

# Session 4 - Exercises

V1.0

JUAN RONDON



## Table of Contents

Android Note taking application.....	2
Intents (Connecting your activities) .....	2

## Android Note taking application

### Intents (Connecting your activities)

In this lab, you will be creating the core functionality for the notes App. First you will be connecting all the activities together (NotesList, EditNote, AddNote) and then you will add the main functionality to each of the activities.

1. Open **NotesList** java file and **remove** all the hardcoded note objects that you created from last Lab. Also **remove** the Toast message for the Listener of each note. Your file should look like the following code.

```
package androidcourse.notes;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;

import java.util.ArrayList;

import androidcourse.notes.Adapters.NotesAdapter;
import androidcourse.notes.Models.Note;


public class NotesList extends AppCompatActivity {
    private ArrayList<Note> notes = new ArrayList<Note>();

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

        final ListView notesListView = (ListView) findViewById(R.id.listView);
        NotesAdapter adapter = new NotesAdapter(this, R.layout.note_row, notes);
        notesListView.setAdapter(adapter);

        notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

            }
        });
    }
}
```

Now you will be creating another listener for the **onClick** action of the new note image 

This time when you click on the image you want to jump to another Activity (**AddNote Activity**), in order to do that you will need to create an Intent and then call the method **startActivityForResult()**. This method will start the activity and it will be expecting a response/result from the started activity. Once we create the new note we need to return to NotesList activity and refresh the List of notes.

The code for the listener will look like this:

```
//listener for add NoteImage
ImageView addNoteImg = (ImageView) findViewById(R.id.addNoteImg);
addNoteImg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //create an intent
        Intent addNoteIntent = new Intent(getApplicationContext(), AddNote.class);
        startActivityForResult(addNoteIntent, ADD_NOTE_REQUEST);
    }
});
```

ADD\_NOTE\_REQUEST is a constant to help Android differentiate one Intent from another (You could have multiple Intents starting from NotesList). Create that constant after your ArrayList of Notes:

```
public class NotesList extends AppCompatActivity {
    private ArrayList<Note> notes = new ArrayList<Note>();
    private static final int ADD_NOTE_REQUEST = 1;
}
```

## 2. Completed code for NotesList:

```
package androidcourse.notes;


import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import java.util.ArrayList;
import androidcourse.notes.Adapters.NotesAdapter;
import androidcourse.notes.Models.Note;

public class NotesList extends AppCompatActivity {
    private ArrayList<Note> notes = new ArrayList<Note>();
    private static final int ADD_NOTE_REQUEST = 1;
    private NotesAdapter adapter;

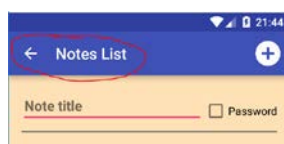
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_notes_list);
        final ListView notesListView = (ListView) findViewById(R.id.listView);
        adapter = new NotesAdapter(this, R.layout.note_row, notes);
        notesListView.setAdapter(adapter);

        notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                //listener for add NoteImage
                ImageView addNoteImg = (ImageView) findViewById(R.id.addNoteImg);
                addNoteImg.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        //create an intent
                        Intent addNoteIntent = new Intent(getApplicationContext(), AddNote.class);
                        startActivityForResult(addNoteIntent, ADD_NOTE_REQUEST);
                    }
                });
            }
        });
    }
}
```

- Open the app in the emulator/device and test that the listener is working properly. (You should be able to navigate from NotesList activity to AddNote activity.)

**Note:** At the moment the only way to return to NotesList activity is by pressing the back button  from the phone.

- Now you will create a return arrow in the menu of **AddNote** so you can go back to **NotesList** without using the back button.



Open **AndroidManifest.xml** file and add the following code to **.AddNote** activity:

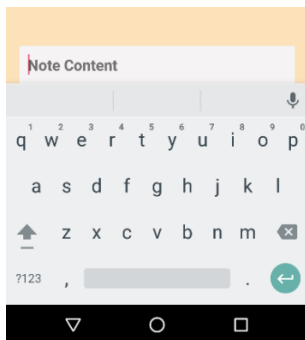
```

<activity
    android:name=".AddNote"
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".NotesList" />
</activity>

```

Meta-data is used to create the return arrow on the left top corner of your activity. The **value** will contain the activity that you want to reach when you press the arrow.

5. Open the app in the emulator/device and test the navigation between activities
6. Navigation works so far but there is an issue with the layout of **AddNote** activity. If you try to type some text in the note content, the keyboard will pop-up and cover most of the layout.



Your next step is to fix this issue. Luckily there is an easy fix for that just adding one line of code inside **AndroidManifest.xml** file. Add **android:windowSoftInputMode="adjustResize"** inside **addNote** activity and **editNote** activity.

Also you will notice an extra property **android:label="Notes List"** this property is used to change the text from the activity menu.

Completed **AndroidManifest.xml** code:

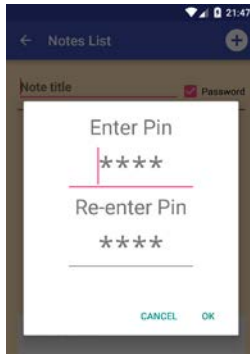
```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="androidcourse.notes">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".NotesList">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".AddNote"
            android:label="Notes List"
            android:windowSoftInputMode="adjustResize">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".NotesList" />
            </activity>
        <activity android:name=".EditNote"></activity>
    </application>
</manifest>

```

- In order to create a note with a password we need to display a dialog to the user so the password can be set.



For this to work we need to create an **xml template layout** for the Pin dialog.

- First you will need to create a custom layout for the pin dialog.  
Create the following layout inside layout resources folder and save it as *pin\_layout.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Enter Pin"
        android:textSize="30sp" />

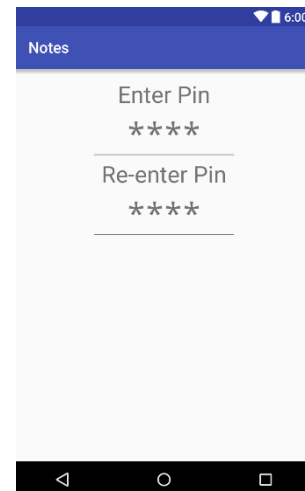
    <EditText
        android:id="@+id/pwd1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:ems="4"
        android:gravity="center_horizontal"
        android:hint="*****"
        android:inputType="numberPassword"
        android:maxLength="4"
        android:textSize="40sp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Re-enter Pin"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/pwd2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:ems="4"
        android:gravity="center_horizontal"
        android:hint="*****"
        android:inputType="numberPassword"
        android:maxLength="4"
        android:textSize="40sp" />

    <TextView
        android:id="@+id/TextView_PwdProblem"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textStyle="bold" />

</LinearLayout>
```



9. Now you will be adding the required functionality inside **AddNote** activity.  
Open **AddNote** java file and write the logic required to **create a new note** and then send it back to **NotesList** activity using an **Intent** so the list of notes can be updated.

```
package androidcourse.notes;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;

import androidcourse.notes.Models.Note;

public class AddNote extends AppCompatActivity {

    private String mPassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_note);
        final CheckBox password = (CheckBox) findViewById(R.id.pwdCheckBox);
        assert password != null;
        password.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                View layout = getLayoutInflater().inflate(R.layout.pin_layout, null);
                final EditText password1 = (EditText) layout.findViewById(R.id.pwd1);
                final EditText password2 = (EditText) layout.findViewById(R.id.pwd2);
                final TextView error = (TextView) layout.findViewById(R.id.TextView_PwdProblem);
                //Listener for the password checkbox
                password2.addTextChangedListener(new TextWatcher() {
                    @Override
                    public void beforeTextChanged(CharSequence s, int start, int count, int after) {}
                    @Override
                    public void onTextChanged(CharSequence s, int start, int before, int count) {}

                    public void afterTextChanged(Editable s) {
                        String strPass1 = password1.getText().toString();
                        String strPass2 = password2.getText().toString();
                        //validate if both passwords are the same
                        if (strPass1.equals(strPass2) && strPass2.length() == 4) {
                            error.setText("Passwords Match");
                            error.setTextColor(Color.GREEN);
                        } else if (strPass2.length() != 4) {
                            error.setText("Password must contain 4 digits");
                            error.setTextColor(Color.RED);
                        } else {
                            error.setText("Passwords do not Match");
                            error.setTextColor(Color.RED);
                        }
                    }
                });
                AlertDialog.Builder builder = new AlertDialog.Builder(AddNote.this);
                builder.setView(layout);
                builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        //if password prompt is cancelled disable checkbox
                        password.setChecked(false);
                        mPassword = null;
                    }
                });
            }
        });
    }
}
```

Continues in next page

```

        builder.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                String strPassword1 = password1.getText().toString();
                String strPassword2 = password2.getText().toString();
                if (strPassword1.equals(strPassword2)) {
                    mPassword = strPassword1;
                }
            }
        });
        AlertDialog passwordDialog = builder.create();
        passwordDialog.show();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.add_note_menu, menu);
    return true;
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.SaveNote) {
        String title = ((EditText) findViewById(R.id.title_add)).getText().toString();
        String contents = ((EditText) findViewById(R.id.title_add)).getText().toString();
        Note note;
        //check if the note has password set.
        if (mPassword == null) {
            note = new Note(title, contents);
        } else {
            note = new Note(title, contents, mPassword);
        }
        //return to NotesList activity
        Intent newNote = new Intent();
        //add new note to the intent
        newNote.putExtra("note", note);
        setResult(RESULT_OK, newNote);
        finish();
    }
    return true;
}
}

```

10. After resolving all the missing dependencies, you should be getting some errors in:

```
newNote.putExtra("note", note);
```

Don't worry for now we'll fix that in step 13. (Note must implement serializable class)

11. `onOptionsItemSelected` method is in charge of registering the click event for saving the note. 

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.SaveNote) {
        String title = ((EditText) findViewById(R.id.title_add)).getText().toString();
        String contents = ((EditText) findViewById(R.id.title_add)).getText().toString();
        Note note;
        //check if the note has password set.
        if (mPassword == null) {
            note = new Note(title, contents);
        } else {
            note = new Note(title, contents, mPassword);
        }
        //return to NotesList activity
        Intent newNote = new Intent();
        //add new note to the intent
        newNote.putExtra("note", note);
        setResult(RESULT_OK, newNote);
        finish();
    }
    return true;
}

```



## 12. The following code is creating the Alert dialog that is used for the password prompt:

```

View layout = getLayoutInflater().inflate(R.layout.pin_layout, null);
final EditText password1 = (EditText) layout.findViewById(R.id.pwd1);
final EditText password2 = (EditText) layout.findViewById(R.id.pwd2);
final TextView error = (TextView) layout.findViewById(R.id.TextView_PwdProblem);
//Listener for the password checkbox
password2.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
    }

    public void afterTextChanged(Editable s) {
        String strPass1 = password1.getText().toString();
        String strPass2 = password2.getText().toString();
        //validate if both passwords are the same
        if (strPass1.equals(strPass2) && strPass2.length() == 4) {
            error.setText("Passwords Match");
            error.setTextColor(Color.GREEN);
        } else if (strPass2.length() != 4) {
            error.setText("Password must contain 4 digits");
            error.setTextColor(Color.RED);
        } else {
            error.setText("Passwords do not Match");
            error.setTextColor(Color.RED);
        }
    }
});
AlertDialog.Builder builder = new AlertDialog.Builder(AddNote.this);
builder.setView(layout);
builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        //if password prompt is cancelled disable checkbox
        password.setChecked(false);
        mPassword = null;
    }
});
builder.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        String strPassword1 = password1.getText().toString();
        String strPassword2 = password2.getText().toString();
        if (strPassword1.equals(strPassword2)) {
            mPassword = strPassword1;
        }
    }
});
AlertDialog passwordDialog = builder.create();
passwordDialog.show();
});

```

## 13. Open Note model class and make the Note class implement Serializable.

```
public class Note implements Serializable {
```

## 14. AddNote activity is complete now. Open NotesList activity and add the following method in order to receive the note sent from addNote activity and add it to the list view.

```

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == ADD_NOTE_REQUEST) {
        if (resultCode == RESULT_OK) {
            //get the new note from the intent
            Note newNote = (Note) data.getSerializableExtra("note");
            notes.add(newNote);
            //notify the adapter and
            // update the listView adapter with the new Note
            adapter.notifyDataSetChanged();
        }
    }
}

```

adapter variable will complain so make sure to declare it as a class variable in order to give it class scope.


15. Open the app in the emulator/device and test adding a few notes with and without password.
16. You will now add the functionality to edit an existing note. First you will be completing the logic for:

```
notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    }
});
```

The idea is that after you click a note, **EditNote** activity will open with the information of the clicked note and from there you can make any changes or even delete the clicked note.

17. Add the required lines to the listener for notesListView:

```
notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Note note = notes.get(position);
        //create intent to edit selected note
        Intent editNoteIntent = new Intent(NotesList.this, EditNote.class);
        editNoteIntent.putExtra("note to edit", note);
        startActivityForResult(editNoteIntent, EDIT_NOTE_REQUEST);
    }
});
```




Similar to what you did with **AddNote**, you will need to create another **request constant**, in this case for edit note:

```
public class NotesList extends AppCompatActivity {
    private ArrayList<Note> notes = new ArrayList<Note>();
    private static final int ADD_NOTE_REQUEST = 1;
    private static final int EDIT_NOTE_REQUEST = 2;
    private NotesAdapter adapter;
```

18. Now you will be adding the required functionality inside **EditNote** activity.  
Open **EditNote** java file and add a global property used to store the note that is coming from the other activity. Also add the required code in order to receive the note from the other activity into **onCreate** method.

```
public class EditNote extends AppCompatActivity {
    private Note noteToEdit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_note);
        //obtaining the note from main activity
        Intent intent = getIntent();
        if (intent.hasExtra("note to edit")) {
            noteToEdit = (Note) intent.getSerializableExtra("note to edit");
        }
        //updating the UI with the note info
        EditText title = (EditText) findViewById(R.id.title_edit);
        title.setText(noteToEdit.getTitle());
        EditText content = (EditText) findViewById(R.id.note_info_edit);
        content.setText(noteToEdit.getContent());
    }
}
```

19. Now run your app, create a new note and then click on the note and you'll see that the note opens in EditNote activity. The next step will be making the changes to the note persistent by adding a listener for  button.
20. You will need to modify your Note model class in order to accommodate for the new functionality. Open Note model class and add the following **set** methods:

```
public void setTitle(String title) {
    this.title = title;
}

public void setContent(String content) {
    this.content = content;
}

public void setLastModified(Date lastModified) {
    this.lastModified = lastModified;
}

public void setPassword(String password) {
    this.password = password;
}
```

21. Create the following method inside **EditNote** activity

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(item.getItemId()==R.id.SaveNote)
    {
        String title = ((EditText) findViewById(R.id.title_edit)).getText().toString();
        String content = ((EditText) findViewById(R.id.note_info_edit)).getText().toString();
        //updating the note
        noteToEdit.setTitle(title);
        noteToEdit.setContent(content);
        //responding to the request from main activity
        Intent editNote = new Intent();
        editNote.putExtra("note", noteToEdit);
        setResult(RESULT_OK, editNote);
        finish();
    }
    return true;
}
```

This method saves the new changes to the existing note and sends the note back to **NotesList** activity. Next step would be to notify the adapter of the changes.

22. Once we get the updated note back to **NotesList** activity we need to search for its position on the list so we can replace the old note with the new one.  
You will create **updateNotesList** method inside NotesList activity to help you with this task:

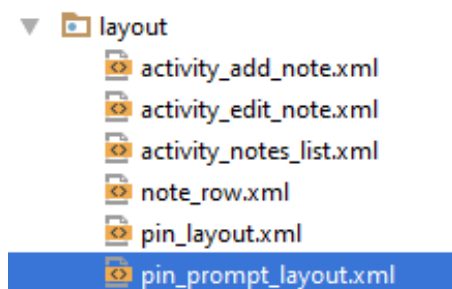
```
private void updateNotesList(Note updatedNote)
{
    Note note = null;
    for(Note n : notes)
    {
        if(updatedNote.getId()== n.getId())
        {
            note = n;
            break;
        }
    }
    int pos = notes.indexOf(note);
    notes.set(pos, updatedNote);
}
```

23. The last step would be to add extra code in order to receive the request and handle it with the help of **onActivityResult** method.

Continues in next page

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == ADD_NOTE_REQUEST) {
        if (resultCode == RESULT_OK) {
            //get the new note from the intent
            Note newNote = (Note) data.getSerializableExtra("note");
            notes.add(newNote);
            //notify the adapter and
            // update the listView adapter with the new Note
            adapter.notifyDataSetChanged();
        }
    } else if (requestCode == EDIT_NOTE_REQUEST) {
        if (resultCode == RESULT_OK) {
            //get the edited note from the intent
            Note updatedNote = (Note) data.getSerializableExtra("note");
            updateNotesList(updatedNote);
            //notify the adapter and
            // update the listView
            adapter.notifyDataSetChanged();
        }
    }
}
```

24. Test your application in the emulator and/or your android device.
25. There is a problem with the application; when a note is selected, it will automatically open for edit mode even if it is password protected.
26. You will be fixing this by creating an alert dialog that prompts the user for a password if the note is password protected.
- Create a new xml resource layout for the password prompt, name the file **pin\_prompt\_layout.xml**



The code for the layout is as follows: (See next page)

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

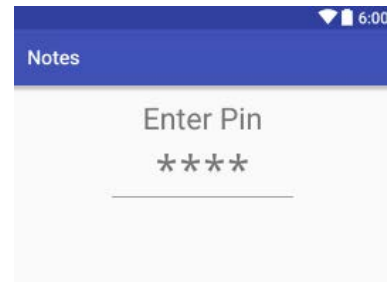
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Enter Pin"
        android:textSize="30sp" />

    <EditText
        android:id="@+id/pwd1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:ems="4"
        android:gravity="center_horizontal"
        android:hint="****"
        android:inputType="numberPassword"
        android:maxLength="4"
        android:textSize="40sp" />

    <TextView
        android:id="@+id/TextView_PwdProblem"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textStyle="bold" />

</LinearLayout>

```



27. Create the following two methods inside **NotesList** activity. (the purpose for this first method is to create the intent for the edit note listener).

```

private void editNoteIntent(Note note) {
    Intent editNoteIntent = new Intent(NotesList.this, EditNote.class);
    editNoteIntent.putExtra("note to edit", note);
    startActivityForResult(editNoteIntent, EDIT_NOTE_REQUEST);
}

```

The purpose of this second method is to display the password prompt.  
(Code in next page)

```

private void displayPinPrompt(final Note note) {
    View layout = getLayoutInflater().inflate(R.layout.pin_prompt_layout, null);
    final EditText password1 = (EditText) layout.findViewById(R.id.pwd1);
    final TextView error = (TextView) layout.findViewById(R.id.TextView_PwdProblem);

    password1.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {}

        public void afterTextChanged(Editable s) {
            String strPass1 = password1.getText().toString();
            //validate if password is correct
            if (!strPass1.equals(note.getPassword())) {
                error.setText("Invalid Password");
                error.setTextColor(Color.RED);
            }
            else
            {
                error.setText("Valid Password");
                error.setTextColor(Color.GREEN);
            }
        }
    });

    AlertDialog.Builder builder = new AlertDialog.Builder(NotesList.this);
    builder.setView(layout);
    builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            dialog.cancel();
        }
    });
    builder.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            String strPassword1 = password1.getText().toString();
            if (strPassword1.equals(note.getPassword())) {
                editNoteIntent(note);
            }
        }
    });
    AlertDialog passwordDialog = builder.create();
    passwordDialog.show();
}

```

28. Resolve all the missing dependencies with **alt + enter**

29. The last step will be to modify the listener for **notesListView** in order to display the password prompt.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_notes_list);
    final ListView notesListView = (ListView) findViewById(R.id.listView);
    adapter = new NotesAdapter(this, R.layout.note_row, notes);
    notesListView.setAdapter(adapter);

    notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            //get the selected note from the list
            Note note = notes.get(position);
            //if note is password protected
            if (note.getPassword() != null) {
                displayPinPrompt(note);
            } else {
                editNoteIntent(note);
            }
        }
    });

    //listener for add NoteImage
    ImageView addNoteImg = (ImageView) findViewById(R.id.addNoteImg);
    addNoteImg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //create an intent
            Intent addNoteIntent = new Intent(getBaseContext(), AddNote.class);
            startActivityForResult(addNoteIntent, ADD_NOTE_REQUEST);
        }
    });
}

```

30. Test the functionality of the app by adding and updating multipole notes.
31. Next session we will add database functionality by using an ORM framework.