Phillip Wellheuser
CS 162-400
4/28/19
Project 2: Zoo Tycoon

Planning:

I think for this project there will be a lot of data to keep track of but using classes and inheritance I should be able to split a large amount of it up into those. I know I'll need each of the classes required by the assignment, and I'll also include my inputValidation and menuShell files to cover some of the work of making menus and validating input, which will also lighten the load.

The animal class and its subclasses should be pretty simple. I may end up putting some methods in the animal class for retrieval of specific data for the different animals especially for the extra credit animal should I choose to do it.

I'm going to divide up as many of the specific tasks in the program into their own functions so that they may be run alone for testing purposes and I'll test them individually that way. Unless I need some new kind of data validation which I haven't used before this project, which I don't suspect I will, I should not really need to do any further testing on that as I've already done it for previous projects.

I think this project comes down to mostly doing a some simple tasks and displaying results and then linking all of those task-result pairs together so they run as a unit. I suppose that's programming in a nutshell… I've done some planning for my zoo and animal classes written by hand. I'll attach an image of that juxtaposed to what I end up with in the end below:

Design:

Requirements:

- Zoo class
    - Starting money bank
    - 3+ types of animals to buy
        - Only 1 or 2 at start
        - Purchases subtract from bank
        - Animals start at 1 day old
    - Day cycle
        - Pay to feed animals
            - Bonus 2 extry types of feed
            - Bonsy feed types modify chance of sickness for sickness random event
            - Subtracts from bank
        - One randomized event
            - Sick animal
                - Kill a random animal
            - Business boom
                - Bonus money for each tiger
            - Animal has a baby/babies
            - Nothing happens
        - Calculate profit
            - Based on number of each animal and their payoff modifiers
            - Add business boom
        - Option to purchase additional animals

                - Subtracts cost from bank per animal
                - Animals start at age 3
            - Increment the day
            - Ask the player if they want to keep playing
                - Print an end message
        -
- Animal, Tiger, Turtle, Penguin classes
    - Bonus animal optional
        - Has the attributes of the animal class and subclasses, but they are definable by the player
    - Have attributes: (no more than these)
        - Age
        - Cost to buy
        - Number of babies
        - Modifier for food cost
        - Modifier for daily payoff
- Input validation/menus
    - does not crash from the undesired input
    - request for input repeatedly until the correct data is inputted.

Testing:
For the animal classes all of their methods were inherited and mostly consisted of getters and setters. A simple test of setting, getting, and printing the results of all of these for the animal parent class is sufficient to me to prove that these classes are functioning as intended.

Test Cases	**Test**	**Expected Outcomes**	**Observed Outcomes**
void wantBonusAnimal();	Run the function alone within the Zoo class and print the results after each prompt to see if it was properly received.	Each user input should be properly assigned to its variable and accessible through the zoo class	As Expected
void zooStartup();	Run the function alone within the Zoo class and print the results after each prompt to see if it was properly received.	Prompts the player to buy 1 or 2 of each animal the zoo can have, including the bonus animal, and does not crash with invalid input	As Expected
void ageAnimals();	Add animals to each array with different age values, call age	All ages will increase by 1 day	As Expected

	Animals, print results to see that they are properly incremented		
void printStats();	call the function	If all of the intended print values, print, it's working	As Expected
void feedAnimals();	Add animals to each array and call the feedAnimals method with each type of feed chosen. Print out the bank status before and after to see that money is being subtracted correctly	The bank values will reflect precisely the amount of money which should have been removed by feeding the animals.	As Expected
void randomEvent();	Call random event and print out a different message for each event which should be called and examin their call frequency with and without the odds of sickness being increased from cheap feed	Each random event will occur with appropriate frequency to their odds and sickness will increase infrequency when the cheap feed variable is true	As expected
void sickAnimal();	Add animals to each array then call sickAnimal with a printout assigned to each different result	The printed results should occur in similar freqnency between the options	As expected
void businessBoom();	Call businessBoom with different numbers of tigers and print the amount made as well as the status of the addBusinessBoom bool which marks it to	The amount of profit should be evenly be divisible by the number of tigers with a quotient between 250 and 500 and the bool should be true	As expected

	be added to profits later		
void babyAnimal();	call babyAnimal with arrays full of animals less than 3 days old as well as with more and mixed then print out different results for each animal chosen to give birth	no babies will be born if the animals are less than 3 and each type of baby will be born with similar fequency.	There were some looping issues, but they were resolved, As expected
void nothingHappens();	call the function	text will print	As expected
void collectIncome();	call the function while running the zoo class using known numbers of tigers, penguins, turtles, and bonus animals, then make sure their profit matches what your math tells you. Make sure the profit per animal and their total profits add up properly	For example: 1 tiger, 1 penguin, 1 turtle. Tigers: $2000, Penguins $100, Turtles: $5, Total: 2105	As expected
void runAnimalStore();	call runAnimalStore through the Zoo class and check to see if animals purchased print out that they call the correct add methods and menus work both with and without bonus animals on	printouts and menus should reflect the animals purchased through the menus	As expected

void buyAnimals(int animalType, int howMany);	call buyAnimals alone in the zoo class and print out the values passed to the add methods and the bank value updates	animals chosen through menus will call the appropriate functions with the appropriate parameters and the bank will be affected appropriately to the number of animals bought	As expected
void removeTigers(int numberToRemove); void addTigers(int numberToAdd, bool addBabies);	Test using Valgrind through my zoo class, add several tigers using addTigers() with different attributes to initialize them, print out attributes from each to show that they've updated, and then remove them from the tiger. Make sure there are no leaks on Valgrind	There will be no leaks on Valgrind	At first there were some issues with deallocating space, but it was fixed.
void removePenguins(int numberToRemove);	Tested similarly to Tigers	Results similar to Tigers	As expected
void removeTurtles(int numberToRemove);	Tested similarly to Tigers	Results similar to Tigers	As expected
void removeBonusAnimals(int numberToRemove);	Tested similarly to Tigers	Results similar to Tigers	As expected
void addTigers(int numberToAdd, bool addBabies);	Tested similarly to Tigers	Results similar to Tigers	As expected
void addPenguins(int numberToAdd, bool addBabies);	Tested similarly to Tigers	Results similar to Tigers	As expected

void addTurtles(int numberToAdd, bool addBabies);	Tested similarly to Tigers	Results similar to Tigers	As expected
void addBonusAnimals(int numberToAdd, bool addBabies);	Tested similarly to Tigers	Results similar to Tigers	As expected

Reflection:

Again, as with the dice project, I took way longer than I felt I should have to complete it. However this time my time sinks were more related to my technical understandings of inheritance and debugging than poorly allocated time. My understanding of what the project was asking for was mostly correct I think but my understanding of how to implement it was not. I think I could have utilized inheritance better, by nesting some of the functions like aging or updating all of the bonus animal attributes, which are part of my zoo class, into my animal and animal subclasses, but here is only so much time to refactor code so I decided it best to continue on rather than go back and fiddle. I think that would make my code more organized and clearer.

I also had a misunderstanding of how inheritance is implemented and struggled with that for a while before finding a solution. I ended up getting my subclasses to work using forward declaration, but I didn't remember reading anything about that in the textbook, so I'm worried it's not the best way to do things. I hope that in grading this, you might be able to inform me better on the matter. Going forward, I'll definitely remember what I've learned from the issue and I hope I have time to review inheritance some more to get a better handle on it.

Another issue I ran into was that I was failing to deallocate memory properly when I added new animals to my arrays. I wasn't deleting the empty animal in the array when I assigned the new one to the pointer, so I ran into some leaks. They were very difficult to find because Valgrind was running into issues with the way I had my strings defined at the start of my zoo class. Previously I had defined all of my strings at the start of the program, rather than just declaring their variables. For some reason, Valgrind was getting hung up on the strings and was, as a result, not very helpful in finding the error. Once I moved all of my string definitions into a separate function to be run upon construction of the object, it was much easier to find the issue as it was just a matter of reading my code carefully and walking through it.

In general, I feel like I'm either using too many variables in my programs or I'm organizing poorly, so I want to look into that as well. It's organized chaos as it is. A whole bunch of variables together, but with so many it's hard to figure out what's what without a lot of looking through it. I want my code to be easier to read than that. I

I think I could have done a better job with my planning. It's hard to know just what will be needed from each method even when you know what they're supposed to be doing and it is pretty straight forward. I think some skill in this will come with practice. It's also difficult to realize when you're thinking too complexly and grouping what is actually several statements into one concept while planning, which can lead to poor planning, time management, and understanding of your program.

I got it working, and I think it does everything it's supposed to, if not in the most effective

manner, but my understanding of programming is coming to be that it is an ongoing battle for improvement in technique and understanding, so I'm satisfied with my program and am ready to move on.