Jonathan Benson
CS 349 - Java Programming with Applications
Homework 1
24 February 2022

Introduction

For Homework 1, I designed an app that lets users learn about different types of bicycles. It also showcases the differences in implementation between a ListView and RecyclerView, which are both implemented as the fragments ListViewFragment and RecycleViewFragment respectively.

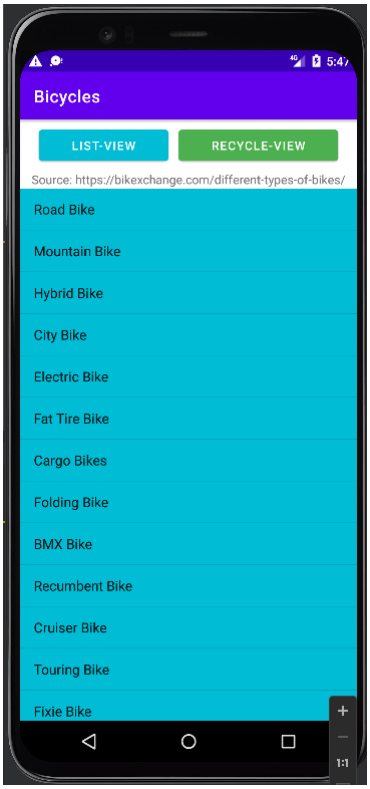
Instructions on how to run the project are at the end of this report.

Github Repository: https://github.com/jonathanbenson/Bicycle-Info-Android-App

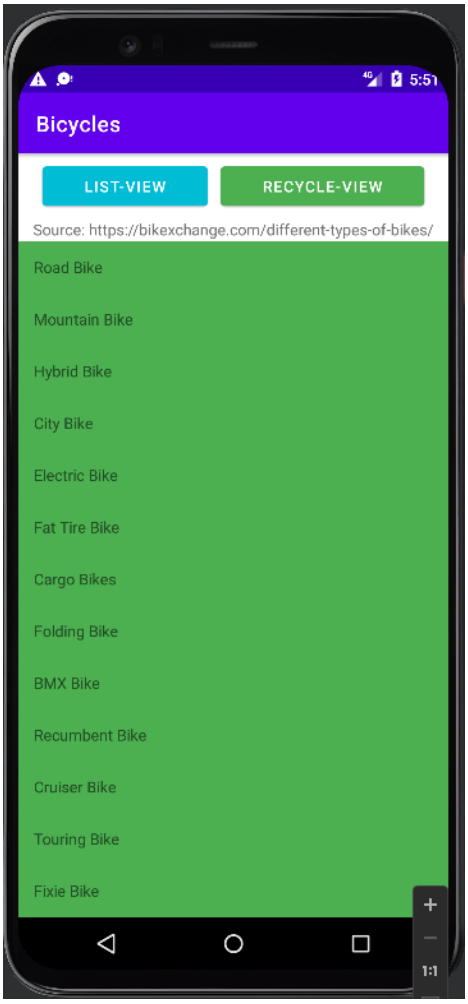
Design

When the app starts up in the MainActivity, the user is met with the following components:

1. Two navigation buttons that allow the user to switch between viewing the bicycles in a ListView and a RecyclerView.
2. A line of text pointing to the source of the information about all of the bicycles in the project.
3. A fragment container that hosts all the bicycles either in the form of a ListView or a RecyclerView.



It's important to note that the color of the button corresponds with the type of container that holds the bicycles. When the application starts for the first time, it loads the bicycles in ListView format first, as indicated in blue. If the user clicks the Recycle-View button, it will reload the fragment container with a RecycleViewFragment and the bicycles in a RecyclerView. The bicycle items in this instance would emit a green color:



When a user clicks on a bicycle item in either the ListView or RecycleView, it will start a new activity that has three components:

1. The title of the bicycle (same as the one that was clicked).
2. The description of the bicycle.
3. An image of the bicycle.
4. A back button that allows the user to return to the main activity.

Demo: if a user selects “Road Bike” from the ListView OR the RecycleView, the following BicycleInfoActivity will load into view:

Upon pressing the “GO BACK” button, the app will navigate back to the MainActivity. If the user selects an item from a ListView before the BikeInfoActivity, the MainActivity will load back into view the ListViewFragment. On the other hand, if the user selected an item from the RecyclerView, the MainActivity will load back the RecycleViewFragment.

Implementation

The relevant functionality starts in the MainActivity.onCreate method. It starts by assigning onClick event listeners to the ListView and RecycleView buttons. At the end, it handles the logic in determining whether to load a ListViewFragment or RecycleViewFragment into its fragment container view. It will by default load a ListViewFragment on startup.

MainActivity.onCreate Method

```
Button listViewButton = this.findViewById(R.id.listViewButton);

MainActivity currentActivity = this;

// Set the list view button's event listener
// When clicked, it will load the FragmentContainerView with a ListViewFragment
listViewButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) { currentActivity.loadFragment(new ListViewFragment()); }
});


Button recycleViewButton = this.findViewById(R.id.recycleViewButton);

// Set the recycle view button's event listener
// When clicked, it will load the FragmentContainerView with a RecycleViewFragment
recycleViewButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        currentActivity.loadFragment(new RecycleViewFragment());
    }
});

Bundle bundle = this.getIntent().getExtras();

// Initialize the FragmentContainerView

// If the app has just started up or it is returning from BikeInfoActivity having previously loaded a ListViewFragment
// then load the ListViewFragment back into the FragnmentContainerView
if (bundle == null || bundle.getString( key: "mainActivityState") == "listView")
    this.loadFragment(new ListViewFragment());

// Else if the app is returning from BikeInfoActivity having previously loaded a RecycleViewFragment
// then load the RecycleViewFragment back into the FragmentContainerView
else if (bundle.getString( key: "mainActivityState") == "recycleView")
    this.loadFragment(new RecycleViewFragment());
```

The ListViewFragment loads the bicycle items into view via its onViewCreated method. It first retrieves the list of bicycles from an array of strings in the standard strings.xml file. It then creates an adapter for its ListView to facilitate the adding of items to the list. Finally, it adds an

onClick event listener for each bicycle item in the list. This onClick function will start a new BikeInfoActivity that will display more information about the bicycle.

ListViewFragment.onViewCreated Method

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    // Populates the list view with the names of the bicycles

    // Get the list of bicycles from a string array in res/values/string.xml
    this.items = getResources().getStringArray(R.array.bicycle_names);

    // Create and set an adapter for the list view to facilitate adding items
    this.adapter = new ArrayAdapter<String>(this.getContext(), android.R.layout.simple_list_item_1, this.items);
    this.listView = (ListView) this.getView().findViewById(R.id.listView1);
    this.listView.setAdapter(this.adapter);

    ListViewFragment currentFragment = this;

    // Create an onClick event listener for each item in the list view
    // It will start a new BikeInfoActivity to display the relevant information about the bike that was clicked on
    this.listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {
            currentFragment.loadBikeInfoActivity(position);
        }
    });

}
```

The implementation of the RecycleViewFragment is similar to that of the ListView. Its parent RecycleViewFragment starts in the onViewCreated method just like the ListViewFragment. It first retrieves the list of bicycles from an array of strings in the standard strings.xml file. Then it initializes the RecyclerView by creating and setting both its adapter and layout manager.

RecycleViewFragment.onViewCreated Method

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState) {

    // Retrieve the names of the bicycle types
    this.items = getResources().getStringArray(R.array.bicycle_names);

    // Get the recycler view that will hold the names of the bicycles
    this.recycleView = view.findViewById(R.id.recycleView1);

    // Initialize and set the adapter for the recycler view
    this.adapter = new RecycleViewAdapter(this.items, clickListener: this);
    this.recycleView.setAdapter(this.adapter);

    // Initialize and set the layout manager to manage the recycler view
    this.layoutManager = new LinearLayoutManager(this.getActivity());
    this.recycleView.setLayoutManager(new LinearLayoutManager(this.getContext()));


}
```

The recycler view's adapter is implemented in its own specialized RecycleViewAdapter class. This adapter contains inside it a ViewHolder class that handles rendering the bicycle items in the recycler view and the onClick events of its bicycle items.

RecycleViewAdapter.ViewHolder Class

```
public static class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    // The ViewHolder class holds the view that represents each bicycle in the recycler view
    // It also handles the onClick event

    private final TextView textView;
    private final RecycleViewClickListener clickListener;

    public ViewHolder(View view, RecycleViewClickListener clickListener) {
        super(view);

        // Initialize the onClick event listener which is sourced from the RecycleViewFragment
        this.clickListener = clickListener;
        view.setOnClickListener(this);

        // Initialize the view that represents each item in the list
        this.textView = (TextView)view.findViewById(R.id.textView);

    }

    @Override
    public void onClick(View view) {
        // Calls the onClick handler from the RecycleViewFragment
        this.clickListener.recycleViewOnClick(view, this.getLayoutPosition());
    }

    // Return the TextView representing each item in the list
    public TextView getTextView() { return this.textView; }
}
```

When bicycle items are selected either from the ListViewFragment or the RecycleViewFragment, their title, description, and image is loaded into a new BikeInfoActivity. This special activity is initialized in its onCreate method. First, it figures out which bicycle was selected from the ListViewFragment or RecycleViewFragment. It determines this by retrieving the index of the item and then referencing the bicycle title and description string arrays in the standard string.xml project file. Its title and description are populated with the bicycle's title and description respectively. Then it loads the bicycle's image into an ImageView component by matching the bicycle title to the file name. The bicycle images are located in the res/drawable folder as static image files. Finally, it initializes the onClick event listener for its "GO BACK" button to go back to the MainActivity when it is pressed.

BikeInfoActivity.onCreate Method

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bike_info);

    // Figure out which bicycle was clicked by retrieving its index from the intent
    Intent intent = this.getIntent();
    int bikeIndex = intent.getIntExtra( name: "bikeIndex",  defaultValue: 0);

    // Get the title and description of the bike from the bike index
    // The bike tiles and descriptions are stored as string arrays in the string.xml file
    String bikeTitle = getResources().getStringArray(R.array.bicycle_names)[bikeIndex];
    String bikeDescription = getResources().getStringArray(R.array.bicycle_descriptions)[bikeIndex];

    // Retrieve the title of the bike by indexing the array of bike titles
    TextView bikeTitleView = this.findViewById(R.id.bikeTitle);
    bikeTitleView.setText(bikeTitle);

    // Get the description of the bike by indexing the array of bike descriptions
    TextView bikeDescriptionView = this.findViewById(R.id.bikeDescription);
    bikeDescriptionView.setText(bikeDescription);

    // Get the image of the bike and load it into the ImageView
    this.populateImage(bikeTitle);


    // Set the "go back" button's onClick listener to BikeInfoActivity.goBack()
    // This will cause the MainActivity to start back up when the go back button is clicked
    Button backButton = this.findViewById(R.id.backButton);
    BikeInfoActivity currentActivity = this;

    backButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            currentActivity.goBack();
        }
    });

}
```

How to Run

1. Extract the submission folder's contents into a directory you will remember.
2. Open Android studio.
3. Select "File->Open…" and select the "Bicycle-Info-Android-App" folder contained in the extracted submission folder.
4. Make sure you have the latest version of Android Studio
    a. Select "Help->Check for updates…" and install the necessary updates.
5. Make sure you have a virtual device installed so you can run the app. The following link contains instructions on how to install one.
    a. https://developer.android.com/studio/run/managing-avds
6. Click the green, right-facing triangle at the top to run the app.