Tucker Lavell
CS162 Fall 2017
Project 2: Zoo Tycoon
Documentation

Problem

Create a Zoo Tycoon Game. When the game starts it should ask the Zoo owner which animals they want to buy and how many. Once they choose the animals the first day will start, technically all animals bought at the beginning will be 2 days old on the first turn. At the beginning of a turn, the animals will age 1 day, then a random event will happen. Those random events will be a sickness occurs to an animal and it will die, the zoo will have a boom in attendance and will make extra money that day, a baby is born (to an animal that is more than 3 days old), or nothing will happen. At the end of the day it will ask the Zoo owner if they want to keep playing or end the game. There will be a running total of income and expenses kept to monitor money to determine game overs. To further the user experience and prevent as many errors as possible, the whole program should have a robust input validation.

Design

Animal

- Int Age
- Double Cost
- Int Number of Babies
- Double Base Food Cost
- Double Payoff
- Getters and setters for all

Zoo

- Animals[10]
- Double Bank
- Double income
- Int numTigers
- Int numPenguins
- Int numTurtles
- Zoo(), ~Zoo()
- beginingOfDay()
- endOfDay()
- dailyEvent()

Game

Start:

Begin with $100k

Ask how many of each animal to buy.

Subtract from bank

Animals are 1 day old

Turn:

Each animal ages 1 day.

Calc feeding cost for each animal

Subtract cost from bank

Random event

Determine if any animals are old enough to have children

rand() 1-4(3) (maybe add probabilities so an animal isn't likely to die 25%)

Calc profit for day (turn) + bonus.

Ask if they want to buy an adult animal at end of day.

If yes, animal starts at 3 days old.

If out of money, print game over and endgame else, ask keep playing or end game?

If play, loop to next turn, else print ending Bank (maybe show profit or loss)

Test Table

Test Case	Input Values	Expected Outcomes	Observed Outcomes
Start zoo, with user inputs Bank = 100k	Tigers = 2; Penguins = 1; Turtles = 2;	Zoo starts with 5 animals, Bank = 78,800	Tigers = 2; Penguins = 1; Turtles = 2 Bank = 78,800
feedCost	Tigers = 2; Penguins = 2; Turtles = 2;	Tigers cost = 100; Penguins cost = 20; Turtles cost = 10; feedCost = 130	Tigers cost = 20; Penguins cost = 20; Turtles cost = 20; feedCost = 60;
The only tiger gets sick	Tigers = 1;	Tigers = 0;	Division by 0 error
An adult turtle has a baby	Turtles = 1; Age = 3;	Turtles = 2; Age 1 = 3; Age 2 = 0;	Turtles = 2; Age 1 = 3; Age 2 = 1;
A baby turtle has a baby	Turtles = 1; Age = 1;	Turtles = 1; Age 1 = 1;	Turtles = 2; Age 1 = 1; Age 2 = 1;
Buy an adult tiger at end of day	Tigers = 1; Age 1 = 1;	Tigers = 2; Age 1 = 1; Age 2 = 3;	Tigers = 2; Age 1 = 1; Age 2 = 1;
Run out of money	Bank = $0.00;	End game	Game ended

Reflection

This project went much better for me than the first one. Though it has taken me about a full week to work on, I was much more prepared, and I also gave myself enough time. My biggest takeaway from this project is trying to learn to write neater code for larger assignments. This and ant both felt like I let the code get really messy, which made it hard to debug, and stressful to edit. To me this project seemed more straightforward than the ant, but it still went under some heavy design changes. Writing it was going super well until I got to the random event. For me this was the hardest thing to implement, of those the hardest for me was having a baby.

I feel Like I learned a lot about inheritance from this project. My first issue with the program was learning about inheritance. At first feed cost was not calculating properly, it took me forever to figure out why. I rewatched the lectures twice, and after a little extra googling I figured out it was because I was not using virtual functions correctly. I had them all declared as virtual, but I was never actually setting a feed cost for each animal. I ended up having the same problem with age as well, but it was much easier to fix when I already understood the problem.

My first problem with the daily event came with the original formula I had for a sick animal dying. The way I had it set up, I would get a division by zero error if there was only 1 of that type of animal. If it hadn't been for the test plan, I may have accidently overlooked that requirement. The next problem with the daily event was learning the proper functionality of rand() since I did "25%" chance of each event happening. I would test with rand() selecting in the range I wanted, but on baby (50-75) I was getting numbers greater than 75, so I hardcoded it for testing. Designing baby also ran me into this problem. Because of this, the program was not choosing who was supposed to have the baby properly, so everyone was having babies, even babies. It was extremely confusing at first, but after a little googling I found a formula that seems to works well for any range you throw at it.

After the random event I ran into a small problem with buying a new animal, it took me forever to figure out what was wrong, it was an off by 1 error with my arrays. The next issue I ran into was resizing my arrays at max capacity. I couldn't get it to work and I ran out of time unfortunately. The messiness of the code contributed to this.