Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

Animal Class (Base)

Member Variables
● Age ● Cost - cost to purchase one animal ● Babies - litter size for each animal ● FoodCost - holds the specific cost of food for each animal ● Payoff - holds the specific payout for each animal
Member Functions
● Constructor( ) ○ Set age to -1 so animals added to the zoo can be distinguished from those that have just initialized the array ○ Set the base food cost in Animal class so derived classes can simply multiply their coefficient by the value main) ● isAdult() ○ Will take no parameters, should return a boolean variable ○ If the animal is greater than or equal to three days old, return true ● getPayoff() ○ Should return the payoff for the specific animal ● getBabies() ○ Should return the babies ● getCost() ○ To return the cost of each animal ● setAge() ○ Takes an integer as a parameter, sets the age member variable and does not return a value. ● incrementAge() ○ Increases age by 1

Tiger Class - derived from animal class

Member Variables
● No extra member variables necessary
Member Functions
● Constructor( ) ○ Initialize all member variables with constants ○ Babies should be set to 1 ○ Cost should be set to 10,000 ○ Food cost should be set to 5 times the base cost ○ Payoff should be set to 0.2 times the cost

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

Penguin Class - derived from animal class

Member Variables
● No extra member variables necessary
Member Functions
● Constructor( ) ○ Initialize all member variables with constants ○ Babies should be set to 5 ○ Cost should be set to 1000 ○ Food cost should be set to 1 times the base cost ○ Payoff should be set to 0.1 times the cost

Turtle Class

Member Variables
● No extra member variables necessary
Member Functions
● Constructor( ) ○ Initialize all member variables with constants ○ Babies should be set to 10 ○ Cost should be set to 100 ○ Food cost should be set to 0.5 times the base cost ○ Payoff should be set to 0.05 times the cost

Zoo Class

Member Variables
● Pointer to an array for each species of animal ○ Tiger array, penguin array, turtle array ● Variable to count the number of each species ○ tigerCount, penguinCount, turtleCount ● Variable to count the max capacity in each exhibit ○ tigerCap, penguinCap, turtleCap ● Variable to hold the state of the game ○ Bankrupt, finished or otherwise ● Variables to hold daily profit and expenses for each animal ○ tigerProfits, penguinProfits, turtleProfits ○ tigerExpenses, penguinExpenses, turtleExpenses ● Pointers representing each bit of information for the current species selected

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

- currSpp - points to current species array
- sppCount - for animal count
- sppCap - for exhibit capacity
- sppExpenses and sppProfits

- Variable to hold the bonus applied when random event is triggered
- dailyProfit, dailyExpenses and balance for calculating zoo finances
- Day - to keep track of how many days have passed
- Arrays holding species types
  - String array to hold species names, both uppercase and lowercase
  - Array holding enumerated data type for each species

Member Functions

- Constructor( )
  - Should take the starting cash for the zoo as a parameter
  - Allocate memory for a starting exhibit size of 10 for each animal
  - Initialize the species profits and expenses variables to 0
  - Initialize the daily profit and expenses variables to 0
  - Initialize all animal counts to 0 and the animal capacities to 10
  - Set the start balance equal to the value passed in as a parameter
  - Set the starting day as 1

- Destructor
  - Should free the dynamically allocated memory for each species

- setCurrSpp
  - Takes a Species type as a parameter
  - If the type is tiger
    - Set pointer currSpp equal to tigers array
    - Set sppCount equal to the address of tigerCount
    - Set sppCap equal to the address of tigerCap
    - Set sppProfits equal to address of tigerProfits
    - Set sppExpenses equal to address of tigerExpenses
  - Else set the pointers to penguin variables or turtle variables

- expandExhibit()
  - Takes a Species type as a parameter and use it to call setCurrSpp( ) function
  - Create a temporary pointer for allocating a new array
  - Use if/else statements to determine which class should be instantiated in the array pointed to by temp
    - Ex. If the parameter is a Tiger allocate a new array of Tigers that is twice the size of the current capacity
  - Age is the only member variable that will differ for each object, so set the temporary array with the correct ages with a for loop where i = 0 , and i must be less than the current species cap
  - Delete the memory pointed to by the currSpp pointer and set it equal to the temp pointer
  - Multiply the value pointed to by the sppCap pointer by 2
  - Use if/else statements to set the specific species pointers to exhibits to the same block of memory that currSpp points to (new array)

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

- addAnimal()
  - Takes a Species type, int for age and int for quantity as parameters
  - Set the current species depending on the the type passed in
  - Use a for loop to repeat for the correct quantity of animals added
    - Increment the species count by 1
    - If the species count is greater than the capacity of the exhibit, call the expandExhibit function
    - Use the set age function to initialize age to the value passed in

- buyAnimal()
  - Takes a Species type, int for age and int for quantity as parameters
  - Set the current species depending on the the type passed in
  - Calculate the total cost by multiplying cost by quantity
  - Compare the total cost to the zoo balance
    - If the total cost is greater than the balance, set the gameState to bankrupt
    - Else, call the addAnimal function and deduct the total cost from the balance

- getBalance()
  - Takes no parameters and returns the current balance

- getGameState()
  - Takes no parameters and returns the current state of the game

- adultInExhibit()
  - Takes a Species type as a parameter and returns a boolean variable
  - Call setCurrSpp() to set the current species depending on type passed in
  - Use a for loop to iterate through the exhibit
    - Use the Animal class isAdult() function to check
    - When an adult is found the function returns true
    - Else it will return false

- babyBorn()
  - Takes no parameters and will return a boolean variable
  - Create a boolean flag that will signal if its possible for any baby to be born and set it to false
  - Create a vector for storing the species that have adults in their exhibit capable of giving birth
  - Use a for loop to iterate through all species exhibits
    - If there is an adult in the exhibit, add the type to the vector and set babyPossible flag equal to true
  - If at least one of the exhibits has an adult, then calculate a random number between 0 and the max size of the vector - 1
    - Use the number generated to get the species at that index in the vector and add an animal of that type to the zoo
  - Else, if no babies are possible in any exhibit return false

- sickness()
  - Takes no parameters and will return a boolean variable
  - Create a variable to hold the number of exhibits that currently do not house any animals
  - Iterate through all species exhibits and increment the variable if there are no animals in the exhibit

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

- If the number of exhibits without animals is equal to the total number of exhibits then return false
- Else continue to do the following if the species chosen has no animals in the exhibit
  - Generate a random number between 0 and the total exhibits - 1
  - Set current species using the value generated to index into the list of species
  - Return true that a sickness has occurred

- attendanceBoost()
  - Takes no parameters and will return a boolean variable
  - If there are no tigers in the zoo, return false
  - Else, generate a random number between 250 and 500 to act as the bonus
    - Multiply the base bonus by the number of tigers

- pickFeed()
  - Takes no parameters and returns no value
  - Will set the feedTypeCoef which will be used to calculate feeding costs and the probability of sickness occurring
  - If cheap feed is chosen
    - Set the feedTypeCoef to 0.5
  - If generic is chose
    - Set the feedTypeCoef to 1
  - If premium is chose
    - Set the feedTypeCoef to 2

- randomEvent()
  - Takes no parameters and returns no value
  - Calculate the probability of a sickness occurring by dividing base value of 50 (50/200 = 25% original probability) by feedTypeCoef.
    - Will give value of 100 for cheap feed (50% probability), 50 or base for generic (25% probability) and a value of 25 for premium feed (12.5% probability)
  - Generate a value between 1 and 200
  - Compare the calculated prob value against the random number, if it is greater than the value calculated:
    - Generate a random number between 1 and 3 for the three remaining possible events (babyBorn, attendanceBoost, noEvent
  - Else
    - Trigger the sickness event
  - If any of the events return a bool value of false, then trigger the noEvent() function

- advanceDay()
  - Takes no parameters and returns no value
  - Nested loop to iterate first through the types of animals and then each instance of the species
    - Increment the age of each animal
  - Increment the day
  - Use a for loop to iterate through each species and reset the profit and expense information
  - Reset the daily profits, expenses and the potential bonus to 0

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

- calcExpenses()
  - Takes no parameters and returns no value
  - Iterate through the list of species using a for loop
    - Calculate species specific expenses by multiplying base food cost for that species by the feedTypeCoef and the species count
    - Add the species expenses to the daily expenses
  - Check that the daily expenses does not exceed the balance. If so:
    - Change the gameState to bankrupt
  - Else, deduct the dailyExpenses from the balance

- calcProfits()
  - Takes no parameters and returns no value
  - Iterate through the list of species using a for loop
    - Calculate species specific expenses by multiplying base food cost for that species by the feedTypeCoef and the species count
    - Add the species expenses to the daily expenses
  - Check that the daily expenses does not exceed the balance. If so:
    - Change the gameState to bankrupt
  - Else, deduct the dailyExpenses from the balance

- animalPrices()
  - Displays the price list for all of the animals

- dailySummary()
  - Takes no parameters and returns no value
  - Displays all of the financials for the current day
  - Day number, animal count, specific animal profit and expenses, daily expenses and profits , bonus if applicable and total balance

Main

- Welcome message and instructions
- Choose difficulty
  - Allow user to choose starting balance in bank (3 options)
  - Set the startBalance variable to the value chosen
- Instantiate Zoo passing in the starting balance in the bank
- Instruction set 2:
  - To start the game you must choose one or two animals of each type. Please enter the quantity of each below. You start with three animal species. Would you like to add your own?
  - If yes, prompt for: Species, Cost, Feeding needs, babies
- Display the price list for the animals
  - Display balance
  - Use a for loop to prompt for each type of animal
  - Have the user buy the quantity specified
  - Display balance afterwards

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

- While the gamestate does not equal finished or bankrupt
    - advanceDay()
    - Display starting the current day
    - Display animals being fed. Total cost and current balance
    - If gamestate =! bankrupt
        - Display random event that happened
        - Calculate the profit
        - Display day end summary
        - Prompt to add a new animal
        - If yes, choose from list and add the animal to the zoo
            - If gameState!=bankrupt
                - Display option to quit or continue playing
                - If quit, set game state ==finished

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

TEST TABLE

Test	Input	Expected Outcome	Actual Outcome
Base class and derived class functions	Use driver to test animal classes separate from zoo class. Create an object from each class and call all getter functions on it. Set the age using the setter function	Correct information regarding each animal should be displayed to the screen.	As expected,
Exhibit memory allocated correctly and expands when necessary Use driver function to test zoo separate from main.	Use a printExhibit function with in the Zoo class to print each exhibit and display the age of the animals within. Use the buyAnimal() function to test adding beyond the exhibit capacity and observe the output.	Expect each exhibit to display an age of '-1' indicating that no animals are currently in the exhibit. Count should be set to 0 and capacity to 10. As animals are purchased, count should increase accordingly. Exhibit should expand when animals exceed exhibit size. Capacity displayed should be double the previous size. When exhibit is expanded, all derived class attributes should still display correctly	As expected
setCurrSpp function Use driver function to test zoo separate from main.	Use print statements to check that the correct species is being set Print statements on species type, count, cap, expenses, profits. Test each using print statements individually. - Ex. Set tiger count, cap, expenses and profits to 1, 2, 3 and 4 respectively. Have other species values set to 0 while testing.	When species is set, all values should print as assigned. No value of 0 should be returned.	As expected

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

	- Repeat for each species		
addAnimal() Use driver function to test zoo separate from main.	Use print statements to check that species count is incrementing properly and expand exhibit is called as needed Test adding each species while species count is at 10 Test adding another animal after the exhibit has been expanded	Expect the capacity count to double when the species count is at 10 and a new animal is added. After exhibit is expanded, expect the capacity to remain the same and count to increase.	As expected.
buyAnimal() Use driver function to test zoo separate from main.	Test buying one and two of each species to ensure that the correct cost is deducted from the balance Create an instance of the zoo with a low enough start balance to bankrupt the zoo. ● Ex. Zoo balance of 15000 - attempt to buy 2 tigers ● Zoo balance of 9000 attempt to buy 1 tiger	Purchasing 1 tiger should deduct $10,000, purchasing 2 should deduct $20,000 Purchasing 1 penguin should deduct $1000, purchasing 2 should deduct $2000 Purchasing 1 turtle should deduct $100, purchasing 2 should deduct $200 Expect that the gameState should change to bankrupt if there is not enough money in the bank and the transaction will not complete	Function allows you to purchase 2 tigers and exceed the balance - Fix: make sure to multiply the cost by the quantity of animals before comparing to the balance
adultInExhibit() Use driver function to test zoo separate from main.	Test an instance of an exhibit having 1 adult vs. no adults Repeat for all species	Expect that the driver will print the return value of the function as true if there is an adult present and return false if there is not.	As expected
babyBorn() Use driver function to test zoo separate from main	Set all exhibits to have 0 adults Set only one exhibit to have an adult Set all exhibits to have adults ● Test age set at 3 and at 5 - both ages	Expect that for 0 adults in exhibit no animals will give birth If only one exhibit has adults only that specific species should be able to give birth and should do so everytime the function is called	As expected

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

	should produce babies Print each exhibit to the screen Call the actual function for each species to ensure the correct number of babies are added and the age is correctly initialized	If all exhibits have adults, expect that they will have babies with similar frequency Expect tigers to give birth to 1 baby, penguins to give birth to 5 babies, turtles to give birth to 10 babies	
sickness() Use driver function to test zoo separate from main	Set up a zoo that has no animals in any exhibit ● Use a print statement to signify which path of the if else statement was taken Set up a zoo with animals in 2 exhibits but not a third Set up a zoo with all exhibits containing animals ● Have each animal in the exhibit be assigned different ages ● Use print statements to follow which element was chosen	If there are no animals in any exhibit expect a prompt stating so With animals in some exhibits and not others expect that only exhibits with animals will experience a sickness Expect that elements of the array are properly shifting after an animal has been selected to be sick	When only tigers are in the zoo a statement is printed to show multiple animals have died when there are no animals in the exhibit Fix: code is set to keep generating a random number until an exhibit that has at least 1 animal is chosen therefore the statement that an animal died has to be outside of that loop
attendanceBoost() Use driver function to test zoo separate from main	Set up 3 zoos with varying levels of tigers - test the attendanceBoost function while other exhibits have animals present Set up a zoo with no tigers present.	Function should print the base bonus and display the correct total value for the quantity of tigers specified No bonus should be issued	As expected
pickFeed()	Possible inputs: 1, 3, 0, 4, 2, 10000, -1, hi, p, 8 i, 89.4, ji89	Should reject any values outside of the range of 1 - 3 Should reject any non-integer values Should set the feedTypecoef to	As expected

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

		the appropriate value	
randomEvent() Use driver function to test zoo separate from main - use in conjunction with pickFeed()	Test all of the events with 0 animals in exhibit. Simply use a print statement to determine which event was chosen before the noEvent() if statement is executed Choose each type of feed at least 10 times in a row and see which event chain was triggered Use a print statement to display each of the random numbers that was generated	For cheap feed, expect a probability of sickness occurring ~50% of the time For generic feed, expect equal probability of all events For premium feed, expect sickness to occur ~12.5% of the time.	Cheap feed was actually making sickness less likely Fix: had accidentally multiplied by the feedtypeCoef instead of dividing
advanceDay() Use driver function to test zoo separate from main	Set up a zoo with animals in all exhibits ● use the printExhibit function in the Zoo class to display ages for each animal before and after the advanceDay() function is called Print the current day	Expect all of the ages up the number of animals in the exhibit to increment. Elements in the array without animals should remain as -1.	As expected.
calcExpenses()	Use the main program to test ● Test several iterations and view the finances of the zoo by using the dailySummary function	The correct cost of feed should be deducted from the balance based on quantity and species If the cost to feed the animals is greater than the remaining balance, the game should not proceed	Balance is not accurately reflected in the daily summary Fix: the bonus variable was not originally initialized in the Zoo constructor producing a garbage value when added to the balance
playMenu()	Choose to play and quit Possible inputs: 1, 3, 0, 4, 2, 10000, -1, hi, p, 8 i, 89.4, ji89	Play should launch the game and quit should immediately exit Should reject non-integer	Program is not immediately quitting after choosing exit Fix: provide a

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

		inputs and values outside of the range 1-2	return statement within the if statement to exit the program while still keeping zoo setup functions outside of the while loop
Zoo setup in main()	Test purchasing 1 and 2 animals of each Test purchasing 2 tigers on the hardest difficulty	Should accurately update the balance Buying 2 tigers should bankrupt the player before they can purchase any more animals	Program still allows user to enter other animals after going bankrupt Fix: add the condition that i must be less than the number of species and also the game must not be set to finished
Choosing animal at the end of the day	Possible inputs: y, n,Y, N, a, x, ABC, 2, 4, 1.0, hi there	Should reject any characters that are not y and n and will reject any non-char inputs Choosing yes should display the list of animal options Choosing no should prompt the player to quit or keep playing Should quit if the player goes bankrupt	As expected
General integer input validation	**Possible Inputs:** 1, 2, 0, 3, 100, -1, hhhh, 123], 123 j, jj224 Test the max and min values and edge cases	non integer values or values outside the range will reprompt	As expected

Project 2 - Zoo Tycoon: Design, Test Table & Reflection

Nicole Reynoldson
7/20/19

Reflection

One of the biggest challenges I had with this program was deciding how to organize my classes and deciding which functions should be controlled where. Looking back on the finished program, I definitely feel that I could have further broken down the zoo class in a way that would make it more organized, but I just didn't have the time to polish the program as much as I would have liked.

When I first tried writing the program I realized that I was repeating a lot of the same code, just changing the keyword from tigers, penguins, turtles etc. To improve the reusability of the code, I created the setCurrSpecies() function which is probably the most powerful and useful bit of code in the whole program. I wrote an enumerated data type to represent the possible species and then stored one of each in an array. The setCurrSpp function will use pointers to point to species specific information depending on which Species type is passed in to the function. Storing the types in an array means that you can then easily cycle through all the species in the zoo with a single for loop or nested for loop (for cycling through all animals) instead of having to repeat the specific actions of the function for each species type.

One of the hurdles I then had with the setCurrSpp() function was finding a way to more generally expand the exhibits. The currSpp pointer is used in the program to point to an exhibit of animals. I used a temporary pointer to create a new exhibit twice the size of the first, then had the currSpp pointer free the current exhibit and point to the new one. However, what I failed to do in the first go around was to have the actual exhibit pointer also point to the new block of memory (not just the currSpp pointer) otherwise I would be left with dangling pointers.

Some other initial issues I encountered involved the sickness() function. I had not originally written the code to handle a situation in which there were no animals in the zoo. Though unlikely, this is a possible scenario and the game can proceed if there are no animals so long as the user still has money in the zoo to purchase new ones. For the attendanceBoost() function I also had to consider the case in which there were no tigers in the zoo. I decided that if any of the random events were not possible (no adults to give birth, no animals to be sick, no tigers to give a bonus) that I would not print any information regarding the intended event to the screen and instead just have the default noEvent() function be triggered.