What's in this lab 😊

In this lab we will learn how to:

- Create user interfaces.
- Linking Activities together.

Creating User Interfaces: Layouts

First let us explore some basic layouts and introduce some views\controls too.

**LinearLayout**

The LinearLayout arranges views in a single column or a single row. Child **views** can be arranged either vertically or horizontally.
TableLayout

The TableLayout groups views into rows and columns. You use the <TableRow> element to designate a row in the table. Each row can contain one or more views. Each view you place within a row forms a cell. The width of each column is determined by the largest width of each cell in that column.

RelativeLayout (Nice one!)

The RelativeLayout enables you to specify how child views are positioned relative to each other.
Relative layout depends on:

**dp** — Density-independent pixel. 160dp is equivalent to one inch of physical screen size. This is the recommended unit of measurement when specifying the dimension of views in your layout. You can specify either “dp” or “dip” when referring to a density-independent pixel.

**sp** — Scale-independent pixel. This is similar to dp and is recommended for specifying font sizes.

---

**Lab Work 1**

- Create the following interface:

Remember:
- You can use Multiple Layouts!
- Use margins
Creating User Interfaces: Views \ Controls

Views or controls are simply buttons, labels, text fields and so on but with their names changed ^_^

We knew these views as JLabel, JTextField and so on but here they are named as follows:

All of them have properties that can be manipulated, with time we will use all of them!

For now we need to know how to change their Text and Id

Text is what the user sees.
Id is what the code sees.

And to change them we right-click on any view and set them, see the figure
Creating User Interfaces: Access Views From Java Code

Up to now we have only created the views but cannot see them from Java code. In order to get access to them from code we use the method

\textbf{findViewById(id)}

to get a pointer to our views, however this method returns a View object and we have to cast it to the right view for example:

\begin{verbatim}
EditText et = (EditText)findViewById(R.id.editText1);
Button myButton = (Button)findViewById(R.id.button_id)
\end{verbatim}

**Listeners**

There are a lot of methods and listeners that we will use ,, for now we will introduce the listener of a button :

\begin{verbatim}
Button myButton = (Button)findViewById(R.id.button_id)
myButton.setOnClickListener(new OnClickListener()
{
    public void onClick(View arg0) {
        // TODO Auto-generated method stub
        Your code goes here!!
    }
});
\end{verbatim}

**Lab Work 2**

1. For the previously designed interface change the texts to be as shown and assign some IDs for your views.
2. Get references to the button , both EditTexts and Result TextView.
3. Implement Listener for the button with the following logic:

\begin{itemize}
    \item If name is android and pass is 123123 set result to : success ! otherwise set it to Failed!
\end{itemize}
Intents

Intents are probably the most unique, and important, concept in Android development.

Intents provide a mechanism for message-passing within and between applications. Using Intents you can broadcast a desired action (such as dialing the phone or editing a contact) system-wide for other applications to handle. Intents are an important core component of Android.

Using Intents to Launch Activities

Simple one !!
Create New Intent(current activity.this, target activity.class) object and pass it to startActivity(Intent) method.

```java
Intent i = new Intent(Lab2.this, SecondActivity.class);
startActivity(i);
```

Don't Forget to Register all your activities in the AndroidManifest.xml File

Implicit intents

Last lab we said that almost everything we see is an activity ,, so we don't have to re construct the Dial-up activity for example in order to make a call !

So android provides some implicit actions that can be caught and be handled by the Operating System.

So how to use them , Also Simple :D
Create an implicit intent(Action you want, data input) and start it 😊

```java
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.amazon.com"));
startActivity(i);
```
Another example:

```
Intent(Intent.ACTION_DIAL, Uri.parse("tel:+651234567"));
startActivity(i);
```

We will work on implicit intents on a later lab 😊

---

### Lab Work 3

1. Create an activity that contains a Textview "This is a new activity! : welcome"
2. If sign in is ok form the last lab work : move to this new activity.

---

### Passing Data to the Started Activity

In some situations we need to give the activity an input data to work with, this can be done as following

**Original Activity**

```
Intent i = new Intent(Lab2.this, SecondActivity.class);
i.putExtra("name", "nour");
i.putExtra("pass", 123123);
i.putExtra("rememberMe", true);
startActivity(i);
```

**Started Activity**

```
Intent i = this.getIntent(); // the intent who started this activity
//extract the data
String x = i.getStringExtra("name");
int y = i.getIntExtra("pass", -1); //-1 : default value if null is returned
boolean z = i.getBooleanExtra("rememberMe", false); //false : Default value
```
Lab Work 4

1. Modify the last lab work so you send the name and pass entered in EditTexts to the second activity.
2. Modify the label for second activity to be: welcome "name", your pass is "pass"

Starting an Activity for a result

In some cases we need to get results from the started activity. In this case we call it Sub Activity. This is not that different from the normal activity starting. But we mark it with an integer and use the method startActivityForResult

```java
final public static int SECOND_ACTIVITY = 2;
final public static int THIRD_ACTIVITY = 3;

startActivityForResult(secondIntent, SECOND_ACTIVITY);
startActivityForResult(secondIntent, THIRD_ACTIVITY);

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case SECOND_ACTIVITY:
            if(resultCode==Activity.RESULT_OK){
                name = data.getStringExtra("user");
                pass = data.getStringExtra("pass");
            } else if (resultCode==Activity.RESULT_CANCELED){
                //Do Something
            }
            break;
        case THIRD_ACTIVITY:
            if(resultCode==Activity.RESULT_OK) {
                //not supported yet
            } else if(resultCode==Activity.RESULT_OK){
                //do something
            }
            break;
    }
}
```
Lab Work 5

1. Add a button named "Lunch new activity for result" to the login interface, get it from code and implement its action listener.
2. Design an activity with an EditText and Done button.
3. Implement the button so that it returns the value from EditText to the main activity.
4. Display the returned text on the Result TextView.

Homework

- Will be posted tonight 😊