1. Define a class Book with attributes title, author, and pages. Create an object of this class and print its attributes..

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

# Create an object of the Book class
my_book = Book("1984", "George Orwell", 328)

# Print the attributes of the book
print("Title:", my_book.title)
print("Author:", my_book.author)
print("Pages:", my_book.pages)
```

```
Title: 1984
Author: George Orwell
Pages: 328
```

2. Add a method display_info to the Book class that prints the book details using the self object. Call this method for the created object..

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

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)

# Create an object of the Book class
my_book = Book("1984", "George Orwell", 328)

# Call the display_info method for the created object
my_book.display_info()
```

```
Book Title: 1984
Author: George Orwell
Pages: 328
```

3. Add a class variable book_count to the Book class that keeps track of the number of book objects created. Update this variable in the class constructor and print the total number of books after creating multiple objects.

s.

```python
class Book:
    # Class variable to keep track of the number of Book objects
created
    book_count = 0

    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
        # Increment the class variable each time a Book object is
created
        Book.book_count += 1

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)

# Create multiple objects of the Book class
book1 = Book("1984", "George Orwell", 328)
book2 = Book("To Kill a Mockingbird", "Harper Lee", 281)
book3 = Book("The Great Gatsby", "F. Scott Fitzgerald", 180)

# Call the display_info method for each created object
book1.display_info()
book2.display_info()
book3.display_info()

# Print the total number of books created
print("Total number of books created:", Book.book_count)
```

```
Book Title: 1984
Author: George Orwell
Pages: 328
Book Title: To Kill a Mockingbird
Author: Harper Lee
Pages: 281
Book Title: The Great Gatsby
Author: F. Scott Fitzgerald
Pages: 180
Total number of books created: 3
```

4. Modify the Book class to include a private member _isbn (underscore prefix) and a method set_isbn to set its value. Demonstrate the usage of public and private members by creating an object and trying to access these members directly.

y.

```python
class Book:
    # Class variable to keep track of the number of Book objects
created
    book_count = 0

    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
        self._isbn = None  # Private member
        Book.book_count += 1

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)
        print("ISBN:", self._isbn)  # Display ISBN

    def set_isbn(self, isbn):
        self._isbn = isbn  # Method to set the private member

# Create an object of the Book class
```

```
my_book = Book("1984", "George Orwell", 328)

# Set the ISBN using the setter method
my_book.set_isbn("978-0451524935")

# Call the display_info method to show book details
my_book.display_info()

# Accessing public members
print("Public Member - Title:", my_book.title)
print("Public Member - Author:", my_book.author)

# Attempting to access the private member directly (not recommended)
try:
    print("Private Member - ISBN:", my_book._isbn)
except AttributeError as e:
    print("Error accessing private member:", e)
```

```
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935
Public Member - Title: 1984
Public Member - Author: George Orwell
Private Member - ISBN: 978-0451524935
```

5. Create a subclass EBook that inherits from the Book class and adds an attribute file_size. Override the display_info method to include file size information. Demonstrate polymorphism by calling display_info on both Book and EBook objects.

s.

```
class Book:
    # Class variable to keep track of the number of Book objects
created
    book_count = 0

    def __init__(self, title, author, pages):
        self.title = title
```

```
        self.author = author
        self.pages = pages
        self._isbn = None  # Private member
        Book.book_count += 1

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)
        print("ISBN:", self._isbn)  # Display ISBN

    def set_isbn(self, isbn):
        self._isbn = isbn  # Method to set the private member


class EBook(Book):
    def __init__(self, title, author, pages, file_size):
        super().__init__(title, author, pages)  # Call the parent
constructor
        self.file_size = file_size  # EBook-specific attribute

    def display_info(self):
        # Call the parent display_info method
        super().display_info()
        print("File Size:", self.file_size, "MB")  # Display file size
information


# Create a Book object
my_book = Book("1984", "George Orwell", 328)
my_book.set_isbn("978-0451524935")

# Create an EBook object
my_ebook = EBook("Digital Fortress", "Dan Brown", 384, 2.5)
my_ebook.set_isbn("978-0-345-48625-3")

# Call the display_info method on both Book and EBook objects
print("\nDisplaying Book Information:")
my_book.display_info()

print("\nDisplaying EBook Information:")
my_ebook.display_info()


Displaying Book Information:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935

Displaying EBook Information:
```

```
Book Title: Digital Fortress
Author: Dan Brown
Pages: 384
ISBN: 978-0-345-48625-3
File Size: 2.5 MB
```

6. Define another class AudioBook with attributes title, author, duration and a method display_info. Write a function print_book_info that takes any book object (Book,EBook, or AudioBook) and calls its display_info method, demonstrating polymorphism.sm.

```
class Book:
    # Class variable to keep track of the number of Book objects
created
    book_count = 0

    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
        self._isbn = None  # Private member
        Book.book_count += 1

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)
        print("ISBN:", self._isbn)  # Display ISBN

    def set_isbn(self, isbn):
        self._isbn = isbn  # Method to set the private member


class EBook(Book):
    def __init__(self, title, author, pages, file_size):
        super().__init__(title, author, pages)  # Call the parent
constructor
        self.file_size = file_size  # EBook-specific attribute

    def display_info(self):
        # Call the parent display_info method
```

```
        super().display_info()
        print("File Size:", self.file_size, "MB")  # Display file size
information


class AudioBook:
    def __init__(self, title, author, duration):
        self.title = title
        self.author = author
        self.duration = duration  # Duration in minutes

    def display_info(self):
        print("Audio Book Title:", self.title)
        print("Author:", self.author)
        print("Duration:", self.duration, "minutes")


# Function to print book information
def print_book_info(book):
    book.display_info()  # Call the display_info method for any book
object


# Create objects of different book types
my_book = Book("1984", "George Orwell", 328)
my_book.set_isbn("978-0451524935")

my_ebook = EBook("Digital Fortress", "Dan Brown", 384, 2.5)
my_ebook.set_isbn("978-0-345-48625-3")

my_audiobook = AudioBook("Becoming", "Michelle Obama", 120)

# Print information for each book type using the print_book_info
function
print("\nDisplaying Book Information:")
print_book_info(my_book)

print("\nDisplaying EBook Information:")
print_book_info(my_ebook)

print("\nDisplaying AudioBook Information:")
print_book_info(my_audiobook)


Displaying Book Information:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935

Displaying EBook Information:
```

```
Book Title: Digital Fortress
Author: Dan Brown
Pages: 384
ISBN: 978-0-345-48625-3
File Size: 2.5 MB

Displaying AudioBook Information:
Audio Book Title: Becoming
Author: Michelle Obama
Duration: 120 minutes
```

7. Create a class Library that contains a list of Book objects. Add methods to add a book to the library and display all books in the library. Demonstrate containership by creating a library object and adding book objects to it.

t.

```python
class Book:
    # Class variable to keep track of the number of Book objects
created
    book_count = 0

    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
        self._isbn = None  # Private member
        Book.book_count += 1

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)
        print("ISBN:", self._isbn)  # Display ISBN

    def set_isbn(self, isbn):
        self._isbn = isbn  # Method to set the private member


class Library:
    def __init__(self):
```

```
        self.books = []  # List to store Book objects

    def add_book(self, book):
        """Add a Book object to the library."""
        self.books.append(book)

    def display_all_books(self):
        """Display all books in the library."""
        if not self.books:
            print("No books in the library.")
            return

        print("Books in the Library:")
        for book in self.books:
            book.display_info()
            print()  # Add a newline for better readability


# Create some Book objects
book1 = Book("1984", "George Orwell", 328)
book1.set_isbn("978-0451524935")

book2 = Book("To Kill a Mockingbird", "Harper Lee", 281)
book2.set_isbn("978-0061120084")

book3 = Book("The Great Gatsby", "F. Scott Fitzgerald", 180)
book3.set_isbn("978-0743273565")

# Create a Library object
my_library = Library()

# Add books to the library
my_library.add_book(book1)
my_library.add_book(book2)
my_library.add_book(book3)

# Display all books in the library
my_library.display_all_books()
```

```
Books in the Library:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935

Book Title: To Kill a Mockingbird
Author: Harper Lee
Pages: 281
ISBN: 978-0061120084

Book Title: The Great Gatsby
```

```
Author: F. Scott Fitzgerald
Pages: 180
ISBN: 978-0743273565
```

8. Explain the concepts of reusability and delegation in OOP with examples. Modify the Library class to include a method find_book_by_title that delegates the responsibility of searching for a book to a helper method _search.h.

```python
class Book:
    def __init__(self, title, author, pages):
        self.title = title
        self.author = author
        self.pages = pages
        self._isbn = None  # Private member

    def display_info(self):
        print("Book Title:", self.title)
        print("Author:", self.author)
        print("Pages:", self.pages)
        print("ISBN:", self._isbn)  # Display ISBN

    def set_isbn(self, isbn):
        self._isbn = isbn  # Method to set the private member


class Library:
    def __init__(self):
        self.books = []  # List to store Book objects

    def add_book(self, book):
        """Add a Book object to the library."""
        self.books.append(book)

    def display_all_books(self):
        """Display all books in the library."""
        if not self.books:
            print("No books in the library.")
            return
```

```
        print("Books in the Library:")
        for book in self.books:
            book.display_info()
            print()  # Add a newline for better readability

    def find_book_by_title(self, title):
        """Find a book by its title."""
        found_book = self._search(title)
        if found_book:
            print("Book found:")
            found_book.display_info()
        else:
            print(f"No book found with the title: '{title}'")

    def _search(self, title):
        """Helper method to search for a book by its title."""
        for book in self.books:
            if book.title.lower() == title.lower():  # Case
insensitive search
                return book
        return None


# Create some Book objects
book1 = Book("1984", "George Orwell", 328)
book1.set_isbn("978-0451524935")

book2 = Book("To Kill a Mockingbird", "Harper Lee", 281)
book2.set_isbn("978-0061120084")

# Create a Library object
my_library = Library()

# Add books to the library
my_library.add_book(book1)
my_library.add_book(book2)

# Display all books in the library
my_library.display_all_books()

# Find a book by title
my_library.find_book_by_title("1984")  # Should find the book
my_library.find_book_by_title("Moby Dick")  # Should not find the book
```

```
Books in the Library:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935
```

```
Book Title: To Kill a Mockingbird
Author: Harper Lee
Pages: 281
ISBN: 978-0061120084

Book found:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935
No book found with the title: 'Moby Dick'
```

9. Modify the Book class to make title, author, and pages private. Provide public getter and setter methods to access and modify these attributes, demonstrating data abstraction and encapsulation.n.

```python
class Book:
    # Class variable to keep track of the number of Book objects
created
    book_count = 0

    def __init__(self, title, author, pages):
        self.__title = title    # Private member
        self.__author = author  # Private member
        self.__pages = pages    # Private member
        self.__isbn = None      # Private member
        Book.book_count += 1

    # Getter for title
    def get_title(self):
        return self.__title

    # Setter for title
    def set_title(self, title):
        self.__title = title

    # Getter for author
    def get_author(self):
        return self.__author

    # Setter for author
    def set_author(self, author):
```

```
        self.__author = author

    # Getter for pages
    def get_pages(self):
        return self.__pages

    # Setter for pages
    def set_pages(self, pages):
        if pages > 0:  # Validation to ensure pages is positive
            self.__pages = pages
        else:
            print("Pages must be a positive integer.")

    # Getter for ISBN
    def get_isbn(self):
        return self.__isbn

    # Setter for ISBN
    def set_isbn(self, isbn):
        self.__isbn = isbn  # Method to set the private member

    def display_info(self):
        print("Book Title:", self.__title)
        print("Author:", self.__author)
        print("Pages:", self.__pages)
        print("ISBN:", self.__isbn)  # Display ISBN


class Library:
    def __init__(self):
        self.books = []  # List to store Book objects

    def add_book(self, book):
        """Add a Book object to the library."""
        self.books.append(book)

    def display_all_books(self):
        """Display all books in the library."""
        if not self.books:
            print("No books in the library.")
            return

        print("Books in the Library:")
        for book in self.books:
            book.display_info()
            print()  # Add a newline for better readability

    def find_book_by_title(self, title):
        """Find a book by its title."""
        found_book = self._search(title)
```

```
        if found_book:
            print("Book found:")
            found_book.display_info()
        else:
            print(f"No book found with the title: '{title}'")

    def _search(self, title):
        """Helper method to search for a book by its title."""
        for book in self.books:
            if book.get_title().lower() == title.lower():  # Case
insensitive search
                return book
        return None


# Create some Book objects
book1 = Book("1984", "George Orwell", 328)
book1.set_isbn("978-0451524935")

book2 = Book("To Kill a Mockingbird", "Harper Lee", 281)
book2.set_isbn("978-0061120084")

# Create a Library object
my_library = Library()

# Add books to the library
my_library.add_book(book1)
my_library.add_book(book2)

# Display all books in the library
my_library.display_all_books()

# Find a book by title
my_library.find_book_by_title("1984")  # Should find the book
my_library.find_book_by_title("Moby Dick")  # Should not find the book

# Modify book attributes using setter methods
book1.set_pages(350)  # Update pages
book1.set_title("Nineteen Eighty-Four")  # Update title

print("\nUpdated Book Information:")
book1.display_info()
```

```
Books in the Library:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935

Book Title: To Kill a Mockingbird
Author: Harper Lee
```

```
Pages: 281
ISBN: 978-0061120084

Book found:
Book Title: 1984
Author: George Orwell
Pages: 328
ISBN: 978-0451524935
No book found with the title: 'Moby Dick'

Updated Book Information:
Book Title: Nineteen Eighty-Four
Author: George Orwell
Pages: 350
ISBN: 978-0451524935
```