Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es

TEST PLAN AND TEST RESULTS

Pablo Cuesta Sierra, Pablo Fernández Alegrea, Álvaro Zamanillo Sáez

Application: ARCH Theater Hall

Date: May 14, 2021

Pablo Fernández Alegre pablo.fernandezalegre@estudiante.uam.es
Álvaro Zamanillo Sáez alvaro.zamanillo@estudiante.uam.es
Pablo Cuesta Sierra pablo.cuestas@estudiante.uam.es

Index

Pablo Fernández Alegre pablo.fernandezalegre@estudiante.uam.es
Álvaro Zamanillo Sáez alvaro.zamanillo@estudiante.uam.es
Pablo Cuesta Sierra pablo.cuestas@estudiante.uam.es

1. TEST CASE: PURCHASE/RESERVATION

1.1 Use Case 1: Purchase/Reservation

Primary Actor: Registered user
Stakeholders and Goals:
Registered user: to make a reservation or a purchase for a scheduled performance.
Manager: has previously created an event and scheduled a performance to offer tickets for it.
Preconditions:
1. At least one performance of an event has already been scheduled by the manager
2. The user who wants to purchase/reserve is registered in the application.
3. If it's a reserve operation, the limit time for doing reservations hasn't been surpassed.
4. In order to pay with a pass, the user must have purchased it before, and the areas/cycle parameters must match between the pass and the ticket.

Success guarantee (Post-conditions):
1. If it is a purchase operation the user will get:
   a. The amount of tickets he has purchased with the name of the show, time of the performance, price, area, and seat if it is a sitting area on them
   b. A file with format .pdf with a unique identifier code.
2. If it is a reserve operation, the reservation will be saved on the system until it is confirmed or cancelled before the expiration time, after which it will be automatically cancelled.
3. The amount of tickets purchased or reserved won't be longer available for the rest of users.

Main Success Scenario:
1. User logs in the system.
2. Searches for the event.
3. Selects a performance of that event.
4. Selects an area.
5. Chooses automatic or manual purchase
   a. Automatic: the customer selects the number of tickets and one of the following options:
      i. centered in rows and columns
      ii. centered in lower rows
      iii. centered in upper rows
      iv. farthest from all sold seats
   b. Manual: Customer specifies the number of tickets and selects an area, and seats if it's a sitting area.
6. The user selects purchase or reservation.
   a. Purchase:
      i. Paying:
         1. With a credit card (extern service).
         2. With an annual pass.

3. Applying the price reduction of a cycle pass and then paying with a credit card.

ii. The ticket(s) and the pdf assigned to the purchase are created.

b. Reservation: The reservation is saved in the system.

Extensions (Alternative paths):

6c. If there are not enough tickets, the user can sign up for the waiting list in order to get a notification when he logs in whenever there are available tickets.

Technology and Data Variations List:

The ticket will be generated as a pdf document, including the authentication code of the purchase.

Frequency:

This is one of the most frequent actions in the system (only less frequent than searching); it will be performed, at least, tens of times for each scheduled performance.

Open Issues:

We have yet to specify how the authentication code is generated and whether the user gets any sort of ticket or pdf when they make a reservation, or if the reservation is just in the system.

1.2. Test case design

Preconditions:

1. The theater has an event “La casa de Bernarda Alba”
2. This event has at least one future performance with available tickets for the 27-07-21.
3. A registered user has searched for the event and the performance.
4. The maximum number of tickets for performance and client has been established at 3.

Scenario (for purchase of tickets):

1. Register user clicks on button “Get tickets” for the first performance of “La casa de Bernarda Alba”.
2. System shows the PurchaseTicketPanel. This panel shows a tree with the different areas of the theater, a selection panel, a list with the selected tickets and a payment panel.
3. User navigates through the tree.
4. Every time he clicks on an area of the tree, the selection panel will change whether it is a sitting area or standing.
   a. When clicking on a sitting area (B or A1), the system will display a selection panel with two tabs: manual or automatic mode.
   b. When clicking on the standing area (A2) , the system shows a simple panel to introduce the number of tickets.
5. For each area, the user can select tickets to be added to his purchase list:

   a. In SittingAreaB, the user selects 3 tickets in the center by clicking the option “Centered”, typing ‘3’ in the text field and pressing the button “Add tickets”.

   b. In SittingAreaA, the user selects 1 ticket for row 0 and column 0 by clicking in the checkBox with that index and then confirms the action with the button “Add tickets”.

c. For the StandingArea the user adds another ticket by typing '1' in the text field and pressing once again the button "Add tickets".

6. After each selection operation, the system adds the tickets to the "shopping cart".
   a. System prevents the user from adding repeated tickets, if seat (0,0) is selected again and the button "Add tickets" is pressed, nothing occurs.
7. The user has 3 options (for the test, we follow b):
   a. clear all the list and restart from step 3.
   b. confirm the purchase by paying or reserving.
   c. exit by clicking in any of the navigation buttons in the top panel. The user shall not press such button in the test so as to check the purchasing features.
8. The user can decide whether to reserve the tickets or buy them with a card or a valid pass. For this test, card pay method will be chosen
   a. First, the user introduces a invalid number "111". The system notifies the purchase operation has failed.
   b. In the second attempt, the user introduces a valid number "1234567890987654". System must limit the number of tickets purchased to 3 (the theater has been configured with this parameter).
   c. The user is notified via panel that only 3 tickets of the selected have been correctly purchased.
9. The user ends the purchase by clicking on the "Home" button and the system redirects the user to his home panel where the updated number of tickets must be shown.
10. The user can check that the 3 purchased tickets have been added to his list by clicking on the "My tickets" button.

Difference with the use case diagram:

In the use case, it was possible to pay simultaneously a list of tickets with a pass and card. In our design, passes can only be used when a single ticket is selected for simplicity sake. By doing this, we do not reduce the functionality, although it is true that two different purchase actions are needed if you want to buy more than one ticket and use a pass.

Another difference that makes our design more flexible than the one described in the use case, is that you can buy/reserve tickets for different areas in the same purchase operation. This is mainly achieved thanks to the tree with the areas that allows the navigation through the whole theater.

1.3. Test execution result

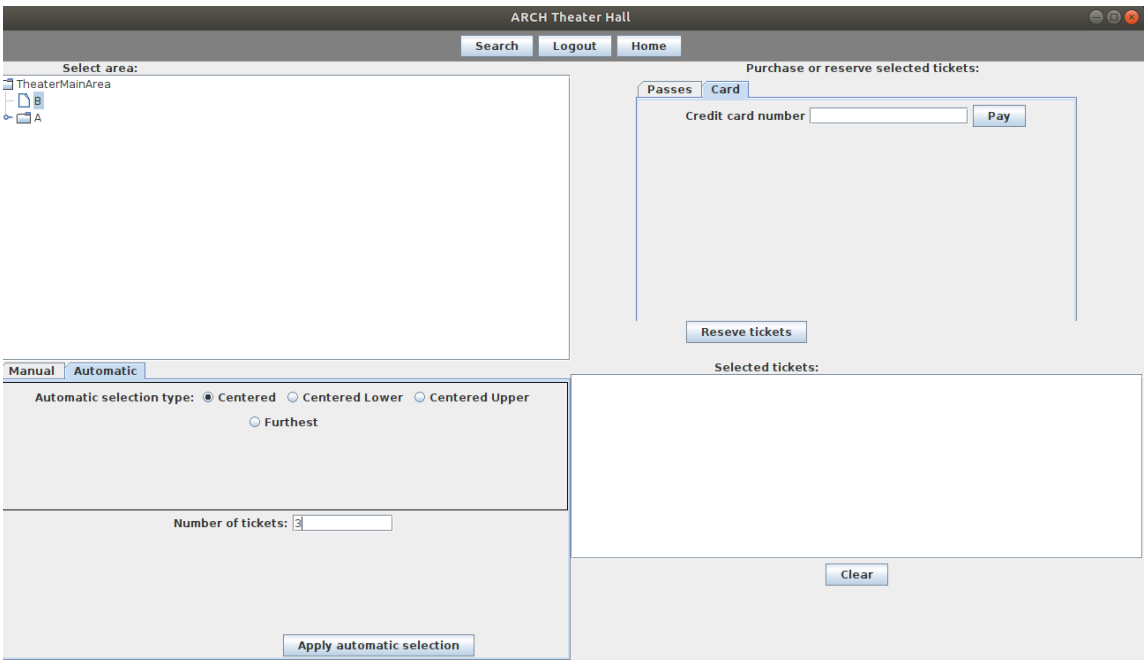
1. Screenshots of the different windows.

The number in brackets associates the screenshot to the step of the test.

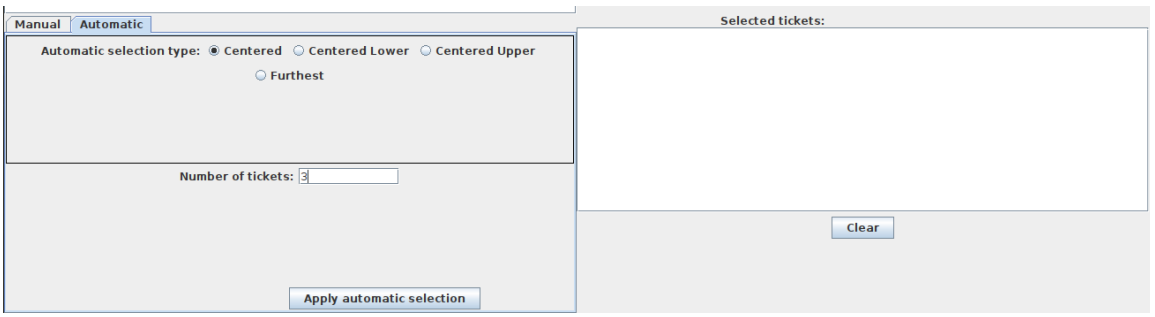
Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es

Previous panel to the Purchase/Reserve tickets panel (1)



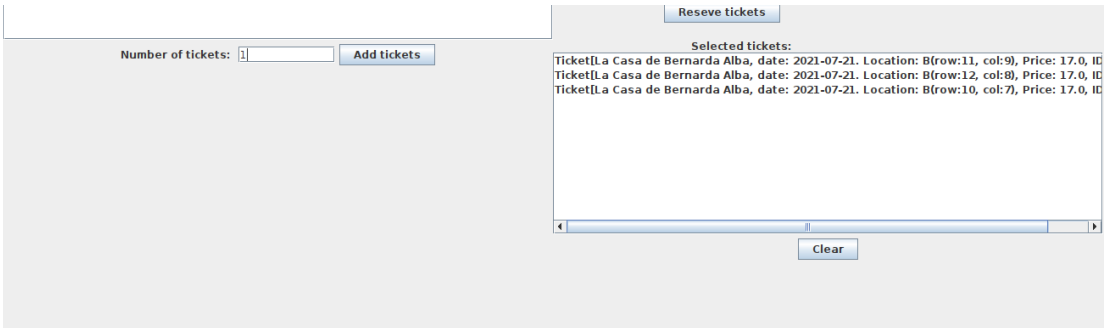
Purchase tickets panel main view. In upper left corner there is the area tree panel for the navigation (3)



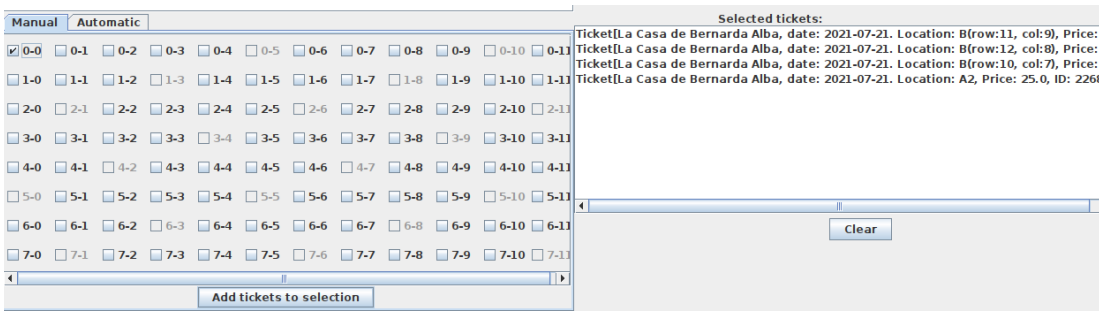
Automatic panel selection, 3 tickets centered are going to be selected (5a).

Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

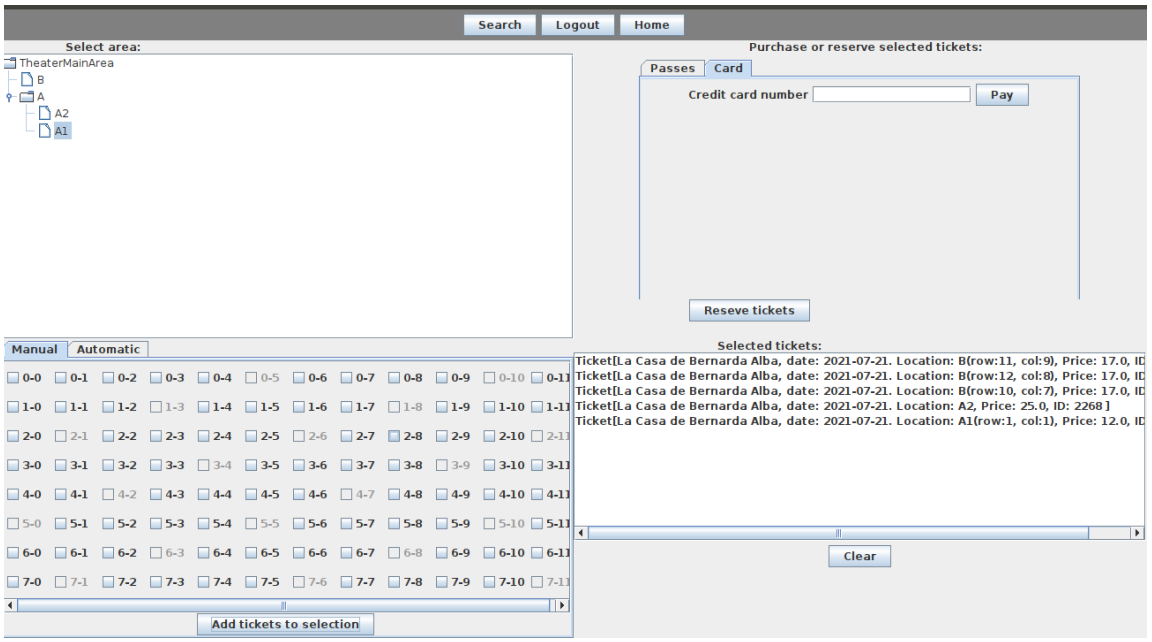
pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es



Standing area selection. 1 ticket is going to be selected. The tickets selected in 5a have been added to the shopping cart (right part of the screenshot). (5b)



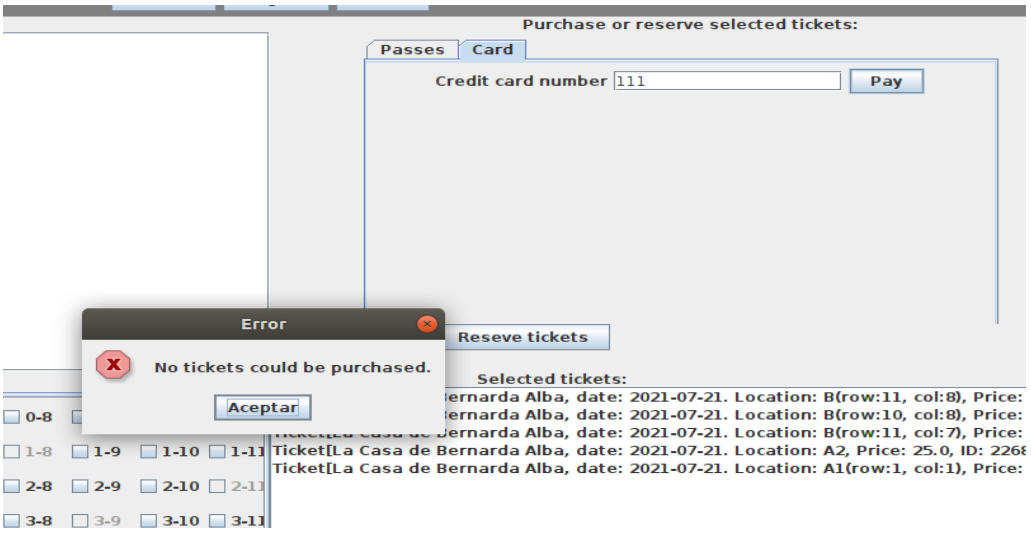
Manual selection panel. Seat (0,0) is going to be selected. (5c)



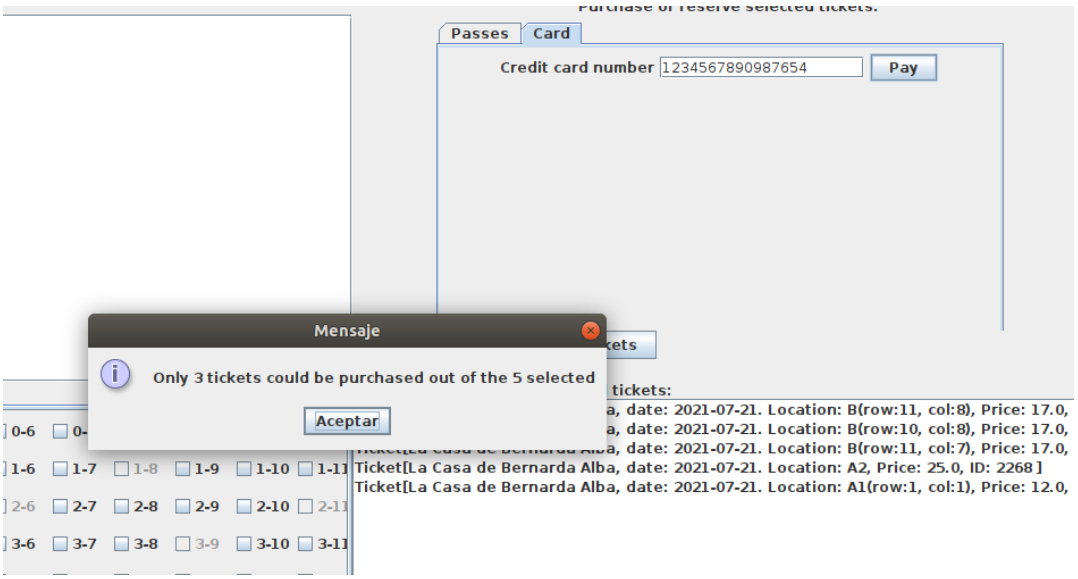
Five tickets have been selected. Step prior to the purchase confirmation

Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

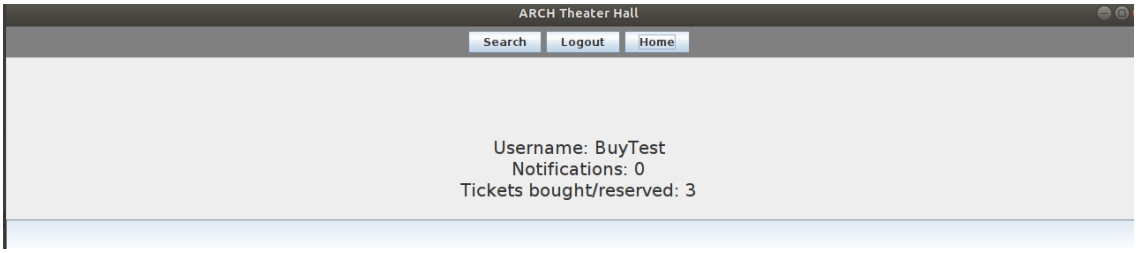
pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es



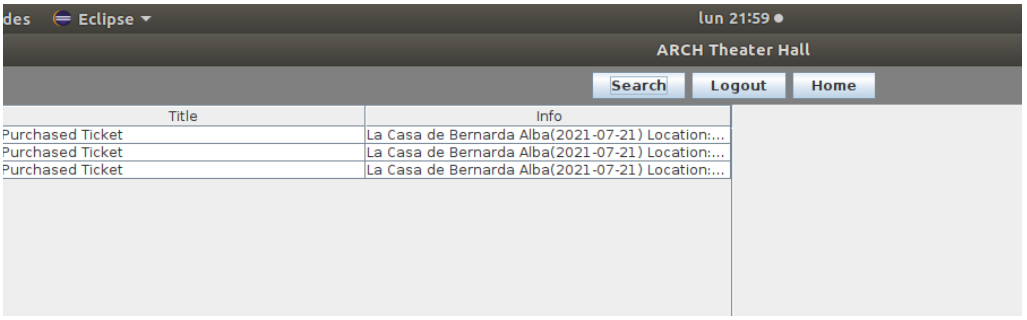
The purchase operation fails because of an invalid card number (8a).



Purchase of the maximum tickets allowed by the theater parameter. (8b)



Home panel of the user with the updated number of tickets (9)

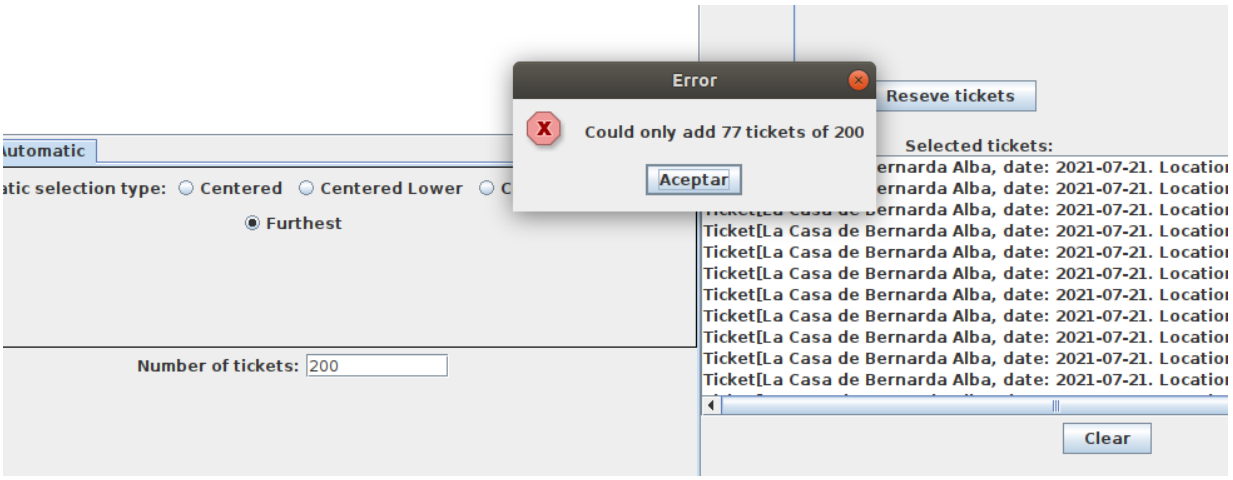


List of the user's ticket updated (10)

2. Comments comparing the result of the test with the description in the test case. In case they do not agree, indicate the reasons for it.

The test described before gives the expected output. However during this part of testing, we ran more tests regarding the purchase/ reserve functionality, as well as the waiting list. In these tests, we tried different combinations of actions; many of them would be extremely bizarre for a common user. During these extensive tests we found an error:

When using the automatic selection, if more tickets than the available are required, the expected behaviour is that the available tickets are added to the shopping cart and a warning panel is displayed.



Expected output when requiring more tickets than the available with automatic selection

Nevertheless, if the number of tickets typed is higher than the available number of tickets for all the areas, an exception occurs (null pointer). This is due to a mistake in the code part. The method for the automatic selection (that wraps the different automatic methods) has a check condition regarding the number of available tickets taking into account all areas—it does not make sense; it should take into account only the area passed as parameter if anything. Anyway, this *if* block should be deleted because our philosophy for the automatic selection (followed in the other methods) is returning the maximum between the requested tickets and the available.

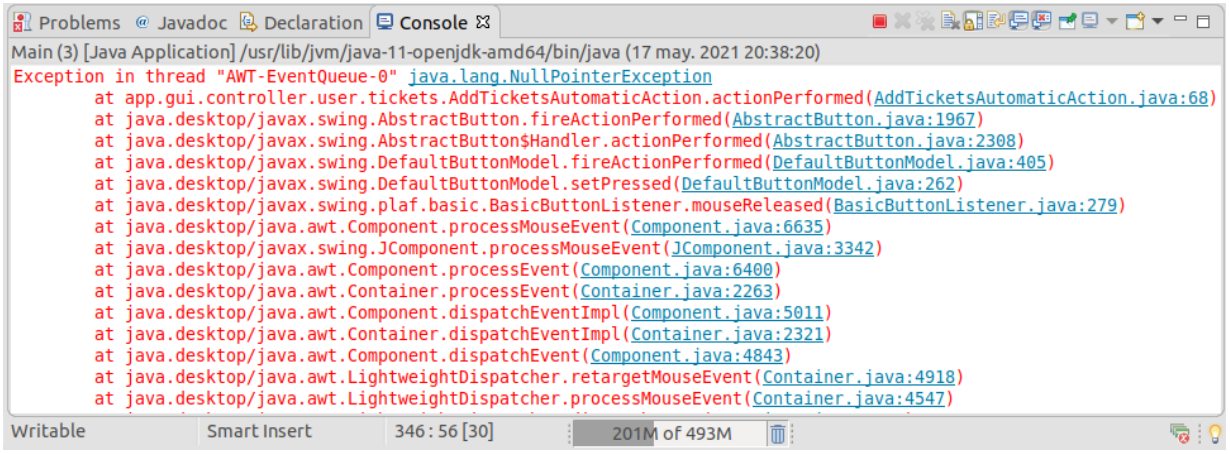
```
8⊖    public List<Ticket> selectAutomatic(SittingArea area, int numTickets, AutomaticSelectio
9
0         if(getAvailableTickets().size() < numTickets){
1             return null;
2         }
```

Code in class Performance

If this condition is not met, the method returns null which causes the exception in the action listener associated with the "Add tickets" button in the instruction list.size() (tickets variable is null).

```
List<Ticket> tickets = performance.selectAutomatic(area, nTickets, type);
int size = tickets.size();
```

Code in the action listener



We did not find this problem until this part of exhaustive testing. There are many errors regarding the mentioned *if* block. At this point of the project, they seem pretty obvious; they were not so obvious at the beginning though. The two most remarkable are:

a. Returning null instead of an empty list is such a bad practice and error prone as it has been shown.
b. The check condition contradicts the logic of the other methods about automatic selection.

Without any doubt, the best solution (and also the simplest) would be to delete that *if* block

Pablo Fernández Alegre pablo.fernandezalegre@estudiante.uam.es
Álvaro Zamanillo Sáez alvaro.zamanillo@estudiante.uam.es
Pablo Cuesta Sierra pablo.cuestas@estudiante.uam.es

2. TEST CASE: AREA CONFIGURATION

2.1 Use Case 2: Area Configuration

Primary Actor : Manager

Stakeholders and Goals:

Manager: To establish a new configuration of the theater hall based on a list of areas that can be composite.

Preconditions:

1. The manager has logged in the system.
2. There are no scheduled performances in the theater hall.

Success guarantee (Post-conditions):

1. If a new area has been created, it has been configured with all its data specified and is ready to be selected when creating an event.

Main Success Scenario:

1. The manager selects the option to configure the areas of the hall.
2. The manager selects the option to create a new area.
   a1. The area can be created outside every other area.
   a2. The area can be created inside another area.
   b. Select between composite area or simple area.
      i. If it is a simple area, configure: sitting/standing and seat configuration/capacity.
3. Go back to step 2 or end.

Alternative Path:

In step 2, another option that can be selected is the deletion of an existing area.

Special Requirements:

At the end, there can not be any composite area without any areas inside.

Frequency:

Low frequency based on the restriction of no events planned. The area configuration can only be done during periods of low activity when there are no performances planned yet.

Open Issues:

Does it make sense to have a composite area with only one other area inside of it?

Pablo Fernández Alegre pablo.fernandezalegre@estudiante.uam.es
Álvaro Zamanillo Sáez alvaro.zamanillo@estudiante.uam.es
Pablo Cuesta Sierra pablo.cuestas@estudiante.uam.es

2.2. Test case design:

Preconditions:

1. No performances are scheduled.

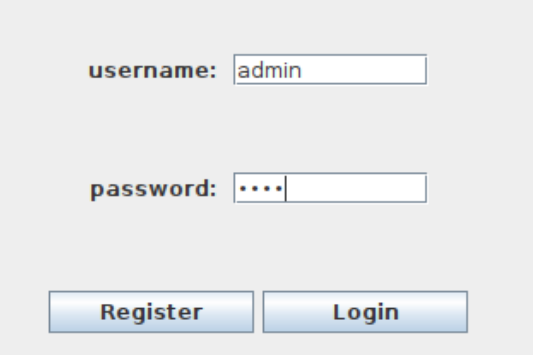
Scenario:

1. The system shows the login window
2. Manager logs in using his username (“admin”) and password (“pass”) and clicks on the button “login”.
3. The system shows the admin control window.
4. Manager clicks on the button “Modify areas”.
5. System shows the area configuration window.
6. Manager selects the area “Theater Main Area” (area where all areas are contained) and clicks on the button “Add area”.
7. System shows the area creation panel.
8. Manager writes the new area’s name (“Box”) and selects the composite area tab. He will click on “Create”.
9. System shows the updated area configuration panel with the new composite area.
10. Manager selects the area “Box” (area where all areas are contained) and clicks on the button “Add area”.
11. System shows the area creation panel.
12. Manager writes the new area’s name (“Left Box”) and selects the sitting area tab. He will introduce the number of rows (4) and columns (5) of that area and click on “Create”.
13. System shows the updated area configuration panel with the new area.
14. Manager selects the area “Box” (area where all areas are contained) and clicks on the button “Add area”.
15. System shows the area creation panel.
16. Manager writes the new area’s name (“Right Box”) and selects the sitting area tab. He will introduce the number of rows (5) and columns (4) of that area and click on “Create”.
17. System shows the updated area configuration panel with the new area.
18. Manager selects the area “Theater Main Area” (area where all areas are contained) and clicks on the button “Add area”.
19. System shows the area creation panel.
20. Manager writes the new area’s name (“Amphitheater”), selects the composite area tab and clicks on “Create”.
21. System shows the updated area configuration panel with the new area.
22. Manager selects the area “Amphitheater” and clicks on the button “Add area”.
23. System shows the area creation panel.
24. Manager writes the new area’s name (“Standing Amphi”) and selects the standing area tab, introduces the capacity of the area (75) and clicks on “Create”.
25. System shows the updated area configuration panel with the new area.

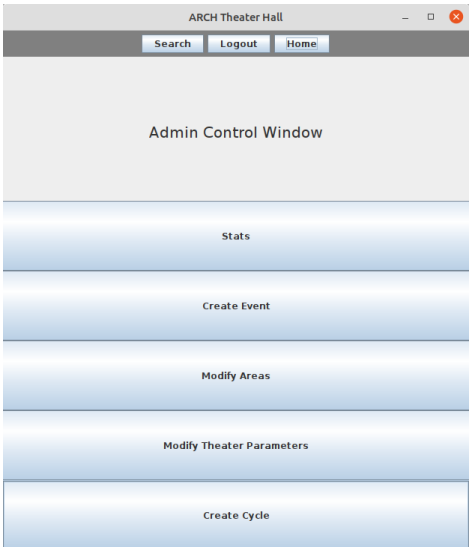
26. Manager selects the area “Amphitheater” and clicks on the button “Add area”.
27. System shows the area creation panel.
28. Manager writes the new area’s name (“Box”) and selects the sitting area tab. He will introduce the number of rows (6) and columns (10) and click on “Create”.
29. The system will display an error message as there already is an area with the same name.
30. The user changes the name of the area by “Sitting Amphi” and clicks on create.
31. System shows the updated area configuration panel with the new area.
32. Manager selects the area “Box” and clicks on “Remove Area”.
33. System shows the updated area configuration panel without the area removed (and its sub areas removed too).
34. Without selecting any area, the manager clicks on “Remove Area”.
35. System shows a message saying that an area has to be selected in order to be removed.
36. Manager selects the area “TheaterMainArea” and clicks on “Remove Area”.
37. System shows a message saying that this main area cannot be removed.

2.3. Test execution result

1. Screenshots of the windows.

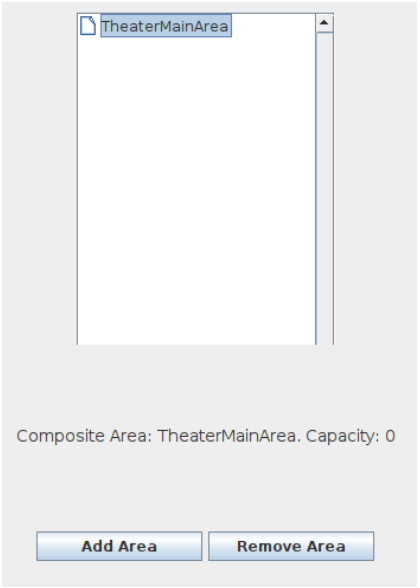


Login panel (step 1)

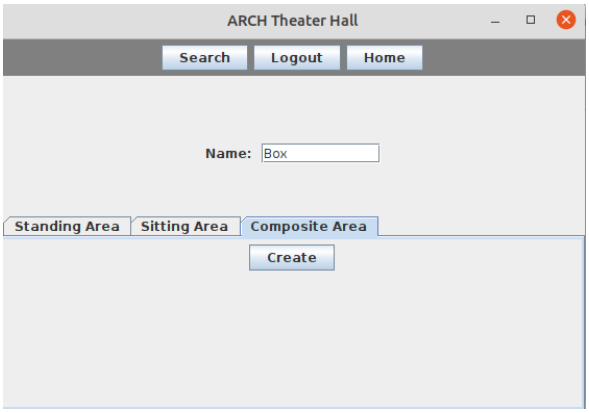


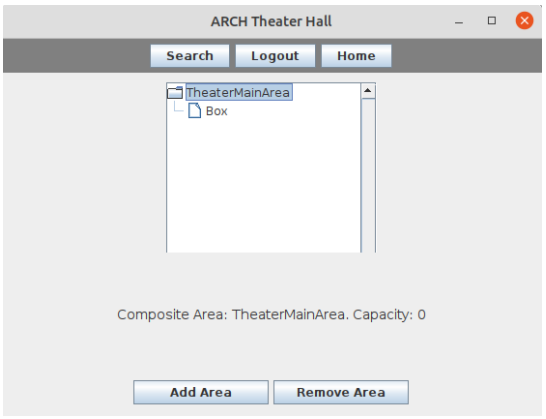
Admin control window (step 3)

pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es



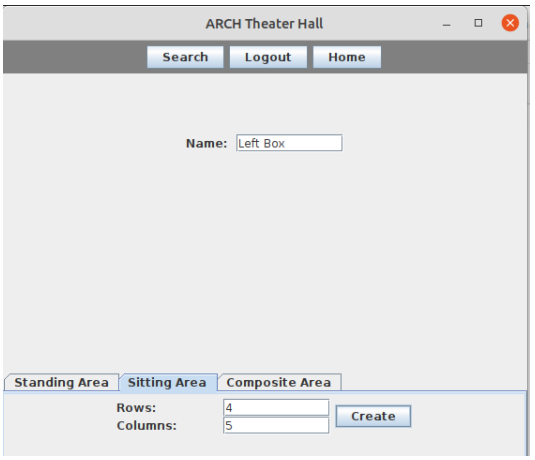
Step 5

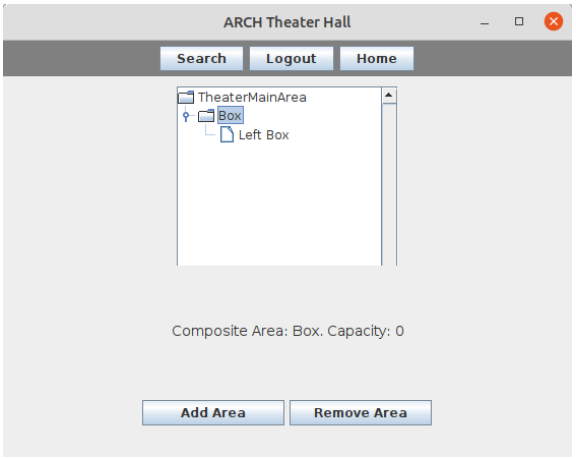




Step 7-8

Step 9-10



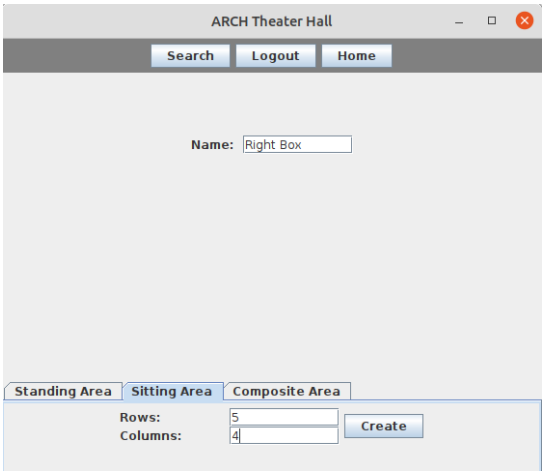


Step 11-12

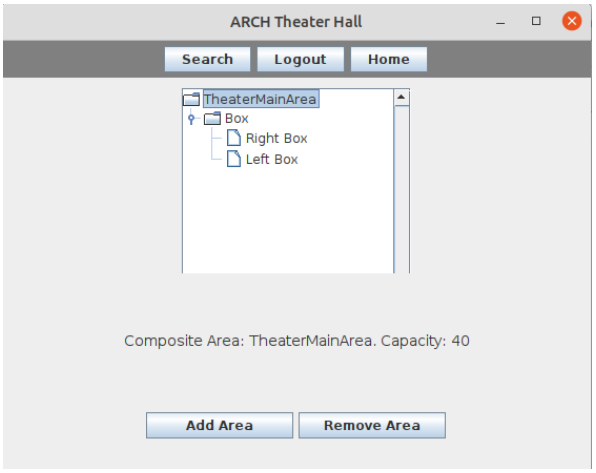
Step 13-14

Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

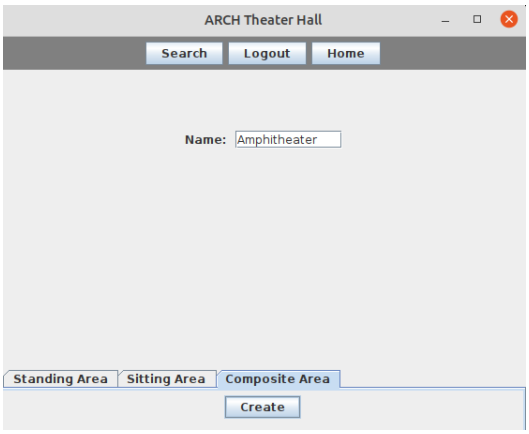
pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es



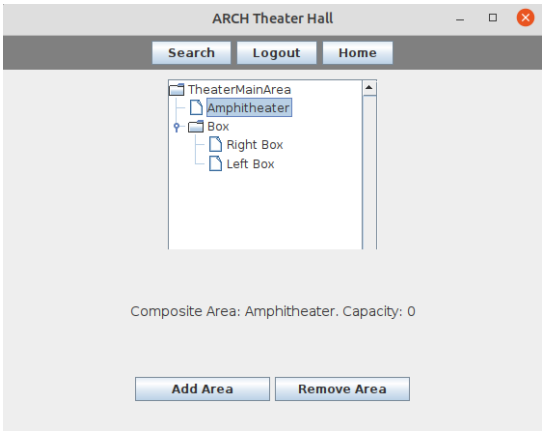
Step 15-16



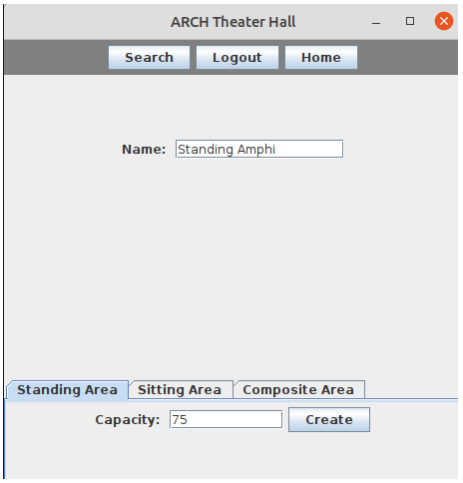
Step 17-18



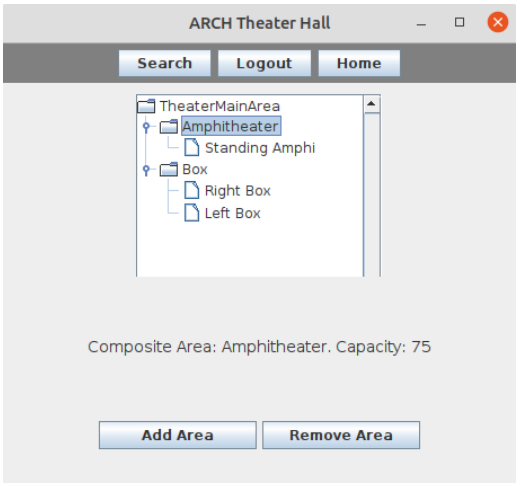
Step 19-20



Step 21-22



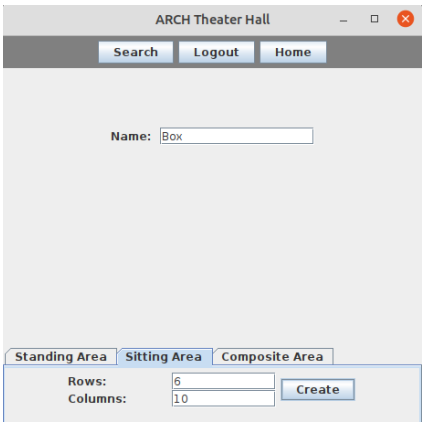
Step 23-24



Step 25-26

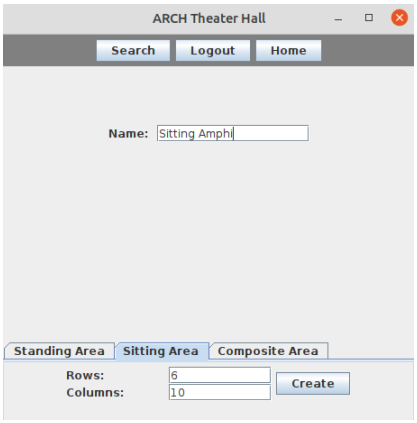
Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

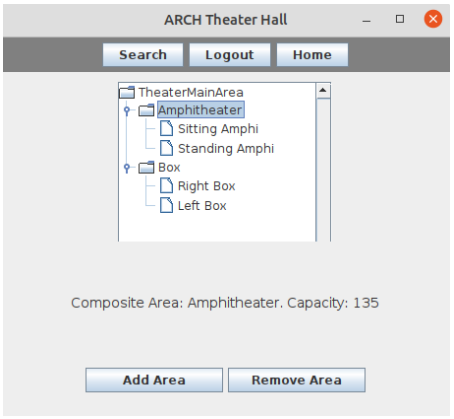
pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es



Step 27-28

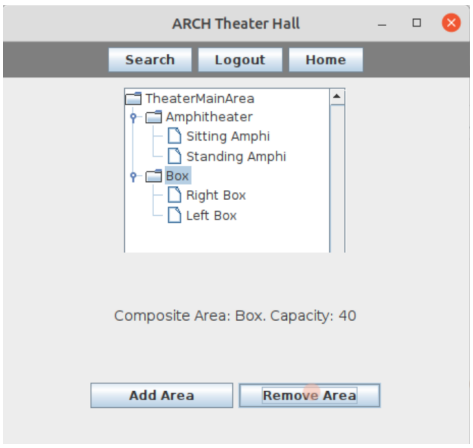
Step 29

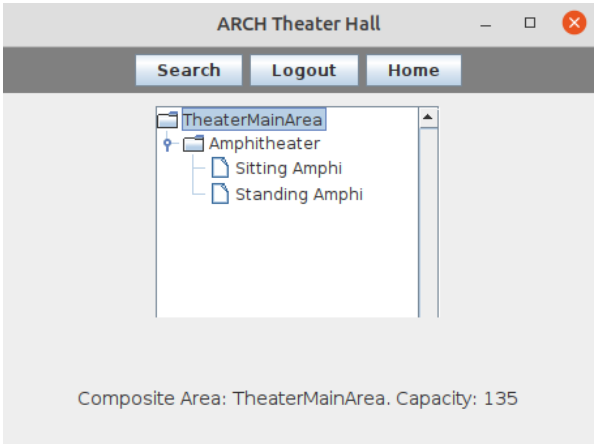




Step 30

Step 31



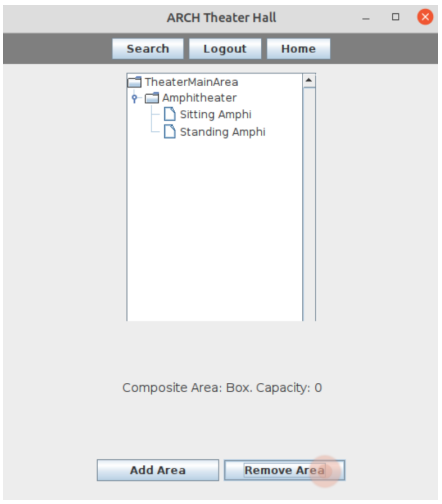


Step 32

Step 33

Pablo Fernández Alegre
Álvaro Zamanillo Sáez
Pablo Cuesta Sierra

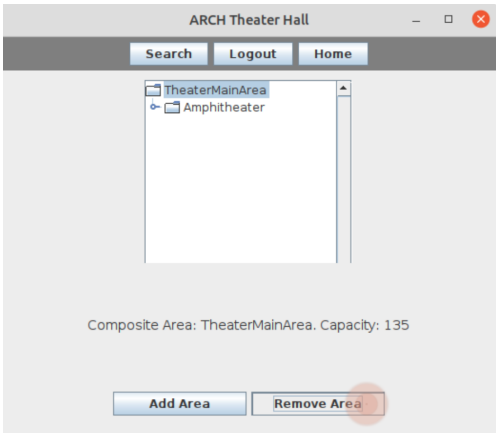
pablo.fernandezalegre@estudiante.uam.es
alvaro.zamanillo@estudiante.uam.es
pablo.cuestas@estudiante.uam.es



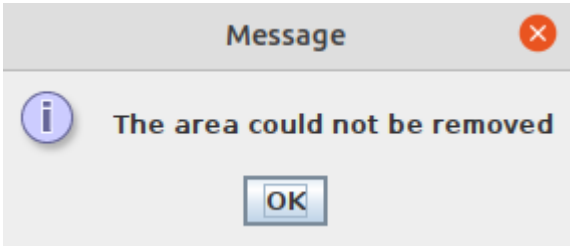
Step 34

```
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException: Cannot invoke "javax.swing.tree.DefaultMutableTreeNode.getUserObject()" because "selectedNode" is null
        at app.gui.controller.areas.RemoveAreaAction.actionPerformed(RemoveAreaAction.java:47)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1967)
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2308)
        at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:405)
        at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
        at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
        at java.desktop/java.awt.Component.processMouseEvent(Component.java:6614)
        at java.desktop/javax.swing.JComponent.processMouseEvent(JComponent.java:3342)
        at java.desktop/java.awt.Component.processEvent(Component.java:6379)
        at java.desktop/java.awt.Container.processEvent(Container.java:2263)
        at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:4990)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2321)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4822)
        at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4919)
        at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Container.java:4548)
        at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4489)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2307)
        at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2769)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4822)
        at java.desktop/java.awt.EventQueue.dispatchEventImpl(EventQueue.java:772)
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:721)
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:715)
        at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:95)
        at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:745)
        at java.desktop/java.awt.EventQueue$5.run(EventQueue.java:743)
        at java.base/java.security.AccessController.doPrivileged(AccessController.java:391)
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:85)
        at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:742)
        at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:109)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
        at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
```

Step 35 (Error)



Step 36



Step 37

Pablo Fernández Alegre pablo.fernandezalegre@estudiante.uam.es
Álvaro Zamanillo Sáez alvaro.zamanillo@estudiante.uam.es
Pablo Cuesta Sierra pablo.cuestas@estudiante.uam.es

2. Comments comparing the result of the test with the description in the test case. In case they do not agree, indicate the reasons for it.

All steps from the scenario correspond with the expected result except for the step number 35:

There is a small set of cases in which an area is not selected in the tree that contains the areas as nodes. One of them is when the application is first executed, and another is when an area has just been deleted. This second case is where we got the error.

When we try to remove an area without selecting an area, a message should appear, because it is a mistake to remove an area without selecting one. Instead, an exception is unhandled by the program, and the stack trace appears in the console window. To the user, nothing would appear abnormal, because nothing is modified in the areas of the theater when trying to delete an area without selecting any area. That could be something normal in the application. However, this exception should be handled within the code.

This error should be solved either by handling the case when an area is not selected:

```
47          String name = (String) selectedNode.getUserObject();
48          Area area = Application.getInstance().getArea(name);
```

Here there is no guarantee that an area has been selected, and a NullPointerException can be thrown.

```
try {
    lastNode=(DefaultMutableTreeNode)tree.getLastSelectedPathComponent();
    parentArea = Application.getInstance().getArea((String)(lastNode.getUserObject()));
}catch(Exception exc) {
    JOptionPane.showMessageDialog(createAreaPanel, "You have to select a parent area",
            "Error", JOptionPane.ERROR_MESSAGE);
    return;
}
```

This is what we have in a similar controller, which is what should have been done here.