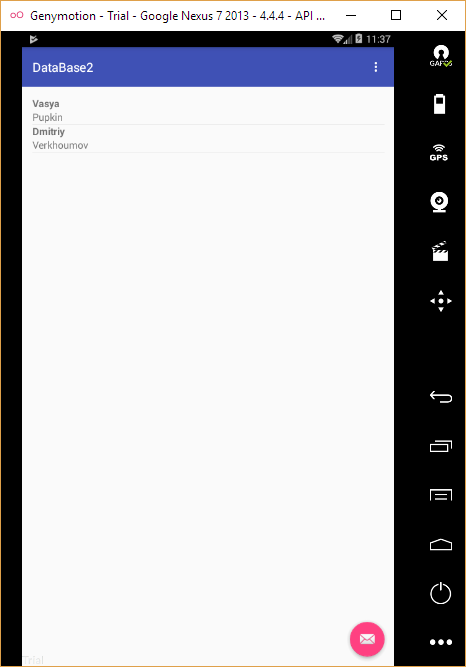
**DataBase2**

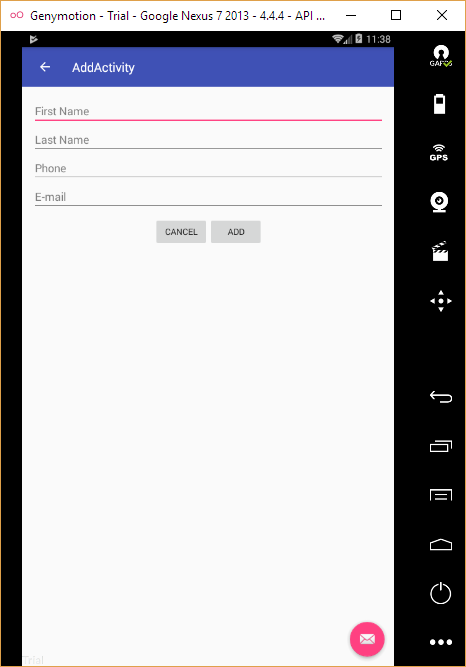
В тексте задания есть 4 ошибки. **Первая** в 17-18 пункте, где сначала создаётся пункт меню menu\_show.xml, а потом в методе onCreateOptionsMenu формируется ссылка на show\_contact, а должно быть menu\_show. **Вторая** в 5 пункте, где говорится, что надо создать DataBase.java, хотя далее везде используется DatabaseManager. **Третья** в 7 пункте, где почему-то в курсоре обращение к полям идёт по индексу (cursor.getString(N)), хотя в первой работе по базе данных обращение было с определением индекса по названию поля: cursor.getString(cursor.getColumnIndex(FIRST\_NAME\_FIELD)). **Четвёртая** ошибка в пункте 8. Там используется метод getWritableDatabase, хотя согласно логике метода надо использовать getReadableDatabase.

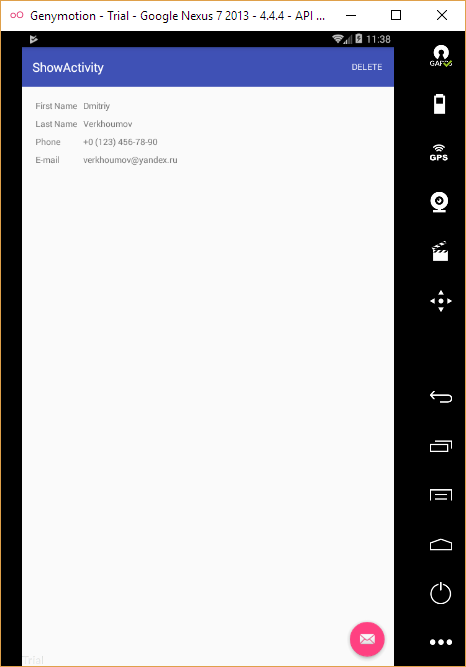
Исходный код проекта:

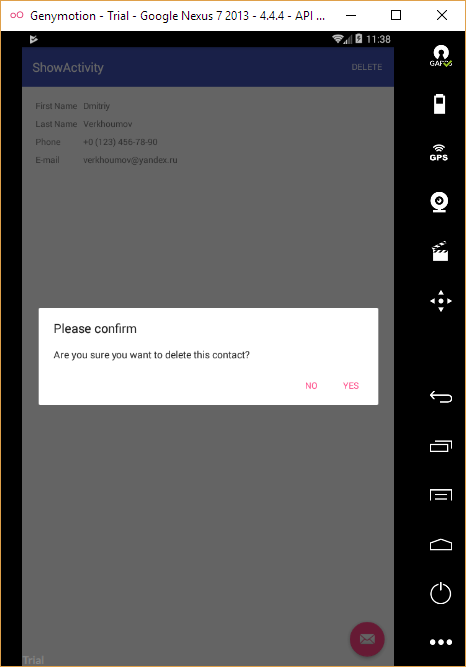
<https://github.com/verkhoumov/AndroidLabs/tree/master/DataBase2>

Скриншоты экрана:









Исходный код.

MainActivity.java:

1. **package** ru.verkhoumov.database2;
3. **import** android.content.Intent;
4. **import** android.database.Cursor;
5. **import** android.os.Bundle;
6. **import** android.support.design.widget.FloatingActionButton;
7. **import** android.support.v4.widget.CursorAdapter;
8. **import** android.support.v4.widget.SimpleCursorAdapter;
9. **import** android.support.v7.app.AppCompatActivity;
10. **import** android.support.v7.widget.Toolbar;
11. **import** android.view.View;
12. **import** android.view.Menu;
13. **import** android.view.MenuItem;
14. **import** android.widget.AdapterView;
15. **import** android.widget.ListAdapter;
16. **import** android.widget.ListView;
18. **public** **class** MainActivity **extends** AppCompatActivity {
19. DataBase dataBase;
21. @Override
22. **protected** **void** onCreate(Bundle savedInstanceState) {
23. **super**.onCreate(savedInstanceState);
24. setContentView(R.layout.activity\_main);
25. Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
26. setSupportActionBar(toolbar);
28. FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
29. fab.setOnClickListener(**new** View.OnClickListener() {
30. @Override
31. **public** **void** onClick(View view) {
32. startActivity(**new** Intent(MainActivity.**this**, AddActivity.**class**));
33. }
34. });
36. *// Ссылка на ListView.*
37. ListView listView = (ListView) findViewById(R.id.listView);
39. dataBase = **new** DataBase(**this**);
41. Cursor cursor = dataBase.getContactsCursor();
42. startManagingCursor(cursor);
44. ListAdapter adapter = **new** SimpleCursorAdapter(
45. **this**,
46. android.R.layout.two\_line\_list\_item, *// Список в две строки.*
47. cursor,
48. **new** String[] {
49. DataBase.FIRST\_NAME\_FIELD, *// Имя в первой строке.*
50. DataBase.LAST\_NAME\_FIELD *// Фамилия во второй строке.*
51. },
52. **new** **int**[] {
53. android.R.id.text1,
54. android.R.id.text2,
55. },
56. CursorAdapter.FLAG\_REGISTER\_CONTENT\_OBSERVER
57. );
59. listView.setAdapter(adapter);
60. listView.setChoiceMode(ListView.CHOICE\_MODE\_SINGLE);
61. listView.setOnItemClickListener(
62. **new** AdapterView.OnItemClickListener() {
63. @Override
64. **public** **void** onItemClick(AdapterView<?> adapterView, View view, **int** position, **long** id) {
65. *// Генерируем новую активность и запускаем её.*
66. Intent intent = **new** Intent(getApplicationContext(), ShowActivity.**class**);
67. intent.putExtra("id", id);
68. startActivity(intent);
69. }
70. }
71. );
72. }
74. @Override
75. **public** **boolean** onCreateOptionsMenu(Menu menu) {
76. *// Inflate the menu; this adds items to the action bar if it is present.*
77. getMenuInflater().inflate(R.menu.menu\_main, menu);
78. **return** **true**;
79. }
81. @Override
82. **public** **boolean** onOptionsItemSelected(MenuItem item) {
83. *// Handle action bar item clicks here. The action bar will*
84. *// automatically handle clicks on the Home/Up button, so long*
85. *// as you specify a parent activity in AndroidManifest.xml.*
86. **int** id = item.getItemId();
88. *//noinspection SimplifiableIfStatement*
89. **if** (id == R.id.action\_settings) {
90. **return** **true**;
91. }
93. **return** **super**.onOptionsItemSelected(item);
94. }
95. }

ShowActivity.java:

1. **package** ru.verkhoumov.database2;
3. **import** android.content.DialogInterface;
4. **import** android.os.Bundle;
5. **import** android.support.design.widget.FloatingActionButton;
6. **import** android.support.design.widget.Snackbar;
7. **import** android.support.v7.app.AlertDialog;
8. **import** android.support.v7.app.AppCompatActivity;
9. **import** android.support.v7.widget.Toolbar;
10. **import** android.util.Log;
11. **import** android.view.Menu;
12. **import** android.view.MenuItem;
13. **import** android.view.View;
14. **import** android.widget.TextView;
16. **public** **class** ShowActivity **extends** AppCompatActivity {
17. **long** contactId;
19. @Override
20. **protected** **void** onCreate(Bundle savedInstanceState) {
21. **super**.onCreate(savedInstanceState);
22. setContentView(R.layout.activity\_show);
23. Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
24. setSupportActionBar(toolbar);
26. Bundle extras = getIntent().getExtras();
28. **if** (extras != **null**) {
29. *// Достаём ID контакта, переданный через Intent.*
30. contactId = extras.getLong("id");
32. *// Соединение с базой данных.*
33. DataBase dataBase = **new** DataBase(**this**);
35. *// Получаем информацию и контакте.*
36. Contact contact = dataBase.getContact(contactId);
38. *// Если контакт найден, подставляем данные в таблицу.*
39. **if** (contact != **null**) {
40. ((TextView) findViewById(R.id.first\_name)).setText(contact.getFirstName());
41. ((TextView) findViewById(R.id.last\_name)).setText(contact.getLastName());
42. ((TextView) findViewById(R.id.phone)).setText(contact.getPhone());
43. ((TextView) findViewById(R.id.email)).setText(contact.getEmail());
44. } **else** {
45. Log.d("db", "Contact undefined");
46. }
47. }
48. }
50. @Override
51. **public** **boolean** onCreateOptionsMenu(Menu menu) {
52. getMenuInflater().inflate(R.menu.menu\_show, menu);
54. **return** **true**;
55. }
57. @Override
58. **public** **boolean** onOptionsItemSelected(MenuItem item) {
59. **switch** (item.getItemId()) {
60. **case** R.id.action\_delete:
61. deleteContact();
62. **return** **true**;
63. **default**:
64. **return** **super**.onOptionsItemSelected(item);
65. }
66. }
68. **private** **void** deleteContact() {
69. **new** AlertDialog
70. .Builder(**this**)
71. .setTitle("Please confirm")
72. .setMessage("Are you sure you want to delete this contact?")
73. .setPositiveButton("Yes", **new** DialogInterface.OnClickListener() {
74. **public** **void** onClick(DialogInterface dialog, **int** whichButton) {
75. DataBase dataBase = **new** DataBase(getApplicationContext());
76. dataBase.deleteContact(contactId);
77. dialog.dismiss();
78. finish();
79. }
80. })
81. .setNegativeButton("No", **new** DialogInterface.OnClickListener() {
82. **public** **void** onClick(DialogInterface dialog, **int** whichButton) {
83. dialog.dismiss();
84. }
85. })
86. .create()
87. .show();
88. }
89. }

AddActivity.java:

1. **package** ru.verkhoumov.database2;
3. **import** android.os.Bundle;
4. **import** android.support.design.widget.FloatingActionButton;
5. **import** android.support.design.widget.Snackbar;
6. **import** android.support.v7.app.AppCompatActivity;
7. **import** android.support.v7.widget.Toolbar;
8. **import** android.view.View;
9. **import** android.widget.TextView;
11. **public** **class** AddActivity **extends** AppCompatActivity {
13. @Override
14. **protected** **void** onCreate(Bundle savedInstanceState) {
15. **super**.onCreate(savedInstanceState);
16. setContentView(R.layout.activity\_add);
17. Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
18. setSupportActionBar(toolbar);
20. FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
21. fab.setOnClickListener(**new** View.OnClickListener() {
22. @Override
23. **public** **void** onClick(View view) {
24. Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG)
25. .setAction("Action", **null**).show();
26. }
27. });
28. getSupportActionBar().setDisplayHomeAsUpEnabled(**true**);
29. }
31. **public** **void** cancel(View view) {
32. finish();
33. }
35. **public** **void** addContact(View view) {
36. *// Ссылки на поля формы.*
37. TextView firstName = (TextView) findViewById(R.id.first\_name);
38. TextView lastName = (TextView) findViewById(R.id.last\_name);
39. TextView phone = (TextView) findViewById(R.id.phone);
40. TextView email = (TextView) findViewById(R.id.email);
42. *// Содержимое полей.*
43. String \_firstName = firstName.getText().toString();
44. String \_lastName = lastName.getText().toString();
45. String \_phone = phone.getText().toString();
46. String \_email = email.getText().toString();
48. *// Новый контакт.*
49. Contact contact = **new** Contact(\_firstName, \_lastName, \_phone, \_email);
51. *// Добавляем контакт в базу данных.*
52. DataBase dataBase = **new** DataBase(**this**);
53. dataBase.addContact(contact);
55. finish();
56. }
57. }

Contact.java:

1. **package** ru.verkhoumov.database2;
3. **public** **class** Contact {
4. **private** **long** id;
5. **private** String firstName;
6. **private** String lastName;
7. **private** String phone;
8. **private** String email;
10. **public** Contact(String firstName, String lastName, String phone, String email) {
11. **this**.firstName = firstName;
12. **this**.lastName = lastName;
13. **this**.phone = phone;
14. **this**.email = email;
15. }
17. **public** **long** getId() {
18. **return** id;
19. }
21. **public** **void** setId(**long** id) {
22. **this**.id = id;
23. }
25. **public** String getFirstName() {
26. **return** firstName;
27. }
29. **public** **void** setFirstName(String firstName) {
30. **this**.firstName = firstName;
31. }
33. **public** String getLastName() {
34. **return** lastName;
35. }
37. **public** **void** setLastName(String lastName) {
38. **this**.lastName = lastName;
39. }
41. **public** String getPhone() {
42. **return** phone;
43. }
45. **public** **void** setPhone(String phone) {
46. **this**.phone = phone;
47. }
49. **public** String getEmail() {
50. **return** email;
51. }
53. **public** **void** setEmail(String email) {
54. **this**.email = email;
55. }
56. }

DataBase.java:

1. **package** ru.verkhoumov.database2;
3. **import** android.content.ContentValues;
4. **import** android.content.Context;
5. **import** android.database.Cursor;
6. **import** android.database.sqlite.SQLiteDatabase;
7. **import** android.database.sqlite.SQLiteOpenHelper;
8. **import** android.util.Log;
10. **public** **class** DataBase **extends** SQLiteOpenHelper {
11. **public** **static** **final** String TABLE\_NAME = "contacts\_v2";
12. **public** **static** **final** String ID\_FIELD = "\_id";
13. **public** **static** **final** String FIRST\_NAME\_FIELD = "first\_name";
14. **public** **static** **final** String LAST\_NAME\_FIELD = "last\_name";
15. **public** **static** **final** String PHONE\_FIELD = "phone";
16. **public** **static** **final** String EMAIL\_FIELD = "email";
18. **public** DataBase(Context context) {
19. **super**(context, "contacts\_db", **null**, 1);
20. }
22. @Override
23. **public** **void** onCreate(SQLiteDatabase sqLiteDatabase) {
24. Log.d("db", "onCreate");
26. String sql = "CREATE TABLE IF NOT EXISTS `" + TABLE\_NAME + "`"
27. + " (`" + ID\_FIELD + "` INTEGER, `"
28. + FIRST\_NAME\_FIELD + "` TEXT, `"
29. + LAST\_NAME\_FIELD + "` TEXT, `"
30. + PHONE\_FIELD + "` TEXT, `"
31. + EMAIL\_FIELD + "` TEXT, PRIMARY KEY (`"
32. + ID\_FIELD + "`));";
33. sqLiteDatabase.execSQL(sql);
34. }
36. @Override
37. **public** **void** onUpgrade(SQLiteDatabase sqLiteDatabase, **int** i, **int** i1) {
38. Log.d("db", "onUpgrade");
40. *// Пересоздание таблицы.*
41. sqLiteDatabase.execSQL("DROP TABLE IF EXISTS `" + TABLE\_NAME + "`");
42. onCreate(sqLiteDatabase);
43. }
45. **public** Contact addContact(Contact contact) {
46. Log.d("db", "addContacts");
48. *// Формируем новую запись для последующей вставки в базу данных.*
49. ContentValues row = **new** ContentValues();
50. row.put(FIRST\_NAME\_FIELD, contact.getFirstName());
51. row.put(LAST\_NAME\_FIELD, contact.getLastName());
52. row.put(PHONE\_FIELD, contact.getPhone());
53. row.put(EMAIL\_FIELD, contact.getEmail());
55. *// Добавляем новую запись в базу данных.*
56. SQLiteDatabase sqLiteDatabase = **this**.getWritableDatabase();
57. **long** id = sqLiteDatabase.insert(TABLE\_NAME, **null**, row);
59. *// Закрываем текущее соединение.*
60. sqLiteDatabase.close();
62. contact.setId(id);
64. **return** contact;
65. }
67. **public** Contact getContact(**long** id) {
68. Log.d("db", "getContact");
70. SQLiteDatabase sqLiteDatabase = **this**.getReadableDatabase();
72. Cursor cursor = sqLiteDatabase.query(
73. TABLE\_NAME,
74. **new** String[] { ID\_FIELD, FIRST\_NAME\_FIELD, LAST\_NAME\_FIELD, PHONE\_FIELD, EMAIL\_FIELD },
75. "`" + ID\_FIELD + "`" + " = ?",
76. **new** String[] { String.valueOf(id) },
77. **null**, **null**, **null**, **null**
78. );
80. **if** (cursor != **null**) {
81. cursor.moveToFirst();
83. *// В коде задания были указаны сразу индексы полей. Почему?*
84. Contact contact = **new** Contact(
85. cursor.getString(cursor.getColumnIndex(FIRST\_NAME\_FIELD)),
86. cursor.getString(cursor.getColumnIndex(LAST\_NAME\_FIELD)),
87. cursor.getString(cursor.getColumnIndex(PHONE\_FIELD)),
88. cursor.getString(cursor.getColumnIndex(EMAIL\_FIELD))
89. );
91. *// ID записи.*
92. contact.setId(cursor.getLong(cursor.getColumnIndex(ID\_FIELD)));
94. **return** contact;
95. }
97. **return** **null**;
98. }
100. **public** Cursor getContactsCursor() {
101. Log.d("db", "getContactsCursor");
103. *// В коде задания берётся getWritableDatabase, хотя необходимо только чтение данных.*
104. String sql = "SELECT \* FROM `" + TABLE\_NAME + "`";
105. SQLiteDatabase sqLiteDatabase = **this**.getReadableDatabase();
107. **return** sqLiteDatabase.rawQuery(sql, **null**);
108. }
110. **public** **void** deleteContact(**long** id) {
111. Log.d("db", "deleteContact");
113. *// Удаляем запись из базы данных.*
114. SQLiteDatabase sqLiteDatabase = **this**.getWritableDatabase();
115. sqLiteDatabase.delete(
116. TABLE\_NAME,
117. "`" + ID\_FIELD + "`" + " = ?",
118. **new** String[] { String.valueOf(id) }
119. );
121. *// Закрываем текущее соединение.*
122. sqLiteDatabase.close();
123. }
124. }

content\_main.xml:

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<RelativeLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:id="@+id/content\_main"
6. android:layout\_width="match\_parent"
7. android:layout\_height="match\_parent"
8. android:paddingBottom="@dimen/activity\_vertical\_margin"
9. android:paddingLeft="@dimen/activity\_horizontal\_margin"
10. android:paddingRight="@dimen/activity\_horizontal\_margin"
11. android:paddingTop="@dimen/activity\_vertical\_margin"
12. app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"
13. tools:context="ru.verkhoumov.database2.MainActivity"
14. tools:showIn="@layout/activity\_main"**>**
16. **<ListView**
17. android:id="@+id/listView"
18. android:layout\_width="match\_parent"
19. android:layout\_height="match\_parent" **/>**
20. **</RelativeLayout>**

content\_add.xml:

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<RelativeLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. android:paddingBottom="@dimen/activity\_vertical\_margin"
8. android:paddingLeft="@dimen/activity\_horizontal\_margin"
9. android:paddingRight="@dimen/activity\_horizontal\_margin"
10. android:paddingTop="@dimen/activity\_vertical\_margin"
11. app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"
12. tools:context="ru.verkhoumov.database2.AddActivity"
13. tools:showIn="@layout/activity\_add"
14. android:id="@+id/content\_add"**>**
16. **<EditText**
17. android:layout\_width="match\_parent"
18. android:layout\_height="wrap\_content"
19. android:id="@+id/first\_name"
20. android:hint="@string/first\_name" **/>**
22. **<EditText**
23. android:layout\_width="match\_parent"
24. android:layout\_height="wrap\_content"
25. android:id="@+id/last\_name"
26. android:hint="@string/last\_name"
27. android:layout\_below="@+id/first\_name" **/>**
29. **<EditText**
30. android:layout\_width="match\_parent"
31. android:layout\_height="wrap\_content"
32. android:id="@+id/phone"
33. android:hint="@string/phone"
34. android:layout\_below="@+id/last\_name" **/>**
36. **<EditText**
37. android:layout\_width="match\_parent"
38. android:layout\_height="wrap\_content"
39. android:id="@+id/email"
40. android:hint="@string/email"
41. android:layout\_below="@+id/phone" **/>**
43. **<LinearLayout**
44. android:layout\_width="match\_parent"
45. android:layout\_height="match\_parent"
46. android:gravity="center\_horizontal"
47. android:layout\_below="@+id/email"
48. android:layout\_marginTop="10dp"**>**
50. **<Button**
51. android:id="@+id/cancelBtn"
52. android:layout\_width="wrap\_content"
53. android:layout\_height="wrap\_content"
54. android:text="@string/cancel\_button\_text"
55. android:onClick="cancel" **/>**
57. **<Button**
58. android:id="@+id/addBtn"
59. android:layout\_width="wrap\_content"
60. android:layout\_height="wrap\_content"
61. android:text="@string/add\_button\_text"
62. android:onClick="addContact" **/>**
63. **</LinearLayout>**
64. **</RelativeLayout>**

content\_show.xml:

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<TableLayout** xmlns:android="http://schemas.android.com/apk/res/android"
3. xmlns:app="http://schemas.android.com/apk/res-auto"
4. xmlns:tools="http://schemas.android.com/tools"
5. android:layout\_width="match\_parent"
6. android:layout\_height="match\_parent"
7. android:paddingBottom="@dimen/activity\_vertical\_margin"
8. android:paddingLeft="@dimen/activity\_horizontal\_margin"
9. android:paddingRight="@dimen/activity\_horizontal\_margin"
10. android:paddingTop="@dimen/activity\_vertical\_margin"
11. app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"
12. tools:context="ru.verkhoumov.database2.ShowActivity"
13. tools:showIn="@layout/activity\_show"**>**
15. **<TableRow**
16. android:layout\_width="match\_parent"
17. android:layout\_height="match\_parent"
18. android:padding="5dp"**>**
20. **<TextView**
21. android:text="@string/first\_name"
22. android:paddingEnd="10dp" **/>**
24. **<TextView**
25. android:id="@+id/first\_name" **/>**
26. **</TableRow>**
28. **<TableRow**
29. android:layout\_width="match\_parent"
30. android:layout\_height="match\_parent"
31. android:padding="5dp"**>**
33. **<TextView**
34. android:text="@string/last\_name"
35. android:paddingEnd="10dp" **/>**
37. **<TextView**
38. android:id="@+id/last\_name" **/>**
39. **</TableRow>**
41. **<TableRow**
42. android:layout\_width="match\_parent"
43. android:layout\_height="match\_parent"
44. android:padding="5dp"**>**
46. **<TextView**
47. android:text="@string/phone"
48. android:paddingEnd="10dp" **/>**
50. **<TextView**
51. android:id="@+id/phone" **/>**
52. **</TableRow>**
54. **<TableRow**
55. android:layout\_width="match\_parent"
56. android:layout\_height="match\_parent"
57. android:padding="5dp"**>**
59. **<TextView**
60. android:text="@string/email"
61. android:paddingEnd="10dp" **/>**
63. **<TextView**
64. android:id="@+id/email" **/>**
65. **</TableRow>**
66. **</TableLayout>**