Lee, Melisa
leemeli@oregonstate.edu
CS 162-400
Project 2: Design + Reflection

Program Design:

Starting money amount: 100,000$
Make all transaction whole numbers for simplicity
Set base food as: 10$
Kill animal at last element of sub array
If user purchased an item pushing bank account to 0 exit game

Reuse validate function from previous assignments

Classes
Zoo
Animal
Tiger
Penguin
Turtle

Zoo Class
Member Variable (private):

Animal **zoo	//dynamic array of pointers point to subarray of each animal
Int bankAcc	//store current bank account funds
Int numPenguins	//store current number of penguins
Int numTigers	//store current number of tigers
Int numTurtle	//store current number of turtles
Int bonus	//store bonus

Member Function(public):
startEvent
    Description: randomly select a event: sick/birth/bonus/nothing
    Input: nothing
    Output: nothing
sickEvent
    Description: randomly select an animal type to kill remove from dynamic array
    Input: nothing
    Output: nothing
Pesudocode

```
  randomly pick an animal to die
    if tiger
```

```
        kill last tiger of the array
    if penguin
        kill last
    if turtle
        kill
```

Boom

Description: display and calculate random bonus amount base on num of tigers and add it to the bank

Input: nothing

Output: nothing

babyBorn

Description: select a random animal type to have a baby if first random animal type is not old enough chose another.

Input: nothing

Output: nothing

Pesudocode:

```
  randomly pick an animal
    iterate through array to check if animal is old enough
      if animal is old enough (age >=3)
        add babies to zoo depending on each possible baby for that animal type set age to 0
      else
        exit iteration to pick a random num - animal chosen
    display message no animal is old enough do nothing

  //randoNum(int min, int max)

  randomly pick an animal
    if tiger
      check array if any is old enough
    else
      check another
```

welcomeZoo

Description: display instruction to user. ask/validate/store starting num of animals

Input: nothing

Output: nothing

dayIteration:

Description: follow game flow, age animals, feed animal, random event, profit info, ask user to buy adult animal and ask if they would like to continue to play.

Input: nothing

Output: nothing

Pesudocode:

```
Each iteration = day
   start of day
      - increase all animal days to one
      - display and count food cost for all animals
         - if not enough funds notify user and exit Game
      - subtract cost of food from bankAcc
      - display current bankAcc
   randomize even occurs
   End of day
      - calculate total profit (payoff for each animal + bonus)
      - ask user if they would like to buy an adult animal
         if yes
            - display animal to choose from
             - add adult to zoo dynamic array and set age to 3
   Ask user if they would like to continue or leave
```

adultAnimal

Description: add adult animal type to the the zoo

Input: nothing

Output: nothing

checkBankAcc

Description: check if the bank acc have enough fund for a purchase

Input: int value

Output: nothing

Animal Class [base class]

Member Variable (private):

Int age	//store age of animal
Int cost	//store cost of animal
Int numberOfBabies	//store number of possible baby animal can have
Int baseFoodCost	//store cost of food for animal
Int payoff	//store payoff rate for animal

Member Function(public):

Default constructor

Animal constructor

Description: accept parameters to set to private member variables

Input: accepts 5 integers for age, cost, numberOfBabies, baseFoodCost and pay off.

Output: nothing

Tiger class [derived]

private:

public:

Tiger constructor
Description: accept parameter and set to private member variable
Input: tiger(age, cost, numBaby, foodCost, payOff)
Output:

Penguin class [derived]
Private:
Public:
Constructor
Input: (age, cost, numBaby, foodCost, payOff)
Output: Nothing

Turtle class [derived]
private:

Public:
Constructor
Input: (age, cost, numBaby, foodCost, payOff)
Output: Nothing

Test Table

Test Case 1: Baby Born Event
This test case checks if the function successfully creates a baby and is added into the array by console log each element of the array before and after birth.

Animals	Input	Driver function	Expected	Actual
Tiger	Age = 1	babyBorn()	No birth	No birth
Penguin	Age = 1	babyBorn()	No birth	No birth
Turtle	Age =1	babyBorn()	No birth	No birth

Test Case 2: Baby Born Event
This test case checks after the day 1 occurs and the user decides to buy an adult animal.

Animals	Input	Driver function	Expected	Actual
Tiger	Age = 3, 4	babyBorn()	Birth	Birth

Penguin	Age = 3	babyBorn()	Birth	Birth
Turtle	Age =3	babyBorn()	Birth	Birth

Test Case 3: Boom Event
This test case checks if the bonus is calculated correctly

Animals	Input (number of each animal)	Driver function	Expected	Actual
Tiger	2	boomEvent()	2* random num	600
Penguin	2	boomEvent()		
Turtle	1	boomEvent()		

Test Case 4: Sick Event
This test case checks the death of each animal. Determines the success of removal of a animal by console log each element of the array before and after death.

Animals	Input (number of each animal)	Driver function	Expected	Actual
Tiger	2	sickEvent()	1	1
Penguin	1	sickEvent()	0	0
Turtle	1	sickEvent()	0	0

Reflection

A design change I ended up making was my animal type constructor for tiger, penguin and turtle. Since the animal cost, possible number of babies produced, price of food and the payout for each type was a set amount. Instead of passing the value each time it was passed through the constructor. The age of the animal was the only changing value, therefore it was passed in as a parameter when the object is created.

While doubling the size of the array, I re-read the project requirements. I found out that doubling the array size should only occur when the animal type has reached its limit. Therefore, I created private member variables to hold the array size for each animal type.

I found it really helpful to write out the process by hand when iterating through my array. It also helped figure why I was having memory leaks. At first I was creating an array of objects, but I had to create a pointer pointing to an array holding objects.
For an example:
Zoo = [Tiger, Penguin, Turtle]
Tiger[0] = [TigerObj1, TigerObj2...]
Penguin[0] = [PenguinObj1, PenguinObj2..]
Turtle[0] = [TurtleObj1, TurtleObj2..]

Writing each step out has also helped me figure out that I was copying an array instead of a pointer to an array.
The steps I took are listed below:
You have to "move" all elements of the array to temp array
- iterate through the array and copy all elements to temp array
- besides the last element (element i want to delete)

delete pointer pointing to that array
recreate a pointer to the temp array

Overall, this project has helped me get better with using pointers and keeping track of them. I felt that I was repeating the copy array often for each class. I think I could have created another function that passes in an integer representing the type of animal. Then within the function have if and else if statements for each type and execute the copying of the array.