```
package library;

public interface Issueable {
	boolean issuebook();
	String returnbooks();
   boolean isAvailable();
}
package library;

public class Book implements Issueable {
 private String title;
 private String author;
 private String ISBN;
 private boolean available;

 public Book(String title,String author,String ISBN) {
	 this.title=title;
	 this.author=author;
	 this.ISBN=ISBN;
	 this.available=true;
	 }
@Override
public boolean issuebook() {
	if(!available) {
		return false;
	}
	available=false;
	return true;
}
@Override
public String returnbooks() {
```

```
        if(!available) {
                available=true;
                return "Return Successful";
        }
        return "Return failed:Book not borrowed";
}
@Override
public boolean isAvailable() {
        return available;
}
public String getTitle()
{
        return title;
}
public String getAuthor()
{
        return author;
}
public String getISBN()
{
        return ISBN;
}
}
package library;
import java.util.*;

public class Library {
private List<Book> bookList = new ArrayList<>();
public void includeBook(Book newBook) {
        bookList.add(newBook);
}
```

```
public String borrowBook(String ISBN) {
            for(Book b : bookList) {
                        if (ISBN.equals(b.getISBN())) {
          if (b.issuebook()) {
             return "Publication checkout successful";
          } else {
             return "Publication currently unavailable";
          }
       }
   }
   return "Publication not in collection";
}

public String bringBackBook(String ISBN) {
   for (Book b : bookList) {
      if (ISBN.equals(b.getISBN())) {
         return b.returnbooks();
      }
   }
   return "Publication not in collection";
}

public int collectionSize() {
   return bookList.size();
}

public List<Book> getAllPublications() {
   return bookList;
}
}
package library;
```

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

 class LibraryUnitTests {
    private Library libraryInstance;
    private Book samplePublication;

    @BeforeEach
    void initializeTestingEnvironment() {
        libraryInstance = new Library();
        samplePublication = new Book("Digital Fortress", "Dan Brown", "978-0552151696");
        libraryInstance.includeBook(samplePublication);
    }

    @Test
    @DisplayName("Verify successful book checkout process")
    void checkoutProcessVerification() {
        String firstAttempt = libraryInstance.borrowBook("978-0552151696");
        String secondAttempt = libraryInstance.borrowBook("978-0552151696");

        assertAll("Checkout process verification",
            () -> assertEquals("Publication checkout successful", firstAttempt),
            () -> assertEquals("Publication currently unavailable", secondAttempt)
        );
    }

    @Test
    @DisplayName("Validate book return functionality")
```

```
void returnProcessValidation() {
    libraryInstance.borrowBook("978-0552151696");

    String firstReturn = libraryInstance.bringBackBook("978-0552151696");
    String secondReturn = libraryInstance.bringBackBook("978-0552151696");

    assertAll("Return process validation",
        () -> assertEquals("Return Successful", firstReturn),
        () -> assertEquals("Return failed:Book not borrowed", secondReturn)
    );
}

@Test
@DisplayName("Test publication availability status")
void availabilityStatusCheck() {
    assertTrue(samplePublication.isAvailable(),
            "Publication should be initially available");

    libraryInstance.borrowBook("978-0552151696");

    assertFalse(samplePublication.isAvailable(),
             "Publication should be unavailable after checkout");
}

@Test
@DisplayName("Handle non-existent publications")
void nonExistentPublicationHandling() {
    String checkoutResult = libraryInstance.borrowBook("INVALID-ISBN");
    String returnResult = libraryInstance.bringBackBook("INVALID-ISBN");

    assertAll("Non-existent publication handling",
```

```
            () -> assertEquals("Publication not in collection", checkoutResult),
            () -> assertEquals("Publication not in collection", returnResult)
        );
    }

    @Test
    @DisplayName("Test collection inventory methods")
    void inventoryManagementTests() {
        assertEquals(1, libraryInstance.collectionSize(),
                "Initial collection size should be 1");

        assertFalse(libraryInstance.getAllPublications().isEmpty(),
                "Publications list should not be empty");
    }
}
```