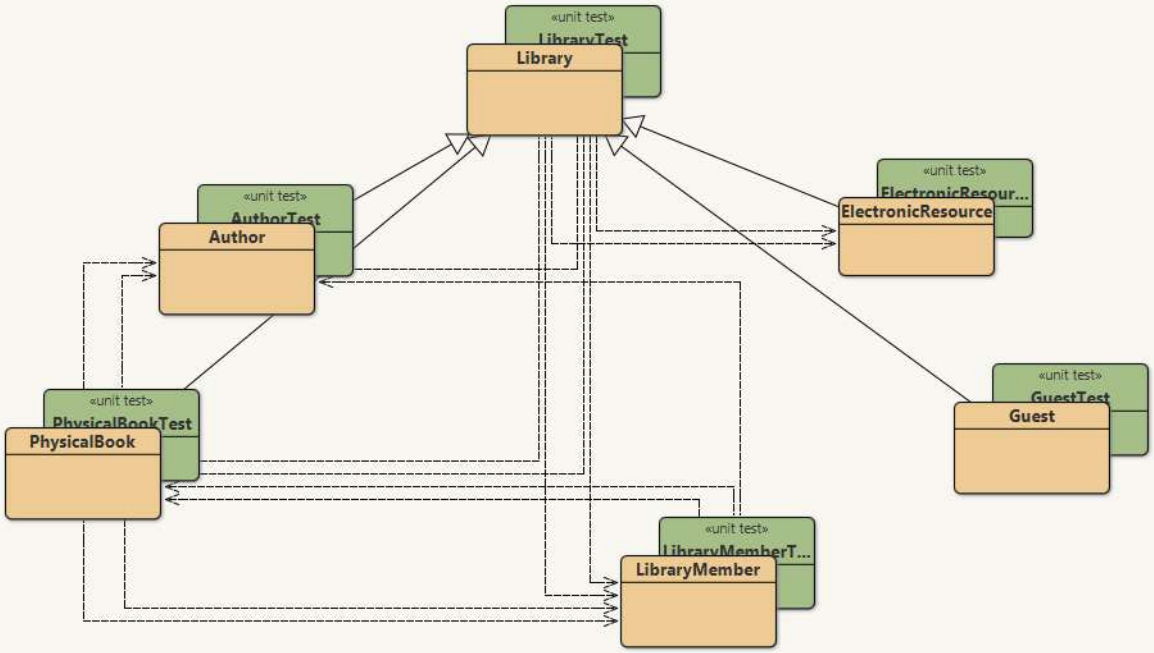
«unit test»
LibraryTest
Library
«unit test»
AuthorTest
Author
«unit test»
ElectronicResour...
ElectronicResource
«unit test»
PhysicalBookTest
PhysicalBook
«unit test»
GuestTest
Guest
«unit test»
LibraryMemberT...
LibraryMember

```java
import java.util.*;
/**
 * A class to store the data of Library Members
 *
 * @author Deepak
 * @version 30.11.2023
 */
public class LibraryMember
{
    // instance variables
    private String memberId;
    private String memberName;
    private String memberContact;
    private ArrayList<PhysicalBook> booksBorrowed;

    /**
     * Constructor for class LibraryMember objects.
     */
    public LibraryMember(String memid, String memname, String memcontact)
    {
        memberId = memid;
        memberName = memname;
        memberContact = memcontact;
        booksBorrowed = new ArrayList<>();
    }

    /**
     * A method to get the Member Id of Library Member.
     */
    public String getMemberId()
    {
        return memberId;
    }

    /**
     * A method to set the Member Id of Library Member.
     */
    public void setMemberId(String memid)
    {
        memberId = memid;
    }

    /**
     * A method to get the Member Name of Library Member.
     */
    public String getMemberName()
    {
        return memberName;
    }
```

```java
    /**
     * A method to set the Member Name of Library Member.
     */
    public void setMemberName(String memname)
    {
        memberName = memname;
    }

    /**
     * A method to get the Member Contact of Library Member.
     */
    public String getMemberContact()
    {
        return memberContact;
    }

    /**
     * A method to set the Member Contact of Library Member.
     */
    public void setMemberContact(String memcontact)
    {
        memberContact = memcontact;
    }

    /**
     * An ArrayList to store the data of the PhysicalBook class.
     */
    public ArrayList<PhysicalBook> getBooksborrowed()
    {
        return booksBorrowed;
    }

    /**
     * A method to add the books borrowed by Library Member.
     */
    public void addBooksBorrowed(PhysicalBook book)
    {
        if (booksBorrowed.contains(book))
        {
            System.out.println("Book already borrowed");
        }
        else
        {
            booksBorrowed.add(book);
        }

    }

    /**
     * A method to print the details of Library Member.
```

```
     */
    public void printDetails()
    {
        System.out.println("Member Id: " + memberId);
        if (booksBorrowed.isEmpty())
        {
            System.out.println("No books borrowed");
        }
        else
        {
            System.out.println("Books borrowed: ");
            for (PhysicalBook book : booksBorrowed)
            {
                System.out.println("* " + book.getBookTitle());
            }
        }
    }

    /**
     * A method to get the the number of books borrowed by the Library Member.
     */
    public int getNumberOfBooksBorrowed()
    {
        return booksBorrowed.size();
    }
}
```

```java
import java.util.*;
import javax.swing.*;
/**
 * A class for the guest of the library.
 * @author Deepak
 * @version 30.11.2023
 */
public class Guest extends Library
{
    // Fields of the class
    private String guestName;
    private String guestId;
    private int durationOfAccess;

    /**
     * Constructor for class Guest objects.
     */
    public Guest(String gName, String gId, int duraccess)
    {
        guestName = gName;
        guestId = gId;
        if ( duraccess <= 3)
        {
            durationOfAccess = duraccess;
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Access permitted for max 3 days");
        }
    }

    /**
     * A method to get the name of the guest.
     */
    public String getGuestName()
    {
        return guestName;
    }

    /**
     * A method to set the name of the guest.
     */
    public void setGuestName(String gName)
    {
        guestName = gName;
    }

    /**
     * A method to get the id of the guest.
     */
```

```java
    public String getGuestId()
    {
        return guestId;
    }

    /**
     * A method to set the id of the guest.
     */
    public void setGuestId(String gId)
    {
        guestId = gId;
    }

    /**
     * A method to get the access duration of the guest.
     */
    public int getDurationOfAccess()
    {
        return durationOfAccess;
    }

    /**
     * A method to set the access duration of the guest.
     */
    public void setDurationOfAccess(int durAccess)
    {
        if (durationOfAccess <= 3)
        {
            durationOfAccess = durAccess;
            System.out.println("Access permitted");
        }
        else
        {
            JOptionPane.showMessageDialog(null,"Access permitted for max 3 days");
        }
    }

    /**
     * A method to borrow book by the guest.
     */
    public void borrowBook()
    {
        JOptionPane.showMessageDialog(null, "Only members can borrow book");
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * The test class ElectronicResourceTest.
 *
 * @author  Deepak
 * @version 30.11.2023
 */
public class ElectronicResourceTest
{
    /**
    * Testing the method getEbook.
    */
    @Test
    public void getEbook()
    {
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        assertEquals("Ebook", electron1.getEbook());
    }

    /**
    * Testing the method setEbook.
    */
    @Test
    public void setEbook()
    {
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        electron1.setEbook("OOP");
    }

    /**
    * Testing the method getJournal.
    */
    @Test
    public void getJournal()
    {
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        assertEquals("Journal", electron1.getJournal());
    }

    /**
    * Testing the method setJournal.
    */
    @Test
    public void setJournal()
    {
```

```java
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        electron1.setJournal("OOP");
    }

    /**
    * Testing the method isDownloadable.
    */
    @Test
    public void isDownloadable()
    {
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        assertEquals(false, electron1.isDownloadable());
    }

    /**
    * Testing the method setDownloadable.
    */
    @Test
    public void setDownloadable()
    {
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        electron1.setDownloadable(true);
    }

    /**
    * Testing the method printDetails.
    */
    @Test
    public void printDetails()
    {
        ElectronicResource electron1 = new ElectronicResource("Ebook", "Journal");
        electron1.printDetails();
    }
}
```

```java
/**
 * A class for ElectronicResource.
 *
 * @author Deepak
 * @version 30.11.2023
 */
public class ElectronicResource extends Library
{
    // instance fields of the class
    private boolean downloadable;
    private String ebook;
    private String journal;

    /**
     * Constructor for objects of class ElectronicResource
     */
    public ElectronicResource(String eb, String jrnl)
    {
        ebook = eb;
        journal = jrnl;
    }

    /**
     * A method to get the ebook.
     */
    public String getEbook()
    {
        return ebook;
    }

    /**
     * A method to set the ebook.
     */
    public void setEbook(String eb)
    {
        ebook = eb;
    }

    /**
     * A method to get the journal.
     */
    public String getJournal()
    {
        return journal;
    }

    /**
     * A method to set the journal.
     */
```

```
    public void setJournal(String jrnl)
    {
        journal = jrnl;
    }

    /**
     * A method to check if the resource is downloadable or not.
     */
    public boolean isDownloadable()
    {
        return downloadable;
    }

    /**
     * A method to set the value(true or false) if the resource is downloadable or
not.
     */
    public void setDownloadable(boolean download)
    {
        downloadable = download;
    }

    /**
     * A method to print the details of resources.
     */
    public void printDetails()
    {
        System.out.println("Category: Ebook");
        System.out.println("Ebook Title: " + ebook);
        System.out.println("Download Allowed: " + downloadable);
        System.out.println("Category: Journal");
        System.out.println("Journal Name: " + journal);
        System.out.println("Download Allowed: " + downloadable);
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * The test class AuthorTest.
 *
 * @author  Deepak
 * @version 30.11.2023
 */
public class AuthorTest
{
    /**
    * Testing the method getAuthorFirstName.
    */
    @Test
    public void getAuthorFirstName()
    {
        Author author1 = new Author();
        assertEquals("", author1.getAuthorFirstName());
    }

    /**
    * Testing the method setAuthorFirstName.
    */
    @Test
    public void setAuthorFirstName()
    {
        Author author1 = new Author();
        author1.setAuthorFirstName("Deepak");
    }

    /**
    * Testing the method getAuthorSurname.
    */
    @Test
    public void getAuthorSurname()
    {
        Author author1 = new Author();
        assertEquals("", author1.getAuthorSurname());
    }

    /**
    * Testing the method setAuthorSurname.
    */
    @Test
    public void setAuthorSurname()
```

```
    {
        Author author1 = new Author();
        author1.setAuthorSurname("Rawal");
    }

    /**
    * Testing the method getAuthorAddress.
    */
    @Test
    public void getAuthorAddress()
    {
        Author author1 = new Author();
        assertEquals("", author1.getAuthorAddress());
    }

    /**
    * Testing the method setAuthorAddress.
    */
    @Test
    public void setAuthorAddress()
    {
        Author author1 = new Author();
        author1.setAuthorAddress("Address1");
    }

    /**
    * Testing the method showDetails.
    */
    @Test
    public void showDetails()
    {
        Author author1 = new Author();
        author1.showDetails();
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * The test class GuestTest.
 *
 * @author  Deepak
 * @version 30.11.2023
 */
public class GuestTest
{
    /**
    * Testing the method getGuestName.
    */
    @Test
    public void getGuestName()
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        assertEquals("Deepak", guest1.getGuestName());
    }

    /**
    * Testing the method setGuestName.
    */
    @Test
    public void setGuestName()
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        guest1.setGuestName("Rawal");
    }

    /**
    * Testing the method getGuestId.
    */
    @Test
    public void getGuestId()
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        assertEquals("G1", guest1.getGuestId());
    }

    /**
    * Testing the method setGuestId.
    */
    @Test
    public void setGuestId()
```

```
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        guest1.setGuestId("GG1");
    }

    /**
    * Testing the method getDurationOfAccess.
    */
    @Test
    public void getDurationOfAccess()
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        assertEquals(3, guest1.getDurationOfAccess());
    }

    /**
    * Testing the method setDurationOfAccess.
    */
    @Test
    public void setDurationOfAccess()
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        guest1.setDurationOfAccess(2);
    }

    /**
    * Testing the method borrowBook.
    */
    @Test
    public void borrowBook()
    {
        Guest guest1 = new Guest("Deepak", "G1", 3);
        guest1.borrowBook();
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;

/**
 * The test class PhysicalBookTest.
 *
 * @author  Deepak
 * @version 30.11.2023
 */
public class PhysicalBookTest
{
    /**
    * Testing the method getIsbn.
    */
    @Test
    public void getIsbn()
    {
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        assertEquals("ABC123", physical1.getIsbn());
    }

    /**
    * Testing the method setIsbn.
    */
    @Test
    public void setIsbn()
    {
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "AB123", "BookTitle");
        physical1.setIsbn("ABC123");
    }

    /**
    * Testing the method getBookTitle.
    */
    @Test
    public void getBookTitle()
    {
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        assertEquals("BookTitle", physical1.getBookTitle());
    }

    /**
    * Testing the method setBookTitle.
```

```
    */
    @Test
    public void setBookTitle()
    {
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "Btitle");
        physical1.setBookTitle("BookTitle");
    }

    /**
    * Testing the method getAuthor.
    */
    @Test
    public void getAuthor()
    {
        Author author1 = new Author();
        author1.showDetails();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        assertEquals("","","");
    }

    /**
    * Testing the method setAuthor.
    */
    @Test
    public void setAuthor()
    {
        Author author1 = new Author();
        author1.setAuthorFirstName("Deepak");
        author1.setAuthorSurname("Rawal");
        author1.setAuthorAddress("Author Address");
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        physical1.setAuthor(author1);
    }

    /**
    * Testing the method getLibMember.
    */
    @Test
    public void getLibMember()
    {
        LibraryMember libraryM1 = new LibraryMember("Libmember1", "Deepak", "member
contact");
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        libraryM1.printDetails();
        assertEquals(null, physical1.getLibMember());
    }

    /**
```

```
    * Testing the method setLibMember.
    */
    @Test
    public void setLibMember()
    {
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        LibraryMember libraryM1 = new LibraryMember("libmember1", "Deepak",
"memcontact");
        libraryM1.setMemberId("mem1");
        libraryM1.setMemberName("member");
        libraryM1.setMemberContact("memcontact");
        physical1.setLibMember(libraryM1);
    }

    /**
    * Testing the method addDamage.
    */
    @Test public void addDamage()
    {
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        physical1.addDamage("missing pages");
        physical1.addDamage("untidy");
        ArrayList damages = physical1.getDamages();
        assertEquals(2, damages.size());
        assertTrue(damages.contains("missing pages"));
        assertTrue(damages.contains("untidy"));
    }

}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import java.util.*;

/**
 * The test class LibraryMemberTest.
 *
 * @author  Deepak
 * @version 30.11.2023
 */
public class LibraryMemberTest
{
    /**
    * Testing the method getMemberId.
    */
    @Test
    public void getMemberId()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        assertEquals("memberid", libraryM1.getMemberId());
    }

    /**
    * Testing the method setMemberId.
    */
    @Test
    public void setMemberId()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        libraryM1.setMemberId("memid");
        assertEquals("memid", libraryM1.getMemberId());
    }

    /**
    * Testing the method getMemberName.
    */
    @Test
    public void getMemberName()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        assertEquals("Deepak", libraryM1.getMemberName());
    }

    /**
```

```
    * Testing the method setMemberName.
    */
    @Test
    public void setMemberName()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        libraryM1.setMemberName("DR");
        assertEquals("DR", libraryM1.getMemberName());
    }

    /**
    * Testing the method getMemberContact.
    */
    @Test
    public void getMemberContact()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        assertEquals("membercontact", libraryM1.getMemberContact());
    }

    /**
    * Testing the method setMemberContact.
    */
    @Test
    public void setMemberContact()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        libraryM1.setMemberContact("memcontact");
        assertEquals("memcontact", libraryM1.getMemberContact());
    }

    /**
    * Testing the method getBooksBorrowed.
    */
    @Test
    public void getBooksBorrowed()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        assertEquals(new ArrayList(), libraryM1.getBooksborrowed());
    }

    /**
    * Testing the method addBookBorrowed.
    */
    @Test public void addBooksBorrowed()
    {
```

```
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        libraryM1.addBooksBorrowed(physical1);
        assertEquals(1, libraryM1.getNumberOfBooksBorrowed());
    }

    /**
    * Testing the method printDetails.
    */
    @Test public void printDetails()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        libraryM1.printDetails();
    }

    /**
    * Testing the method getNumberOfBooksBorrowed.
    */
    @Test public void getNumberOfBooksBorrowed()
    {
        LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
        Author author1 = new Author();
        PhysicalBook physical1 = new PhysicalBook(author1, "ABC123", "BookTitle");
        libraryM1.addBooksBorrowed(physical1);
        assertEquals(1, libraryM1.getNumberOfBooksBorrowed());
    }

}
```

```java
/**
 * A class that Stores the details of the Author.
 *
 * @author Deepak
 * @version 1.0
 */
public class Author extends Library
{
    // instance fields of the class Author
    private String authorFirstName;
    private String authorSurname;
    private String authorAddress;

    /**
     * Constructor for class Author objects.
     */
    public Author()
    {
        // initialising instance fields
        authorFirstName = "";
        authorSurname = "";
        authorAddress = "";
    }

    /**
     * A getter method for authorFirstName
     */
    public String getAuthorFirstName()
    {
        return authorFirstName;
    }

    /**
     * A setter method for authorFirstName
     */
    public void setAuthorFirstName(String fn)
    {
        authorFirstName = fn;
    }

    /**
     * A getter method for authorSurname
     */
    public String getAuthorSurname()
    {
        return authorSurname;
    }

    /**
```

```
     * A setter method for authorSurname
     */
    public void setAuthorSurname(String sn)
    {
        authorSurname = sn;
    }

    /**
     * A getter method for authorAddress
     */
    public String getAuthorAddress()
    {
        return authorAddress;
    }

    /**
     * A setter method for authorAddress
     */
    public void setAuthorAddress(String address)
    {
        authorAddress = address;
    }

    /**
     * A method to show the ddetails of the author
     */
    public void showDetails()
    {
        System.out.println("\nAuthor: " + authorFirstName + " " + authorSurname +
"\nAuthor Address: " + authorAddress);
    }
}
```

```java
import java.util.*;
/**
 * The Lbrary class takes the values from various classes i.e. ElectronicResource,
 * PhysicalBook, LibraryMember.
 *
 * @author Deepak
 * @version 30.11.2023
 */
public class Library
{
    //instance field
    private ArrayList<Object> catalogue;
    /**
     * Constructor for class Library objects.
     */
    public Library()
    {
        catalogue = new ArrayList<>();
    }

    /**
     * A method to get the Catalogue.
     */
    public ArrayList<Object> getCatalogue()
    {
        return catalogue;
    }

    /**
     * A method to set the Catalogue.
     */
    public void setCatalogue(ArrayList<Object> cat)
    {
        catalogue = cat;
    }

    /**
     * A method to print the details of the Catalogue.
     */
    public void printDetails()
    {
        System.out.println("Catalogue:- ");
        for (Object resource : catalogue)
        {
            if (resource instanceof PhysicalBook)
            {
                ((PhysicalBook) resource).printDetails();
            }
            else if (resource instanceof ElectronicResource)
            {
```

```
                ((ElectronicResource) resource).printDetails();
            }
        }
    }

    /**
     * A method to check if the Catalogue contains a particular resource.
     */
    public boolean containResource(Object resource)
    {
        return catalogue.contains(resource);
    }

    /**
     * A method to modify the author name in the resource of the Catalogue.
     */
    public void modifyAuthorName(PhysicalBook book, String newAuthorFirstName)
    {
        if (catalogue.contains(book))
        {
            if (book.getAuthor() != null)
            {
                book.getAuthor().setAuthorFirstName(newAuthorFirstName);
            }
            else
            {
                System.out.println("Cannot find Author Data");
            }
        }
        else
        {
            System.out.println("Resurce object not found in catalogue");
        }
    }

    /**
     * A method to search a resource by its title in the Catalogue.
     */
    public void searchTitle(String bookTitle)
    {
        int count = 0;
        for (Object resource : catalogue)
        {
            if (resource instanceof PhysicalBook)
            {
                if (((PhysicalBook) resource).getBookTitle().equals(bookTitle))
                {
                    ((PhysicalBook) resource).printDetails();
                    count += 1;
                }
```

```
            }
        }
        System.out.println("\nTotal resources: " + count);
    }

    /**
     * A method to search a resource by the author's surname in the Catalogue.
     */
    public void searchAuthorSurname(String authSurname)
    {
        int count = 0;
        for (Object resource : catalogue)
        {
            if (resource instanceof PhysicalBook)
            {
                if (((PhysicalBook) resource).getAuthor() != null)
                {
                    String authorSurname = ((PhysicalBook)
resource).getAuthor().getAuthorSurname();
                    if (authorSurname.equalsIgnoreCase(authSurname))
                    {
                        ((PhysicalBook) resource).printDetails();
                        count += 1;
                    }
                }
            }
        }
        System.out.println("\nTotal resource found: " + count);
    }

    /**
     * A method to remove a resource by from the Catalogue.
     */
    public void removeResource(Object resource)
    {
        if (catalogue.contains(resource))
        {
            if (resource instanceof PhysicalBook)
            {
                PhysicalBook book = (PhysicalBook) resource;
                if (book.getLibMember() != null)
                {
                    System.out.println("Resource on loan");
                }
                else
                {
                    catalogue.remove(resource);
                }
            } else
            {
```

```
                catalogue.remove(resource);
            }
        }
        else
        {
            System.out.println("Resource object not found");
        }
    }

    /**
     * A method to remove a resource by its position in the Catalogue.
     */
    public void removeResourceFromPosition(int position)
    {
        if (position >= 0 && position < catalogue.size())
        {
            Object resource = catalogue.get(position);
            if (resource instanceof PhysicalBook)
            {
                PhysicalBook book = (PhysicalBook) resource;
                if (book.getLibMember() != null)
                {
                    System.out.println("Resource on loan");
                }
                else
                {
                    catalogue.remove(position);
                }
            }
            else
            {
                catalogue.remove(position);
            }
        }
        else
        {
            System.out.println("Resource not found at this position");
        }
    }

    /**
     * A method to print the details of the books available in the Catalogue.
     */
    public void printBooksAvailable()
    {
        System.out.println("Books Available: \n");
        for (Object resource : catalogue)
        {
            if (resource instanceof PhysicalBook)
            {
```

```
                PhysicalBook book = (PhysicalBook) resource;
                if (book.isAvailable())
                {
                    book.printDetails();
                }
            }
        }
    }
    /**
     * A method to get the number of resources in the Catalogue.
     */
    public int getNumberOfResources()
    {
        return catalogue.size();
    }

    /**
     * A method to add a resource in the Catalogue.
     */
    public void addResource(Object resource)
    {
        if (catalogue.contains(resource))
        {
            System.out.println("Resource object already exists in the catalogue");
        }
        else
        {
            catalogue.add(resource);
        }
    }

    /**
     * A method to lend a book in the Catalogue to a library member.
     */
    public void lendBook(PhysicalBook book, LibraryMember libmember) {
        if (catalogue.contains(book))
        {
            if (book.isAvailable())
            {
                book.setLibMember(libmember);
                libmember.addBooksBorrowed(book);
            }
            else
            {
                System.out.println("Book already borrowed");
            }
        }
        else
        {
            System.out.println("Book not found");
```

```
        }
    }

    /**
     * A method to return a book in the Catalogue lent by a  library member.
     */
    public void returnBook(PhysicalBook book, boolean recordDamage, String
damageDescription)
    {
        if (catalogue.contains(book))
        {
            book.setLibMember(null);
            if (recordDamage)
            {
                book.addDamage(damageDescription);
            }
            System.out.println("Book returned");
        }
        else
        {
            System.out.println("Book not found");
        }
    }

    /**
     * A method to print the details of all physical books in the Catalogue.
     */
    public void allPhysicalBook()
    {
        System.out.println("Physical Books:");
        for (Object resource : catalogue)
        {
            if (resource instanceof PhysicalBook)
            {
                ((PhysicalBook) resource).printDetails();
            }
        }
    }

    /**
     * A method to print the details of all electronic resources in the Catalogue.
     */
    public void allElectronicResources()
    {
        System.out.println("Electronic Resources: ");
        for (Object resource : catalogue)
        {
            if (resource instanceof ElectronicResource)
            {
                ((ElectronicResource) resource).printDetails();
```

```
            }
        }
    }
}
```

```java
import java.util.*;
/**
 * A class that stores the details of the physical book in the library.
 *
 * @author Deepak
 * @version 30.11.2023
 */
public class PhysicalBook extends Library
{
    //instance variables
    private String isbn;
    private String bookTitle;
    private Author author;
    private LibraryMember libMember;
    private ArrayList<String> damages;

    /**
     * Constructor for class PhysicalBook objects.
     */
    public PhysicalBook(Author auth, String isbnumber, String bTitle)
    {
        isbn = isbnumber;
        bookTitle = bTitle;
        author = auth;
        libMember = null;
        damages = new ArrayList<>();

    }

    /**
     * Method to get the ISBN of the book.
     */
    public String getIsbn()
    {
        return isbn;
    }

    /**
     * Setter method for the ISBN of the book.
     */
    public void setIsbn(String isbnumber)
    {
        isbn = isbnumber;
    }

    /**
     * Method to get the Title of the book.
     */
    public String getBookTitle()
    {
```

```
        return bookTitle;
    }

    /**
     * Method to set the Title of the book.
     */
    public void setBookTitle(String ttl)
    {
        bookTitle = ttl;
    }

    /**
     * Method to get the author object of the Author class.
     */
    public Author getAuthor()
    {
        return author;
    }

    /**
     * Method to set the author object of the Author class.
     */
    public void setAuthor(Author auth)
    {
        author = auth;
    }

    /**
     * Method to get the LibraryMember object of the LibraryMember class.
     */
    public LibraryMember getLibMember()
    {
        return libMember;
    }

    /**
     * Method to set the LibraryMember object of the LibraryMember class.
     */
    public void setLibMember(LibraryMember libMember)
    {
        libMember = libMember;
    }

    /**
     * Method to get the damages of the book from the ArrayList.
     */
    public ArrayList<String> getDamages()
    {
        return damages;
    }
```

```java
    /**
     * Method to set the damages of the book in the ArrayList.
     */
    public void setDamages(ArrayList<String> damage)
    {
        damages = damage;
    }

    /**
     * Method to add the damages of the book in the ArrayList.
     */
    public void addDamage(String damage)
    {
        damages.add(damage);
    }

    /**
     * Method to check if a book is available.
     */
    public boolean isAvailable()
    {
        return libMember == null;
    }

    /**
     * Method to print the details of the books.
     */
    public void printDetails()
    {
        System.out.println("\nTitle: " +bookTitle);
        if (libMember != null)
        {
            System.out.println("Borrowed by: " +libMember.getMemberId());
        }
        else
        {
            System.out.println("Available");
        }

        if (author != null)
        {
            System.out.println("Author: " + author.getAuthorFirstName() +
author.getAuthorSurname());
        }
        else
        {
            System.out.println("Cannot find Author");
        }
```

```
        if (!damages.isEmpty())
        {
            System.out.println("Damage to the book: " + damages);
        }
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import java.util.*;
import java.io.PrintStream;
import java.io.ByteArrayOutputStream;

/**
 * The test class for Library class.
 *
 * @author  Deepak
 * @version 30.11.2023
 */
public class LibraryTest
    {
        /**
         * Testing the method getCatalogue.
         */
        @Test public void tetCatalogue()
        {
            Library library1 = new Library();
            assertNotNull(library1.getCatalogue());
        }

        /**
         * Testing the method setCatalogue.
         */
        @Test public void setCatalogue()
        {
            Library library1 = new Library();
            ArrayList catalogue = new ArrayList<>();
            library1.setCatalogue(catalogue);
            assertEquals(catalogue, library1.getCatalogue());
        }

        /**
         * Testing the method containResource.
         */
        @Test public void containResource()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            library1.addResource(physical1);
            assertTrue(library1.containResource(physical1));
        }

        /**
```

```
         * Testing the method modifyAuthorName.
         */
        @Test public void modifyAuthorName()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            author1.setAuthorFirstName("Deepak");
            physical1.setAuthor(author1);
            library1.addResource(physical1);
            library1.modifyAuthorName(physical1, "Deepak");
            assertEquals("Deepak", physical1.getAuthor().getAuthorFirstName());
        }

        /**
         * Testing the method searchTitle.
         */
        @Test public void searchTitle()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            library1.addResource(physical1);
            library1.addResource(physical2);
            library1.searchTitle("CS994");
            assertEquals(2, library1.getNumberOfResources());
        }

        /**
         * Testing the method searchAuthorSurname.
         */
        @Test public void searchAuthorSurname()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            Author author2 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            author1.setAuthorSurname("Raw");
            author2.setAuthorSurname("Rawal");
            physical1.setAuthor(author1);
            physical2.setAuthor(author2);
            library1.addResource(physical1);
            library1.addResource(physical2);
            library1.searchAuthorSurname("Raw");
            assertEquals(2, library1.getNumberOfResources());
        }
```

```
         * Testing the method removeResource.
         */
        @Test public void removeResource()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            library1.addResource(physical1);
            library1.removeResource(physical1);
            assertFalse(library1.containResource(physical1));
        }

        /**
         * Testing the method removeResourceFromPosition.
         */
        @Test public void removeResourceFromPosition()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            library1.addResource(physical1);
            library1.addResource(physical2);
            library1.removeResourceFromPosition(1);
            assertFalse(library1.containResource(physical2));
        }

        /**
         * Testing the method printBooksAvailable.
         */
        @Test public void printBooksAvailable()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            library1.addResource(physical1);
            library1.addResource(physical2);
            library1.printBooksAvailable();
            String expectedOutput = "Books Available: \n" + "Title: BookTitle
\nAuthor: " + "Title: java  \nAuthor:  ";
            assertFalse(library1.toString().contains(expectedOutput));
        }

        /**
         * Testing the method getNumberPfResources.
         */
        @Test public void getNumberOfResources()
```

```
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            library1.addResource(physical1);
            library1.addResource(physical2);
            assertEquals(2, library1.getNumberOfResources());
        }

        /**
         * Testing the method addResources.
         */
        @Test public void addResource()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            library1.addResource(physical1);
            library1.addResource(physical2);
            assertTrue(library1.containResource(physical1));
            assertTrue(library1.containResource(physical2));
        }

        /**
         * Testing the method lendBook.
         */
        @Test public void lendBook()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            LibraryMember libraryM1 = new LibraryMember("memberid", "Deepak",
"membercontact");
            library1.addResource(physical1);
            library1.lendBook(physical1, libraryM1);
            assertTrue(library1.containResource(physical1));
        }

        /**
         * Testing the method returnBook.
         */
        @Test public void returnBook()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            physical1.setLibMember(new LibraryMember("DR1", "DR", "dr"));
```

```
        }

        /**
         * Testing the method allPhysicalBook.
         */
        @Test public void allPhysicalBook()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            PhysicalBook physical2 = new PhysicalBook(author1, "java", "java");
            ElectronicResource electron1 = new ElectronicResource("Ebook",
"Journal");
            library1.addResource(physical1);
            library1.addResource(physical2);
            library1.addResource(electron1);
            ByteArrayOutputStream outContent = new ByteArrayOutputStream();
            System.setOut(new PrintStream(outContent));
            library1.printDetails();
            System.setOut(System.out);
            String expectedOutput = "Books:\n" + "Physical Book " + "Author: \n" +
"Title: BookTitle " + "ISBN: ABC123 " + "Physical Book " + "Author: \n" + "Title:
java " + "ISBN: java";
            assertFalse(library1.toString().contains(expectedOutput));
        }

        /**
         * Testing the method allElectronicResources.
         */
        @Test public void allElectronicResources()
        {
            Library library1 = new Library();
            Author author1 = new Author();
            PhysicalBook physical1 = new PhysicalBook(author1, "ABC123",
"BookTitle");
            ElectronicResource electron1 = new ElectronicResource("ebook1",
"journal1");
            ElectronicResource electron2 = new ElectronicResource("ebook1",
"journa2");
            library1.addResource(physical1);
            library1.addResource(electron1);
            library1.addResource(electron2);
            ByteArrayOutputStream outContent = new ByteArrayOutputStream();
            System.setOut(new PrintStream(outContent));
            library1.printDetails();
            System.setOut(System.out);
            String expectedOutput = "E-Resources:\n"; expectedOutput += "Deepak";
expectedOutput += "Rawal"; expectedOutput += "Address1"; expectedOutput += "ebook1";
expectedOutput += "journal1"; expectedOutput += "ebook1"; expectedOutput += "journal2
";
            assertFalse(outContent.toString().contains(expectedOutput));
```

PROJECT TITLE: Library System
PURPOSE OF PROJECT: Creating a Library System
VERSION or DATE: 30th November 2023
HOW TO START THIS PROJECT: Creating objects
AUTHOR: Deepak
USER INSTRUCTIONS:
The Library System indicated in this project include classes such as:

- Library
- PhysicalBook
- Author
- ElectronicResource
- LibraryMember
- Guest

Here, the Library class acts as a superclass or a parent class, where the PhysicalBook, ElectronicResource, LibraryMember and Guest class acts as a subclass or a child class. The inheritance concept is required to establish a relationship between different resources of the library, which includes PhysicalBook and ElectronicResource. This concept helps in creating a unified catalogue which in turn benefits us in maintaining different resources of the library in a consistent manner.

As the library class is a superclass, it can collect various types of resources from various classes. By which we can access and maintain resources in the collection of resources or we can say that we can handle the resources in this catalog in an effective and efficient way. The class PhysicalBook and ElectronicResource being a subclass have their own unique specific features and working as their class indicates.

The subclass LibraryMember represents library members of the Library who are allowed to borrow books and access the electronic resources. The class LibraryMember inherits from the supercass Library which allows members to access different types of resources.

As stated, the class Guest also acts as a subclass of the class Library(superclass). This class presents guest of the library who have limitations such as able to access the library for maximum of 3 days and are not allowed to borrow books but can access the electronic resources.

To conclude, we can say that inheritance gives a better opportunity to make our work more flexible. It also allow the user to reuse the code as we can make changes to the functions in the superclass from where the subclasses inherits.