HW3:

Fragment Keyboard

Martin Mudenda Bbela

2582912

Overview

Develop a notepad app with custom keypad:

- Top fragment: display the note entered
- Bottom fragment: display several rows of letters, numbers, and symbols (each as a button) for text input; when a user pushes a button, the corresponding input is appended in the top fragment

Method

I used the updated pattern to create this a listener between the two fragments in the main activity. In this case I create a Button Press View Model that will host a string as truth. This will be used to communicate between the two Fragments.

```
public class ButtonPressViewModel extends ViewModel {
    private final MutableLiveData<String> string = new MutableLiveData<~>();
    public LiveData<String> getString() { return string; }

    public LiveData<String> getStringData() { return string; };

    public void setString(String newString){

        string.setValue(newString);

    }
}
```

The Main activity will be completely bare. The layout will hold two Fragments that I will Statically add two fragments in the layout itself

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView"
        android:name="com.example.hw3fragmentkeyboard.KeysPressedFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView2"
        android:name="com.example.hw3fragmentkeyboard.KeyboardFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="3" />
</LinearLayout>
```

There will be two Fragments. The First we will discuss is the KeyboardFragment that will have a string of Truth. This will be appended to as keys are pressed and will also be deleted from with the backspace button. The ViewModel is taken from the ViewModelProvider so it is a singleton. This will be given the new String which will be sent to the other Fragment.

```java
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    btn1 = view.findViewById(R.id.button1);
    btn2 =  view.findViewById(R.id.button2);
    btn3 = view.findViewById(R.id.button3);
    btnbackSpace = view.findViewById(R.id.button4);

    bpViewModel = new ViewModelProvider(requireActivity()).get(ButtonPressViewModel.class);
    keyboardString = "";


    btn1.setText("\uD83D\uDE0A"); //😊
    btn2.setText("\uD83D\uDE2F"); //😯
    btn3.setText("\uD83D\uDC4F\uD83C\uDFFF");// 👏 that supposed to be a black hands slapping and not a weird smudge
    btnbackSpace.setText("BackSpace");

    btn1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { updateKeys("\uD83D\uDE0A"); }
    });
    btn2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { updateKeys("\uD83D\uDE2F"); }
    });
    btn3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) { updateKeys("\uD83D\uDC4F\uD83C\uDFFF"); }
    });
    btnbackSpace.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(getActivity().getBaseContext(),  text: "backspace", Toast.LENGTH_LONG).show();
            backspace();
        }
    });
}
```

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_keyboard, container,  attachToRoot: false);
}

void updateKeys(String key){
    keyboardString = keyboardString+key;
    bpViewModel.setString(keyboardString);
}

void backspace(){
    if(keyboardString.length() >1)
        keyboardString = keyboardString.substring(0, keyboardString.length()-1);
    bpViewModel.setString(keyboardString);

};
```

The next fragment, the KeyPressedFragment, will take the ButtonPressViewModel and listen for when the string has been updated and use it to update the the TextView

```java
public static KeysPressedFragment newInstance() { return new KeysPressedFragment(); }
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    tvString =  view.findViewById(R.id.textViewString);

    buttonPressViewModel =  new ViewModelProvider(requireActivity()).get(ButtonPressViewModel.class);

    buttonPressViewModel.getStringData().observe(getViewLifecycleOwner(), new Observer<String>() {
        @Override
        public void onChanged(String s) {
            Toast.makeText(getActivity().getBaseContext(), s, Toast.LENGTH_LONG).show();
            tvString.setText(s);
        }
    });

}

@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.keys_pressed_fragment, container,  attachToRoot: false);

    return view;
}
```

Result

Users are able to type out emoji strings in our minimalist keyboard as well as delete from them as needed