Assignment 05 – Getting Loopy

Due: Friday, November 3, 2017 @ 11:59pm

Repetition is important. And mentally exhausting. Let's practice getting good at it. (The repetition thing, not the exhaustion thing.)

Since being able to read code in languages you aren't familiar with is a useful skill, you will be translating loops from other well-known programming languages into Java. Don't panic.

Gender Bias Challenge

Good luck with that playing God thing.

Adapted this from a Martin Gardner book - Entertaining Mathematical Puzzles - where he explains he himself got it from a book by George Gamow and Marvin Stern.

Image there's some jerkwad misogynistic military leader of some kingdom that rules his people with an iron fist. He decrees that in order to increase the size of his (males-only) army, that from this point forward, a woman can only have multiple children if she gives birth to a boy - if she gives birth to a girl, she can no longer have any children.

Assuming that the people follow this rule without question, what effect do you think this policy will have on the male:female ratio over time? Let's build a simulation to find out!

MAKE THIS HAPPEN:

1. Create a table showing the results of running 10 simulations, each consisting of the ratio of boys to girls born to 10,000 mothers under this policy.
    a. The first row is this header row: `Run#[2 spaces]M[1 space]:[1 space]F[newline]`
    b. The next 10 rows have this format:
        i. `<run#>[2 spaces]1[1 space]:[1 space]<# females to 1 male>[newline]`
        ii. 1. The `<run#>` part is in a field of width 4.
        iii. 1. The `<# females to 1 male>` part must be rounded to 5 decimal places. (Ex. 2.13001)
    c. If you deviate from this format, the automated tests will fail.

Example output (doesn't show correct numbers, though!):

```
Run#  M : F
   1  1 : 2.00132
   1  1 : 1.92931
    etc…
```

NOTES:

- We'll assume that every birth results in a single child (no twins, triplets, etc.), that every child lives, and that the odds of having a boy is 50%.
- You'll likely find it useful to look through section 4.10 (Application: Random Numbers and Simulations) in the text.
- Taking a skim through the Random class in the Java 8 API would be useful. Some methods you'll find there are more readable than others, especially for this challenge.
- You **must** use `Random randomGenerator = new Random(1);` when you are ready to test your code with Cucumber. This way, your results (if you've coded things correctly) will match what the tests are looking for. (Do you understand why?) If you're **not** running the automated tests and just want to play with your code manually, you should use `Random()`.

GPA Chart Challenge

It's full of stars.

A small college wants to be able to quickly see how well their students are doing. You have been contracted to design, then implement a program to create a program that prints out a simple graphical representation of the GPA distribution of students.

At this college, a student earns Honours status if the GPA is 3.6 or above. They fail if it is under 2.0.

Your boss likes asterisks. A lot. So you decide to use asterisks to do the graphical representation, 'cause sucking up and all.

MAKE THIS HAPPEN:

1. Prompt for and read in the number of students.
   a. You can assume that the user enters in an integer – but you can't assume that it will be a valid one!
   b. If the user enters in a number <= 0, give a descriptive error message and terminate the program.
2. Read in the GPA for each student. Don't prompt for these numbers.
   a. You can assume all GPAs entered are legal, that is, values >= 0.0 and <= 4.0.
   b. Each GPA is separated from the next by white space (spaces, tabs, newlines).
3. Print out a graph that shows the distribution of GPA categories for those students. (see the example below)

Examples

Assume that there were 10 students: 3 with Honours, 5 who passed, and 2 who failed. Then we would output the following:

```
GPA Distribution

Honours   :***
Pass      :*****
Fail      :**
```

Assume that there were 5 students: 1 with Honours, 4 who passed, and 0 who failed. Then we would output the following:

```
GPA Distribution

Honours   :*
Pass      :****
Fail      :
```

NOTES:

1. Your graph should have the title GPA Distribution.
2. The label field (with 'Honours', 'Pass', 'Fail') should have a width of 10.
3. If a GPA category has 0 students in it (nobody failed, for example), you still need to print the label and colon – you just won't print any stars.
4. You will only be given *one* automated test for this challenge - you are expected to come up with more on your own. You are strongly encouraged to test your code manually. *Look carefully at the output produced - you must match it exactly for your code to pass **our** automated tests!*

Go Loop Translation Challenge

Lost in Translation.

“If I had to describe Go with one word it’d be ‘sensible’.” – Christoffer Hallas

MAKE THIS HAPPEN:

Turn this Go loop into its Java equivalent.

You are encouraged to play with this code. You can type in your input in the Input section and then type Ctrl+Enter to run the code on that input. The result will appear in the Output section. If you create a situation that causes an infinite loop, you should refresh the page.

```
package main
import "fmt"

func main() {
    const STARTING_LIFE = 100

    lifeLeft := STARTING_LIFE
    isAlive := true
    changeDisplay := ""

    for {
        var lifeChange int
        fmt.Scanf("%d", &lifeChange)

        changeDisplay = "HEAL"
        if lifeChange <= 0 {
            changeDisplay = "DMG"
        }

        lifeLeft += lifeChange

        if lifeLeft > STARTING_LIFE {
            lifeLeft = STARTING_LIFE
        } else if lifeLeft <= 0 {
            lifeLeft = 0;
            isAlive = false
        }

        if lifeChange < 0 {
            lifeChange *= -1
        }

        fmt.Printf("%5s(%03d): LIFE %03d\n", changeDisplay, lifeChange, lifeLeft)

        if !isAlive {
            break;
        }
    }

    fmt.Println("Dead now.")
}
```

NOTES:

1. Go syntax hints:
   - Go uses `:=` instead of `=` for assignment.
   - Go uses `const` to declare constants.
   - `fmt.Scanf("%d", &var)` reads an integer from the keyboard and stores in into variable `var`.
   - Go's `Printf` works just like Java's `printf`.
   - Go doesn't have a while loop; instead a for loop is used that breaks out when a certain exit condition is true.
2. Do **not** use a break to accomplish this task.
3. Although Go does not have while loops, Java does - so do **not** use a for loop to accomplish this task.
4. You will only be given *one* automated test for this challenge - you are expected to come up with more on your own. You are strongly encouraged to test your code manually. *Look carefully at the output produced - you must match it exactly for your code to pass* ***our*** *automated tests!*

Submission Instructions

Create a folder named `Asg05_`*`Lastname_firstname`* and place your BlueJ project (the contents of the `solution` folder) in the folder. Submit your source code to the submit folder (`I:\Labs\CompSci\Submit\1501\00x`). If you are submitting your files via the web interface from off campus, you will need to compress your files into one .zip file.