Kimberly Tom

CS 162

Project 2

Design

1. Main
    a. Main is where we initialize the zoo tycoon game
        i. Key functions and data members
            1. show zoo tycoon main menu
2. Zoo
    a. has the main menu function of the game
    b. has the following functions:
        i. purchasing animals, tiger, penguin, or turtle, and subtracting cost from bank
        ii. call to function to increase animal age
        iii. show stats of zoo, including number of adult and baby animals
        iv. calculate cost of food and subtract from bank
        v. random event function for sick animal, animal birth, and tiger bonus
            1. random generator to pick which event
        vi. remove sick animal function
            1. random generator to pick which animal
        vii. add animal babies function
            1. random generator to pick which animal
        viii. generate bonus income for tigers

3. Animal
    a. holds the age of animal, cost of animal, number of babies of animal, base food cost of animal, and revenue of animal
    b. function to increase animals' age
4. Tiger
    a. inherits protected members and functions from Animal class
    b. one constructor takes age as a parameter
    c. default constructor sets tiger's attributes
5. Penguin
    a. inherits protected members and functions from Animal class
    b. one constructor takes age as a parameter
    c. default constructor sets penguin's attributes
6. Turtle
    a. inherits protected members and functions from Animal class
    b. one constructor takes age as a parameter
    c. default constructor sets turtle's attributes

7. Input Validation
    a. check user's animal purchase choice and tell user to choose again if invalid choice is selected

TEST CASE	INPUT VALUES	EXPECTED OUTCOMES	OBSERVED OUTCOMES
Start menu, start	1	starts game	starts game
start menu, quit	2	quits game	quits game
start menu letter/number combinations	1a -2g b	invalid, prompt user for selection	invalid, prompt user for selection
start menu out of bounds	0, -5, 3	invalid, prompt user for selection	invalid, prompt user for selection
animal purchase on day 1, 1-2 animals for tigers, penguins, and turtles	1, 2	1 = purchase 1 animal of type 2 = purchase 2 animal of type shows correct costs deducted from bank account	1 = purchase 1 animal of type 2 = purchase 2 animal of type shows correct costs deducted from bank account
animal purchase on day 1, out of bounds	0, 30 10, -5	invalid, prompt user for selection	invalid, prompt user for selection
animal purchase, letter/number combination	k 2b h1	invalid, prompt user for selection	invalid, prompt user for selection
next day menu, continue to next day	1	continues to next day, and starts next day's functions, including correct food cost function, random event function with correct results (remove animal for sickness, tiger bonus, add animal for birth), correct age++ of animal	continues to next day, and starts next day's functions, including correct food cost function, random event function with correct results (remove animal for sickness, tiger bonus, add animal for birth), correct age++ of animal
next day menu, quit	2	shows zoo animal counts, bank balance, and how many days user completed	shows zoo animal counts, bank balance, and how many days user completed
next day menu, letter/number combination, out of bounds	1n, b2 0, 3, -5, 10	invalid, prompt user for selection	invalid, prompt user for selection

animal purchase after day 1, out of bounds	-5, 0, 5, 20	invalid, prompt user for selection	invalid, prompt user for selection
animal purchase after day 1, valid	1, 2, 3, 4	1 = purchase tiger and add tiger to zoo (age 3) 2 = purchase penguin and add penguin to zoo (age 3) 3 = purchase turtle and add turtle to zoo (age 3) 4 = no purchase of animals, no change to animal count	1 = purchase tiger and add tiger to zoo (age 3) 2 = purchase penguin and add penguin to zoo (age 3) 3 = purchase turtle and add turtle to zoo (age 3) 4 = no purchase of animals, no change to animal count
check for memory leaks	valgrind /.project2	no memory leaks	no memory leaks

Reflection

One major design change I made was regarding input validation. Originally, I wanted to put most of my input validation code in the inputValidation.cpp. However, I realized it would be easier in this case to leave it in the Zoo class. There was a ton of input validation in this program compared to the Langton Ant project, and since the code is short for it, it was easier to see what was going on when I kept it in Zoo without having to worry about making a mistake when returning the value back from the inputValidation.cpp.

Another change I made was where I initialized my variables. Originally, I was setting them to 0 in the zoo.hpp file. However, I figured this would be better in the default constructor as well as in functions themselves so not to get their values accidentally altered if only that one particular function is utilizing it.

An issue I had early on was my code wasn't working the second time user tries to buy animals. It was exiting. I finally realized it was because I declared a new array of same name twice which was causing it to crash.

One issue I had problems with was with resizing the array. Originally, I just wanted to try to get my code working and thought I could work on the resizing of array later. However, my program kept crashing when I was testing it. It took me a while to figure out how to resize it, with use of a temporary array. I realized that I can keep track of the current size and double it with use of a variable called newSize, so if the animal count from a purchase or birth goes over the initial size of 10, I can double the size to 20 and have that doubled size stored in tigerSize, for example, then use the newly doubled tigerSize to create the new array. Another problem I ran into was everytime baby animals were born, it would reset my age for my other animals of same type to age 0. I was not copying over the values correctly to my turtle array, for example. I had to make sure that I copied the values to each animal type's entry. Before, I did not do this and was just trying to send the values to the whole array.

Another issue that took me a while to figure out was how to randomly choose an animal to die or be born. I could easily use a random function to pick an animal to die or be born, but what if no animals of that type exist, or there were no adult types of that randomly chosen animal. I realized I could use a do while loop and put another random code inside of the do portion of the loop. I also used this to check that if there are no existing animals of any type to die, or could give birth, I could take that into account in my do while loop as well, and have it return an integer that would allow the program to not implement a death or birth. Otherwise, it would randomly choose another animal until there is an existing animal type that could work with the random event.

I learned a lot from this project. It helped me get a lot more comfortable with using parent classes and sub classes. It also helped me understand arrays better. I was initially confused about how the Animal array and how it points to the animal type (tiger, turtle, penguin). I'm also now getting a lot used to deallocating memory with arrays. It also made me realize the importance of dynamically creating an array. My program was crashing when I was testing it before I made the size big enough to hold all the animals of each type. Another thing I learned was how to set the attributes of only a few

entries in an array and not the entire array, such as when an animal type has babies. Overall, I think I learned a lot about arrays and feel more comfortable using parent classes/child classes.