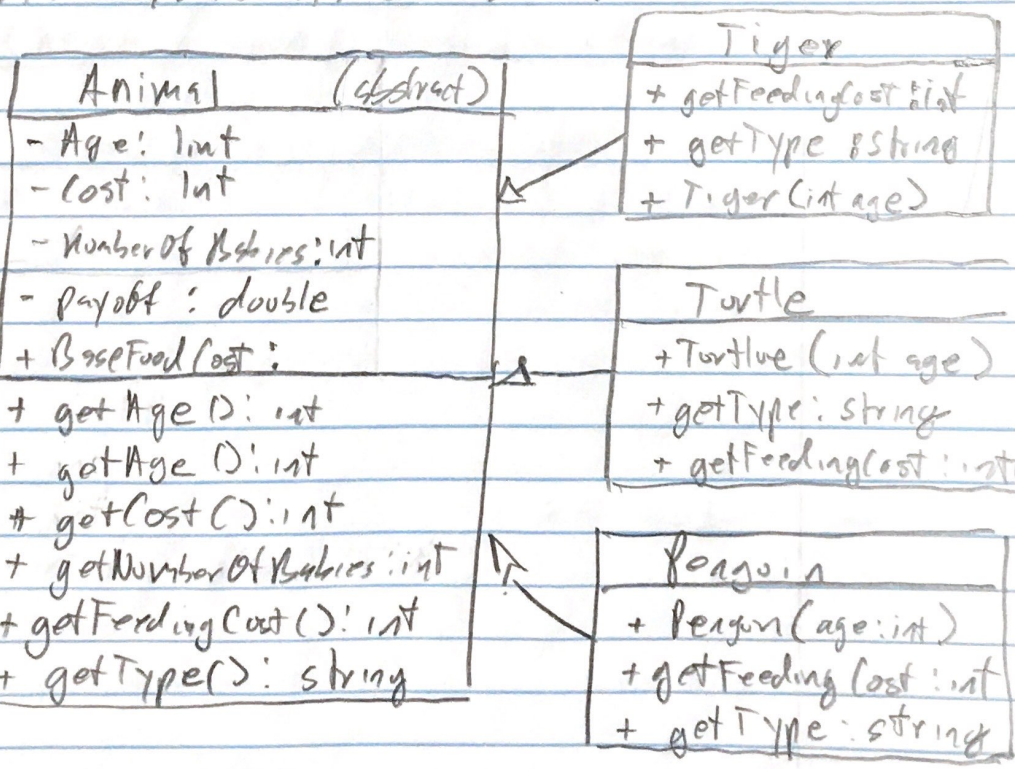
PSEUDOCODE

DISPLAY GAME BILLBOARD
PROMPT USER HOW MANY ANIMALS TO BUY (1-2 ONLY)
SHOW BANK ACCOUNT BALANCE AFTER PURCHASES (START = $100K)
START COUNTING DAYS
GENERATE RANDOM EVENTS OUT FOUR POSSIBILITIES
(e.g. animal born, sickness, ATTENDANCE BOOM (add to bank)
PROMPT USER TO BUY AN ANIMAL
DISPLAY BANK ACCOUNT BALANCE
ASKS USER IF CONTINUE PLAYING
REPEAT SIMULATION LOOP

UML:



DynamicArray
- Animal ** obj - length : int - nextIndex : int
+ DynamicArray () + DynamicArray () : ~ + add (Animal * item) : void + removeAt (index : int) : void + size ( ) : int

Zoo
- moneyInBank : int - DynamicArray ( )
+ Zoo () + run : void + getNumber (min : int, max : int) : int + getChoice (input : string) : string

Zoo simulation execution

SEE NEXT PAGE

EXECUTION DIAGRAM (SORT OF)

RUN FN

- START WITH 100K IN BANK
- EMPTY ANIMAL ARRAYS

PROMPT USER HOW MANY ANIMALS TO BUY
DO THIS FOR TIGERS, PENGUINS & TURTLES
ADD ANIMALS TO RESPECTIVE ARRAYS
AND DEDUCT COSTS FROM BANK ACCOUNT

e.g. `for(int i=0; i< numTigers; i++)`
`Tiger * tiger = new Tiger[1];`
`animals.add(tiger);`
`moneyInBank -= tiger -> getCost();`

START DO-WHILE LOOP
- set Bonus, event, r, profit = 0
- day to 1
- display bank account balance

→ DISPLAY DAY:
for loop to add age to animals
- deduct feeding cost for animals
- from Bank account

↳ GENERATE RANDOM EVENT
- USE SWITCH STATEMENT FOR FOUR POSSIBLE EVENTS

'C1' 1) sick
2) random → death remove AT().

'C2' attendance boom
- for loop animal size
- add money → random $[250-500]

Cz' - Animal is born
- generate random animal
to have baby
for loop as long as age >= 3
if Tigor
animals.add(new Tigor(0));
FOR ALL ANIMALS
else if no random adult Animal
choose available adult animal
to birth baby

DEFAULT:
→ Nothing happened Today

OUT OF SWITCH
For loop to get payoff
add bonuses to profit
profits to bank account

PROMPT IF USER WANT BOY ANIMALS
IF-ELSE FOR ANIMALS

PRINT ACCOUNT BALANCE FOR DAY
day++;
PROMP USER IF CONTINUE TO PLAY?
WHILE == "Y" and "y"

→ getNumber fn —
- verify input

REFLECTION - - - - - - - - - - - - - - - - - -

PROJECT 2 AFTER COMPLETING LAB 4 WAS NOT AS MOUNTAINOUS AS IT SEEMED THE FIRST TIME I READ IT. IMPLEMENTING THE CLASSES WASN'T SO HARD ANYMORE. POLYMORPHISM AND VIRTUAL FUNCTIONS NO LONGER SEEMED ESOTERIC TO ME. THE BIGGEST CHALLENGES FACING ME WERE HOW TO IMPLEMENT THE DYNAMIC ARRAYS AND HOW TO DESIGN THE SIMULATION LOGIC WITH THE PRIOR LABS/PROJECTS, THE DYNAMIC ARRAYS WERE SIMPLY PART OF THE CONSTRUCTOR. I DON'T RECALL CREATING ADD/REMOVE FUNCTIONS SO I DID SOME CONSULTING ONLINE - SINCE I FELT INADEQUATE IN IMPLEMENTING. FROM HERE, THE BEST POSSIBLE SOLUTION WAS TO CREATE A DYNAMIC ARRAY CLASS TO PLACE THE ADD/REMOVE FUNCTIONS. I THINK THIS WAS THE BEST WAY TO MOVE FORWARD.

PUTTING THE SIMULATION LOGIC TOGETHER WAS A REAL PAIN IN THE ASS. LIKE IN THE FIRST PROJECT, I STRUGGLED TO MOVE THE ANT, EVEN WITH THE HELP OF A PRO. SOFTWARE ENGINEER! WE TRIED IMPLEMENTING FUNCTIONS TO DICTATE THE ANTS MOVEMENT - BOT FACED MANY A ROADBLOCK. IN THIS PROJECT, I DECIDED TO CREATE ONE FUNCTION - .200. RUNS; AND SURROUND THE LOGIC IN WITH 2 ARGS 00 -

WHILE LOOP. THE RANDOM EVENTS CRIED OUT FOR A SWITCH/ STATEMENT. I CODED THE 4 POSSIBLE CASES - THEN GOT STUCK. I SWITCHED TO INPUT VALIDATION AND BROUGHT OVER THE SSTREAM FUNCTION -- JUST IMPROVED. I CAME BACK TO 200 RUN() FN AND IMPLEMENTED USER PROMPTS FOR ANIMALS, INITIALIZING THE DYNAMIC ARRAY ANIMAL INVENTORIES. ONCE I DID THAT, EVERY ELSE FELL IN TO PLACE. JUST TESTING, BREAKING, FIXING -- REPEATING. IF YOU TAKE A LOOK AT CASE '3' -- THAT WAS THE MONSTER OF THIS PROJECT -- BUT I WAS EQUIPPED THIS TIME, with the help of my tutor, notes, and some online sources.

TEST CASE	INPUT VALUES	DRIVER FUNCTIONS	EXPECTED OUTCOMES	OBSERVED OUTCOMES
INPUT = 1, 2 Animal purchases	1, 2	Main Zoo.run();	How many tigons?	SAME AS EXPECTED
INPUT > 4 Animal purchases	4, 8, 13	Main Zoo.run();	re-prompt for # of animals	SAME AS EXPECTED
INPUT < 0	0, -1	Main Zoo.run();	re-prompt for # of animals	SAME AS EXPECTED
AFTER DAY 1	none	Main Zoo.run() Dynamic Array animal classes	Random event	random event
AFTER DAY 2	none	//	random event	Same as day 1, rand() broken. Missing library directive