Zoo Tycoon

Goals

- Develop multiple classes from program requirements
- Implement inheritance

For this project, we will write a zoo tycoon game using classes and inheritance. Zoo tycoon is a game that allows players to run a zoo business. Different types of animals cost different prices, have different maintenance costs, and of course, return a different profit at the end of each day. For this game, **the player will be the proud owner of a virtual zoo that has spaces to house tigers, penguins and turtles.**

Requirements

Animal Class

The Animal class has the following member variables:

- **Age**
  - Adult if age >= 3 days
  - Baby if age < 3 days
- **Cost**
  - Tiger cost $10,000
  - Penguin cost $1,000
  - Turtle cost $100
- **Number of Babies**
  - Tigers have 1 baby
  - Penguins have 5 babies
  - Turtles have 10 babies
- **Base Food Cost**
  - You can get this base food cost from the user or set it as a constant. Example base food cost per animal per day: $10.
  - Tigers have a feeding cost of 5 times the base cost
  - Penguins have a feeding cost that is the same as the base cost
  - Turtles have a feeding cost that is 50% the base cost
- **Payoff**
  - A tiger's payoff per day is 20% of their cost per animal. (not counting bonus)
  - A penguin's payoff per day is 10% of their cost per animal
  - A turtle's payoff per day is 5% of their cost per animal

Note: please do not modify the variables names or add more member variables to this class.

Game Flow:

The player begins with a **specific amount of money in the bank**, e.g. 100,000 dollars. At the start, the user needs to buy three types of animals (tigers, penguins, turtles) to start the business. Each type should have a quantity of **either 1 or 2.** For each animal bought, the cost is subtracted from the bank. **All newly bought animals are 1 day old.**

Each turn is a "day". At the beginning of the day, all animals increase age by 1 day, and the user needs to pay the feeding cost of each animal. Feeding is required so the animals don't die. After the feeding cost is subtracted from the bank, one randomized event takes place during the day. You can determine how to implement the random functions by yourself. The random function will pick one random event from the following list:

Random Events:

1. **A sickness occurs to an animal in the zoo:**
    1. Pick an animal at random that will die
    2. Remove one animal of that type from the exhibit. (dynamic array in the zoo)
2. **A boom in zoo attendance occurs:**
    1. Generate a random bonus for the day, 250-500 dollars for each tiger in the zoo
    2. Add the bonus payoff for each tiger to the total payoff of the day as a reward
3. **A baby animal is born:**
    1. Pick an animal at random to have a baby
    2. Check if there is an animal old enough to be a parent (age >= 3), add babies to the zoo depending on the "number of babies" specific to the type of animal. If no animal is old enough of the randomly selected type, pick another type of animal. Baby animals start at age 0. For simplicity, you don't need to consider the gender of the adult animals in order to have babies. One adult animal is good enough to have babies.

Note: If no animals are able to give birth to baby animals in the zoo, your program needs to be able to recognize this and recover.

4**. Nothing happens**

After the random event, calculate the profit for the day based on the number of each animals and their payoff. If there is a bonus for the day, add it to the profit

as well. Before the day ends, ask the player if they would like to buy an **adult** animal. If they do, ask for the type of animal they would like, then add the animal to the zoo and subtract that cost from the bank. The adult animal that is bought will be 3 days old.

After the end of a day (the end of the game play loop), prompt user whether to keep playing or end the game. If the user has no money, print a message to tell the user the game is over, and end the game.

Class requirements

The following classes are required: **zoo, animal, tiger, penguin, and turtle**. The program also must **use inheritance**; the tiger, penguin, and turtle class must inherit from animal class.

Also, the zoo class should have a **dynamic array** for each type of animal. Each dynamic array should **have a capacity of 10 animals to start with**. If more animals are added, you should **resize the dynamic array by doubling the starting capacity to hold more animals**.

Input Validation

The requirements of input validation have been specified in the Project 1, Lab 3 documents. **Reminder:** Make sure to check for incorrect data types entered by the user and continue to re-prompt for input until valid input is received.

Reflection Document

The requirements for the reflection document have been specified in the Project 1 document.

Make sure your document has **design descriptions, test tables, and reflections**. The reflections include: changes in design, problems encountered, and how you solve those problems.

Extra Credit

New Animal: (5pts)

Add a new class for your new animal. It should also inherit from the Animal class. This should also allow the user to dynamically create a new animal during runtime, re-prompting them for each trait.

Status messages: (5pts)

- Put a message into a text file for each random event. For example, during a boom in zoo attendance, print:

"Today is National Tiger Day! Tigers generate money today! You made: 360 extra dollars for each tiger you own!"

- Use ifstream to read messages from the created text file.

Different Feed Types: (5pts)

Allow user to pick between 3 different types of feed at the start of each day.

Cheap: Half as expensive for all animals, sickness becomes twice as likely to occur.

Generic: behaves normally.

Premium: Twice as expensive for all animals, sickness becomes half as likely to occur.

What you need to Submit

- All the program files including header and source files (.cpp/.hpp)
- Makefile
- Textfile (If you did extra credit)
- Your reflection pdf file

Important: Put all the files in a single .zip file and submit it on Canvas. There should be no internal directories.

Grading

- Programming style and documentation - 10%
- Create the Animal class and object - 10%
- Create the Penguin, Turtle, and Tiger class objects – 15%
- Implement the game:
  - Implement the program to manage the feed cost and payoff at the end of each day - 10%

    - Implement the random events - 10%
    - Implement the game loop for each “day” - 20%
    - Resize the dynamic array correctly when the number of items exceeds the list capacity - 10%
- Implement input validation functions - 5%
- Reflection document including: design description, test table (test plans, test results), and the reflection: - 10%

PreviousNext