

**Assignment\_04**

**May 27, 2025**

**CSC241 – Object Oriented Programming**

*Muhammad Rizwan Shafiq*

*Reg# SP24-BCS-069*

*Section: BCS-3B*

**Assignment 04 (CLO-3)**

# Create a class Book that has name(String), publisher (String) and an author (Person).

## Write five objects of Book Class in a file named “BookStore”.

## Write a function that displays all Books present in file “BookStore”.

## Write a function that asks the user for the name of a Book and searches the record against that book in the file “BookStore”.

import java.util.Scanner;

import java.io.Serializable;

import java.io.FileOutputStream;

import java.io.FileInputStream;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

class Person implements Serializable {

  private String name;

  public Person(String name) {

    this.name = name;

  }

  public String getName() {

    return name;

  }

}

class Book implements Serializable {

  private String name;

  private String publisher;

  private Person author;

  public Book(String name, String publisher, Person author) {

    this.name = name;

    this.publisher = publisher;

    this.author = author;

  }

  public String getName() {

    return name;

  }

  public String getPublisher() {

    return publisher;

  }

  public Person getAuthor() {

    return author;

  }

  public void display() {

    System.out.println("Book Name: " + name);

    System.out.println("Publisher: " + publisher);

    System.out.println("Author: " + author.getName());

    System.out.println("------------------------");

  }

}

public class Question\_01 {

  public static void writeBooksToFile(Book[] books) {

    try {

      ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("BookStore.ser"));

      for (Book b : books) {

        oos.writeObject(b);

      }

      oos.close();

      System.out.println("Books written to BookStore.ser successfully.");

    } catch (Exception e) {

      e.printStackTrace();

    }

  }

  public static void displayAllBooks() {

    try {

      ObjectInputStream ois = new ObjectInputStream(new FileInputStream("BookStore.ser"));

      while (true) {

        try {

          Book book = (Book) ois.readObject();

          book.display();

        } catch (Exception e) {

          break; // End of file

        }

      }

      ois.close();

    } catch (Exception e) {

      e.printStackTrace();

    }

  }

  public static void searchBookByName() {

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter Book Name to Search: ");

    String searchName = sc.nextLine();

    boolean found = false;

    try {

      ObjectInputStream ois = new ObjectInputStream(new FileInputStream("BookStore.ser"));

      while (true) {

        try {

          Book book = (Book) ois.readObject();

          if (book.getName().equalsIgnoreCase(searchName)) {

            System.out.println("Book Found:");

            book.display();

            found = true;

            break;

          }

        } catch (Exception e) {

          break;

        }

      }

      ois.close();

      if (!found) {

        System.out.println("Book not found in the store.");

      }

    } catch (Exception e) {

      e.printStackTrace();

    }

    sc.close();

  }

  public static void main(String[] args) {

    Book[] books = new Book[5];

    books[0] = new Book("OOP Concepts", "Oxford", new Person("John Smith"));

    books[1] = new Book("Java Mastery", "Pearson", new Person("Alice Johnson"));

    books[2] = new Book("Clean Code", "Prentice Hall", new Person("Robert Martin"));

    books[3] = new Book("Data Structures", "McGraw Hill", new Person("Sarah Allen"));

    books[4] = new Book("Algorithms", "MIT Press", new Person("Thomas Cormen"));

    writeBooksToFile(books);

    displayAllBooks();

    searchBookByName();

  }

}

**OUTPUT:**

A screen shot of a computer

AI-generated content may be incorrect.

# Create an ATM System with Account as the Serializable class. Write ten objects of Account in a file. Now write functions for withdraw, deposit, transfer and balance inquiry.

## Each function should ask for the account number on which specific operation should be done.

## All changes in Account object should be effectively represented in the file.

import java.io.\*;

import java.util.\*;

import java.io.Serializable;

import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

class Account implements Serializable {

  private int accountNumber;

  private String holderName;

  private double balance;

  public Account(int accountNumber, String holderName, double balance) {

    this.accountNumber = accountNumber;

    this.holderName = holderName;

    this.balance = balance;

  }

  public int getAccountNumber() {

    return accountNumber;

  }

  public String getHolderName() {

    return holderName;

  }

  public double getBalance() {

    return balance;

  }

  public boolean deposit(double amount) {

    if (amount > 0) {

      balance += amount;

      return true;

    } else {

      System.out.println("Deposit amount must be positive.");

      return false;

    }

  }

  public boolean withdraw(double amount) {

    if (amount <= 0) {

      System.out.println("Withdrawal amount must be positive.");

      return false;

    }

    if (balance >= amount) {

      balance -= amount;

      return true;

    } else {

      System.out.println("Insufficient balance.");

      return false;

    }

  }

  public void display() {

    System.out.println("Account #: " + accountNumber);

    System.out.println("Name: " + holderName);

    System.out.println("Balance: $" + balance);

    System.out.println("---------------------------");

  }

}

// ATM System

public class Question\_02 {

  static final String FILE\_NAME = "accounts.ser";

  public static List<Account> loadAccounts() {

    List<Account> accounts = new ArrayList<>();

    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE\_NAME))) {

      while (true) {

        try {

          Account acc = (Account) ois.readObject();

          accounts.add(acc);

        } catch (EOFException e) {

          break;

        }

      }

    } catch (Exception e) {

      e.printStackTrace();

    }

    return accounts;

  }

  public static void saveAccounts(List<Account> accounts) {

    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE\_NAME))) {

      for (Account acc : accounts) {

        oos.writeObject(acc);

      }

    } catch (Exception e) {

      e.printStackTrace();

    }

  }

  public static Account findAccount(List<Account> accounts, int accNo) {

    for (Account acc : accounts) {

      if (acc.getAccountNumber() == accNo)

      return acc;

    }

    return null;

  }

  public static void withdraw() {

    Scanner sc = new Scanner(System.in);

    List<Account> accounts = loadAccounts();

    System.out.print("Enter Account Number: ");

    int accNo = sc.nextInt();

    Account acc = findAccount(accounts, accNo);

    if (acc != null) {

      System.out.print("Enter amount to withdraw: ");

      double amt = sc.nextDouble();

      if (acc.withdraw(amt)) {

        System.out.println("Withdrawal successful.");

      }

    } else {

      System.out.println("Account not found.");

    }

    saveAccounts(accounts);

  }

  public static void deposit() {

    Scanner sc = new Scanner(System.in);

    List<Account> accounts = loadAccounts();

    System.out.print("Enter Account Number: ");

    int accNo = sc.nextInt();

    Account acc = findAccount(accounts, accNo);

    if (acc != null) {

      System.out.print("Enter amount to deposit: ");

      double amt = sc.nextDouble();

      if (acc.deposit(amt)) {

        System.out.println("Deposit successful.");

      }

    } else {

      System.out.println("Account not found.");

    }

    saveAccounts(accounts);

  }

  public static void balanceInquiry() {

    Scanner sc = new Scanner(System.in);

    List<Account> accounts = loadAccounts();

    System.out.print("Enter Account Number: ");

    int accNo = sc.nextInt();

    Account acc = findAccount(accounts, accNo);

    if (acc != null) {

      acc.display();

    } else {

      System.out.println("Account not found.");

    }

  }

  public static void transfer() {

    Scanner sc = new Scanner(System.in);

    List<Account> accounts = loadAccounts();

    System.out.print("Enter Source Account Number: ");

    int fromAccNo = sc.nextInt();

    System.out.print("Enter Destination Account Number: ");

    int toAccNo = sc.nextInt();

    System.out.print("Enter amount to transfer: ");

    double amt = sc.nextDouble();

    Account fromAcc = findAccount(accounts, fromAccNo);

    Account toAcc = findAccount(accounts, toAccNo);

    if (fromAcc != null && toAcc != null) {

      if (fromAcc.withdraw(amt) && toAcc.deposit(amt)) {

        System.out.println("Transfer successful.");

      } else {

        System.out.println("Error in Transfer.");

      }

    } else {

      System.out.println("One or both accounts not found.");

    }

    saveAccounts(accounts);

  }

  public static void main(String[] args) {

    // Initialize accounts only once if file doesn't exist

    File file = new File(FILE\_NAME);

    if (!file.exists()) {

      List<Account> initialAccounts = new ArrayList<>();

      for (int i = 0; i < 10; i++) {

        initialAccounts.add(new Account(1000 + i, "User" + (i + 1), 1000.0));

      }

      saveAccounts(initialAccounts);

      System.out.println("Initialized 10 default accounts.");

    }

    try (Scanner sc = new Scanner(System.in)) {

      while (true) {

        System.out.println("\n--- ATM MENU ---");

        System.out.println("1. Withdraw");

        System.out.println("2. Deposit");

        System.out.println("3. Transfer");

        System.out.println("4. Balance Inquiry");

        System.out.println("5. Exit");

        System.out.print("Choose: ");

        int choice = sc.nextInt();

        switch (choice) {

          case 1: withdraw(); break;

          case 2: deposit(); break;

          case 3: transfer(); break;

          case 4: balanceInquiry(); break;

          case 5: System.exit(0);

          default: System.out.println("Invalid choice.");

        }

      }

    }

  }

}

**OUTPUT:**

A black screen with white text

AI-generated content may be incorrect.

# Q3. Create a Java application to manage books for a library using permanent storage. The system should allow the following operations:

## Add a New Book:

### Each Book should have a bookID, bookName and status.

## Borrow Book.

### Borrow a book from the available books.

## Save the new state of Book object.

## Delete Book.

### Delete a book based on bookID.

### Update the storage.

import java.io.\*;

import java.util.\*;

import java.io.Serializable;

class Book implements Serializable {

  private int bookID;

  private String bookName;

  private String status;

  public Book(int bookID, String bookName, String status) {

    this.bookID = bookID;

    this.bookName = bookName;

    this.status = status;

  }

  public int getBookID() {

    return bookID;

  }

  public String getBookName() {

    return bookName;

  }

  public String getStatus() {

    return status;

  }

  public void setStatus(String status) {

    this.status = status;

  }

  public void display() {

    System.out.println("Book ID: " + bookID);

    System.out.println("Book Name: " + bookName);

    System.out.println("Status: " + status);

    System.out.println("----------------------");

  }

}

// Library Manager

public class Question\_03 {

  static final String FILE\_NAME = "library\_books.ser";

  public static List<Book> loadBooks() {

    List<Book> books = new ArrayList<>();

    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE\_NAME))) {

      while (true) {

        try {

          Book b = (Book) ois.readObject();

          books.add(b);

        } catch (EOFException e) {

          break;

        }

      }

    } catch (Exception e) {

      // file might not exist yet — that’s okay

    }

    return books;

  }

  public static void saveBooks(List<Book> books) {

    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE\_NAME))) {

      for (Book b : books) {

        oos.writeObject(b);

      }

    } catch (Exception e) {

      e.printStackTrace();

    }

  }

  public static void addBook() {

    Scanner sc = new Scanner(System.in);

    List<Book> books = loadBooks();

    System.out.print("Enter Book ID: ");

    int id = sc.nextInt();

    sc.nextLine(); // consume newline

    System.out.print("Enter Book Name: ");

    String name = sc.nextLine();

    books.add(new Book(id, name, "Available"));

    saveBooks(books);

    System.out.println("Book added successfully.");

  }

  public static void borrowBook() {

    Scanner sc = new Scanner(System.in);

    List<Book> books = loadBooks();

    System.out.print("Enter Book ID to Borrow: ");

    int id = sc.nextInt();

    boolean found = false;

    for (Book b : books) {

      if (b.getBookID() == id && b.getStatus().equalsIgnoreCase("Available")) {

        b.setStatus("Borrowed");

        found = true;

        System.out.println("Book borrowed successfully.");

        break;

      }

    }

    if (!found) {

      System.out.println("Book not found or already borrowed.");

    }

    saveBooks(books);

  }

  public static void deleteBook() {

    Scanner sc = new Scanner(System.in);

    List<Book> books = loadBooks();

    System.out.print("Enter Book ID to Delete: ");

    int id = sc.nextInt();

    boolean removed = books.removeIf(b -> b.getBookID() == id);

    if (removed) {

      System.out.println("Book deleted successfully.");

    } else {

      System.out.println("Book not found.");

    }

    saveBooks(books);

  }

  public static void showAllBooks() {

    List<Book> books = loadBooks();

    if (books.isEmpty()) {

      System.out.println("No books in the library.");

    } else {

      for (Book b : books) {

        b.display();

      }

    }

  }

  public static void main(String[] args) {

    try (Scanner sc = new Scanner(System.in)) {

      while (true) {

        System.out.println("\n--- Library Menu ---");

        System.out.println("1. Add New Book");

        System.out.println("2. Borrow Book");

        System.out.println("3. Delete Book");

        System.out.println("4. Show All Books");

        System.out.println("5. Exit");

        System.out.print("Choose: ");

        int choice = sc.nextInt();

        switch (choice) {

          case 1: addBook(); break;

          case 2: borrowBook(); break;

          case 3: deleteBook(); break;

          case 4: showAllBooks(); break;

          case 5: System.exit(0);

          default: System.out.println("Invalid choice.");

        }

      }

    }

  }

}

**OUTPUT:**

