Testing and Debugging

# Section 1. Issues found in Customer.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| Constant declarations | TRANSACTION\_FEE should be private since it’s not used outside the class. | Changing public keyword to private. |
| Constant declarations | COLLECTION\_FEE is not used anywhere | Remove COLLECTION\_FEE |
| Constructor | “this.adress” is set to value equal to “name” passed in constructor parameters. | Set “this.adress” equal “adress”. |
| getAddress() | getAdress returns name | Return address instead of name. |
| getTicketPrice() | First two lines are unnecessary, increases the length of code without any significant benefit to performance. | Since bookings size is checked in for loop, first two lines can be deleted, no need to check if the size is equal twice. |
| getTicketPrice() | For loop starts from second element in bookings ArrayList. | Change change int i = 1 to int i = 0 |
| getTicketPrice() | Price variable is integer, possibility for data loss | Chang int price to double price |
| getDeductionPrice() | First two lines are unnecessary, increases the length of code without any significant benefit to performance. | Same situations as in getTicketPrice(), first two line can be removed. |
| getDeductionPrice() | If (counter == 3) is checked even if the counter has not increased, there is no need to check the counter if no adult tickets were bought. | Put if(counter==3) and everything in it’s scope into previous if statement’s scope after counter++. |
| getPostagePrice() | Method returns a hard coded number instead of constant. | Change 2.5 to POSTAGE\_FEE |
| getPostagePrice() | Method is public but is only used inside of the class | Change to private |
| getGrandTotal() | Deduction price is added to total price instead of being substracted. | Change +getDeductionPrice() to -getDeductionPrice() |
| getGrandTotal() | Postage fee is added to total price without checking the method of delivery | Change POSTAGE\_FEE to getPostagePrice() |

# Section 2. Issues found in Show.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| Top of Show class definition | dateAndTime is set to public | Set dateAndTime to private |
| toString() | Adds 18+ to the name if show is child friendly, should do the oposite. | Change if (isChildFriendly()) to if (!isChildFriendly()) |

# Section 3. Issues found in ShowBooking.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| Constant declarations | ADULT\_PRICE is set to 10 should be 12.5 | Change constant to 12.5 |
| Class variable\* | Description at the end of the document | Description at the end of the document |
| addTicket(Ticket type) | Inappropriate comment “no smelly children allowed” | Remove the comment since following code is self explanatory |
| addTicket(Ticket type) | Checks if show is not child friendly OR the type of of ticket is CHILD | Change || to && because both statements should be true, not either. |
| getPrice() | Loop continues if i is less than getTickets().size()-1, this does not ensure that all elements ar checked. | Remove the -1 |
| getPrice() | Wrong price is added to total price. | Swap price+=ADULT\_PRICE with price += CONCESSORY\_PRICE |
| getNumOfConcTicket() | Function does not account for STUDENT type tickets | Add + getNumOfTicket(Ticket.STUDENT) to treturn |

# Section 4. Brief report on testing and debugging strategies

Since the project was not of wide scope, I have mostly used brute force approach on debugging the code. I went through each line having in mind what which class and method should do and fixed most obvious errors in code, such as getter method returning wrong variable, constant set to wrong value and other similar errors which would most likely be created due to distraction of a programmer. After that I went through the code again and fixed errors regarding accessibility of class variables, in most cases a variable was set to public where there was no need for outside classes to access it. Finally, I looked through the code again and try to detect any unnecessary code, also once again searched for logic errors. An error which was the most difficult to spot was usage of a static variable in a class where multiple instances of the same class could have been created. After the debugging stage was over, some tests were created while trying to mimic user behaviour in real-life and also attempting to break the system. Attempts to break the system included:

* Trying to book a ticket when there are no more seats available.
* Trying to book a child’s ticket when a show is for adults only.
* Checking if 342 discount system works and if it still works after reducing number of adult tickets booked.

The reason why JUnit testing was not used is because all the methods of every class were fairly small in size and the code was readable, which made it easy to predict the outcome of separate methods.

The reason why Eclipse debugger was not used is because code structure was good, which made it easy to follow separate values, hence tracking individual values using debugger was unnecessary.

Possibility for overflows were not taken into account, due to the fact that all the variables would not exceed the limitations in real life.

\*Class variable NUM\_TICKETS\_AVAILABLE problem description and potential fixes:

I have found a flaw in the logic of the booking system and I am not sure whether it was designed in such way on purpose. NUM\_TICKETS\_AVAILABLE variable in ShowBooking class is defined as private and static, which means that it’s value persists even if several instances of the class is created, this introduces a bug which does not allow to book tickets even if they should be available. For example, if there is more than 1 show, the total number of tickets available to book in the system is still 252, instead of being 252 for each show and being tracked separately. If this is the planned design then no changes should be made to the field. Otherwise, possible fixes:

If it was intended to limit total number of tickets for a booking, the variable (NUM\_TICKETS\_AVAILABLE) should be private int instead of private static int, this way each ShowBooking instance can track its own number. This, however introduces another problem, where any show can be overbooked simply by creating more than one ShowBooking.

If it was intended to limit total number of tickets available per show, which complies with real-world rules, where in one room only one show takes place at the time and the room can fit limited number of people, there is another fix. Instead of declaring NUM\_TICKETS\_AVAILABLE and MAX\_NUM\_TICKETS in ShowBooking class it should be put in Show class, this way each show has the same maximum number of tickets available (since it is static and final), but each instance of the Show class has it’s own NUM\_TICKETS\_AVAILABLE. The variable for tickets available should have a getter and a setter, it must also be set to maximum number of tickets in the constructor of the show. When addTicket method in ShowBooking class is called, it should be checked whether a show (private field of ShowBooking instance) has available tickets by writing something similar to if(show.getNumTicketsAvailable()>0) { show.setNumTicketsAvailable(show.getNumTicketsAvailable()-1) } and then continue the logic of addTicket method by having in mind that NUM\_TICKETS\_AVAILABLE is in Show class. This fix would allow multiple users to make multiple bookings for more than one show until the show does not have any more tickets available.