Camille Marcojos

CS 162 – Fall 2019

Project 2: Zoo Tycoon

**Initial Design**

* Zoo has animals
* Tiger/Penguin/Turtle is an animal
* Zoo class controls the game

Zoo Class

* Member variable:
  + Funds
  + Animal object
* Member functions
  + Start game
  + Set funds
  + Get funds
  + Addfunds
  + subtractfunds
  + Add animal
  + Remove animal
  + Random event
  + BuyAnimal
  + Animalborn
  + Sickanimal
  + Feedanimal

Game flow:

* Start game
* Set budget
* User buys animal
  + User has to specify quantity of animals (either 1 or 2)
    - Add animals to zoo
      * Age = 1 day
  + Subtract from funds
* Start day
  + Increase age
  + Feed animals
  + Random Event
  + Calculate profit
  + Buy adult animal
  + Replay
  + Start over or quit

Base class: Animal class

* Derived class: Tiger
  + Get/Set payoff
  + Get/set base food cost
* Derived class: Penguin
  + Get/Set payoff
  + Get/set base food cost
* Derived class: Turtle
  + Get/Set payoff
  + Get/set base food cost

Animal Class

* Member variables:
  + Age
  + Cost
  + Number of babies
  + Base food cost
  + Payoff
* Member functions
  + constructor
  + Get/set age
  + Get/set cost
  + Get/set # of babies
  + Get/set base food cost
  + Get/set payoff
  + destructor

Reflection

This was a very fun, but challenging project and the end product differed significantly from my original design plan. The initial design plan helped me out in creating a basic game flow and deciding on what classes to make and how many, however, it wasn’t until I actually started working on the program that I came to realize how much my design was missing such as the many different types of functions.

When I was working on this project I was unsure how I could track of the animals in my zoo. I had initially wrote code that dynamically allocated three different animal arrays and then I would add the specific animal objects into this array. It worked fine until I ran into the problem of how to remove an animal object when an animal unexpectedly dies. I had initially thought to decrement the count, but that would still leave me with memory leaks because I didn’t deallocate that memory. After reading some insightful posts on piazza and advice from the instructor, I changed my approach to dynamically allocating an array of pointers to pointers of animal objects. This helped with removing an animal because I could just deallocate the memory of the array index that’s holding the animal and then have it point to null.

Another realization I came across how much I duplicated code so that they can be specific to the animal. I am sure there is another way to make it more succinct and efficient, but due to time constraints and beginner’s knowledge, I did not have the chance to work it out. After completing this assignment, I’ve realized how much more I need to learn, but I am very excited for what’s to come.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Input Values** | **Driver Functions** | **Expected Outcomes** | **Observed Outcomes** |
| Input too low | Input < 0 | int getChoice()  int threeChoices()  int numAnimals()  …all do-while loops | Not a valid choice, re-prompt user for a valid input | Not a valid choice, re-prompt user |
| Input = 0 | Input = 0 | int getChoice()  int threeChoices()  int numAnimals()  …all do-while loops | Not a valid choice, re-prompt user for a valid input | Not a valid choice, re-prompt user for a valid input |
| Input in correct range | 0 || 1 for start menu/replay/buy adult animal; 1-3 for adult animal | int getChoice()  int threeChoices()  iint numAnimals()  …all do-while loops | Prompt user for the next input | Prompt user for the next input |
| Input in extreme high | Input > 1000; for threeChoice=”invalid command.” | int getChoice()  int threeChoices()  int numAnimals()  …all do-while loops | Not a valid choice, re-prompt user for a valid input bt 1 and limit | Not a valid choice, re-prompt user for a valid input bt 1 and limit |
| Input is a character | A-Z or a-z | int getChoice()  int getBoardInput()  int getAntStart(int x)  int getSteps()  …all do-while loops | Not a valid choice, re-prompt user for a valid input | Not a valid choice, re-prompt user for a valid input |