CSCI 448 – Lab 01B
Friday, January 20, 2023
LAB IS DUE BY **Tuesday, January 31, 2023 11:59 PM**!!

We'll now expand our MonsterLab to be able more flexible about which Monster is being displayed and we'll start displaying multiple Monsters at once.

Step 1 – Model – Create the Monster class

In the prior lab, we were working on the View and presentation side of the app, so we created the presentation package. Now that we're switching to the Model side of the app, we will create another package to hold our model files.

Create another subpackage, this time called `monsterlab.data`. Then create a Kotlin data class titled `Monster`.

Add the following members to the constructor:

- `imageId: Int`
- `nameId: Int`
- `descId: Int`

All should be immutable once the object is created. These values were store the IDs for the resources that correspond to the monster's drawable image, as well as the name & description strings.

And that's it! We'll come back to our Model later on.

Step 2 – View – Display a Monster

Part 2.I – Refactor `MonsterCard`

With our prior `MonsterCard` composable, we hardcoded in the exact monster to display. We'll now make it an actual function and provide it some state data as a parameter. Modify the function to accept a Monster object as a parameter.

Now, instead of hardcoding in the `R` references, we'll set the Image painter to be the object's `imageId`. The three string references will then be the `nameId` twice and finally the `descId`. This should make your inner coder very happy. Our composable is now abstracted and we've encapsulated the state information into our Model object.

Wait. We broke our Preview (and the Activity!). Let's fix our Preview. We need to give it an actual monster to display.

Inside your Preview composable, first create a `youngMike` monster object. Set the arguments as follows:

- `imageId = R.drawable.monsters_university_character_young_mike_icon`
- `nameId = R.string.name_monster_mike`

- `descId = R.string.description_monster_mike`

Then pass this object to the `MonsterCard` composable. Your Preview should be restored to how it was.

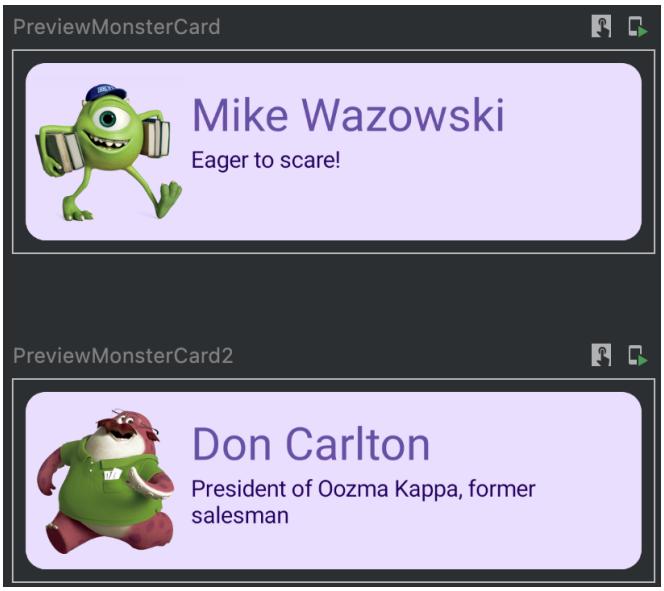
Let's begin to reap the benefits of our flexible, abstracted, encapsulated, extendable (& awesome!) architecture. We'll create a second (!) preview to show the information for another monster. First, add the following strings to your `strings.xml` file:

String Name	String Value
`name_monster_don_carlton`	`Don Carlton`
`description_monster_don_carlton`	`President of Oozma Kappa, former salesman`

We'll now create a second preview for Don:

```
@Preview
@Composable
fun PreviewMonsterCard2() {
  val donCarlton = Monster(
          imageId = R.drawable.monsters_university_don_carlton_icon,
          nameId = R.string.name_monster_don_carlton,
          descId = R.string.description_monster_don_carlton)
  MonsterCard(donCarlton)
}
```

Updating the preview, we should now see two instances of our `MonsterCard`.



Part 2.II – Create a `MonsterList`

Now that we can create multiple monsters, we'll display multiple monsters in a list. In the `presentation` package, create another Kotlin file called `MosnterList`. Create the following composable tree:

- `MonsterList`
  - `Column`
    - `MonsterCard( monster = Monster(`*`mike`*`) )`
    - `MonsterCard( monster = Monster(`*`don`*`) )`

Add a preview for the list and see both our cards at once!

Part 2.III – Put The `MonsterList` Into `MonsterLabScreen`

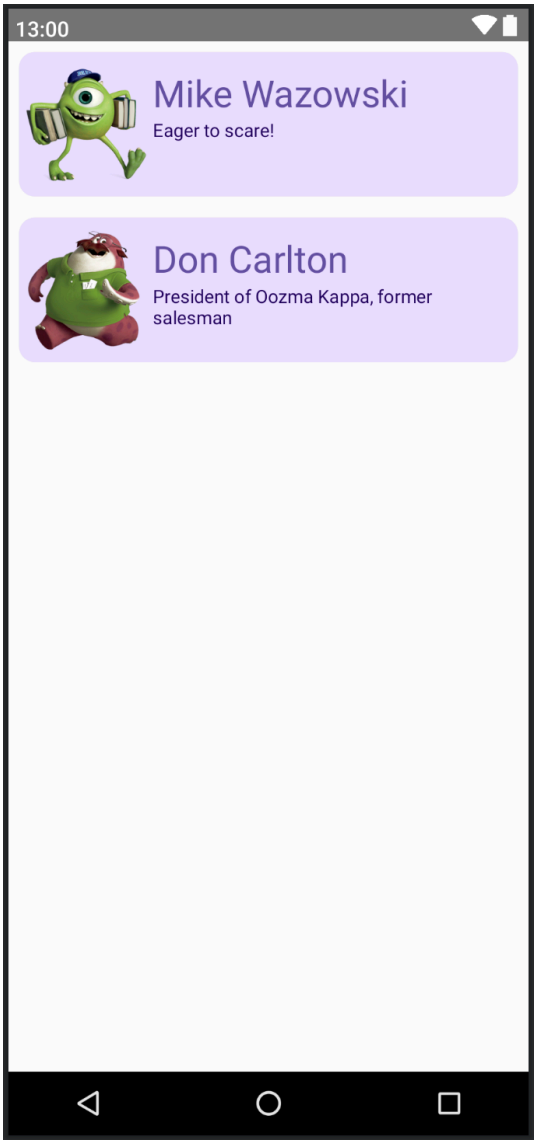
We've updated our View and added a new composable. Over in `MainActivity`, we can now update our `MonsterLabScreen` to display the list of cards instead of the single `MonsterCard`. Verify both the preview and deploying the app shows our two monsters.

Take notice that our cards no longer fill the entire screen and are sized as desired. Previously, our composable tree consisted of

- `MonsterLabTheme`
  - `Surface( modifier = fillMaxSize() )`
    - `MonsterLabScreen`
      - `MonsterCard`
        - `Card`
          - `...`

The first emitted UI child inside the Surface was the `Card`, which was sized to fill the whole screen. We've now modified the composable tree to be

- MonsterLabTheme
    - Surface(modifier = fillMaxSize())
        - MonsterLabScreen
            - MonsterList
                - Column
                    - MonsterCard
                        - Card
                            - ...
                    - MonsterCard
                        - Card
                            - ...



Now the first emitted UI child is the Column. The Column now has its size set to fill the whole screen and our individual Cards our sized as necessary. This is an instance where you can use the Developer Options > Show layout bounds to see how each item is sized.

Step 3 – Deploy Your App

Once you have the screen composable set, deploy your app to your device. While the colors may differ (based on light/dark theme), you should now have Mike Wazowski and Don Carlton on your device!

When Lab01B is complete, continue on to Lab01C to add even more monsters to your monster lab.

LAB IS DUE BY **Tuesday, January 31, 2023 11:59 PM**!!