# A4 BUG2 REPORT

# Name: Ty Saunders

Student ID: 11748199

# Assignment details: Debugging task

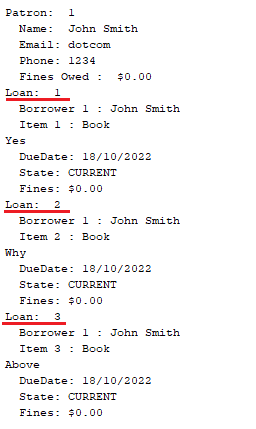
# **Bug 2 Report:**

## Bug 2: Loan Limit is being exceeded.

When an Patron borrows items, the loan limit is supposed to be enforced after 2 items, but instead it is being enforced after 3 items. When Patron returns again later, they are able to 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.

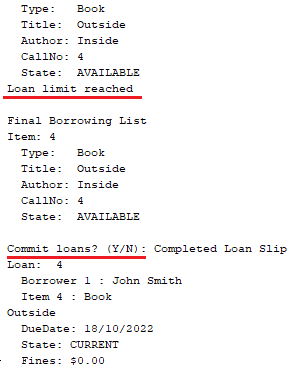
**To reproduce bug 2, issue 1:**

Starting with a patron with 0 loans, keep borrowing items until loan limit reach is displayed – this is incorrectly happening after 3 items instead of 2.



**To reproduce bug 2, issue 2:**

Starting with a patron who has exceeded the loan limit, try to borrow item again , it is incorrectly allowing them to borrow 1 more item before enforcing the limit.

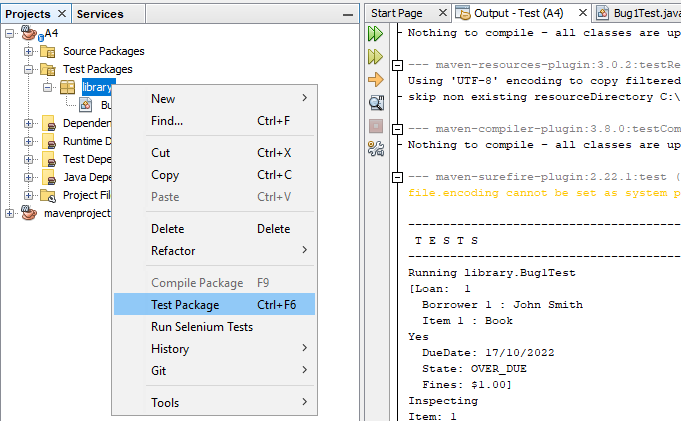


# **Bug 1 Replication:**

**System Test:** The steps to manually reproduce the error are in **Bug2SystemTest.docx**.

**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**

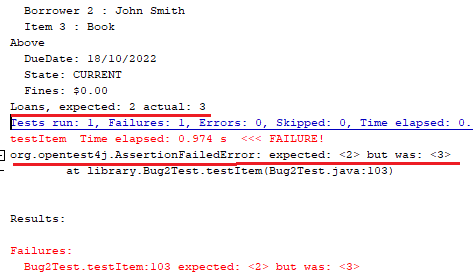
The test can be run by opening the project, right-click the library test package and click “Test Package”:



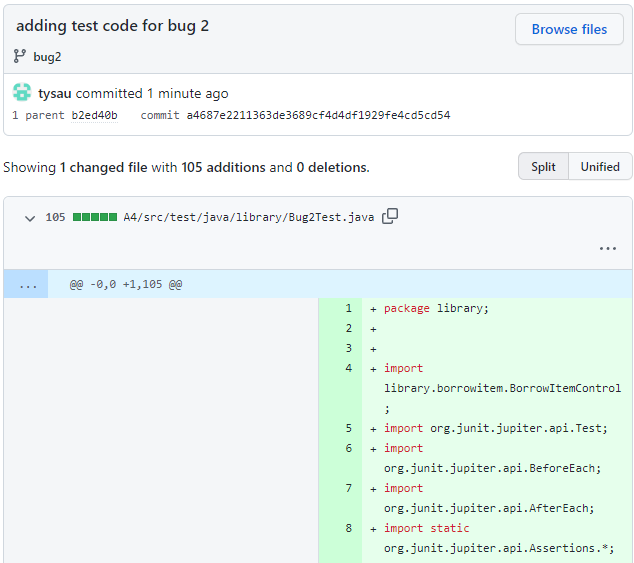
**A screenshot of the test code:**

# 

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



Version control: Screenshot of commit for the test code:



# **Bug 2 Tracing:**

**H0 – BorrowItemControl is not correctly enforcing the loan limit**

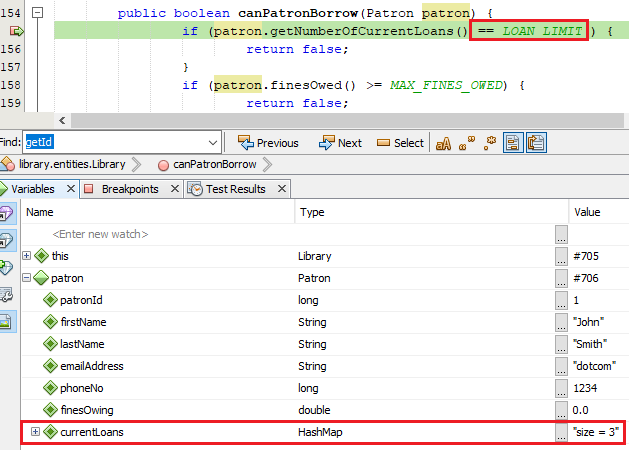
T0 – Visually inspect BorrowItemControl to understand how the borrow process works

R0 – 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. 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.



**H2 – BorrowItemControl itemScanned method is not correctly enforcing the loan limit**

T2 – Visually inspect itemScanned method to understand how it works

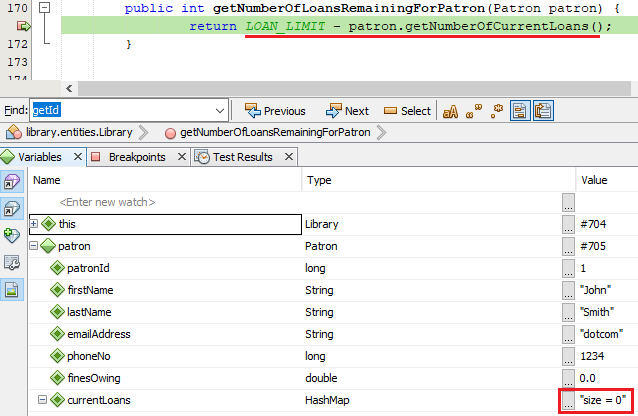
R2 – See that line 79 is checking the result of a method in library: getNumberOfLoansRemainingForPatron minus the pending Item List and deciding to enforce the item limit based on that. Need to inspect

getNumberOfLoansRemainingForPatron.

**H3 –** getNumberOfLoansRemainingForPatron could be returning incorrect limit

T3 – Visually inspect what getNumberOfLoansRemainingForPatron is doing

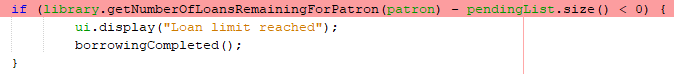
R3 – The method is taking the LOAN\_LIMIT which is a constant 2 and subtracting the current loans the patron has. In this case we have a new patron borrowing for the first time who will have 0 loans to this method will be returning 2. This method is returning correctly.



**H4 – BorrowItemControl itemScanned method incorrect logic to decide 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.



**In the case of this bug these values will be:**

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

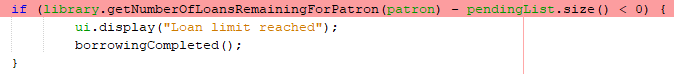
0 is not less than 0;

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:**

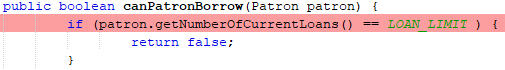
**The root cause –**

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



**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.



**Resolution**

Update the code in BorrowItemControl line 79 and Library line 156 to rectify these issues.

**Fix details:**

**Plan:**

**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.**

**Results after fix deployment:**

**Bug 2.2 (borrowing more in later visit):**

**Before fix test results:**

****

**After fix – the excess has reduced to 3:**

****

**Bug 2.1 (Borrowing more than 2 items)**

**Before fix test results:**

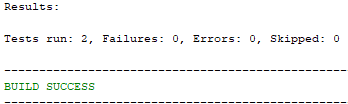
****

**After fix – the test case is now passing:**

****

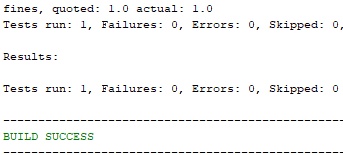
**Run all tests for Bug 2:**

**Both tests are passing:**

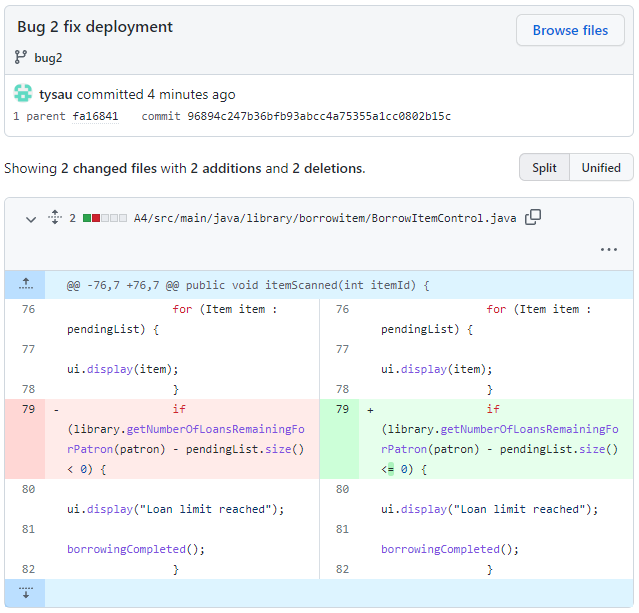
****

**Regression test – confirm Bug 1 is still fixed:**

**Bug 1 is still passing:**

****

**Fix deployed, committed to remote repo - Github commit screenshot:**

****