```
'''1. Scenario: Library Management System
Creating a simple library management system where:
• Library handles book details.
• Member handles member details.
• LibraryManagement combines the features of both Library
and Member
and allows borrowing books.'''
class Library:
    def __init__(self):
        self.books = {}

    def add_book(self, title, author, copies):
        if title not in self.books:
            self.books[title] = {'author': author, 'copies': copies}
        else:
            self.books[title]['copies'] += copies

    def display_books(self):
        if self.books:
            print("Books in the library:")
            for title, details in self.books.items():
                print(f"Title: {title}, Author: {details['author']}, Copies
Available: {details['copies']}")
        else:
            print("No books available in the library.")

    def check_availability(self, title):
        return self.books.get(title, {}).get('copies', 0)


class Member:
    def __init__(self, member_id, name):
        self.member_id = member_id
        self.name = name
        self.borrowed_books = []
```

```
    def borrow_book(self, book_title):
        self.borrowed_books.append(book_title)

    def return_book(self, book_title):
        if book_title in self.borrowed_books:
            self.borrowed_books.remove(book_title)

    def display_borrowed_books(self):
        if self.borrowed_books:
            print(f"{self.name} has borrowed:")
            for book in self.borrowed_books:
                print(f"- {book}")
        else:
            print(f"{self.name} has no borrowed books.")
'''2. Scenario: Food Delivery System
Create a system where:
• Restaurant handles the menu and food preparation.
• Delivery manages the delivery details and rider information.
• Order combines both Restaurant and Delivery to process
food orders
and manage delivery logistics'''
class Restaurant:
    def __init__(self):
        self.menu = {}
    def add_food_item(self, name, price, preparation_time):
        self.menu[name] = {'price': price, 'preparation_time':
preparation_time}
    def display_menu(self):
        if self.menu:
            print("Menu:")
            for name, details in self.menu.items():
                print(f"{name}: ${details['price']} (Preparation Time:
{details['preparation_time']} minutes)")
        else:
            print("Menu is empty.")
```

```
    def prepare_food(self, food_item):
        if food_item in self.menu:
            print(f"Preparing {food_item}... (Time:
{self.menu[food_item]['preparation_time']} minutes)")
            return self.menu[food_item]['preparation_time']
        else:
            print(f"{food_item} is not available on the menu.")
            return 0
class Delivery:
    def __init__(self):
        self.riders = {}
    def add_rider(self, rider_id, name):
        self.riders[rider_id] = {'name': name, 'status': 'Available'}
    def assign_rider(self, rider_id, order_id):
        if rider_id in self.riders:
            self.riders[rider_id]['status'] = f"Delivering order
{order_id}"
            print(f"Rider {self.riders[rider_id]['name']} is assigned to
order {order_id}.")
            return rider_id
        else:
            print(f"Rider {rider_id} not found.")
            return None
    def track_delivery(self, rider_id):
        if rider_id in self.riders:
            print(f"Rider {self.riders[rider_id]['name']} status:
{self.riders[rider_id]['status']}")
        else:
            print("Rider not found.")
class Order:
    def __init__(self, restaurant, delivery):
        self.restaurant = restaurant
        self.delivery = delivery
        self.orders = {}
    def create_order(self, order_id, food_items, rider_id):
```

```
        if not food_items:
            print("No food items selected.")
            return
        preparation_time = 0
        for food_item in food_items:
            preparation_time +=
self.restaurant.prepare_food(food_item)
        rider = self.delivery.assign_rider(rider_id, order_id)
        if rider is not None:
            print(f"Order {order_id} is being processed. Total
preparation time: {preparation_time} minutes.")
            self.orders[order_id] = {'food_items': food_items,
'rider_id': rider, 'status': 'Preparing'}
            self.delivery.track_delivery(rider_id)
        else:
            print("Delivery assignment failed.")
    def complete_order(self, order_id):
        if order_id in self.orders:
            self.orders[order_id]['status'] = 'Delivered'
            rider_id = self.orders[order_id]['rider_id']
            self.delivery.riders[rider_id]['status'] = 'Available'
            print(f"Order {order_id} has been delivered.")
            self.delivery.track_delivery(rider_id)
        else:
            print(f"Order {order_id} not found.")
restaurant = Restaurant()
restaurant.add_food_item("Pizza", 12.99, 15)
restaurant.add_food_item("Burger", 8.99, 10)
restaurant.add_food_item("Pasta", 10.49, 12)
delivery = Delivery()
delivery.add_rider(1, "John")
delivery.add_rider(2, "Alice")
order_system = Order(restaurant, delivery)
restaurant.display_menu()
order_system.create_order(101, ["Pizza", "Pasta"], 1)
```

```
delivery.track_delivery(1)
order_system.complete_order(101)
delivery.track_delivery(1)
```

```
class LibraryManagement:
    def __init__(self, library):
        self.library = library
        self.members = {}

    def add_member(self, member):
        self.members[member.member_id] = member

    def borrow_book(self, member_id, book_title):
        if member_id not in self.members:
            print(f"No member found with ID: {member_id}")
            return

        member = self.members[member_id]
        if self.library.check_availability(book_title) > 0:
            member.borrow_book(book_title)
            self.library.books[book_title]['copies'] -= 1
            print(f"{member.name} has borrowed '{book_title}'.")
        else:
            print(f"Sorry, '{book_title}' is not available in the library.")

    def return_book(self, member_id, book_title):
        if member_id not in self.members:
            print(f"No member found with ID: {member_id}")
            return

        member = self.members[member_id]
        if book_title in member.borrowed_books:
            member.return_book(book_title)
            self.library.books[book_title]['copies'] += 1
            print(f"{member.name} has returned '{book_title}'.")
        else:
            print(f"{member.name} has not borrowed '{book_title}'.")

    def display_all_books(self):
        self.library.display_books()

    def display_member_books(self, member_id):
        if member_id not in self.members:
            print(f"No member found with ID: {member_id}")
            return
        self.members[member_id].display_borrowed_books()
```

```
library = Library()
library.add_book("The Great Gatsby", "F. Scott Fitzgerald", 5)
library.add_book("1984", "George Orwell", 3)

member1 = Member(1, "Krish")
member2 = Member(2, "Harshu")

library_management = LibraryManagement(library)
library_management.add_member(member1)
library_management.add_member(member2)

library_management.display_all_books()
library_management.borrow_book(1, "The Great Gatsby")
library_management.display_member_books(1)
library_management.display_all_books()

library_management.return_book(1, "The Great Gatsby")
library_management.display_member_books(1)
library_management.display_all_books()
```