

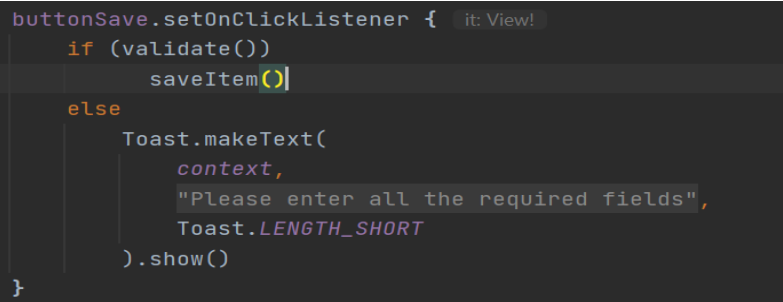
Functions

1. Encapsulate Conditionals in Functions

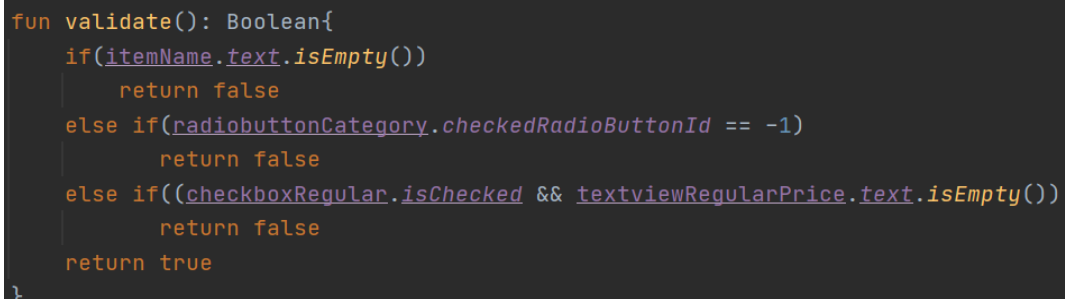
Bad practice:

```
buttonSave.setOnClickListener {
    if (itemName.text.isEmpty() && radiobuttonCategory.checkedRadioButtonId != -1 &&
        ((checkboxRegular.isChecked && textviewRegularPrice.text.isEmpty()) ||
         (checkboxMedium.isChecked && textviewMediumPrice.text.isEmpty()) ||
         (checkboxLarge.isChecked && textviewLargePrice.text.isEmpty())))
        saveItem()
    else
        Toast.makeText(
            context,
            getString(R.string.save_error_message),
            Toast.LENGTH_SHORT
        ).show()
}
```

Good practice:



```
buttonSave.setOnClickListener { it: View!
    if (validate())
        saveItem()
    else
        Toast.makeText(
            context,
            "Please enter all the required fields",
            Toast.LENGTH_SHORT
        ).show()
}
```



```
fun validate(): Boolean{
    if(itemName.text.isEmpty())
        return false
    else if(radiobuttonCategory.checkedRadioButtonId == -1)
        return false
    else if((checkboxRegular.isChecked && textviewRegularPrice.text.isEmpty())
        return false
    return true
}
```

2. Do not Repeat

Bad practice:

Repeating the same code in more than one place.

```

if (orders.isNotEmpty()) {
    with(binding) {
        this: FragmentOrdersBinding
        ordersHistoryView.visibility = View.VISIBLE
        noOrdersView.visibility = View.GONE
    }
}

val orders = databaseHelper.filterOrdersByDate(selectedDate)
with(binding) {
    this: FragmentOrdersBinding
    ordersHistoryView.visibility = View.VISIBLE
    noOrdersView.visibility = View.GONE
}
recyclerView.adapter = OrdersAdapter(orders, ADMIN, requireContext())

```

Good Practice:

```

if (orders.isNotEmpty()) {
    setOrdersHistoryScreen()
    recyclerView.adapter = OrdersAdapter(orders, USER, requireContext())
} else {
    setEmptyOrdersScreen()
}

val orders = databaseHelper.filterOrdersByDate(selectedDate)
setOrdersHistoryScreen()
recyclerView.adapter = OrdersAdapter(orders, ADMIN, requireContext())

private fun setOrdersHistoryScreen() {
    with(binding) {
        this: FragmentOrdersBinding
        ordersHistoryView.visibility = View.VISIBLE
        noOrdersView.visibility = View.GONE
    }
}

```

3. A Function should have only one responsibility.

Bad practice:

```

public void displayUser(int userId){
    Users user=null;
    for (Users u:LibraryDb.getUsers()) {
        if(userId==u.getUserId())
            user=u;
    }
    System.out.println("User Id: " + user.getUserId() + ", User Name: " + user.getName() + " ")
}

```

displayUser function should be responsible only for displaying the user but here it is getting the user from DB and Displaying.

Good Practice:

```
public Users getCurrentUser(int userId) {
    Users currentUser=null;
    for (Users u:LibraryDb.getUsers()) {
        if(userId==u.getUserId())
            currentUser=u;
    }

    return currentUser;
}

public void displayUser(int userId){
    Users user = getCurrentUser(userId);
    System.out.println("User Id: " + user.getUserId() + ", User Name: " + user.getName() +
}
```

4. Do not use Flag Arguments

Bad Practice:

```
fun saveItem(isAddItem: Boolean){
    if(isAddItem){
        AdminHandler.addPizza(name,
            R.drawable.tandooripaneer,
            Category.valueOf(category),
            sizeAndPrice,requireContext())
        listener.refreshPizzaList()
    }
    else
    {
        AdminHandler.updatePizza(
            selectedpizza.id,
            name,
            R.drawable.tandooripaneer,
            sizeAndPrice,
            Category.valueOf(category),requireContext()
        )
        listener.refreshPizzaList()
    }
}
```

Here single responsibility is violated.

Good Practice:

Instead of passing a boolean value isAddItem to saveItem function, create two functions addItem() and editItem().

```
isTaskEdit = args.getInt(KEY_TASK_TYPE,TASK_ADD_EVENT) == TASK_EDIT_EVENT;
```

```
if(isTaskEdit)
    editItem()
Else
    addItem()
```

