Assignment 7 Solution

Submitted by : Yameen Ali

Go to Github repository

Question 1

(A)

Create a Python class named Vehicle with attributes make, model, and year. Implement a method named drive() that prints "The vehicle is being driven." Create a subclass named Car that inherits from the Vehicle class and adds an attribute fuel_type. Demonstrate the use of single class inheritance by creating an instance of the Car class and invoking the drive() method.

In [78]:
```python
class Vehicle:
    def __init__(self,make,model,year):
        self.make = make
        self.model= model
        self.year = year

    def drive(self):
        print("The Vehicle is being Driven")

class Car(Vehicle):
    def __init__(self,make,model,year,fuel_type):
        super().__init__(make,model,year)
        self.fuel_type = fuel_type

my_car = Car('honda','city','2024','Gasoline')
my_car.drive()
```

```
The Vehicle is being Driven
```

(B)

Extend the previous example by introducing a new subclass named ElectricCar that inherits from the Car class. Add an attribute charge_time to represent the time required for charging the electric car. Demonstrate multilevel inheritance by creating an instance of the ElectricCar class and invoking the drive() method inherited from the Vehicle class.

In [79]:
```python
class ElectricCar(Car):
    # Subclass representing an electric car
    def __init__(self, make, model, year, fuel_type, charge_time):
        super().__init__(make, model, year, fuel_type)
        self.charge_time = charge_time  # Time required to charge the electric car

# Make an instance of the ElectricCar class
my_electric_car = ElectricCar("Tesla", "Model S", 2023, "Electricity", "8 hours")

# Invoking the drive() method
my_electric_car.drive()  # Method from Vehicle class
```

```
The Vehicle is being Driven
```

(C)

Create three separate classes named Car, Motorcycle, and Truck, each inheriting from the Vehicle class. Implement unique attributes and methods for each subclass, such as fuel_type for Car, engine_size for Motorcycle, and capacity for Truck. Finally, demonstrate multiclass inheritance by creating an object of a class that inherits from multiple parent classes and invoking its methods.

In [77]:
```python
class Vehicle:
    # Base class representing a vehicle
    def __init__(self, make, model, year):
        self.make = make    # Make of the vehicle
        self.model = model  # Model of the vehicle
        self.year = year    # Year of the vehicle

    def drive(self):
        # Method to simulate driving the vehicle
        print("The Vehicle is being Driven")

class Car(Vehicle):
    # Subclass representing a car
    def __init__(self, make, model, year, fuel_type):
        super().__init__(make, model, year)
        self.fuel_type = fuel_type  # Type of fuel the car uses

    def refuel(self):
        # Method to simulate refueling the car
        print("Refueling the car with: ", self.fuel_type)

class MotorCycle(Vehicle):
    # Subclass representing a motorcycle
    def __init__(self, make, model, year, engine_size):
        super().__init__(make, model, year)
        self.engine_size = engine_size  # Size of the motorcycle's engine

    def start(self):
        # Method to simulate starting the motorcycle
        print("Starting the motorCycle With Engine Size: ", self.engine_size)

class Truck(Vehicle):
    # Subclass representing a truck
    def __init__(self, make, model, year, capacity):
        super().__init__(make, model, year)
        self.capacity = capacity  # Capacity of the truck

    def load(self):
        # Method to simulate loading the truck
        print("Loading the truck with Capacity: ", self.capacity)


# Creating instances of each subclass separately
car = Car("Toyota", "Camry", 2022, "Gasoline")
motorcycle = MotorCycle("Honda", "CBR", 2021, "1000cc")
truck = Truck("Ford", "F-150", 2020, "2000kg")
loader = Truck("Ford", "F-150", 2024, "5000kg")

# Invoking methods
car.drive()  # Method from Vehicle class
car.refuel()  # Method from Car class

motorcycle.drive()  # Method from Vehicle class
motorcycle.start()  # Method from Motorcycle class

truck.drive()  # Method from Vehicle class
truck.load()   # Method from Truck class

loader.load()  # Method from Truck class
```

```
The Vehicle is being Driven
Refueling the car with:  Gasoline
The Vehicle is being Driven
Starting the motorCycle With Engine Size:  1000cc
The Vehicle is being Driven
Loading the truck with Capacity:  2000kg
Loading the truck with Capacity:  5000kg
```

Question 2

Designing a Library Management System

In [83]:

```
class Library:
    # Class representing a library
    def __init__(self, name, librarian):
        # Initialize the library with a name, a list of books, a list of members, and a librarian
        self.name = name  # Name of the library
        self.books = []   # List to store books in the library
        self.members = [] # List to store library members
        self.librarian = librarian  # Librarian responsible for managing the library

    def add_book(self, book):
        # Method to add a book to the library
        self.books.append(book)  # Add the book to the library's collection
        print(f"Book '{book.title}' added to the library.")  # Print confirmation message

    def remove_book(self, book):
        # Method to remove a book from the library
        if book in self.books:  # Check if the book is in the library
            self.books.remove(book)  # Remove the book from the library's collection
            print(f"Book '{book.title}' removed from the library.")  # Print confirmation message
        else:
            print("Book not found in the library.")  # Print error message if book not found

    def checkout_book(self, book, member):
        # Method to allow a member to check out a book
        if book in self.books and book.available:  # Check if the book is available in the library
            book.check_out()  # Mark the book as checked out
            print(f"Book '{book.title}' checked out by {member.name}.")  # Print confirmation message
        elif book in self.books and not book.available:  # Check if the book is already checked out
```

```python
            print(f"Book '{book.title}' is already checked out.")  # Print error message
        else:
            print("Book not found in the library.")  # Print error message if book not found

    def return_book(self, book):
        # Method to allow a member to return a book to the library
        if book in self.books and not book.available:  # Check if the book is checked out
            book.return_book()  # Mark the book as returned to the library
            print(f"Book '{book.title}' returned to the library.")  # Print confirmation message
        elif book in self.books and book.available:  # Check if the book is already available
            print(f"Book '{book.title}' is already in the library.")  # Print error message
        else:
            print("Book not found in the library.")  # Print error message if book not found

    def add_member(self, member):
        # Method to add a member to the library
        self.members.append(member)  # Add the member to the library's members
        print(f"Member '{member.name}' added to the library.")  # Print confirmation message

    def remove_member(self, member):
        # Method to remove a member from the library
        if member in self.members:  # Check if the member is in the library
            self.members.remove(member)  # Remove the member from the library's members
            print(f"Member '{member.name}' removed from the library.")  # Print confirmation message
        else:
            print("Member not found in the library.")  # Print error message if member not found

    def display_books(self):
        # Method to display all available books in the library
        if self.books:  # Check if there are books in the library
            print("Available Books in the Library:")  # Print heading
            for book in self.books:  # Iterate over each book
                # Print details of the book
                print(f"Title: {book.title}, Author: {book.author}, ISBN: {book.isbn}, Available: {book.availab
        else:
            print("No books available in the library.")  # Print message if no books are available

    def display_members(self):
        # Method to display all library members
        if self.members:  # Check if there are members in the library
            print("Library Members:")  # Print heading
            for member in self.members:  # Iterate over each member
                # Print details of the member
                print(f"Name: {member.name}, Member ID: {member.member_id}")
        else:
            print("No members in the library.")  # Print message if no members are available


# Class representing a book
class Book:
    def __init__(self, title, author, isbn):
        # Initialize the book with a title, author, ISBN, and availability
        self.title = title      # Title of the book
        self.author = author    # Author of the book
        self.isbn = isbn        # ISBN of the book
        self.available = True  # Availability of the book

    # Method to mark the book as checked out
    def check_out(self):
        self.available = False  # Set the availability of the book to False when checked out

    # Method to mark the book as returned to the library
    def return_book(self):
        self.available = True  # Set the availability of the book to True when returned

# Define a class representing a Member
class Member:
    def __init__(self, name, member_id):
        self.name = name        # Name of the member
        self.member_id = member_id  # Member ID

# Define a class representing a Librarian
class Librarian:
    def __init__(self, name):
        self.name = name  # Name of the librarian

    def manage_library(self, library):
        # Here you can add logic for managing the library, such as adding/removing books and members
        pass

def display_menu():
    # Function to display the main menu of the library management system
    print("Welcome to the Library Management System")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Checkout Book")
    print("4. Return Book")
    print("5. Add Member")
    print("6. Remove Member")
    print("7. Display Books")
    print("8. Display Members")
    print("9. Quit")

def main():
    # Main function to run the library management system
    librarian = Librarian("John Doe")  # Create a librarian object
    library = Library("Central Library", librarian)  # Create a library object

    while True:  # Run the main loop until the user chooses to quit
        display_menu()  # Display the main menu
        choice = input("Enter your choice: ")  # Get user input for choice

        if choice == '1':
            # Option to add a book
            title = input("Enter book title: ")
            author = input("Enter author: ")
            isbn = input("Enter ISBN: ")
            book = Book(title, author, isbn)
            library.add_book(book)

        elif choice == '2':
            # Option to remove a book
            title = input("Enter book title to remove: ")
            for book in library.books:
                if book.title == title:
                    library.remove_book(book)
                    break
            else:
                print("Book not found in the library.")

        elif choice == '3':
            # Option to checkout a book
            title = input("Enter book title to checkout: ")
            member_id = input("Enter member ID: ")
            for book in library.books:
                if book.title == title:
                    for member in library.members:
                        if member.member_id == member_id:
                            library.checkout_book(book, member)
                            break
                    else:
                        print("Member not found in the library.")
                    break
            else:
                print("Book not found in the library or already checked out.")

        elif choice == '4':
            # Option to return a book
            title = input("Enter book title to return: ")
            for book in library.books:
                if book.title == title:
                    library.return_book(book)
                    break
            else:
                print("Book not found in the library or already available.")

        elif choice == '5':
            # Option to add a member
            name = input("Enter member name: ")
            member_id = input("Enter member ID: ")
            member = Member(name, member_id)
            library.add_member(member)

        elif choice == '6':
            # Option to remove a member
            member_id = input("Enter member ID to remove: ")
            for member in library.members:
                if member.member_id == member_id:
                    library.remove_member(member)
                    break
            else:
                print("Member not found in the library.")

        elif choice == '7':
            # Option to display all books in the library
            library.display_books()
        elif choice == '8':
            # Option to display all members in the library
            library.display_members()

        elif choice == '9':
            # Option to quit the program
            print("Exiting...")
            break

        else:
            print("Invalid choice. Please try again.")  # Print error message for invalid choice


if __name__ == "__main__":
    main()  # Run the main function when the script is executed
```

```
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Book 'Operating System' added to the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Available Books in the Library:
Title: Operating System, Author: Yameen, ISBN: 01, Available: True
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Member 'Yameen Ali' added to the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Library Members:
Name: Yameen Ali, Member ID: 001
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Book 'Operating System' checked out by Yameen Ali.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
```

```
Available Books in the Library:
Title: Operating System, Author: Yameen, ISBN: 01, Available: False
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Book 'Operating System' returned to the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Book 'Operating System' removed from the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Member 'Yameen Ali' removed from the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
No books available in the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
No members in the library.
Welcome to the Library Management System
1. Add Book
2. Remove Book
3. Checkout Book
4. Return Book
5. Add Member
6. Remove Member
7. Display Books
8. Display Members
9. Quit
Exiting...
```

Question 3

Developing a Payment Processing Module for an E-commerce Platform

In [114…

```
#  abc module is being used to define abstract base classes and methods through the ABC class and the abstractm
from abc import ABC, abstractmethod

# Abstract base class
class PaymentMethod(ABC):

    @abstractmethod
```

```python
    def process_payment(self):
        pass

    @abstractmethod
    def validate_payment(self):
        pass

# subclass representing Credit Card Payment Method
class CreditCard(PaymentMethod):
    def __init__(self,card_number,expiry_date,cvv):
        self.card_number = card_number
        self.expiry_date = expiry_date
        self.cvv = cvv

    def process_payment(self):
        print("Processing Credit Card Payment...")

    def validate_payment(self):
        # Dummy validation
        return len(self.card_number) == 16 and self.expiry_date and self.cvv

# subclass representing Paypal Payment Method
class PayPal(PaymentMethod):
    def __init__(self,email,password):
        self.email = email
        self.password = password

    def process_payment(self):
        print("Processing PayPal Payment...")

    def validate_payment(self):
        return self.email and self.password

# subclass representing Crypto Currency Payment Method
class CryptoCurrency(PaymentMethod):
    def __init__(self,wallet_address):
        self.wallet_address = wallet_address

    def process_payment(self):
        print("Processing Crypto Currency payment...")

    def validate_payment(self):
        return len(self.wallet_address) >= 20

def payment_method_testing():
    credit_card = CreditCard("2130123912311230","12/23","567")
    paypal  = PayPal("yameenshah012@gmail.com","123456789")
    crypto_currency = CryptoCurrency("131Fhask2131201239")

    payment_methods = [credit_card,paypal,crypto_currency]
    for method in payment_methods:
        method.process_payment()
        if method.validate_payment():
            print("Payment Validation Successful.")

        else:
            print("Payment Validation Failed")

if __name__ == "__main__":
    payment_method_testing()
```

```
Processing Credit Card Payment...
Payment Validation Successful.
Processing PayPal Payment...
Payment Validation Successful.
Processing Crypto Currency payment...
Payment Validation Failed
```