Clinton Hawkes
CS162-400
Project 2 Design Document

Zoo Tycoon

Design:

Program will have a main function, a Zoo class, an Animal class, a Penguin class, a Tiger class, a Turtle class, and an input validation function.

1) Main function – only used to seed the rand() function, create a Zoo object and call the Zoo::run() function to start the game.
    A) Create a bool variable “State” to hold the current state of the game (run or quit)
    B) Call the menu function to display the main menu. Set State equal to return menu value
        I. Ask user if they want to begin building a zoo or quit
            a) If quit, return false to the run function. (game will end)
            b) If user wants to build a zoo
                i. User chooses if they want to purchase 1 or 2 penguins
                    1) Cost of penguin(s) deducted from bank account
                ii. User chooses if they want to purchase 1 or 2 tigers
                    1) Cost of tiger(s) deducted from bank account
                iii. User chooses if they want to purchase 1 or 2 turtles
                    1) Cost of turtle(s) deducted from bank account
                iv. Menu returns true to the run function when done
    C) While loop is setup that runs until the State variable is false. Each loop is one day
        I. Console screen is cleared
        II. ageAnimals() is called to increase the age of every animal by 1
        III. checkIfAdult() is called to set three bool variables (penguinParent, tigerParent, and turtleParent) to true if each type of animal has at least one with an age of 3
        IV. The dayNumber variable is incremented (simulating start of new day)
        V. “Day number #” is printed out on the screen, as well as current account balance and inventory of each type of zoo animal.
        VI. The cost to feed the current zoo animal inventory is calculated and a statement is printed on the console informing the user of the cost
            a) Cost to feed the animals is deducted from bank account
        VII. “Today’s Random Event” is printed to the console
            a) randomEvent() is called
                i. Random number between 0 and 4 is generated
                    1) If random number is 0, sickness() is called. This simulated one of the animals in the zoo getting sick and dying
                        A) Function checks to make sure there is at least one animal in the zoo that can get sick and dye

I. If not, “Your zoo does not have any animals” is printed to the console

B) If there are animals, a random number between 0 and 2 is generated

I. If num = 0 and there are penguins, one of the penguins is removed from the zoo.

a) If num = 0 and there are no penguins, another random number is generated

II. If num = 1 and there are tigers, one of the tigers is removed from the zoo.

a) If num = 1 and there are not tigers, another random number is generated

III. If num = 2 and there are turtles, one of the turtles is removed from the zoo.

a) If num = 1 and there are no turtles, another random number is generated

IV. A statement specifying which animal got sick and died is printed to the console

2) If random number is 1, boom() is called. Simulates a boom in attendance at the zoo

A) Function checks to see if there are any tigers in the zoo for the boom to occur

I. If no tigers, a statement informing the user a boom could not occur due to lack of tigers is printed to the console

B) If there are tigers, a random number between 250 and 500 is generated.

I. The number of tigers at the zoo is multiplied by the random number. This represents the bonus that is earned because of the boom.

II. The bonus is added to the bank account

III. A statement informing the user that a boom in attendance occurred, as well as the bonus amount earned is printed to the console

3) If random number is 2, babyBorn() is called. Simulates one of the animals at the zoo giving birth

A) Function checks to see if there is an animal at the zoo that can give birth. This information is obtained from the penguinParent, tigerParent and turtleParent variables. (at start of the day these variables are set to true if able to give birth)

I. If no animals are able to give birth, a statement informing the user of this is printed to the console

B) If there is at least one animal that can give birth, a random number between 0 and 2 is generated.

I. If num = 0 and there are penguins, then there are 5 new penguins added to the zoo

a) The age of the new penguins is set to 0

b) Statement informing user that 5 new penguins were hatched is printed to the console

II. if num = 1 and there are tigers, then there is 1 new tiger added to the zoo

a) The age of the new tiger is set to 0

b) Statement informing user that a new tiger was born is printed to the console

III. if num = 2 and there are turtles, then 10 new turtles are added to the zoo

a) The age of the new turtles is set to 0

b) Statement informing user that 10 new turtles hatched is printed to the console

4) If random number is 3, there is no random event for the day

A) Statement informing user that there is no random event happening to day is printed to the console

VIII. Statement informing user of the revenue earned for the day is printed to the console

a) Revenue is calculated by multiplying the number of each type of animal by it's respective payout amount.

b) Revenue for the day is added to the bank account

IX. User is asked if they would like to purchase an adult animal.

a) If no, program moves on to the next code block

b) if yes, buyAdult() is called.

i. Information is printed to the console detailing the user's current account balance and the name/price of each type of animal. User can choose from 3 animals or no purchase.

1) If user chooses option 1 (penguin), function checks to make sure there is enough money in the bank to make the purchase

A) If there is not enough money, the user is asked to make another selection

B) If there is enough money, the new adult penguin is added to the zoo

I. The age of the new penguin is set to 3

2) If user chooses option 2 (tiger), function checks to make sure there is enough money in the bank to make the purchase

A) If there is not enough money, the user is asked to make another selection

B) If there is enough money, the new tiger is added to the zoo

I. The age of the new tiger is set to 3

3) If user chooses option 3 (turtle), function checks to make sure there is enough money in the bank to make the purchase

A) If there is not enough money, the user is asked to make another selection

B) If there is enough money, the new turtle is added to the zoo

I. The age of the new turtle is set to 3

4) If user chooses option 4, no animal is purchased and the user returns to the run() function

X. Bank account balance is printed to the console

XI. User is asked if they would like to start the next day or quit

a) If the user chooses to quit

i. State variable is set to false

1) While loop is exited and the program is terminated

b) If the user chooses to start the next day

i. Nothing is changed. Loop continues

XII. endOfDay() is called (resets variables that calculated beginning of each day)

a) Bonus variable is set to 0

b) penguinParent variable is set to false

c) tigerParent variable is set to false

d) turtleParent variable is set to false

XIII. If bank account balance drops to 0 or below at the end of any day, user loses the game and a message stating they went bankrupt is printed to the console

a) State is set to false and game is exited

XIV. If bank account balance goes above 50 million at the end of any day, user wins the game and a message stating they retired is printed

a) State is set to false and game is exited

Zoo Class: Used to implement Zoo Tycoon. Main() takes a back seat and uses the Zoo class to call all supporting functions to play the game.

Variables:	Functions:
Ifstream inFile	Zoo()
Ofstream outFile	~Zoo()
String message	Bool mainMenu()
Bool penguinParent	Void run()
tigerParent	Void randomevent()
turtleParent	Void sickness()
Long bankBalance	Void boom()
Int dayNumer	Void babyBorn()
bonus	Void addPenguin()
numPenguins	Void addTiger()
numTigers	Void addTurtle()
numTurtles	Void removePenguin()
penguinArraySize	Void removeTiger()
tigerArraySize	Void removeTurtle()
TurtleArraySize	Int dailyCost()
Animal** penguinList	Int dailyPayoff()
tigerList	Void ageAnimals()
turtleList	Void checkIfAdult()
	Void endOfDay()
	Void buyAdult()

Animal Class: Used to store info about a Animal object.

Variables:	Functions:
Const int BASE_FOOD_COST	Animal()
Int age	Int getAge()
cost	Void setAge(int)

numberOfBabies	Int getCost()
payoff	Int getNumberOfBabies()
	Virtual int getDailyFoodCost()
	Virtual int getPayOff()
	Void increaseAge()

***Functions getDailyFoodCost() and getPayoff() are virtual functions and will be overridden by the derived classes function.

Penguin Class: Used to store info about a Penguin object. Derived from Animal class

Variables:	Functions:
***Inherits Animal variables	***Inherits Animal functions
	Penguin()
	Int getDailyFoodCost()
	Int getPayoff()

Tiger Class: Used to store info about a Tiger object. Derived from Animal class

Variables:	Functions:
***Inherits Animal variables	***Inherits Animal functions
	Tiger()
	Int getDailyFoodCost()
	Int getPayoff()

Turtle Class: Used to store info about a Turtle object. Derived from Animal class

Variables:	Functions:
***Inherits Animal variables	***Inherits Animal functions
	Turtle()
	Int getDailyFoodCost()
	Int getPayoff()

Menu Function: The menu functions print out greetings, instructions and options to the user. The functions will get input from the user and return the appropriate bool to the main().

Validation Function: The validation function will receive input from the user, check that the input is the correct data type, check if the input is within the specified range, and if it fails these test, the user will be prompted to enter a valid input. The validation functions will return a value of the indicated type back to the main() after all the tests are passed.

Test Plan for integerValidation()

Test Case	Input Value	Expected Result	Observed Result
Char	A, c, +, .	“Please enter a valid integer.”	“Please enter a valid integer.”
String	Four, .five, yes	“Please enter a valid integer.”	“Please enter a valid integer.”
Float	6.31, 07.0, 8.8	“Please enter a valid integer.”	“Please enter a valid integer.”
< Min (set at 1)	0, -1, -5	“Please enter a valid integer.”	“Please enter a valid integer.”
> Max (set at 4)	5, 9999	“Please enter a valid integer.”	“Please enter a valid integer.”
Lower limit (min=1)	1	Function accept and return value to main()	Function accepted 1 and returned 1 to main()
Upper limit (max=2)	2	Function accept and return value to main()	Function accepted 2 and returned 2 to main()

Test Plan for Zoo Tycoon

Test Case	Description	Expected Result
Penguin Object Creation	Create a new Penguin	1. Penguin object should be added on the heap 2. Penguin member variables should be set to values in constructor 3. penguinList should have a pointer pointing to the new Penguin object 4. numPenguins should be incremented to account for the new penguin
Tiger Object Creation	Create a new Tiger	1. Tiger object should be added on the heap 2. Tiger member variables should be set to values in constructor 3. tigerList should have a pointer pointing to the new Tiger object 4. numTigers should be incremented to account for the new Tiger
Turtle Object Creation	Create a new Turtle	1. Turtle object should be added on the heap 2. Turtle member variables should be set to values in constructor 3. turtleList should have a pointer pointing to the new Turtle object 4. numTurtles should be incremented to account for the new Turtle
Number of Penguins	1. Minimal number of penguins 2. Buy 11 penguins 3. Buy 50 penguins	1. Minimum penguins should be 0. The number of penguins should not be allowed to become negative 2. Array holding penguins should resize to 20 when 11 penguins are purchased. No memory issues should appear. 3. Array holding penguins should resize to 80 when there are 50 penguins purchases. No memory issues should appear.

Number of Tigers	1. Minimal number of tigers 2. Buy 11 tigers 3. Buy 50 tigers	1. Minimum tigers should be 0. The number of tigers should not be allowed to become negative 2. Array holding tigers should resize to 20 when 11 tigers are purchased. No memory issues should appear. 3. Array holding tigers should resize to 80 when there are 50 tigers purchases. No memory issues should appear.
Number of Turtles	1. Minimal number of turtles 2. Buy 11 turtles 3. Buy 50 turtles	1. Minimum turtles should be 0. The number of turtles should not be allowed to become negative 2. Array holding turtles should resize to 20 when 11 turtles are purchased. No memory issues should appear. 3. Array holding turtles should resize to 80 when there are 50 turtles purchases. No memory issues should appear.
Memory Leak	1. Start and run program using valgrind. Buy 2 penguins Buy 2 tigers Buy 2 turtles Don’t buy anymore animals Play for 10 days Quit 2. Start and run program using valgrind. Buy 2 penguins Buy 2 tigers Buy 2 turtles Continue to play until the number of each animal is > 50 Quit 3. Start program and immediately quit.	There should be no memory leaks for any of these cases.
Random Event	Random event is called every “day”	1. Events chosen should be randomly distributed 2. Random number generator should only output numbers between 0 and 3 3. Switch statement should correctly choose which event happens based on the random number
Sickness	RandomEvent() chooses the sickness event 1. A penguin gets sick 2. A tiger gets sick	1. As long as there is a penguin in the zoo, a penguin should be removed when it is chosen. If there are no penguins in the zoo, another random number is generated so another animal gets sick instead. 2. As long as there is a tiger in the zoo, a tiger

	3. A turtle gets sick 4. No animals in zoo	should be removed when it is chosen. If there are no tigers in the zoo, another random number is generated so another animal gets sick instead. 3. As long as there is a turtle in the zoo, a turtle should be removed when it is chosen. If there are no turtles in the zoo, another random number is generated so another animal gets sick instead. 4. A message stating there are no animals in the zoo should print and no animals are removed.
Boom	RandomEvent() chooses a boom to happen. 1. No tigers in zoo 2. 1 tiger in zoo 3. 13 tigers in zoo	1. A message should print out that there are no tigers in zoo for the user to receive a bonus for the boom. No bonus. 2. Rand() should generate a number between 250 and 500. A statement should print out stating the number that was generated. The bonus amount should be added to bank account. Another statement should print saying bonus amount earned 3. Rand should generate a number between 250 and 500. A statement should print out stating the number that was generated. The number should then be multiplied by the number of tigers in the zoo(13 currently) and stored in the bonus variable. The bonus amount should be added to the bank account
Babies Born	RandomEvent() chooses the babiesBorn event 1. There are no adult animals in zoo 2. Penguin is chosen to have babies 3. Tiger is chosen to have baby 4. Turtle is chosen to have babies	1. A statement should print informing the user that there are no adult animals able to have babies. 2. If there is an adult Penguin and it is chosen to have babies, 5 new babies should be added to the zoo. The age of the babies should be set to 0. If there is a penguin but not an adult, another animal will be chosen to have babies. 3. If there is an adult Tiger and it is chosen to have babies, 1 new baby should be added to the zoo. The age of the baby should be set to 0. If there is a tiger but not an adult, another animal will be chosen to have babies. 4. If there is an adult Turtle and it is chosen to have babies, 10 new babies should be added to the zoo. The age of the babies should be set to 0. If there is a turtle but not an adult, another animal will be chosen to have babies.
Buy Adult Penguin	1. Buy penguin with 0 in account 2. Buy penguin with > 100 in account	1. User should not be allowed to purchase a penguin if they don’t have enough money 2. User should be able to purchase a penguin. A new Penguin object should be created. The

		numpenguins variable should be incremented. The cost of a penguin should be taken out of the account. The age of the new penguin should be set to 3.
Buy Adult Tiger	1. Buy tiger with 0 in account 2. Buy tiger with > 100 in account	1. User should not be allowed to purchase a tiger if they don't have enough money 2. User should be able to purchase a tiger. A new Tiger object should be created. The numtigers variable should be incremented. The cost of a tiger should be taken out of the account. The age of the new tiger should be set to 3.
Buy Adult Turtle	1. Buy turtle with 0 in account 2. Buy turtle with > 100 in account	1. User should not be allowed to purchase a Turtle if they don't have enough money 2. User should be able to purchase a turtle. A new Turtle object should be created. The numTurtles variable should be incremented. The cost of a turtle should be taken out of the account. The age of the new turtle should be set to 3.
Main Menu	Press 1 to start building zoo	1. Current balance should print on screen 2. User is asked how many penguins they want to purchase (1 or 2) 3. User is asked how many tigers they want to purchase (1 or 2) 4. User is asked how many turtles they want to purchase (1 or 2)
Main Menu	Press 2 to exit	Program should exit
Screen Printing (Day loop)	Each day, the screen should print the correct information in sequential order.	1. Day number should print out 2. Current account balance should print 3. Zoo inventory should print 4. Cost to feed zoo animals prints 5. Random event prints 6. Daily payout is displayed 7. User asked to purchase adult 8. End of day balance printed 9. User asked to continue, quit
Purchase Adult Animal (Print Out)	Choose to purchase a new adult animal at the end of the day	1. Current account balance should show 2. All animals with their associated cost should print 3. Option not to purchase an animal should be shown
Balance <= 0	Set bankBalance = 0	At the end of the day, if the account balance is zero or negative, the game should end and user is notified they went bankrupt.
Balance >= 50 million	Set bankBalance = 50000000	At the end of the day, if the account balance is >= 50000000, the game should end and user is notified they went into retirement.

Reflection:

When I first started working on this project, I found it to be somewhat easy. I felt the topics taught and used in this project were the same as the ones used in the dice game. I'm sure is a lot of overlap, but it was kind of a relief. I really struggled with the dice program, so it was kind of nice to understand everything that was going on in this project. It made it really easy for me to plan this assignment out and I really didn't have any difficulty during that process. This does not mean it wasn't time consuming though. I spent many, MANY hours on this project. It was really good practice and I feel like I am becoming proficient with c++, because of the 20-30 hours spent on these projects. That is how I learn the best though. Repetition, repetition, repetition.

Issues:

There were a couple issues I experienced while completing this assignment, and they both had to do with memory access. (SegFault). The first issue started to occur shortly after any of the animal counts hit 11. This immediately made me think the issue was coming from having to resize the array holding that animal's pointers. At first I couldn't narrow it down, because the turtle went over 11, but the segfault happened when buying a tiger. I thought it had to be the same animal as the one with the array resized. It took me a bit longer to realize that when my array was being resized, I was deleting the animal(tiger, penguin, or turtle) objects and not just the old pointer arrays. I was still trying to reference the objects that were deleted, so this is where the segfault came from. I fixed the array resizing code and the issue went away. I sure was happy after figuring that out.

The second time I started to see the segfault was shortly after I began to implement the extra credit that writes to a file and then reads from the file. I was able to write the messages to the output file just fine, but I was having a difficult time getting the message to print to the screen correctly. The content of the output was okay, the formatting was off. I tried a couple different ways to alter the way the text printed, but nothing was quite right. I started searching online and got deep into read buffers and other c-string related functions that were over my head. I tried to implement a read buffer on ifstream(rdbuff()), and it printed the file to the console perfectly. The formatting was on point and I felt like a genius. I then quit the program and got the segfault. The error message was too cryptic for me to even begin narrowing down the cause, but I knew that it had to be the read buffer. It was the only think I had changed in an otherwise perfectly working program. I went back to the drawing board and came up with the solution you see now to write/read the messages for the program. If I had more time, I may have researched the topic more and made the read buffer solution work. Good learning experience for me.

Changes:

The first iteration of my program did not include Animal pointers to the Penguin, Tiger and Turtle objects. I started with arrays of pointers that matched the type of object it was pointing to. Both methods worked just fine, but I soon found out that the specifications of the program required the base class pointer to be pointing to the objects. It was an easy fix. I had the classes setup correctly, so I just had to go type in Animal** in place of the Turtle** Tiger** and Penguin**. Not a big change, but I thought I would mention it.

Another change that I made later in the assignment was creating a function that checks to see if each animal type has an adult that can have babies. I originally included this process in the babyBorn function that was called by the randomEvent function, but the function was getting too long and complicated. By separating the "checking" and "doing" portion of that function, the code is easier to read and follow.

I decided to add an endOfDay function to my program to reset some of the variables back to a null or zero value. There are a few variables in my program that hold important values specific to that day. Animals are added and removed each day, so many of the functions rely on the variables to tell them the current state of a specific attribute. I have these variables reset at the end of each day so they can be calculated fresh the next day.