Android Application Implementing ListView Using Linked List

Data Structures
CSD-223

Submitted by:

Kashish Srivastava (185014)
Dipesh Kumar (185015)
Akash Rana (185034)

CSE (4 Year) : 4^{th} Semester

Under the guidance of

Dr. Nitin Gupta
Assistant Professor, CSE Department
National Institute of Technology, Hamirpur

Department of Computer Science and Engineering
National Institute of Technology, Hamirpur

Abstract

Our goal was to develop a android application Implementing ListView using datastructure such as LinkedList. This Android application is basically coded in Kotlin. This Android application will provide our student and other people with the brief information of each and every department (i.e about different branches of engineering that are available in our institute) to have the better idea of every department of our institute and its courses available in that respective department. By developing this Android application, our basic idea was to provide the clear idea of how linked list, queue, stack and sets can be used to develop a well working an application.

Contents

Introduction

The era of mobile technology opens the window to the android app development. The websites are vanishing and the mobile phones are emerging. It‘s high time to change-form conventional form websites to apps, which has become the part of our daily routine. We are here introducing self made android application which would be a miniature part of our college website. It not only works as a website but also as a small college management software. This android application will provide our student and other managing staffs with the proper information of each and every department and of about its courses and placement. This android application is also a mobile version of the part of our institute official website.

This android application also uses different data structures in the development of the same. The data structures used in its development are linked list, queue, arrays, hash table etc. These all the data structure are pre provided by the android development for the development of such application.

List view is used in this application to groups several items and display them in vertical scrollable list which is a list of department of our institute and similarly linked list is used in this application to link last element with the recent one and also it allocates memory dynamically and hence these data structures are used in our application.

This project is basically designed to provide the better idea of how data structures i.e linked list,list view,hash table and arrays can be used to develop real life android application for the betterment of the coming youth.

Requirements and Installation

The software and programming languages that are required to build an android application and we should also have prior knowledge about the following:-

Kotlin 1.32

Kotlin is a statically typed programming language for Java Virtual Machine (JVM) and JavaScript. Described as a general-purpose language, Kotlin introduces functional features to support Java interoperability. The Kotlin project was born out of the aspiration for heightened productivity. The goal was to improve the coding experience in a way that was both practical and effective.

Android Studio 3.6.1

Android Studio is an IDE, which is essentially an interface where you can enter your code (primarily Java or Kotlin) and access all the different tools necessary for development. Android Studio allows you to access libraries and APIs from the Android software development kit(SDK‘s).

Android Virtual Device

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. The AVD Manager is an interface you can launch from Android Studio that helps you create and manage AVDs.

Components used

4.1 ListView (View)

Android ListView is a view which contains the group of items and displays in a scrollable list. ListView is implemented by importing android.widget.ListView class. ListView is a default scrollable which does not use other scroll view.

ListView uses Adapter classes which add the content from data source (such as string array, array, database etc) to ListView. Adapter bridges data between an AdapterViews and other Views (ListView, ScrollView etc).

```
public interface List
implements Collection<E>

java.util.List<E>

Known indirect subclasses
AbstractList<E>, AbstractSequentialList<E>, ArrayList<E>, CopyOnWriteArrayList<E>, LinkedList<E>, Stack<E>, Vector<E>
```

4.2 Linked List (Data Structure)

Doubly-linked list implementation of the List and Deque interfaces. Implements all optional list operations, and permits all elements (including null).

All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index.

```
public class LinkedList
extends AbstractSequentialList<E> implements List<E>, Deque<E>, Cloneable, Serializable

java.lang.Object
  ↳ java.util.AbstractCollection<E>
      ↳ java.util.AbstractList<E>
          ↳ java.util.AbstractSequentialList<E>
              ↳ java.util.LinkedList<E>
```

ListView Layout Design

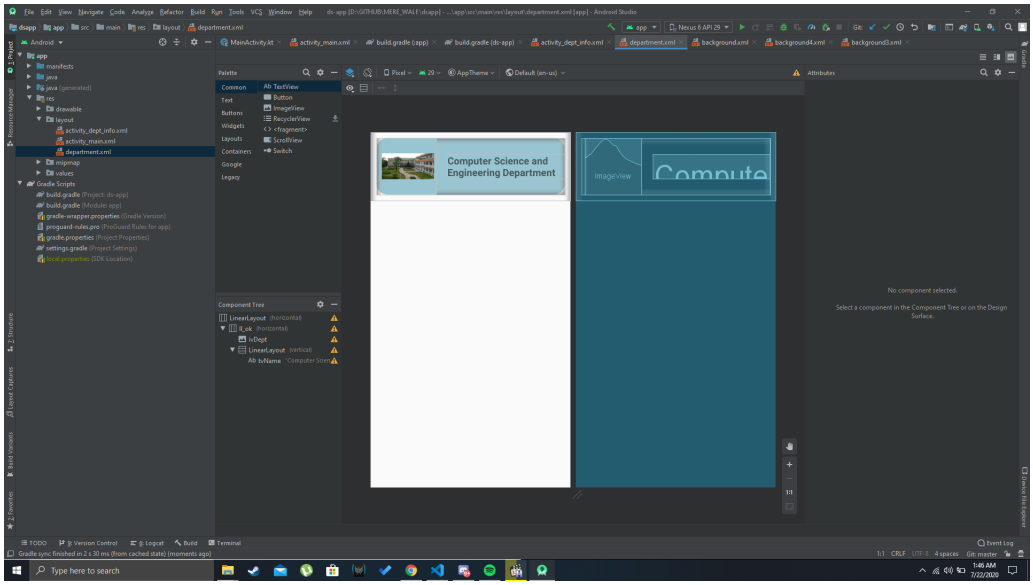
The display of elements in a list is a very common pattern in mobile applications. The user sees a list of items and can scroll through them. Typically the user interacts with the list via the toolbar, for example, via a button which refreshes the list. Individual list items can be selected. This selection can update the toolbar or can trigger a detailed screen for the selection.

Every line in the widget displaying the data consists of a layout which can be as complex as you want. A typical line in a list has an image on the left side and two text lines in the middle as we have used here. With Department picture on the left and Name of Department centered on the right.

Listing 5.1: department.xml

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="5pt">

    <LinearLayout
        android:id="@+id/ll_ok"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0"
        android:background="@drawable/background"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/ivDept"
            android:layout_width="50pt"
            android:layout_height="50pt"
            android:layout_marginLeft="3pt"
            android:layout_weight="0"
            android:padding="2pt"
            app:srcCompat="@drawable/cse" />

        <LinearLayout
            android:layout_width="match_parent"
```

```
            android:layout_height="match_parent"
            android:layout_gravity="center|center_horizontal|center_vertical"
            android:layout_weight="0"
            android:gravity="center|center_horizontal|center_vertical"
            android:orientation="vertical"
            android:paddingLeft="10pt">

            <TextView
                android:id="@+id/tvName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Computer Science and Engineering Department"
                android:textSize="20sp"
                android:textStyle="bold" />

        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```



We have to design the layout only once as this layout will be used for every item on the ListView. This the crucial step as without designing a layout we wont be able to display the view anywhere , say activity_main.xml layout or any other layout for that matter

Load ListView with data using LinkedList

Now, that the Layout for ListView has been created ,it needs to be populated with data. In order to do this we use LinkedList. First ,LinkedList is loaded with all the data that needs to be ported to ListView like we have below in the LinkedList named ListOfDept.

Listing 6.1: MainActivity.kt

```
import ...
class MainActivity : AppCompatActivity() {

  var ListOfDept = LinkedList<Department>()
  var adapter:DeptAdapter?=null
  override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      //load departments

      ListOfDept.add(Department("Computer Science and Engineering",
      getString(R.string.CSE_des),R.drawable.cse))

      ListOfDept.add(Department("Electronic and Comunication Engineering",
      getString(R.string.ECE_des)   ,R.drawable.ece))

      ListOfDept.add(Department("Civil Engineering",
      getString(R.string.CE_des),R.drawable.ce))


      ListOfDept.add(Department("Mechanical Engineering",
      getString(R.string.ME_des),R.drawable.me))


      ListOfDept.add(Department("Chemical Engineering",
      getString(R.string.CE_des),R.drawable.che))

      ListOfDept.add(Department("Electrical Engineering",
      getString(R.string.EE_des),R.drawable.ee))

      //initialise adapter with array list
      adapter = DeptAdapter(this,ListOfDept)
```

```
        tvListView.adapter = adapter

    }
```

After that we need to declare and implement our custom Adapter .An adapter manages the data model and adapts it to the individual entries in the widget. An adapter extends the BaseAdapter class. The adapter would inflate the layout for each row in its getView() method and assign the data to the individual views in the row. The adapter is assigned to the ListView via the setAdapter method on the ListView object.

Listing 6.2: Defining adapter in MainActivity.kt

```
class DeptAdapter:BaseAdapter{

        var ListOfDept = LinkedList<Department>()
        var context:Context?=null
        constructor(context: Context, ListOfDept: LinkedList<Department>):supe
            this.ListOfDept = ListOfDept
            this.context = context
        }

        override fun getView(position: Int, convertView: View?, parent: ViewGr
            val Dept = ListOfDept[position]
            var inflater = context!!.getSystemService(Context.LAYOUT_INFLATER_
            var myView = inflater.inflate(R.layout.department,null)
            myView.tvName.text = Dept.name
            myView.ivDept.setImageResource(Dept.image!!)
            when(position%4){
                1-> myView.ll_ok.setBackground(this.context!!.getDrawable(R.dr
                2-> myView.ll_ok.setBackground(this.context!!.getDrawable(R.dr
                3-> myView.ll_ok.setBackground(this.context!!.getDrawable(R.dr
                0-> myView.ll_ok.setBackground(this.context!!.getDrawable(R.dr

            }
            myView.setOnClickListener{
                val intent = Intent(context,DeptInfo::class.java)
                intent.putExtra("name",Dept.name!!)
                intent.putExtra("des",Dept.des!!)
                intent.putExtra("image",Dept.image!!)
                context!!.startActivity(intent)
            }
            return myView
        }
```

```
    override fun getItem(position: Int): Any {
        return ListOfDept[position]
    }

    override fun getItemId(position: Int): Long {
        return position.toLong()
    }

    override fun getCount(): Int {
        return ListOfDept.size
    }

}
```

Main 2 functions in our adapter are getView() and getCount(). getCount() returns the size of LinkedList which will be used in getView() func with ultimately return us view. We have customized getView() function to return every element in list with different background. And a setOnClickListener has been setup to take us to another activity which gives detail about that particular department.

ListView events

After populating ListView with LinkedList data using our custom Adapter , new activity needs to be created in order so that on clicking an item in ListView we are taken to Details about that Department.

Listing 7.1: setOnClickListener example

```
myView.setOnClickListener{
    val intent = Intent(context,DeptInfo::class.java)
    intent.putExtra("name",Dept.name!!)
    intent.putExtra("des",Dept.des!!)
    intent.putExtra("image",Dept.image!!)
    context!!.startActivity(intent)
}
```

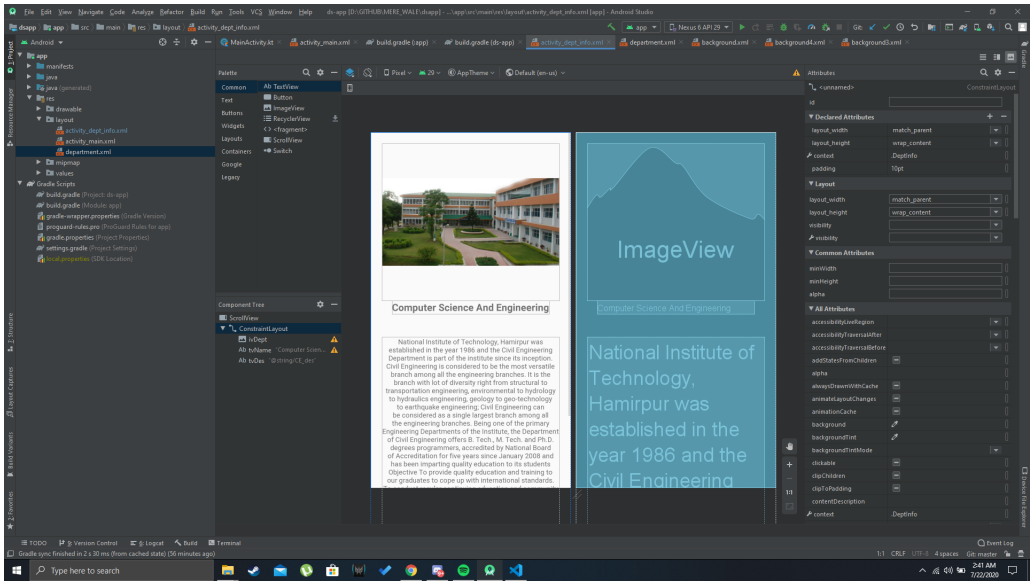
Here we have and example of such event which is triggered on clicking on an element of ListView.

Other than that we need to create a new activity and a layout. The new layout we create must contain the picture of department , name and description of the Department. We use Constraint we rather than using LinearLayout that we used to wrap the listview. Our new Layout looks something like this.

Listing 7.2: activity_dept_info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView 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">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10pt"
        tools:context=".DeptInfo">

        <ImageView
            android:id="@+id/ivDept"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.473"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/cse" />

        <TextView
            android:id="@+id/tvName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Computer Science And Engineering"
            android:textSize="20sp"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/ivDept"
            app:layout_constraintVertical_bias="0.0" />

        <TextView
            android:id="@+id/tvDes"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="48dp"
            android:gravity="center|center_horizontal|center_vertical"
            android:text="@string/CE_des"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.652"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/tvName" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
```



Last thing required to implement is a new activity that is linked to the new layout we just created. In the new activity we will provide the Name,description and the image of department which will be given to the Layout.

Listing 7.3: DeptInfo.kt

```
import ...
class DeptInfo : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dept_info)
        val bundle: Bundle? = intent.extras
        val name = bundle?.getString("name")
        val des = bundle?.getString("des")
        val image = bundle?.getInt("image")
        ivDept.setImageResource(image!!)
        tvName.text = name
        tvDes.text = des
    }
}
```

Output

All the Components required for the app are successfully implemented . Now we can just compile the whole code and used an AVD(Android Virtual Device) to view the finished application . We can also build the app as apk and use on an actual android device. Built app will look something like:

Finished Android Application

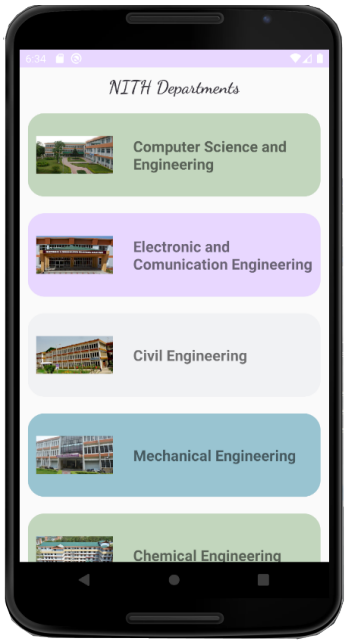


Figure 8.1:
MainActivity.kt

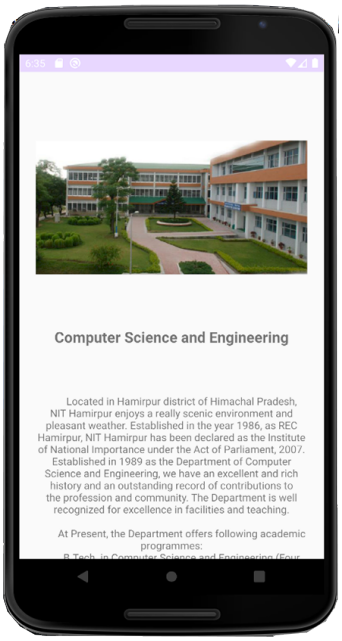


Figure 8.2: DeptInfo.kt

Conclusion

This Project was a great opportunity for us to discover new fields and ways of working. We have successfully implemented ListView with LinkedList in an android application. LinkedList provides us with efficient form of storing data which needs to be retrieved sequentially and ListView provides us with the ability to dynamically create lists with custom layouts and with functionality of incepting tasks(such as opening a new activity) on clicking on a list item.

All these Components combined provide us with an effective android application that displays the description of various departments of Engineering in National Institute of Technology Hamirpur.

References

The source code and other files related to this project can be found at
`https://github.com/cannibalcheeseburger/ds-app.git`

We used following references while working with this project:-

- Natarajan Raman, Eunice Adutwumwaa Obugyei, Learning Kotlin by Building Android Applications: Explore the Fundamentals of Kotlin by Building Real-world Android Applications.
- Samuel Urbanowicz: Kotlin Standard Library Cookbook.
- `https://youtu.be/PJ3RdfJ4Np8`
- `http://developer.android.com/reference/packages.html`
- `http://developer.android.com/guide/topics/ui/index.html`