NAME - PARTH GOYAL
ROLL NO - 24014820028
COURSE - BTECH - CSE - DS
SEMESTER - 3rd

JAVA ASSIGNMENT

CITY LIBRARY DIGITAL MANAGEMENT SYSTEM

```
import java.io.*;
import java.util.*;

class Book implements comparable <BOOK> {
    int id;
    String title;
    String author;
    String category;
    boolean issued;

    Book (int id, String title, String author, String
        category, boolean issued){
            this.id = id;
            this.title = title;
            this.author = author;
            this.category = category;
            this.issued = issued;
}

void display (){
    system.out.printIn ("ID" +id + "| Title:"
        + title + "| Author:" + Author + "
```

Teacher's Signature.....

```
| category : " + category + " | Issued: " + (issued ? "Yes" :
    "NO"));
}

public int compareTo(Book other) {
    return this.title.compareToIgnoreCase
    (other.title);
}

String toCSV() {
    return id + "," + title + "," + author +
    "," + category + "," + (issued ? "1" : "0");
}

static Book fromCSV(String line) {
    String[] p = line.split(",", 5);
    if (p.length < 5) return null;
    try {
        int id = Integer.parseInt(p[0]);
        String title = p[1];
        String author = p[2];
        String category = p[3];
        boolean issued = "1".equals(p[4].trim());
        return new Book(id, title, author, category,
                        issued);
    } catch (Exception e) {
        return null;
    }
  }
}
```

Teacher's Signature.....

Page No.
Date

```
class Member{
    int id;
    String name;
    String email;
    List <Integer> issuedBooks;

    Member(int id, String name, String email, List
            <Integer> issuedBooks) {
            this.id = id;
            this.name = name;
            this.email = email;
            this.issuedBooks = (issuedBooks != null) ?
             issuedBooks : new ArrayList<>();
    }
    void display() {
        System.out.println("ID: " +id+ " | Name: "
         +name+ " | Email: " +email+ " | Issued
         Book Ids: " + (issuedBooks.isEmpty()?
         "None": issuedBooks));
    }

    String toCSV() {
        if (issuedBooks.isEmpty()) {
            return id + "," + name + "," +email+
            "," - ";
        }
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i <issuedBooks.size(); i++){
            if (i>0) sb.append(";");
            sb.append(issuedBooks.get(i));
        }
```

Teacher's Signature.....

```
        return id + "," + name + "," + email + "," +
        + sb.toString();
    }

    Static Member fromCsv (String line) {
        String [] p = line.split(",", 4);
        if (p.length < 4) return null;
        try {
            int id = Integer.parseInt (p[0]);
            String name = p[1];
            String email = p[2];
            String rest = p[3].trim();
            List<Integer> list = new ArrayList<>
                ();
            if (!rest.equals("-")) {
                String [] ids = rest.split (";");
                for (String s : ids) {
                    try { list.add (Integer.parseInt
                (s.trim())); } catch (Exception ignored)
                {}
                }
            }
            return new Member (id, name, email,
                list);
        } catch (Exception e) {
            return null;
        }
    }

    public class CityLibraryDigitalManagementSystem
    {
```

Teacher.s Signature.....

```
private Map <Integer, BOOK> books = new
    Hashmap <>();
private Map <Integer, Member> members = new
    Hashmap <>();
private Set <String> categories = new Hasnset
    <>();

private final String BOOKS_FILE = "books.txt";
private final String MEMBERS_FILE = "members.
    txt";

private Scanner sc = new Scanner (System.in)

public static void main (String [] args ){
    cityLibraryDigitalManagementSystem app = new
    cityLibraryDigitalManagementSystem ();
    app.loadAll ();
    app.runMenu ();
    app.saveAll ();
    System.out.println ("Program finished.");
}

private void loadAll () {
    loadBooks ();
    loadMembers ();
}

private void loadBooks (){
    File f = new File (BOOKS_FILE);
    try {
```

Teacher.s Signature.....

```
        if (!f.exists()) f.createNewFile();
        BufferedReader br = new BufferedReader
        (new FileReader(f));
        String line;
        while ((line = br.readLine()) != null)
            {
            Book b = Book.fromCSV(line);
            if (b != null) {
                books.put(b.id, b);
                if (b.category != null &&
        !b.category.isEmpty()) categories.add
            (b.category);
            }
        }
        br.close();
    } catch (Exception e) {
        System.out.println("Error loading
        books: " + e.getMessage());
    }
}

private void loadMembers() {
    File f = new File(MEMBERS_FILE);
    try {
        if (!f.exists()) f.createNewFile();
        BufferedReader br = new BufferedReader
        (new FileReader(f));
        String line;
        while ((line=br.readLine()) != null) {
            Member m = fromCSV(line);
            if (m != null) members.put(m.id,m);
```

Teacher's Signature.....

```
            }
            br.close();
        } catch (Exception e) {
            System.out.println("Error loading
            members: " + e.getMessage());
        }
}

private void saveAll() {
        saveBooks();
        saveMembers();
    }
    private void saveBooks(){
        try {
            BufferWriter bw = new BufferedWriter
            (new FileWriter(BOOKS_FILE, false));
            for (Book b: books.values()){
                bw.write(b.toCSV());
                bw.newLine();
            }
            bw.close();
        } catch (Exception e){
            System.out.println("Error saving
            books: " + e.getMessage());
        }
}

private void saveMembers(){
        try {
            BufferedWriter bw = new BufferedWriter
            (new FileWriter(Members_file, false));
```

Teacher's Signature.....

```
        for (Member m : members.values()) {
            bw.write(m.toCSV());
            bw.newLine();
        }
        bw.close();
    } catch (Exception e) {
        System.out.println("Error saving
        members: " + e.getMessage());
    }
}

private void runMenu() {
    while (true) {
        System.out.println("\n City Library
            Menu:");
        System.out.println("1. Add Book");
        System.out.println("2. Add Member");
        System.out.println("3. Issue Book");
        System.out.println("4. Return Book");
        System.out.println("5. Search Books");
        System.out.println("6. Sort Books");
        System.out.println("7. Show All Books");
        System.out.println("8. Show All Members");
        System.out.println("9. Exit");
        System.out.print("Enter choice: ");
        String ch = sc.nextLine().trim();
        switch (ch) {
            case "1": addBook(); break;
            case "2": addMember(); break;
            case "3": issueBook(); break;
            case "4": returnBook(); break;
```

Teacher's Signature.....

```
            case "5": searchBooks(); break;
            case "6": sortBooksMenu(); break;
            case "7": showAllBooks(); break;
            case "8": showAllMembers(); break;
            case "9": return;
            default: System.out.println
            ("Invalid choice. Try again.");
              break;
        }
    }
}

private int nextBookId() {
    if (books.isEmpty()) return 101;
    return Collections.max(books.keySet())+1;
}

private int nextMemberId() {
    if (members.isEmpty()) return 201;
    return Collections.max(members.keyset
        ()) + 1;
}

private void addBook(){
    int id = nextBookId();
    System.out.print("Title:"); String title=
      sc.nextLine().trim();
    System.out.print("Author: "); String author=
      sc.nextLine().trim();
    System.out.print("Category: "); String
      category=sc.nextLine().trim();
```

Teacher's Signature.....

```
        Book b = new Book (id, title, author,
            category, false);
        books.put (id, b);
        if (!category.isEmpty ()) categories.add
            (category);
        saveBooks ();
        System.out.println ("Book added with
            ID " + id);
    }

    private boolean simpleEmailValid (String
            email) {
        if (email == null) return false;
        int at = email.indexOf ("@");
        int dot = email.lastIndexOf (".");
        return at > 0 && dot > at + 1 && dot
            < email.length () - 1;
    }

    private void addMember () {
        int id = nextMemberId ();
        System.out.print ("Name: "); String
        name = sc.nextLine ().trim ();
        String email;
        while (true) {
            System.out.print ("Email: "); email=
            sc.nextLine ().trim ();
            if (simpleEmailValid (email)) break;
            System.out.println ("Invalid email
                Try again.");
        }
```

```
Member m = new Member (id, name,
email, new ArrayList <>());
members.put (id, m);
saveMembers ();
System.out.println("Member added
  with Id " + id);
}

private void issueBook () {
    try {
        System.out.print ("Member ID: ");
    int mid = Integer.parseInt (sc.
nextLine ().trim ());
    Member m = members.get (mid);
    if (m == null) {system.out.println
    ("Member not found."); return;}

    System.out.print ("Book Id: ");
int bid = Integer.parseInt (sc.nextLine
() .trim ());
BOOK b = books.get (bid);
if (b == null) {system.out.println
("Book not found."); return; }

if (b.issued) { system.out.println
("Book already issued."); return; }

b.issued = true;
if (!m.issuedBooks.contains (bid))
  m.issuedBooks.add (bid);
saveAll ();
```

Teacher's Signature.....

```
            System.out.println("Book" + bid +
            " Issued to member" + mid);
        } catch (NumberFormatException e){
            System.out.println("Invalid
            numeric input.");
        }
    }

    private void returnBook(){
        try {
            System.out.print("Member Id: "); int
            mid = Integer.parseInt(sc.nextLine().
            trim());
            Member m = members.get(mid);
            if (m==null){System.out.println
            ("Member not found."); return;}

            System.out.print("BOOK ID:"); int
            bid = Integer.parseInt(sc.nextLine
            ().trim());
            Book b = books.get(bid);
            if (b==null){System.out.println("Book
            not found."); return;}

            if (!b.issued){System.out.println("Book
            is not issued."); return;}

            b.issued = false;
            m.issuedBooks.remove(Integer.valueOf
            (bid));
            saveAll();
```

Teacher's Signature.....

```
            System.out.println("Book" + bid + "returned
            by member" + mid);
        } catch (NumberFormatException e) {
            System.out.println("Invalid
            numeric input.");
        }
    }

    private void searchBooks() {
        System.out.print("search by 1) Title 2)
        Author 3) category: ");
        String opt = sc.nextLine().trim();
        System.out.print("Enter keyword:");
        String kw = sc.nextLine().trim().
        toLowerCase();
        List<Book> res = new ArrayList<>();
        for (Book b: books.values()) {
            if ("1".equals(opt) && b.title.
            toLowerCase().contains(kw)) res.add
            (b);
            if ("2".equals(opt) && b.author.
            toLowerCase().contains(kw)) res.add(b);
            if ("3".equals(opt) && b.category.
            toLowerCase().contains(kw)) res.add(b);
        }
        if (res.isEmpty()) System.out.println
        ("No Books found.");
        else for (Book b: res) b.display();
    }

    private void sortBooksMenu() {
```

Teacher's Signature.....

```
System.out.print("Sort 1) Title 2) Author
 3) Category: ");
String opt = sc.nextLine().trim();
List<Book> list = new ArrayList<>
 (Books.values());
Switch (opt) {
   case "1": Collections.sort(list);
   break;
case "2": List.sort (new comparator
   <Book> (){
   public int compare (Book a,
    Book b) {return a.author.
compare ToIgnoreCase (b.author);}
}); break;
case "3": list.sort (new comparator
   <BOOK> () {
    public int compare (BOOK a,
   Book b) {return a.category.
 compareToIguoreCase (b.category);}
}); break;
default system.out.println ("Invalid
choice. Showing unsorted."); break;
   }
   for (Book b : list) b.display ();
}
private void showAllBooks () {
    if (books.isEmpty ()) System.out.println
    ("NO books in library");
} else for (BOOK b : books.values ()) b.display
    ();
}
```

Teacher.s Signature.....

```
private void showAllMembers(){
    if(members.isEmpty()) System.out.
    println("NO members registered.");
    else for(Member m: members.
    values()) m.display();
}
}
```

Teacher's Signature.....