Book Class Hierarchy

```
abstract class Book {
    protected String title;
    protected String author;
    protected String isbn;
    protected boolean isAvailable;
    public Book(String title, String author, String isbn) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
        this.isAvailable = true;
    }
    public void displayInfo() {
        System.out.println("Title: " + title +
                ", Author: " + author +
                ", ISBN: " + isbn +
                ", Available: " + isAvailable);
    }
    public boolean isAvailable() {
        return isAvailable;
    }
    public void setAvailability(boolean status) {
        this.isAvailable = status;
    }
    public String getIsbn() {
        return isbn;
    }
}
```

Fiction Class

```
class Fiction extends Book {
    private String genre;
    public Fiction(String title, String author, String isbn, String genre) {
        super(title, author, isbn);
        this.genre = genre;
    }
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Genre: " + genre);
    }
}
```

Non Fiction class

```
class NonFiction extends Book {
    private String subject;
    public NonFiction(String title, String author, String isbn, String subject) {
        super(title, author, isbn);
        this.subject = subject;
    }
    @Override
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Subject: " + subject);
    }
}
```

Reference Class

```
class Reference extends Book {
    private String edition;
```

```
  public Reference(String title, String author, String isbn, String edition) {
      super(title, author, isbn);
      this.edition = edition;
   }
   @Override
   public void displayInfo() {
      super.displayInfo();
      System.out.println("Edition: " + edition);
   }
}
```

Library Class

```
import java.util.*;
class Library {
   private Map<String, Book> books = new HashMap<>();
   public void addBook(Book book) {
      if (books.containsKey(book.getIsbn())) {
         throw new IllegalArgumentException("Duplicate book ISBN not allowed.");
      }
      books.put(book.getIsbn(), book);
   }
   public void removeBook(String isbn) {
      if (!books.containsKey(isbn)) {
         throw new NoSuchElementException("Book not found in library.");
      }
     books.remove(isbn);
   }
   public Book searchBook(String isbn) {
```

```
        if (!books.containsKey(isbn)) {
            throw new NoSuchElementException("Book not found.");
        }
        return books.get(isbn);
    }
    public void displayAllBooks() {
        for (Book book : books.values()) {
            book.displayInfo();
            System.out.println("--------------------");
        }
    }
}
```

User Class

```
class User {
    private String name;
    private String libraryCardNumber;
    private List<Book> borrowedBooks;
    private static final int MAX_LIMIT = 3;
    public User(String name, String libraryCardNumber) {
        this.name = name;
        this.libraryCardNumber = libraryCardNumber;
        this.borrowedBooks = new ArrayList<>();
    }
    public void borrowBook(Book book) {
        if (!book.isAvailable()) {
            throw new IllegalStateException("Book is currently unavailable.");
        }
```

```
        if (borrowedBooks.size() >= MAX_LIMIT) {
            throw new IllegalStateException("Borrow limit exceeded.");
        }
        borrowedBooks.add(book);
        book.setAvailability(false);
    }
    public void returnBook(Book book) {
        if (!borrowedBooks.contains(book)) {
            throw new IllegalArgumentException("This book was not borrowed by user.");
        }
        borrowedBooks.remove(book);
        book.setAvailability(true);
    }
    public void displayBorrowedBooks() {
        System.out.println("Borrowed Books by " + name + ":");
        for (Book book : borrowedBooks) {
            book.displayInfo();
        }
    }
}
```