Books.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Demo_Library_Management
{
    public class Book
    {
        public string Title { get; set; }
        public string Author { get; set; }
        public string ISBN { get; set; }
        public bool IsBorrowed { get; private set; }

        public Book(string title, string author, string isbn)
        {
            Title = title;
            Author = author;
            ISBN = isbn;
            IsBorrowed = false;
        }

        public void Borrow()
        {
            if (IsBorrowed)
                throw new InvalidOperationException("Book is already borrowed.");
            IsBorrowed = true;
        }

        public void Return()
        {
            if (!IsBorrowed)
                throw new InvalidOperationException("Book is not currently
borrowed.");
            IsBorrowed = false;
        }
    }
}
```

Library.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Demo_Library_Management
```

```
{
    public class Library
    {
        public List<Book> Books { get; private set; } = new List<Book>();
        public List<Borrower> Borrowers { get; private set; } = new
List<Borrower>();

        public void AddBook(Book book)
        {
            Books.Add(book);
        }

        public void RegisterBorrower(Borrower borrower)
        {
            Borrowers.Add(borrower);
        }

        public void BorrowBook(string isbn, string libraryCardNumber)
        {
            var book = Books.FirstOrDefault(b => b.ISBN == isbn);
            var borrower = Borrowers.FirstOrDefault(b => b.LibraryCardNumber ==
libraryCardNumber);

            if (book == null || borrower == null)
                throw new Exception("Book or Borrower not found.");

            borrower.BorrowBook(book);
        }

        public void ReturnBook(string isbn, string libraryCardNumber)
        {
            var book = Books.FirstOrDefault(b => b.ISBN == isbn);
            var borrower = Borrowers.FirstOrDefault(b => b.LibraryCardNumber ==
libraryCardNumber);

            if (book == null || borrower == null)
                throw new Exception("Book or Borrower not found.");

            borrower.ReturnBook(book);
        }

        public List<Book> ViewBooks()
        {
            return Books;
        }

        public List<Borrower> ViewBorrowers()
        {
            return Borrowers;
        }
    }
}
```

Borrower.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Demo_Library_Management
{
    public class Borrower
    {
        public string Name { get; set; }
        public string LibraryCardNumber { get; set; }
        public List<Book> BorrowedBooks { get; private set; }

        public Borrower(string name, string cardNumber)
        {
            Name = name;
            LibraryCardNumber = cardNumber;
            BorrowedBooks = new List<Book>();
        }

        public void BorrowBook(Book book)
        {
            book.Borrow();
            BorrowedBooks.Add(book);
        }

        public void ReturnBook(Book book)
        {
            book.Return();
            BorrowedBooks.Remove(book);
        }
    }
}
```

UnitTest1.cs

```
using NUnit.Framework;
using Demo_Library_Management;
using System.Linq;
using Demo_Library_Management;

namespace Demo_Library_Management.Tests
{
    public class LibraryTests
    {
        private Library library;

        [SetUp]
        public void Setup()
        {
            library = new Library();
        }
```

```
[Test]
public void TestAddBook()
{
    var book = new Book("C#", "John Doe", "123");
    library.AddBook(book);

    Assert.AreEqual(1, library.Books.Count);
    Assert.AreEqual("C#", library.Books[0].Title);
}

[Test]
public void TestRegisterBorrower()
{
    var borrower = new Borrower("Parth", "ABC123");
    library.RegisterBorrower(borrower);

    Assert.AreEqual(1, library.Borrowers.Count);
    Assert.AreEqual("Parth", library.Borrowers[0].Name);
}

[Test]
public void TestBorrowBook()
{
    var book = new Book("C#", "John Doe", "123");
    var borrower = new Borrower("Parth", "ABC123");
    library.AddBook(book);
    library.RegisterBorrower(borrower);

    library.BorrowBook("123", "ABC123");

    Assert.IsTrue(book.IsBorrowed);
    Assert.Contains(book, borrower.BorrowedBooks);
}

[Test]
public void TestReturnBook()
{
    var book = new Book("C#", "John Doe", "123");
    var borrower = new Borrower("Parth", "ABC123");
    library.AddBook(book);
    library.RegisterBorrower(borrower);
    library.BorrowBook("123", "ABC123");

    library.ReturnBook("123", "ABC123");

    Assert.IsFalse(book.IsBorrowed);
    Assert.IsFalse(borrower.BorrowedBooks.Contains(book));
}

[Test]
public void TestViewBooks()
{
    library.AddBook(new Book("Book1", "Author1", "111"));
    library.AddBook(new Book("Book2", "Author2", "222"));

    var books = library.ViewBooks();
    Assert.AreEqual(2, books.Count);
```

```
        }

        [Test]
        public void TestViewBorrowers()
        {
            library.RegisterBorrower(new Borrower("Parth", "111"));
            library.RegisterBorrower(new Borrower("Aryan", "222"));

            var borrowers = library.ViewBorrowers();
            Assert.AreEqual(2, borrowers.Count);
        }
    }
}
```

Output: