**Goals:**

Write a zoo tycoon game using classes and inheritance. The user should be able to manage a zoo containing tigers, penguins, and turtles. All of which cost different amounts, have different maintenance costs, and return a different amount of profit at the end of each day.

**ZooTycoon Main:**

* Prompt the user to either begin the game or quit.
* While the user decides to play the game:
  + Set the starting budget for the user.
  + Set the number of days to be simulated.
  + Prompt the user to "buy" three types of animals. (The user can by either 1 or 2 of each animal.)
    - For each animal bought, reduce the cost from the bank.
    - Set the age of each purchased animal to 1 day old.
  + Increment the current day.
  + For each day up to the total number of days to be played:
    - Increment the age of each animal by 1 day.
    - Subtract the cost of feeding each animal from the bank.
    - Call one random event:
      * Sickness: one random animal will be removed from the zoo
      * Boom in attendance: random bonus between $250 and $500 for each tiger in the zoo
      * Baby is born: If the animal is old enough to be a parent (>= 3 days), add babies to the zoo.
    - Calculate the profit for the day based on the number of each animal present and their respective payoff, adding any additional bonuses. Add total to the bank.
    - Ask the user if they would like to buy another animal.
      * If so, ask which kind, add one to the zoo, and subtract the cost from the bank.
    - Ask the user if they would like to continue or quit.
    - If the user has no money, print a message stating the game is over and end the game.

**Zoo class:**

Properties:

* bank: tracks the amount of money available to the user.
* tigers: a dynamic array used to store all the tigers present at the zoo.
* penguins: a dynamic array used to store all the penguins present at the zoo.
* turtles: a dynamic array used to store all the turtles present at the zoo.
* dailyBonus: stores the amount of any bonus received. Initialized to zero each day and reassigned if the random event for the day is a bonus.
* randomEvents: an enumerable used to store the different types of random events that can occur

Methods:

* resizeWhenFull: checks if the given array of animals is full (10 animals), and doubles the capacity if it is.
* calcTotalPayoff: calculates the payoff for the day by getting the result of calcAnimalPayoff for each animal, and adding any bonus.
* calcAnimalPayoff: calculates the payoff for the given array of animals.
* getRandomEvent: randomly selects and returns an event.
* checkFunds: determines whether the user has any money left. If they do, the game continues. Otherwise, print a message stating the game is over.

**Animal class:**

Properties:

* age: tracks the age of the animal.
* cost: determines the cost of the specific type of animal.
* numberOfBabies: determines how many babies the animal can have.
* baseFoodCost: determines the base cost of food for the animals.
* payoff: determines the amount of money received per animal, per day.

Methods:

* Get method for the age.

**Tiger class: Inherits from Animal**

* cost: 10,000
* numberOfBabies: 1
* baseFoodCost: 5 x base
* payoff: 20% of cost (2,000)

**Penguin class: Inherits from Animal**

* cost: 1,000
* numberOfBabies: 5
* baseFoodCost: same as base
* payoff: 10% of cost (100)

**Turtle class: Inherits from Animal**

* cost: 100
* numberOfBabies: 10
* baseFoodCost: 50% of base
* payoff: 5% of cost (5)