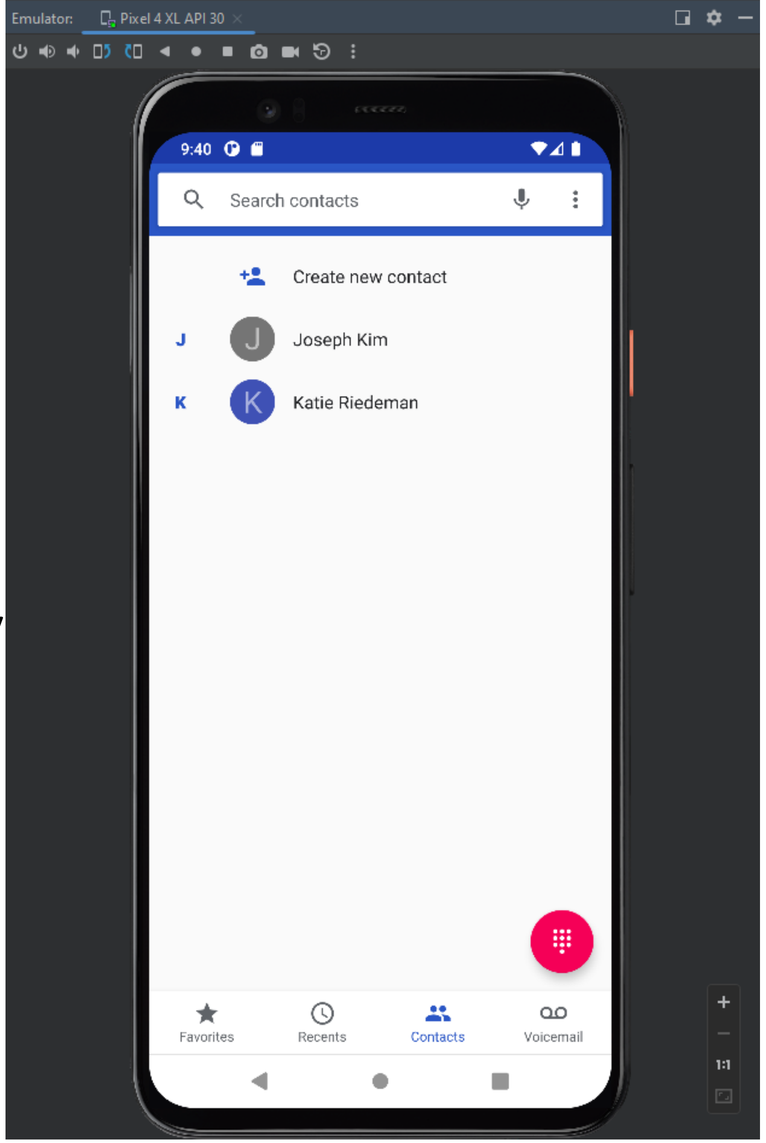
Reading the contacts

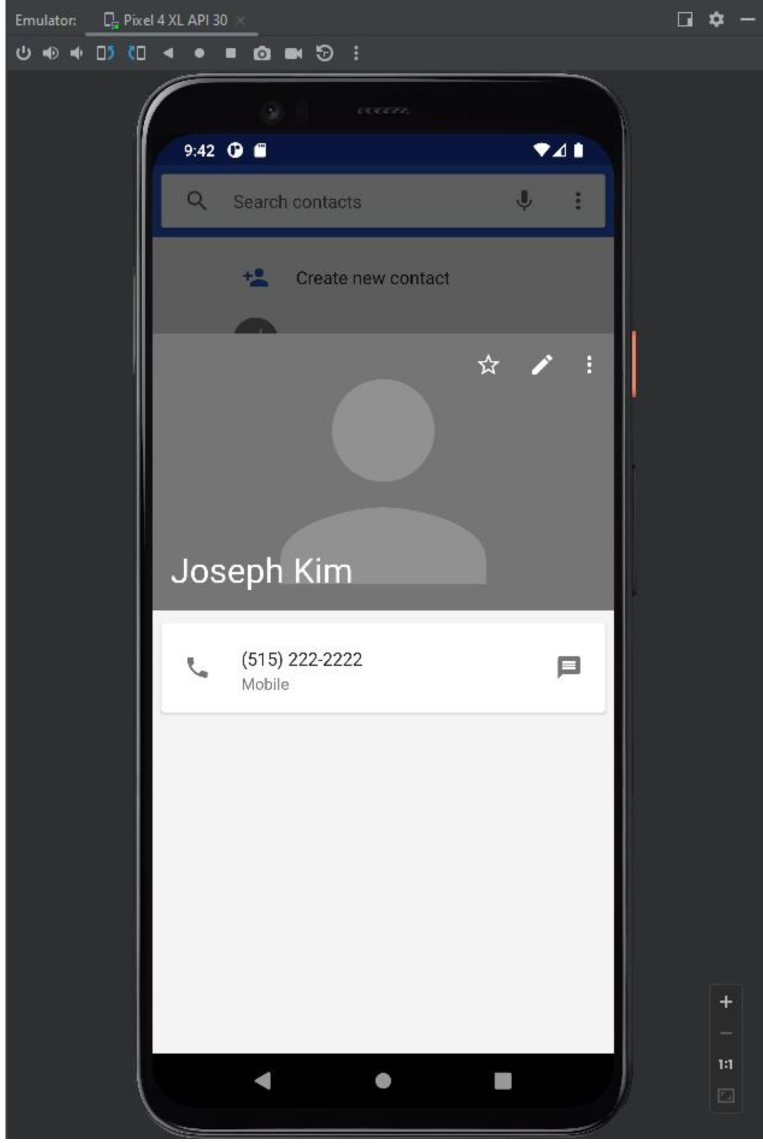
Dr. Charles Yu

- Demo1:
  - ContactsPhoneNumber
    - It still needs user’s providing access rights in the run time

ContactsPhone Number

- Let's see what I have in the emulator.
  - The contacts are randomly input into the emulator
  - Click the 1st contact, then we can see the detail

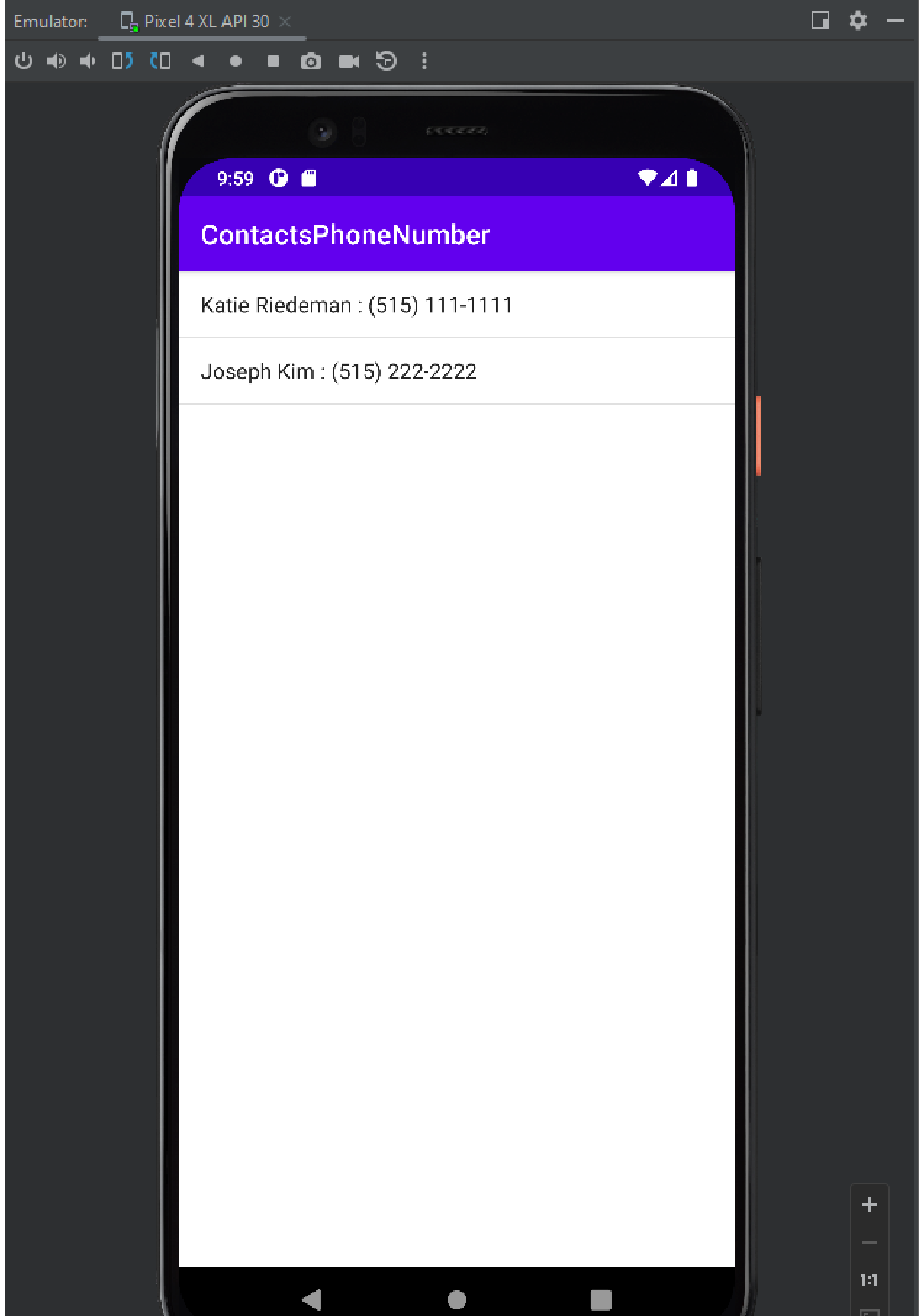




ContactsPhoneNumber

- Run the App and

let’s see how it looks like?



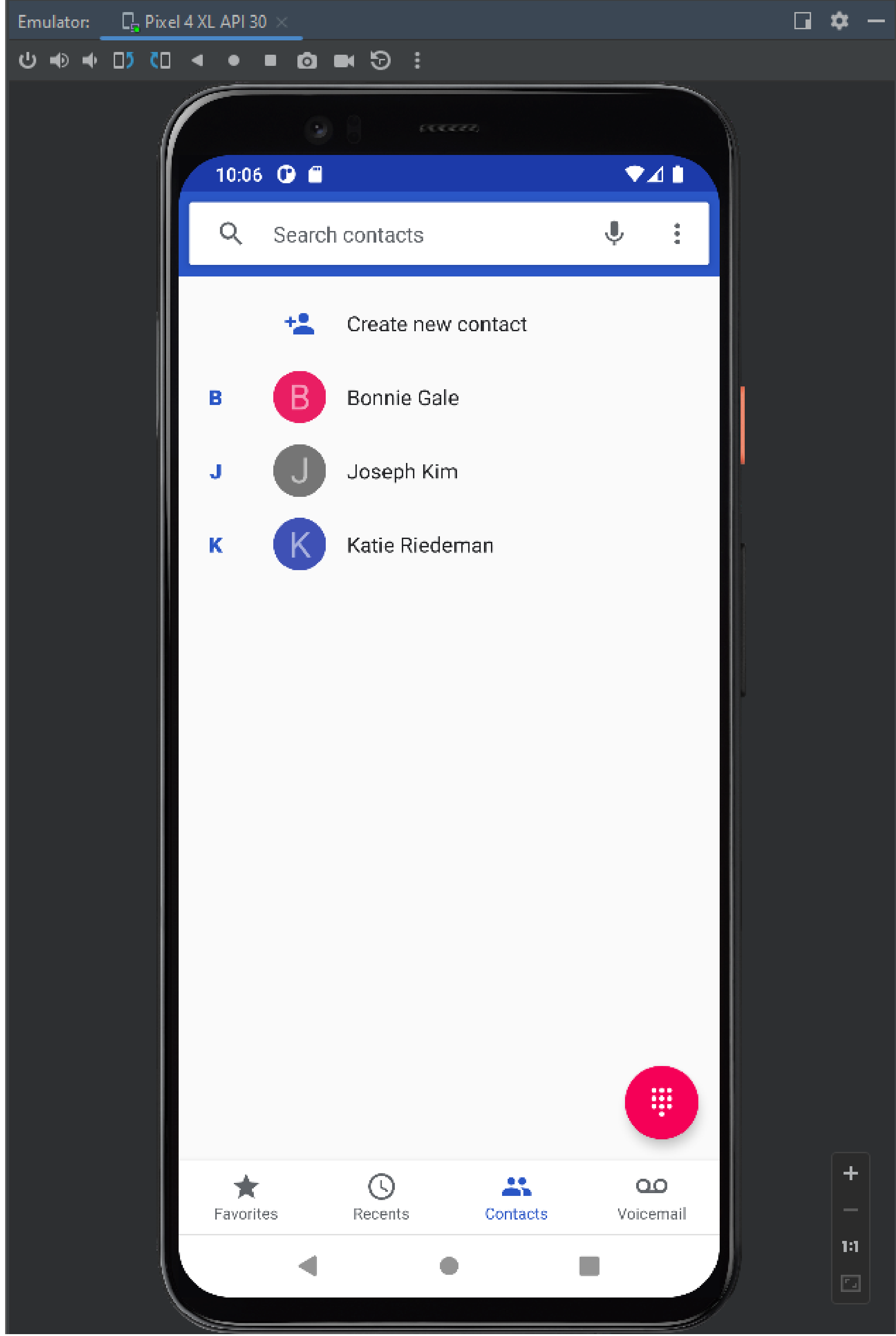
ContactsPhoneNumber

Katie Riedeman : (515) 111-1111

Joseph Kim : (515) 222-2222

ContactsPhoneNumber

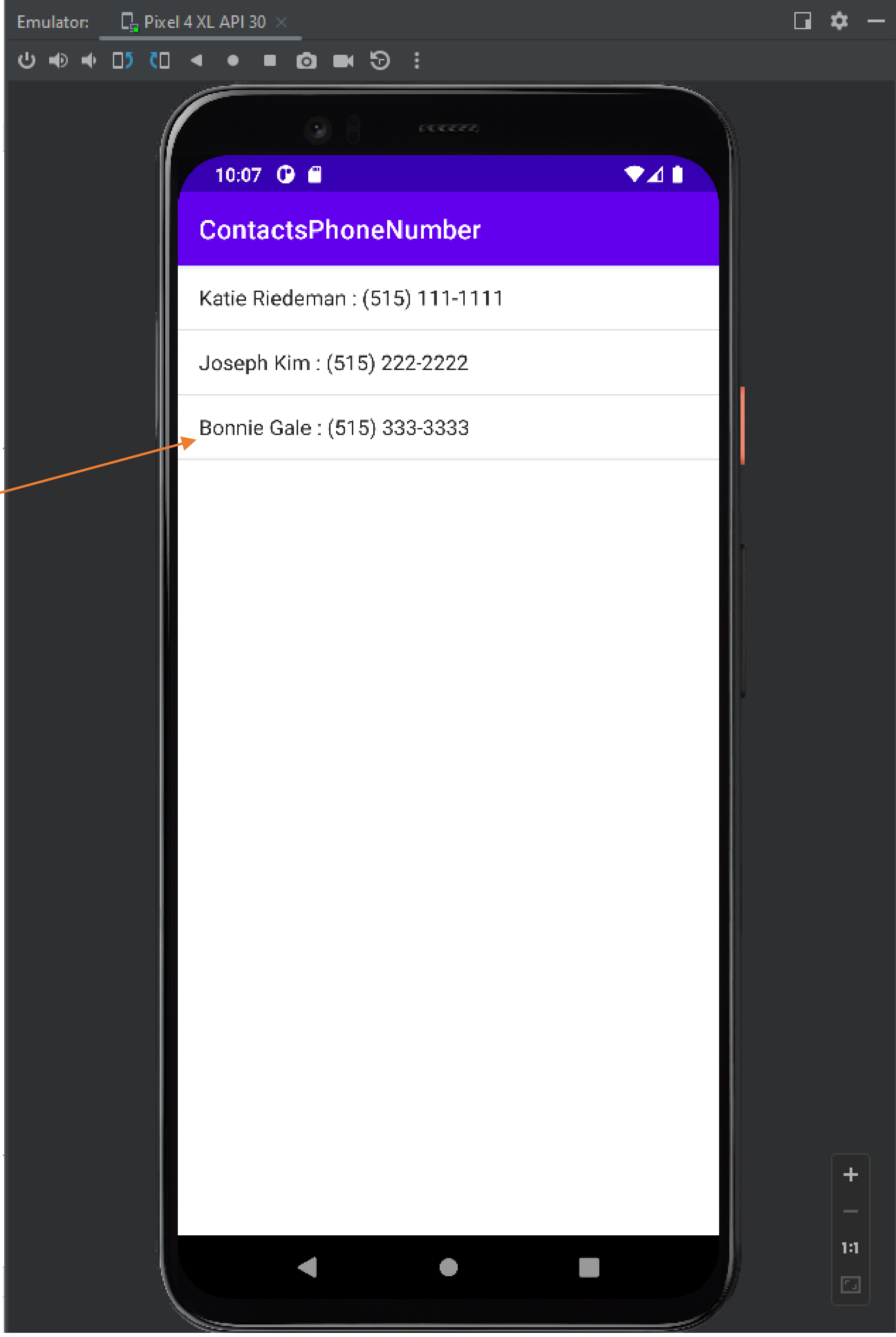
- If we go back to the "Phone app" → Contacts
- Add a new contact



ContactsPhoneNumber

- Go back to the ContactsPhone Number App
- You will see one more contact

is read into the app



ContactsPhoneNumber

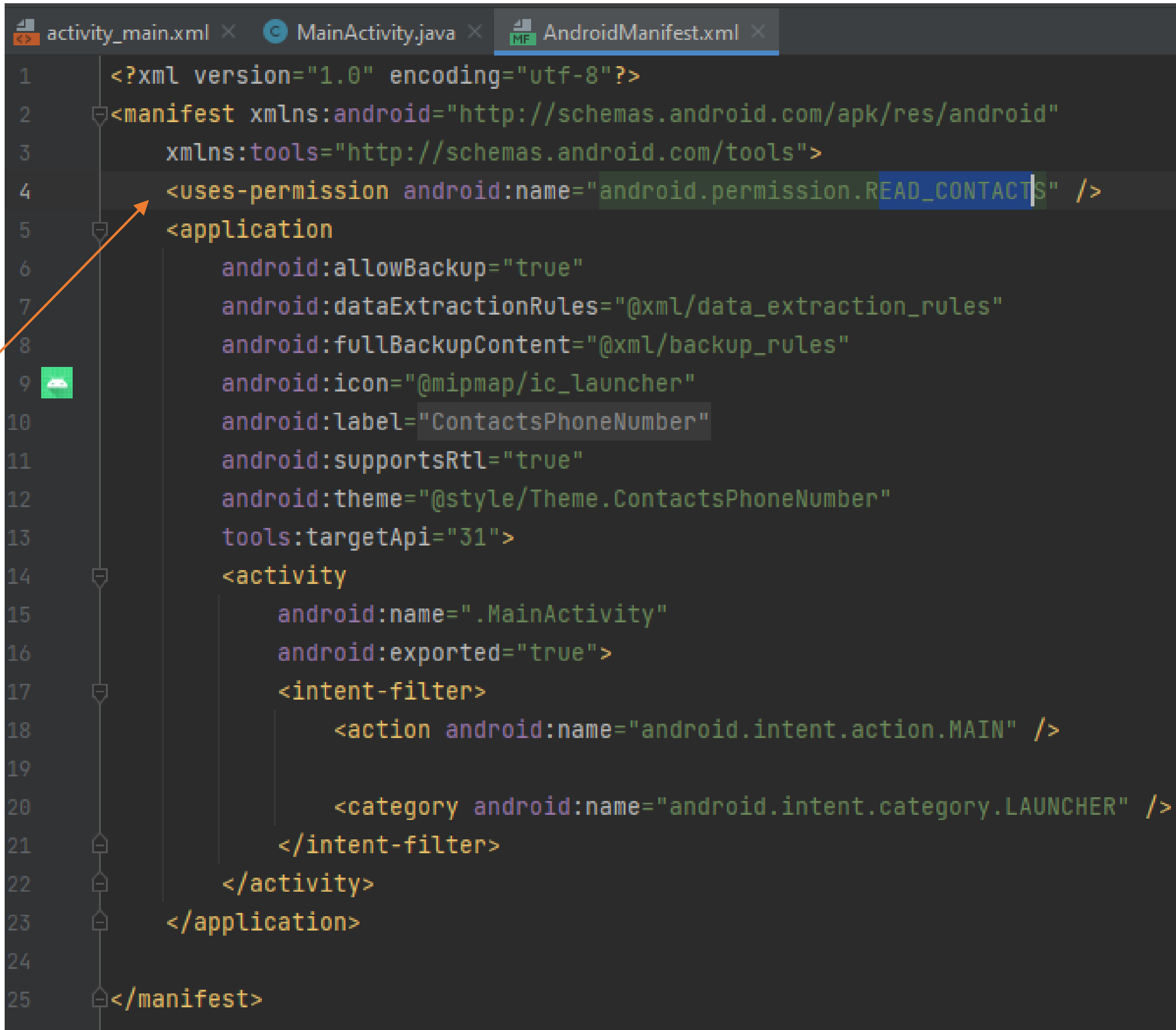
Katie Riedeman : (515) 111-1111

Joseph Kim : (515) 222-2222

Bonnie Gale : (515) 333-3333

ContactsPhoneNumber

- Let’s see our AndroidManifest.xml
- Nothing is special
  - Just one Activity!
- Watch out!
  - **android.permission.READ_CONTACTS**
- Let’s see MainActivity.java

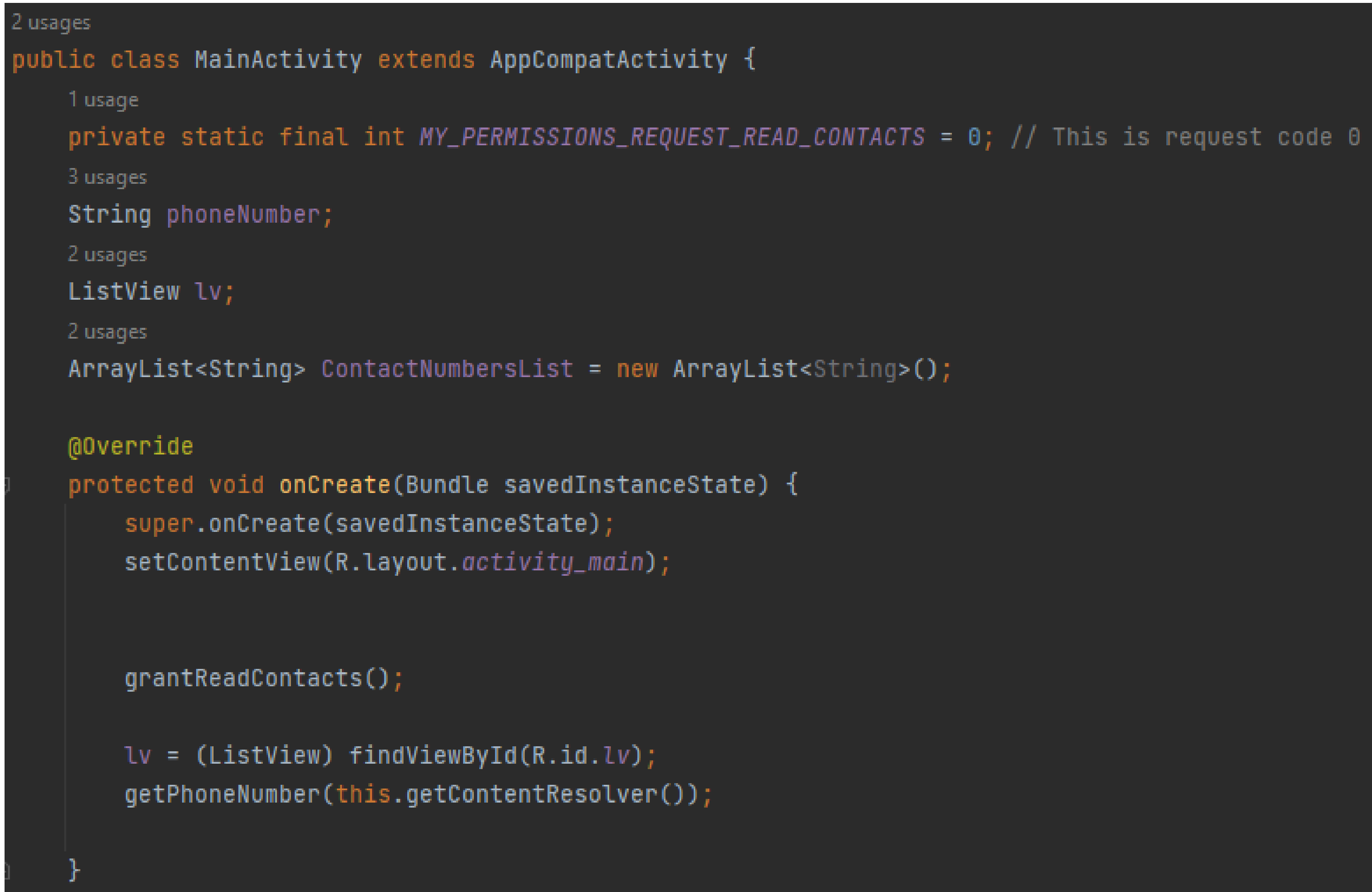


activity_main.xml × MainActivity.java × AndroidManifest.xml ×

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="ContactsPhoneNumber"
        android:supportsRtl="true"
        android:theme="@style/Theme.ContactsPhoneNumber"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

ContactsPhoneNumber

- This is our onCreate() looks like
- Let’s see our activity_main.
- You will know how the contact list is shown over there



```
2 usages
public class MainActivity extends AppCompatActivity {
    1 usage
    private static final int MY_PERMISSIONS_REQUEST_READ_CONTACTS = 0; // This is request code 0
    3 usages
    String phoneNumber;
    2 usages
    ListView lv;
    2 usages
    ArrayList<String> ContactNumbersList = new ArrayList<String>();

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


        grantReadContacts();

        lv = (ListView) findViewById(R.id.lv);
        getPhoneNumber(this.getContentResolver());

    }
```

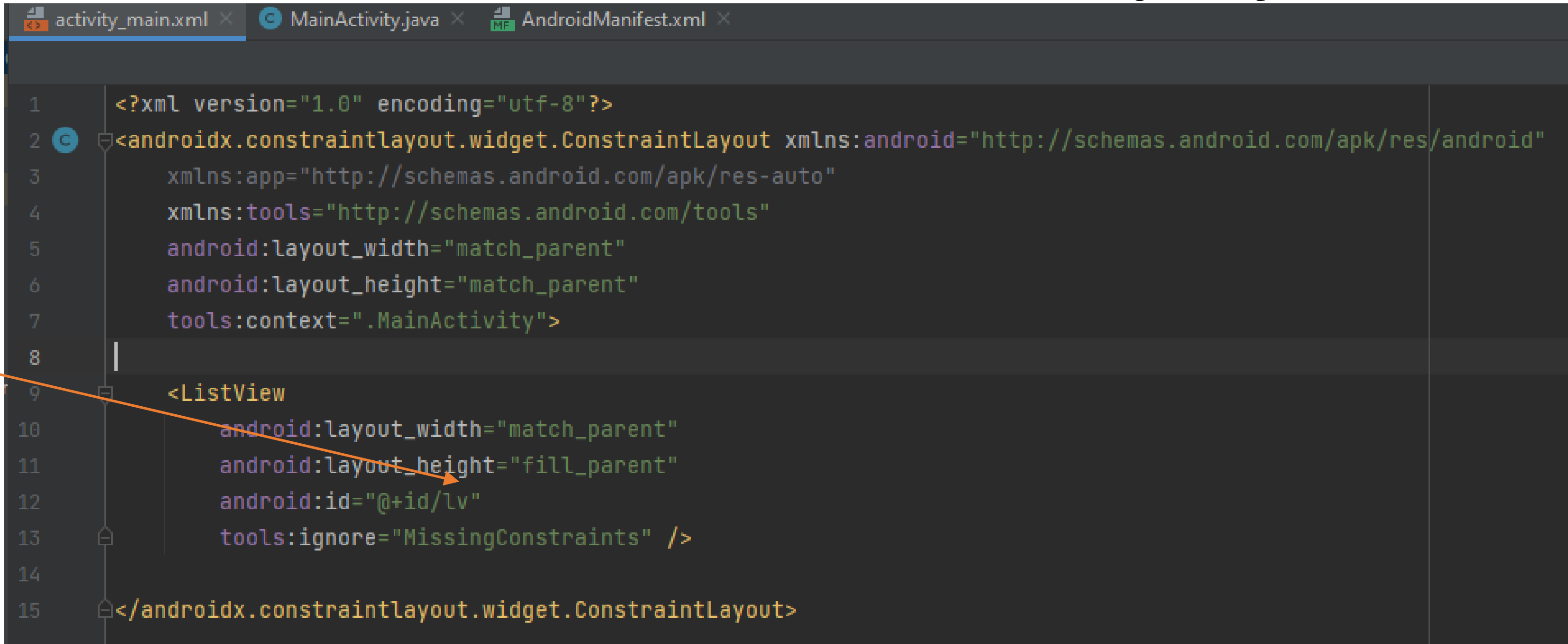
ContactsPhoneNumber

- I use the ListView as my UI component.
- I will show you how to use the **instance (object) of ListView** to bind with an **adapter**
- The **adapter** is populated with the **data** from the **return of a query**.
- Query for contacts in

this phone

- The id is called “lv”,

we will use that later

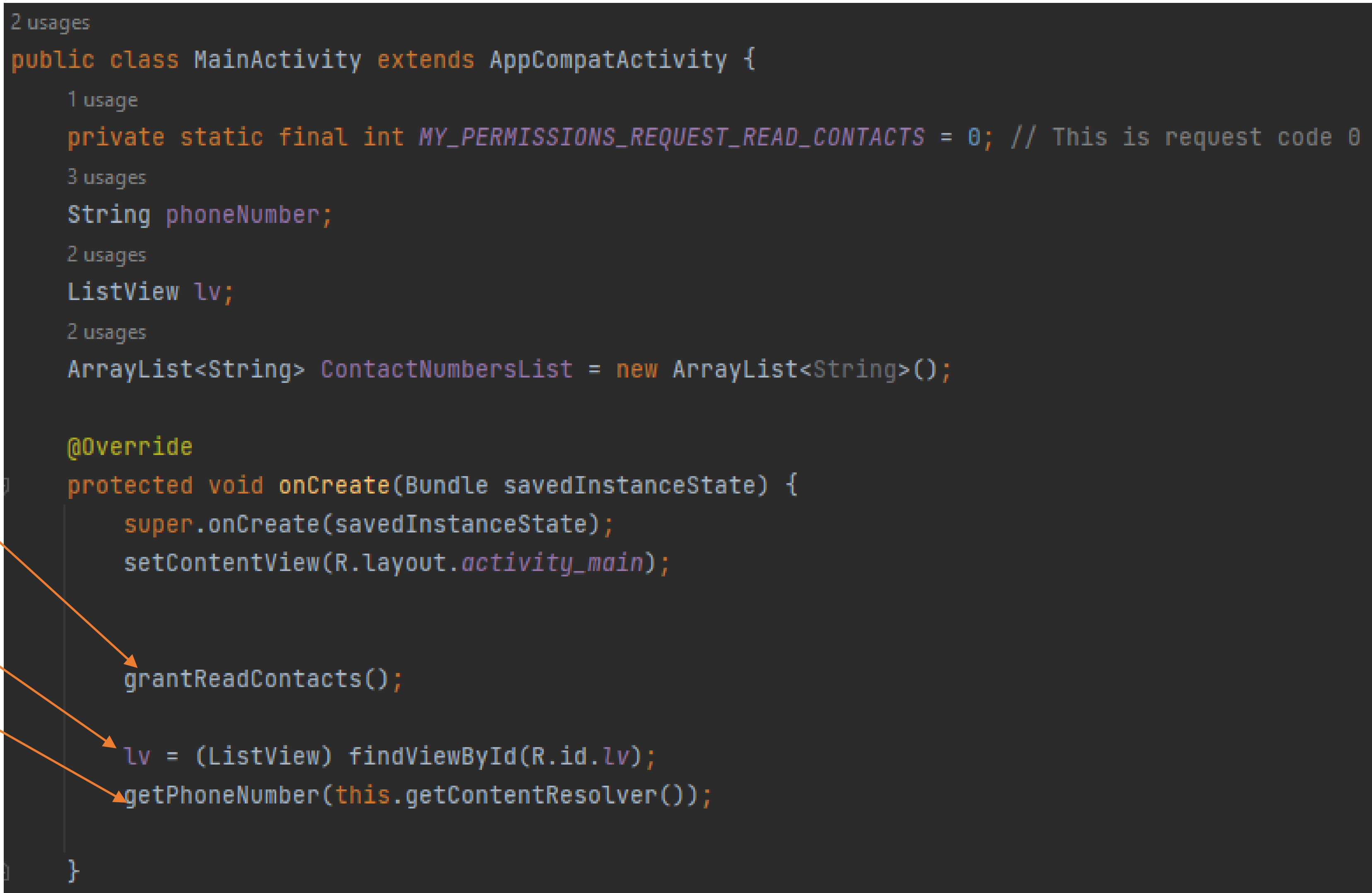


activity_main.xml MainActivity.java AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    tools:context=".MainActivity">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:id="@+id/lv"
        tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

ContactsPhoneNumber

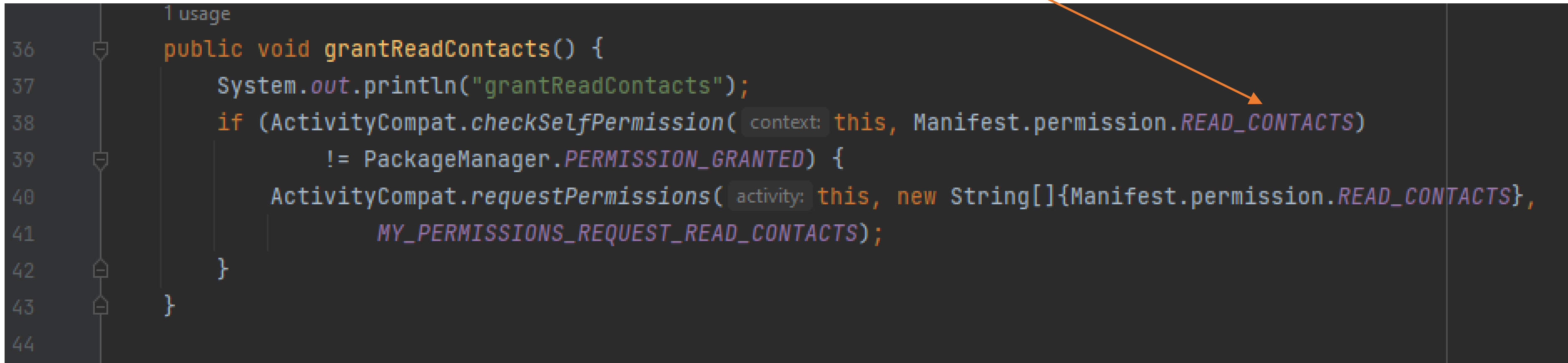
- Grant the access rights
- We instantiate the “lv”
- Then, get the phone number, and form a list



```
2 usages
public class MainActivity extends AppCompatActivity {
    1 usage
    private static final int MY_PERMISSIONS_REQUEST_READ_CONTACTS = 0; // This is request code 0
    3 usages
    String phoneNumber;
    2 usages
    ListView lv;
    2 usages
    ArrayList<String> ContactNumbersList = new ArrayList<String>();

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


        grantReadContacts();

        lv = (ListView) findViewById(R.id.lv);
        getPhoneNumber(this.getContentResolver());

    }
```

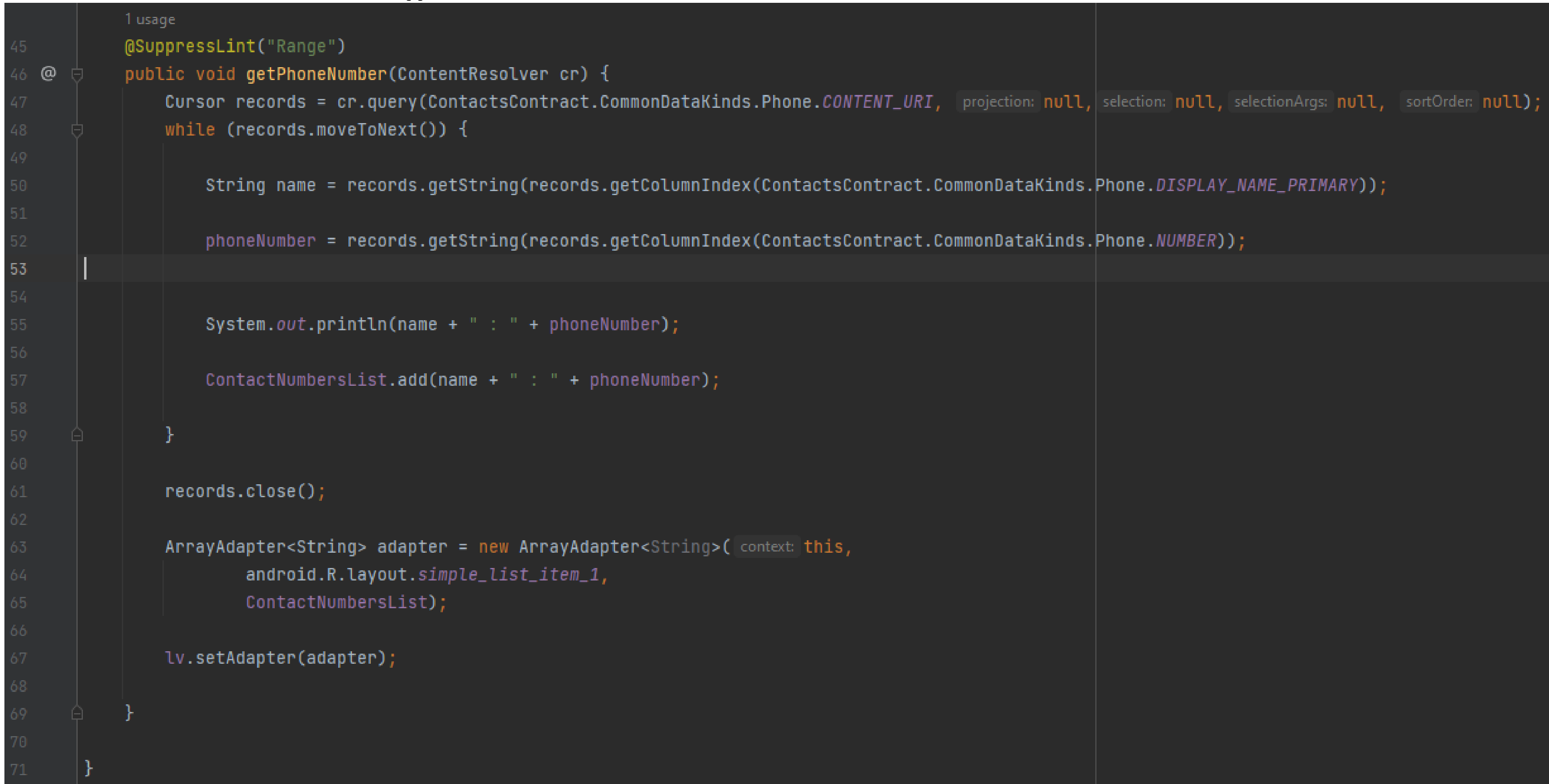
ContactsPhoneNumber

- This permission has to be the same as the one we specified in the AndroidManifest.xml
- The way to request the run-time permission from the end user is the same as we had seen earlier (This is still old style, around 2018~2022)
- (If we have enough time left, I can talk about the latest approach to request multiple access rights in one time)


```
1 usage
public void grantReadContacts() {
    System.out.println("grantReadContacts");
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.READ_CONTACTS)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.READ_CONTACTS},
                MY_PERMISSIONS_REQUEST_READ_CONTACTS);
    }
}
```

ContactsPhoneNumber

- Finally, our getPhoneNumber()…



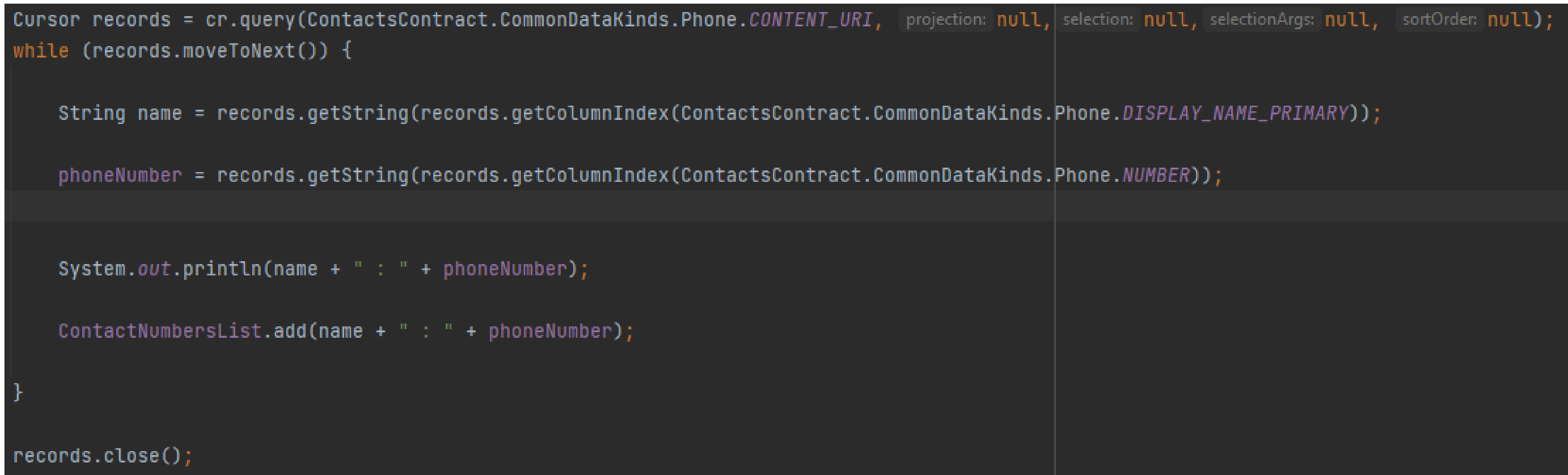
```
    1 usage
    @SuppressLint("Range")
    public void getPhoneNumber(ContentResolver cr) {
        Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
        while (records.moveToNext()) {

            String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

            phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


            System.out.println(name + " : " + phoneNumber);

            ContactNumbersList.add(name + " : " + phoneNumber);

        }

        records.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
                android.R.layout.simple_list_item_1,
                ContactNumbersList);

        lv.setAdapter(adapter);

    }

}
```

ContactsPhoneNumber

- The 1st thing, @SuppressLint(“Range”)
  - The **lint** tool is used in the Android Studio. It checks the Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.
  - If you are interested, go to check this page
    - https://developer.android.com/studio/write/lint#:~:text=The%20lint%20tool%20checks%20your,when%20you%20build%20your%20app.
  - The reason I want to do this, is…
    - I want to suppress some “warning” message during the compilation.

ContactsPhoneNumber

- Cursor? What is this? (Contacts data is a kind of internal database…)
  - When we are performing a database query, the cursor is used to access the result set which is returned from the query
  - The cursor can be seen as a pointer, which is pointing to the result set. Or, we can say it is a kind of interface.
    - https://developer.android.com/reference/android/database/Cursor

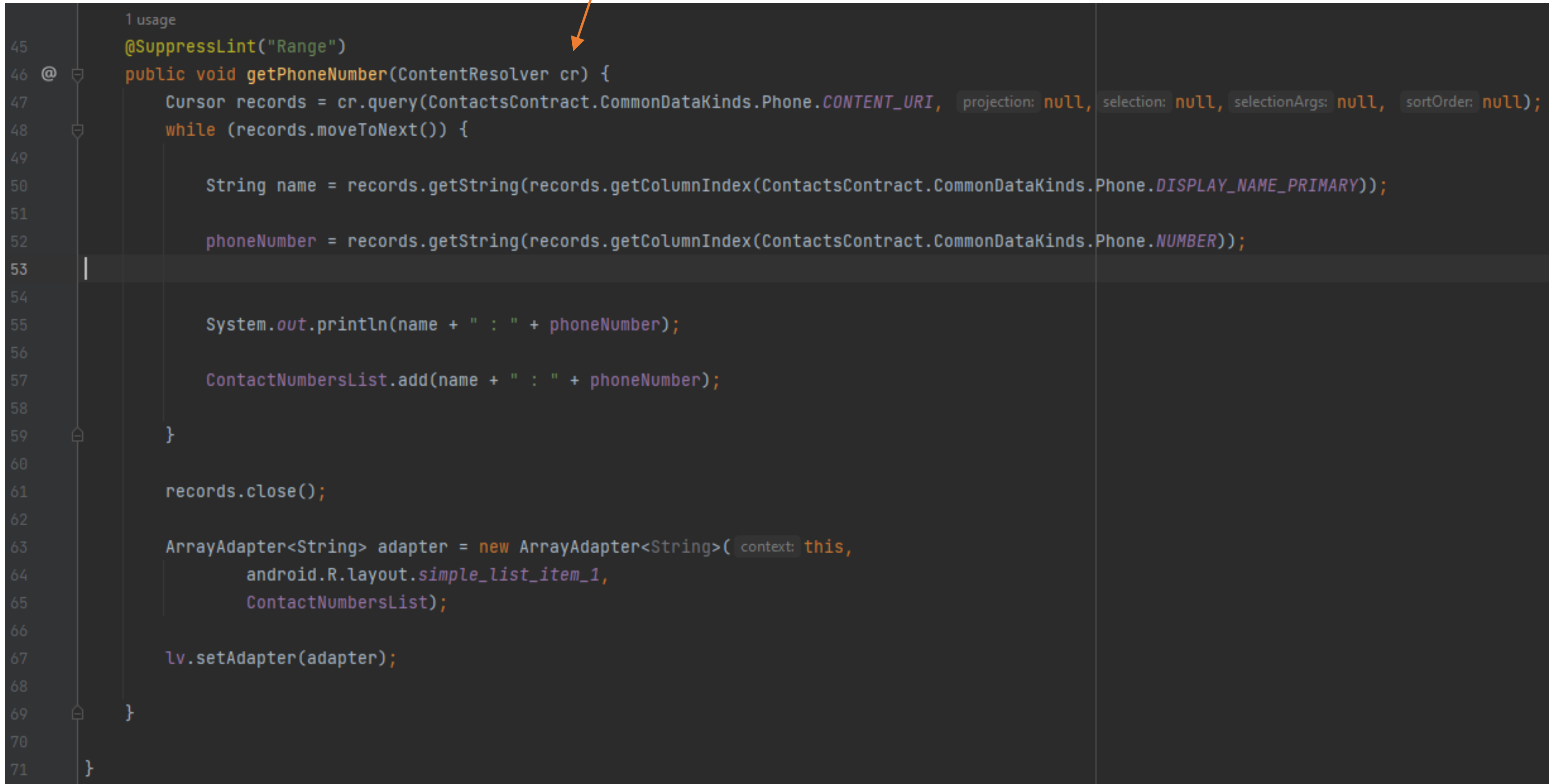


```
Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
while (records.moveToNext()) {

    String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

    phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


    System.out.println(name + " : " + phoneNumber);

    ContactNumbersList.add(name + " : " + phoneNumber);

}

records.close();
```

ContactsPhoneNumber

We use “this.getContentResolver()” in the onCreate() to get the “cr”

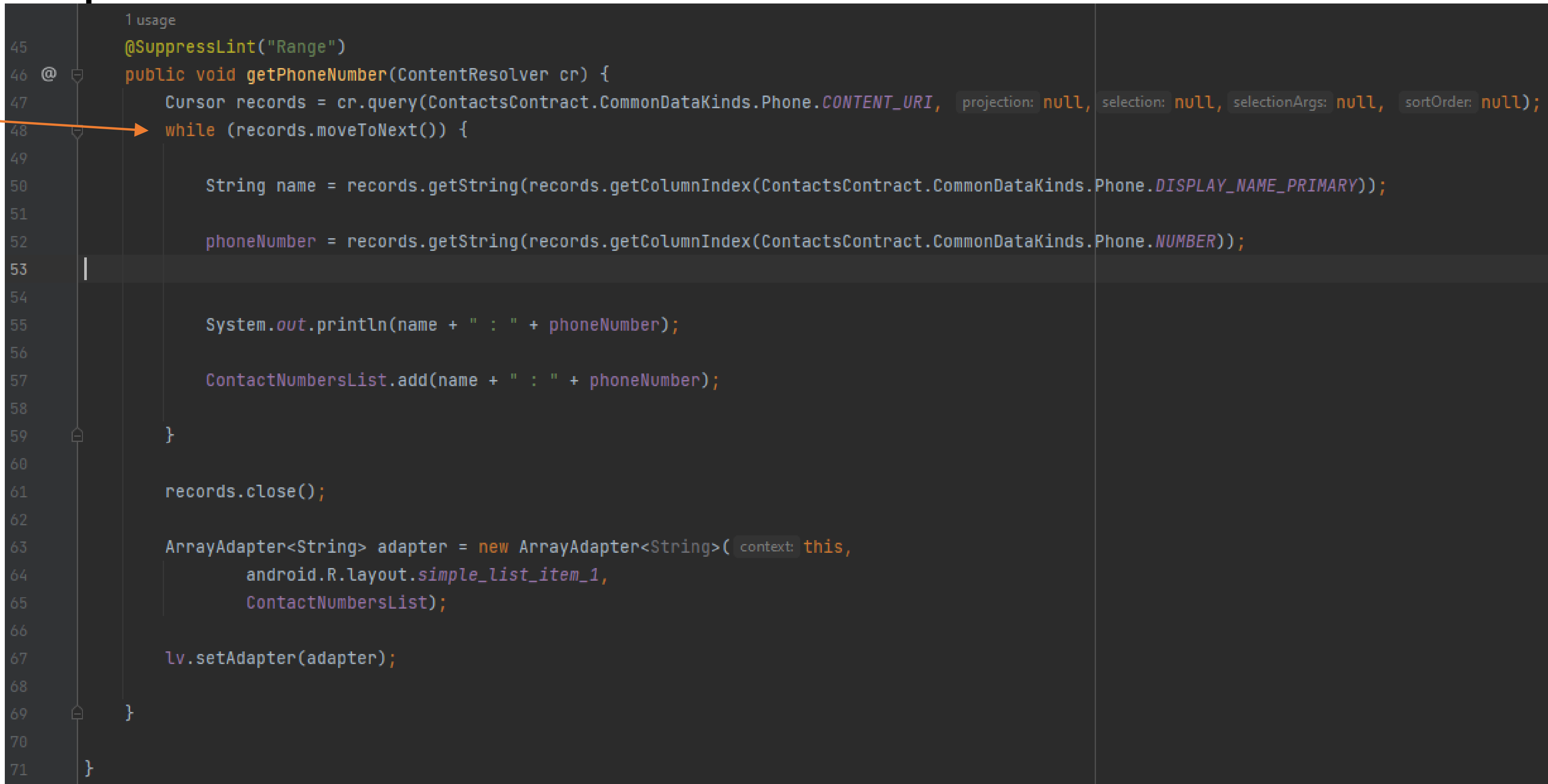
- Inside of the getPhoneNumber()
  - We need the ContentResolver (cr) to execute the query



```
    1 usage
    @SuppressLint("Range")
    public void getPhoneNumber(ContentResolver cr) {
        Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
        while (records.moveToNext()) {

            String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

            phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));



            System.out.println(name + " : " + phoneNumber);

            ContactNumbersList.add(name + " : " + phoneNumber);

        }

        records.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
                android.R.layout.simple_list_item_1,
                ContactNumbersList);

        lv.setAdapter(adapter);

    }

}
```

ContactsPhoneNumber

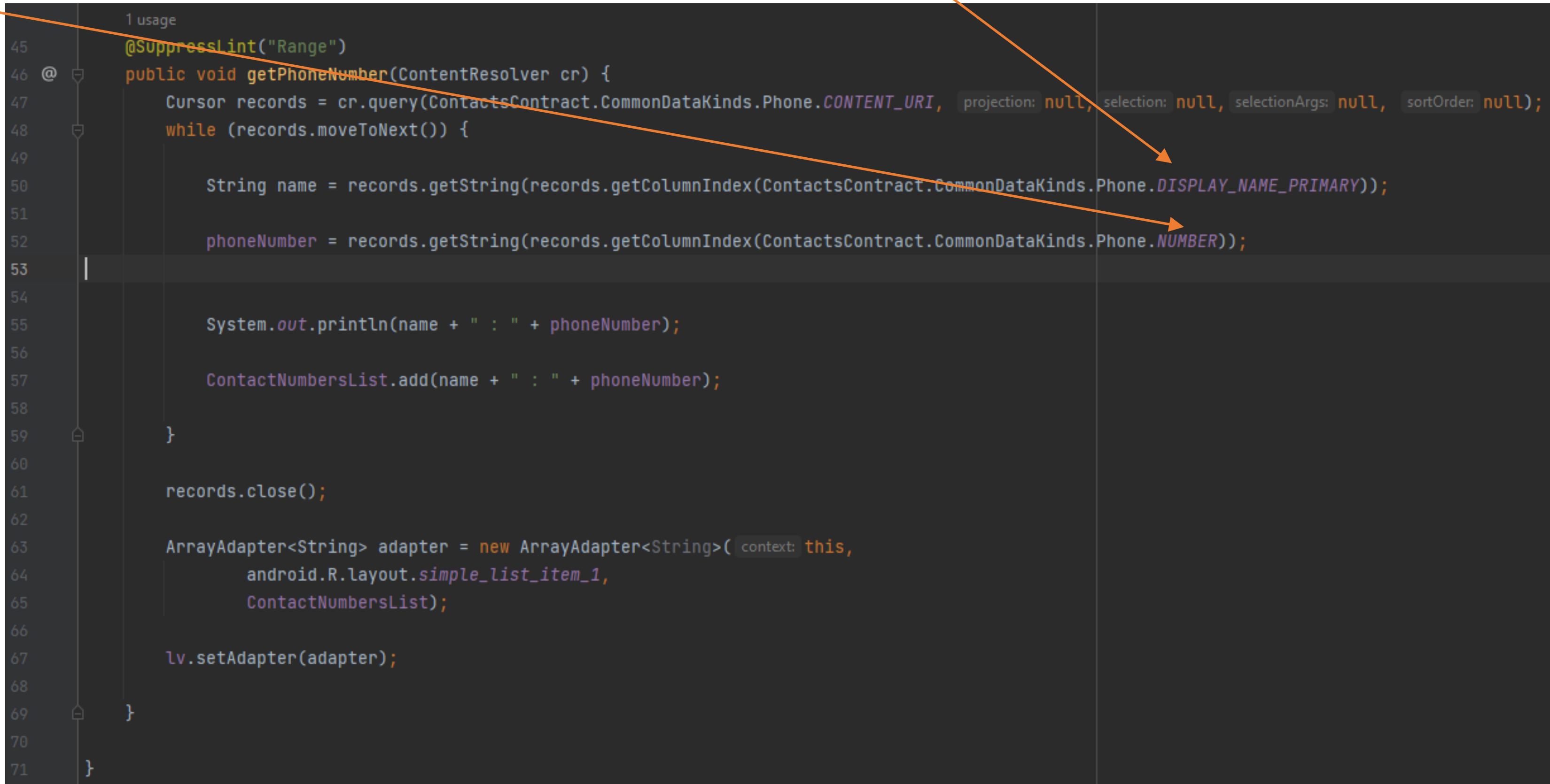
- The while loop means, I’m going to read everything to the end of the cursor “records” can point to



```
    1 usage
    @SuppressLint("Range")
    public void getPhoneNumber(ContentResolver cr) {
        Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null, selection: null, selectionArgs: null,  sortOrder: null);
        while (records.moveToNext()) {

            String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

            phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


            System.out.println(name + " : " + phoneNumber);

            ContactNumbersList.add(name + " : " + phoneNumber);

        }

        records.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
                android.R.layout.simple_list_item_1,
                ContactNumbersList);

        lv.setAdapter(adapter);

    }

}
```

ContactsPhoneNumber

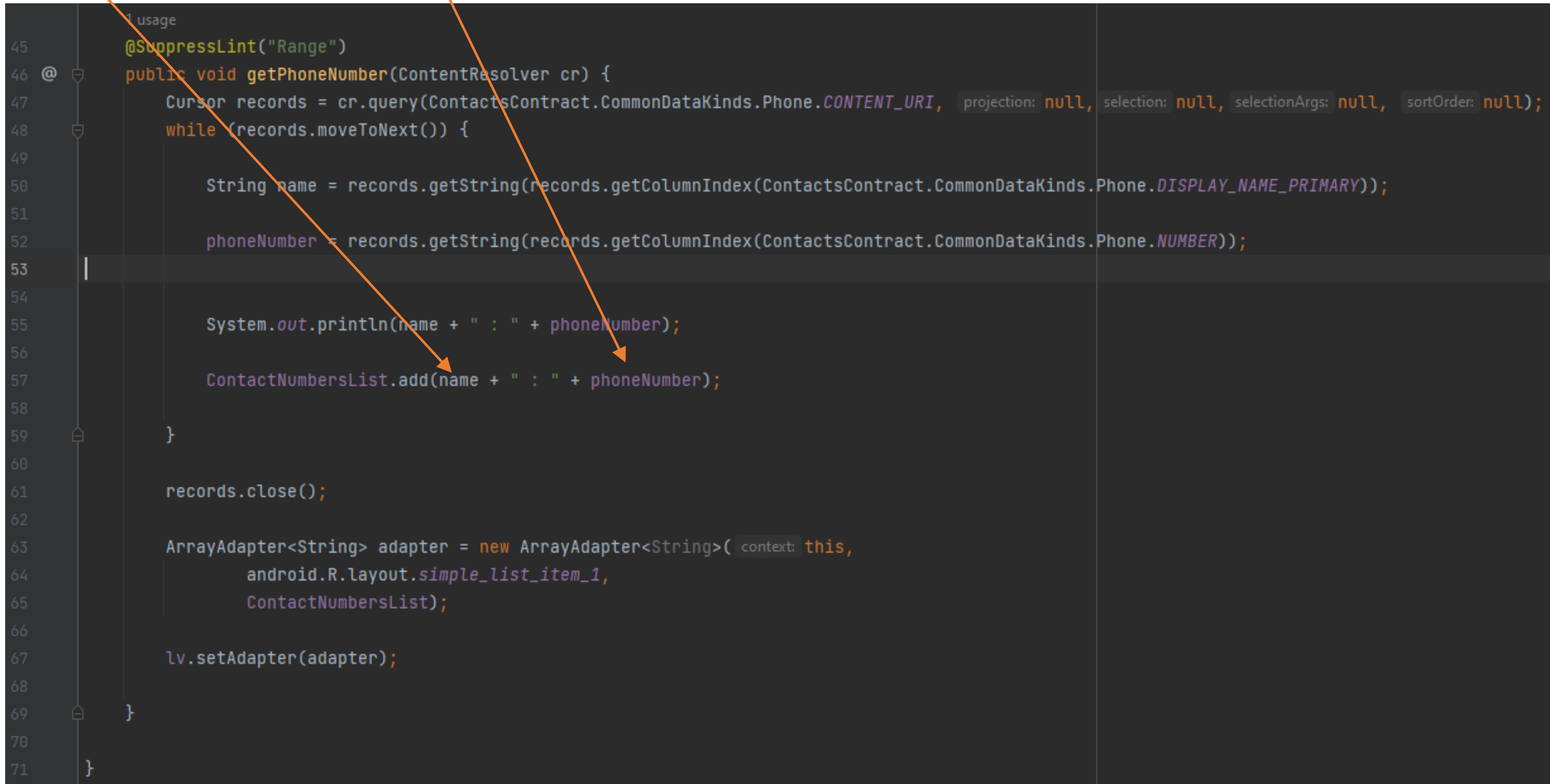
- We can use "ContactsContract" to index the "name" and the "phone number" as **indices** of **columns** we want to retrieve.



```
    1 usage
    @SuppressLint("Range")
    public void getPhoneNumber(ContentResolver cr) {
        Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
        while (records.moveToNext()) {

            String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

            phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


            System.out.println(name + " : " + phoneNumber);

            ContactNumbersList.add(name + " : " + phoneNumber);

        }

        records.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
                android.R.layout.simple_list_item_1,
                ContactNumbersList);

        lv.setAdapter(adapter);

    }

}
```

ContactsPhoneNumber

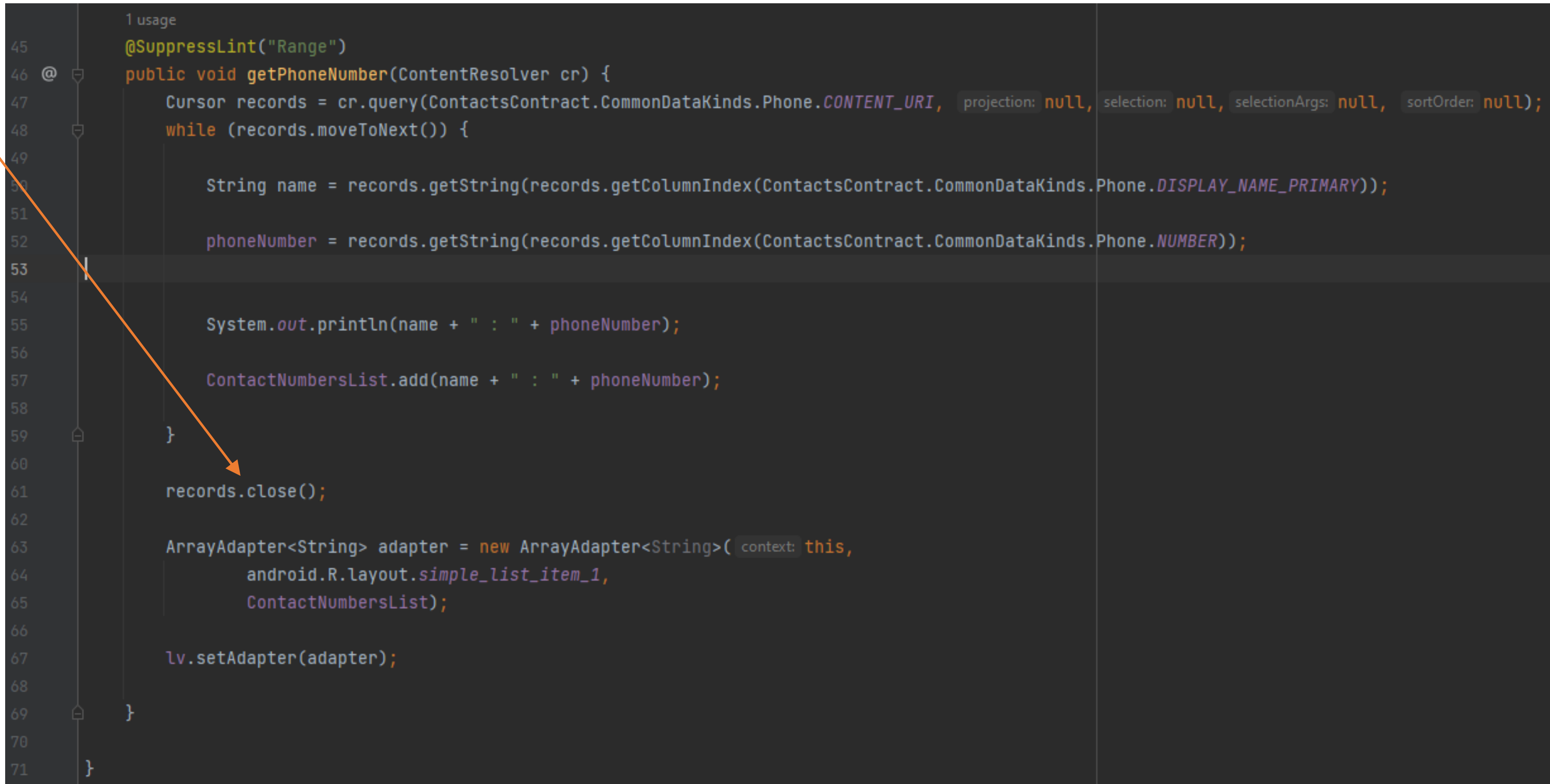
- Finally, we use the “ContactNumberList”, which is an array list of string to collect the name + phone numbers



```
    1 usage
    @SuppressLint("Range")
    public void getPhoneNumber(ContentResolver cr) {
        Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
        while (records.moveToNext()) {

            String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

            phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


            System.out.println(name + " : " + phoneNumber);

            ContactNumbersList.add(name + " : " + phoneNumber);

        }

        records.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
                android.R.layout.simple_list_item_1,
                ContactNumbersList);

        lv.setAdapter(adapter);

    }

}
```

ContactsPhoneNumber

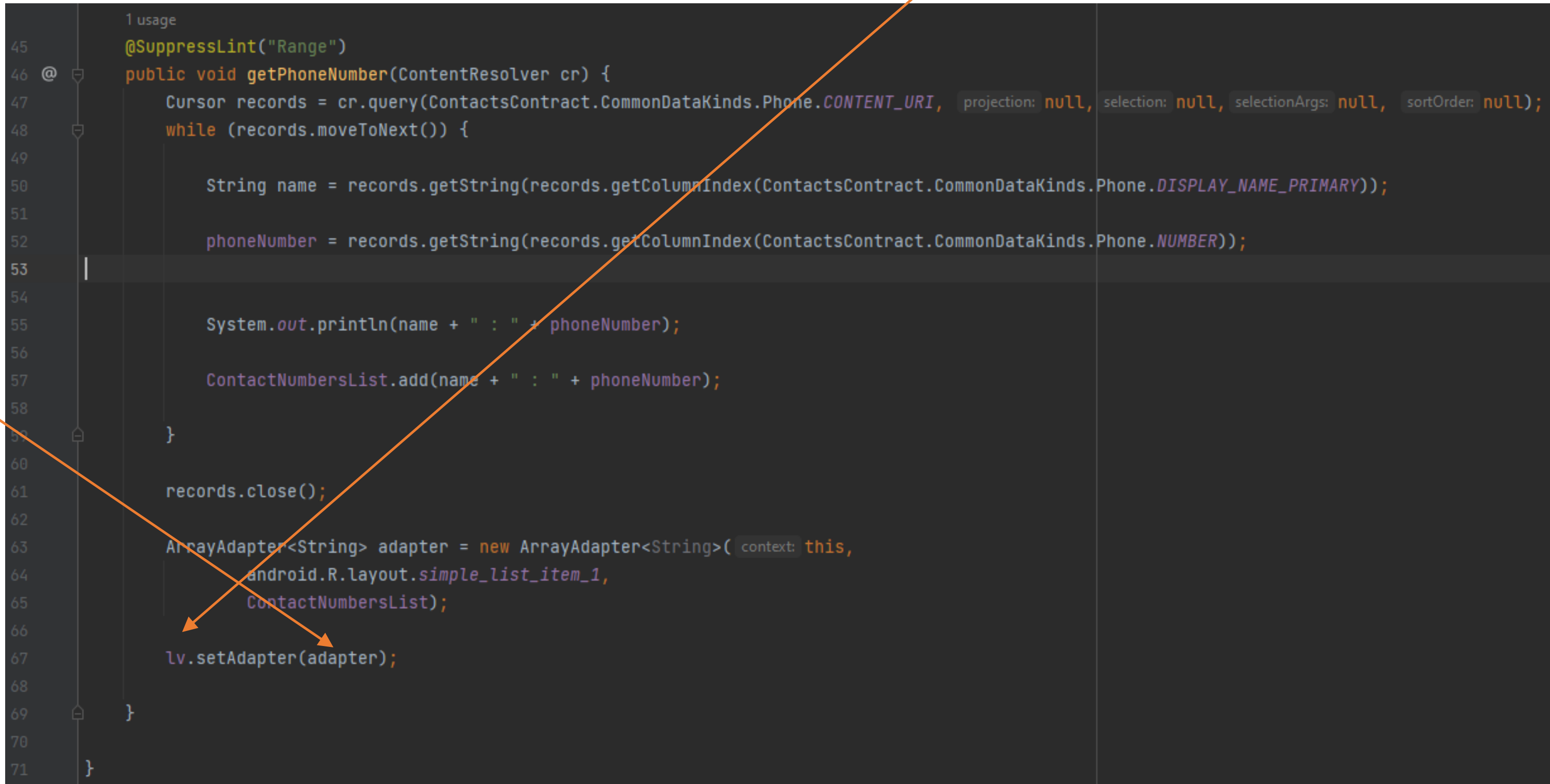
- We need to close the cursor, in case to introduce unnecessary memory leaks



```
1 usage
@SuppressLint("Range")
public void getPhoneNumber(ContentResolver cr) {
    Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, projection: null, selection: null, selectionArgs: null, sortOrder: null);
    while (records.moveToNext()) {

        String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

        phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


        System.out.println(name + " : " + phoneNumber);

        ContactNumbersList.add(name + " : " + phoneNumber);

    }

    records.close();

    ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
            android.R.layout.simple_list_item_1,
            ContactNumbersList);

    lv.setAdapter(adapter);

}

}
```

ContactsPhoneNumber

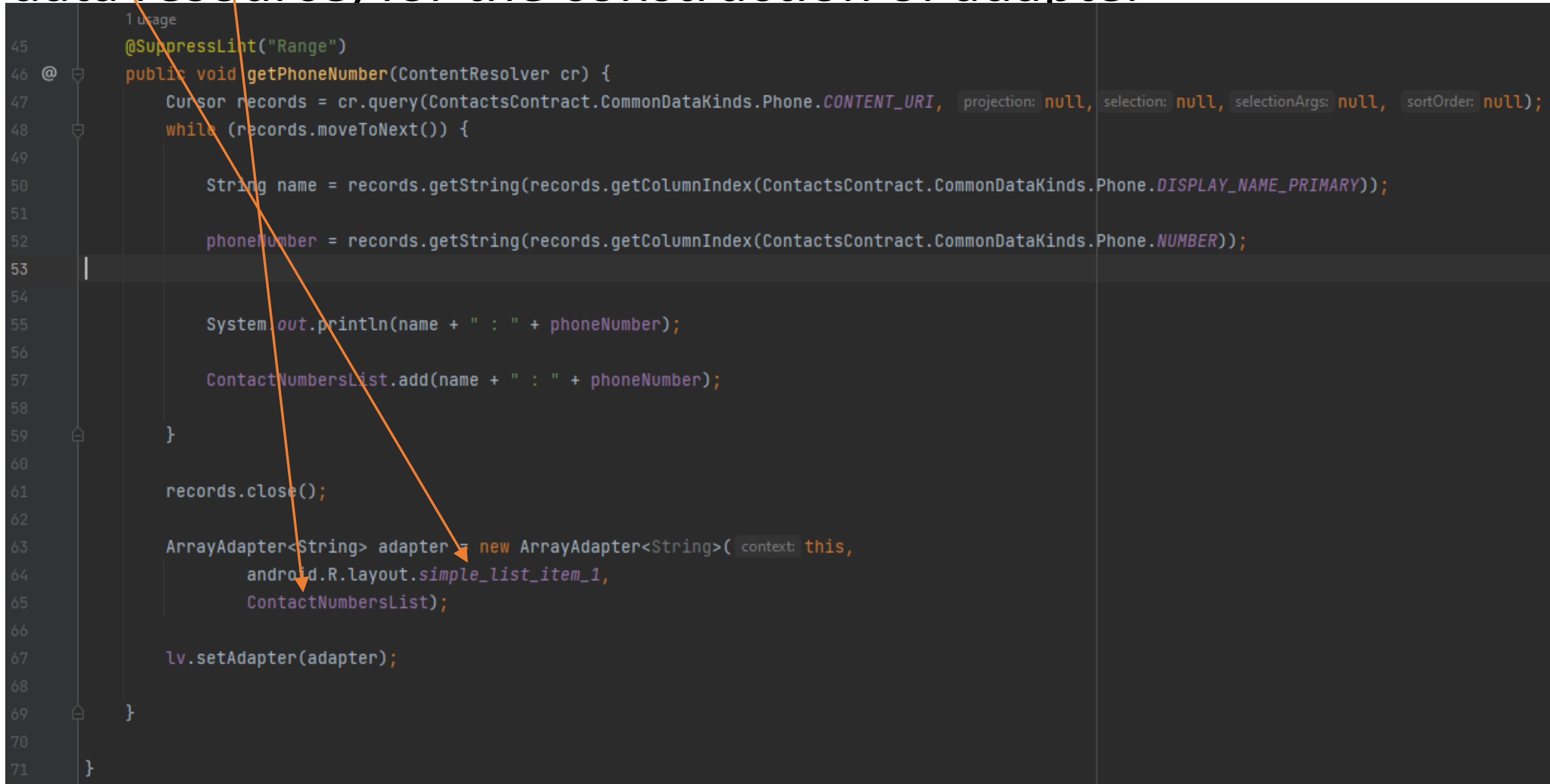
- In order to show something onto the UI, we need a ListView (lv)
- “lv” needs to reflect the changes from the query we just made.
- “lv” needs an “adapter”



```
    1 usage
    @SuppressLint("Range")
    public void getPhoneNumber(ContentResolver cr) {
        Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, projection: null, selection: null, selectionArgs: null, sortOrder: null);
        while (records.moveToNext()) {

            String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

            phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


            System.out.println(name + " : " + phoneNumber);

            ContactNumbersList.add(name + " : " + phoneNumber);

        }

        records.close();

        ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
                android.R.layout.simple_list_item_1,
                ContactNumbersList);

        lv.setAdapter(adapter);

    }

}
```

ContactsPhoneNumber

- The adapter is set by using an “ArrayAdapter” of “String”.
  - We need to specify the style
  - We need to specify the data resource, for the construction of adapter



```
1 usage
@SuppressLint("Range")
public void getPhoneNumber(ContentResolver cr) {
    Cursor records = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
    while (records.moveToNext()) {

        String name = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME_PRIMARY));

        phoneNumber = records.getString(records.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));


        System.out.println(name + " : " + phoneNumber);

        ContactNumbersList.add(name + " : " + phoneNumber);

    }

    records.close();

    ArrayAdapter<String> adapter = new ArrayAdapter<String>( context: this,
            android.R.layout.simple_list_item_1,
            ContactNumbersList);

    lv.setAdapter(adapter);

}

}
```