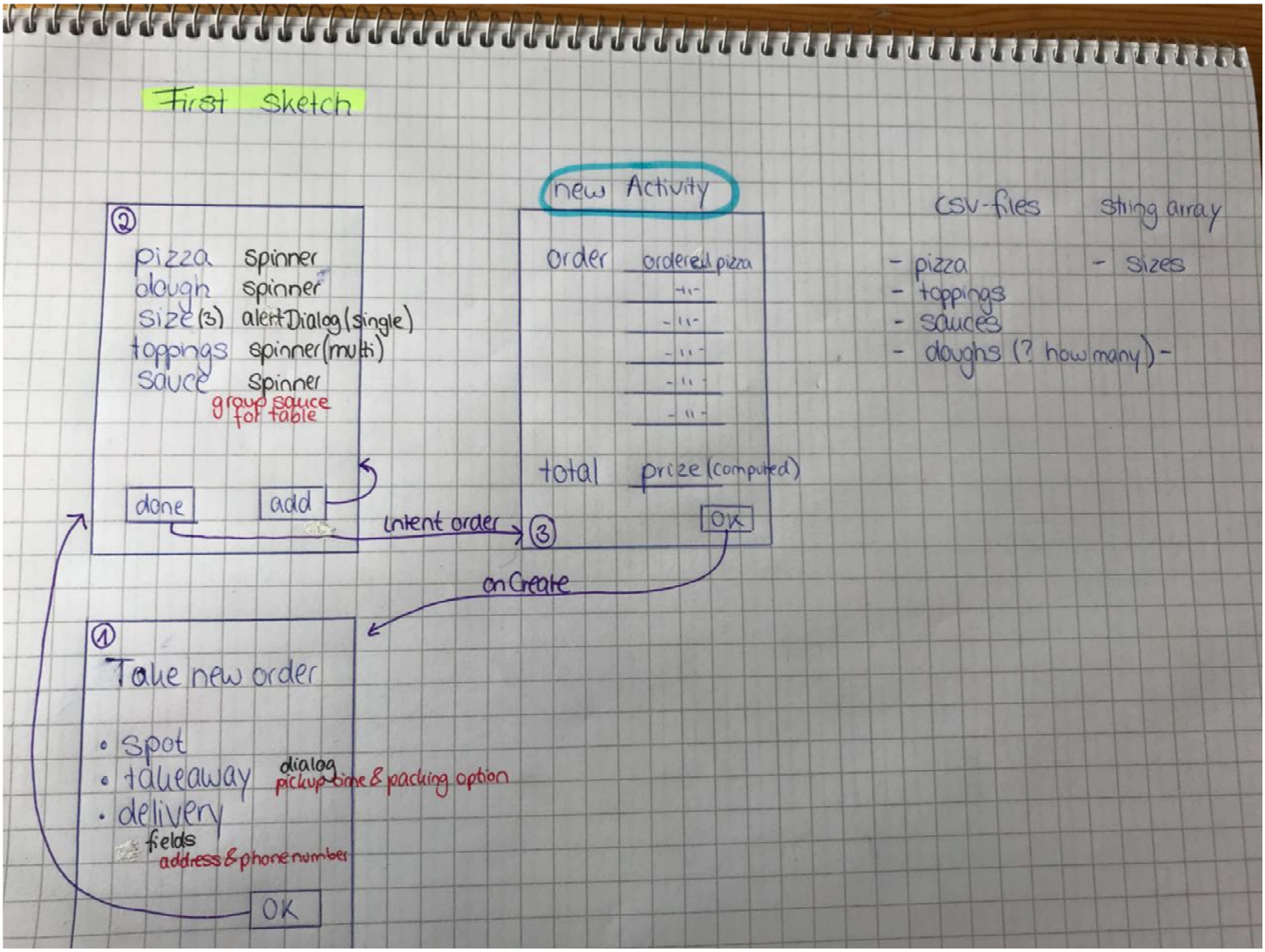
Content of Documentation

Design Documents

First Sketch

After reading the requirements of the examination project, we made a first sketch to visualize them. Even though we changed quite a few things in the process of programming because we found better solutions, we decided to show our whole development process in this document.



Use Cases

Use Case customer(s) come into pizzeria to order

Actor: 1 to multiple customers, waiter, pizza shop

1. Customer(s) enter pizzeria
2. Waiter asks customer(s) which order type he wants
3. Customer(s) tells the waiter which one
    a. In the Pizzeria
        1.Waiter enters the order type
        2.Waiter enters table number

3.Return to use case at 4)
   b. Takeaway
      1.Waiter enters the order type
      2.Return to use case at 4)

Repetition (4.-6.) until every consumer ordered

4. Customer tells the waiter what he wants
5. The waiter selects the ordered pizza, dough and size
6. Optional: Customer orders additional toppings and/or sauce
   a. If 3a), it is possible to have sauce for the whole table.
      1. Customer picks sauce
      2. Customer orders sauce for the whole table
      3. Return to use case at 9)
   b. Sauce per customer
      1.Customer picks sauce
      2.Return to use case at 7)
7. If 2b) selected, Customer tells pickup time and packing option
8. If 2b) selected, Waiter enters information
9. Waiter submits order to pizza shop
10. Pizza shop receives order
11. Pizza shop sends confirmation of order
12. Waiter receives confirmation
13. End of use-case

Use Case customer calls pizzeria to order

Actor: 1 customer, waiter, pizza shop

1. Telephone rings in pizzeria
2. Waiter answers call
3. Customer tells the waiter which one
   a. In the Pizzeria
      1.Waiter enters the order type
      2.Waiter enters table number
      3.Return to use case at 4)
   b. Takeaway
      1.Waiter enters the order type
      2.Return to use case at 4)
   c. Delivery
      1.Waiter enters the order type
      2.Return to use case at 4)

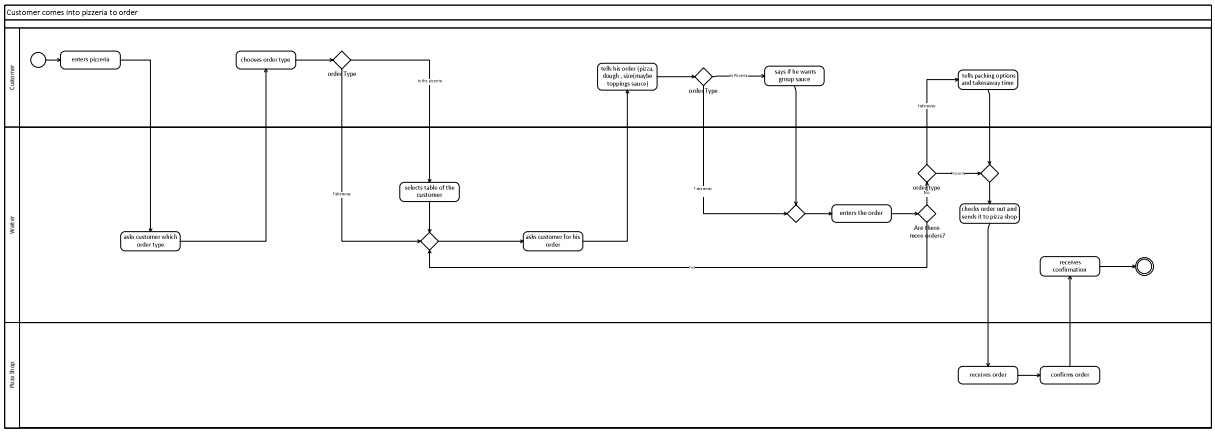
Repetition (4.-6.) until consumer is finished with his order

4. Customer tells the waiter what he wants
5. The waiter selects the ordered pizza, dough and size
6. Optional: consumer orders additional toppings and/or sauce
   a. If 3a), it is possible to have sauce for the whole table.
      1. Customer picks sauce
      2. Customer orders sauce for the whole table
      3. Return to use case at 11)
   b. Sauce per customer
      1.Customer picks sauce
      2.Return to use case at 7)
7. If 2b) selected, Customer tells pickup time and packing option

8. If 2b) selected, Waiter enters information
9. If 2c) selected, Customer tells delivery address and phone number
10. If 2c) selected, Waiter enters information
11. Waiter submits order to pizza shop
12. Pizza shop receives order
13. Pizza shop sends confirmation of order
14. Waiter receives confirmation
15. End of use-case

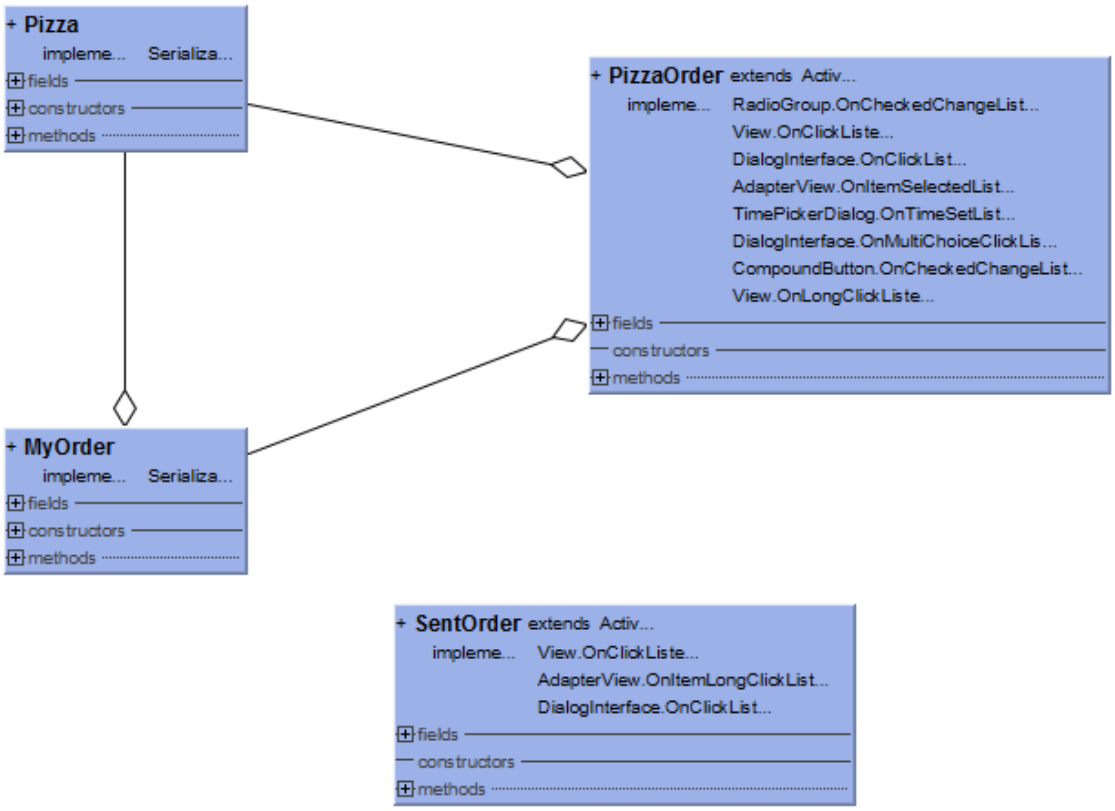
Activity Diagram Customer comes into pizzeria

The activity diagram shows the use case of the customer coming into the pizzeria and placing his order. It is created in Visio and the original is included in the documents folder to be able to read it better.



Class Diagram

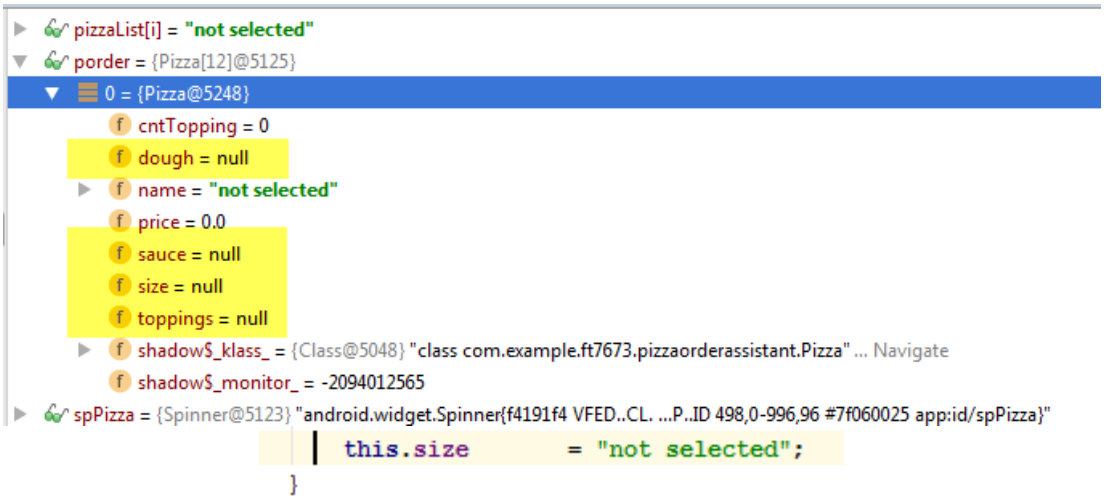
The class diagram shows the relationships of class in our developed application. The following pictures portrays the overview. To look closer into the fields, constructor and methods of each class, the file is separately added to the document folder of the project. It might happen when opening the class diagram file that there are synchronizing problems. These can be solved by moving one of the class boxes and the aggregations connect correctly again.



Testing Report

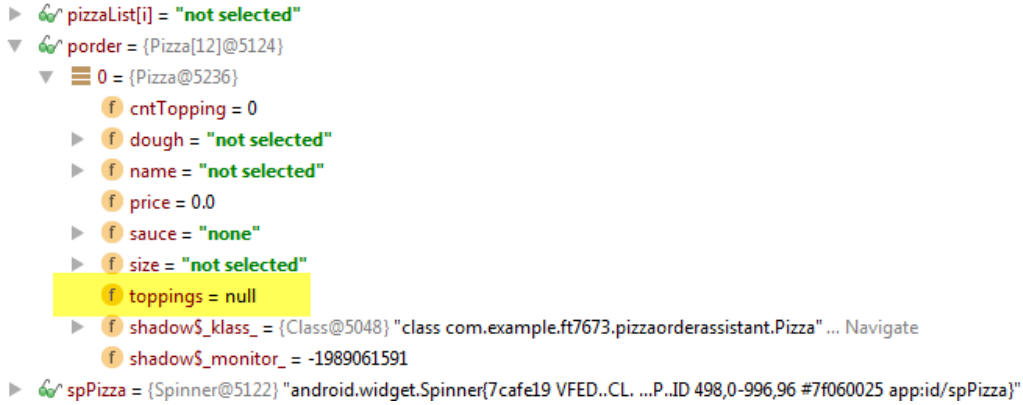
In the process of programming a major part is the testing of the software. This allows to catch mistake while the process proceeds.

One of the beginning difficulties was to get the spinner items up and running. The problem occurred since the string-arrays were longer than the list of items that were put in. Because some part of the string arrays were filled with NULL values. However, the processing of null values creates errors in the following functions. That is the reason why this occasion needs to be catched in advance.
This was fixed by doing an array with the optimal size by looking how many items there are included in the csv and creating the array on runtime.



The name of the pizza is set to “not selected” but as long as no pizza is selected the user is not able to proceed because the rest of the layout is hidden. Anyway, this does not solve the problems for the other variables. It is solved by the implemented solution of setting the pizza dough and the pizza size to “not selected” and the sauce to “none” in the constructor. This is due to the fact that it is necessary to select a dough and size for the pizza but the consumer is not obligated to select a sauce.
When the user want to proceed but the mandatory values of dough and size are based on the written strings. If they still include the string “not selected”, a Toast appears on the screen to remind the user to enter the missing values before proceeding. If not, the user can proceed.

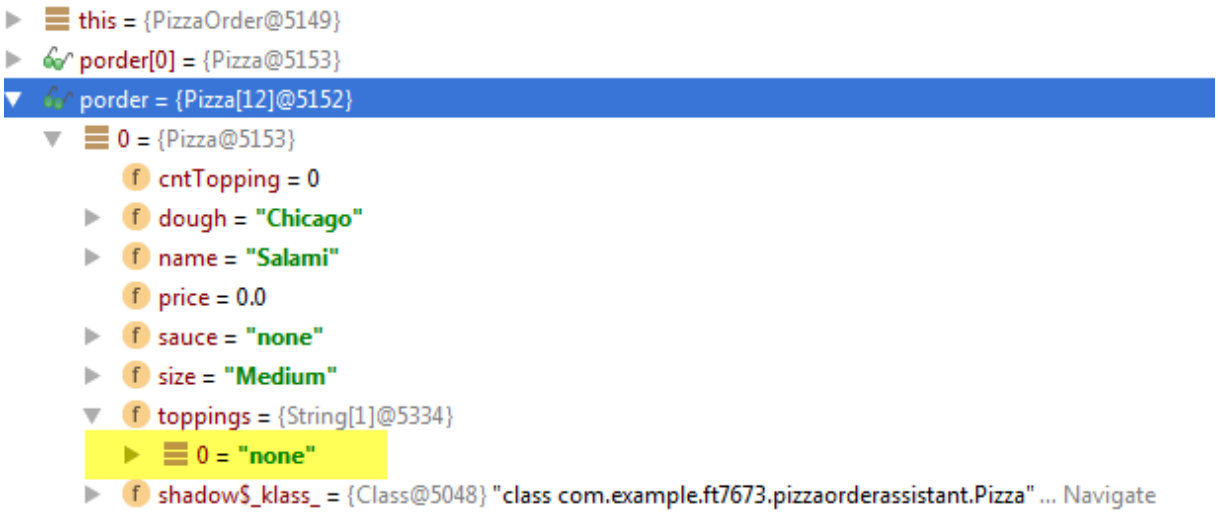
This solved half of the problem except the String array toppings value.



As the consumer is also not obligated to add additional toppings to its pizza, the goal is to set the first String in the String array to “none”. This logic is added to the function “fillTops()”. It checks if there have been any selected items in the toppings. If yes it just writes them down but if not the none value is written down.

```
private String[] fillTops() {
    int counter = 0;
    for (int i = 0; i < toppingList.length; i++) {
        if (boolTop[i] == true) {
            counter++;
        }
    }
    String[] help = new String[counter];
    counter = 0;
    for (int i = 0; i < toppingList.length; i++){
        if (boolTop[i] == true) {
            help[counter] = toppingList[i];
            counter++;
        }
    }
    if(counter == 0){
        help = new String[1];
        help[0] = "none";
    }
    return help;
}
```

This amendment to the existing function solved the problem as shown in the following screenshot.



After getting the Strings for the pizza values, the prices needed to be added up to calculate the total in the end. This requires reading them out of the CSV-file, too. However, there were some problems regarding getting the prices into the same array as the names of the sauces, pizzas etc. The first thought was to put it in a two dimensional array but this idea was abandoned rather quickly. This was due to the fact that we weren’t able to put a two dimensional array into the spinners. To solve this problem an extra double array was initialized and all the values of the csv-files were put in the same position as the associated names.

This solved the problem of carrying the prices around and was also done with the toppings.

```
not selected;0.00
Magherita;4.50
Salami;6.50
Ham;6.50
Funghi;7.00
```

```
while ((lineReader = reader.readLine()) != null) {

    output = lineReader.split( regex: ";");


    pizzaList[counter]  = output[0];
    pizzaPrice[counter] = Double.parseDouble(output[1]);
```

The next problem which had to be solved is about the cleaning of the code when the application was more or less working. Due to the fact that there were still some strings that where directly written into the coding as you can see in the following screenshot with the sauce these should be replaced with the strings out of the resource folder. An Example for this is how it is done with the dough variable. Later in the cleaning, it was tried to start the application but after calling the second layout from the first one the application crashed. After a long time of searching with the debugger, the reason was found in the Pizza constructor with the dough. Even though we invested quite some time in finding a solution for this we couldn't which resulted in the act of writing it back to the way it was. We did some research and the reason we couldn't call the string resources was to the fact that they are dynamic and not static.
While testing we tried to work around it with an onCreate() to be able to call the resources. Unfortunately, this just caused more problems. Additionally, this class was never meant to be used as another activity but just to simplify the handling of pizza object.

```
public class Pizza extends Activity implements Serializable{

    private String name;
    private String dough;
    private String size;
    private String[] toppings;
    private String sauce;
    private boolean tableSauce;
    private double price;

    public Pizza(String name){
        this.name = name;
        this.dough      = getResources().getString(R.string.stringNotSelected);
        this.sauce      = "none";
        this.size       = "not selected";
```


While trying to verify that our coding was working properly regarding the amount of pizzas who could been added and scroll through all of them in the "listview", some new bugs occurred to us. The tested features worked as planned, but the when checking the order information of the pizzas a problem with the listed topping appeared. None of the ordered pizzas should include toppings. The order information of the first pizza was shown correctly. But as shown in the following screenshots, the list of toppings was just added up which led to the other screenshot which is taken from the last pizza. The pizza just received the toppings from the pizza above it and added the toppings which she included on top of it. Like this there was a huge string of none toppings from the 24 pizzas before the last one.

This behavior was caused by the initialization of the String “saveTop” outside of the for-loop.

```
public String[] getMoreOrder(){
    String[]    toSave     = new String[order.length];
    String      saveTop    = "";

    for(int i = 0; i < order.length; i++){
        if(order[i] == null){
            break;
        }else{
            for(int j = 0; j < order[i].getPizzaToppings().length; j++){
                if (j+1 != order[i].getPizzaToppings().length){
                    saveTop = saveTop + order[i].getPizzaToppings()[j] + ", ";
                }else{
                    saveTop = saveTop + order[i].getPizzaToppings()[j];
                }
            }
            toSave[i] = "Dough: " + order[i].getPizzaDough() + "\n" +
                        "Toppings: " + saveTop + "\n" +
                        "Sauce: " + order[i].getPizzaSauce() + "\n" +
                        "Price: " + order[i].getPizzaPrice() + " €";
        }
    }
    return toSave;
}
```

By placing this inside of the for-loop, it always overwrites the old one and solves this unwished behavior.

```
public String[] getMoreOrder(){
    String[]    toSave     = new String[order.length];
    String      saveTop;

    for(int i = 0; i < order.length; i++){
        saveTop = "";
        if(order[i] == null){
            break;
        }else{
            for(int j = 0; j < order[i].getPizzaToppings().length; j++){
                if (j+1 != order[i].getPizzaToppings().length){
                    saveTop = saveTop + order[i].getPizzaToppings()[j] + ", ";
                }else{
                    saveTop = saveTop + order[i].getPizzaToppings()[j];
                }
            }
            toSave[i] = "Dough: " + order[i].getPizzaDough() + "\n" +
                        "Toppings: " + saveTop + "\n" +
                        "Sauce: " + order[i].getPizzaSauce() + "\n" +
                        "Price: " + order[i].getPizzaPrice() + " €";
        }
    }
    return toSave;
}
```

Another behavior was caused when the pizza was changed after entering the information about size and dough already. This was caused by the new instantiation of a new pizza object which does not have the information. For this reason, the function “checkPizzaInfo()” would always return false and would prevent proceeding.

```
@Override
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
    if (adapterView == spPizza) {

        Pizza help = new Pizza(pizzaList[i]);
        porder[cntPiz] = help;
        priceHelperPz = pizzaPrice[i];

        if(spPizza.getSelectedItem().toString().equals(getResources()
                .getString(R.string.stringNotSelected))){
            orderView.setVisibility(View.INVISIBLE);

        }else {
            orderView.setVisibility(View.VISIBLE);
        }
    }
```

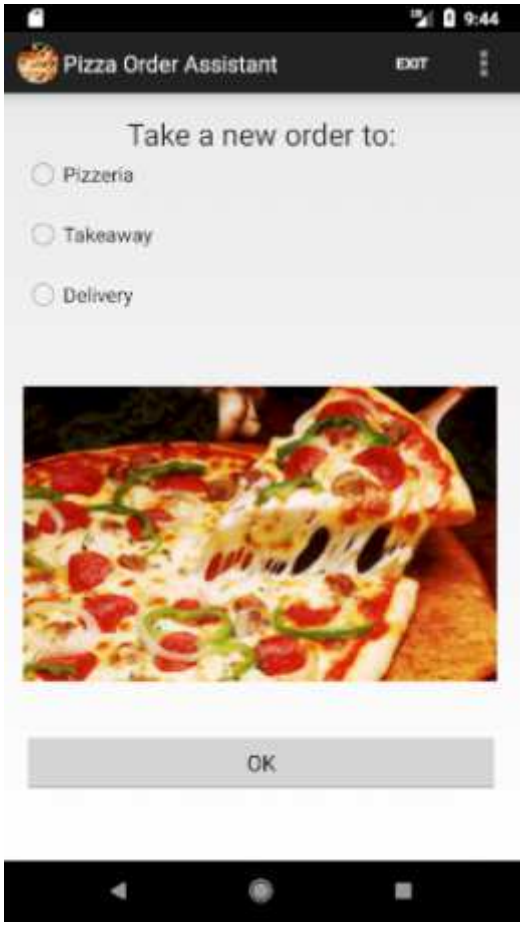
But to keep the option for the customer to change his mind on the ordering pizza, we added some helping variables which keep the information. That way the pizza can be changed but the other information are still going to be valid.

```
@Override
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
    String      hlpDough = null;
    String      hlpSize = null;
    String      hlpSauce = null;
    String[]    hlpToppings = null;
    boolean     overwritten = false;
    if (adapterView == spPizza) {
        if(porder[cntPiz] != null) {
            hlpDough = porder[cntPiz].getPizzaDough();
            hlpSize = porder[cntPiz].getPizzaSize();
            hlpSauce = porder[cntPiz].getPizzaSauce();
            hlpToppings = porder[cntPiz].getPizzaToppings();
            overwritten = true;
        }

        Pizza help = new Pizza(pizzaList[i]);
        porder[cntPiz] = help;
        priceHelperPz = pizzaPrice[i];

        if ( overwritten == true) {
            porder[cntPiz].setPizzaDough(hlpDough);
            porder[cntPiz].setPizzaSauce(hlpSauce);
            porder[cntPiz].setPizzaSize(hlpSize);
            porder[cntPiz].setPizzaToppings(hlpToppings);
            overwritten = false;
        }
```

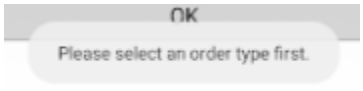
Manual

The following screenshot shows how the waiter/waitress places a new order.

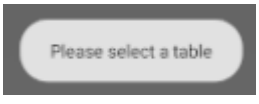
The first step to do is to select where the food should be consumed. These choices will be later important for the different outcomes like writing down a delivery address and phone number or a pickup time.



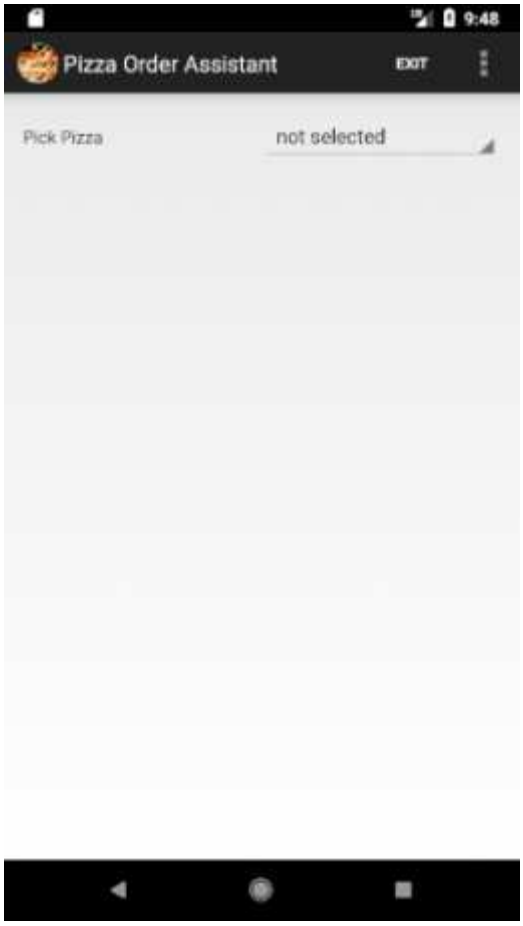
If nothing is selected, the waiter cannot proceed to the next layout and a Toast kindly ask the waiter to select an option.



If the option of eating in the pizzeria is selected, the waiter is asked to enter the table number in an AlertDialog to assign the order to a table. If he does not enter a table number, a Toast reminds him of doing so before going on.



If an order type is selected, the next layout is going to be shown. But as shown, it start with just the pizza to select. This forced the waiter to select a pizza before proceeding further and prevents empty/faulty orders.

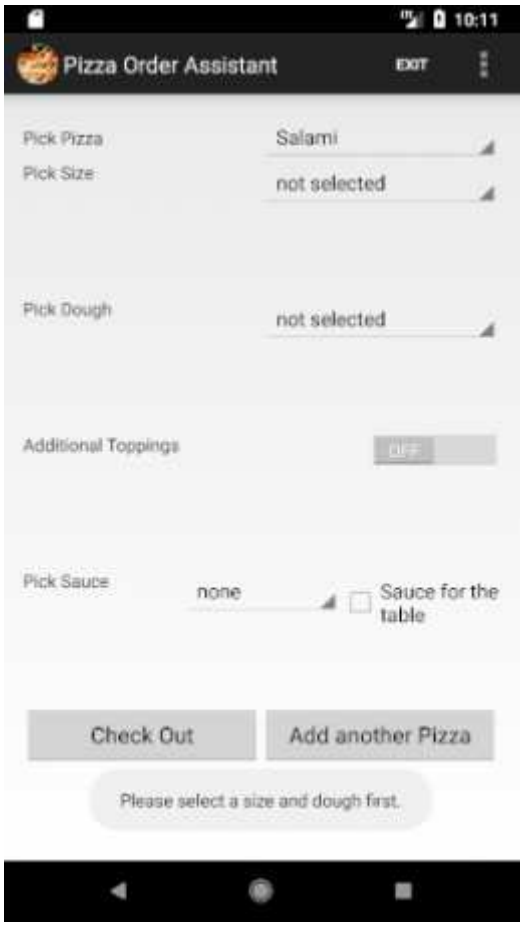


After picking a pizza out of the spinner, the whole layout will be shown. But if the pizza is put back to not selected the view will vanish again.

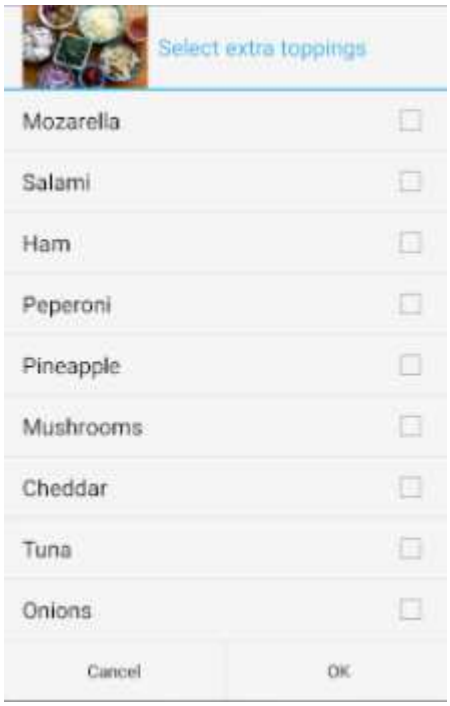
To proceed any further and use the button on the bottom the waiter has to pick at least the dough and the size from the spinners since these are necessities for the pizza. If he wants to proceed without these information, a Toast is going to tell him which information is still missing. Either both as shown in the screenshot, or just the one which is still missing.

A sauce and additional topping can also be picked for the pizza but are not necessary. The sauce is also chosen from a spinner.

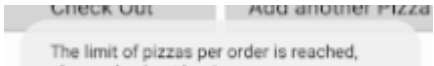
If eating in the pizzeria, the checkbox “Sauce for the table” is visible and allows to order a bigger size of sauce which is for the whole table to share. The other order types do not show this checkbox. If just the checkbox it clicked but no sauce is chosen, another Toast is going to show up to remind the waiter to enter the sauce before proceeding.



If the switch for the toppings is pressed, an additional AlertDialog will open and toppings can be selected. To save the selection of toppings, it is needed to press the OK button of the AlertDialog. If nothing is selected the switch button will stay off, otherwise it will be turned on.
If the cancel button is pressed all of the clicked toppings will disappear and the toppings will be empty again.



After taking the order for one pizza, it is possible to take orders for up to 24 more pizzas by pressing the “add another pizza”. If the customer wants to order more than this, the user has to check the order out, send it to the pizza shop and start the next order. A Toast is going to inform him if this situation accrues.



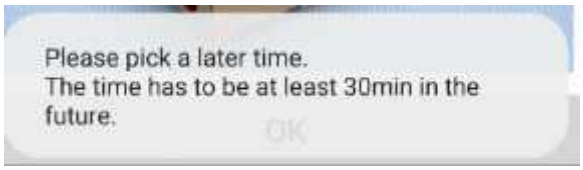
When all orders are added, it is time to check out through the “Check Out” button. Depending on the order type, the next layout or new activity is shown.

If the option of takeaway was chosen in the beginning, it is this one.

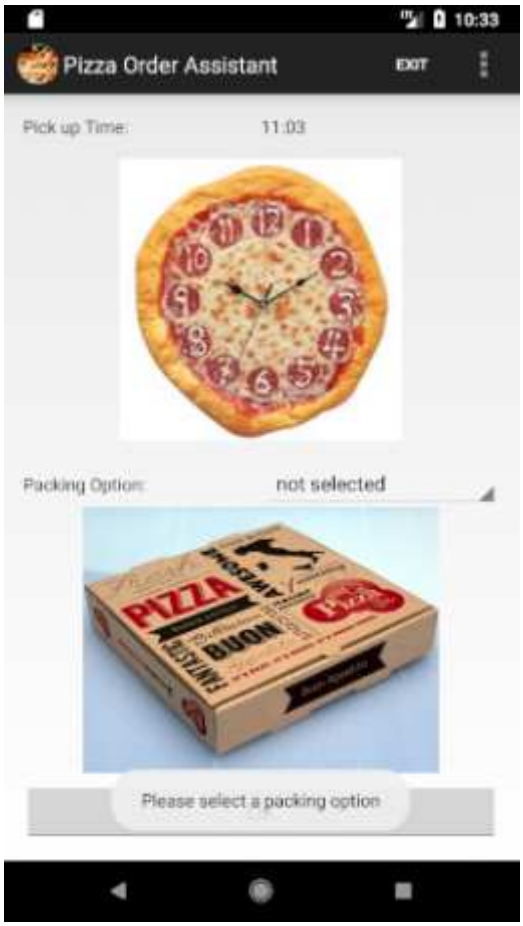
The packing option can be selected out of a spinner.

By clicking on the phrase “Long click here to set the time”, a time picker opens to choose a time. The entered time is already set to 30min in the future because that is the minimum time it takes to produce the order.

If the time is in the past or not at least 30 minutes in the future, a Toast will ask the waiter to pick a later time and erase the selected time.



As long as not both fields are successfully filled out, it is not possible to proceed and a Toast will ask the waiter to fill out the missing fields. If it is filled out, a new Activity will be launched by pressing the OK button. This sends out the order.



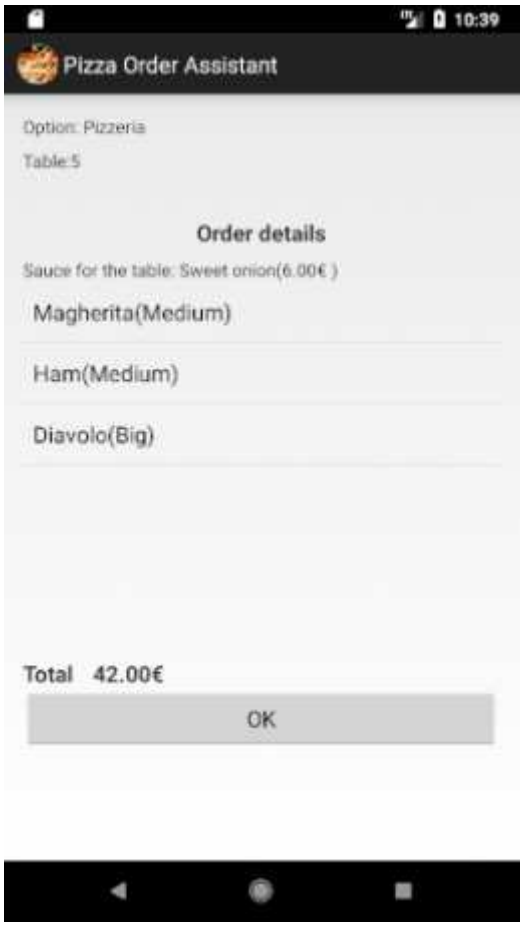
If the option of delivery was chosen at the beginning, it is the following layout to show.
The delivery address needs to be filled out with the address and the phone number of the customer. The phone number field only takes digits. Both fields need to be filled out to proceed. If this is not the case, Toast are going to show up as in the takeaway option to remind the waiter to enter the missing information. Otherwise, the new Activity will be launched by pressing the OK button.

If the option of eating in the pizzeria was chosen, the new Activity is going to be launched directly.

This new Activity shows the pizza shop the whole order. The top included information about the order type and their information. That means that if it is eaten in the pizzeria, the table number. If it a takeaway order, the pickup time and the packing option. If it is for delivery, the delivery address and phone number.

Underneath the order is listed. The first line included (if ordered) the sauces for the table. Afterwards the pizzas are listed with name and size.

The last line shows the total of the whole order.



By long clicking on one of the pizzas, an AlertDialog is going to show up and the display further information (dough, topping, sauce and the total pizza price). The sauce of the pizza which ordered the group sauce is set to “none” as the sauce is separately listed.

After clicking the OK button, this Activity will be closed and a confirmation message is send to the previous Activity. The Activity will be set to the beginning again and a Toast informs the waiter that the pizza shops received the order successfully.

Besides these functionalities, the application includes a menu too. This menu has an exit button to close the whole application. Furthermore, there is a cancel button to cancel the whole order and get back to the first layout.