CS 570: Programming Foundations
Fall 2017

Homework Assignment #4

Due date: Wednesday, October 25, 2017 by 11:59 PM.

Note: This and all assignments given in this course can be and must be solved using **only** the materials that have been discussed in class. Do not look for alternative methods that have not been covered as part of this course.

Program 1 (60 points):

Rangers at the Serengeti National Park are in need of a program to simulate the ecological balance between lion and gazelle populations. Gazelles eat from an unlimited supply of grass; lions eat gazelles. Lions are the only obstacle to increasing population growth for the gazelles. The population of lions increases as the population of gazelles increases.

You have been asked to write a program to calculate the populations of gazelles and lions over a 1000-day period with population counts displayed every 100 days only. The program must request from the user the initial numbers of gazelles and lions. The following formulas are to be used for calculating the day-to-day changes in the gazelle (G) and lion (L) populations:

$$G_{Tomorrow} = (1 + a) \cdot G_{Today} - c \cdot G_{Today} \cdot L_{Today}$$
$$L_{Tomorrow} = (1 - b) \cdot L_{Today} + c \cdot d \cdot G_{Today} \cdot L_{Today}$$

Where:

a = 0.01	Fractional increase in gazelle population without competition from lions
b = 0.008	Fractional decrease in lion population without gazelle to eat
c = 0.00005	Likelihood that a lion will encounter and eat a gazelle
d = 0.015	Fractional increase in lion population attributed to a devoured gazelle.

Finally, as summary information, print the highest individual daily population number of lions and of gazelles and the corresponding average values over the 1000-day time period.

Program details:

The program should welcome the user and prompt the user for input data with clear understandable messaging. Add proper error handling for erroneous input (e.g. values less than one). Make sure the user gets an explanatory error message if the supplied input is invalid. In case of invalid input, the user should be asked to try again until the input provided is valid.

The number of iterations (1000 days) must be defined as a declared constant integer. Values for **a**, **b**, **c**, and **d** have to be defined as constant as well. Choose the proper data types.

The program must use a repetition structure that loops 1000 times. Within the loop, using today's values for **G** and **L**, compute the corresponding tomorrow values for **G** and **L**. While it is impossible to have a portion of a gazelle or of a lion, use `double` data types for all **G** and **L** values anyway.

Use a selection structure to identify a 100th iteration, allowing displaying on the monitor the population values for that iteration.

Use selection structures to determine the highest number individual daily population of both lions and gazelles.

You also need to determine the overall sum of gazelles and lions (for calculating the average) over the 1000-day period to be displayed as summary information.

Summary information has to be printed to the output. Here is a sample run of a typical solution to this problem:

```
This program simulates the ecological balance between lion and gazelle populations
********************************************************************************

Please enter the initial number of gazelles (a value greater than zero): 0
You must enter a value greater than zero. Try again.

Please enter the initial number of gazelles (a value greater than zero): 100

Please enter the initial number of lions (a value greater than zero): -9
You must enter a value greater than zero. Try again.

Please enter the initial number of lions (a value greater than zero): 10

Gazelle vs. Lion population periods of 100 days
Day      Gazelles    Lions
---------------------------------
100      261.35      4.54
200      695.92      2.10
300      1868.44     1.03
400      5034.46     0.59
500      13582.62    0.50
600      36608.72    1.27
700      94150.93    53.13
800      1837.37     1519.48
900      21.57       697.13
1000     5.22        312.47

Average number of gazelles: 15524.98
Average number of lions: 253.77
Maximum number of gazelles: 103654.24
Maximum number of lions: 1678.19

End of population simulation program
```

Note on formatting output to print the table:

In order to be able to display the table in a pleasant and easy to read format (e.g. population figures with 2 decimal places, the three columns under the headings and aligned to the left, etc.) you may want to use either **`System.out.printf()`** or **`System.out.format()`** methods of **`System.out`**. These output methods allow you to specify formatting information for the objects to be displayed, such as the size of the printing field, the alignment (left, right), and the number of decimal places after the decimal point (among other things)

The first parameter in either one of these methods is a string that specifies the format. The format is specified using converters and flags. For example:

```
double pi = Math.PI;
System.out.format("%-4s%10.3f%n", "pi: ", pi);
```

will display **pi** as a number with 3 decimal places after the decimal point, using a printing field of size 10, aligned to the left:

```
pi:       3.142
```

In this example:

%-4s indicates that the object to be displayed is a string (**s**), that will be printed in a field of size **4**, aligned to the left (**-**) of the printing space.

%10.3f indicates that the object to be displayed is a floating-point number (**f**), that it should be displayed in field of size **10**, and with **3** decimal places after the decimal point.

%n is to advance to the next line after displaying **pi** (similar to **\n**)

Converts commonly used include **f** for floating-point numbers, **s** for strings, **n** for new line. Flags commonly used include **–** to align to the right.

You can find more information at:

https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

Program 2 (40 points):

Write a recursive method that:

- Receives a integer number **n** as its parameter
- Prints the out all the even numbers between **n** and **0** (inclusive)
- Returns the number of even number between **0** and **n**, for any positive integer **n**

Make sure that you include a **base case**. What is the **recursive case**? Let the recursive case solve a simpler problem.

Test your recursive method in a **`main`** method that:

- Asks the user to enter a positive integer value, or 0 to end the program
- Validates the user's input: if the user enters a negative value, the program should issue an error message and ask the user to try again until a valid value is entered
- Calls the recursive method and prints the returned value
- Goes back to ask the user to enter a positive integer value, or 0 to end the program (*hint*: you need a loop)

Here is a sample run:

```
Enter a positive integer or 0 to end: -9
Error: you entered a negative value. Try again.

Enter a positive integer or 0 to end: 9
The even integers between 0 and 9 are: 8 6 4 2 0
There are 5 even integers between 0 and 9

Enter a positive integer or 0 to end: 4
The even integers between 0 and 4 are: 4 2 0
There are 3 even integers between 0 and 4

Enter a positive integer or 0 to end: 21
The even integers between 0 and 21 are: 20 18 16 14 12 10 8 6 4 2 0
There are 11 even integers between 0 and 21

Enter a positive integer or 0 to end: 0
Bye
```

Note: Please make sure to submit well-written programs for these programming tasks. Good identifier names, useful comments, indentation, and spacing will be some of the criteria that will be used when grading this assignment.

Grading

Criteria for Program 1	Points
Code Compiles Correctly – no compilation errors	15
Code runs properly on several tests and produces the correct results	10
Program Construction: all the required parts are there and logical flow is correct, including calculations, the use of selection statements and loops.	10
User's input is validated	10
Calculation results and table are formatted appropriately	5
User Interface: the program provides clear instructions for the user so he/she understands what the program does, what input need to be provided and what the output of the program means.	5
Code Style: good identifier names, named constants as needed, comments, indentation and spacing	5

Criteria for Program 2	Points
Code Compiles Correctly – no compilation errors	6
Code runs properly on several tests and produces the correct results	6
Program Construction: all parts are there and logical flow is correct, including loops, and selection statements.	6
Recursive method correctly implemented.	8

User's input is validated	5
User Interface: the program provides clear instructions for the user so he/she understands what the program does, what the output of the program means.	4
Code Style: good identifier names, comments, indentation and spacing	5

How to submit your assignment

- Assignments must be submitted via Blackboard Learn.
  - Please note that assignments submitted via email will not be accepted.
  - Late assignments will not be accepted. Your work must be uploaded and submitted by 11:59 PM on the date it's due.

- For this assignment you must submit your source code, that is your **`.java`** files:
  - For the lion vs. gazelle population problem, submit the corresponding **`.java`** file.
  - For the even numbers problem, submit the corresponding **`.java`** file.
  - Do not submit files in any other formats – if you do, your assignment will not be graded.

Academic Honesty

You must be the sole original author of the solution you submit. You must compose all program and written material yourself. All material taken from outside sources must be appropriately cited.