2. In the export window that opens, under the General Folder, select Archive File and click Next
3. In the next window, your project should be selected. If not click it.
4. Click *Browse* after the "to archive file" box, enter in the name of a zip file (the same as your project name above with a zip extension, such as

Comp1510Lab08BloggsF.zip *if* your name is Fred Bloggs) and select a folder to save it. Save should take you back to the archive file wizard with the full path to the save file filled in. Then click Finish to actually save it.

5. Submit the resulting export file *and* your answers document as the instructor tells you.