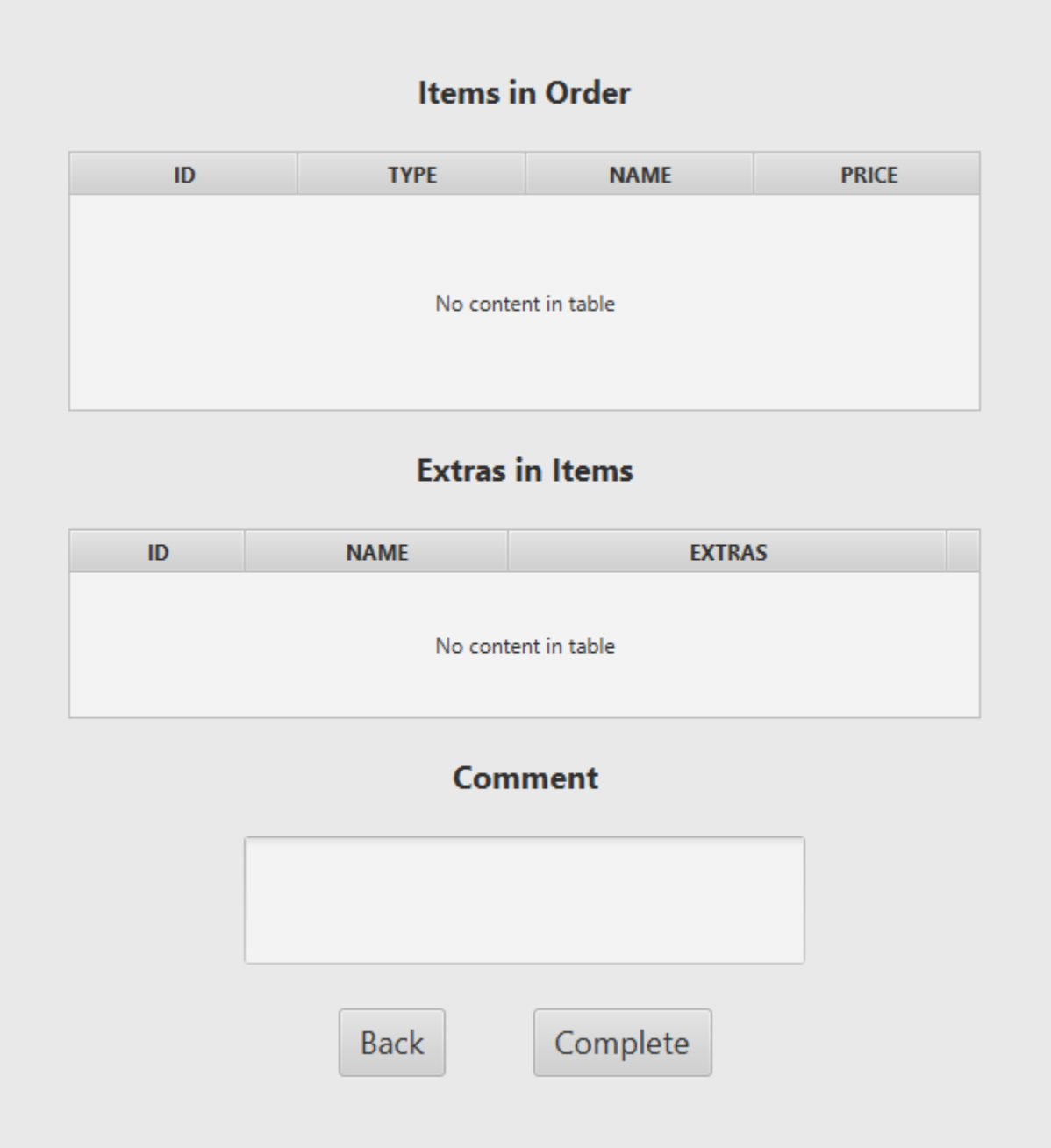
Extras in Barista View - design

The essential concern for the barista is to know if the customer put any extras for his items. To fulfill this case, the OrderDetailsView has to be modified accordingly. New TableView is introduced containing three columns - id and name of the item, and chosen extras. This provides necessary information for the person responsible for preparation of the order.



Extras in Barista View - implementation

To display the correct extras for the specific item, those two properties had to be combined. As a consequence, the ExtraInItemProperty class was introduced.

```
public class ExtraInItemProperty
{
  private ItemProperty item;
  private IntegerProperty id;
  private StringProperty name;
  private ArrayList<ExtraProperty> extras;

  public ExtraInItemProperty(ItemProperty item, ArrayList<ExtraProperty> extras)
  {
    this.item = item;
    this.name = item.nameProperty();
    this.id = item.idProperty();
    this.extras = extras;
  }
```

Inside the OrderDetailViewController the columns are assigned proper values and the reset method is invoked. The reset method calls the method with the same name from ViewModel and after that the items in the table are set using the ObservableList of type ExtraInItemProperty.

OrderDetailViewController - init and reset methods

```
    TableColumn idColTemp = (TableColumn) extrasTable.getColumns().get(0);
    TableColumn nameColTemp = (TableColumn) extrasTable.getColumns().get(1);
    TableColumn extraColTemp = (TableColumn) extrasTable.getColumns().get(2);

    idColTemp.setCellValueFactory(
        new PropertyValueFactory<ExtraInItemProperty, Integer>( s: "id"));
    nameColTemp.setCellValueFactory(
        new PropertyValueFactory<ExtraInItemProperty, String>( s: "name"));
    extraColTemp.setCellValueFactory(
        new PropertyValueFactory<ExtraInItemProperty, String>( s: "ExtrasString"));
    reset();
}

public void reset()
{
    viewModel.reset();
    extrasTable.setItems(viewModel.getExtrasInItems());
    itemsTable.setItems(viewModel.getSelectedOrder().getItemList());
    commentArea.setText(viewModel.getSelectedOrder().commentProperty().get());
}
```

First of all, the list is cleared to avoid duplicates. The next step is to loop through the ItemList and fill the extrasInItems with the items and their extras. The method getExtrasInItems simply returns this variable.

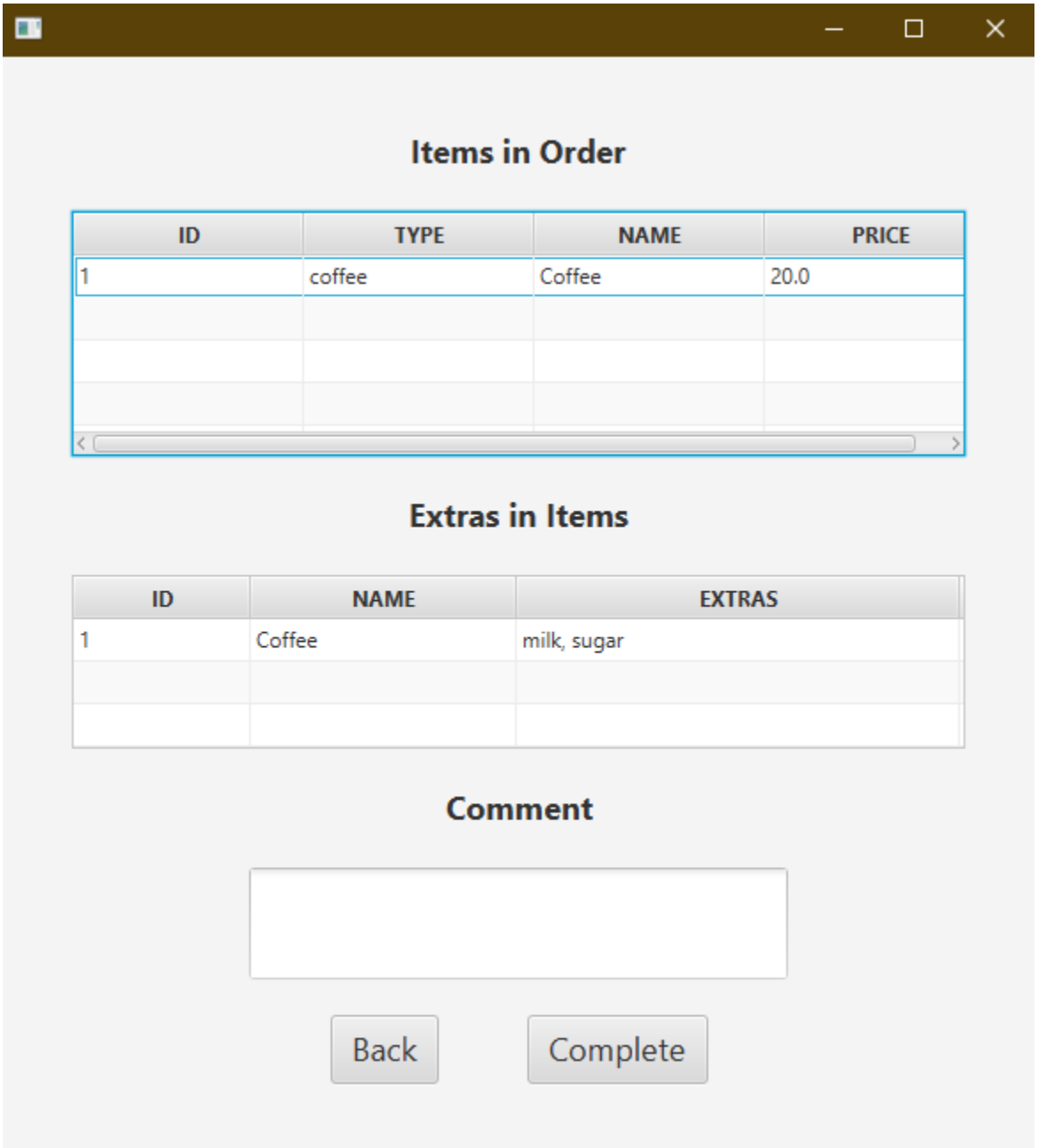
OrderDetailViewModel - reset method

```
public void reset()
{
  extrasInItems.clear();
  OrderProperty order = handler.getSelectedOrder();
  for(ItemProperty item: order.getItemList())
  {
    if(item.getExtras().size()>0)
    {
      extrasInItems.add(new ExtraInItemProperty(item, item.getExtras()));
    }
  }
}

public ObservableList<ExtraInItemProperty> getExtrasInItems()
{
  return extrasInItems;
}
```

To sum up this part, the window displaying the extras for the item is shown.

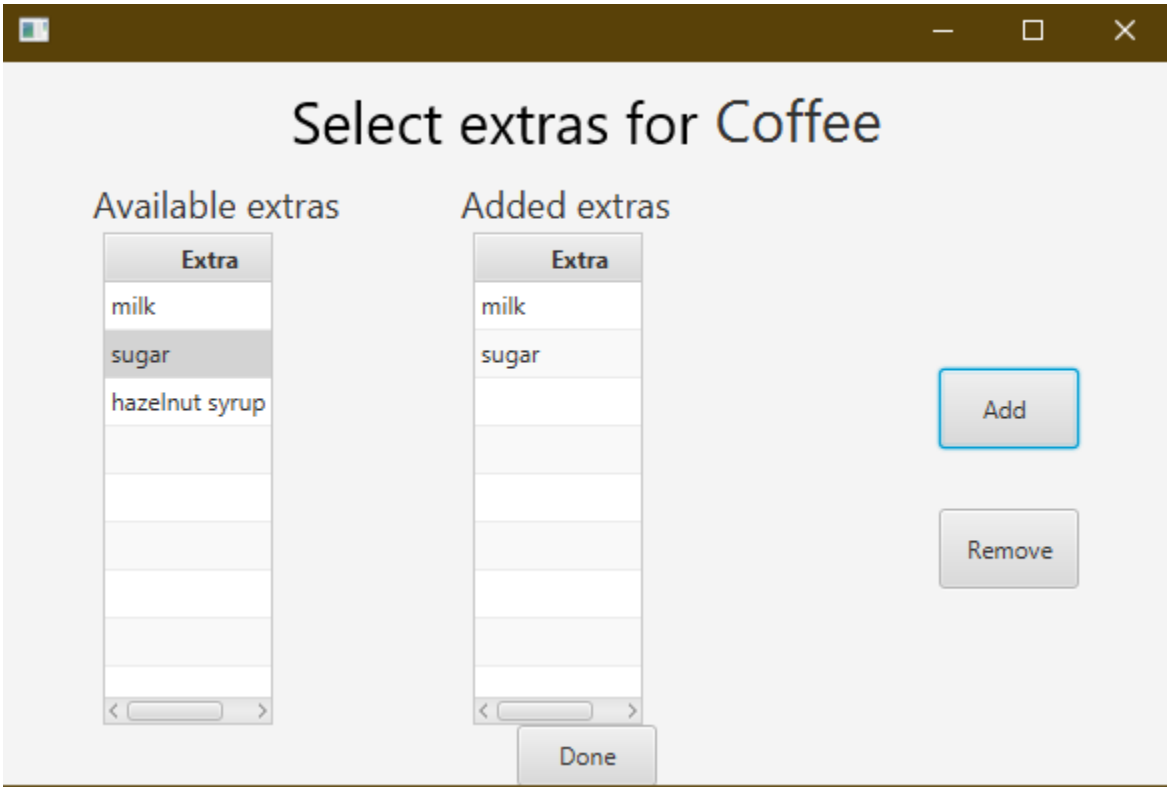
OrderDetailView



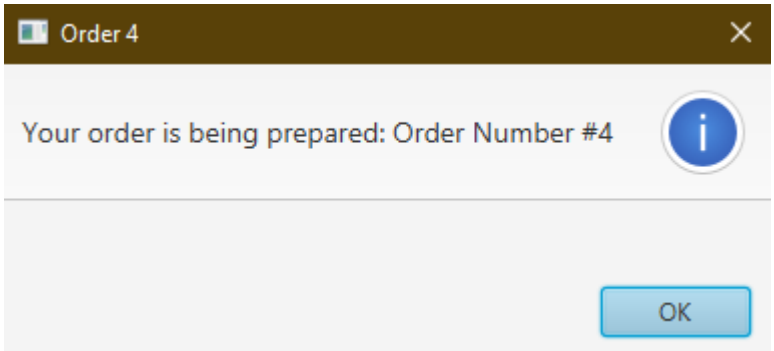
Extras in Barista View - testing

Integration testing was used to test the correctness of display of extras for a specific item in the OrderDetailView. To start the process the customer adds the desired extras for the item and pays for the order. This is crucial to test whether the extras added by the customer are accurately reflected in the OrderDetailView.

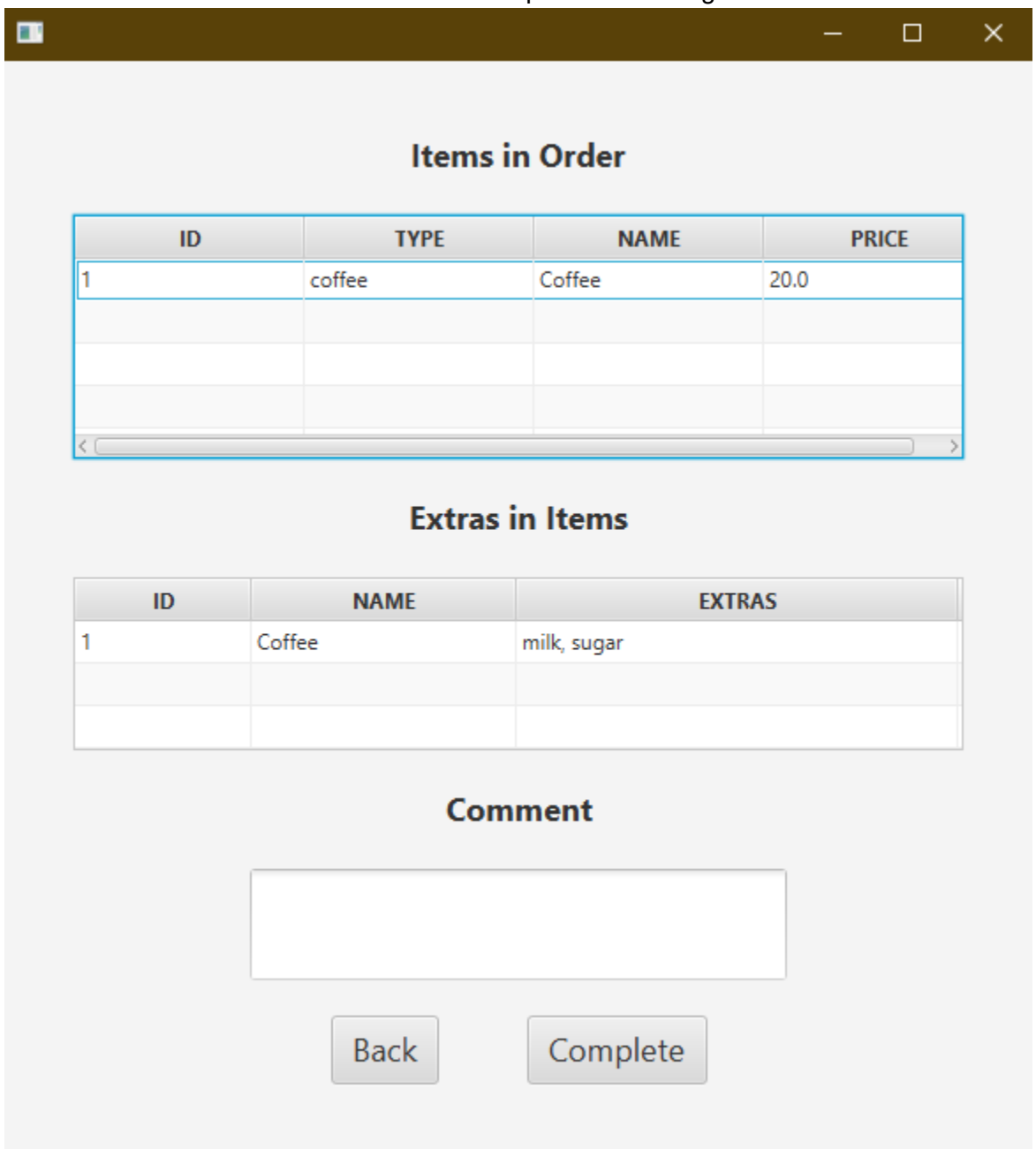
ExtraView



Alert with the order number.



Extras are correctly fetched from the database further down to the Model and eventually displayed in the OrderDetailView. This use case has passed the integration test.

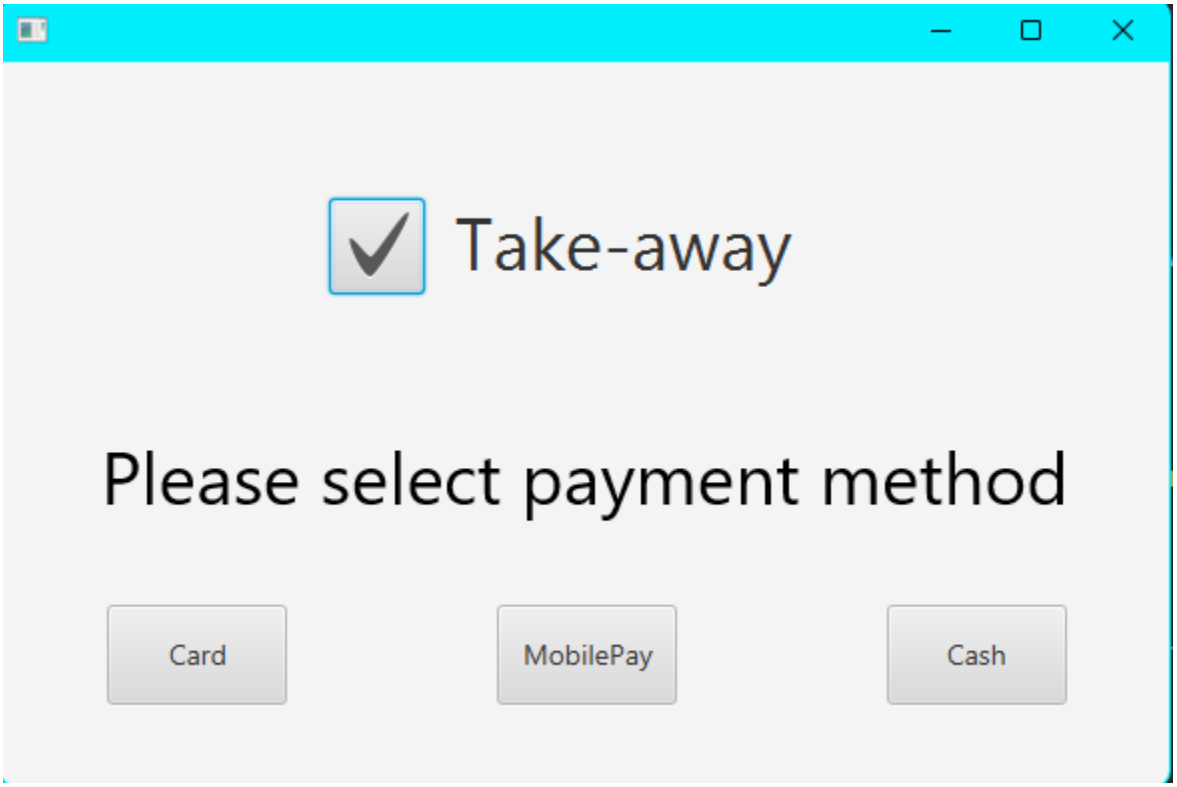


PAYMENT VIEW

Design

The PaymentView was designed to satisfy the customers wishes according to the Product Backlog items 13 and 18. The first wants to give the customers the ability of choosing the payment method they would like to proceed with three choices: Card, MobilePay or Cash. The latter is for the customers to be able to choose whether they want their order to be prepared for on-site consumption or take-away.

Paymentview



For this purpose, a checkbox was used. For selecting the payment method, the customer can click on one of the three buttons, each representing a different payment method.

Implementation

Payments done through Card and MobilePay are implemented in the same way: an if statement is checking the selectedProperty of the checkbox, and setting the order as Take-away if it is true. This feature was added in the final parts of the construction phase, therefore it was solved by adding “TAKE-AWAY” to the description of the order.

Methods in the PaymentViewController class

```
@FXML private void onCard()
{
  if (isTakeAway())
  {
    paymentViewModel.setIsTakeAway();
  }

  orderNumberMessageNonCash(paymentViewModel.payWithCardOrMP());

}

@FXML private void onMobilePay()
{
  if (isTakeAway())
  {
    paymentViewModel.setIsTakeAway();
  }
  orderNumberMessageNonCash(paymentViewModel.payWithCardOrMP());
}

@FXML private void onCash()
{
  if (isTakeAway())
  {
    paymentViewModel.setIsTakeAway();
  }
  orderNumberMessageCash(paymentViewModel.payWithCash());
}
```

The methods from the viewmodel are calling the method payForOrder(boolean isCash). The parameter is set to true inside payWithCash() and to false inside (payWithCardOrMP).

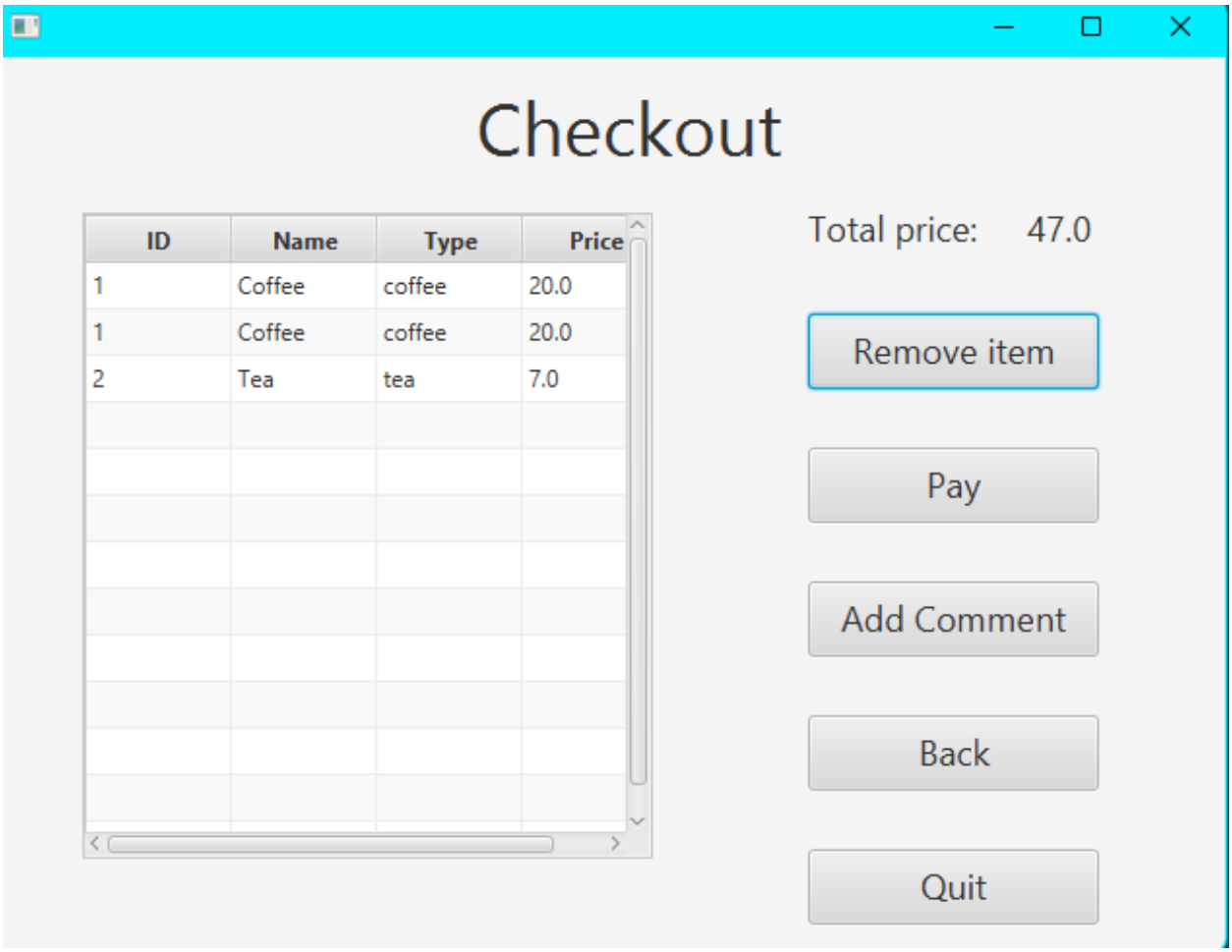
payForOrder(boolean isCash) inside the ModelManager

```
@Override public int payForOrder(boolean isCash)
{
  if (isCash)
  {
    order.setStatus("unpaid");
  }
  return submitOrder();
}
```

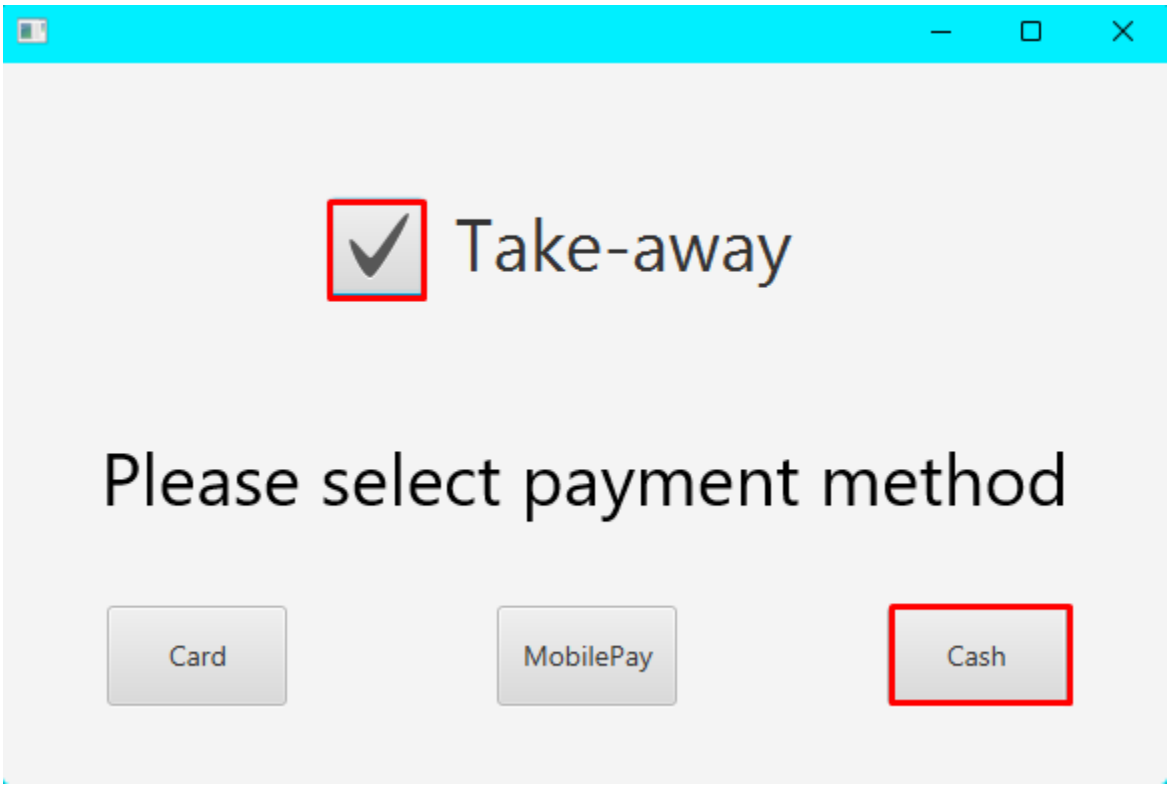
If the order is to be paid in cash, the status is set to unpaid and it goes to the cashier, otherwise the order is submitted.

Integration Testing

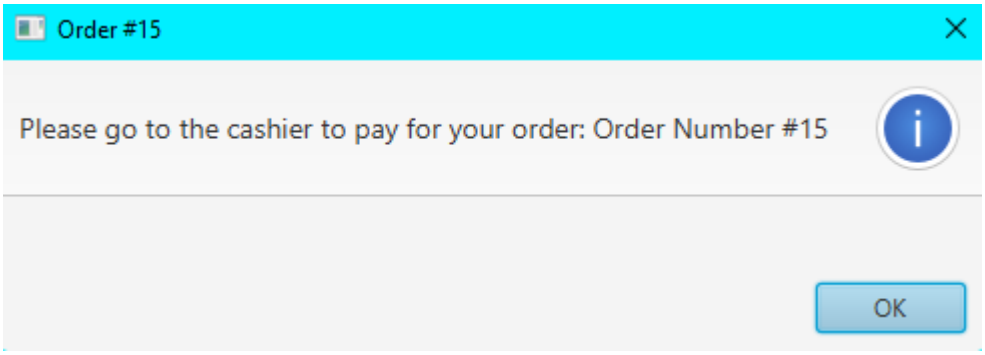
The customer is adding three items to an order and clicking Pay


Checkout
ID
Name
Type
Price
1
Coffee
coffee
20.0
1
Coffee
coffee
20.0
2
Tea
tea
7.0
Total price: 47.0
Remove item
Pay
Add Comment
Back
Quit

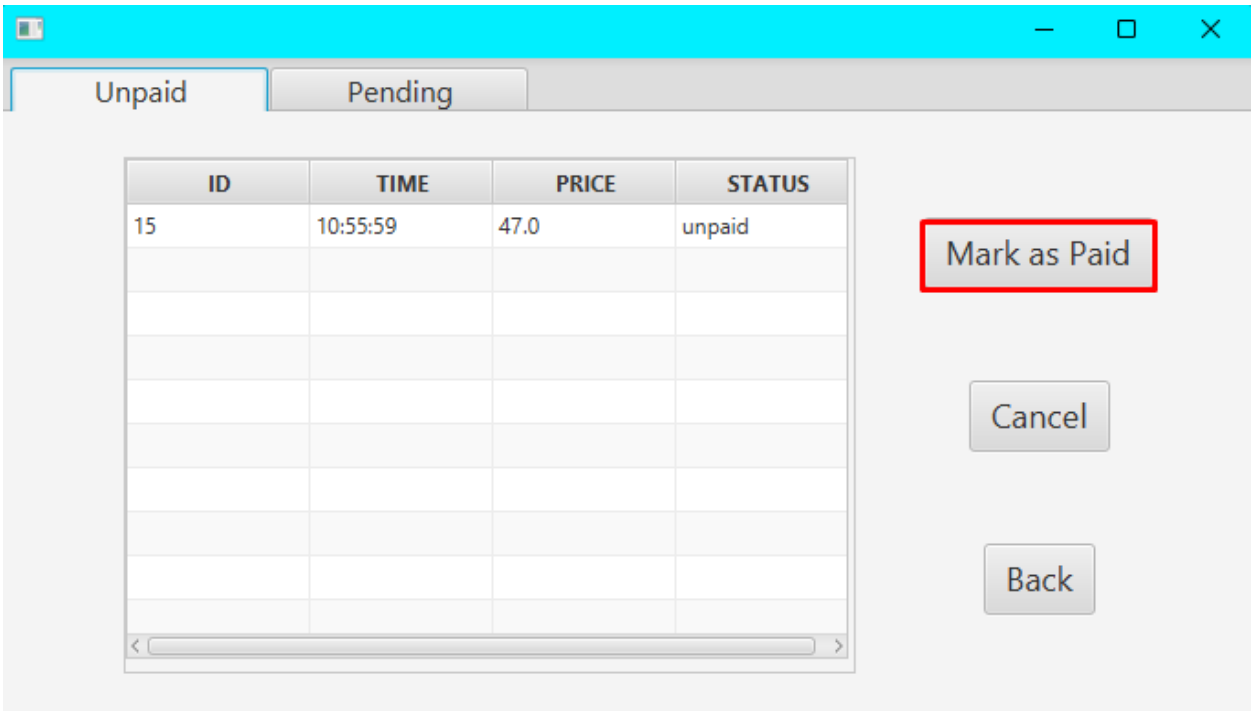
The customer is selecting Take-away, and Cash payment



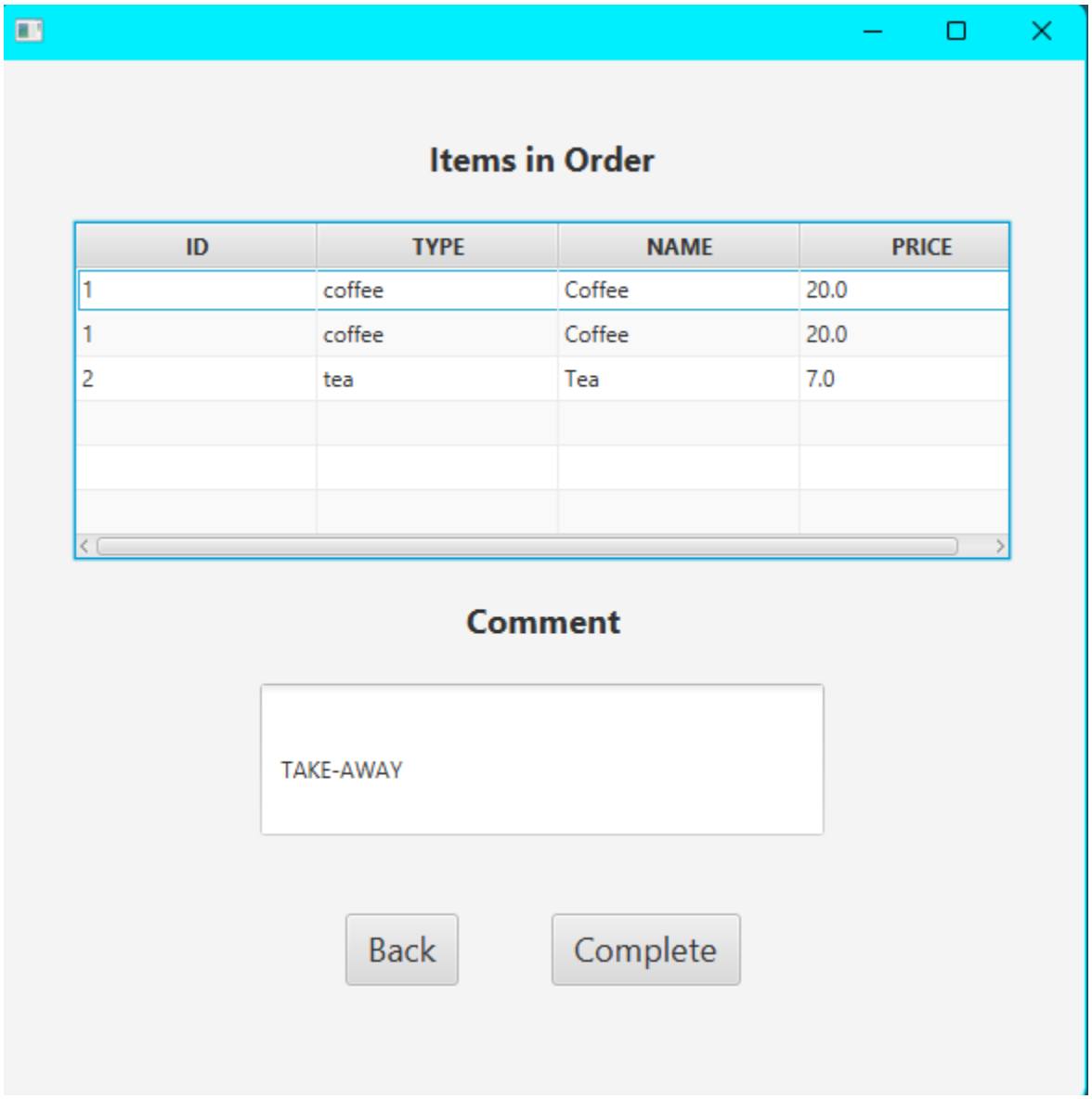
An alert is displayed for the customer with order ID #15



The cashier marks order #15 as paid after the customer handed him the cash



The barista receives the pending order #15. “TAKE-AWAY” is added to the description



Order #15 is in the database with the description

	order_id ▲ 1	comment	datetime	price	status
1	1		2022-05-20 11:48:07.643609	28	completed
2	2		2022-05-20 13:18:02.708809	34	completed
3	3	TAKE-AWAY	2022-05-20 13:22:35.601387	40	completed
4	4		2022-05-20 13:23:17.256170	20	completed
5	5	TAKE-AWAY	2022-05-20 13:25:27.644014	20	completed
6	6	TAKE-AWAY	2022-05-20 13:26:29.622582	20	completed
7	7	TAKE-AWAY	2022-05-20 13:39:10.092066	20	completed
8	8	TAKE-AWAY	2022-05-20 13:39:53.208484	40	completed
9	9	TAKE-AWAY	2022-05-23 09:40:57.785062	40	completed
10	10	TAKE-AWAY	2022-05-23 09:41:19.154698	20	completed
11	11		2022-05-23 09:41:33.697519	47	completed
12	12		2022-05-23 10:17:50.655096	40	completed
13	13	TAKE-AWAY	2022-05-23 10:19:01.806000	40	completed
14	14		2022-05-23 10:22:03.184382	14	completed
15	15	TAKE-AWAY	2022-05-23 10:55:59.668725	47	pending