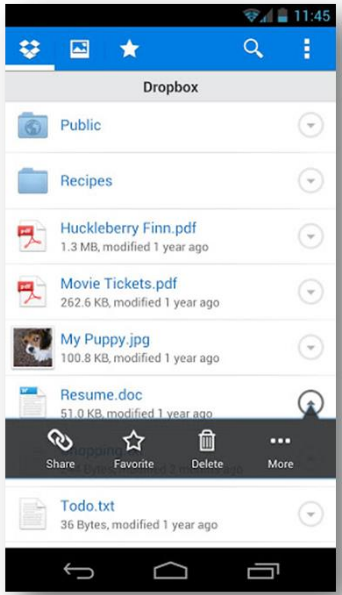
What are we going to learn?

- How to create you ListView in Android application
- How to create a custom ListView
- Fragment in Android Development
- Communication between different fragments
- Fragment transaction

ListView in Android

- ListView is a view group that displays a list of scrollable items
- Adapter: Pulls content from a source and converts each item result into a view


11:45

Dropbox

Public

Recipes

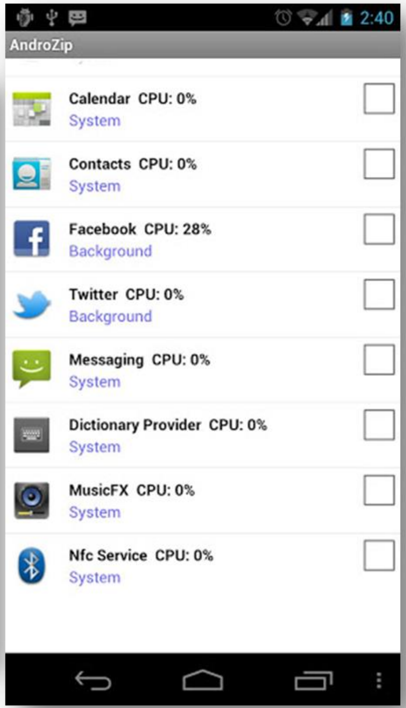
Huckleberry Finn.pdf
1.3 MB, modified 1 year ago

Movie Tickets.pdf
262.6 KB, modified 1 year ago

My Puppy.jpg
100.8 KB, modified 1 year ago

Resume.doc
51.0 KB, modified 1 year ago

Todo.txt
36 Bytes, modified 1 year ago


2:40

AndroZip

Calendar CPU: 0%
System

Contacts CPU: 0%
System

Facebook CPU: 28%
Background

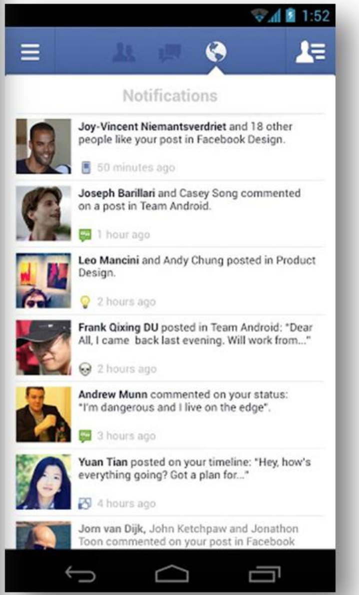
Twitter CPU: 0%
Background

Messaging CPU: 0%
System

Dictionary Provider CPU: 0%
System

MusicFX CPU: 0%
System

Nfc Service CPU: 0%
System


1:52

Notifications

Joy-Vincent Niemantsverdriet and 18 other people like your post in Facebook Design.
50 minutes ago

Joseph Barillari and Casey Song commented on a post in Team Android.
1 hour ago

Leo Mancini and Andy Chung posted in Product Design.
2 hours ago

Frank Qixing DU posted in Team Android: "Dear All, I came back last evening. Will work from..."
2 hours ago

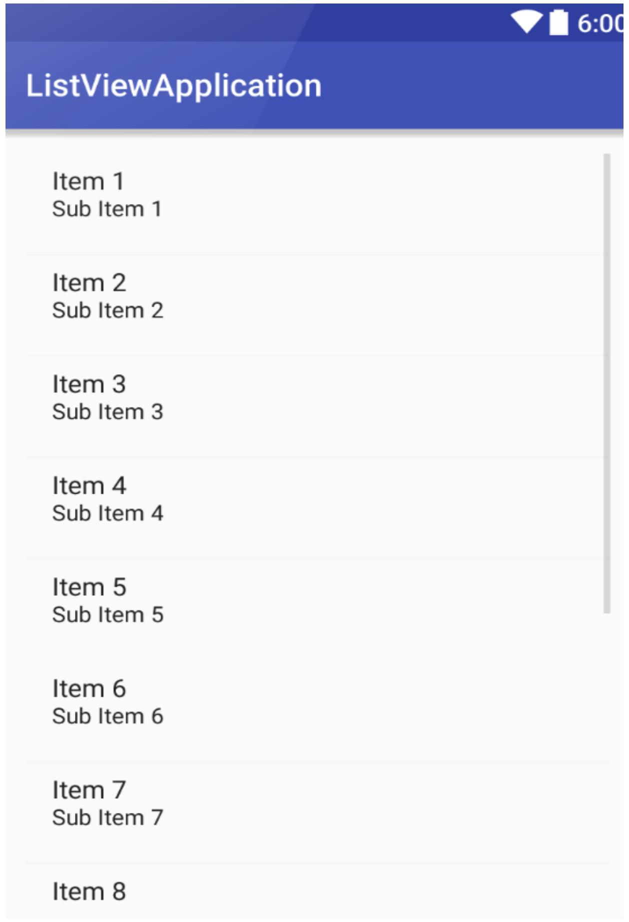
Andrew Munn commented on your status: "I'm dangerous and I live on the edge".
3 hours ago

Yuan Tian posted on your timeline: "Hey, how's everything going? Got a plan for..."
4 hours ago

Jorn van Dijk, John Ketchpaw and Jonathon Toon commented on your post in Facebook

Step 1: Create a ListView XML Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.xunhu.listviewapplication.MainActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/myListView"
        >
    </ListView>

</RelativeLayout>
```



6:00

ListViewApplication

Item 1
Sub Item 1

Item 2
Sub Item 2

Item 3
Sub Item 3

Item 4
Sub Item 4

Item 5
Sub Item 5

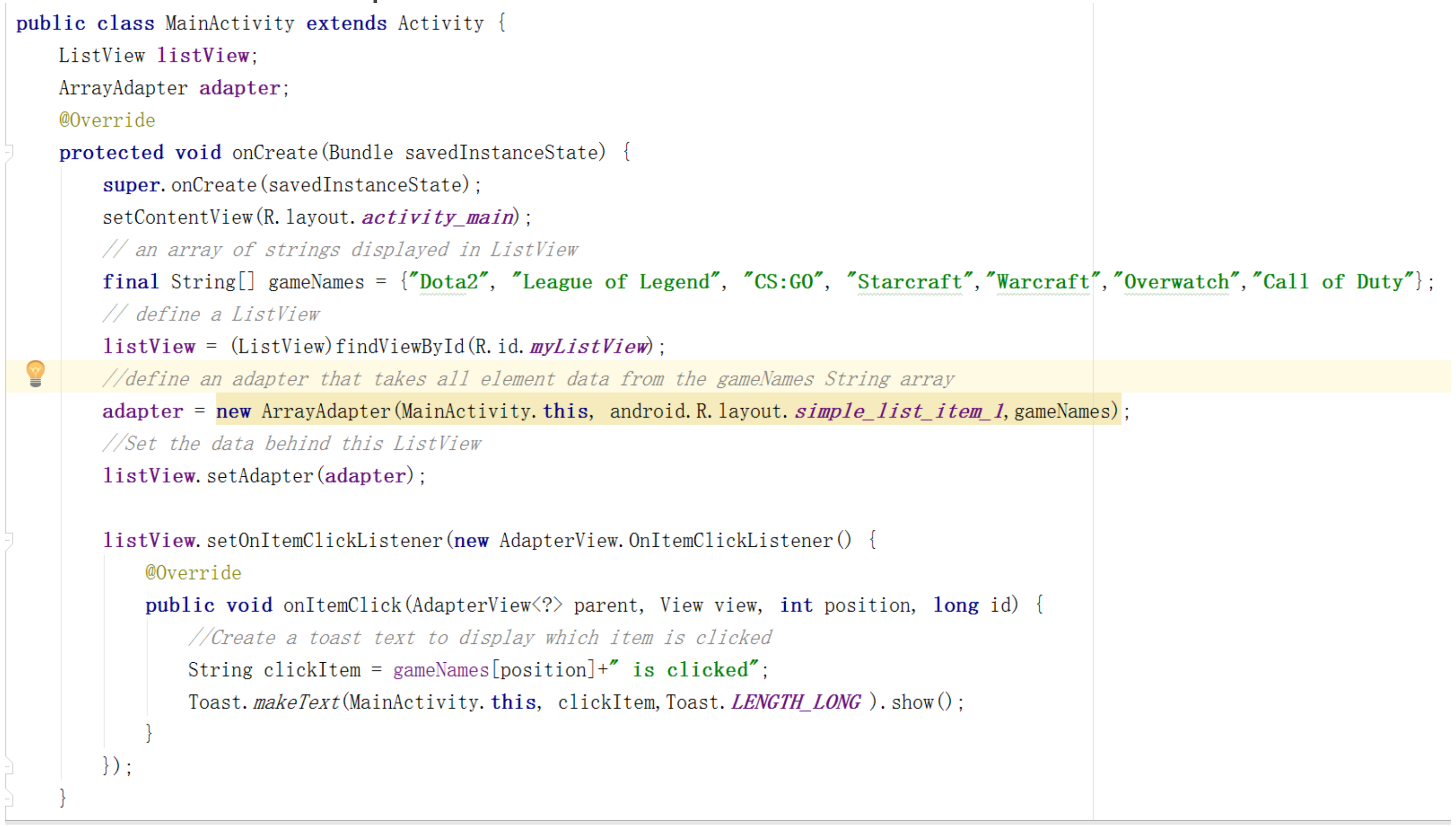
Item 6
Sub Item 6

Item 7
Sub Item 7

Item 8

Step 2: ListView and Adapter in Java File

- Create a ListView and an adapter in Java File



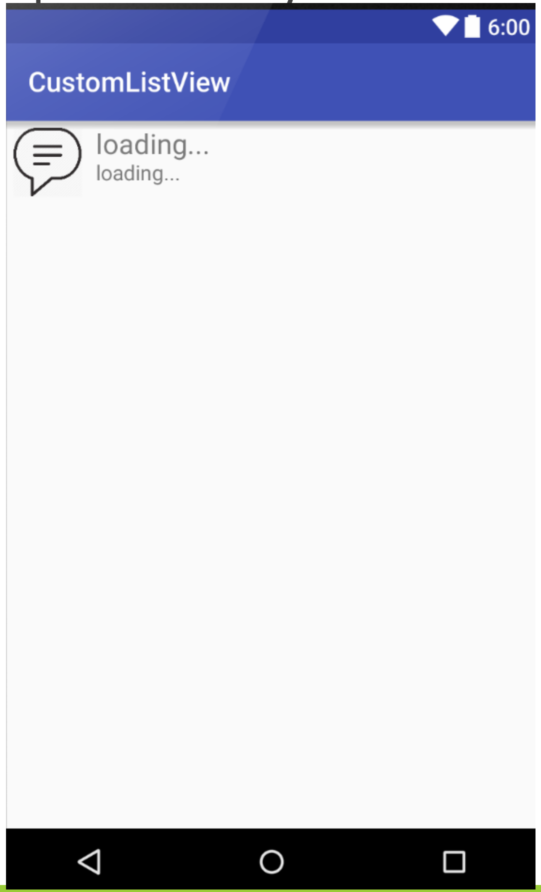
```java
public class MainActivity extends Activity {
    ListView listView;
    ArrayAdapter adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // an array of strings displayed in ListView
        final String[] gameNames = {"Dota2", "League of Legend", "CS:GO", "Starcraft","Warcraft","Overwatch","Call of Duty"};
        // define a ListView
        listView = (ListView)findViewById(R.id.myListView);
        //define an adapter that takes all element data from the gameNames String array
        adapter = new ArrayAdapter(MainActivity.this, android.R.layout.simple_list_item_1,gameNames);
        //Set the data behind this ListView
        listView.setAdapter(adapter);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                //Create a toast text to display which item is clicked
                String clickItem = gameNames[position]+" is clicked";
                Toast.makeText(MainActivity.this, clickItem,Toast.LENGTH_LONG ).show();
            }
        });
    }
```

How to create your custom ListView

- Create the custom ListView using ArrayAdapter
- Create your custom XML layout for each item of ListView
- Create a entity class such as User, Product, and Customer class
- Create a your own class to extend ArrayAdapter class
- Modify the content of getView() method in your adapter class
- Set the adapter for your ListView

How to create your custom ListView

Step 1: Create your custom XML layout for each list item.

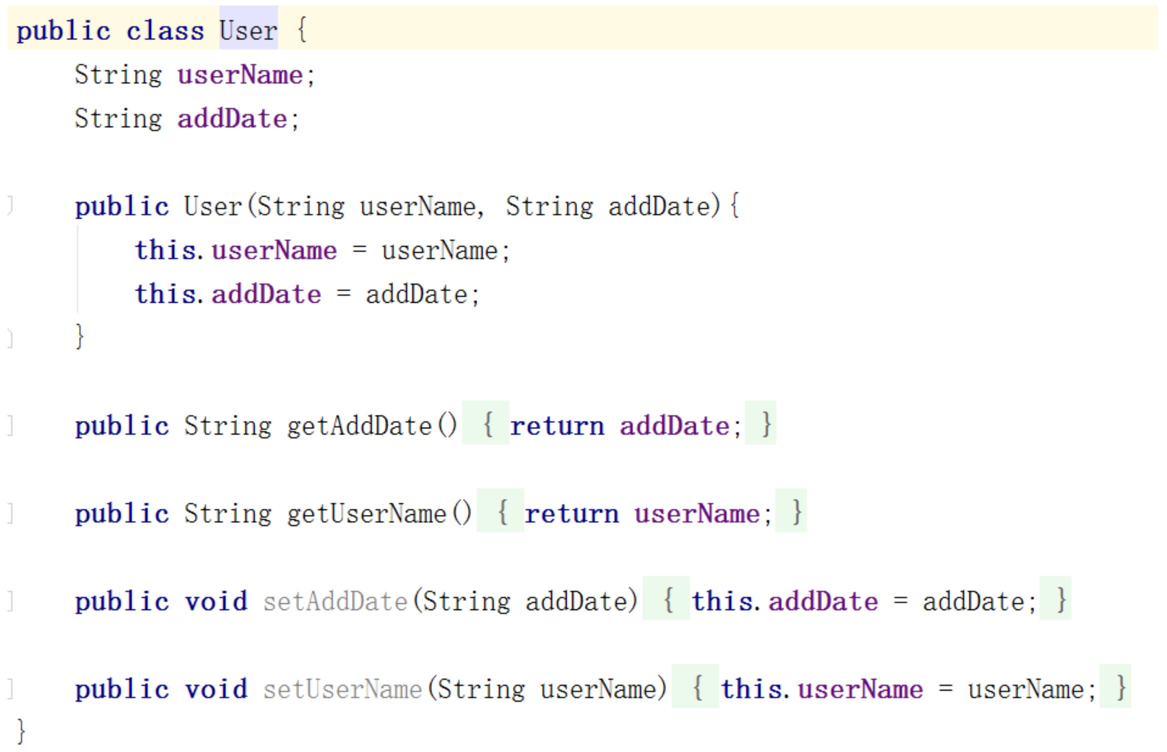


```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <ImageView
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:src="@drawable/message_icon"
        android:layout_margin="5dp"
        android:id="@+id/ivMessage"
        />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/ivMessage"
        android:orientation="vertical"
        android:layout_margin="5dp"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="loading..."
            android:id="@+id/tvUsername"
            android:textSize="20dp"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="loading..."
            android:id="@+id/tvAddDate"
            android:textSize="15dp"
            />
    </LinearLayout>
</RelativeLayout>
```

How to create your custom ListView

Step 2: Create a entity class such as User, Product, and Customer class

In this case, we create an User class.

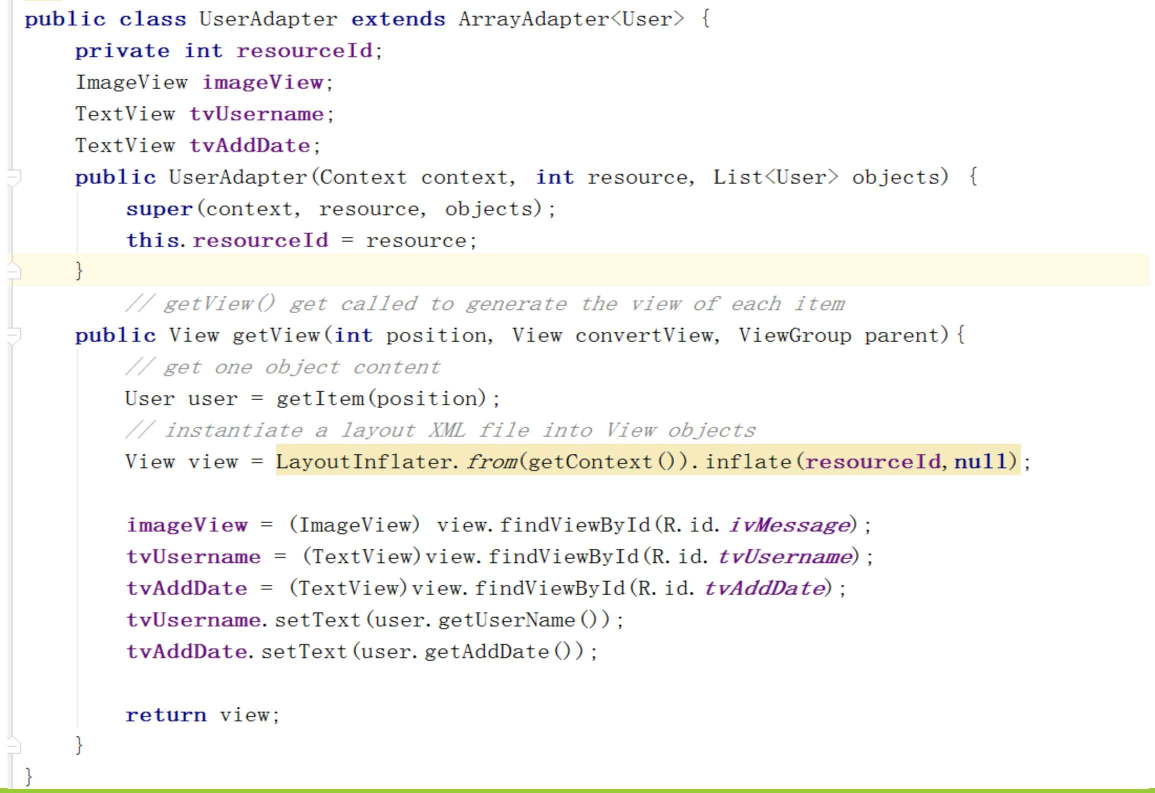


```java
public class User {
    String userName;
    String addDate;

    public User(String userName, String addDate){
        this.userName = userName;
        this.addDate = addDate;
    }

    public String getAddDate() { return addDate; }

    public String getUserName() { return userName; }

    public void setAddDate(String addDate) { this.addDate = addDate; }

    public void setUserName(String userName) { this.userName = userName; }
}
```

How to create your custom ListView

Step 3: Create a your own adapter class to extend ArrayAdapter class

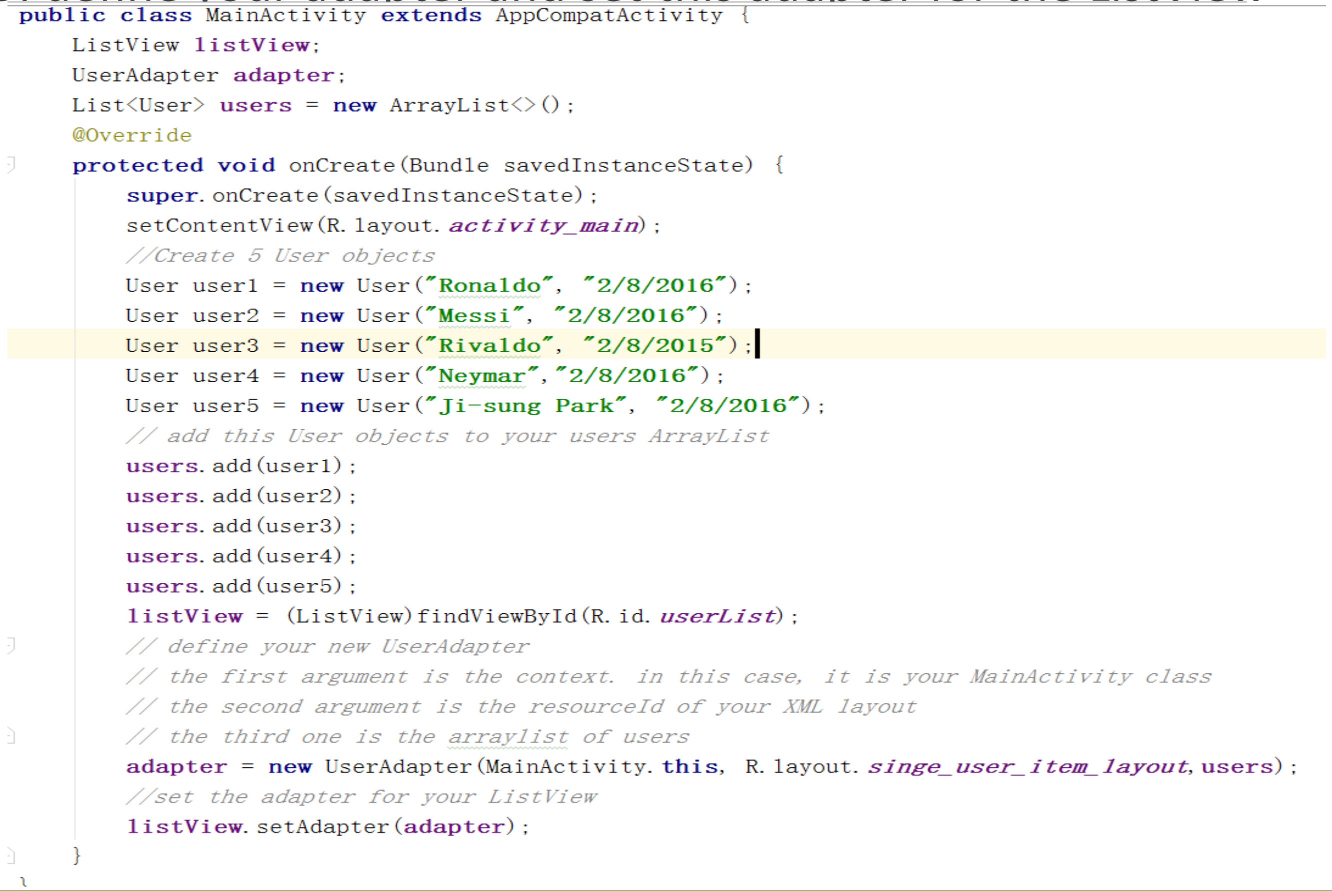
Step 4: Modify the content of getView() method in your adapter class



```java
public class UserAdapter extends ArrayAdapter<User> {
    private int resourceId;
    ImageView imageView;
    TextView tvUsername;
    TextView tvAddDate;
    public UserAdapter(Context context, int resource, List<User> objects) {
        super(context, resource, objects);
        this.resourceId = resource;
    }
        // getView() get called to generate the view of each item
    public View getView(int position, View convertView, ViewGroup parent){
        // get one object content
        User user = getItem(position);
        // instantiate a layout XML file into View objects
        View view = LayoutInflater.from(getContext()).inflate(resourceId,null);

        imageView = (ImageView) view.findViewById(R.id.ivMessage);
        tvUsername = (TextView)view.findViewById(R.id.tvUsername);
        tvAddDate = (TextView)view.findViewById(R.id.tvAddDate);
        tvUsername.setText(user.getUserName());
        tvAddDate.setText(user.getAddDate());

        return view;
    }
}
```

How to create your custom ListView

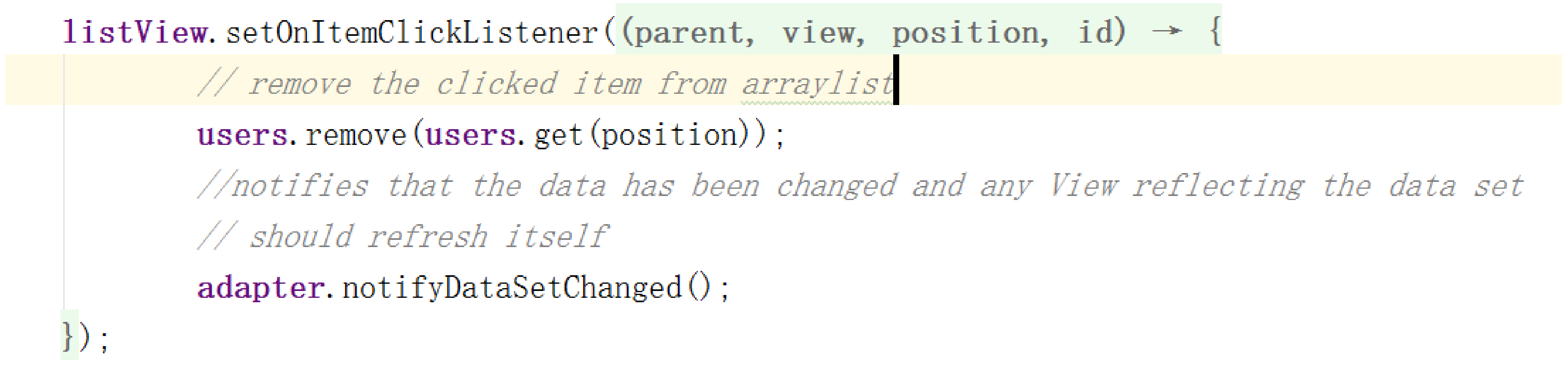
Step 5: define your adapter and set this adapter for the ListView



```java
public class MainActivity extends AppCompatActivity {
    ListView listView;
    UserAdapter adapter;
    List<User> users = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Create 5 User objects
        User user1 = new User("Ronaldo", "2/8/2016");
        User user2 = new User("Messi", "2/8/2016");
        User user3 = new User("Rivaldo", "2/8/2015");
        User user4 = new User("Neymar","2/8/2016");
        User user5 = new User("Ji-sung Park", "2/8/2016");
        // add this User objects to your users ArrayList
        users.add(user1);
        users.add(user2);
        users.add(user3);
        users.add(user4);
        users.add(user5);
        listView = (ListView)findViewById(R.id.userList);
        // define your new UserAdapter
        // the first argument is the context. in this case, it is your MainActivity class
        // the second argument is the resourceId of your XML layout
        // the third one is the arraylist of users
        adapter = new UserAdapter(MainActivity.this, R.layout.singe_user_item_layout,users);
        //set the adapter for your ListView
        listView.setAdapter(adapter);
    }
```

How to update your ListView

- Invoke notifyDataSetChanged() when the data has been changed



```
listView.setOnItemClickListener((parent, view, position, id) → {
        // remove the clicked item from arraylist
        users.remove(users.get(position));
        //notifies that the data has been changed and any View reflecting the data set
        // should refresh itself
        adapter.notifyDataSetChanged();
});
```

How to optimize your ListView

- Increase the efficiency of ListView by ViewHolder

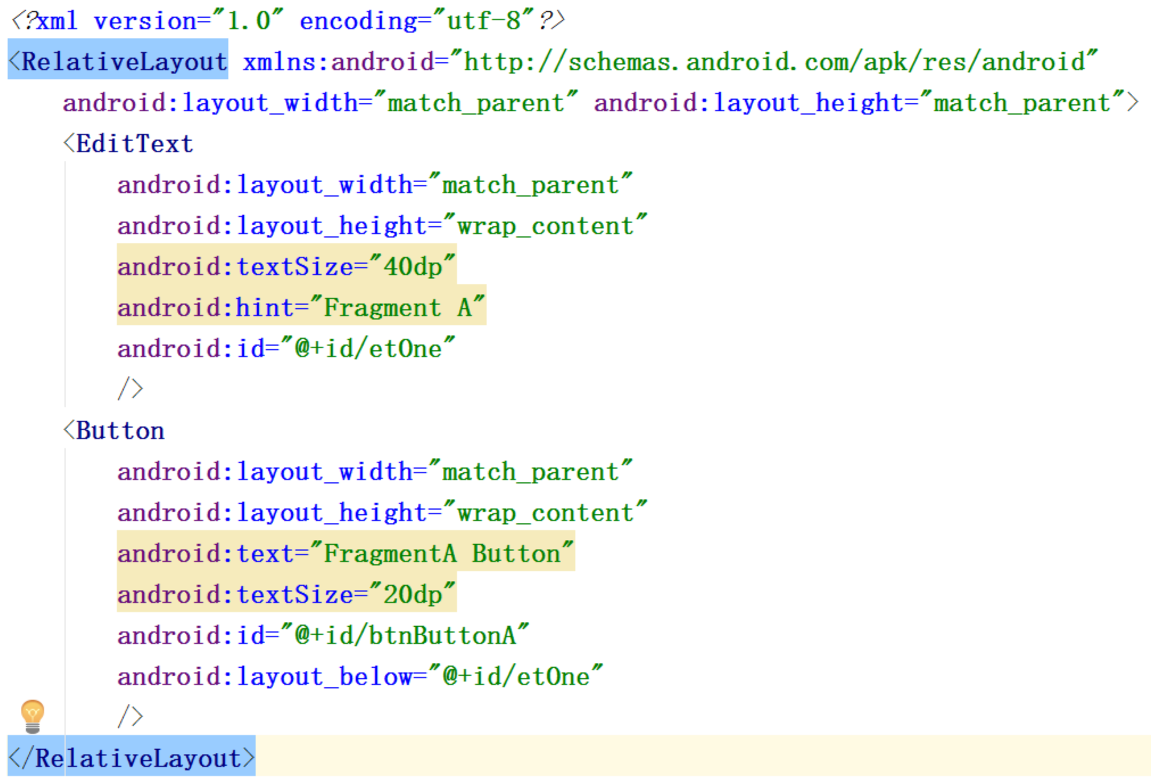
```java
public View getView(int position, View convertView, ViewGroup parent){
    // get one object content
    User user = getItem(position);
    System.out.println("@ "+user.getUserName());
    ViewHolder viewHolder;
    View view;
    if (convertView==null){
        //create a new ViewHolder
        viewHolder = new ViewHolder();
        view = LayoutInflater.from(getContext()).inflate(resourceId,null);
        // store instance components into the viewHolder
        viewHolder.imageView = (ImageView) view.findViewById(R.id.ivMessage);
        viewHolder.tvUsername =(TextView)view.findViewById(R.id.tvUsername);
        viewHolder.tvAddDate = (TextView)view.findViewById(R.id.tvAddDate);
        // store the viewHolder into the view
        view.setTag(viewHolder);
    }else {
        view = convertView;
        //recover the viewHolder that store the previous instance components again
        viewHolder = (ViewHolder) view.getTag();
    }
    viewHolder.tvUsername.setText(user.getUserName());
    viewHolder.tvAddDate.setText(user.getAddDate());
    return view;
}
// Define a ViewHolder class
class ViewHolder{
    ImageView imageView;
    TextView tvUsername;
    TextView tvAddDate;
}
```

Fragment

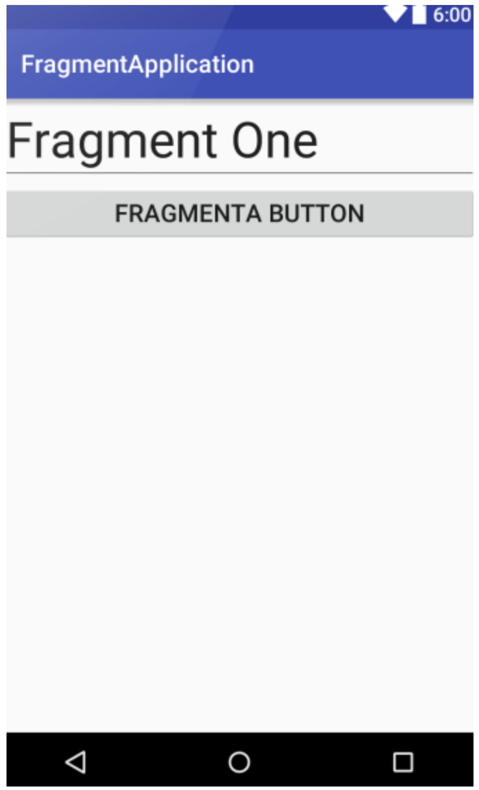
- Fragment represents a portion of user interface in an Activity
- You can use multiple fragments in one activity or reuse a fragment in multiple activities
- A fragment must always embedded in an activity
- A fragment also has a lifecycle directly affected by the host activity’s lifecycle
- FragmentManager provides a function findFragmentById()
- FragmentManager manages fragment transactions such as adding a fragment, replacing a fragment with another fragment, or removing a fragment

How to create a simple fragment

- Step 1: Create a layout for your fragment

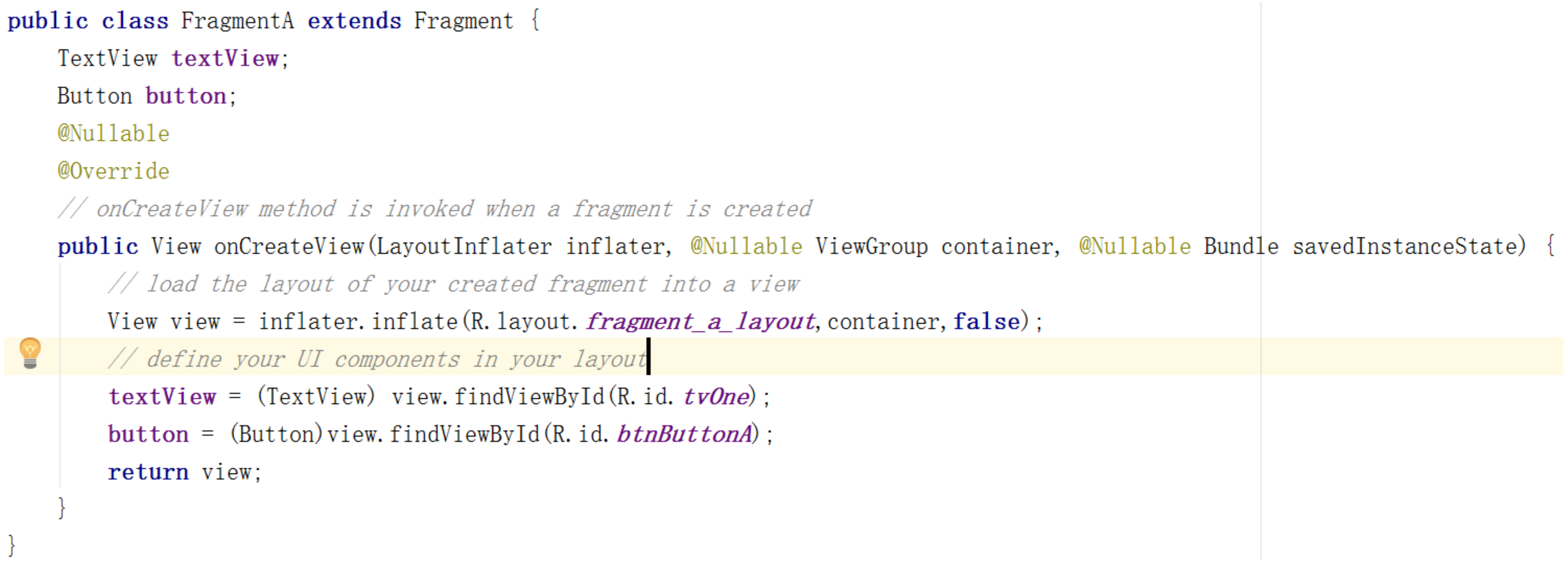


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="40dp"
        android:hint="Fragment A"
        android:id="@+id/etOne"
        />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="FragmentA Button"
        android:textSize="20dp"
        android:id="@+id/btnButtonA"
        android:layout_below="@+id/etOne"
        />
</RelativeLayout>
```



How to create a simple fragment

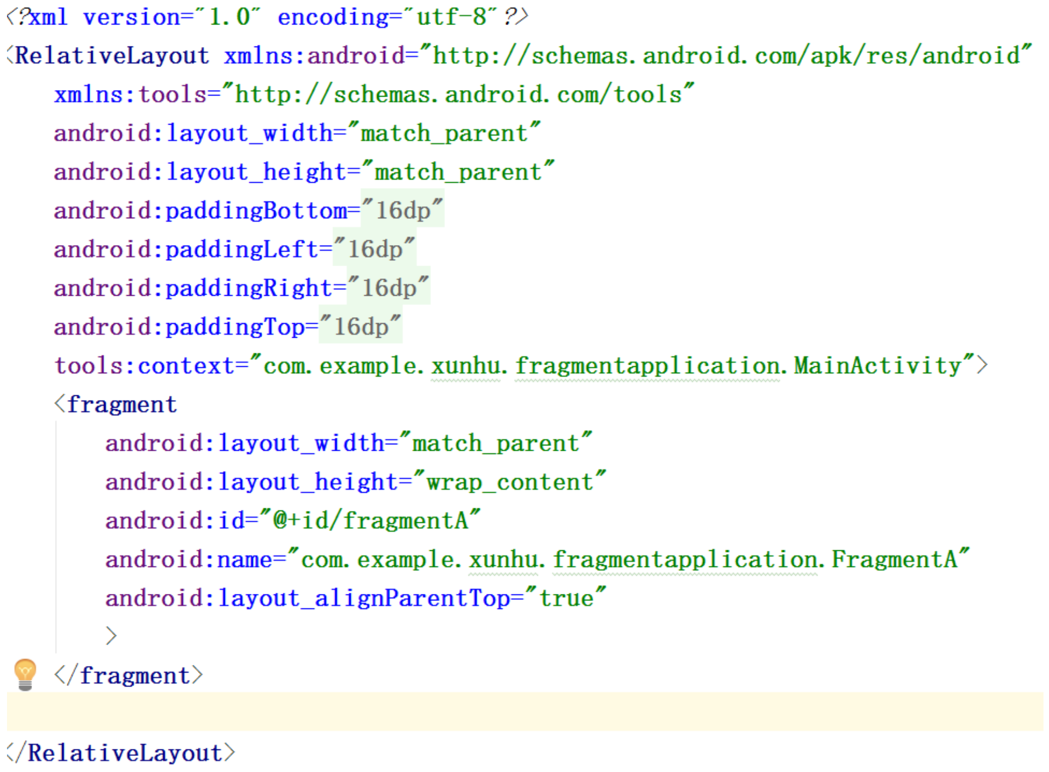
Step 2: Create a Fragment Java class and rewrite onCreateView() method to load the layout of your fragment



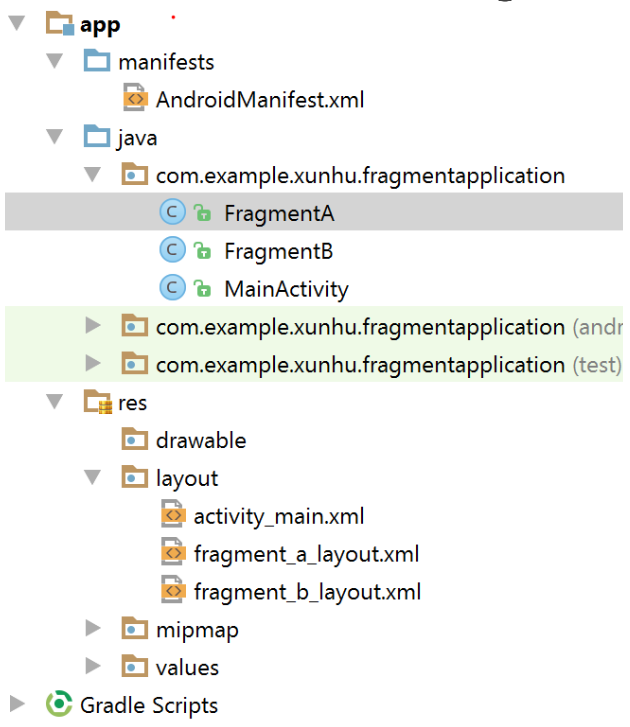
```java
public class FragmentA extends Fragment {
    TextView textView;
    Button button;
    @Nullable
    @Override
    // onCreateView method is invoked when a fragment is created
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        // load the layout of your created fragment into a view
        View view = inflater.inflate(R.layout.fragment_a_layout,container,false);
        // define your UI components in your layout
        textView = (TextView) view.findViewById(R.id.tvOne);
        button = (Button)view.findViewById(R.id.btnButtonA);
        return view;
    }
}
```

How to create a simple fragment

Step 3: Create a fragment tag in your activity layout file, and then name that fragment tag as the path of your fragment java class



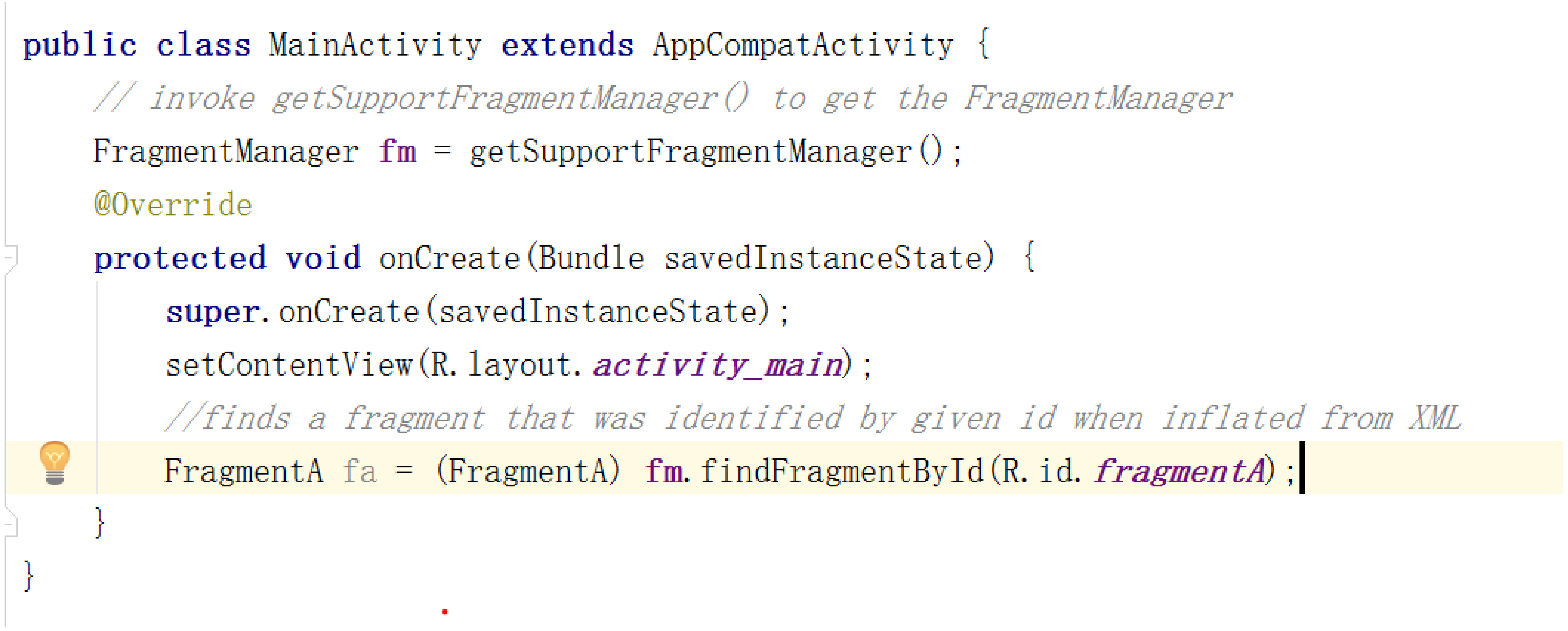
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.xunhu.fragmentapplication.MainActivity">
    <fragment
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/fragmentA"
        android:name="com.example.xunhu.fragmentapplication.FragmentA"
        android:layout_alignParentTop="true"
        >
    </fragment>

</RelativeLayout>
```



```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▼ com.example.xunhu.fragmentapplication
        FragmentA
        FragmentB
        MainActivity
    ▶ com.example.xunhu.fragmentapplication (andr
    ▶ com.example.xunhu.fragmentapplication (test)
  ▼ res
      drawable
    ▼ layout
        activity_main.xml
        fragment_a_layout.xml
        fragment_b_layout.xml
    ▶ mipmap
    ▶ values
▶ Gradle Scripts
```

How to create a simple fragment

Step 4: Load your activity xml layout into the activity class



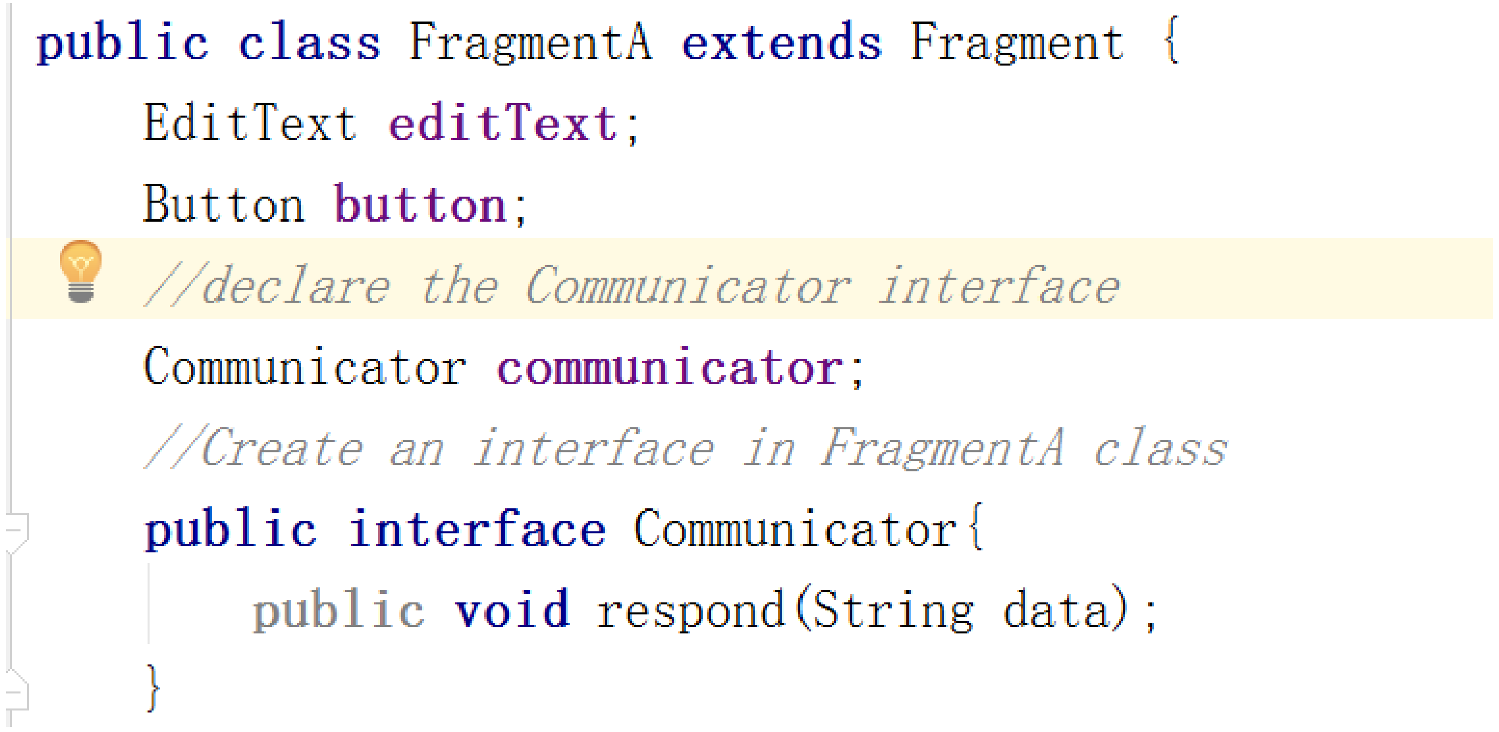
```
public class MainActivity extends AppCompatActivity {
    // invoke getSupportFragmentManager() to get the FragmentManager
    FragmentManager fm = getSupportFragmentManager();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //finds a fragment that was identified by given id when inflated from XML
        FragmentA fa = (FragmentA) fm.findFragmentById(R.id.fragmentA);
    }
}
```

Communication Between Fragments

- Different fragments communicate with each other via activity
- Define an interface in the Fragment class and implement the interface within the activity
- The fragment captures the interface implementation during its onAttach() callback method and then call the interface methods in order to communicate with the Activity

Communication Between Fragments

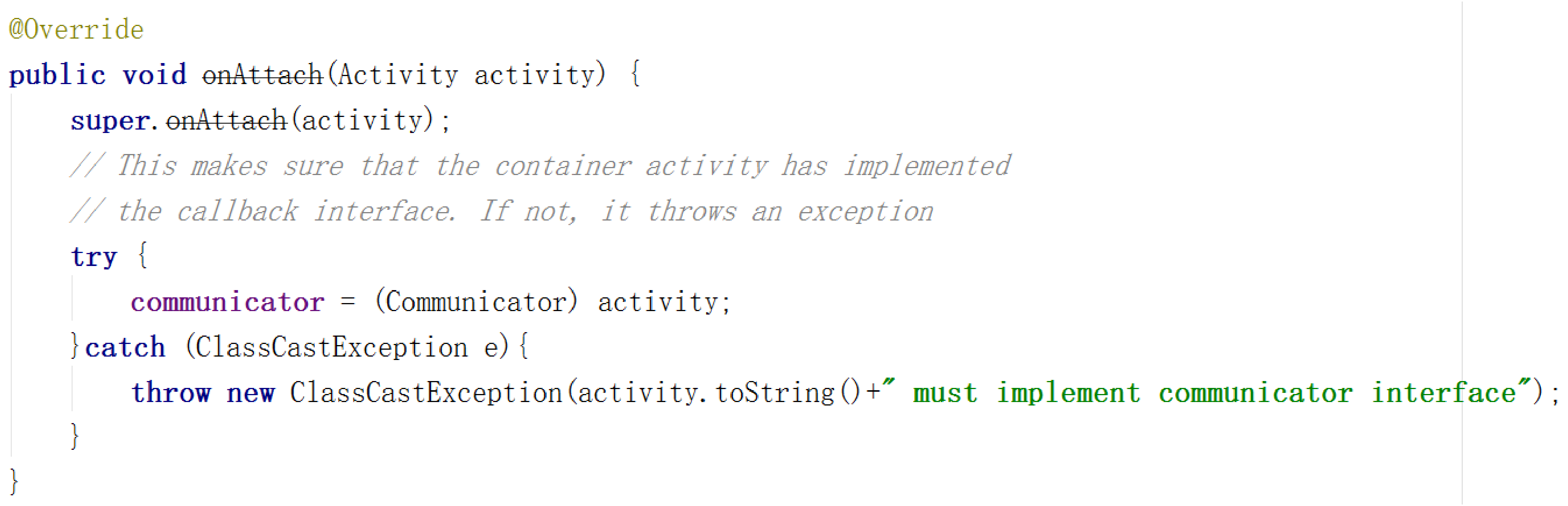
Step 1: Create an interface within fragment class and declare the interace



```
public class FragmentA extends Fragment {
    EditText editText;
    Button button;
    //declare the Communicator interface
    Communicator communicator;
    //Create an interface in FragmentA class
    public interface Communicator{
        public void respond(String data);
    }
```

Communication between Fragments

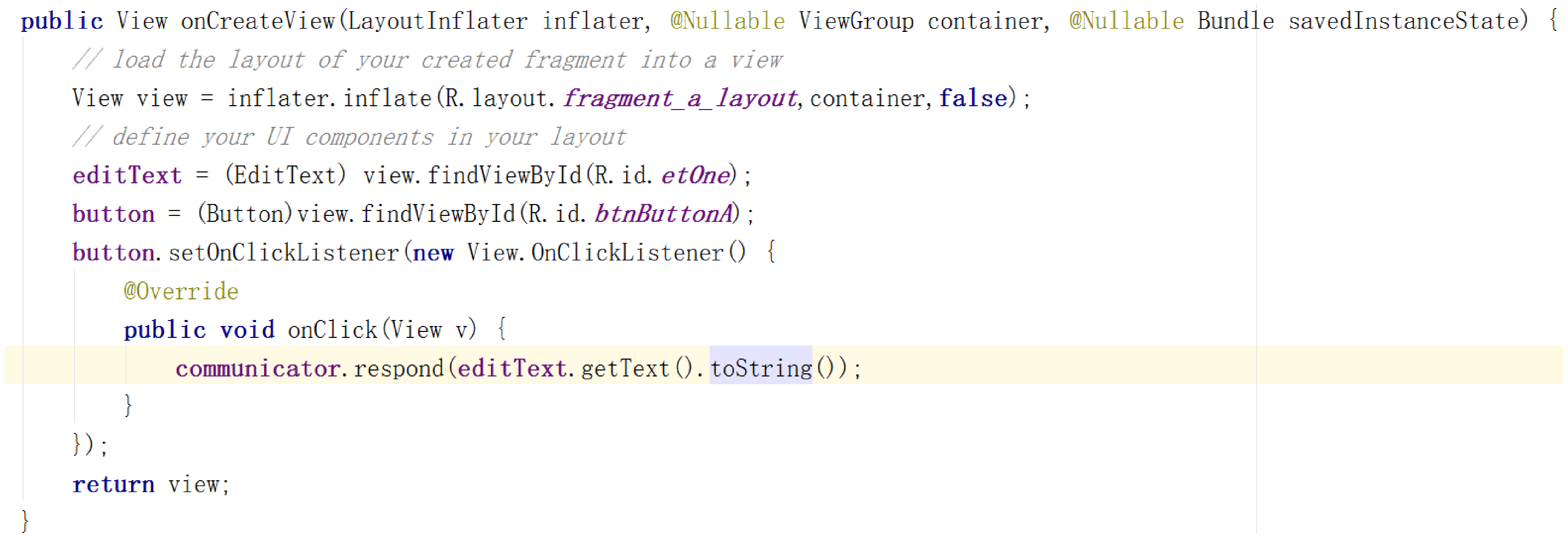
Step 2: captures the interface implementation within onAttach() method in Fragment class



```
@Override
public void ~~onAttach~~(Activity activity) {
    super.~~onAttach~~(activity);
    // This makes sure that the container activity has implemented
    // the callback interface. If not, it throws an exception
    try {
        communicator = (Communicator) activity;
    }catch (ClassCastException e){
        throw new ClassCastException(activity.toString()+" must implement communicator interface");
    }
}
```

Communication between Fragments

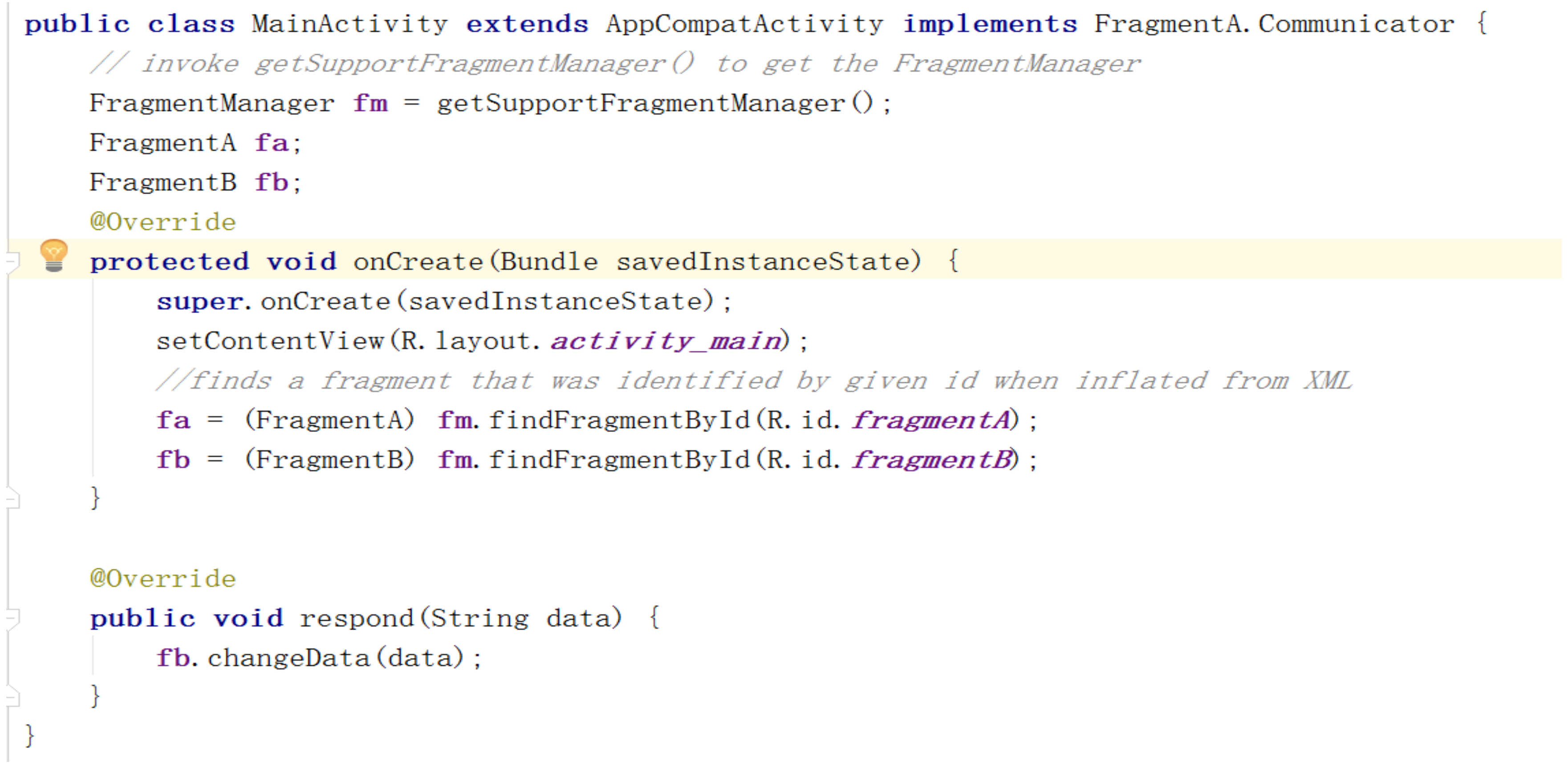
Step 3: call the interface method to communicate with the container activity



```java
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    // load the layout of your created fragment into a view
    View view = inflater.inflate(R.layout.fragment_a_layout, container, false);
    // define your UI components in your layout
    editText = (EditText) view.findViewById(R.id.etOne);
    button = (Button)view.findViewById(R.id.btnButtonA);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            communicator.respond(editText.getText().toString());
        }
    });
    return view;
}
```

Communication between Fragments

Step 4: implement interface in Activity class

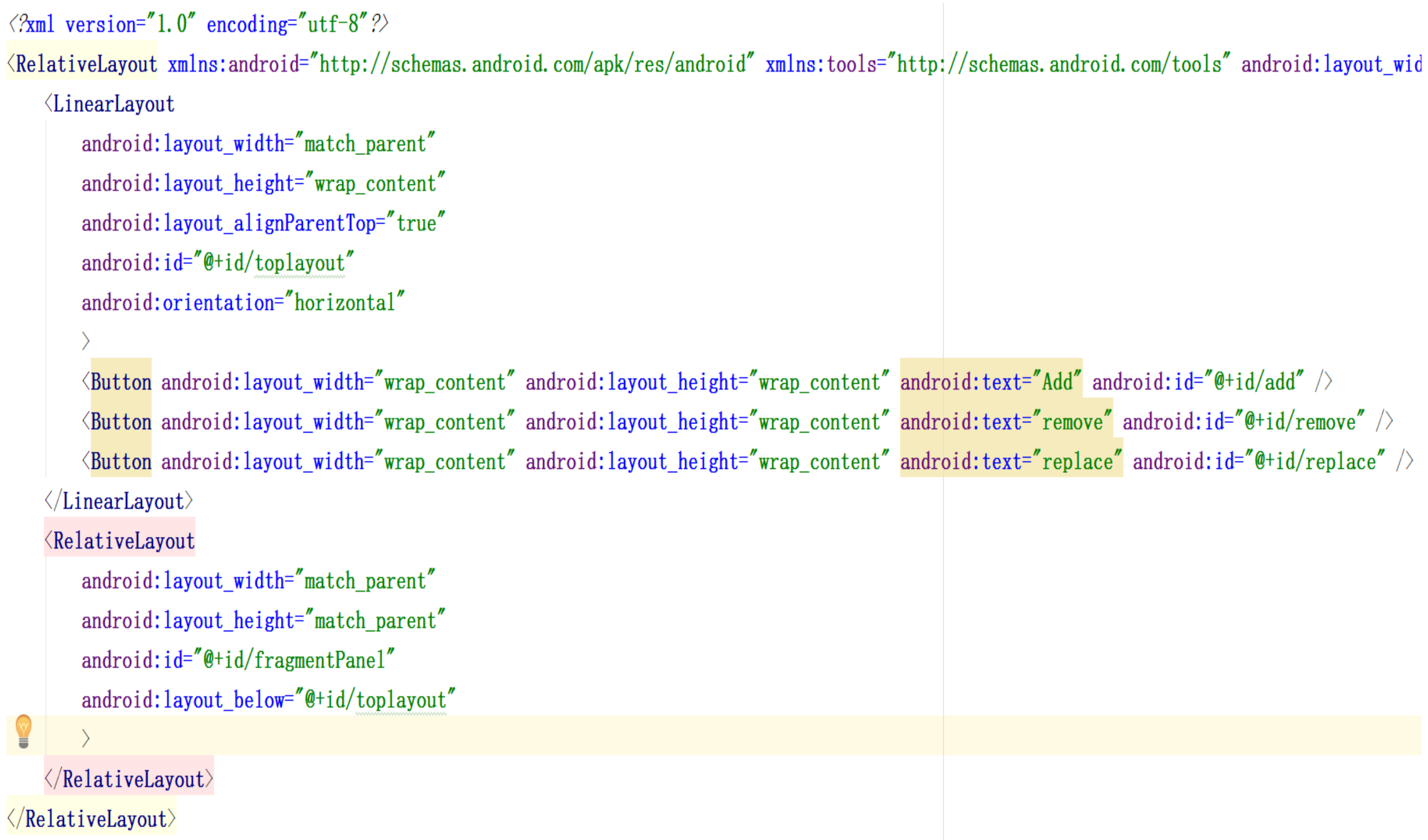


```java
public class MainActivity extends AppCompatActivity implements FragmentA.Communicator {
    // invoke getSupportFragmentManager() to get the FragmentManager
    FragmentManager fm = getSupportFragmentManager();
    FragmentA fa;
    FragmentB fb;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //finds a fragment that was identified by given id when inflated from XML
        fa = (FragmentA) fm.findFragmentById(R.id.fragmentA);
        fb = (FragmentB) fm.findFragmentById(R.id.fragmentB);
    }

    @Override
    public void respond(String data) {
        fb.changeData(data);
    }
}
```

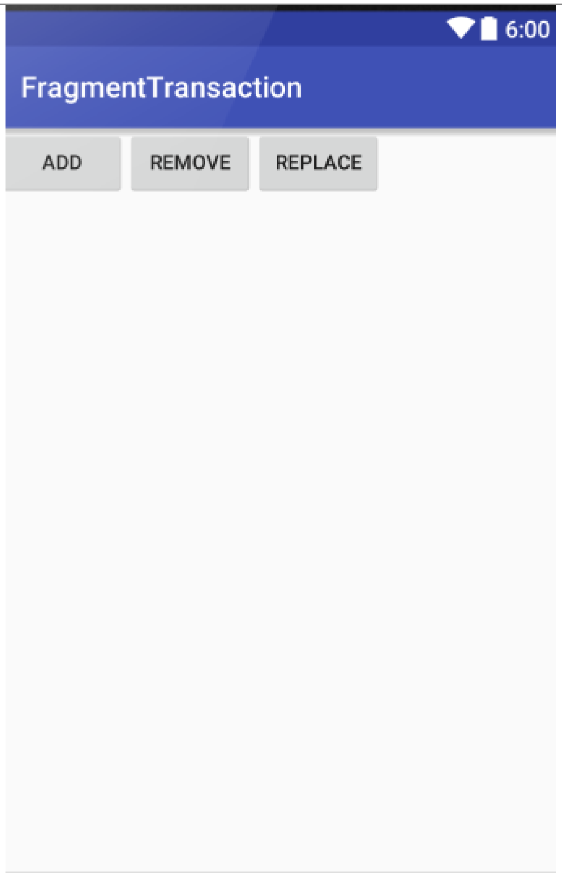
Fragment Transactions

- add(): add a fragment
- replace(): replace a fragment with another fragment in the activity
- remove(): remove a fragment from the activity

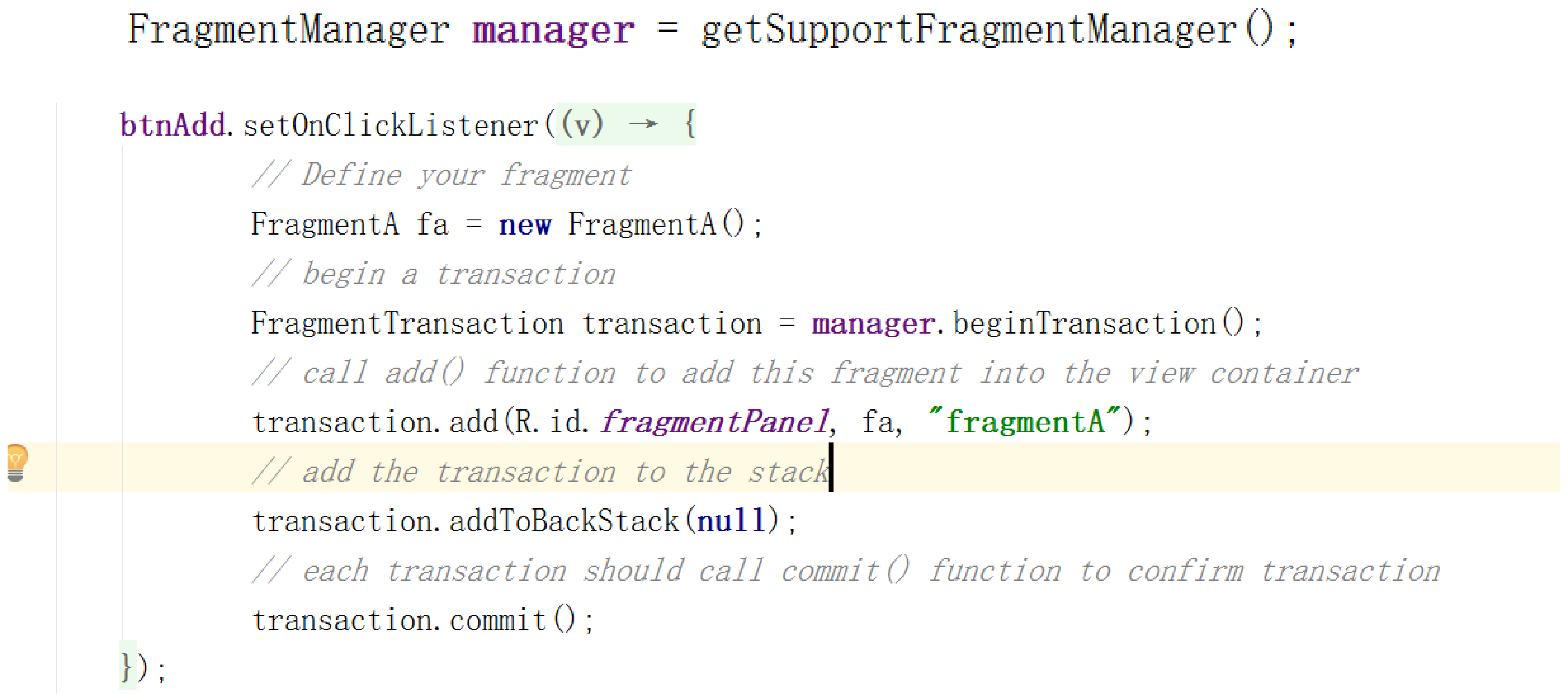
Add fragment to a layout



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout_wid
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:id="@+id/toplayout"
        android:orientation="horizontal"
        >
        <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Add" android:id="@+id/add" />
        <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="remove" android:id="@+id/remove" />
        <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="replace" android:id="@+id/replace" />
    </LinearLayout>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/fragmentPanel"
        android:layout_below="@+id/toplayout"
        >
    </RelativeLayout>
</RelativeLayout>
```

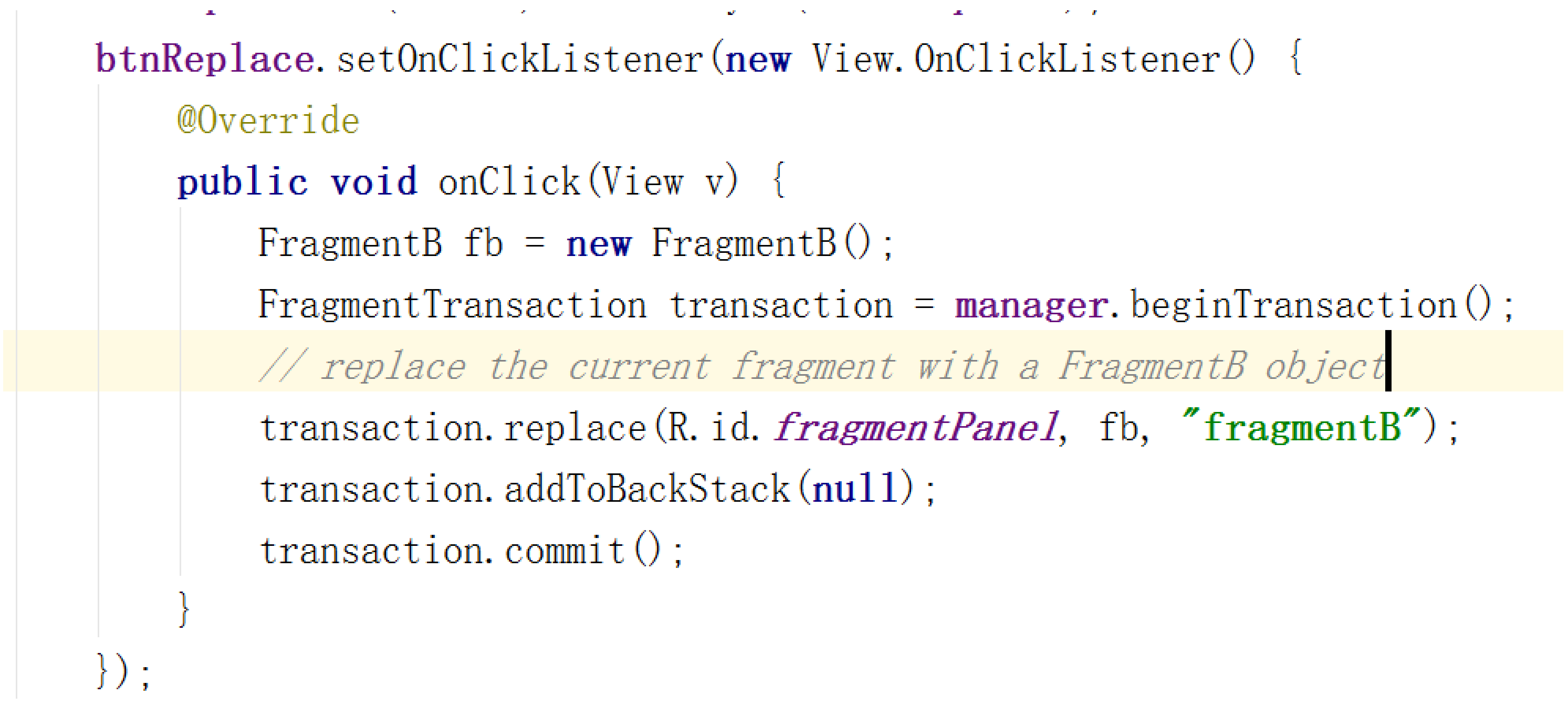


Add a fragment



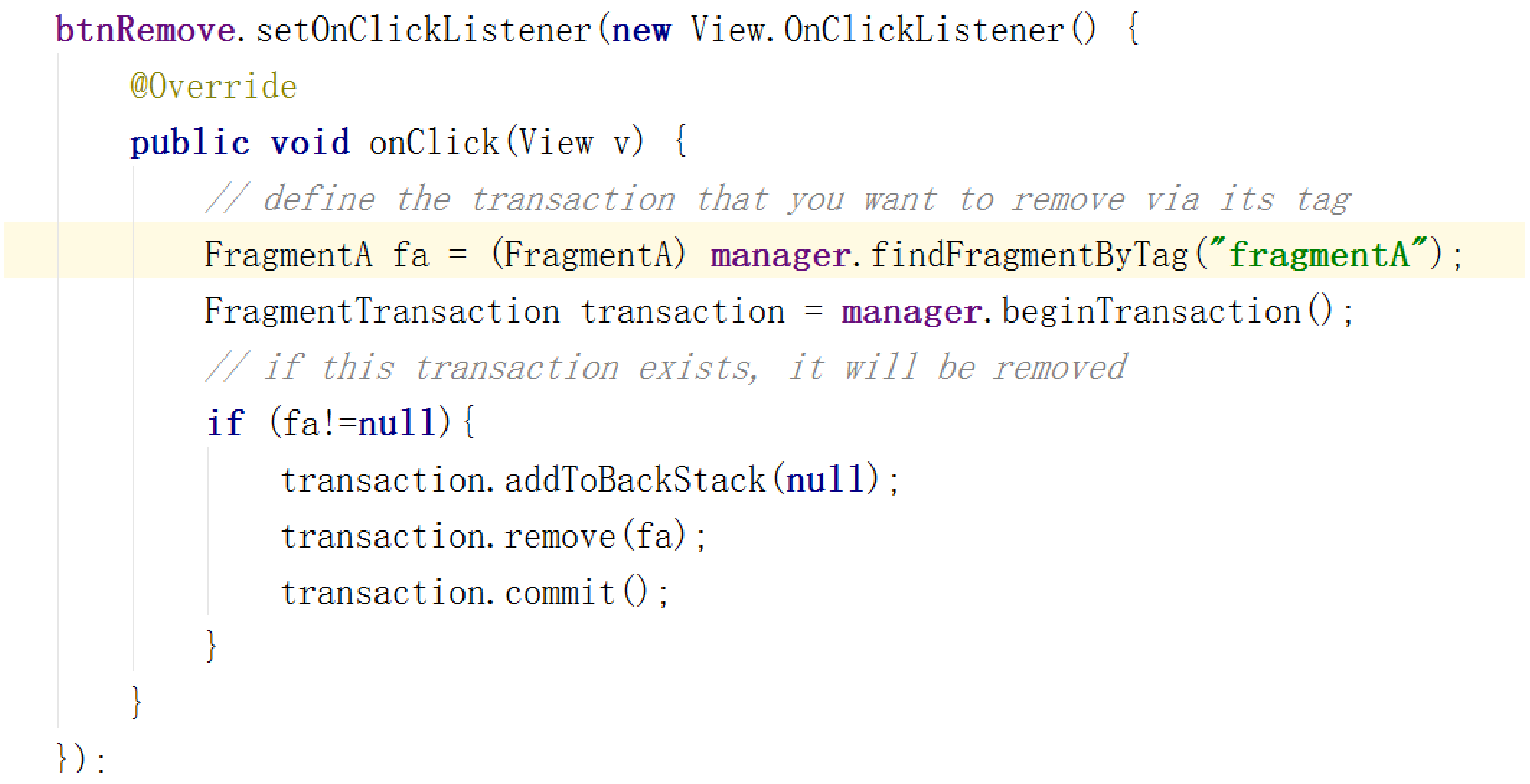
```
FragmentManager manager = getSupportFragmentManager();

btnAdd.setOnClickListener((v) → {
        // Define your fragment
        FragmentA fa = new FragmentA();
        // begin a transaction
        FragmentTransaction transaction = manager.beginTransaction();
        // call add() function to add this fragment into the view container
        transaction.add(R.id.fragmentPanel, fa, "fragmentA");
        // add the transaction to the stack
        transaction.addToBackStack(null);
        // each transaction should call commit() function to confirm transaction
        transaction.commit();
});
```

Replace a transaction



```
btnReplace.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FragmentB fb = new FragmentB();
        FragmentTransaction transaction = manager.beginTransaction();
        // replace the current fragment with a FragmentB object
        transaction.replace(R.id.fragmentPanel, fb, "fragmentB");
        transaction.addToBackStack(null);
        transaction.commit();
    }
});
```

Remove a transaction



```java
btnRemove.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // define the transaction that you want to remove via its tag
        FragmentA fa = (FragmentA) manager.findFragmentByTag("fragmentA");
        FragmentTransaction transaction = manager.beginTransaction();
        // if this transaction exists, it will be removed
        if (fa!=null){
            transaction.addToBackStack(null);
            transaction.remove(fa);
            transaction.commit();
        }
    }
});
```

Slide Pages

- ViewPager
- FragmentStatePagerAdapter