Catherine Wallin
CSC-17A-49285
12/14/2020

Project 2:
The Game of Life

Introduction

I chose the board game, “The Game of Life” for my project.

It is a multiplayer game (2-4) players. Players progress through the board and encounter various life events such as marriage, careers, taxes, children, and more. Once the players retire, all of their earnings and debts are totalled and the player with the most money wins the game.

The game is a very popular family game to have fun.

Project Size: Lines

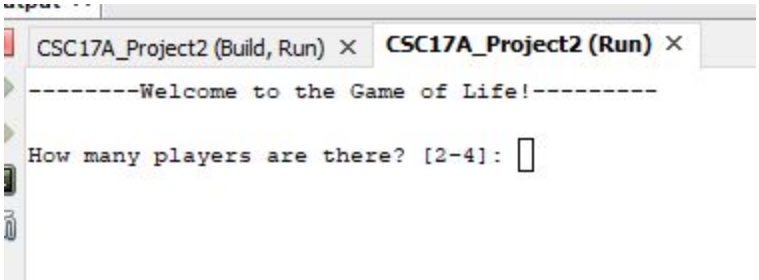
Description

I used classes to hold most of the game data and functions, such as player data, job data, and game statistics. The players “move” through an array. Depending on the element in the array, they are guided to a function that goes through the desired set of instructions. At the end, the numbers are added up and the player with the most money wins. This is a continuation of my first project.

Sample Input/Output

Inputs:

1. The number of players [2-4]
2. “college” or “career”
3. 1 or 2 to select job
4. ‘y’ or ‘n’ to trade salaries with a players
5. 1-4(depending on number of players) to trade salaries



```
--------------Career Selection---------------


Player 1
Pick career or college: college
```

```
-----------------------------------------------

Player 1
You Spun 2
You can trade salaries with any player.
Here are the salaries of every player:
Player 1 100000.00
Player 2 50000.00

Would you like to change salaries with another player? (y/n): u
Invalid Input!!! Please enter 'y' or 'n': y
Which player would you like to trade salaries with? (example: '1'): 1
```

Classes(UML)

Player
- career : JobBase - salary : double - married : bool - degree : bool - children : int - totalMoney : double - totalDebt : double - currentBoardPosition : int - lifeTileCount : int
+ Player() : + Player(&player : const Player) : + setCareer(job : *JobBase) : void + setSalary(randomNum : int) : void + setMarried(x : bool) : void + setDegree(x : bool) : void + setTotalMoney(tm : double) : void + setTotalDebt(td : double) : void + setCurrentBoardPosition(cbp : int) : void + setLifeTileCount(ltc : int) : void + getCareer() : JobBase* + getSalary() : double + getMarried() : bool + getDegree() : bool + getChildren() : int + getTotalMoney() : double + getTotalDebt () : double + getCurrentBoardPosition() : int + getLifeTileCount() : int + lifeTileSpace() : int + getMoneySpace(num : int) : double + payDay() : double + payPlayer(num : int) : void + paidByPlayer(num : int) : void + taxesDueSpace() : double + taxRefundSpace() : float + getRaiseSpace() : double + changeSalary(newSalary : double) : void + caughtSpeeding() : void + ticketPayment() : void + payDebt() : void

FinalPlayerStats (Derived from Player
- finalPlace : int - finalScore : float
+ FinalPlayerStats() : void + FinalPlayerStats(&player : const Player) : void + setFinalScore() : void + setFinalPlace(x : int) : void + getFinalScore() : float + getFinalPlace() : int

JobBase (Abstract Class)
- position : string - minSalary : double - maxSalary : double - taxes : float - degreeRequired : bool
+ JobBase() : void + JobBase(p : string, min : double, max : double, t : float) : void + getPosition() : string + getMinSalary() : double + getMaxSalary() : double + getTaxes() : float + getDegreeRequired() : bool + printRequirement() : void

Job (Derived from JobBase)
+ Job() : JobBase() : void + Job(p : string, min : double, max : double, t : float) : JobBase(p, min, max, t) : void + printRequirement() : void

CollegeJob (Derived from JobBase)
+ CollegeJob() : void + CollegeJob(p : string, min : double, max : double, t : float) : JobBase(p, min, max, t) : void + printRequirement() : void

GameStats
- maxBoardPosition : int - maxMoney : float - minMoney : float
+ GameStats() : void + getMaxBoardPosition() : int + getMaxMoney() : float + getMinMoney() : float + setMaxBoardPos(newMax : int) : void + setMaxMoney(newMaxMoney : double) : void + setMinMoney(newMin : double) : void + compare(x : T, y : T) : bool

Flowchart

(PDF in Write Up Folder)

Pseudocode

Ask user for number of players
Read from the keyboard the number of players (to use to loop)

Loop through players
    Pick college or career
    Generate two jobs based on career path
    Read from the keyboard the user selection
    Store job and its members inside the player structure

Loop through turns until a player has reached the end of the board
    Loop through players
        Random number generator
        Add random number generator to the player's current place
        Use the added numbers to "move" to that element of the array
        Execute the function that coordinates to that element

Add together the total money and debts of each player
Display each player's game stats and finishing place

Checklist(PDF in Write Up folder)

Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
13		Classes			
	1 to 3	Instance of a Class	49	4	
	4	Private Data Members	GameStats.h 17-21 FinalPlayerStats.h 17-18	4	Never Public
	5	Specification vs. Implementation	Player.h + Player.cpp GameStats.h + GameStats.cpp FinalPlayerStats.h + FinalPlayerStats.cpp	4	.h vs. .cpp files Always split
	6	Inline	Player.h: 44-52	4	
	7, 8, 10	Constructors	JobBase.h: 25, 33 main.cpp: 53, 54	4	I use a default constructor as a placeholder in my career array of objects
	9	Destructors	Job.h: 30	4	
	12	Arrays of Objects	84, 85, 86	4	
	16	UML	Write-Up	4	
14		More about Classes			

	1	Static	Spin.h	5	
	2	Friends	GameStats.h: 43	2	GameStats class is a friend of FinalPlayerStats class
	4	Copy Constructors	Player.h: 32 Main.cpp 129	5	copy starting player data to an object to use when printing results at the end of the game
	5	Operator Overloading	Player.cpp: 74, 108, 149	8	Overload 3 operators
	7	Aggregation	Player.h: 19	6	
15		Inheritance			
	1	Protected members	Player.h: 19-27	6	
	2 to 5	Base Class to Derived	FinalPlayerStats.h: 15	6	FinalPlayerStats derives from Player class
	6	Polymorphic associations	Job.h: 26 CollegeJob.h: 30	6	Virtual function in Job Bass to print a message depending on regular/college job
	7	Abstract Classes	JobBase.h	6	
16		Advanced Classes			
	1	Exceptions	main.cpp: 250-256 Player.cpp: 43-45	6	
	2 to 4	Templates	GameStats.h: 38-41 main.cpp: 373, 381	6	template function used to compare starting data with final player stats
	5	STL	main.cpp: 276	6	
		Sum		100	