UNIVERSITÀ DI SALERNO

Corso di laurea in Informatica

Mobile Programming

Prof. De Prisco

Prova scritta del

esempio 7

NOME: _____

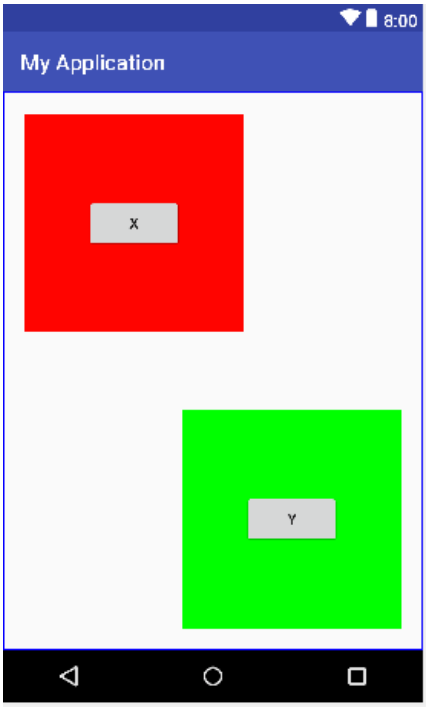
COGNOME:_____

MATRICOLA: _____

Domande	Punti
1	/10
2	/10
3	/10
4	/10
5	/10
6	/10
7	/10
8	/10
9	/10
10	/10
TOTALE	/100

Si completi il codice XML che descrive il layout riportato in figura. Il RelativeLayout già scritto nel codice corrisponde all'intero schermo. Le due aree colorate, quella in alto rosso e quella in basso verde, sono a distanza di 20dp dal bordo e contengono al centro un pulsante, con testo "X" quello nel riquadro in alto e "Y" quello nel riquadro in basso.

```
<RelativeLayout 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">
```




```
<RelativeLayout 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"
  android:padding="20dp">

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="100dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:background="#FF0000">

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="X"/>

  </LinearLayout>

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="100dp"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:background="#00FF00">

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Y"/>

  </LinearLayout>

</RelativeLayout>
```

```
</RelativeLayout>
```

Sapendo che nel file XML del layout è presente il seguente snippet

```
<ListView
      android:id="@+id/mylistview"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">
 </ListView>
```

per il posizionamento del ListView, si dica cosa manca al seguente codice che permette di visualizzare una lista di nomi e quando l'utente fa click su un nome di mostrare il nome in un Toast. Oltre a spiegare cosa manca, sul retro della pagina, si completi il codice scrivendo le parti mancanti

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends Activity {
   public ListView listView;

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

      String [] array = {"Pasquale","Maria","Michele","Antonella", "Vincenzo",
            "Teresa", "Roberto", "Rossella", "Antonio", "Luca",
            "Francesca", "Andrea", "Marco", "Elisa", "Anna"};

      listView = (ListView)findViewById(R.id.mylistview);
      ArrayAdapter<String> arrayAdapter =
            new ArrayAdapter<String>(this, R.layout.list_element, R.id.textViewList, array);
      listView.setAdapter(arrayAdapter);

      listView.setOnItemClickListener(new OnItemClickListener() {
         @Override
         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            String str = listView.getItemAtPosition(position).toString();
            Toast.makeText(getApplicationContext(),
                  "Click su posizione n."+position+": " +str, Toast.LENGTH_LONG).show();
         }
      });
   }
}
```

Stai scrivendo un'app che utilizza il sensore GPS che richiede un utilizzo consistente della batteria. Riguardo al ciclo di vita della tua app, che accortezza devi usare per evitare di consumare inutilmente la batteria? Motivare la risposta e fornire qualche dettaglio su cosa fare.

Per evitare di consumare inutilmente la batteria quando si utilizza il sensore GPS, è importante gestire correttamente il ciclo di vita dell'applicazione in modo da accendere e spegnere il GPS solo quando è strettamente necessario.

Un modo per farlo è utilizzare i metodi onResume() e onPause() dell'Activity. Nel metodo onResume() si dovrebbe accendere il GPS, mentre nel metodo onPause() si dovrebbe spegnere il GPS. In questo modo, quando l'app non è in esecuzione in primo piano, il GPS verrà spento, riducendo il consumo della batteria.

È inoltre possibile utilizzare il metodo requestLocationUpdates() della classe LocationManager fornendo un intervallo di tempo di aggiornamento più lungo (ad esempio, ogni 10 minuti) invece di richiedere aggiornamenti in continuo.

Infine, è importante ricordare di rimuovere i listener di posizione quando non sono più necessari, ad esempio quando l'app viene chiusa o viene passata in secondo piano per un periodo di tempo prolungato. Ciò può essere fatto utilizzando il metodo removeUpdates() della classe LocationManager.

Che differenza c'è fra un intent implicito e un intent esplicito? Si facciano degli esempi.

Un intent esplicito è un intent che specifica esattamente quale componente deve essere avviato dall'intent. Ciò viene fatto specificando il nome del package e il nome esatto dell'Activity che si desidera avviare.

Un intent implicito, invece, non specifica esattamente quale componente deve essere avviato. Invece, fornisce un'azione da eseguire e lascia che il sistema decida quale componente è in grado di eseguire tale azione.

Esempio di intent esplicito:

Copy code
```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```
Esempio di intent implicito:

Copy code
```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.example.com"));
startActivity(intent);
```
Nell'esempio di intent implicito, l'azione da eseguire è "visualizzare" e il dato da visualizzare è un URL. Il sistema potrebbe quindi scegliere di avviare il browser per visualizzare il URL.

Si consideri la classe AsyncTask e si supponga di dover scrivere i metodi per il trasferimento di un file da un server. Si specifichi cosa fare in onPreExcute, doInBackground, publishProgress, onProgressUpdate, e onPostExecute per poter viusalizzare correttamente lo stato di avanzamento del trasferimento fornendo una una percentuale di completamento. Si scriva il codice. **NOTA:** non è necessario fornire tutto il codice per il trasferimento del file, ma solo le istruzioni relative alla visualizzazione dell'avanzamento del trasferimento.

Ecco come potrebbero essere implementati i metodi della classe AsyncTask per visualizzare correttamente lo stato di avanzamento del trasferimento di un file da un server:

Copy code

```
private class DownloadTask extends AsyncTask<URL, Integer, Long> {
    protected void onPreExecute() {
        // Mostra una finestra di dialogo o una barra di progresso per indicare che il trasferimento è iniziato
    }

    protected Long doInBackground(URL... urls) {
        int count = urls.length;
        long totalSize = 0;
        for (int i = 0; i < count; i++) {
            // Scarica il file
            totalSize += downloadFile(urls[i]);
            // Pubblica l'avanzamento del trasferimento
            publishProgress((int) ((i / (float) count) * 100));
        }
        return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
        // Aggiorna la barra di progresso o la finestra di dialogo con la percentuale di completamento del
trasferimento
    }

    protected void onPostExecute(Long result) {
        // Nascondi la barra di progresso o la finestra di dialogo e mostra il risultato del trasferimento
    }
}
```

Il metodo onPreExecute() viene eseguito prima che il trasferimento inizi. In questo metodo si potrebbe mostrare una finestra di dialogo o una barra di progresso per indicare che il trasferimento è iniziato.

Il metodo doInBackground(URL... urls) viene eseguito per scaricare il file. In questo metodo si può utilizzare il metodo downloadFile(URL url) per scaricare il file. Si può anche utilizzare il metodo publishProgress(Integer... values) per pubblicare l'avanzamento del trasferimento.

Il metodo onProgressUpdate(Integer... progress) viene eseguito ogni volta che viene pubblicato un nuovo progresso dal metodo doInBackground(). In questo metodo si può aggiornare la barra di progresso o la finestra di dialogo con la percentuale di completamento del trasferimento.

Il metodo onPostExecute(Long result) viene eseguito dopo che il trasferimento è stato completato. In questo metodo si può nascondere la barra di progresso o la finestra di dialogo e mostrare il risultato del trasferimento.

Se due frammenti di un'activity devono comunicare è buona prassi di programmazione implementare tale comunicazione non in modo diretto da frammento a frammento ma passando attraverso l'activity che ospita i frammenti (quindi la frammento che vuole inviare la comunicazione lo fa interagendo con l'activity ospitante e poi questa interagisce con il frammento che deve ricevere la comunicazione).

Perché è una buona prassi di programmazione?

Si descriva un modo per implementare la comunicazione fra due frammenti attraverso l'activity ospitante.

È una buona prassi di programmazione implementare la comunicazione fra due frammenti attraverso l'activity ospitante perché questo rende più facile e flessibile la gestione dei frammenti.

Se i frammenti comunicano direttamente l'uno con l'altro, potrebbero esserci dipendenze forti tra di loro e ciò potrebbe rendere difficile modificare o rimuovere i frammenti in futuro. Inoltre, se i frammenti sono gestiti da diverse activity, la comunicazione diretta tra di loro potrebbe non essere possibile.

Per implementare la comunicazione fra due frammenti attraverso l'activity ospitante, è possibile utilizzare una interfaccia e un callback. In particolare, si può definire un'interfaccia con i metodi di callback che il frammento che vuole inviare la comunicazione dovrà implementare. L'activity ospitante implementerà questa interfaccia e fornirà un'istanza del frammento che deve ricevere la comunicazione.

Ecco un esempio di come potrebbe essere implementata la comunicazione fra due frammenti attraverso l'activity ospitante:

Copy code

```
public class MainActivity extends Activity implements Fragment1.OnFragmentInteractionListener {

    @Override
    public void onFragmentInteraction(String message) {
        // Qui si può ricevere la comunicazione dal primo frammento e inoltrare il messaggio al secondo frammento
        Fragment2 fragment2 = (Fragment2) getSupportFragmentManager().findFragmentById(R.id.fragment2);
        fragment2.updateText(message);
    }
}

public class Fragment1 extends Fragment {
    // Interfaccia per la comunicazione con l'activity ospitante
    public interface OnFragmentInteractionListener {
        void onFragmentInteraction(String message);
    }

    private OnFragmentInteractionListener mListener;

    // Assicurarsi che l'activity ospitante implementi l'interfaccia
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof OnFragmentInteractionListener) {
            mListener = (OnFragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString() + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    // Invia un messaggio all'activity ospitante
    public void sendMessage(String message) {
        if (mListener != null) {
            mListener.onFragmentInteraction(message);
        }
    }
}

public class Fragment2 extends Fragment {
    // Aggiorna il testo del frammento con il messaggio ricevuto
        public void updateText(String message) {
        TextView textView = (TextView) getView().findViewById(R.id.text_view);
       textView.setText(message);
  }
}
```

Che cosa è un MotionEvent? Come viene utilizzato? Si faccia un esempio.

Un MotionEvent è un evento che viene generato quando viene rilevato un movimento sulla superficie di input di un dispositivo, ad esempio il tocco di un dito sullo schermo di uno smartphone o il movimento di un mouse su un computer.

Un MotionEvent viene utilizzato per fornire informazioni sulla posizione, il movimento e l'azione dell'evento di input. Ad esempio, un MotionEvent può fornire informazioni sulla posizione del tocco sullo schermo, il movimento del dito durante il tocco e l'azione che ha causato l'evento, ad esempio il tocco, il trascinamento o il rilascio.

Un MotionEvent viene utilizzato comunemente per gestire gli eventi di input nella classe View o in una sottoclasse di View, ad esempio una Button. Ecco un esempio di come potrebbe essere utilizzato un MotionEvent per gestire un evento di tocco su un pulsante:
La seguente classe Pentagramma implementa un widget customizzato che permette di inserire delle note in un pentagramma. Le note sono memorizzate con degli oggetti Nota che specificano la posizione nel pentagramma tramite delle coordinate cartesiane. Completare la classe Pentagramma per permettere di inserire nuove note (il costruttore, la funzione onLayout e onDraw già ci sono anche se non sono riportati i dettagli, quindi non serve aggiungere niente per queste funzioni). Cosa si deve fare, all'esterno di questa classe, dopo avere inserito o cancellato una nota affinché la modifica venga visualizzata?

```
public class Pentagramma extends View {
ArrayList<Nota> notesList = new ArrayList<Nota>();
public Pentagramma(Context c, int screen_w, int screen_h) {
super(c);
//Costruttore, inizializza l'oggetto
}
@Override
protected void onLayout(boolean b, int x1, int y1, int x2, int y2) {
//Questa funzione specifica le dimensioni
}
@Override
protected void onDraw(Canvas canvas) {
//Questa funzione disegna il pentagramma e le note in notesList
}
```

La seguente classe Pentagramma implementa un widget customizzato che permette di inserire delle note in un pentagramma. Le note sono memorizzate con degli oggetti Nota che specificano la posizione nel pentagramma tramite delle coordinate cartesiane. Completare la classe Pentagramma per permettere di inserire nuove note (il costruttore, la funzione `onLayout` e `onDraw` già ci sono anche se non sono riportati i dettagli, quindi non serve aggiungere niente per queste funzioni). Cosa si deve fare, all'esterno di questa classe, dopo avere inserito o cancellato una nota affinché la modifica venga visualizzata?

```
public class Pentagramma extends View {
      ArrayList<Nota> notesList = new ArrayList<Nota>();

      public Pentagramma(Context c, int screen_w, int screen_h) {
         super(c);
         //Costruttore, inizializza l'oggetto
      }

     @Override
          protected void onLayout(boolean b, int x1, int y1, int x2, int y2) {
          //Questa funzione specifica le dimensioni
      }

      @Override
      protected void onDraw(Canvas canvas) {
           //Questa funzione disegna il pentagramma e le note in notesList
      }
```

Per permettere di inserire nuove note nella classe Pentagramma, puoi aggiungere un metodo aggiungiNota che accetta come argomento un oggetto Nota e lo aggiunge alla lista notesList:

Copy code
```
public void aggiungiNota(Nota nota) {
   notesList.add(nota);
}
```
Per cancellare una nota, puoi aggiungere un metodo rimuoviNota che accetta come argomento un oggetto Nota e lo rimuove dalla lista notesList:

Copy code
```
public void rimuoviNota(Nota nota) {
   notesList.remove(nota);
}
```
Dopo avere inserito o cancellato una nota, per far sì che la modifica venga visualizzata, è necessario chiamare il metodo invalidate dell'oggetto Pentagramma. Questo farà sì che il sistema chiami nuovamente il metodo onDraw della classe Pentagramma, aggiornando così il pentagramma con la nuova lista di note. Ad esempio:

Copy code
```
Pentagramma pentagramma = new Pentagramma(this, screen_w, screen_h);
Nota nota = new Nota(x, y);
pentagramma.aggiungiNota(nota);
pentagramma.invalidate();
```

Il metodo onMeasure utilizzato nella determinazione della reale grandezza di un elemento dell'albero delle view utilizza un ogggetto MeasureSpec che codifica con un intero sia la dimensione in pixel dell'altezza e della larghezza dell'elemento che una "modalità"; tali quantità possono essere recuperate dall'oggetto MeasureSpecm usando i seguenti metodi:

```
int widthMode = MeasureSpec.getMode(widthMeasureSpec);
int width = MeasureSpec.getSize(widthMeasureSpec);
int heightMode = MeasureSpec.getMode(heightMeasureSpec);
int height = MeasureSpec.getSize(heightMeasureSpec);}
```

I valori che il metodo getMode restituisce sono:

1. MeasureSpec.EXACTLY
2. MeasureSpec.AT_MOST
3. MeasureSpec.UNSPECIFIED

A cosa servono?

I valori restituiti dal metodo getMode dell'oggetto MeasureSpec indicano la modalità di misura utilizzata per specificare la dimensione dell'elemento dell'albero delle view.

MeasureSpec.EXACTLY: indica che la dimensione dell'elemento è specificata esattamente in pixel. Ad esempio, se un elemento ha una larghezza di 100 pixels, la modalità di misura sarà MeasureSpec.EXACTLY.

MeasureSpec.AT_MOST: indica che la dimensione dell'elemento non può superare una determinata misura specificata in pixel. Ad esempio, se un elemento ha una larghezza massima di 100 pixels, la modalità di misura sarà MeasureSpec.AT_MOST.

MeasureSpec.UNSPECIFIED: indica che la dimensione dell'elemento non è specificata in alcun modo e può assumere qualsiasi valore. Questa modalità viene spesso utilizzata quando la dimensione dell'elemento dipende dal contenuto dell'elemento stesso, ad esempio quando un elemento contiene una lunga stringa di testo.

Le activity sono una della 4 componenti fondamentali di Android (e quella che abbiamo studiato in dettaglio durante il corso). Quali sono le altre 3 e a cosa servono?

Services: sono componenti che eseguono attività in background, senza un'interfaccia utente. Ad esempio, un service potrebbe essere utilizzato per effettuare il download di dati da un server o per riprodurre un file audio in background.

Broadcast receivers: sono componenti che ricevono e gestiscono gli intent broadcast, ovvero messaggi che vengono inviati da Android o da altre app per avviare un'attività o per notificare un evento. Ad esempio, un broadcast receiver potrebbe essere utilizzato per ricevere una notifica quando la batteria del dispositivo è scarica o quando viene connesso un nuovo dispositivo Bluetooth.

Content providers: sono componenti che gestiscono l'accesso ai dati archiviati in un'app, rendendo possibile a altre app l'accesso a questi dati. Ad esempio, un content provider potrebbe gestire l'accesso ai contatti della rubrica del dispositivo o ai file multimediali archiviati in un'app di musica.

Prova scritta

Pagina aggiuntiva

Prova scritta

Pagina aggiuntiva

Prova scritta

Pagina aggiuntiva