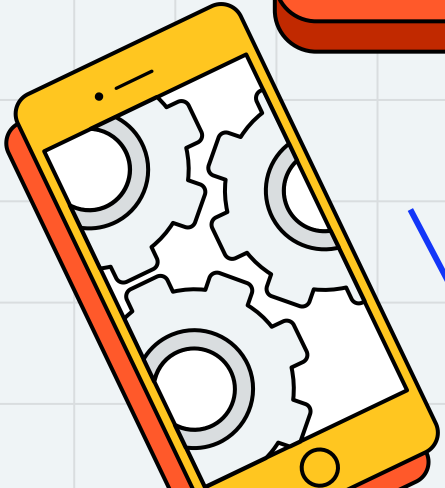
# Application Programming

Hend Alkittawi

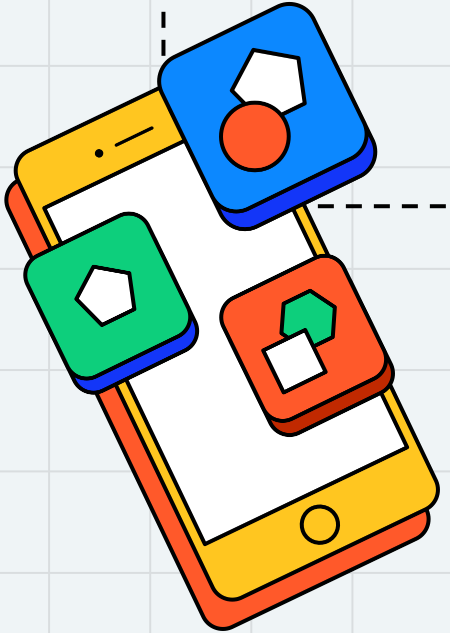


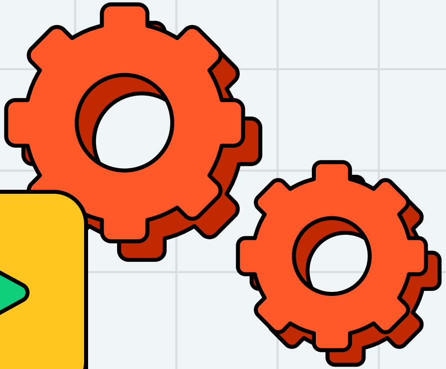
# MVC Architectural Pattern

Model-View-Controller Architectural Pattern

# CODE DEMO

- Discuss the design of a bank management app



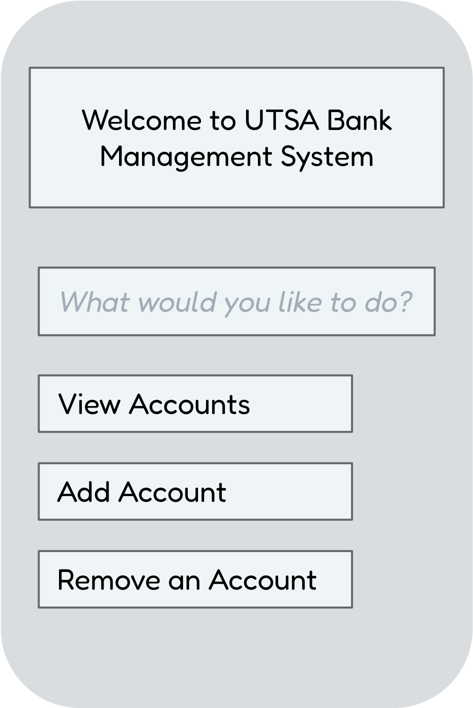


# MODEL-VIEW-CONTROLLER

```
Welcome to UTSA Bank Management System
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
1
Name: UTSA Bank
Account - name: Ash type:  checking
Account - name: Sam type:  saving
Account - name: Jen type:  checking
Account - name: Mia type:  saving
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
3
Enter account name:
Mia
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
2
Enter name and type:
Jim checking
```

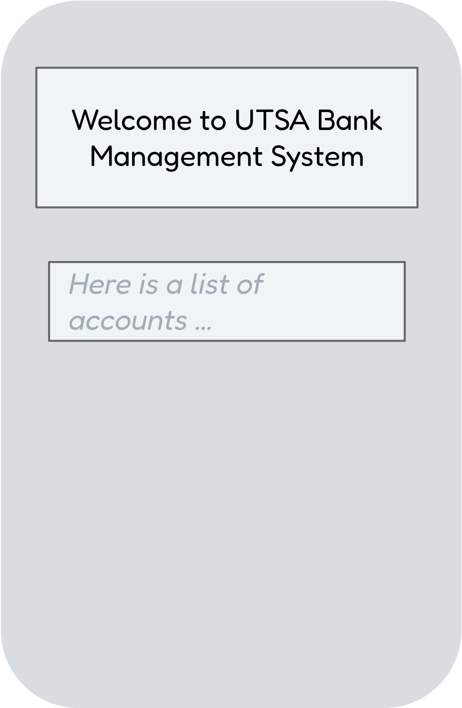
```
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
1
Name: UTSA Bank
Account - name: Ash type:  checking
Account - name: Sam type:  saving
Account - name: Jen type:  checking
Account - name: Jim type: checking
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
4
```

# MODEL-VIEW-CONTROLLER



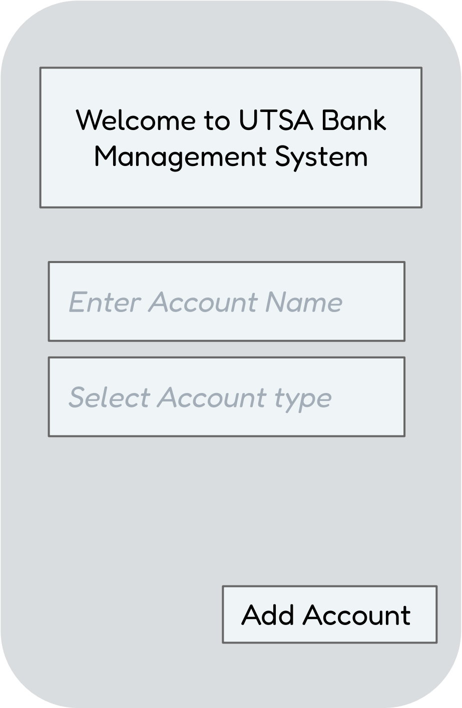
Welcome to UTSA Bank Management System

*What would you like to do?*



Welcome to UTSA Bank Management System

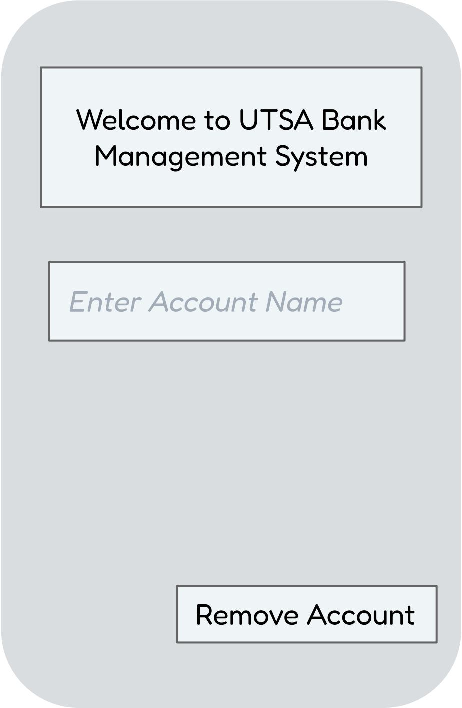
*Here is a list of accounts ...*



Welcome to UTSA Bank Management System

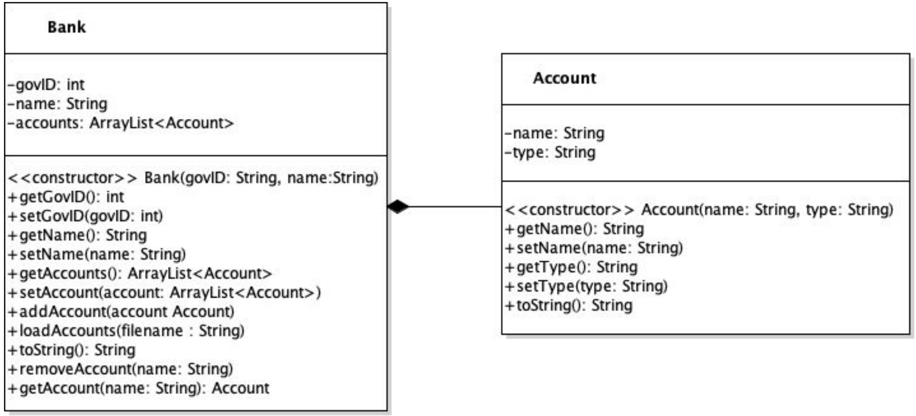
*Enter Account Name*

*Select Account type*



Welcome to UTSA Bank Management System

*Enter Account Name*



Bank

-govID: int
-name: String
-accounts: ArrayList<Account>

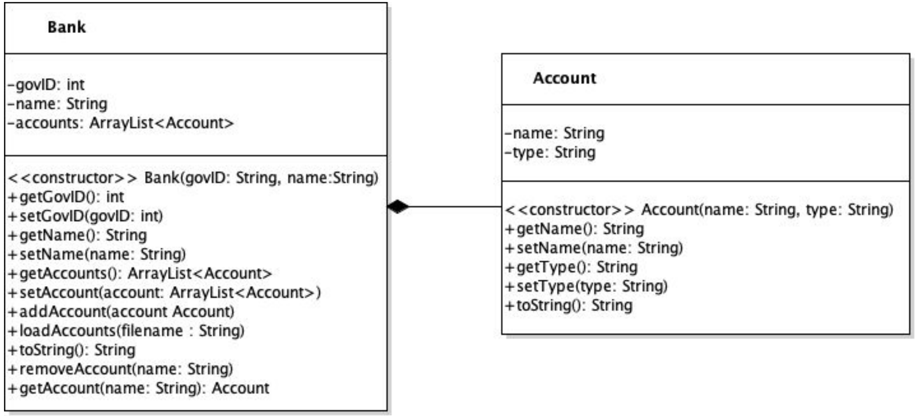
<<constructor>> Bank(govID: String, name:String)
+getGovID(): int
+setGovID(govID: int)
+getName(): String
+setName(name: String)
+getAccounts(): ArrayList<Account>
+setAccount(account: ArrayList<Account>)
+addAccount(account Account)
+loadAccounts(filename : String)
+toString(): String
+removeAccount(name: String)
+getAccount(name: String): Account

Account

-name: String
-type: String

<<constructor>> Account(name: String, type: String)
+getName(): String
+setName(name: String)
+getType(): String
+setType(type: String)
+toString(): String

```
Welcome to UTSA BankManagement System
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
1
Name: UTSA Bank
Account - name: Ash type:  checking
Account - name: Sam type:  saving
Account - name: Jen type:  checking
Account - name: Mia type:  saving
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
3
Enter account name:
Mia
What would you like to do?
1. Display Accounts
2. Add Account
3. Remove Account
4. Exit
2
Enter name and type:
Jim checking
```



**Bank**

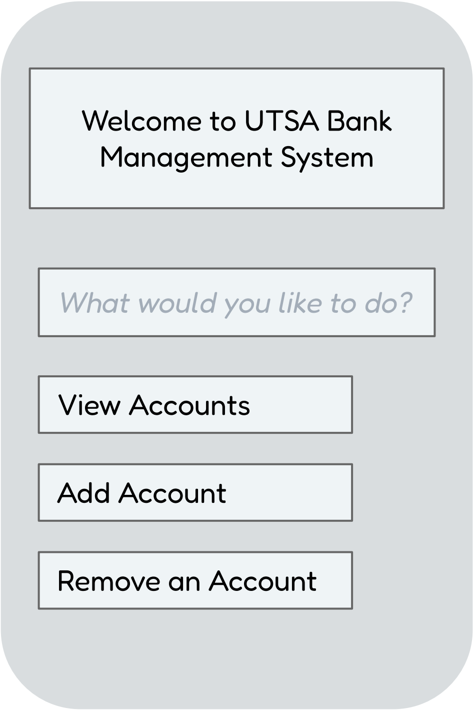
-govID: int
-name: String
-accounts: ArrayList<Account>

<<constructor>> Bank(govID: String, name:String)
+getGovID(): int
+setGovID(govID: int)
+getName(): String
+setName(name: String)
+getAccounts(): ArrayList<Account>
+setAccount(account: ArrayList<Account>)
+addAccount(account Account)
+loadAccounts(filename : String)
+toString(): String
+removeAccount(name: String)
+getAccount(name: String): Account

**Account**

-name: String
-type: String

<<constructor>> Account(name: String, type: String)
+getName(): String
+setName(name: String)
+getType(): String
+setType(type: String)
+toString(): String



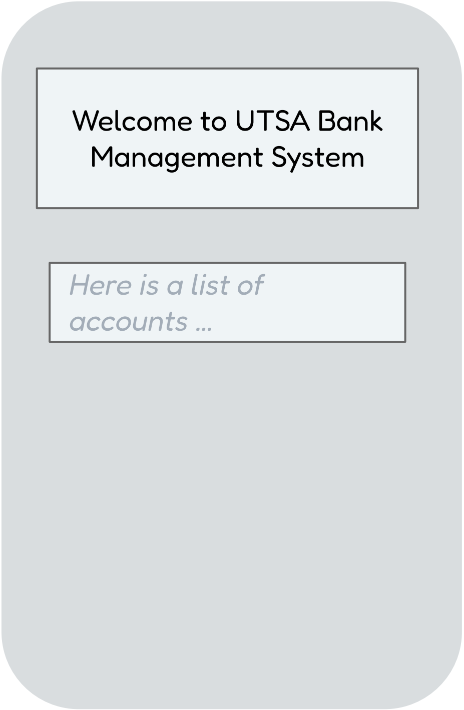
Welcome to UTSA Bank Management System

*What would you like to do?*

View Accounts

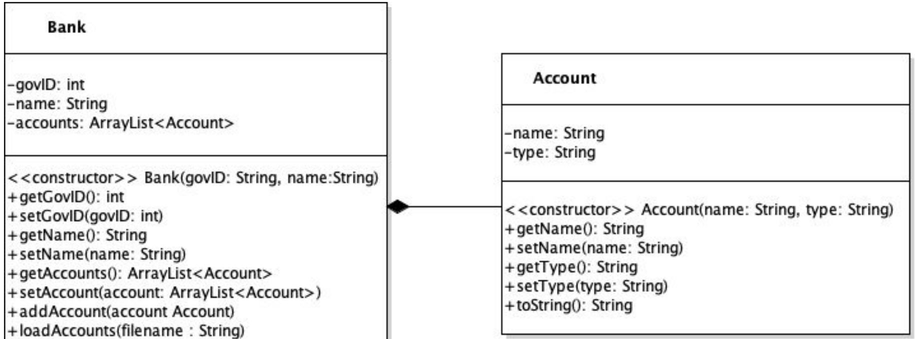
Add Account

Remove an Account



Welcome to UTSA Bank Management System

*Here is a list of accounts ...*



**Bank**

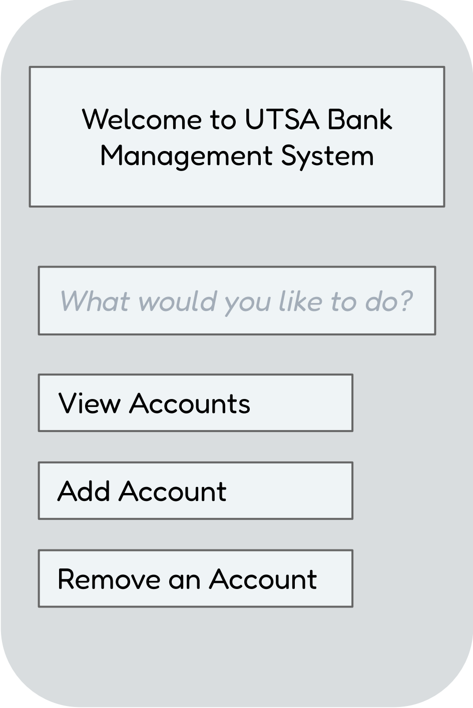
-govID: int
-name: String
-accounts: ArrayList<Account>

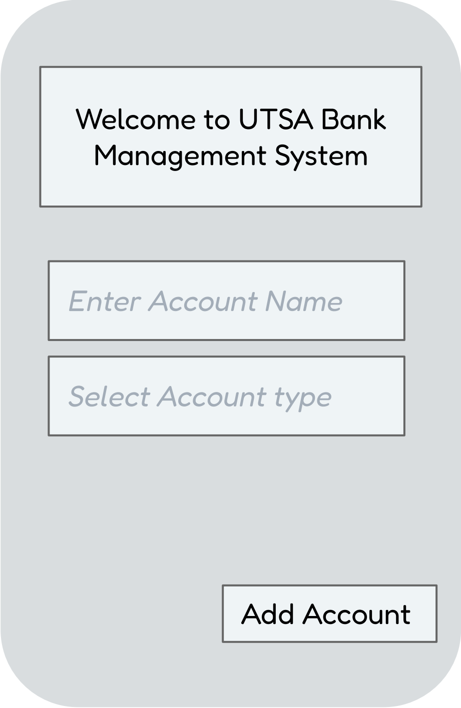
<<constructor>> Bank(govID: String, name:String)
+getGovID(): int
+setGovID(govID: int)
+getName(): String
+setName(name: String)
+getAccounts(): ArrayList<Account>
+setAccount(account: ArrayList<Account>)
+addAccount(account Account)
+loadAccounts(filename : String)
+toString(): String
+removeAccount(name: String)
+getAccount(name: String): Account

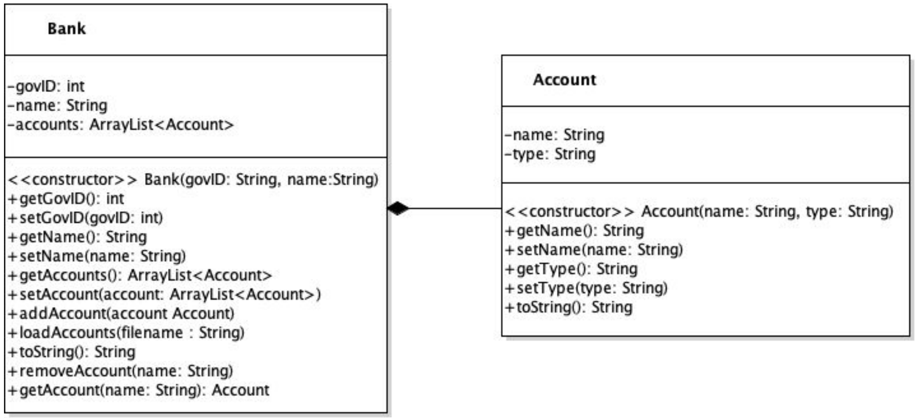
**Account**

-name: String
-type: String

<<constructor>> Account(name: String, type: String)
+getName(): String
+setName(name: String)
+getType(): String
+setType(type: String)
+toString(): String


Welcome to UTSA Bank Management System
What would you like to do?
View Accounts
Add Account
Remove an Account





**Bank**

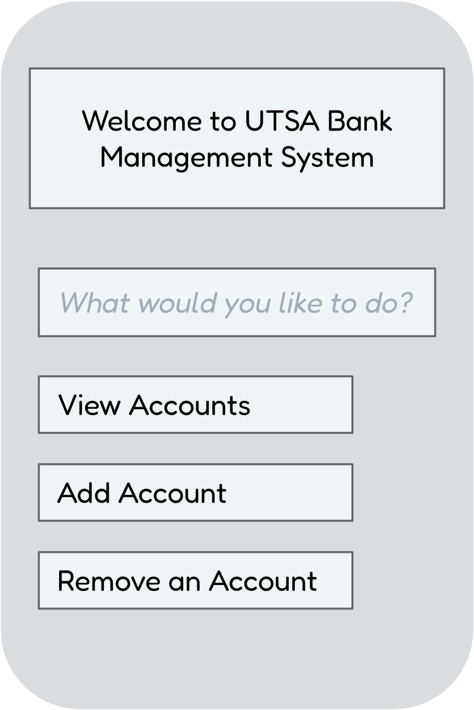
-govID: int
-name: String
-accounts: ArrayList<Account>

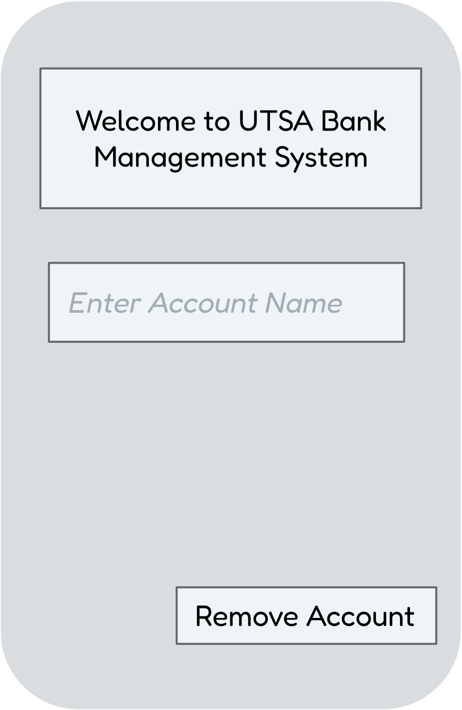
<<constructor>> Bank(govID: String, name:String)
+getGovID(): int
+setGovID(govID: int)
+getName(): String
+setName(name: String)
+getAccounts(): ArrayList<Account>
+setAccount(account: ArrayList<Account>)
+addAccount(account Account)
+loadAccounts(filename : String)
+toString(): String
+removeAccount(name: String)
+getAccount(name: String): Account

**Account**

-name: String
-type: String

<<constructor>> Account(name: String, type: String)
+getName(): String
+setName(name: String)
+getType(): String
+setType(type: String)
+toString(): String


Welcome to UTSA Bank Management System
What would you like to do?
View Accounts
Add Account
Remove an Account

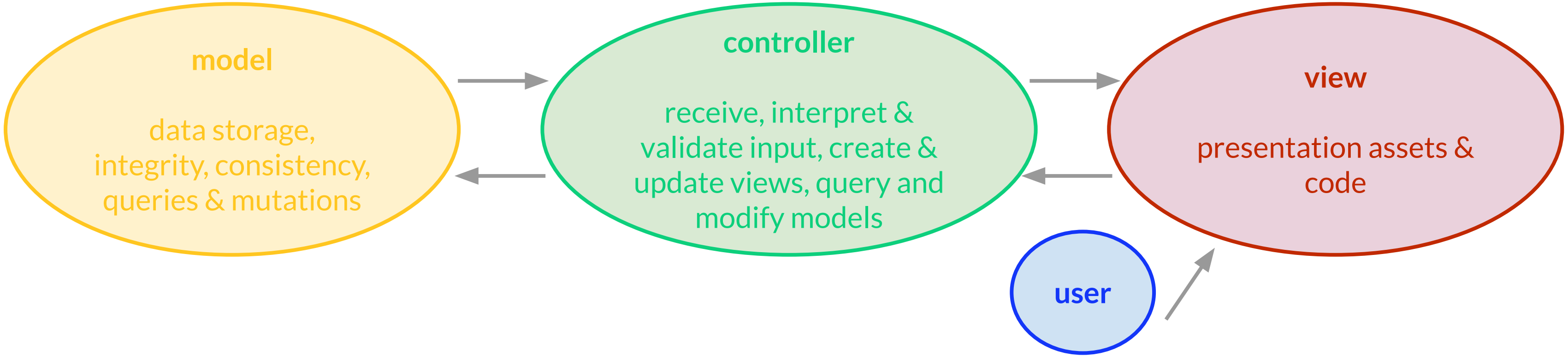


# MODEL-VIEW-CONTROLLER

- An **architectural pattern** captures the design structures of various systems and elements of software so that they can be reused.
- Android applications can be designed around an architecture called **model-view-controller** or **MVC**.
- In MVC, all objects in your application must be a model object, a view object, or a controller object.
- MVC helps you design and understand an application, and makes classes easier to reuse.

# MODEL-VIEW-CONTROLLER

- MVC is an architectural pattern which helps separate
  - the application logic from the user interface
  - the control between the user interface and the application logic



**model**

data storage, integrity, consistency, queries & mutations

**controller**

receive, interpret & validate input, create & update views, query and modify models

**view**

presentation assets & code

**user**

# MODEL

- The **model** represents the **data** and the rules that govern access to and updates of this data, and the **business logic**
- A model often serves as a software **approximation of a real-world process**
- Designed and implemented as one or more classes
  - typically model the things your app is concerned with such as a user, a product, a photo, at television show, … etc.
- Model objects have no knowledge of the UI; its purpose is holding and managing data.

# VIEW

- View objects know how to draw themselves on the **screen**
- The view **renders** the contents of a model
- The view specifies exactly how the model data should be **presented**
- If the model data changes, the view must update its presentation as needed
- Android provides a wealth of configurable view classes. You can also create custom view classes.

# CONTROLLER

- Controller objects tie the View and Model objects together. They contain **application logic**.
- Controllers are designed to **respond** to various events triggered by view objects and to **manage the flow of data** between the model layer and the view layer.
  - The controller translates the user's interactions with the view into actions that the model will perform
- In Android, a controller is typically a subclass of **Activity**, **Fragment**, or **Service**.
- Implemented as one or more classes

# THANK YOU!

## DO YOU HAVE ANY QUESTIONS?

@ hend.alkittawi@utsa.edu

🕒 By Appointment

📍 Online