UNIVERSITÀ DI SALERNO

Corso di laurea in Informatica

Mobile Programming

Prof. De Prisco

Prova scritta del

esempio 10

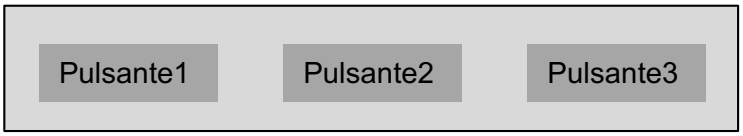
NOME: _____

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 consideri il LinearLayout indicato in figura.



Si costruisca tale layout utilizzando un file statico che specifica i primi 2 pulsante ed inserendo dinamicamente il terzo pulsante (dettagliare il file XML e lo snippet di codice che serve ad aggiungere il terzo pulsante).

```
<LinearLayout xmlns:android="http://
schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal">

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button 1" />

   <Button
      android:id="@+id/button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Button 2" />

</LinearLayout>
```

```
LinearLayout layout = findViewById(R.id.linear_layout);

Button button3 = new Button(this);
button3.setText("Button 3");
layout.addView(button3);
```

Si consideri il seguente frammento di codice che definisce un OnItemClickListener per un ListView.

```
listView.setOnItemClickListener(new OnItemClickListener() {
  @Override
  public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String  str  = listView.getItemAtPosition(position).toString();
    // Fai qualcosa con l'elemento ...
  }
});
```

Si spieghi il ruolo del parametro position.

Il parametro position indica la posizione dell'elemento nella lista che è stato cliccato. Questo è importante perché permette di sapere quale elemento della lista è stato selezionato, che può essere utile per eseguire determinate azioni in base all'elemento selezionato. Ad esempio, potresti voler recuperare il valore di un elemento specifico in una lista di oggetti in base alla sua posizione nella lista. In questo caso, potresti utilizzare il parametro position per individuare l'elemento nella lista e quindi recuperare il valore associato.

Si consideri la seguente situazione: un'app viene lanciata e l'utente interagisce con l'app; ad un certo punto l'utente ruota il dispositivo e continua ad interagire con l'app; infine l'utente chiude l'app tramite un apposito pulsante. In quali stati è passata l'app e quali metodi sono stati chiamati dal momento in cui l'app viene lanciata al momento in cui termina?

Quando un'app viene lanciata, viene chiamato il metodo onCreate() della classe Activity associata alla schermata iniziale dell'app. Quando l'utente interagisce con l'app, vengono chiamati altri metodi della classe Activity in base alle azioni dell'utente, come ad esempio onResume() quando l'app torna in primo piano o onPause() quando l'app viene messa in background.

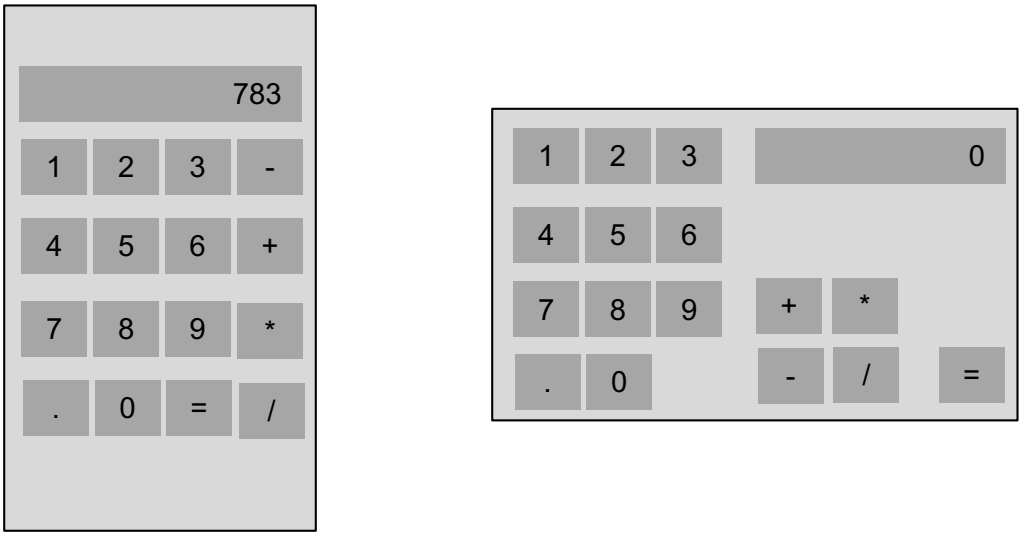
Quando l'utente ruota il dispositivo, viene chiamato il metodo onConfigurationChanged() della classe Activity, che viene utilizzato per gestire le modifiche alla configurazione del dispositivo, come ad esempio la rotazione dello schermo.

Quando l'utente chiude l'app tramite il pulsante di chiusura, viene chiamato il metodo onDestroy() della classe Activity, che viene utilizzato per liberare risorse e terminare l'esecuzione dell'app.

In sintesi, durante il suo ciclo di vita, l'app passerà attraverso i seguenti stati e verranno chiamati i seguenti metodi:

Lanciamento dell'app: onCreate()
Interazione dell'utente con l'app: onResume()
Mettere l'app in background: onPause()
Riportare l'app in primo piano: onResume()
Ruotare il dispositivo: onConfigurationChanged()
Chiudere l'app: onDestroy()

Stai usando il tuo smartphone per fare dei conti con un app “calcolatrice”, come mostrato nella figura a sinistra. Involontariamente ruoti il dispositivo e ti ritrovi con la situazione descritta nella figura a destra. Il valore “783” è diventato “0” ed la disposizione dei tasti è cambiata. Cosa è successo? Cosa ha fatto bene e cosa cosa ha sbagliato il programmatore dell’app? Cosa avrebbe dovuto fare per ovviare all’errore?



Quando si ruota il dispositivo, l'orientamento dello schermo cambia, il che può influire sull'aspetto e il comportamento dell'app. In questo caso, sembra che l'app non sia stata progettata per gestire la rotazione del dispositivo, il che ha causato il reset del valore visualizzato e il cambiamento della disposizione dei tasti.

Il programmatore dell'app avrebbe dovuto prevedere la possibilità che l'utente potesse ruotare il dispositivo durante l'utilizzo dell'app e implementare il codice necessario per gestire questa situazione. Un modo per farlo sarebbe stato utilizzare il metodo onConfigurationChanged() della classe Activity per gestire le modifiche alla configurazione del dispositivo, come ad esempio la rotazione dello schermo. In questo modo, l'app avrebbe potuto mantenere il suo stato e il suo aspetto durante la rotazione del dispositivo, evitando il reset del valore visualizzato e il cambiamento della disposizione dei tasti.

Si consideri il seguente snippet di codice dell'activity ActivityA:

```
Intent intent = new Intent(getBaseContext(), ActivityB.class);
intent.putExtra(”NOME", “Roberto”);
intent.putExtra(“VALORE”,32);
intentputExtra(“STATO”,true);
startActivity(intent);
```

Si scriva lo snippet di codice per l'Activity B per recuperare i valori inseriti nell'intent.

Intent intent = getIntent();
String nome = intent.getStringExtra("NOME");
int valore = intent.getIntExtra("VALORE", 0); // il secondo parametro è il valore di default nel caso in cui il valore non sia stato impostato nell'intent
boolean stato = intent.getBooleanExtra("STATO", false); // il secondo parametro è il valore di default nel caso in cui il valore non sia stato impostato nell'intent

Per quali situazioni occorre utilizzare la classe AsyncTask? Si faccia un esempio.

La classe AsyncTask viene utilizzata per eseguire operazioni in background e comunicare i risultati dell'elaborazione al thread principale dell'interfaccia utente. Questo è utile per evitare di rallentare l'interfaccia utente eseguendo operazioni lunghe o di grandi dimensioni sullo stesso thread.

Un esempio comune di utilizzo di AsyncTask potrebbe essere il caricamento di dati da un'API web in un'applicazione Android. Se si effettua la chiamata all'API e il parsing dei dati sullo stesso thread dell'interfaccia utente, ciò potrebbe causare rallentamenti o blocchi dell'interfaccia utente. Utilizzando AsyncTask, invece, si può eseguire la chiamata all'API e il parsing dei dati in background, quindi utilizzare il metodo onPostExecute per aggiornare l'interfaccia utente con i dati restituiti.

Ecco un esempio di una semplice classe AsyncTask che effettua una chiamata HTTP e restituisce il risultato come una stringa:

Copy code
```
private class DownloadTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        try {
            URL url = new URL(urls[0]);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            InputStream inputStream = connection.getInputStream();
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(inputStreamReader);
            StringBuilder result = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                result.append(line);
            }
            return result.toString();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        // Utilizzare il risultato qui (ad esempio, aggiornare l'interfaccia utente)
    }
}
```
Nell'esempio qui sopra, la chiamata HTTP viene effettuata nel metodo doInBackground, mentre l'aggiornamento dell'interfaccia utente con il risultato avviene nel metodo onPostExecute.
Per eseguire l'AsyncTask, si potrebbe chiamare come segue:

Copy code
```
new DownloadTask().execute("https://example.com");
```
In questo modo l'url che vogliamo interrogare verra passato al metodo doInBackground come primo parametro del metodo execute.

Cosa è l'albero delle view? Quali sono e come funzionano le due fasi necessarie per la visualizzazione del layout?

L'albero delle view è una struttura gerarchica che rappresenta il layout dell'interfaccia utente di un'app Android. Ogni view (ad esempio, un pulsante, una casella di testo, ecc.) è un nodo nell'albero e può avere figli, che sono altre view contenute all'interno di essa.

La visualizzazione del layout in Android segue tre fasi principali:

La fase di layout: in questa fase, ogni view nell'albero calcola le sue dimensioni e la posizione in base alle dimensioni dei propri figli e alle proprietà del layout.

La fase di misurazione: in questa fase, ogni view nell'albero chiede ai propri figli di misurarsi, utilizzando i metodi onMeasure() e measure(). In questo modo, ogni view può calcolare le proprie dimensioni e posizioni in base alle misure dei propri figli.

La fase di disegno: infine, ogni view nell'albero disegna se stessa e i propri figli sullo schermo utilizzando i metodi onDraw() e draw().

Le tre fasi descritte avvengono ogni volta che cambia qualche elemento della UI e si cerca di richiedere un refresh. Come ad esempio un cambio di dimensione dello schermo, oppure un cambio di layout

Inoltre, durante la fase di layout e di disegno, le view possono anche ricevere e gestire eventi touch.

Come funzionano le animazioni? Si faccia un esempio.

Le animazioni in Android consentono di creare transizioni fluide tra due stati di una view o di un'intera schermata. Ci sono due modi principali per creare animazioni in Android:

Animazioni di sistema: sono animazioni predefinite fornite dal sistema Android e possono essere utilizzate per animare cambiamenti di posizione, dimensione, trasparenza e altro ancora. Ad esempio, puoi utilizzare un'animazione di sistema per far scorrere una view fuori dallo schermo o per ingrandire un'immagine quando viene toccata.

Animazioni personalizzate: sono animazioni create da te utilizzando il framework di animazione di Android. Con le animazioni personalizzate, puoi creare qualsiasi tipo di animazione che desideri, ad esempio una rotazione o un'oscillazione, utilizzando il codice Java o XML.

Ecco un esempio di come utilizzare un'animazione di sistema per far scomparire una view cliccata:

Copy code

```
view.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    // Carica l'animazione di sistema per far scomparire la view
    Animation fadeOut = AnimationUtils.loadAnimation(getBaseContext(), R.anim.fade_out);
    // Imposta l'animazione per la view
    view.startAnimation(fadeOut);
    // Dopo che l'animazione è terminata, nascondi la view
    view.setVisibility(View.GONE);
  }
});
```

Si descriva l'utilizzo di un sensore spiegando cosa si deve fare per utilizzarlo. Arricchire la spiegazione con prozioni di codice.

Per utilizzare un sensore in un'app Android, devi seguire questi passaggi:

Includere il permesso di lettura del sensore nel file AndroidManifest.xml dell'app:

<uses-permission android:name="android.permission.SENSOR_SERVICE"/>
Ottenere un'istanza del sensore dall'oggetto SensorManager:

SensorManager sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
Implementare una classe che estenda SensorEventListener e sovrascrivere i metodi onSensorChanged() e onAccuracyChanged():

```
private class MySensorEventListener implements SensorEventListener {
  @Override
  public void onSensorChanged(SensorEvent event) {
    // Gestisci gli eventi del sensore qui
  }

  @Override
  public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Gestisci i cambiamenti di accuratezza qui
  }
}
```

Registrare il listener per il sensore:

```
SensorEventListener sensorListener = new MySensorEventListener();
sensorManager.registerListener(sensorListener, sensor, SensorManager.SENSOR_DELAY_NORMAL);
```

Quando hai finito di utilizzare il sensore, assicurati di annullare la registrazione del listener per risparmiare le risorse del dispositivo:

```
sensorManager.unregisterListener(sensorListener);
```

In questo modo puoi utilizzare un sensore in un'app Android. Ogni volta che il sensore rileva un cambiamento, il metodo onSensorChanged() viene chiamato e puoi gestire gli eventi del sensore nel tuo codice.

Si discuta del meccanismo dei frammenti. Si faccia un esempio per illustrarne l'utilità.

I frammenti sono componenti dell'interfaccia utente di un'applicazione Android che possono essere riutilizzati in diverse attività. Ogni frammento rappresenta una parte dell'interfaccia utente e può essere gestito indipendentemente dalle altre parti dell'interfaccia utente. Ciò significa che è possibile utilizzare lo stesso frammento in più attività, rendendo più facile la creazione di interfacce utente complesse e flessibili.

Un esempio di utilizzo dei frammenti potrebbe essere un'app di e-commerce che include una scheda di navigazione con diverse opzioni per filtrare i prodotti in base al prezzo, alla categoria, ecc. Invece di creare un'interfaccia utente diversa per ogni opzione di filtro, è possibile creare un singolo frammento per il filtro e riutilizzarlo in tutte le attività dell'app. Ciò consente di mantenere un'interfaccia utente coerente e di ridurre la quantità di codice da scrivere e mantenere.