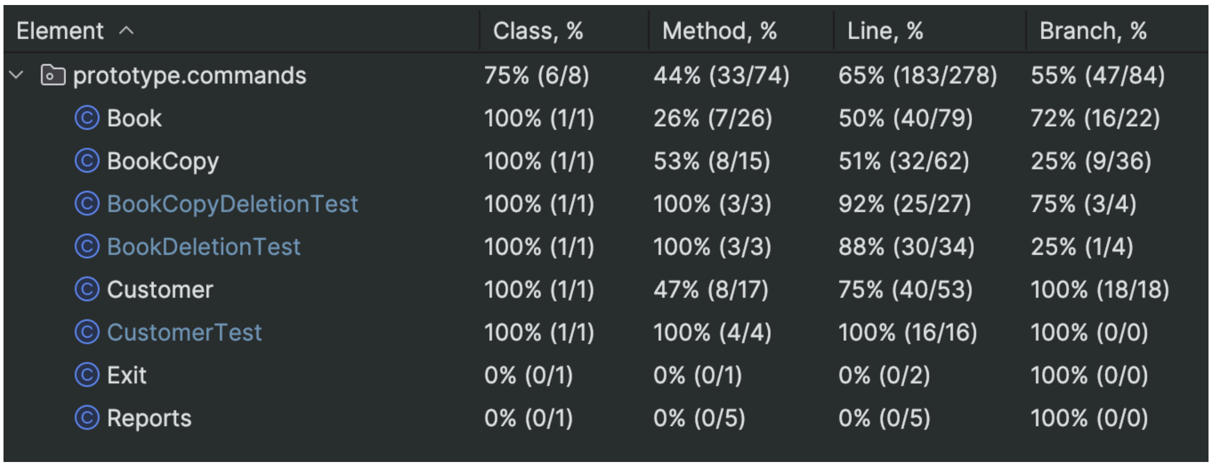
Test Case Coverage Report

Overview Code Coverage



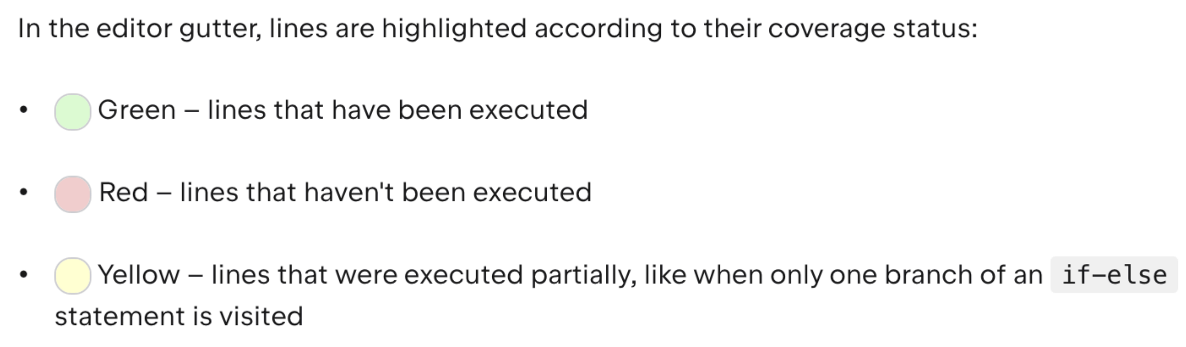
Element	Class, %	Method, %	Line, %	Branch, %
prototype.commands	75% (6/8)	44% (33/74)	65% (183/278)	55% (47/84)
Book	100% (1/1)	26% (7/26)	50% (40/79)	72% (16/22)
BookCopy	100% (1/1)	53% (8/15)	51% (32/62)	25% (9/36)
BookCopyDeletionTest	100% (1/1)	100% (3/3)	92% (25/27)	75% (3/4)
BookDeletionTest	100% (1/1)	100% (3/3)	88% (30/34)	25% (1/4)
Customer	100% (1/1)	47% (8/17)	75% (40/53)	100% (18/18)
CustomerTest	100% (1/1)	100% (4/4)	100% (16/16)	100% (0/0)
Exit	0% (0/1)	0% (0/1)	0% (0/2)	100% (0/0)
Reports	0% (0/1)	0% (0/5)	0% (0/5)	100% (0/0)

(Disclaimer: somehow the test code coverage in IntelliJ only worked when the test classes themselves also were in the commands directory)

Regarding the important classes Book, BookCopy and Customer there are some differences regarding the code coverage.

Book

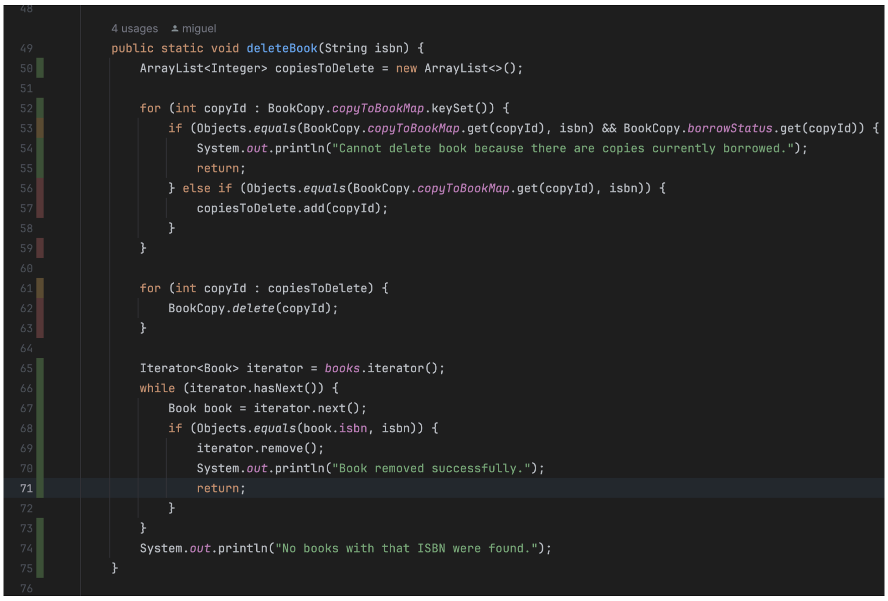
The test cases cover 26% of the methods, 50% of the code lines and 72% of the branches of this class. Though it must be added (also for the other two classes) that since every getter and setter counts as a method the percentage isn't that evident regarding the important methods of the class.
So let's have a closer look at some methods. While some methods are covered completely there are also some methods which are only covered partly.
But before I want to quickly highlight the color code as stated in the IntelliJ documentation (Ref.: https://www.jetbrains.com/help/idea/code-coverage.html#read_the_coverage_data):



In the editor gutter, lines are highlighted according to their coverage status:

- 🟩 Green – lines that have been executed
- 🟥 Red – lines that haven't been executed
- 🟨 Yellow – lines that were executed partially, like when only one branch of an `if-else` statement is visited

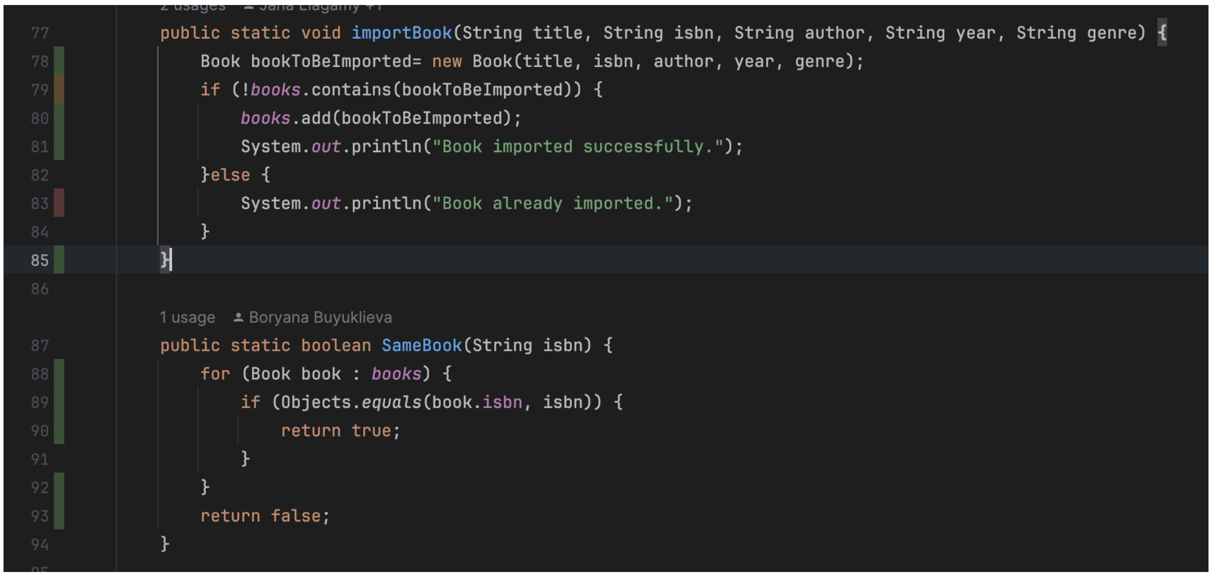
DeleteBook method



```
48
        4 usages   miguel
49      public static void deleteBook(String isbn) {
50          ArrayList<Integer> copiesToDelete = new ArrayList<>();
51
52          for (int copyId : BookCopy.copyToBookMap.keySet()) {
53              if (Objects.equals(BookCopy.copyToBookMap.get(copyId), isbn) && BookCopy.borrowStatus.get(copyId)) {
54                  System.out.println("Cannot delete book because there are copies currently borrowed.");
55                  return;
56              } else if (Objects.equals(BookCopy.copyToBookMap.get(copyId), isbn)) {
57                  copiesToDelete.add(copyId);
58              }
59          }
60
61          for (int copyId : copiesToDelete) {
62              BookCopy.delete(copyId);
63          }
64
65          Iterator<Book> iterator = books.iterator();
66          while (iterator.hasNext()) {
67              Book book = iterator.next();
68              if (Objects.equals(book.isbn, isbn)) {
69                  iterator.remove();
70                  System.out.println("Book removed successfully.");
71                  return;
72              }
73          }
74          System.out.println("No books with that ISBN were found.");
75      }
76
```

This method shows in a very good manner that there are parts which are covered completely by the test cases (green), covered only partially (yellow) and that there are even parts which aren't covered at all (e. g. when there's a condition which isn't met like in line 56).

ImportBook and SameBook Method



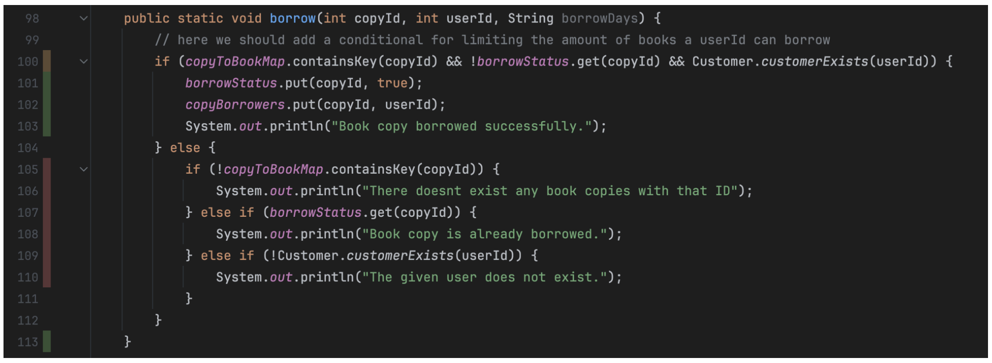
```
77      public static void importBook(String title, String isbn, String author, String year, String genre) {
78          Book bookToBeImported= new Book(title, isbn, author, year, genre);
79          if (!books.contains(bookToBeImported)) {
80              books.add(bookToBeImported);
81              System.out.println("Book imported successfully.");
82          }else {
83              System.out.println("Book already imported.");
84          }
85      }
86
        1 usage   Boryana Buyuklieva
87      public static boolean SameBook(String isbn) {
88          for (Book book : books) {
89              if (Objects.equals(book.isbn, isbn)) {
90                  return true;
91              }
92          }
93          return false;
94      }
```

This also applies to the importBook method as the case that the Book was already imported doesn't seem to be covered by the tests, which is also the reason why line 79 is marked yellow since there only one branch of this condition seems to be executed when running the tests.

BookCopy

The test cases cover 53% of the methods, 51% of the code lines and 25% of the branches of this class.

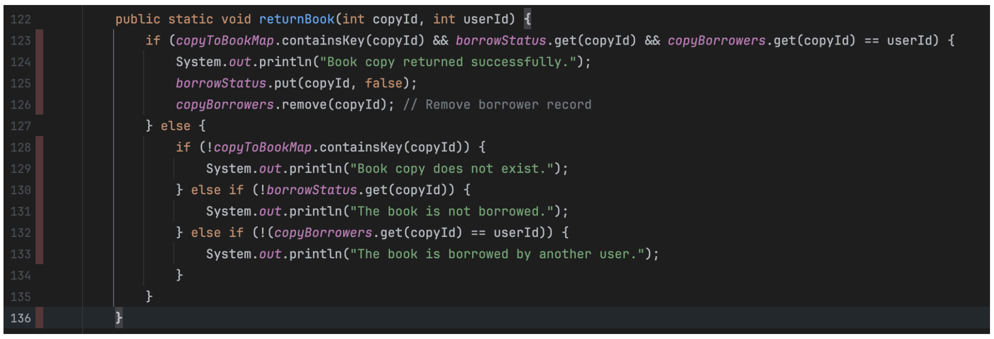
Borrow Method



```java
98   public static void borrow(int copyId, int userId, String borrowDays) {
99       // here we should add a conditional for limiting the amount of books a userId can borrow
100      if (copyToBookMap.containsKey(copyId) && !borrowStatus.get(copyId) && Customer.customerExists(userId)) {
101          borrowStatus.put(copyId, true);
102          copyBorrowers.put(copyId, userId);
103          System.out.println("Book copy borrowed successfully.");
104      } else {
105          if (!copyToBookMap.containsKey(copyId)) {
106              System.out.println("There doesnt exist any book copies with that ID");
107          } else if (borrowStatus.get(copyId)) {
108              System.out.println("Book copy is already borrowed.");
109          } else if (!Customer.customerExists(userId)) {
110              System.out.println("The given user does not exist.");
111          }
112      }
113  }
```

One reason for this branch percentage seems to be the borrow method, since the method in line 100 contains three different conditions which are checked. But the tests themselves seem to cover only the case when all three conditions are met.

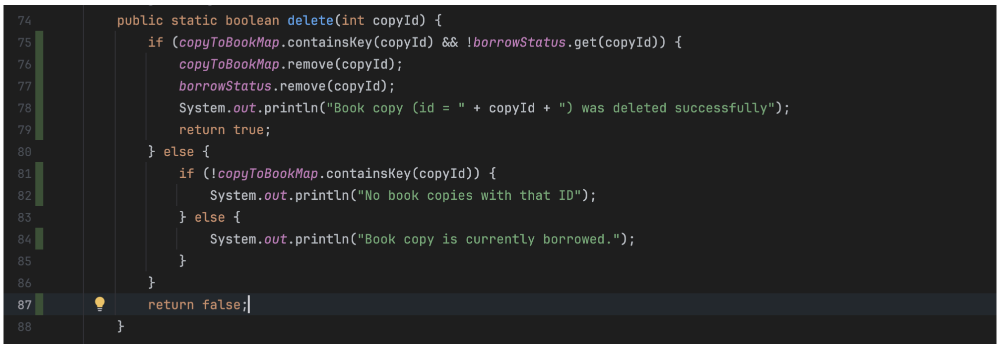
ReturnBook Method



```java
public static void returnBook(int copyId, int userId) {
    if (copyToBookMap.containsKey(copyId) && borrowStatus.get(copyId) && copyBorrowers.get(copyId) == userId) {
        System.out.println("Book copy returned successfully.");
        borrowStatus.put(copyId, false);
        copyBorrowers.remove(copyId); // Remove borrower record
    } else {
        if (!copyToBookMap.containsKey(copyId)) {
            System.out.println("Book copy does not exist.");
        } else if (!borrowStatus.get(copyId)) {
            System.out.println("The book is not borrowed.");
        } else if (!(copyBorrowers.get(copyId) == userId)) {
            System.out.println("The book is borrowed by another user.");
        }
    }
}
```

The return book method on the on hand isn't covered by the tests at all while the

Delete Method



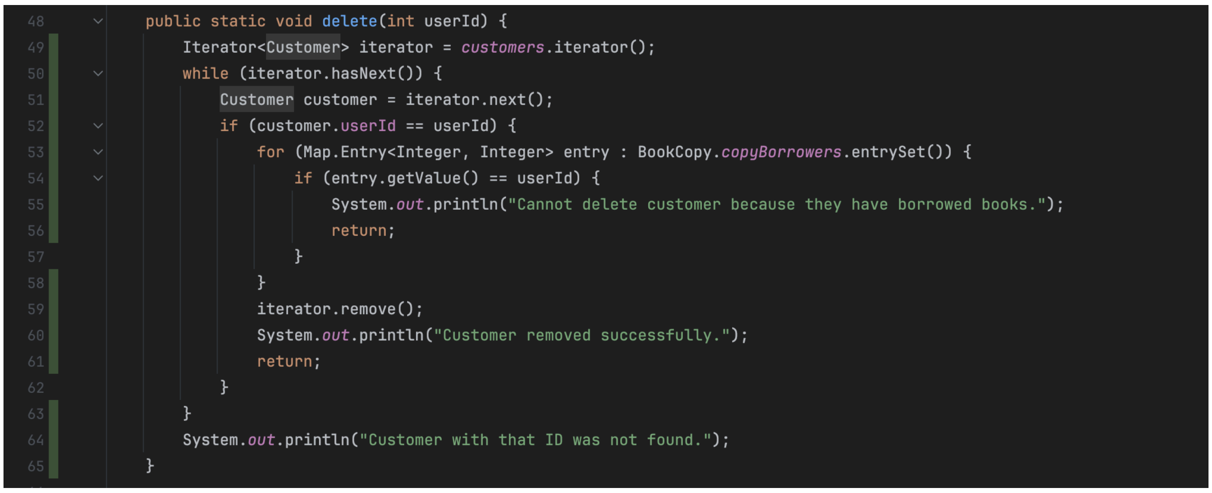
```java
public static boolean delete(int copyId) {
    if (copyToBookMap.containsKey(copyId) && !borrowStatus.get(copyId)) {
        copyToBookMap.remove(copyId);
        borrowStatus.remove(copyId);
        System.out.println("Book copy (id = " + copyId + ") was deleted successfully");
        return true;
    } else {
        if (!copyToBookMap.containsKey(copyId)) {
            System.out.println("No book copies with that ID");
        } else {
            System.out.println("Book copy is currently borrowed.");
        }
    }
    return false;
}
```

is covered in its entirety.

Customer

The test cases cover 47% of the methods, 75% of the code lines and 100% of the branches of this class.

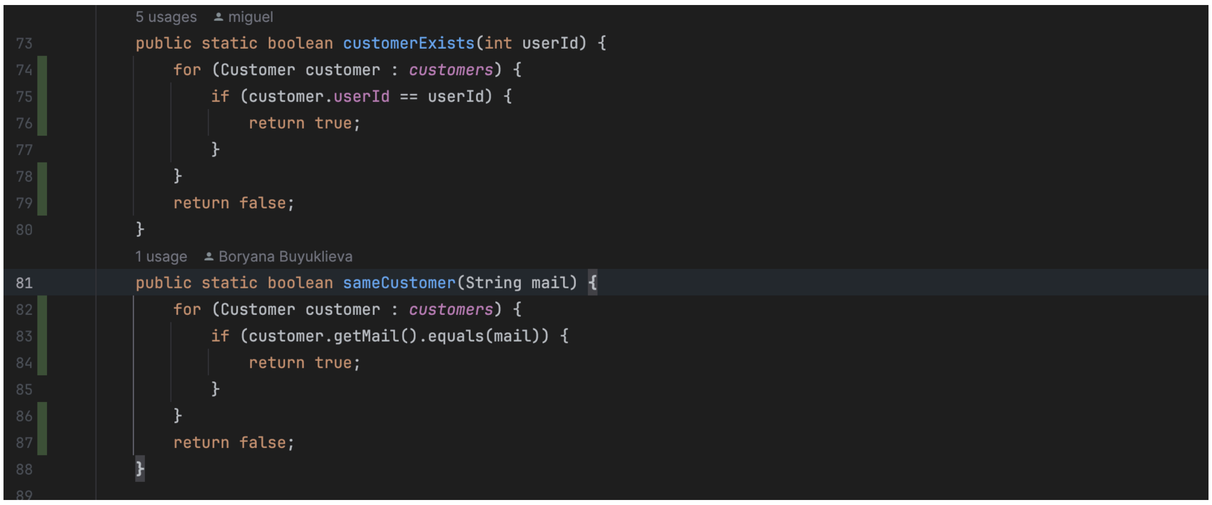
Delete Method



```java
public static void delete(int userId) {
    Iterator<Customer> iterator = customers.iterator();
    while (iterator.hasNext()) {
        Customer customer = iterator.next();
        if (customer.userId == userId) {
            for (Map.Entry<Integer, Integer> entry : BookCopy.copyBorrowers.entrySet()) {
                if (entry.getValue() == userId) {
                    System.out.println("Cannot delete customer because they have borrowed books.");
                    return;
                }
            }
            iterator.remove();
            System.out.println("Customer removed successfully.");
            return;
        }
    }
    System.out.println("Customer with that ID was not found.");
}
```

Here in the delete method all three different conditions when deleting a customer are covered by the three different tests in the test class

CustomerExist and SameCustomer Method



```java
5 usages  miguel
public static boolean customerExists(int userId) {
    for (Customer customer : customers) {
        if (customer.userId == userId) {
            return true;
        }
    }
    return false;
}
1 usage  Boryana Buyuklieva
public static boolean sameCustomer(String mail) {
    for (Customer customer : customers) {
        if (customer.getMail().equals(mail)) {
            return true;
        }
    }
    return false;
}
```

Also those two methods are covered completely by the test cases.

Reference:

- https://www.baeldung.com/cs/code-coverage