PIYUSH KUMAR MISHRA

230957212

ROLL N0:70

WEEK 6:

Exercise 1:

You are tasked with designing a Library Management System for a local library. The library has both EBooks (digital format) and Printed Books (physical copies), and the system should allow members (both students and teachers) to borrow books. The library also has a librarian who manages the addition and removal of books.

Your system should include the following features:

1. Book Management:

o Books can be either EBooks or Printed Books.

o Each book should have a title, author, and ISBN.

o EBooks should have a file format, while Printed Books should have a page count.

2. Member Management:

o The library has members who can either be students or teachers.

o Each member has a name and member ID.

o Members should be able to borrow books.

3. Librarian Management:

o A librarian can add or remove books from the library.

o A librarian can be both a student and a teacher.

4. Library Operations:

o The system should allow the librarian to:

- **Add new books to the library.**
- **Remove books from the library using their ISBN.**
- **Search for books by title or author.**

Requirements:

Using Python and Object-Oriented Programming principles, implement the following:

1. Create a class hierarchy to represent books (including EBooks and Printed Books).

2. Create a class hierarchy to represent members (students and teachers).

3. Implement the functionalities for adding, removing, and searching for books.

4. Demonstrate the following types of inheritance:

o Single Inheritance for books.

o Multiple Inheritance for the librarian, who is both a student and a teacher.

o Hierarchical Inheritance for members (students and teachers).

Tasks:

1. Book Management:

o Define a base class Book with attributes for title, author, and ISBN.

o Define a subclass EBook that adds the attribute for file format.

o Define another subclass PrintedBook that adds the attribute for page count.

2. Member and Librarian Management:

o Define a base class Member with attributes for name and member ID.

o Define two subclasses: Student and Teacher, which inherit from Member.

o Create a Librarian class that inherits from both Student and Teacher (multiple inheritance).

3. Library Class:

o Implement a Library class to manage the collection of books.

o Add methods to the Library class to:

- **Add new books.**
- **Remove a book by its ISBN.**
- **Search for books by title or author.**

4. Demonstration:

o Instantiate a library and add books (both EBooks and Printed Books) to it.

o Demonstrate searching for books using keywords.

o Show how a librarian can add and remove books from the system.

```
class Book:
    def __init__(self, title, author, isbn):
        self.title = title
        self.author = author
        self.isbn = isbn

class EBook(Book):
    def __init__(self, title, author, isbn, file_format):
        super().__init__(title, author, isbn)
        self.file_format = file_format

class PrintedBook(Book):
    def __init__(self, title, author, isbn, page_count):
        super().__init__(title, author, isbn)
        self.page_count = page_count
class Member:
    def __init__(self, name, member_id):
        self.name = name
```

```
        self.member_id = member_id


class Student(Member):
    pass


class Teacher(Member):
    pass


class Librarian(Student, Teacher):
    pass
class Library:
    def __init__(self):
        self.books = []


    def add_book(self, book):
        self.books.append(book)


    def remove_book(self, isbn):
        for book in self.books:
            if book.isbn == isbn:
                self.books.remove(book)
                print(f"Book with ISBN {isbn} removed successfully.")
                return
        print(f"Book with ISBN {isbn} not found.")


    def search_book(self, keyword):
        results = []
```

```
        for book in self.books:
            if keyword in book.title or keyword in book.author:
                results.append(book)
        return results
def main():
    library = Library()

    while True:
        print("Library Management System")
        print("1. Add Book")
        print("2. Remove Book")
        print("3. Search Book")
        print("4. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            title = input("Enter book title: ")
            author = input("Enter book author: ")
            isbn = input("Enter book ISBN: ")
            book_type = input("Enter book type (EBook or PrintedBook): ")

            if book_type == "EBook":
                file_format = input("Enter file format: ")
                book = EBook(title, author, isbn, file_format)
            elif book_type == "PrintedBook":
                page_count = int(input("Enter page count: "))
```

```
            book = PrintedBook(title, author, isbn, page_count)

        library.add_book(book)
        print("Book added successfully.")

    elif choice == "2":
        isbn = input("Enter ISBN of book to remove: ")
        library.remove_book(isbn)

    elif choice == "3":
        keyword = input("Enter keyword to search: ")
        results = library.search_book(keyword)

        if results:
            print("Search results:")
            for book in results:
                print(f"Title: {book.title}, Author: {book.author}, ISBN: {book.isbn}")
        else:
            print("No books found.")

    elif choice == "4":
        break

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

if __name__ == "__main__":
```

main()

OUTPUT:

Library Management System

1. Add Book

2. Remove Book

3. Search Book

4. Exit

Enter your choice: 1

Enter book title: x

Enter book author: y

Enter book ISBN: z

Enter book type (EBook or PrintedBook): EBook

Enter file format: pdf

Book added successfully.

Library Management System

1. Add Book

2. Remove Book

3. Search Book

4. Exit

Enter your choice:

↑↓ for history. Search history with c-↑/c-↓

Exercise 2: Advanced E-Commerce System Utilizing Polymorphism

A rapidly growing online retail company is looking to upgrade its E-Commerce System.

They need the system to manage various types of products, allow users to add them to their

shopping carts, apply discounts, and handle different payment methods. To make the system robust, flexible, and scalable, you decide to use Object-Oriented Programming (OOP) principles.

Objectives:

1. Product Management:

o Products in the system belong to different categories such as Electronics and Clothing. Each product category has its own discount logic.

o Discounts should be applied based on product types, demonstrating method overriding.

2. Shopping Cart Management:

o Users should be able to add multiple items to their shopping carts and see the total cost after discounts.

o The system should allow merging two shopping carts using operator overloading.

3. Payment Processing:

o Customers should be able to process payments using various methods (e.g., credit card, PayPal). Even though Python does not natively support method overloading, it should be simulated to handle different payment methods efficiently.

Functional Requirements:

1. Products:

o Implement a base Product class that represents general products, containing attributes like name and price.

o Create derived classes such as Electronics and Clothing that override the base class method for calculating product discounts.

2. Shopping Cart:

o Implement a ShoppingCart class that can hold a collection of products.

o Overload the + operator to merge two shopping carts into one.

3. Payment Processing:

o Implement a PaymentProcessor class that simulates method overloading to handle different payment methods (e.g., credit card and PayPal) using variable arguments.

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

    def calculate_discount(self):
        return self.price * 0.1  # 10% discount as a placeholder

class Electronics(Product):
    def calculate_discount(self):
        return self.price * 0.15  # 15% discount for Electronics

class Clothing(Product):
    def calculate_discount(self):
        return self.price * 0.05  # 5% discount for Clothing
class ShoppingCart:
    def __init__(self, products=None):
        self.products = products or []

    def add_product(self, product):
        self.products.append(product)
```

```
    def calculate_total(self):
        total = sum(product.price for product in self.products)
        discount = sum(product.calculate_discount() for product in self.products)
        return total - discount

    def __add__(self, other):
        combined_cart = ShoppingCart()
        combined_cart.products = self.products + other.products
        return combined_cart
class PaymentProcessor:
    def process_payment(self, payment_method, *args, **kwargs):
        if payment_method == "credit_card":
            self.process_credit_card_payment(*args, **kwargs)
        elif payment_method == "paypal":
            self.process_paypal_payment(*args, **kwargs)
        else:
            raise ValueError(f"Unsupported payment method: {payment_method}")

    def process_credit_card_payment(self, amount, card_number, expiration_date, cvv):
        # Implement credit card payment processing
        pass

    def process_paypal_payment(self, amount, paypal_email, paypal_password):
        # Implement PayPal payment processing
        pass
def main():
    electronics = Electronics("Smartphone", 500)
```

```
    clothing = Clothing("T-Shirt", 20)

    cart1 = ShoppingCart()
    cart1.add_product(electronics)
    cart1.add_product(clothing)

    cart2 = ShoppingCart()
    cart2.add_product(electronics)

    print("Cart 1 total:", cart1.calculate_total())
    print("Cart 2 total:", cart2.calculate_total())

    merged_cart = cart1 + cart2
    print("Merged cart total:", merged_cart.calculate_total())

    payment_processor = PaymentProcessor()
    payment_processor.process_payment("credit_card", 500, "1234567890123456", "123", "2025")
    payment_processor.process_payment("paypal", 500, "john.doe@example.com", "secret_password")

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

OUTPUT:

Cart 1 total: 444.0

Cart 2 total: 425.0

Merged cart total: 869.0