—Assignment #1—

Make sure to review the submission requirements on pg. 4 of the Lab #1 document.

Follow the Style Guide as given in the Resources section of our conneX course page.

Learning Outcomes

When you have completed this assignment, you should understand:

- How to design, compile, run and check a simple and complete Java program on your own.
- The effect of escape sequences on printed strings.
- How to write basic static methods.
- The flow of control (i.e. the effects of method calls and expression evaluation).
- How to format and document a Java program.

The Golden Ratio is used to help design aesthetically pleasing shapes by artists and architects, and also shows up in a surprising number of ways in nature. The Golden Ratio is also called the "Divine Proportion." It is an irrational number (it cannot be written as the ratio of two integers), and is typically symbolized with the Greek letter ϕ (phi). As a continued fraction pattern, it is the simplest continued fraction, as it consists of all 1's:

$$\phi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ddots}}}} \approx 1.618033988749895\ldots$$

The pattern in the denominator repeats infinitely. The Golden Ratio is not typically included as a constant in part of a programming language's package library, such as π = `Math.PI` in Java. One possible application of ϕ is as some part of a procedural graphics program (artwork generated by math formulae).

Write a Java program called `GoldenRatio.java` that does the following.

Create a variable of type `double` called `phi`.
Set `phi` = (1 + Math.sqrt(5)) / 2.

Estimate ϕ:

(a) create a variable of type `double` called `approxPhi`,

(b) set `approxPhi` to 1.0,

(c) set `approxPhi` to its reciprocal plus 1.0,

(d) repeat Step (c) four more times.

Then print the following statements:

```
The Golden Ratio is approximately <value of approxPhi here>.
The Golden Ratio is <value of phi here>.
```

Compile your program and make sure there are no errors, and you have the above output. Your approximation should agree with the actual value of ϕ in the first two digits.

Next, approximate ϕ again in your program:

(a) set `approxPhi` to 1.0,

(b) set `approxPhi` to its reciprocal plus 1.0,

(c) repeat Step (b) ninety-nine more times.

Then print the following statements:

```
Now for a more accurate approximation.
The Golden Ratio is approximately <value of approxPhi here>.
The Golden Ratio is <value of phi here>.
```

The Golden Ratio can be used to generate a famous sequence of numbers called the Fibonacci numbers. The first Fibonacci number is $f_1 = 1$, and the second is $f_2 = 1$. Any Fibonacci number is the sum of the previous two: $f_{n+2} = f_{n+1} + f_n$, for $n \geq 3$. Observe the first nine Fibonacci numbers:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$$

After your previous code, calculate these Fibonacci numbers as follows. Declare and assign the following variables:

- `int f1 = 1;`
- `int f2 = 1;`
- `int fn = 0;`

Write a for-loop with a loop-control variable `i` that iterates from 3 to 9, inclusive, with each iteration performing the following steps:

(a) set `fn` equal to the sum of `f1` and `f2`

(b) set `f1` equal to `f2`

(c) set `f2` equal to `fn`

(d) `System.out.println("f_" + i + " = " + fn);`

(e) call `checkFibonacci( approxPhi, i )` as described below,

(f) call `starArt( f1, f2 )` as described below.

Now add the method `checkFibonacci` to your program:

```
public static void checkFibonacci( double approxPhi, int n ) {
    double A = Math.pow(approxPhi, n);
    double B = Math.pow(1 - approxPhi, n);
    double Fibonacci = (A - B) / Math.sqrt(5);
    long check_fn = Math.round( Fibonacci );
    System.out.println("check f_" + n + " = " + check_fn);
}
```

Add another method `starArt` to your program that takes two integer parameters, a `height` and a `width`. Inside `starArt`, use two nested loops to print as output a rectangle of star (*) characters, with just less than `2.5*width` characters per line of `height` number of lines. (The terminal/console window typically has non-square character sizes, and so we need this adjustment in width to get proper visual comparison of width against height here.)

Compile your program and make sure there are no errors. The output should give the 3rd to the 9th Fibonacci number, with each number followed by a rectangle of stars in an aspect ratio that is said to be more pleasing to the eye than other aspect ratios.

The ratio of Fibonacci numbers f_{n+1}/f_n also approximates ϕ as n increases. The Golden Ratio and the Fibonacci numbers can be used in many ways to hold the human eye's attention. The arrangement of branches of many trees loosely follow the sequence of Fibonacci numbers in efforts to maximize the amount of light their leaves can photosynthesize. The Mona Lisa—arguably the world's most famous work of art—is among the notable applications of the Golden Ratio.

The Fibonacci numbers are named after an Italian mathematician born in the year 1170, but appeared in history much earlier in the work of Indian mathematics. Fibonacci used this sequence of numbers to describe the growth of a hypothetical rabbit population. As a tribute to this idea, add one more method to your program called `printRabbit` that outputs the following ascii art rabbit:

```
(\(\
( -.-)
o_(")(")
```

This rabbit art uses the backslash, dash, period, double quote, underscore, and lowercase o characters, as well as left and right parenthesis. Make sure to call `printRabbit` at the end of your `main` method.

Grading

Marks will be allocated out of 30 points:

- (0 points) You do not submit any program file.
- (5 points) Your program does not compile. The marker cannot fix the complier error easily, or if they can, the output is severely mismatched with the assignment requirements.
- (10 points) Your program does not compile, but the marker can fix the error easily.
- (15 points) Your program compiles, but only outputs a few of the assignment requirements.
- (20 points) Your program compiles, outputs most of the assignment requirements, and is missing documentation comments or has formatting issues.
- (25 points) Your program compiles, outputs everything required, but is missing documentation comments or has formatting issues.
- (30 points) Your program compiles, outputs everything required, and has proper documentation and formatting.