Lab Report: EECE2140 COMPUTING FUNDAMENTALS FOR ENGINEERS

Student Name: Devin Milan
Northeastern University
College of Engineering
Department of Electrical and Computer Engineering
Course Title: EECE 2140: COMPUTING FUNDAMENTALS FOR ENGINEERS
Instructor: Fatema Nafa

April 1, 2024

Student Information

Assignment: *6*
Student Name: *Devin Milan*
Date: *March 31, 2024*

1 Pseudocode for Python Code

Problem 1: Animal Hierarchy

```
Step 1: Create an Animal class as well as two methods, eat and
sleep
Step 2: Create subclasses, types of animals, that inherit this
class
Step 3: Create eat and sleep methods in these sub classes
Step 4: Create sub classes of those sub classes
Step 5: Create eat and sleep methods in those sub classes
Step 6: Initialize objects from each class, and add them to a list
Step 7: Create a method in Animal that takes an object
and calls its eat and sleep methods
Step 8: Iterate over the object list and call the parent method on
each
```

Problem 2: Shapes - Area and Perimeter

```
Step 1: Create a Shape class
Step 2: Create 3 sub classes: rectangle, triangle, and circle
Step 3: Create methods area and perimeter for each shape
Step 4: Add the corresponding formulas for area and perimeter
depending on which class you are in
Step 5: Initialize objects of each class, and add to list
Step 6: Iterate over the list and invoke the area and perimeter
methods of each object
```

Problem 3: Employee Salary Calculation

```
Step 1: Create an employee class with attributes name and
position
Step 2: Create subclasses hourly, salary, and commission
and add additional attributes based on the needs of each class
Step 3: Create a method calculate salary in each sub class
Step 4: Add the required formula for calculating
an employee of that type's salary
Step 5: Create an object from each sub class and add to list
Step 6: Iterate, and run calculate salary method
```

2 Python Code Documentation

Problem 1: Animal Hierarchy

Explanation: This problem requires creating a parent class, sub classes, and well as sub classes of those sub classes. It's a pretty simple inheritance problem, since the eat and sleep methods are just print statements. The main method runs the two parent methods that then run the eat and sleep methods on the input object

```
# Main animal class
class Animal:
    # These two functions just run the objects specific eat and
    sleep methods
    # They pass in an animal object
    # Not necessarily necessary, but the instructions asked
    def letsEat(animal):
        animal.eat()

    def letsSleep(animal):
        animal.sleep()

# Mammal class - inherits animal
class Mammal(Animal):
    def eat(self):
        print("This␣is␣a␣mammal!␣It␣likely␣eats␣both␣meat␣and
␣␣␣␣␣␣␣␣plants")

    def sleep(self):
        print("This␣is␣a␣mammal!␣It␣likely␣sleeps␣at␣night")

# Bird class - inherits animal
class Bird(Animal):
    def eat(self):
        print("This␣is␣a␣bird!␣Today␣it␣finds␣some␣wonderful
␣␣␣␣␣␣␣␣seed␣to␣eat")

    def sleep(self):
        print("This␣is␣a␣bird!␣It␣finds␣a␣nice␣tree␣to␣sleep␣in")

# Fish class - inherits animal
class Fish(Animal):
    def eat(self):
        print("This␣is␣a␣fish!␣It's␣probably␣finding␣another␣fish
␣␣␣␣␣␣␣␣to␣eat")

    def sleep(self):
        print("This␣is␣a␣fish!␣It␣does␣not␣sleep␣like␣other
␣␣␣␣␣␣␣␣animals")

# Dog class - inherits mammal (multilevel)
class Dog(Mammal):
```

```
    def eat(self):
        print("This␣is␣a␣dog!␣It␣gets␣a␣nice␣bowl␣of␣meat␣to␣eat")

    def sleep(self):
        print("This␣is␣a␣dog!␣It␣finds␣its␣favorite␣spot␣on␣the
␣␣␣␣␣␣␣␣bed␣to␣sleep")

# Eagle class - inherits bird
class Eagle(Bird):
    def eat(self):
        print("This␣is␣an␣eagle!␣It␣finds␣some␣fish␣to␣eat")

    def sleep(self):
        print("This␣is␣an␣eagle!␣It␣locks␣its␣foot␣in␣position␣to
␣␣␣␣␣␣␣␣sleep␣in␣a␣tree")

# Goldfish class - inherits fish
class Goldfish(Fish):
    def eat(self):
        print("This␣is␣a␣goldfish!␣It␣eats␣fish␣food")

    def sleep(self):
        print("This␣is␣a␣goldfish.␣It␣doesn't␣sleep!")

def main():
    # List of animal objects
    animals = [Mammal(),Bird(),Fish(),Dog(),Eagle(),Goldfish()]
    # Iterate over animal list
    for animal in animals:
        # Invoke parent method let's eat and let's sleep on object
        # That function will run the eat and sleep methods
        Animal.letsEat(animal)
        Animal.letsSleep(animal)

main()
```

Problem 2: Shapes - Area and Perim

Explanation: This problem involves creating a parent shape class and subclasses triangle, rectangle, and circle that inherit it. There is no constructor needed in the base class because all of our shapes have different required attributes. In each sub class, we have methods area and perimeter that calculate the formula for area and perimeter for the given shape the class represents. When you invoke the area and perim methods on a given object, it will run the correct method due to polymorphism

```
import math

# Basic shape class
class Shape:
    # parent area method -- should never actually run
    def area():
```

```
        print("This␣is␣the␣main␣shape␣class.")

    # parent perim method -- should never actually run
    def perimeter():
        print("This␣is␣the␣main␣shape␣class.")

# Circle Class - inherits shape
class Circle(Shape):
    # Constructs objects with attribute radius
    def __init__(self, radius):
        self.radius = radius

    # circle area method
    def area(self):
        # Area of a circle
        area = 3.14*(self.radius**2)
        # Display result
        print(f"The␣area␣of␣the␣circle␣with␣radius
␣␣␣␣␣␣␣␣{self.radius}␣is␣{area:.2f}")

    # Circle perim method
    def perimeter(self):
        # Perim of a circle
        perimeter = 2*3.14*self.radius
        # display result
        print(f"The␣perimeter␣of␣the␣circle␣with␣radius
␣␣␣␣␣␣␣␣{self.radius}␣is␣{perimeter:.2f}")

# Rectangle class (inherits shape)
class Rectangle(Shape):
    # Constructs objects with attributes length and width
    def __init__(self,length,width):
        self.length = length
        self.width = width

    # Area method for rectangle
    def area(self):
        # Area of a rectangle formula
        area = self.length*self.width
        # Display result
        print(f"The␣area␣of␣the␣rectangle␣with␣length
␣␣␣␣␣␣␣␣{self.length}␣and␣width␣{self.width}␣is␣{area:.2f}")

    # Rectangle perim method
    def perimeter(self):
        # Perim of a rectangle formula
        perimeter = 2*(self.length+self.width)
        # Display result
        print(f"The␣perimeter␣of␣the␣rectangle␣with␣length
␣␣␣␣␣␣␣␣{self.length}␣and␣width␣{self.width}␣is␣{perimeter:.2f}")
```

```
# Triangle class (inherits shape)
class Triangle (Shape):
    # Constructs objects with attributes base and height
    def __init__(self,base,height):
        self.base = base
        self.height = height

    # Area method for triangle
    def area(self):
        # Formula for area of a triangle
        area = 0.5*self.base*self.height
        # Display results
        print(f"The␣area␣of␣the␣triangle␣with␣base␣{self.base}
␣␣␣␣␣␣␣␣and␣height␣{self.height}␣is␣{area:.2f}")

    # Perim of triangle method (Assumes right triangle)
    def perimeter(self):
        # Calculates hypotenuse
        hyp = math.sqrt((self.base)**2 + (self.height)**2)
        # Calculates perimeter
        perimeter = self.base + self.height + hyp
        # Display results
        print(f"The␣perimeter␣of␣the␣triangle␣with␣base
␣␣␣␣␣␣␣␣{self.base}␣and␣height␣{self.height}␣is␣{perimeter:.2f}")

def main():
    # Creates test objects from each class
    circle = Circle(7)
    rectangle = Rectangle(4,9)
    triangle = Triangle(3,4)
    # List of the objects
    shape_list = [circle,rectangle,triangle]
    # Iterate over the list
    for item in shape_list:
        # Invoke the specific area and perimeter methods for each
        item.area()
        item.perimeter()

main()
```

Problem 3: Employee Salary Calculator

Explanation: We first create a base employee class with name and position. Then, we create subclasses with additional attributes such as hourly wage and hours worked in a year, yearly salary, etc. We calculate their salary based on their type of job (hourly, commission, or salary) by creating a method calculate salary in each subclass, then print the result.

```
# Parent employee class
class Employee:
    # Every employee object is constructed with a name and
```

```
    position
    def __init__(self,name,position):
        self.name = name
        self.position = position

    # Runs the salary method of the specific object
    def runSalaryMethod(self):
        # Runs the salary method for the current object
        self.calculateSalary()

# Child hourly employee class
class HourlyEmployee(Employee):
    # Overrides the employee constructor
    def __init__(self,name,position,hourly_wage,hours_worked):
        # This ensures that name and position are stored at the
        employee level
        # This makes sense since every employee has a name and
        position
        super().__init__(name,position)
        # Additional attributes defined
        self.hourly_wage = hourly_wage
        self.hours_worked = hours_worked

    # Calculate salary method for hourly employee
    def calculateSalary(self):
        # Hours * hourly wage
        salary = self.hourly_wage*self.hours_worked
        # Print salary
        print(f"{self.name}, {self.position}, has a salary of
        ${salary}")

# Child salary employee class
class SalaryEmployee(Employee):
    # Overrides the employee constructor
    def __init__(self,name,position,salary):
        # This ensures that name and position are stored at the
        employee level
        super().__init__(name,position)
        # Defines additional attribute
        self.salary = salary

    # Calculate salary mehthod for salary employees
    def calculateSalary(self):
        # Print salary attribute -- no calculation necessary
        print(f"{self.name}, {self.position},
        has a salary of ${self.salary}")

# Child commission employee class
class CommissionEmployee(Employee):
    # Overrides the employee constructor
    def __init__(self,name,position,price_per_commission,
```

```
    commissions_completed):
        # This ensures that name and position are stored at the
        employee level
        super().__init__(name,position)
        # Defines additional attributes
        self.price_per_commission = price_per_commission
        self.commissions_completed = commissions_completed

    # Calculate salary mehthod for commission employees
    def calculateSalary(self):
        # salary = amount of commissions * price charged
        per each commission
        salary = self.price_per_commission*self.commissions_completed
        print(f"{self.name}, {self.position}, has a salary of
␣␣␣␣␣␣␣␣${salary}")

def main():
    # Defines test objects for each type of employee
    hour = HourlyEmployee('Jesse','Fast Food Worker',15,1500)
    sal = SalaryEmployee('Mary','Electrical Engineer',134000)
    comm = CommissionEmployee('Bob','Car Salesman',550,120)
    # Puts these objects in a list
    employees= [hour,sal,comm]
    # Iteration over the list
    for employee in employees:
        # Run the run salary method, which then runs the calculate
        salary method
        # Could also do employee.calaculateSalary()
        employee.runSalaryMethod()

main()
```

3 Output Demonstration

Problem 1: Animal Hierarchy

```
This is a mammal! It likely eats both meat and plants
This is a mammal! It likely sleeps at night
This is a bird! Today it finds some wonderful seed to eat
This is a bird! It finds a nice tree to sleep in
This is a fish! It's probably finding another fish to eat
This is a fish! It does not sleep like other animals
This is a dog! It gets a nice bowl of meat to eat
This is a dog! It finds its favorite spot on the bed to sleep
This is an eagle! It finds some fish to eat
This is an eagle! It locks its foot in position to sleep in a tree
This is a goldfish! It eats fish food
This is a goldfish. It doesn't sleep!
```

Problem 2: Shapes - Area and Perim

```
The area of the circle with radius 7 is 153.86
The perimeter of the circle with radius 7 is 43.96
The area of the rectangle with length 4 and width 9 is 36.00
The perimeter of the rectangle with length 4 and width 9 is 26.00
The area of the triangle with base 3 and height 4 is 6.00
The perimeter of the triangle with base 3 and height 4 is 12.00
```

Problem 3: Employee Salary Calculation

```
Jesse, Fast Food Worker, has a salary of $22500
Mary, Electrical Engineer, has a salary of $134000
Bob, Car Salesman, has a salary of $66000
```