**Title Page: cs 1011-051 – Lab 5 *Library Classes***

Cody Mott

10/11/2022

**I. Objectives**

This program is a game. It generates a random dollar amount between 0 & 5000, random number of weeks between 0 & 40, tells the player backstory which drives the game, tells the player how they can win, gives the player the option of making x amount for an uncertain number of weeks or doubling their money every week starting with $0.01. The player wins if, when the random number of weeks is reached, they chose the option that made more money. The program then tells them if they win or lose and how much the difference was between accounts.

**II. Requirements**

* Class must be named GrowthRate
* Must contain header
* Must use good style
* Must generate a random dollar amount between 0 - 5000
* Must generate a random number of weeks between 0 – 40
* Must user a creative backstory as a premise for the game
* Must give the player two options
* Must show weekly linear growth and weekly exponential growth
* Must tell the player whether they won or lost
* Must tell the player the difference in account balances
* Must allow the player to keep playing if they so wish

**III. Design:**

//GrowthRate Pseudocode//

START GrowthRate

generate random $ amount between 0-5000

generate random # of weeks between 0-40

display story

ask user if they want random dollar amount added to account every week

or if they want to double the amount in their account starting with $0.01

display weekly growth until random week is hit

compare the amounts and determine if user won

tell the user if they won or not

END GrowthRate

//main Pseudocode//

START main

declare global variables

display story

do while the player wants to keep playing

generate random number of weeks between 0-40

generate random dollar amount between 0-5000

display player options

display winconditions

get player's choice

calculate and display linear and exponential account growth

determine if player won

display message telling player if they won or lost

ask player if they want to play again

end do while

END main

//generateRandomNumber Pseudocode//

START generateRandomNumber

input parameter upper limit

return random number between 0-upper limit

END generateRandomNumber

//displayStory Pseudocode//

START displayStory

display story

END displayStory

//displayOptions Pseudocode//

START displayOptions

input parameter: randomly generated dollar amount

display option 1

display option 2

END displayOptions

//displayWinConditions Pseudocode//

START displayWinConditions

display win conditions

END displayWinConditions

//getPlayerChoice Pseudocode//

START getPlayerChoice

declare boolean error flag variable

do while error is detected

try

ask user to choose option 1 or 2

get user input

assign false to error flag

end try

catch exception

ask user to enter an integer

clear buffer

assign true to error flag

end catch

if player doesnt enter 1 or 2

assign true to error flag

end if

end do while

END getPlayerChoice

//accountGrowth Pseudocode//

START accountGrowth

input parameters: randomized number of weeks, random dollar amount

assign starting balances to linear and exponential growth accounts

if number of weeks is greater than 0

for weeks 1 - number of weeks

display linear account growth

display exponential account growth

add dollar amount to linear account balance

double exponential account balance

end for loop

end if

END accountGrowth

//didPlayerWin Pseudocode//

START didPlayerWin

calculate difference in ending balances

if player chose option 1 and ending linear account balance was greater than ending exponential balance

player won

end if

else if player chose 2 and ending exponential balance was greater

player won

end else if

else

player lost

end else

END didPlayerWin

//displayWinLose Pseudocode//

START displayWinLose

input parameter: week sun explodes

if week sun explodes is less than 1

display message telling user that everyone died

end if

else if player won

display congratulatory message

end else if

else

tell player that they lost

end else

END displayWinLose

//playGameAgain Pseudocode//

START playGameAgain

declare String variable again

declare boolean variable inputError

do while there is an input error

try

ask player if they would like to play again

get input

assign value of flase to inputError

end try

catch exception

tell user to input y or not

flag inputError as true

assign unused String to again variable

clear buffer

end catch

if user input is not a y or n

flag inputError as true

clear buffer

end if

end do while

if user wants to play again

assign true to playAgain

end if

else

assign false to playAgain

display friendly farewell message

end else

end playGameAgain

**III. Test Plan/Test Strategy**

Each method was rigorously tested with following parameters

* If there is a display message does it display correctly?
* if there is a display message based on certain parameters, is the correct message displayed?
* if there is user input, does the scanner get the right input?
* if there is an input error, is the input asked for again?
* are calculations correct?
* if a calculated value is displayed, is the displayed value correct?

When I test methods, I test each method individually. If there is a display message, I will run the method repeatedly tweaking the message until it looks right. If there is a input, I write some code that will display the user input to the console just to make sure that the input is getting gotten correctly. if there are any calculations based on input parameters, I will control in the input values and cross check the calculations by hand and by calculator. If any values are to be displayed to the console, I make sure that they are the correct values. If are any input variables that need to have contingency plans for user errors, I run every incorrect input I can think of, see what happens, then I write code that makes it do what I want it to (which is usually clearing the buffer and looping back to the original prompt asking for an input).

I do not particularly plan my tests before I write my code. I test as I write my code. I will run tests sometimes after writing a line and sometimes after writing a small block of code (i.e. if statements), and I always test test test test test and test after I have written everything. I test how well all the methods work together and I test all the user inputs for the program as a whole.

**IV. Results**

* **V. Discussion**
* What did you learn? What didn’t work? How did you fix it?