BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

Name	**Debjit Ghosal**
UID no.	**2023300065**
Experiment No.	**2**

AIM:	OOP Assignment
Program 1	
PROBLEM STATEMENT :	1) Employee Management :- Question: Design a base class Employee with private attributes for name, employee_id, and salary. Create subclasses FullTimeEmployee and PartTimeEmployee that inherit from Employee and add their respective attributes and methods. Include a method to calculate annual salary in both subclasses and ensure proper access to private variables.
PROGRAM:	

```
class Employee:
    def __init__(self, name, employee_id, salary):
        self.__name = name
        self.__employee_id = employee_id
        self.__salary = salary

    def get_name(self):
        return self.__name

    def get_employee_id(self):
        return self.__employee_id

    def get_salary(self):
        return self.__salary

class FullTimeEmployee(Employee):
    def __init__(self, name, employee_id, salary, annual_bonus):
        super().__init__(name, employee_id, salary)
        self.annual_bonus = annual_bonus
```

```
    def calculate_annual_salary(self):
        return (self.get_salary() * 12) + self.annual_bonus

    def __str__(self):
        return f"FullTimeEmployee(Name: {self.get_name()}, ID:
{self.get_employee_id()}, Salary: {self.get_salary()}, Annual Bonus:
{self.annual_bonus})"

class PartTimeEmployee(Employee):
    def __init__(self, name, employee_id, hourly_rate, hours_per_week):
        super().__init__(name, employee_id, hourly_rate)
        self.hours_per_week = hours_per_week

    def calculate_annual_salary(self):
        weekly_salary = self.get_salary() * self.hours_per_week
        return weekly_salary * 52

    def __str__(self):
        return f"PartTimeEmployee(Name: {self.get_name()}, ID:
{self.get_employee_id()}, Hourly Rate: {self.get_salary()}, Hours Per
Week: {self.hours_per_week})"

# Function to take input and create employees
def create_employees():
    print("Enter Full-Time Employee Details")
    name_ft = input("Enter Full-Time Employee's Name: ")
    id_ft = input("Enter Full-Time Employee's ID: ")
    salary_ft = float(input("Enter Full-Time Employee's Monthly Salary: "))
    bonus_ft = float(input("Enter Full-Time Employee's Annual Bonus: "))

    ft_employee = FullTimeEmployee(name_ft, id_ft, salary_ft, bonus_ft)

    print("\nEnter Part-Time Employee Details")
    name_pt = input("Enter Part-Time Employee's Name: ")
    id_pt = input("Enter Part-Time Employee's ID: ")
    hourly_rate_pt = float(input("Enter Part-Time Employee's Hourly Rate:
"))
```

```
    hours_per_week_pt = int(input("Enter Part-Time Employee's Hours Per Week: "))

    pt_employee = PartTimeEmployee(name_pt, id_pt, hourly_rate_pt, hours_per_week_pt)

    print("\n" + str(ft_employee))
    print(f"Annual Salary: {ft_employee.calculate_annual_salary()}")

    print("\n" + str(pt_employee))
    print(f"Annual Salary: {pt_employee.calculate_annual_salary()}")

create_employees()
```

RESULT:

```
Enter Full-Time Employee Details
Enter Full-Time Employee's Name: Debjit Ghosal
Enter Full-Time Employee's ID: 123
Enter Full-Time Employee's Monthly Salary: 6000
Enter Full-Time Employee's Annual Bonus: 2000

Enter Part-Time Employee Details
Enter Part-Time Employee's Name: Samarth Dave
Enter Part-Time Employee's ID: 456
Enter Part-Time Employee's Hourly Rate: 500
Enter Part-Time Employee's Hours Per Week: 20

FullTimeEmployee(Name: Debjit Ghosal, ID: 123, Salary: 6000.0, Annual Bonus: 2000.0)
Annual Salary: 74000.0

PartTimeEmployee(Name: Samarth Dave, ID: 456, Hourly Rate: 500.0, Hours Per Week: 20)
Annual Salary: 520000.0
```

I have learnt the basic of using objects and classes in python language.

Program 2

PROBLEM STATEMENT :	2) Hotel Booking System Question: Implement a class Room with attributes for room_number, room_type, and price_per_night. Include methods to check availability, book a room, and display booking details. Simulate room bookings with multiple instances of Room.

PROGRAM:

```
class Room:
    def __init__(self, room_number, room_type, price_per_night):
        self.room_number = room_number
        self.room_type = room_type
        self.price_per_night = price_per_night
        self.is_available = True
        self.booked_by = None
        self.booking_start_date = None
        self.booking_end_date = None

    def check_availability(self):
        return self.is_available

    def book_room(self, guest_name, start_date, end_date):
        if self.is_available:
            self.is_available = False
            self.booked_by = guest_name
            self.booking_start_date = start_date
            self.booking_end_date = end_date
            print(f"Room {self.room_number} successfully booked by
{guest_name}.")
        else:
            print(f"Room {self.room_number} is not available for booking.")

    def display_booking_details(self):
        print(f"\nRoom Number: {self.room_number}")
        print(f"Room Type: {self.room_type}")
        print(f"Price per Night: {self.price_per_night:.2f}")
        if self.is_available:
            print("Status: Available")
        else:
            print(f"Status: Booked by {self.booked_by}")
            print(f"Booking Dates:\n From: {self.booking_start_date} To:
{self.booking_end_date}")

# Simulate room bookings
def simulate_booking_system():
```

```
    # Create some room instances
    room_101 = Room(101, "Single", 1500.00)
    room_102 = Room(102, "Double", 2150.00)

    # Display available rooms
    room_101.display_booking_details()
    room_102.display_booking_details()

    # Booking rooms
    guest_name_1 = input("\nEnter guest name for booking Room 101: ")
    start_date_1 = input("Enter booking start date (YYYY-MM-DD): ")
    end_date_1 = input("Enter booking end date (YYYY-MM-DD): ")

    room_101.book_room(guest_name_1, start_date_1, end_date_1)

    guest_name_2 = input("\nEnter guest name for booking Room 102: ")
    start_date_2 = input("Enter booking start date (YYYY-MM-DD): ")
    end_date_2 = input("Enter booking end date (YYYY-MM-DD): ")

    room_102.book_room(guest_name_2, start_date_2, end_date_2)

    # Display updated room status
    room_101.display_booking_details()
    room_102.display_booking_details()

# Run the simulation
simulate_booking_system()
```

RESULT:

```
Room Number: 101
Room Type: Single
Price per Night: 1500.00
Status: Available

Room Number: 102
Room Type: Double
Price per Night: 2150.00
Status: Available

Enter guest name for booking Room 101: Debjit Ghosal
Enter booking start date (YYYY-MM-DD): 2024-12-20
```

```
Enter booking end date (YYYY-MM-DD): 2024-12-28
Room 101 successfully booked by Debjit Ghosal.

Enter guest name for booking Room 102: Saur Deshmukh
Enter booking start date (YYYY-MM-DD): 2024-10-12
Enter booking end date (YYYY-MM-DD): 2024-10-25
Room 102 successfully booked by Saur Deshmukh.

Room Number: 101
Room Type: Single
Price per Night: 1500.00
Status: Booked by Debjit Ghosal
Booking Dates:
 From: 2024-12-20 To: 2024-12-28

Room Number: 102
Room Type: Double
Price per Night: 2150.00
Status: Booked by Saur Deshmukh
Booking Dates:
 From: 2024-10-12 To: 2024-10-25
```

I have learnt to implement a class with attributes and simulate room bookings.
I have done this by method overidding.

Program 3	
PROBLEM STATEMENT:	3) Shape Area Calculation Problem Statement: Design a base class Shape with a method calculate_area(). Derive classes Circle, Rectangle, and Triangle from Shape. Each derived class should implement its own version of calculate_area().

PROGRAM:	# No need for abc module for a simple example
	class Shape:
	  def calculate_area(self):
	  raise NotImplementedError("Subclasses must implement this method")
	
	class Circle(Shape):
	  def __init__(self, radius):
	  self.radius = radius
	
	  def calculate_area(self):
	  return 3.14159 * self.radius * self.radius

```
class Rectangle(Shape):
  def __init__(self, width, height):
    self.width = width
    self.height = height

  def calculate_area(self):
    return self.width * self.height

class Triangle(Shape):
  def __init__(self, base, height):
    self.base = base
    self.height = height

  def calculate_area(self):
    return 0.5 * self.base * self.height

# Example usage
circle = Circle(5)
rectangle = Rectangle(4, 6)
triangle = Triangle(3, 8)

print(f"Circle area: {circle.calculate_area()}")
print(f"Rectangle area: {rectangle.calculate_area()}")
print(f"Triangle area: {triangle.calculate_area()}")
```

RESULT:
Circle area: 78.53975
Rectangle area: 24
Triangle area: 12.0

I have learnt to use method class and find area of various shapes.

Program 4	
PROBLEM STATEMENT:	4) Implement a simple online shopping cart system using classes and objects in Python? The implementation should models a basic online shopping cart system, encapsulating the data and behavior of products and the shopping cart itself.
PROGRAM:	class Product:

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

class ShoppingCart:
  def __init__(self):
    self.items = []

  def add_item(self, product):
    self.items.append(product)

  def remove_item(self, product):
    self.items.remove(product)

  def get_total_cost(self):
    total_cost = 0
    for item in self.items:
      total_cost += item.price
    return total_cost

  def display_items(self):
    for item in self.items:
      print(f"{item.name}: {item.price:.2f}")

# Example usage
laptop = Product("Laptop", 999.99)
mouse = Product("Mouse", 25.50)

cart = ShoppingCart()
cart.add_item(laptop)
cart.add_item(mouse)

cart.display_items()
print(f"Total Cost: {cart.get_total_cost():.2f}")

cart.remove_item(mouse)

cart.display_items()
```

RESULT:
Laptop: 999.99
Mouse: 25.50
Total Cost: 1025.49
Laptop: 999.99

I have learnt about the classes and objects in python by encapsulating the data.

CONCLUSION:	From these four coding projects, I've gained valuable insights into object-oriented programming principles and their practical applications. Firstly, the Employee Management System highlighted the importance of encapsulation and inheritance. By creating a base class with private attributes and deriving specific employee types, I learned how to manage and extend functionality in a structured manner. The `calculate_annual_salary` methods demonstrated how to leverage polymorphism for different employee types. Secondly, the Hotel Booking System provided practical experience with managing state and methods related to availability and booking. This project emphasized how to handle object interactions and maintain consistency in object state through methods like `check_availability` and `book_room`. The Shape Area Calculation exercise reinforced the concept of abstract classes and method overriding. Implementing different shapes with their own area calculations deepened my understanding of how base classes define a common interface while derived classes provide specific implementations. Finally, the Online Shopping Cart System illustrated the management of collections and basic operations like adding, removing, and calculating totals. This project highlighted the importance of encapsulating data and behaviors in classes to create a functional and maintainable system. Overall, these projects reinforced key object-oriented principles and their application in real-world scenarios, enhancing my ability to design and implement robust software solutions.