Huy Nguyen

CS 162 – Intro to CS II

Section 400 – Spring 2019

Project 2 – Test Plan

Animal class

* Test default constructor and gets
  + Age = 0
  + Cost = 0
  + Num babies = 0
  + Payoff = 0
* Test constructor with arguments and gets
  + Age = 5
  + Cost = 100
  + Num babies = 10
  + Payoff = 1000
* Test increment age
  + Get initial age
  + Call increment age
  + Call get age and it should be 1 more than initial age
* Test is adult
  + Create an animal at age 5
    - Run is adult
      * Should return true
  + Create an animal at age 1
    - Run is adult
      * Should return false
  + Create an animal at age 0
    - Call increment age 3 times
    - Run is adult
      * Should return true

Tiger class

* Test default constructor and gets
  + Age = 0
  + Purchase cost = 10000
  + Number of babies = 1
  + Feeding cost = 50
  + Payoff = 2000
  + Name = “Tiger”
* Test constructor with arguments and gets
  + Age = 5
* Test increment age
  + Get initial age
  + Call increment age
  + Call get age and it should be 1 more than initial age
* Test is adult
  + Create an animal at age 5
    - Run is adult
      * Should return true
  + Create an animal at age 1
    - Run is adult
      * Should return false
  + Create an animal at age 0
    - Call increment age 3 times
    - Run is adult
      * Should return true

Penguin class

* Test default constructor and gets
  + Age = 0
  + Purchase cost = 1000
  + Number of babies = 5
  + Feeding cost = 10
  + Payoff = 100
  + Name = “Penguin”
* Test constructor with arguments and gets
  + Age = 5
* Test increment age
  + Get initial age
  + Call increment age
  + Call get age and it should be 1 more than initial age
* Test is adult
  + Create an animal at age 5
    - Run is adult
      * Should return true
  + Create an animal at age 1
    - Run is adult
      * Should return false
  + Create an animal at age 0
    - Call increment age 3 times
    - Run is adult
      * Should return true

Turtle class

* Test default constructor and gets
  + Age = 0
  + Purchase cost = 100
  + Number of babies = 10
  + Feeding cost = 5
  + Payoff = 5
  + Name = “Turtle”
* Test constructor with arguments and gets
  + Age = 5
* Test increment age
  + Get initial age
  + Call increment age
  + Call get age and it should be 1 more than initial age
* Test is adult
  + Create an animal at age 5
    - Run is adult
      * Should return true
  + Create an animal at age 1
    - Run is adult
      * Should return false
  + Create an animal at age 0
    - Call increment age 3 times
    - Run is adult
      * Should return true

Custom Animal class

* Test default constructor and gets
  + Age = 0
  + Purchase cost = 0
  + Number of babies = 0
  + Feeding cost = 0
  + Payoff = 0
  + Name = “”
* Test constructor with arguments and gets
  + Name = “Monkey”
  + Age = 5
* Test increment age
  + Get initial age
  + Call increment age
  + Call get age and it should be 1 more than initial age
* Test is adult
  + Create an animal at age 5
    - Run is adult
      * Should return true
  + Create an animal at age 1
    - Run is adult
      * Should return false
  + Create an animal at age 0
    - Call increment age 3 times
    - Run is adult
      * Should return true

Zoo class

* Test start
  + Run the game
  + Prints welcome message and prints bank account
  + Should prompt to buy 3 types of animals (1 or 2 of each)
  + Creates animals that are 1 day old
  + For each animal created, prints that animal has been added
    - If one of each animal
      * Prints 3 times that said animal was created
    - If 2 of each animal
      * Prints 6 times that said animal was created
  + Subtracts from bank account
* Test run tycoon
  + For each day
  + Prints day number
  + Runs tasks for beginning of day
  + Runs tasks for midday
  + Runs tasks for end of day
* Test beginning of day
  + Animals should be one day older
  + Prints bank account
  + Asks the user for todays feed type
  + Subtract feeding cost from bank
  + Print all animals fed message
  + Print bank account
* Test midday
  + Random even should happen
  + Print status message about the random event that happened
  + Calculate profits
  + Print todays profits have been calculated message
  + Print bank account
  + Ask user to buy a new animal
* Test end of day
  + Prints exhibit count
  + Prints bank account
  + Checks if bankrupt
    - If bankrupt
      * End game
        + Print ending game stats
    - If not bankrupt, continue to ask player if they want to keep playing
  + Asks user if they want to keep playing
    - If no
      * End game
        + Print ending game stats
    - If yes
      * Proceed to the next day
* Test random events
  + Events should happen randomly and not in any order
  + Test nothing happens
    - Print message that nothing happened
  + Test boom in zoo attendance
    - Prints message that a boom in zoo attendance happened
    - Prints extra bonus profit message
      * If no tigers, profits should be 0
  + Test baby is born
    - Randomly selects animal to give birth
    - Prints message that an animal gave birth
    - Prints message that an animal was added to the exhibit
    - When exhibit counts are printed at the end of the day, there should be more of that animal, and maybe even larger exhibit capacity
    - There should not be any babies born to animals that are too young
    - There should not be any babies born if exhibit has no animals
  + Test animal dies
    - When a random animal gets sick and dies, print message that animal has died at what age
    - When exhibit counts are printed, said animal exhibit count should be decreased by 1
      * Unless user bought another one of that animal
    - There should not be any animal deaths if exhibit has no animals
* Test add animal
  + Should make a new animal
  + Add to the exhibit
  + Subtract from bank account
  + Print animal added message
  + If exhibit is full, the exhibit could increase capacity
  + These are visible when animal exhibits and bank accounts are printed at the end of the day
  + If animal type is a custom animal
    - It should successfully create a custom animal by using the overloaded add animal function
* Test remove animal
  + When an animal gets sick
  + Print exhibit count should show that it has one less of that animal
* Test add to exhibit
  + Print exhibit count should print count of animals in exhibit
  + Add an animal
  + Print exhibit count should have increased for that animal
* Test increase exhibit
  + Print exhibit count should print count of animals in exhibit and its capacity
  + Keep adding tigers until past the capacity
  + Print exhibit count should show that the tiger exhibit now has increased by 10
* Test increase age of all animals
  + Start with animals at 2 days old on the first day
    - None should be able to have a baby
  + During the next say
    - They should be able to have a baby
* Test feed all animals
  + Beginning of day should print bank account before feeding
  + Should print bank account again after feeding
  + The price difference should be equal to the cost to feed all animals
  + Test
    - Buy 1 of each animal
    - Day one bank account should be 88,900
    - After feeding
    - Bank account should be 88,835
* Test calculate profits for all animals
  + Buy 1 of each animal
  + Day one bank account should be 88,900
  + After feeding
  + Bank account should be 88,835
  + If no boom in zoo attendance
  + After calculating profits for the day bank account should be 90,940
* Test buy new animal
  + Tiger
    - Buys a new tiger and prints message
  + Penguin
    - Buys a new penguin and prints message
  + Turtle
    - Buys a new turtle and prints message
  + New custom animal
    - Prompts user for animal’s traits
    - Buys a new animal and prints message
  + When exhibit count is printed it should have one more of said animal
* Test prompt for feed type and change food cost multiplier
  + Buy 1 of each animal
  + Day one bank account should be 88,900
  + Should ask user what type of feed
  + Cheap
    - Costs of feeding should be cheaper for the day
    - Bank account should be 88,867.5
  + Generic
    - Cost for feeding should be the same
    - Bank account should be 88,835
  + Premium
    - Cost for feeding should be more expensive
    - Bank account should be 88,770
* Test check if bankrupt and isbankrupt
  + If bank account is less than 1
  + Should end the game with stats
    - Test by buying 10000 animals until out of money
  + Else it should continue to the next day
* Test prompt to keep playing
  + If user says yes
    - Continue to next day
  + If user says no
    - End the game with stats
* Test game over
  + Should tell the user game over
  + Print game stats
  + Print reason for exit
  + Game stops
* Test print exhibit count
  + Prints all the animals in the exhibit
  + Numbers should change when an animal is added, gives birth, or dies
* Test write\_read
  + When a random event happens it should write out to a file
  + Open the file and check if it’s the correct event
  + The same event should print to console