

TESTO

PULSANTE 1

PULSANTE 2

TESTO

```

<RelativeLayout android:xmlns="http://schemas.android.com/apk/res/android"
    android:layout_height="match-parent"
    android:layout_width="match-parent">
    <LinearLayout
        android:layout_height="wrap-content"
        android:layout_width="wrap-content"
        android:orientation="vertical">
        <TextView
            android:layout_height="wrap-content"
            android:layout_width="wrap-content"
            android:gravity="centerHorizontal"
            android:text="Sopra"
            android:textSize="10sp" />
        <LinearLayout
            android:layout_height="wrap-content"
            android:layout_width="wrap-content"
            android:orientation="horizontal">
            <Button
                android:id="@+id/button1"
                android:layout_height="5dp"
                android:layout_width="10dp"
                android:text="Pulsante 1"
                android:textSize="10sp"
                android:gravity="centerHorizontal | centerVertical"
                android:onClick="aziona1" />
            <Button
                android:id="@+id/button2"
                android:layout_height="5dp"
                android:layout_width="10dp"
                android:text="Pulsante 2"
                android:textSize="10sp"
                android:gravity="centerHorizontal | centerVertical"
                android:onClick="aziona2" />
        </LinearLayout>
    </LinearLayout>
    <TextView
        android:layout_height="wrap-content"
        android:layout_width="wrap-content"
        android:gravity="centerHorizontal"
        android:text="Sotto"
        android:textSize="10sp" />
</RelativeLayout>

```

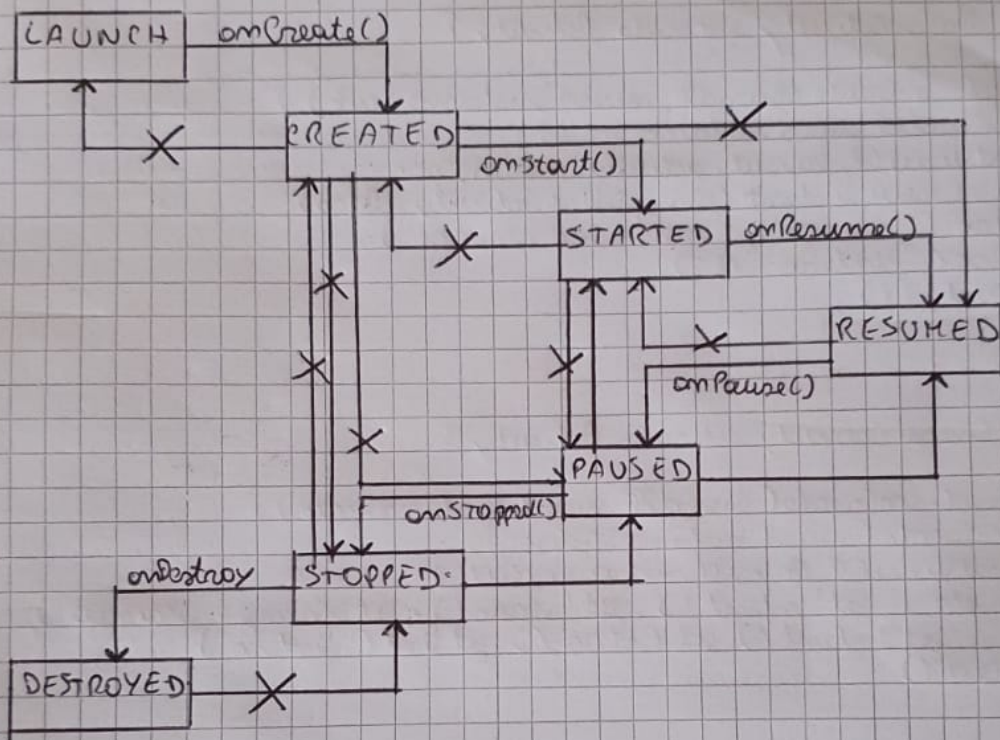
ZAPPÀ, ALFONSO

QUESITO 2

Il metodo `onItemClicked` è un metodo di callback che viene invocato quando un elemento della lista viene cliccato. Per compiere questa azione, il metodo prende in input un `AdapterView`. `AdapterView` è una view che permette di raggruppare tutti gli elementi determinati da un `Adapter`. Oltre `ListView` è possibile usare anche elementi di tipo `GridView` o `Spinner`. Si tratta perlopiù di

ZAPPÀ, ALFONSO

QUESTO 3



ZAPPIA, ALFONSO

QUESTO 4

1) public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

Intent i = new Intent(this, AltroActivity.class);

i.putExtra("Stringa", "Sono una stringa");

i.putExtra("Intero", 139);

startActivity(i);

}

2) public class AltroActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.altro_layout);

String str = getIntent().getExtras().getString("Stringa");

int m = getIntent().getExtras().getInt("Intero");

}

Normalmente, un meccanismo di layout prevede varie fasi: la prima fase di misurazione, poi di layout e infine di disegno. La fase di disegno significa che dopo il posizionamento la View viene disegnata attraverso il metodo `onDraw()`. Ed è qui che interviene anche il metodo `invalidate()`. Esso viene chiamato quando c'è un cambiamento e, a sua volta, chiama `onDraw` sulla View. Una volta chiamato `invalidate()`, si ripete l'intero processo di disegno su tutte le gerarchie della View.

QUESTO 7

Per comprendere `AsyncTask`, ci tocca sapere che, durante la programmazione in Android, l'UI lavora solamente sul thread principale dell'applicazione. Queste procedure ha parecchi limiti, poiché il thread svolge solamente operazioni in background, senza curarsi di aggiornare l'interfaccia utente. E qui interviene la classe `AsyncTask`. Per prime cose, occorre implementare il metodo `doInBackground()`, che conterrà tutte le operazioni che saranno eseguite da un thread secondario. Esistono poi altri metodi, meno indispensabili, la cui implementazione non è però, obbligatoria:

- `onPreExecute()`, viene eseguito prima di `doInBackground()` e serve a definire e configurare alcuni oggetti utili per la successiva esecuzione
- `onProgressUpdate()`, viene invocato da `doInBackground()`, durante l'esecuzione, ed è il metodo che fornisce le operazioni di aggiornamento
- `onPostExecute()`, viene chiamato dopo l'esecuzione di `doInBackground()` e si occupa di mostrare all'utente i risultati dell'operazione

`AsyncTask` prevede anche un altro metodo, detto `onCancelled`, che viene invocato solo in caso di brusche interruzioni.

QUESTO 8

Un'animazione, fondamentalmente, è una risorsa e quindi per definirla è necessario creare una cartella in RES. Un file XML contiene soltanto la configurazione dell'animazione. Il layout della View sarà specificato attraverso l'animazione sarà inserita all'interno del tag `<set>` che racchiude tutte le configurazioni. All'interno del set sarà possibile definire tutte le possibili animazioni con i relativi attributi:

- rotate, l'elemento ruota
- scale, l'elemento si ingrandisce
- alpha, l'elemento sbiadisce in base alle impostazioni di trasparenza
- translate, l'elemento si sposta sullo schermo

Una volta definite le animazioni nel file XML sarà necessario importare la classe `AnimationUtils` all'interno dell'`onCreate`, creare un oggetto di quel tipo e infine, chiamare l'opportuno metodo `startAnimation`.

Cursor è una classe che serve a rappresentare una vista bidimensionale di un qualsiasi database. Cursor viene principalmente usato all'interno del Content Provider, cioè all'interno di contenitori di dati. Usiamo un Cursor poiché in un Content Provider abbiamo a memorizzare i dati all'interno delle tabelle. Non dovremo far altro che impostare una normale query, che restituirà un Cursor, che visualizzerà tutti i dati della query.

QUESITO 10

Un Service, in Android, serve normalmente ad eseguire delle operazioni che richiedono tempi molto lunghi. Un Service non interagisce con l'utente ma esegue operazioni in background. Per usare un service bisogna iniziarlo con `startService()`. Una volta partito, il Service attende l'esecuzione di un evento o il verificarsi di un evento a operazione conclusa. Per interagire con un Service bisogna usare il metodo `bindService()` che può inviare richieste o ricevere risposte. Nota bene. È necessario dichiarare il Service all'interno del Manifest.