LAB CYCLE - 6

Experiment No :1

Date : 18/12/2024

Aim :

Define a class to represent a bank account. Include the following details like name of the depositor, account number, type of account, balance amount in the account. Write methods to assign initial values, to deposit an amount, withdraw an amount after checking the balance, to display details such as name, account number, account type and balance

Pseudocode :

```
Class bankacc:
    Method __init__(self, name, accno, acctype, bal=0.0):
        Initialize name, accno, acctype, bal

    Method deposit(self, amount):
        IF amount > 0 THEN
            Add amount to balance
            PRINT "Amount deposited, Current Balance: {self.bal}"
        ELSE
            PRINT "Amount should be greater than zero"

    Method withdraw(self, amount):
        IF amount <= self.bal THEN
            Subtract amount from balance
            PRINT "Amount debited, Current Balance: {self.bal}"
        ELSE
            PRINT "INSUFFICIENT BALANCE"

    Method disp(self):
        PRINT "\nACCOUNT DETAILS"
        PRINT "NAME: {self.name}"
        PRINT "ACCOUNT NUMBER: {self.accno}"
        PRINT "ACCOUNT TYPE: {self.acctype}"
        PRINT "BALANCE: {self.bal}"

Main:
    GET name, accountno, accounttype, balance as inputs
    Create account using bankacc(name, accountno, accounttype, balance)
```

```
WHILE True:
    DISPLAY menu options (1: Deposit, 2: Withdraw, 3: Display, 4: Exit)
    GET user's choice

    IF choice == 1:
        GET deposit amount
        Call deposit method of account
    ELSE IF choice == 2:
        GET withdrawal amount
        Call withdraw method of account
    ELSE IF choice == 3:
        Call disp method of account
    ELSE IF choice == 4:
        PRINT "Exiting..."
        BREAK
    ELSE:
        PRINT "Invalid choice! Please try again."
```

Source Code :

```
class bankacc:
        def __init__(self,name,accno,acctype,bal=0.0):
                self.name=name
                self.accno=accno
                self.acctype=acctype
                self.bal=bal
        def deposit(self,amount):
                if amount>0:
                        self.bal+=amount
                        print(f"{amount} deposited \n Current Balance:{self.bal}")
                else:
                        printf("Amount should be greater than zero")
        def withdraw(self,amount):
                if amount<=self.bal:
                        self.bal-=amount
                        print(f"{amount} debited \n Current Balance:{self.bal}")
                else:
                        printf("INSUFFICIENT BALANCE")
        def disp(self):
        print("\nACCOUNT DETAILS\n")
        print(f"\nNAME: {self.name}\n")
        print(f"\nACCOUNT NUMBER: {self.accno}\n")
        print(f"\nACCOUNT TYPE: {self.acctype}\n")
        print(f"\nBALANCE: {self.bal}\n")
name=input("\nEnter your name:")
accountno=input("\nEnter Account Number:")
```

```
accounttype=input("\nEnter Account type:")
balance=float(input("\nEnter the initial balance: "))
account=bankacc(name,accountno,accounttype,balance)
while True:
        print("\n 1.DEPOSIT \n")
        print("\n 2.WITHDRAW \n")
        print("\n 3.DISPLAY   \n")
        print("\n 4.EXIT \n")
        choice=int(input("\n Enter your choice: "))
        if choice==1:
                amount=float(input("Enter the amount to be deposited: "))
                account.deposit(amount)
        elif choice == 2:
                amount=float(input("Enter amount to withdraw: "))
                account.withdraw(amount)
        elif choice==3:
                account.disp()
        elif choice==4:
                print("Exiting...")
                break
        else:
                print("Invalid choice! Please try again.")
```

Output :

```
Enter your name:Hari
Enter Account Number:20
Enter Account type: savings
Enter the initial balance: 200

1.DEPOSIT
2.WITHDRAW
3.DISPLAY
4.EXIT


Enter your choice: 1
Enter the amount to be deposited: 20
20.0 deposited
Current Balance: 220.0

1.DEPOSIT
2.WITHDRAW
3.DISPLAY
4. EXIT

Enter your choice: 2
Enter amount to withdraw: 20
20.0 debited
Current Balance: 200.0

1.DEPOSIT
2.WITHDRAW
3.DISPLAY
4.EXIT

Enter your choice: 3
ACCOUNT DETAILS
NAME: Hari
ACCOUNT NUMBER: 20
ACCOUNT TYPE: savings
BALANCE: 200.0

1. DEPOSIT
2.WITHDRAW
3.DISPLAY
4.EXIT
Enter your choice: 4
Exiting…
```

Result : The program is successfully executed and the output is verified.

Experiment No :2

Date: 18/12/2024

Aim :

Create a class Publisher with attributes publisher id and publisher name. Derive class Book from Publisher with attributes title and author.Derive class Python from Book with attributes price and no_of_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.

Pseudocode :

```
Class Publisher:
    Method __init__(self, publisher_id, publisher_name):
        Initialize publisher_id and publisher_name

    Method display(self):
        PRINT "Publisher Details:"
        PRINT "Publisher ID: {self.publisher_id}"
        PRINT "Publisher Name: {self.publisher_name}"

Class Book (inherits Publisher):
    Method __init__(self, publisher_id, publisher_name, title, author):
        Call the constructor of Publisher
        Initialize title and author

    Method display(self):
        Call the display method of Publisher
        PRINT "Book Details:"
        PRINT "Title: {self.title}"
        PRINT "Author: {self.author}"

Class Python (inherits Book):
    Method __init__(self, publisher_id, publisher_name, title, author, price, no_of_pages):
        Call the constructor of Book
        Initialize price and no_of_pages

    Method display(self):
        Call the display method of Book
        PRINT "Python Book Details:"
        PRINT "Price: {self.price}"
        PRINT "Number of Pages: {self.no_of_pages}"
```

```
Main:
    PRINT "Enter Publisher Details:"
    GET publisher_id and publisher_name

    PRINT "Enter Book Details:"
    GET title and author

    PRINT "Enter Python Book Details:"
    GET price and no_of_pages
    Create python_book using Python(publisher_id, publisher_name, title, author, price,
no_of_pages)

    PRINT "Complete Book Information:"
    Call display method of python_book
```

Source Code :

```
class Publisher:
    def __init__(self, publisher_id, publisher_name):
        self.publisher_id = publisher_id
        self.publisher_name = publisher_name

    def display(self):
        print("Publisher Details:")
        print(f"Publisher ID: {self.publisher_id}")
        print(f"Publisher Name: {self.publisher_name}")


class Book(Publisher):
    def __init__(self, publisher_id, publisher_name, title, author):
        super().__init__(publisher_id, publisher_name)
        self.title = title
        self.author = author

    def display(self):
        super().display()
        print("\nBook Details:")
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")


class Python(Book):
    def __init__(self, publisher_id, publisher_name, title, author, price, no_of_pages):
        super().__init__(publisher_id, publisher_name, title, author)
        self.price = price
        self.no_of_pages = no_of_pages
    def display(self):
        super().display()
        print("\nPython Book Details:")
        print(f"Price: {self.price}")
        print(f"Number of Pages: {self.no_of_pages}")

print("Enter Publisher Details:")
publisher_id = input("Publisher ID: ")
publisher_name = input("Publisher Name: ")

print("\nEnter Book Details:")
title = input("Book Title: ")
author = input("Book Author: ")

print("\nEnter Python Book Details:")
price = float(input("Price: "))
no_of_pages = int(input("Number of Pages: "))
python_book = Python(publisher_id,publisher_name,title, author,price, no_of_pages)
print("\nComplete Book Information:")
python_book.display()
```

Output :

```
Enter Publisher Details:
Publisher ID: 101
Publisher Name: O'Reilly Media

Enter Book Details:
Book Title: Learning Python
Book Author: Mark Lutz

Enter Python Book Details:
Price: 49.99
Number of Pages: 600

Complete Book Information:
Publisher Details:
Publisher ID: 101
Publisher Name: O'Reilly Media

Book Details:
Title: Learning Python
Author: Mark Lutz

Python Book Details:
Price: 49.99
Number of Pages: 600
```

Result : The program is successfully executed and the output is verified.

Experiment No :3

Date: 18/12/2024

Aim :

Write a program that has an abstract class Polygon. Derive two classes Rectangle and Triangle from Polygon and write methods to get the details of their dimensions and hence calculate the area.

Pseudocode :

```
Class Polygon (Abstract Base Class):
    Method get_dimensions(self):
        Abstract method to get polygon dimensions

    Method compute_area(self):
        Abstract method to calculate polygon area

Class Rectangle (inherits Polygon):
    Method get_dimensions(self):
        GET length from user
        GET width from user

    Method compute_area(self):
        RETURN length * width

Class Triangle (inherits Polygon):
    Method get_dimensions(self):
        GET base_length from user
        GET height from user

    Method compute_area(self):
        RETURN 0.5 * base_length * height

Main:
    WHILE True:
        PRINT "Polygon Options:"
        PRINT options for Rectangle, Triangle, Exit

        GET user_choice

        IF user_choice == '1':
            CREATE Rectangle object
            CALL get_dimensions method for Rectangle
            PRINT Rectangle area

        ELSE IF user_choice == '2':
            CREATE Triangle object
            CALL get_dimensions method for Triangle
            PRINT Triangle area

        ELSE IF user_choice == '3':
            PRINT "THANK YOU!"
            BREAK

        ELSE:
            PRINT "Invalid choice. Please try again."
```

Source Code :

```
from abc import ABC, abstractmethod
class Polygon(ABC):
    @abstractmethod
    def get_dimensions(self):
        """Abstract method to get polygon dimensions"""
        pass

    @abstractmethod
    def compute_area(self):
        """Abstract method to calculate polygon area"""
        pass
class Rectangle(Polygon):
    def get_dimensions(self):
        """Get rectangle dimensions from user input"""
        self.length = float(input("Enter the length of the rectangle: "))
        self.width = float(input("Enter the width of the rectangle: "))

    def compute_area(self):
        """Calculate and return rectangle area"""
        return self.length * self.width
class Triangle(Polygon):
    def get_dimensions(self):
        """Get triangle dimensions from user input"""
        self.base_length = float(input("Enter the base length of the triangle: "))
        self.height = float(input("Enter the height of the triangle: "))

    def compute_area(self):
        """Calculate and return triangle area"""
        return 0.5 * self.base_length * self.height
while True:
    print("\nPolygon Options\n")
    print("1. Rectangle")
    print("2. Triangle")
    print("3. Exit")

    user_choice = input("Enter your choice from 1 to 3: ")

    if user_choice == '1':
        rect = Rectangle()
        rect.get_dimensions()
        print(f"Rectangle Area: {rect.compute_area()}")

    elif user_choice == '2':
        tri = Triangle()
        tri.get_dimensions()
        print(f"Triangle Area: {tri.compute_area()}")

    elif user_choice == '3':
        print("THANK YOU!")
        break

    else:
        print("Invalid choice. Please try again.")
```

Output :

```
Polygon Options

1. Rectangle
2. Triangle
3. Exit
Enter your choice from 1 to 3: 1
Enter the length of the rectangle: 5
Enter the width of the rectangle: 3
Rectangle Area: 15.0


Polygon Options

1. Rectangle
2. Triangle
3. Exit
Enter your choice from 1 to 3: 2
Enter the base length of the triangle: 4
Enter the height of the triangle: 6
Triangle Area: 12.0
```

```
Polygon Options

1. Rectangle
2. Triangle
3. Exit
Enter your choice from 1 to 3: 3
THANK YOU!
```

Result : The program is successfully executed and the output is verified.

Experiment No :4

Date: 18/12/2024

Aim :

Create a Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

Pseudocode :

```
Class Rectangle:
    Method __init__(self, length, breadth):
        Initialize length and breadth

    Method calculate_area(self):
        RETURN length * breadth

    Method calculate_perimeter(self):
        RETURN 2 * (length + breadth)

    Method compare_area(self, other):
        IF other is not an instance of Rectangle THEN
            RAISE TypeError "Can only compare with another Rectangle object"

        Calculate current_area using calculate_area method
        Calculate other_area using calculate_area method

        IF current_area > other_area THEN
            RETURN "This rectangle is larger than the other rectangle"
        ELSE IF current_area < other_area THEN
            RETURN "This rectangle is smaller than the other rectangle"
        ELSE
            RETURN "Both rectangles have equal area"

Main:
    PRINT "Enter details for first rectangle:"
    GET length1 and breadth1 from user
    CREATE rect1 using Rectangle(length1, breadth1)

    PRINT "Enter details for second rectangle:"
    GET length2 and breadth2 from user
    CREATE rect2 using Rectangle(length2, breadth2)

    PRINT "First Rectangle:"
    PRINT area and perimeter of rect1
    PRINT "Second Rectangle:"
    PRINT area and perimeter of rect2

    PRINT "Comparison:"
    CALL compare_area method for rect1 with rect2 and print result
```

Source Code :

```
class Rectangle:
    def __init__(self, length, breadth):
        self.length = length
        self.breadth = breadth
    def calculate_area(self):
        return self.length * self.breadth
    def calculate_perimeter(self):
        return 2 * (self.length + self.breadth)
    def compare_area(self, other):
        if not isinstance(other, Rectangle):
            raise TypeError("Can only compare with another Rectangle object")

        current_area = self.calculate_area()
        other_area = other.calculate_area()

        if current_area > other_area:
            return f"This rectangle (Area: {current_area}) is larger than the other rectangle (Area: {other_area})"
        elif current_area < other_area:
            return f"This rectangle (Area: {current_area}) is smaller than the other rectangle (Area: {other_area})"
        else:
            return f"Both rectangles have equal area: {current_area}"
print("Enter details for first rectangle:")
length1 = float(input("Enter length: "))
breadth1 = float(input("Enter breadth: "))
rect1 = Rectangle(length1, breadth1)
print("\nEnter details for second rectangle:")
length2 = float(input("Enter length: "))
breadth2 = float(input("Enter breadth: "))
rect2 = Rectangle(length2, breadth2)
print("\nFirst Rectangle:")
print(f"Area: {rect1.calculate_area()}")
print(f"Perimeter: {rect1.calculate_perimeter()}")
print("\nSecond Rectangle:")
print(f"Area: {rect2.calculate_area()}")
print(f"Perimeter: {rect2.calculate_perimeter()}")
print("\nComparison:")
print(rect1.compare_area(rect2))
```

Output :

```
Enter details for first rectangle:
Enter length: 4
Enter breadth: 5

Enter details for second rectangle:
Enter length: 6
Enter breadth: 3

First Rectangle:
Area: 20.0
Perimeter: 18.0

Second Rectangle:
Area: 18.0
Perimeter: 18.0

Comparison:
This rectangle (Area: 20.0) is larger than the other rectangle (Area: 18.0)
```

Result : The program is successfully executed and the output is verified.

Experiment No :5

Date: 18/12/2024

Aim :

Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 times.

Pseudocode :

```
Class Time:
    Method __init__(self, hrs=0, mins=0, secs=0):
        Initialize __hrs, __mins, and __secs to 0
        Call set_time(hrs, mins, secs) to set the time

    Method set_time(self, hrs, mins, secs):
        IF 0 <= hrs < 24 AND 0 <= mins < 60 AND 0 <= secs < 60 THEN
            Set __hrs, __mins, __secs to hrs, mins, secs
        ELSE:
            RAISE ValueError "Invalid time values"

    Method __add__(self, other):
        Calculate total seconds for both Time objects
        Add the total seconds and convert the result to hours, minutes, and seconds
        IF new_hrs >= 24 THEN:
            Set new_hrs = new_hrs % 24 (to ensure time is within a 24-hour range)
        RETURN a new Time object with new_hrs, new_mins, new_secs

    Method display(self):
        PRINT time in the format hh:mm:ss

Main:
    PRINT "Enter the first time:"
    GET hours, minutes, seconds for the first time
    CREATE time1 using Time(hrs1, mins1, secs1)

    PRINT "First Time:"
    CALL display method of time1

    PRINT "Enter the second time:"
    GET hours, minutes, seconds for the second time
    CREATE time2 using Time(hrs2, mins2, secs2)

    PRINT "Second Time:"
    CALL display method of time2
    SUM time1 and time2 using the '+' operator
    PRINT "Sum of the Times:"
    CALL display method of sum_time
```

Source Code :

```
class Time:
    def __init__(self, hrs=0, mins=0, secs=0):
        self.__hrs = 0
        self.__mins = 0
        self.__secs = 0
        self.set_time(hrs, mins, secs)
    def set_time(self, hrs, mins, secs):
        """Set time values, ensuring they are valid."""
        if (0 <= hrs < 24 and
            0 <= mins < 60 and
            0 <= secs < 60):
            self.__hrs = hrs
            self.__mins = mins
            self.__secs = secs
        else:
            raise ValueError("Invalid time values")
    def __add__(self, other):
        """Overload the '+' operator to add two Time objects."""
        total_secs1 = (self.__hrs * 3600 + self.__mins * 60 + self.__secs)
        total_secs2 = (other.__hrs * 3600 + other.__mins * 60 + other.__secs)
        total_secs = total_secs1 + total_secs2
        new_hrs = total_secs // 3600
        remaining_secs = total_secs % 3600
        new_mins = remaining_secs // 60
        new_secs = remaining_secs % 60
        new_hrs %= 24
        return Time(new_hrs, new_mins, new_secs)
    def display(self):
        """Display the time in the format hh:mm:ss."""
        print(f"{self.__hrs:02d}:{self.__mins:02d}:{self.__secs:02d}")
print("Enter the first time:")
hrs1 = int(input("Hours: "))
mins1 = int(input("Minutes: "))
secs1 = int(input("Seconds: "))

time1 = Time(hrs1, mins1, secs1)
print("First Time: ", end="")
time1.display()
print("\nEnter the second time:")
hrs2 = int(input("Hours: "))
```

```
mins2 = int(input("Minutes: "))
secs2 = int(input("Seconds: "))
time2 = Time(hrs2, mins2, secs2)
print("Second Time: ", end="")
time2.display()
sum_time = time1 + time2
print("\nSum of the Times: ", end="")
sum_time.display()
```

Output :

```
Enter the first time:
Hours: 2
Minutes: 45
Seconds: 30
First Time: 02:45:30

Enter the second time:
Hours: 3
Minutes: 20
Seconds: 40
Second Time: 03:20:40

Sum of the Times: 06:06:10
```

Result : The program is successfully executed and the output is verified.