A4 BUG2 REPORT

Name: Ty Saunders

Student ID: 11748199

Assignment details: Debugging task

Bug 2: Loan limit being enforced incorrectly

When a Patron borrows items, the loan limit is supposed to be enforced after 2 items, but instead it is being enforced after 3 items. When a Patron returns later, they can borrow yet another item before the loan limit is enforced again. They can do this over and over until the library runs out of books.

Contents – Click links to jump to section:

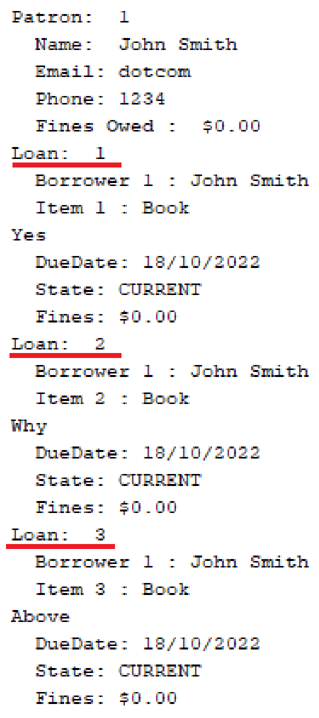
1. Demonstration:

Issue 1:

When a patron with 0 loans borrows multiple items:

Loan limit should be triggered after 2 items

But it is not triggered until after 3 items

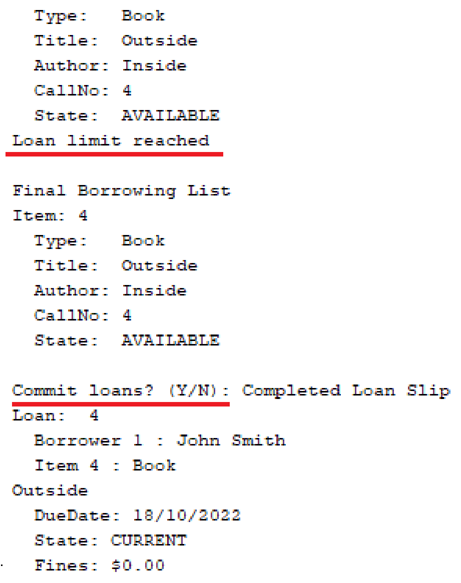


```
Patron:  1
  Name:  John Smith
  Email: dotcom
  Phone: 1234
  Fines Owed :  $0.00
Loan:  1
  Borrower 1 : John Smith
  Item 1 : Book
Yes
  DueDate: 18/10/2022
  State: CURRENT
  Fines: $0.00
Loan:  2
  Borrower 1 : John Smith
  Item 2 : Book
Why
  DueDate: 18/10/2022
  State: CURRENT
  Fines: $0.00
Loan:  3
  Borrower 1 : John Smith
  Item 3 : Book
Above
  DueDate: 18/10/2022
  State: CURRENT
  Fines: $0.00
```

Issue 2:

When a patron who has exceeded the loan limit returns later to try to borrow items again

It is incorrectly allowing them to borrow 1 more item before enforcing the limit.



```
  Type:   Book
  Title:  Outside
  Author: Inside
  CallNo: 4
  State:  AVAILABLE
Loan limit reached

Final Borrowing List
Item: 4
  Type:   Book
  Title:  Outside
  Author: Inside
  CallNo: 4
  State:  AVAILABLE

Commit loans? (Y/N): Completed Loan Slip
Loan:  4
  Borrower 1 : John Smith
  Item 4 : Book
Outside
  DueDate: 18/10/2022
  State: CURRENT
  Fines: $0.00
```

2. Replication

System Test:

The steps to manually replicate the error are contained in a separate document in the same directory as this document: Bug2SystemTest.pdf.

Here is a link to the document on github for quick reference: Bug2SystemTest.pdf

In summary the steps are to begin with a patron who has no loans, then attempt to borrow items until stopped. For issue 2, the steps are to return later and try to borrow more items.

3. Simplification

This section uses Hypothesis / Test / Result process to narrow down the cause to the point where a simplified automated test can be written to reproduce it easily.

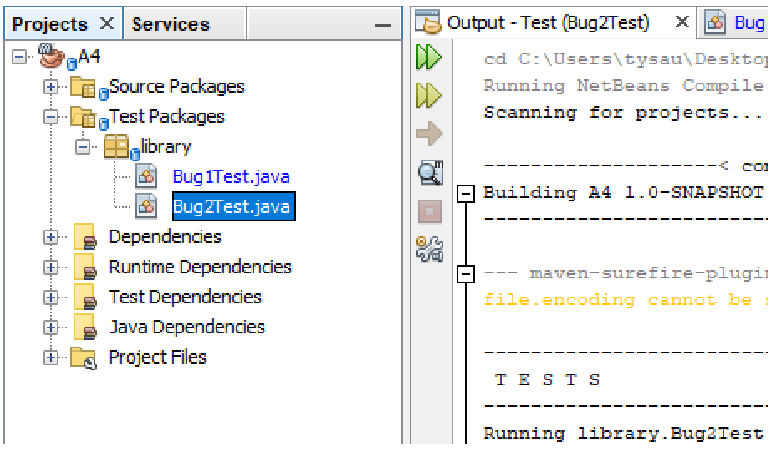
- H1 – BorrowItemControl is not correctly enforcing the loan limit
- T1 – Visually inspect BorrowItemControl to understand how the borrow process works
- R1 – UI sets control state to READY, control sets UI's state to READY, control.CardSwiped is called, which contains a line of code which calls library.canPatronBorrow method which is called once at the beginning of the borrow session. This method in Library needs to be checked. There is another method (itemScanned) which is also deciding if patron can continue to borrow after each item is added.

Result: Automated test can be done from BorrowItemControl class.

Automated Test:

Test case has been written to replicate this bug quickly. It can be run by opening the project, found in this directory: A4\src\test\java\library.Bug2Test.java

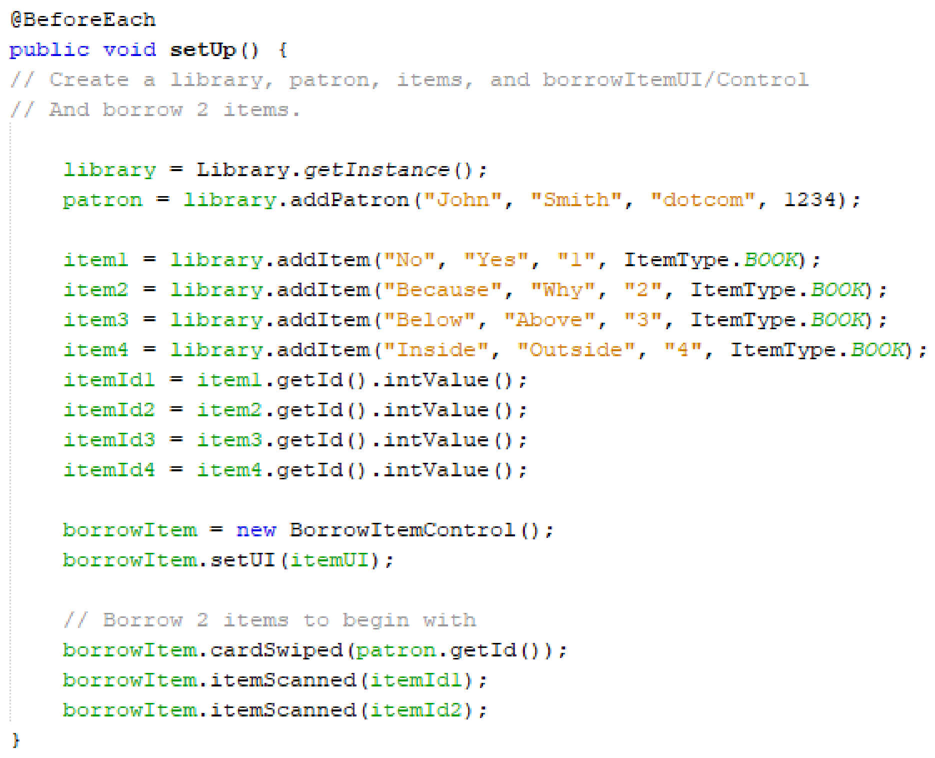
The test can be run by opening the project, right-click Bug2Test.java and click "Test".



```
cd C:\Users\tysau\Deskto
Running NetBeans Compile
Scanning for projects...

--------------------< co
Building A4 1.0-SNAPSHOT
------------------------

--- maven-surefire-plugi
file.encoding cannot be

------------------------
 T E S T S
------------------------
Running library.Bug2Test
```

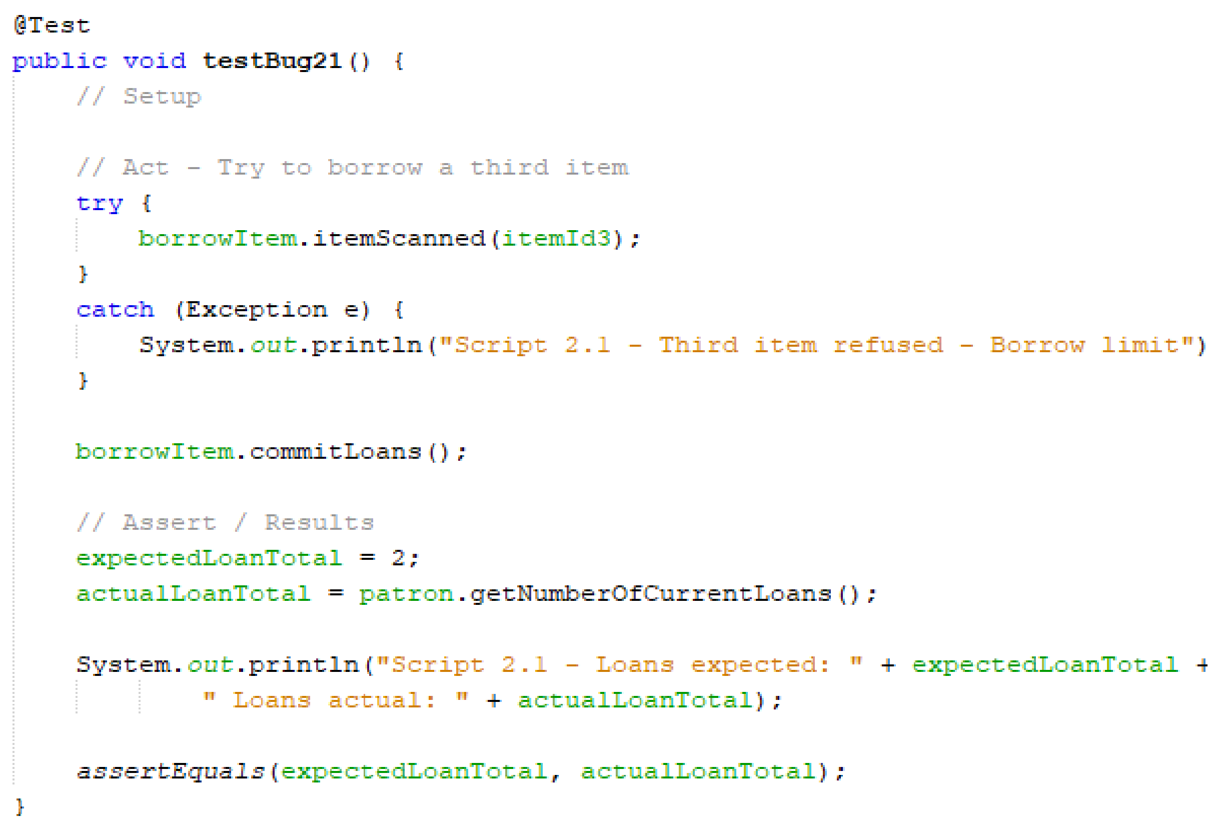
Screenshots of the test code:

Script set up:



```
@BeforeEach
public void setUp() {
// Create a library, patron, items, and borrowItemUI/Control
// And borrow 2 items.

    library = Library.getInstance();
    patron = library.addPatron("John", "Smith", "dotcom", 1234);

    item1 = library.addItem("No", "Yes", "1", ItemType.BOOK);
    item2 = library.addItem("Because", "Why", "2", ItemType.BOOK);
    item3 = library.addItem("Below", "Above", "3", ItemType.BOOK);
    item4 = library.addItem("Inside", "Outside", "4", ItemType.BOOK);
    itemId1 = item1.getId().intValue();
    itemId2 = item2.getId().intValue();
    itemId3 = item3.getId().intValue();
    itemId4 = item4.getId().intValue();

    borrowItem = new BorrowItemControl();
    borrowItem.setUI(itemUI);

    // Borrow 2 items to begin with
    borrowItem.cardSwiped(patron.getId());
    borrowItem.itemScanned(itemId1);
    borrowItem.itemScanned(itemId2);
}
```

Script 2.1

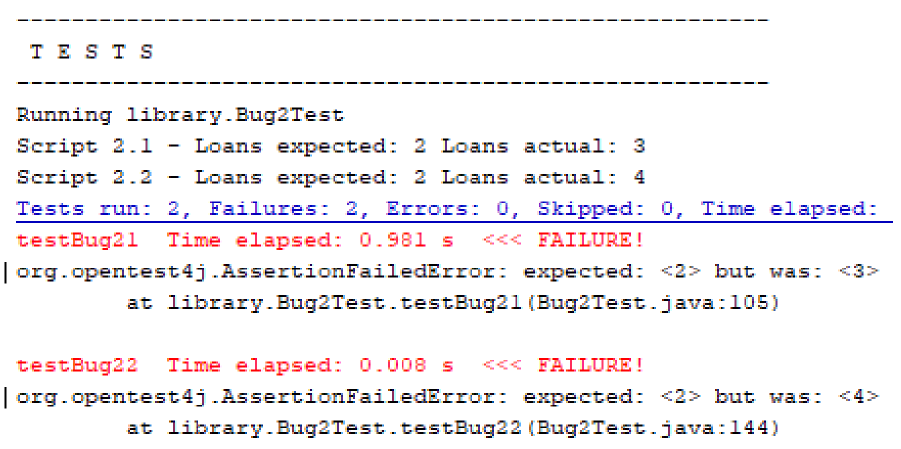


```
@Test
public void testBug21() {
    // Setup

    // Act - Try to borrow a third item
    try {
        borrowItem.itemScanned(itemId3);
    }
    catch (Exception e) {
        System.out.println("Script 2.1 - Third item refused - Borrow limit")
    }

    borrowItem.commitLoans();

    // Assert / Results
    expectedLoanTotal = 2;
    actualLoanTotal = patron.getNumberOfCurrentLoans();

    System.out.println("Script 2.1 - Loans expected: " + expectedLoanTotal +
            " Loans actual: " + actualLoanTotal);

    assertEquals(expectedLoanTotal, actualLoanTotal);
}
```

Script 2.2

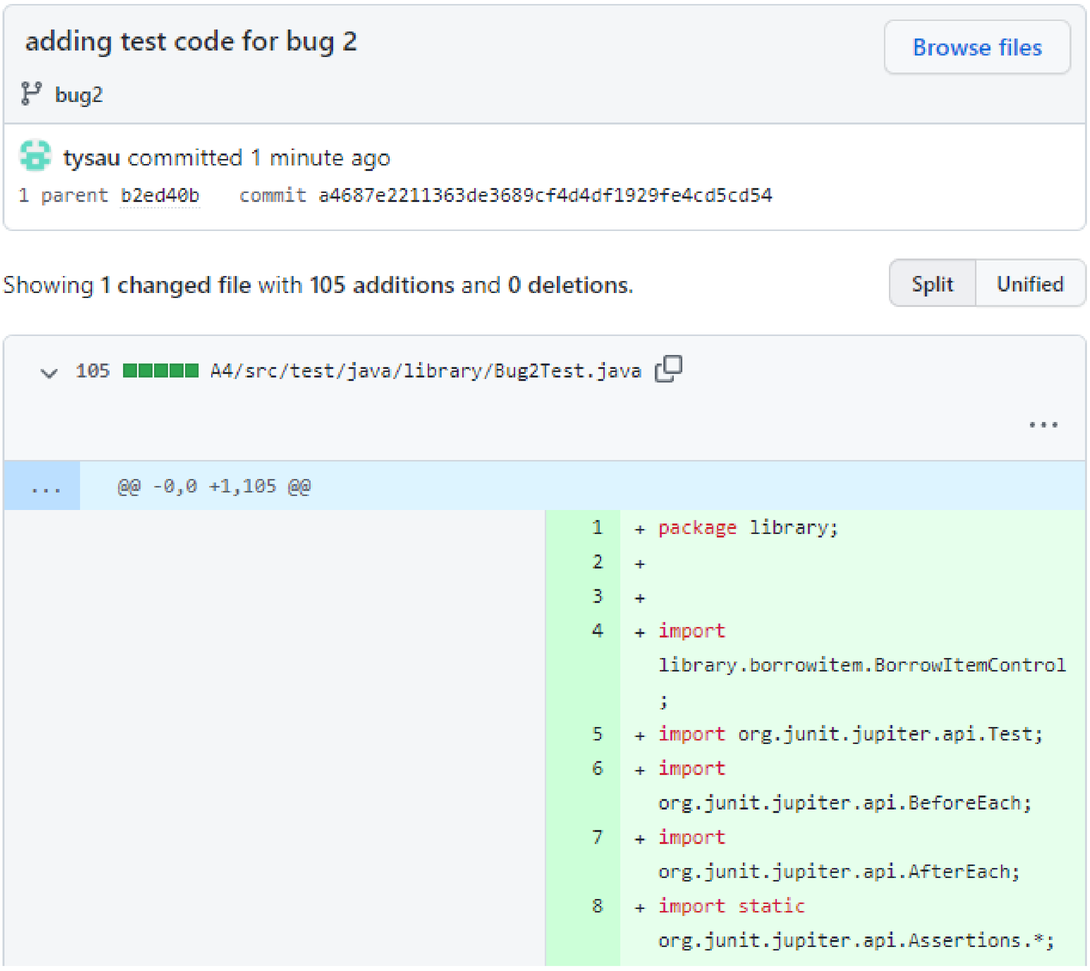
```java
@Test
public void testBug22() {
    // Setup

    // Act: Try to borrow a third item
    try {
        borrowItem.itemScanned(itemId3);
    }
    catch (Exception e) {
        System.out.println("Script 2.2 - Third item refused - Borrow limit")
    }
    borrowItem.commitLoans();

    // Act: Return again and swipe card
    borrowItem = new BorrowItemControl();
    borrowItem.setUI(itemUI);
    borrowItem.cardSwiped(patron.getId());

    // Try to borrow an extra item
    try {
        borrowItem.itemScanned(itemId4);
        borrowItem.commitLoans();
    }
    catch (Exception e) {
        System.out.println("Script 2.2 - Borrow extra item rejected");
    }

    // Assert / Results
    expectedLoanTotal = 2;
    actualLoanTotal = patron.getNumberOfCurrentLoans();
    System.out.println("Script 2.2 - Loans expected: " + expectedLoanTotal +
            " Loans actual: " + actualLoanTotal);

    assertEquals(expectedLoanTotal, actualLoanTotal);
}
```

Screenshot of the test being run and failing as expected before the fix:



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running library.Bug2Test
Script 2.1 - Loans expected: 2 Loans actual: 3
Script 2.2 - Loans expected: 2 Loans actual: 4
Tests run: 2, Failures: 2, Errors: 0, Skipped: 0, Time elapsed:
testBug21  Time elapsed: 0.981 s  <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <2> but was: <3>
        at library.Bug2Test.testBug21(Bug2Test.java:105)

testBug22  Time elapsed: 0.008 s  <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <2> but was: <4>
        at library.Bug2Test.testBug22(Bug2Test.java:144)
```

Version control: Screenshot of commit for the test code:



adding test code for bug 2

Browse files

bug2

tysau committed 1 minute ago

1 parent b2ed40b   commit a4687e2211363de3689cf4d4df1929fe4cd5cd54

Showing 1 changed file with 105 additions and 0 deletions.

Split Unified

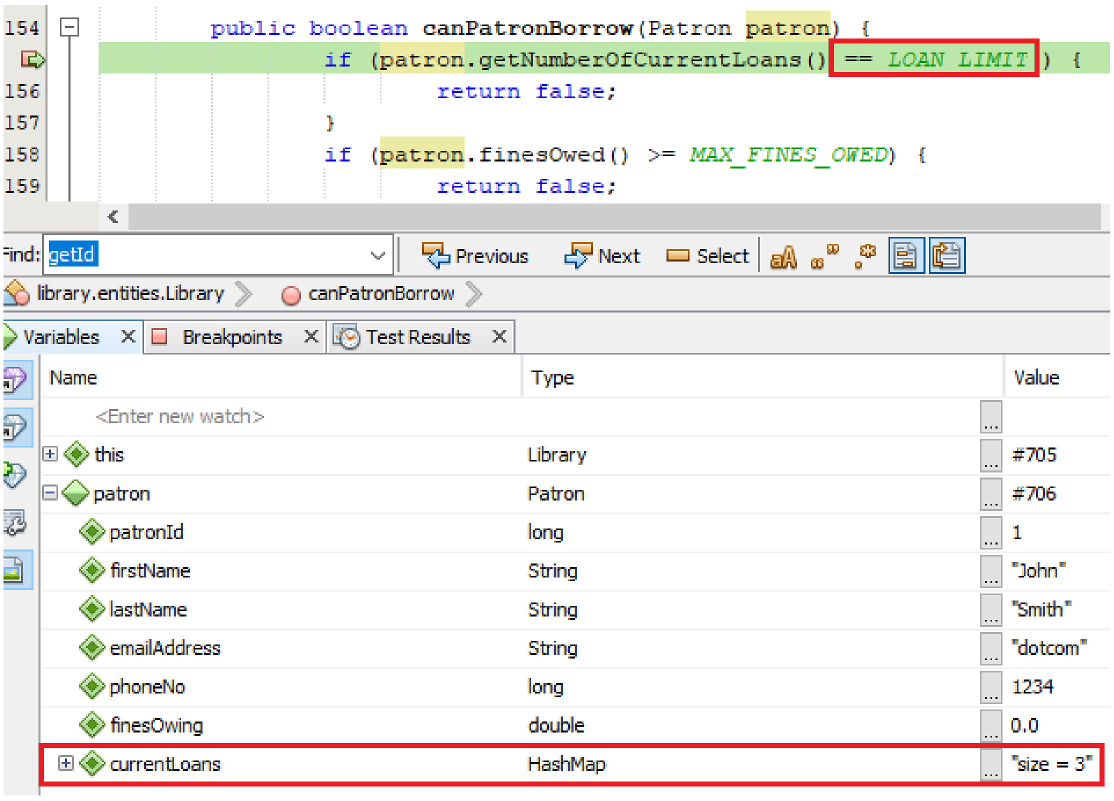
105 A4/src/test/java/library/Bug2Test.java

```
@@ -0,0 +1,105 @@
+ package library;
+
+
+ import library.borrowitem.BorrowItemControl;
+ import org.junit.jupiter.api.Test;
+ import org.junit.jupiter.api.BeforeEach;
+ import org.junit.jupiter.api.AfterEach;
+ import static org.junit.jupiter.api.Assertions.*;
```

Bug 2 Tracing:

In ReturnItemControl method CardSwiped is called, which contains a line of code which calls library.canPatronBorrow method. This method in Library needs to be checked.

- H1 – Library.canPatronBorrow is not correctly determining if patron can borrow.
- T1 – Run the test with break points before and after canPatronBorrow.
- R1 – True – canPatronBorrow is only stopping the patron from taking a loan if they have exactly 2 loans. If the patron somehow has 3 or more, they can continue to borrow. This is not the root cause because some other validation is responsible for letting the patron get above 3 loans, but it could be changed to >=2 for extra robustness.

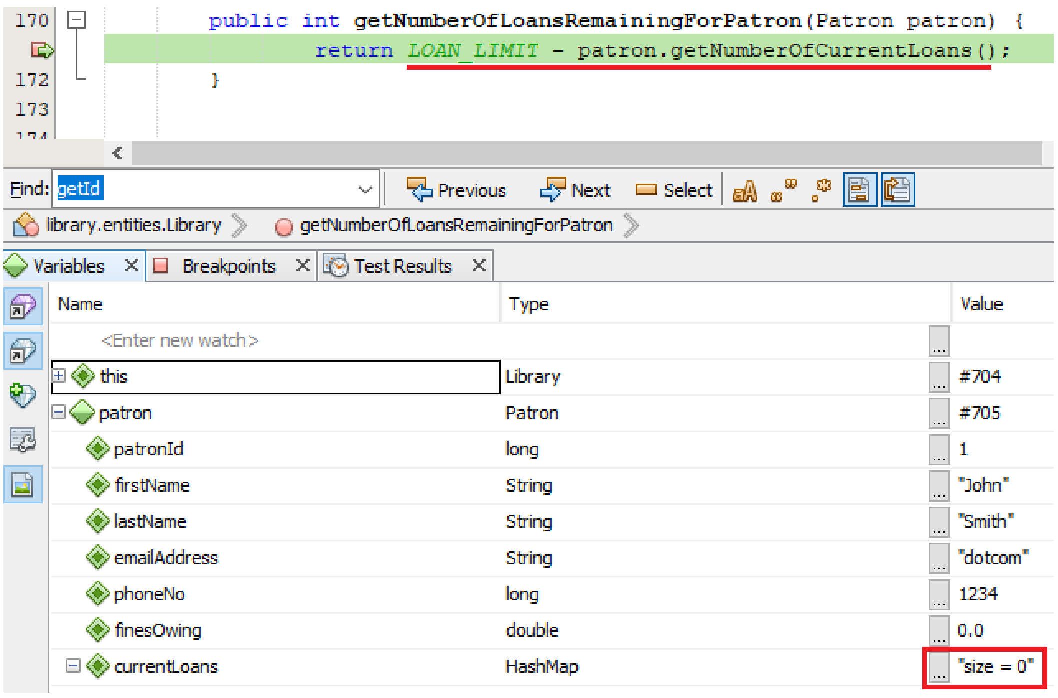


```
	public boolean canPatronBorrow(Patron patron) {
		if (patron.getNumberOfCurrentLoans() == LOAN_LIMIT) {
			return false;
		}
		if (patron.finesOwed() >= MAX_FINES_OWED) {
			return false;
```

Name	Type	Value
<Enter new watch>		
this	Library	#705
patron	Patron	#706
patronId	long	1
firstName	String	"John"
lastName	String	"Smith"
emailAddress	String	"dotcom"
phoneNo	long	1234
finesOwing	double	0.0
currentLoans	HashMap	"size = 3"

- H2 – BorrowItemControl itemScanned method is not correctly enforcing the loan limit
- T2 – Visually inspect itemScanned method to understand how it works
- R2 – See that line 79 is checking the result of a method in library: getNumberOfLoansRemainingForPatron minus the pending Item List and deciding to enforce the item limit based on that. Need to inspect: getNumberOfLoansRemainingForPatron.

- H3 – getNumberOfLoansRemainingForPatron could be returning incorrect limit
- T3 – Visually inspect what getNumberOfLoansRemainingForPatron is doing
- R3 – FALSE: The method is taking the LOAN_LIMIT which is a constant 2 and subtracting the current loans the patron has. In this case we have a new patron borrowing for the first time who will have 0 loans to this method will be returning 2. This method is returning correctly.



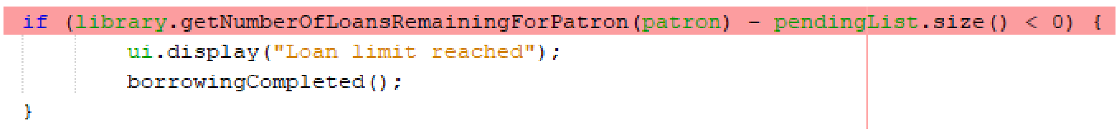
```
    public int getNumberOfLoansRemainingForPatron(Patron patron) {
        return LOAN_LIMIT - patron.getNumberOfCurrentLoans();
    }
```

Find: getId | Previous | Next | Select

library.entities.Library > getNumberOfLoansRemainingForPatron

Name	Type	Value
<Enter new watch>		
this	Library	#704
patron	Patron	#705
patronId	long	1
firstName	String	"John"
lastName	String	"Smith"
emailAddress	String	"dotcom"
phoneNo	long	1234
finesOwing	double	0.0
currentLoans	HashMap	"size = 0"

- H4 – BorrowItemControl itemScanned method is using incorrect logic to determine if patron can borrow.
- T4 – Visually inspect itemScanned again to check the logic.
- R4 – True - We can see that the line of code to determine if a patron can borrow is getting amount of loans remaining and subtracting the pending list size, if this results in less than zero, the loan limit will be enforced. Less than zero means the patron can have minus 1 loans remaining before being stopped.



```
if (library.getNumberOfLoansRemainingForPatron(patron) - pendingList.size() < 0) {
        ui.display("Loan limit reached");
        borrowingCompleted();
}
```

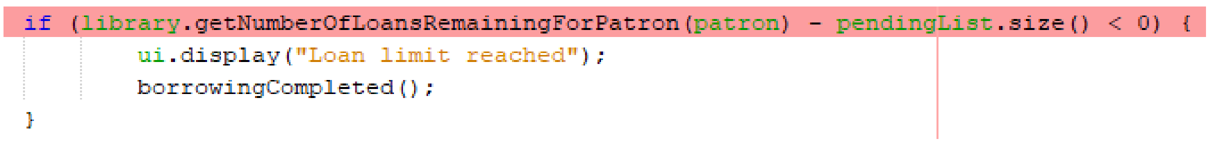
In the case of this bug the values passed to getNumberOfLoansRemaining are:

Library.getNumberOfLoansRemainingForPatron	**2**
pendingList.size()	**2**
Library.getNumberOfLoansRemainingForPatron - pendingList.size()	**0**

0 is not less than 0.
Therefore the code below it to enforce the loan limit will not be executed and the patron can borrow another item.

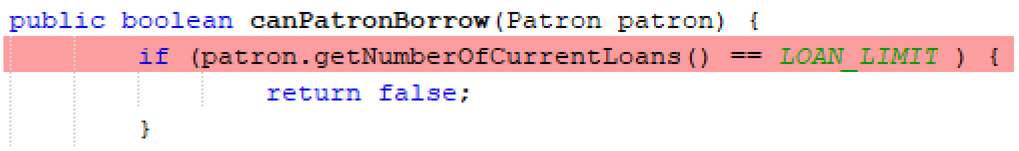
Conclusion of bug tracing:

The root cause –
Maths to calculate if a patron can continue to borrow additional items is wrong in BorrowItemControl.itemScanned() method.



```
if (library.getNumberOfLoansRemainingForPatron(patron) - pendingList.size() < 0) {
        ui.display("Loan limit reached");
        borrowingCompleted();
}
```

Secondary contributor –
If patron somehow manages to exceed 2 loans, they can borrow infinitely after this on separate visits, due to the logic in Library.canPatronBorrow method.



```
public boolean canPatronBorrow(Patron patron) {
        if (patron.getNumberOfCurrentLoans() == LOAN_LIMIT ) {
                return false;
        }
```

Resolution:

Fix: Update the code in BorrowItemControl line 79 and Library line 156 to rectify the issues above.

Fix Plan:

- Use a separate code branch other than main.
- Apply Fix 1, run test to confirm bug 2.2 (returning later to borrow more) is resolved.
- Apply Fix 2, run test to confirm both bug 2.1 (borrowing more than 2 items) and 2.2 are passing.
- Run regression test for bug 1 to confirm it is still fixed.

FIX 1: Library:

Change line 156 to correct the loan limit calculation logic by changing:
== 0
To:
>= 0

Then run test case 2.2 to confirm patron is unable to borrow a 4th item on a later visit.

FIX 2: BorrowItemControl:

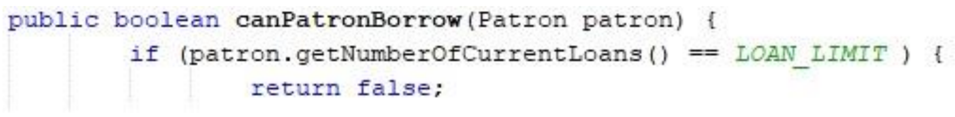
Change line 79 to correct the loan limit calculation logic in line 79 by changing:
< 0
To:
<= 0

Then run test case 2.1 to confirm patron is unable to borrow an extra item in their first borrowing session.

Validation:

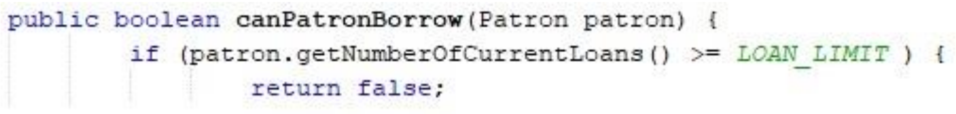
Fix 1 result (borrowing more in later visit):

Code before:



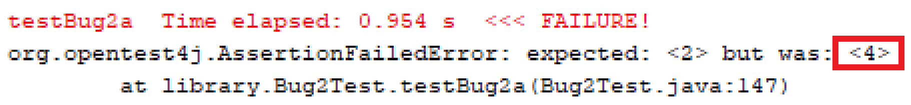
```
public boolean canPatronBorrow(Patron patron) {
        if (patron.getNumberOfCurrentLoans() == LOAN_LIMIT ) {
                return false;
```

Code after:



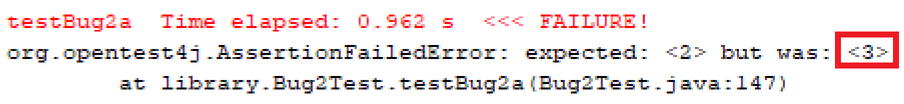
```
public boolean canPatronBorrow(Patron patron) {
        if (patron.getNumberOfCurrentLoans() >= LOAN_LIMIT ) {
                return false;
```

Test results before – 4 items borrowed:



```
testBug2a  Time elapsed: 0.954 s  <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <2> but was: <4>
        at library.Bug2Test.testBug2a(Bug2Test.java:147)
```

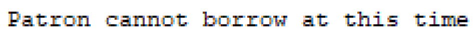
Test results after – 3 items borrowed:



```
testBug2a  Time elapsed: 0.962 s  <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <2> but was: <3>
        at library.Bug2Test.testBug2a(Bug2Test.java:147)
```

User test results after fix:

Patron is denied from borrowing later if they have exceeded the loan limit

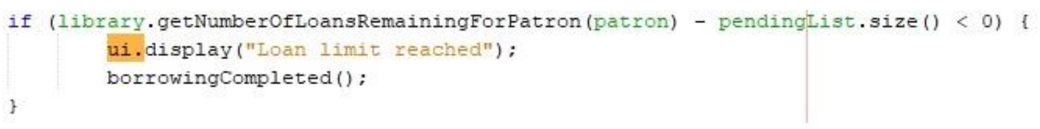


```
Patron cannot borrow at this time
```

Result: FIXED - Borrowing extra items on later visits is resolved

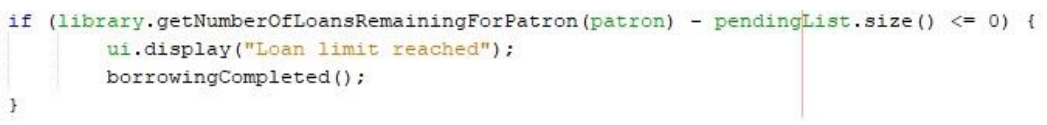
Fix 2 result (Borrowing more than 2 items)

Code before:



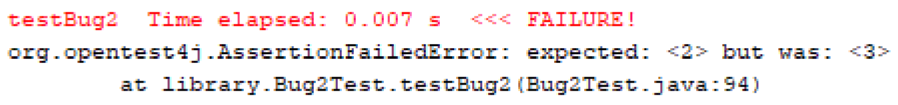
```
if (library.getNumberOfLoansRemainingForPatron(patron) - pendingList.size() < 0) {
        ui.display("Loan limit reached");
        borrowingCompleted();
}
```

Code after:



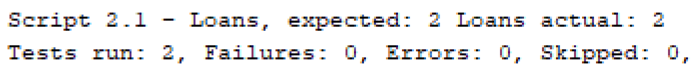
```
if (library.getNumberOfLoansRemainingForPatron(patron) - pendingList.size() <= 0) {
        ui.display("Loan limit reached");
        borrowingCompleted();
}
```

Test results before fix:



```
testBug2  Time elapsed: 0.007 s  <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <2> but was: <3>
        at library.Bug2Test.testBug2(Bug2Test.java:94)
```

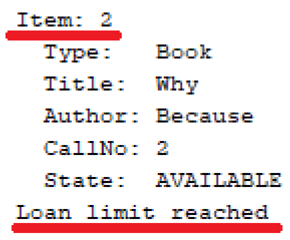
Test results after fix – the test case is now passing:



```
Script 2.1 - Loans, expected: 2 Loans actual: 2
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0,
```

User test results after fix:

Patron can borrow only 2 items – PASS.

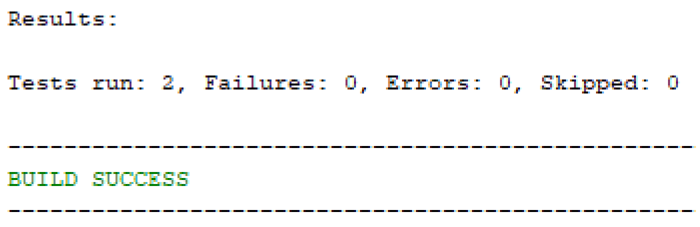


```
 Item: 2
   Type:    Book
   Title:   Why
   Author:  Because
   CallNo:  2
   State:   AVAILABLE
 Loan limit reached
```

Result: **FIXED** - Borrowing more than item limit in a single visit is resolved.

<u>Run all tests for Bug 2:</u>

Both tests are passing:



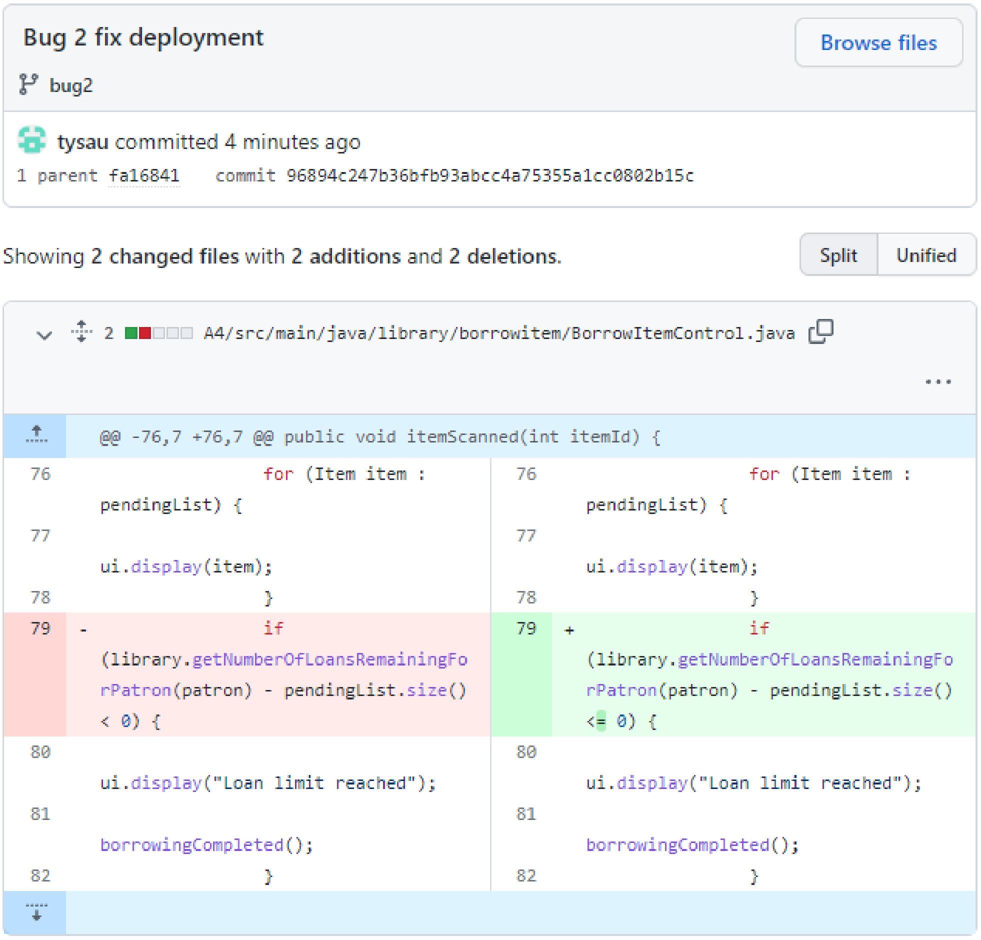
```
Results:

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------
BUILD SUCCESS
------------------------------------------------
```

<u>Regression test – confirm Bug 1 is still fixed:</u>

<u>Bug 1 is still passing:</u>

```
fines, quoted: 1.0 actual: 1.0
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0,

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------
BUILD SUCCESS
------------------------------------------------
```

<u>Fix deployed, committed to remote repo - GitHub commit screenshot:</u>



Bug 2 fix deployment

bug2

tysau committed 4 minutes ago

1 parent fa16841   commit 96894c247b36bfb93abcc4a75355a1cc0802b15c

Showing 2 changed files with 2 additions and 2 deletions.

2 A4/src/main/java/library/borrowitem/BorrowItemControl.java

```diff
@@ -76,7 +76,7 @@ public void itemScanned(int itemId) {
                 for (Item item : pendingList) {
                     ui.display(item);
                 }
-                if (library.getNumberOfLoansRemainingForPatron(patron) - pendingList.size() < 0) {
+                if (library.getNumberOfLoansRemainingForPatron(patron) - pendingList.size() <= 0) {
                     ui.display("Loan limit reached");
                     borrowingCompleted();
                 }
```