UNIVERSITÀ DI SALERNO

Corso di laurea in Informatica

Mobile Programming

Prof. De Prisco

Prova scritta del

esempio 3

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

Il posizionamento degli elementi di un layout avviene (anche) specificando delle misure. Quali sono le unità che si possono utilizzare? Si fornisca una breve descrizione per ognuna di esse.

In Android, è possibile specificare le misure degli elementi di un layout utilizzando diverse unità di misura. Ecco un elenco delle principali unità di misura che è possibile utilizzare:

dp (density-independent pixels): questa è un'unità di misura basata sulla densità dello schermo del dispositivo. Un dp equivale a un pixel su uno schermo con una densità di 160 dpi (dots per inch). L'utilizzo di dp consente di assicurare che gli elementi del layout abbiano lo stesso aspetto su dispositivi con diverse densità di schermo.

sp (scaled pixels): questa è un'unità di misura simile ai dp, ma viene utilizzata per la dimensione del testo. Uno sp equivale a un dp su uno schermo con una densità di 160 dpi, ma viene scalato in base alle impostazioni di grandezza del testo dell'utente.

pt (points): questa è un'unità di misura basata sulla risoluzione dello schermo. Un pt equivale a 1/72 di pollice.

in (inches): questa è un'unità di misura basata sulla dimensione fisica dello schermo. Un in equivale a 2,54 cm.

mm (millimeters): questa è un'unità di misura basata sulla dimensione fisica dello schermo. Un mm equivale a 1/10 di cm.

È importante notare che le unità di misura vengono utilizzate solo per specificare le dimensioni degli elementi del layout. Per posizionare gli elementi del layout, è possibile utilizzare diverse proprietà, come layout_gravity e layout_margin.

Completare il seguente CustomeAdapter per un ListView customizzato sapendo che è stato progettato per visualizzare informazioni su oggetti di tipo "Automobile" e che il fiel di layout R.layout.list_element prevede i seguenti campi: un ImageView (con id idFotoAuto), un TextView (con id idPrezzoAuto), un Button (con id idButtonAuto).
(Opzionale: Cosa comporta la presenza di un pulsante per un eventuale onItemClickListener del listview?)

```
public class CustomAdapter extends ArrayAdapter<Automobile> {
private LayoutInflater inflater;

   public CustomAdapter(Context context, int resourceId, List<Automobile> automobili) {
         super(context, resourceId, objects);
         inflater = LayoutInflater.from(context);
   }

   @Override
   public View getView(int position, View v, ViewGroup parent) {
          if (v == null) {
              v = inflater.inflate(R.layout.list_element, null);
          }

      Automobile automobile = getItem(position);

      ImageView fotoAuto = (ImageView) v.findViewById(R.id.idFotoAuto);
      fotoAuto.setImageResource(automobile.getFoto());

      TextView prezzoAuto = (TextView) v.findViewById(R.id.idPrezzoAuto);
      prezzoAuto.setText(String.valueOf(automobile.getPrezzo()));

      Button buttonAuto = (Button) v.findViewById(R.id.idButtonAuto);
      buttonAuto.setText(automobile.getButtonText());

      return v;
      }
      }
```

Si supponga di aver lanciato un'app A che al momento è l'unica visibile sul display del dispositivo. In che stato si trova l'activity A?

1. Si descriva una situazione per la quale l'app A vada temporaneamente nello stato di pausa per poi ritornare nello stato in cui si trovava prima (senza passare attraverso altri stati)
2. Si descriva una situazione per la quale l'app A vada nello stato di Stop per poi tornare nello stato in cui si trovava prima. Per quali stati dovrà passare?
3. Si descriva una situazione per la quale l'app A venga distrutta per poi essere immediatamente ricreata tornando nello stato in cui si trovava prima.

Se l'app A è visibile sul display del dispositivo e non viene interrotta da altre app o eventi del sistema, si trova nello stato di "running" o "foreground". Una situazione in cui l'app A potrebbe andare temporaneamente in pausa senza passare attraverso altri stati potrebbe essere il seguente: l'utente apre l'app A, quindi apre un'altra app B che copre temporaneamente l'app A. In questo caso, l'app A entra nello stato di "paused" finché l'utente non torna all'app A, ma non passa attraverso lo stato di "stopped" o "destroyed".

Se l'app A viene interrotta da altre app o eventi del sistema che richiedono risorse, come ad esempio una chiamata telefonica o una notifica push, l'app A può passare dallo stato di "paused" allo stato di "stopped". In questo caso, l'app A viene rimossa dalla memoria del dispositivo per liberare risorse per le altre app. Quando l'utente torna all'app A, essa viene ricreata passando prima attraverso lo stato di "created", quindi attraverso lo stato di "started" e infine attraverso lo stato di "resumed" prima di tornare nello stato di "foreground".

Se l'app A viene distrutta dal sistema per liberare risorse, ad esempio quando il dispositivo viene spento o quando l'utente la disinstalla, l'app A deve essere ricreata dall'inizio quando viene rilanciata. In questo caso, l'app passa attraverso gli stessi stati descritti in precedenza: "created", "started" e "resumed" prima di tornare nello stato di "foreground".

Un Intent è composto dalle seguenti parti: Action, Data, Category, Type, Component e Extras. Si fornisca una breve descrizione per ognuna di esse.

In Android, un Intent è un messaggio che viene utilizzato per richiedere un'azione o per avviare un componente di un'app, come ad esempio un'activity o un service. Un Intent è composto da diverse parti:

Action: questa è l'azione che si desidera richiedere. Ad esempio, l'azione "ACTION_VIEW" indica che si desidera visualizzare qualcosa.

Data: questo è il dato su cui si desidera eseguire l'azione. Ad esempio, potrebbe essere un URL o un percorso di file.

Category: questa è una categoria opzionale che può essere utilizzata per fornire ulteriori informazioni sull'Intent. Ad esempio, la categoria "CATEGORY_LAUNCHER" indica che l'Intent deve essere utilizzato per avviare l'app.

Type: questo è il tipo di dati che si desidera gestire con l'Intent. Ad esempio, potrebbe essere "image/jpeg" per indicare che si desidera gestire un'immagine JPEG.

Component: questo è il nome del componente dell'app che si desidera avviare con l'Intent. Ad esempio, potrebbe essere il nome di un'activity o di un service.

Extras: queste sono informazioni aggiuntive che possono essere incluse nell'Intent per fornire ulteriori dettagli o per passare dati ad altri componenti dell'app. Gli Extras possono essere di diversi tipi, come stringhe, numeri o oggetti. Possono essere aggiunti all'Intent utilizzando il metodo putExtra() e recuperati dal componente che riceve l'Intent utilizzando il metodo getIntent().get...Extra(), dove "..." indica il tipo di dato che si desidera recuperare.

State sviluppando un app “DoppioTris“ che utilizza un frammento Scacchiera che implementa opportunamente una scacchiera del gioco. Il file di layout prevede due contenitori per i due frammenti i cui id sono rispettivamente `idContenitore1` e `idContenitore1`. Completate il metodo onCreate creando i due frammenti Scacchiera e posizionandoli nei rispettivi contenitori.

L’oggetto Scacchiera prevede un metodo setAltra(Scacchiera s) che prende in input una Scacchiera s. Tale metodo serve perché ogni modifica alla scacchiera verrà fatta anche alla scacchiera s. In altre parole se si chiama s1.setAltra(s2), ogni modifica fatta a s1 verrà fatta anche a s2 – questo è codificato nel frammento stesso.

Dopo aver creato e posizionato i frammenti fare in modo che ognuna delle scacchiere sia la copia dell’altra.

```
public class Tris extends Fragment implements View.OnClickListener {
  //PER QUESTA CLASSE NON BISOGNA SCRIVERE NIENTE
  //occorre solo utilizzarale nel metodo onCreate
}

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Crea i due frammenti Scacchiera
        Tris scacchiera1 = new Tris();
        Tris scacchiera2 = new Tris();

        // Posiziona i frammenti nei contenitori
        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.add(R.id.idContenitore1, scacchiera1);
        fragmentTransaction.add(R.id.idContenitore2, scacchiera2);
        fragmentTransaction.commit();

        // Imposta ogni scacchiera come copia dell'altra
        scacchiera1.setAltra(scacchiera2);
        scacchiera2.setAltra(scacchiera1);
        }
```

Quali metodi conosci per memorizzare in maniera persistente i dati di un'applicazione (cioè in modo tale che non si perdano se l'app viene terminata e poi rilanciata nuovamente, anche dopo lo spegnimento e la riaccensione del dispositivo)? Fornire una breve descrizione di come utilizzarli e dei vantaggi e svantaggi di ognuno rispetto agli altri.

SharedPreferences: questo è un meccanismo di archiviazione key-value semplice che permette di memorizzare piccole quantità di dati primitivi, come stringhe, numeri o booleani. Le SharedPreferences possono essere condivise fra diverse componenti dell'app e sono salvate in file xml sul disco. Utilizzare le SharedPreferences è semplice: basta ottenere un'istanza delle SharedPreferences con il metodo getSharedPreferences(), quindi utilizzare i metodi put...() per memorizzare i dati e i metodi get...() per recuperarli.

File di testo: è possibile memorizzare i dati in un file di testo sul disco utilizzando le classi FileOutputStream e FileInputStream. Questo metodo è adatto per archiviare grandi quantità di dati o per archiviare dati strutturati, come ad esempio oggetti serializzati. Utilizzare i file di testo per memorizzare i dati richiede un po' più di codice rispetto alle SharedPreferences, poiché è necessario gestire manualmente l'apertura e la chiusura dei file e la serializzazione e deserializzazione dei dati.

Database SQLite: questo è un database relazionale leggero incluso in Android che permette di archiviare dati strutturati in tabelle e di eseguire query sui dati. Utilizzare il database SQLite per memorizzare i dati richiede un po' di lavoro iniziale per creare la struttura delle tabelle e per scrivere le query, ma offre una gran flessibilità e prestazioni elevate una volta impostato.

Ognuno di questi metodi ha i suoi vantaggi e svantaggi. Le SharedPreferences sono semplici da utilizzare ma adatte solo per piccole quantità di dati primitivi. I file di testo sono ad

atti per archiviare grandi quantità di dati o dati strutturati, ma richiedono più codice per l'utilizzo. Il database SQLite è flessibile e adatto per dati strutturati, ma richiede un po' di lavoro iniziale per la configurazione.

In generale, la scelta del metodo di memorizzazione dei dati dipende dalle esigenze specifiche dell'app. Se hai bisogno solo di memorizzare piccole quantità di dati primitivi, le SharedPreferences possono essere una buona scelta. Se hai bisogno di archiviare grandi quantità di dati o dati strutturati, i file di testo o il database SQLite potrebbero essere opzioni migliori.

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 MeasureSpecmtramite 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?

Il metodo onMeasure() viene utilizzato per determinare la dimensione reale di un elemento dell'albero delle view in Android. Per fare questo, viene utilizzato un oggetto MeasureSpec che codifica sia la dimensione in pixel dell'altezza e della larghezza dell'elemento che una "modalità".

I valori che il metodo getMode() può restituire sono i seguenti:

MeasureSpec.EXACTLY: questa modalità indica che la dimensione dell'elemento è nota e non può essere modificata. Ad esempio, se l'elemento ha una dimensione specificata in pixels nel file di layout, la modalità sarà MeasureSpec.EXACTLY.

MeasureSpec.AT_MOST: questa modalità indica che la dimensione dell'elemento non può superare una certa quantità, specificata dal metodo getSize(). Ad esempio, se l'elemento ha una dimensione massima specificata in pixels nel file di layout, la modalità sarà MeasureSpec.AT_MOST.

MeasureSpec.UNSPECIFIED: questa modalità indica che la dimensione dell'elemento non è nota e può essere qualsiasi valore. Questa modalità viene utilizzata di solito quando il genitore dell'elemento non impone alcuna limitazione sulla dimensione dell'elemento. Ad esempio, se l'elemento è una ScrollView che può contenere elementi di dimensioni variabili, la modalità potrebbe essere MeasureSpec.UNSPECIFIED.

I valori restituiti dal metodo getMode() possono essere utilizzati dal metodo onMeasure() per determinare come determinare la dimensione dell'elemento. Ad esempio, se la modalità è MeasureSpec.EXACTLY, l'elemento deve essere ridimensionato per adattarsi alla dimensione specificata. Se la modalità è MeasureSpec.AT_MOST, l'elemento può essere ridimensionato fino a quando non supera la dimensione massima specificata. Se la modalità è MeasureSpec.UNSPECIFIED, l'elemento può essere ridimensionato a qualsiasi dimensione.

Si spieghi come funziona il meccanismo del multitouch (parlando del MotionEvent e di come lo si gestice).

Il multitouch è una funzionalità che permette di riconoscere e gestire più punti di tocco contemporaneamente sullo schermo di un dispositivo Android. Ad esempio, con il multitouch è possibile zoomare su una immagine toccando lo schermo con due dita e allontanandole o avvicinandole.

Il meccanismo del multitouch viene gestito dall'oggetto MotionEvent in Android. Quando un utente tocca lo schermo con un dito, il sistema crea un oggetto MotionEvent contenente informazioni sulla posizione del tocco e lo invia all'app. Se l'utente tocca lo schermo con un secondo dito, il sistema crea un altro oggetto MotionEvent contenente informazioni sulla posizione di questo secondo tocco e lo invia all'app.

Per gestire il multitouch in un'app Android, è necessario implementare un metodo onTouchEvent() che gestisca gli oggetti MotionEvent inviati dal sistema. In questo metodo, è possibile ottenere informazioni sulla posizione dei tocchi utilizzando i metodi getX() e getY(), ottenere informazioni sulla quantità di dita che toccano lo schermo utilizzando il metodo getPointerCount(), e ottenere informazioni su ogni dito individualmente utilizzando il metodo getPointerId().

Quali sono le forme di “notifica” standard messe a disposizione da Android? Fornire una breve descrizione.

In Android ci sono diverse forme di notifica standard a disposizione degli sviluppatori per comunicare con gli utenti. Ecco un elenco delle forme di notifica più comuni:

Notifiche push: queste sono notifiche inviate da un server a un'app su un dispositivo Android. Le notifiche push possono essere utilizzate per inviare aggiornamenti o promemoria agli utenti anche quando l'app non è in esecuzione.

Toast: sono piccole notifiche che appaiono brevemente sullo schermo e scompaiono automaticamente. Sono utili per fornire feedback agli utenti senza interrompere l'esperienza di utilizzo dell'app.

Dialog: sono finestre di dialogo modali che appaiono sullo schermo per richiedere l'attenzione dell'utente. Possono essere utilizzati per mostrare messaggi o per chiedere all'utente di effettuare una scelta.

Status bar notifications: sono notifiche che appaiono nella barra di stato del dispositivo e possono essere utilizzate per informare gli utenti di eventi importanti o per fornire promemoria.

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?

Le altre tre componenti fondamentali di Android sono:

Service: un service è un componente che viene eseguito in background per svolgere operazioni che non sono strettamente legate all'interfaccia utente dell'app. Ad esempio, un service potrebbe essere utilizzato per scaricare file in background o per eseguire operazioni di sincronizzazione dati con un server.

Broadcast receiver: un broadcast receiver è un componente che viene utilizzato per ricevere e gestire i messaggi broadcast inviati dal sistema o da altre app. Ad esempio, un broadcast receiver potrebbe essere utilizzato per ricevere una notifica quando il dispositivo viene spento o quando la connessione a Internet viene interrotta.

Content provider: un content provider è un componente che viene utilizzato per gestire e condividere dati tra diverse app. Ad esempio, un content provider potrebbe essere utilizzato per gestire i contatti della rubrica o per accedere ai file multimediali presenti sul dispositivo.

Prova scritta

Pagina aggiuntiva

Prova scritta

Pagina aggiuntiva

Prova scritta

Pagina aggiuntiva