**Layout**

**Il layout di un app può essere definito sia staticamente, tramite un file XML, che programmaticamente tramite istruzioni nel programma. Si discuta dei vantaggi e svantaggi e si faccia un esempio di un caso in cui è possibile usare solo uno dei due e non l’altro, motivando la risposta.**

Un layout definisce la struttura per un'interfaccia utente nell'app, ad esempio in un'attività . Tutti gli elementi nel layout sono creati usando una gerarchia di oggetti ***View*** e ***ViewGroup***. È possibile dichiarare un layout in due modi:

* ***Dichiarare elementi dell'interfaccia utente in XML***: Android offre un semplice vocabolario XML che corrisponde alle classi e alle sottoclassi di visualizzazione, come quelle per *widget* e *layout*.

***VANTAGGI***: facile da specificare e separa i modo netto la definizione dell’UI dal codice dell’app (facile fare modifiche).

***SVANTAGGI***: elementi statici.

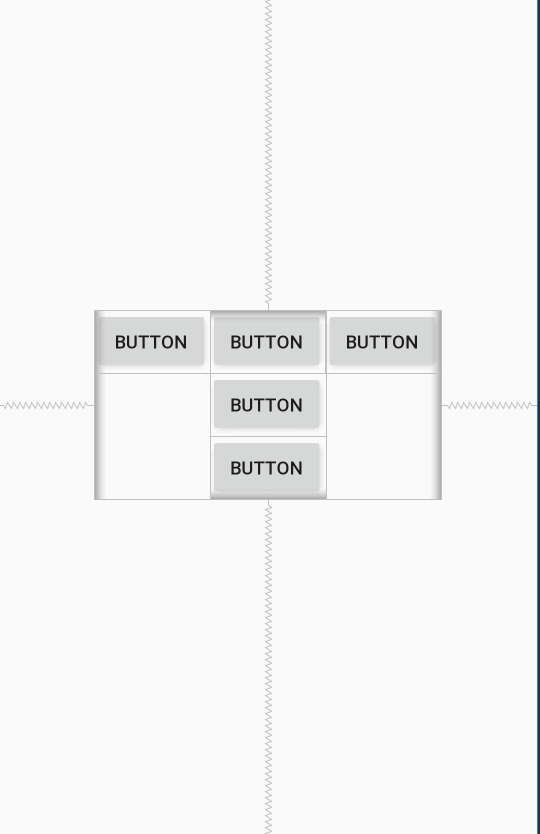
* ***Crea un'istanza di elementi di layout in fase di esecuzione***: l’app può creare oggetti *View* e *ViewGroup* (e manipolarne le proprietà) a livello di codice.

***VANTAGGI***: dinamico, facilmente adattabile.

***SVANTAGGI***: gestione del layout nel codice dell’applicazione.

***Es:*** Si consideri lo sviluppo di una calcolatrice in ***portrait***. In questo caso l’uso dell’XML soddisfa il programma in quanto una volta definita la struttura della calcolatrice, non è necessario apportare modifiche dinamicamente.

**Disegnare nell’apposito spazio sulla destra (che corrisponde al RelativeLayout esterno) i widgets specificati dal seguente codice XML.**



<**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"**>

<**LinearLayout**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:layout\_centerHorizontal="true"**

**android:layout\_centerVertical="true"**

**android:orientation="horizontal"**>

<**Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text**="**Button**" />

<**LinearLayout**

**android:layout\_centerHorizontal="true"**

**android:layout\_centerVertical="true"**

**android:orientation="vertical"**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**>

<**Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text**="**Button**" />

<**Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text**="**Button**" />

<**Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text**="**Button**" />

</**LinearLayout**>

<**Button**

**android:layout\_width="wrap\_content"**

**android:layout\_height="wrap\_content"**

**android:text**="**Button**" />

</**LinearLayout**>

</**RelativeLayout**>

**Ogni widget del layout prevede degli attributi che determinano parte della visualizzazione dello stesso widget all’interno dello schermo. Non tutti gli attributi possono essere applicati a tutti i widget; tuttavia ci sono degli attributi che si applicano a tutti i widget. Fra questi quali ritieni i più significativi o comunque i più utilizzati? Motivare la risposta.**

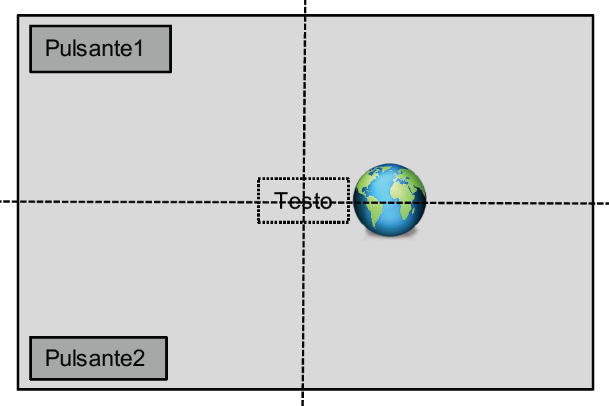
Ogni elemento (View o ViewGroup) supporta degli attributi che specificano l’aspetto grafico, dove visualizzare l’elemento e fornire ulteriori informazioni. Ci sono degli attributi che sono comuni a tutti gli elementi.

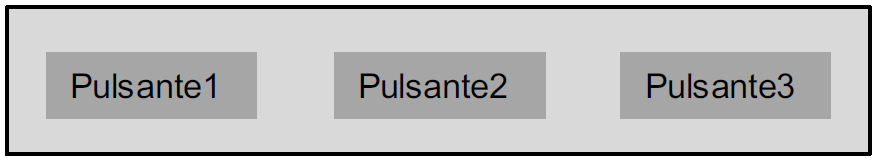
***Es:***

* “*android:id*”: che permette di assegnare un identificativo univoco agli elementi della UI. In questo modo, in una fase programmatica, si potrà ottenere il riferimento a quella variabile, per poter modificare in modo dinamico lo stato dell’oggetto.
* *“android:layout\_height”* – *“android:layout\_weight”:* il primo specifica l’altezza di base della View, è un attributo obbligatorio per qualsiasi View all’interno del layout. Il secondo specifica la larghezza di base della View, ha le stessa specifica del precedente. Ci sono varie unità disponibili (px, dp, etc…) per entrambi.
* *“android:background”*: Consente di definire il colore di sfondo di un widget (utile per verificare se il widget è stato inserito correttamente nel layout).
* “*android:margin*”: viene utilizzato per creare spazio attorno agli elementi, al di fuori di qualsiasi bordo definito.
* “*android:padding*”: viene utilizzato per generare spazio attorno al contenuto di un elemento.

**Si scriva un frammento di codice XML che descriva il lavout raffigurato a fianco scegliendo opportunamente gli elementi da utilizzare (le linee punteggiate non sono elementi del layout ma indicano il centro del layout).**

<RelativeLayout   
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Pulsante1"  
 android:layout\_alignParentLeft="true"/>  
  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Pulsante2"  
 android:layout\_alignParentBottom="true"/>  
  
 <TextView  
 android:id="@+id/TV"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:text="Testo"/>  
  
 <ImageView  
 android:layout\_toRightOf="@id/TV"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:layout\_centerInParent="true"  
 android:src="@drawable/world1" />  
</RelativeLayout>



**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).**

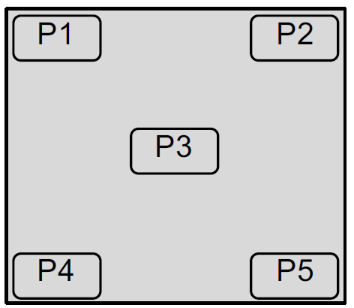
**Si consideri il LinearLayout indicato in figura:**

Button but= new Button(this);  
but.setText("Pulsante3");  
LinearLayout ll= findViewById(R.id.*ll1*);  
LinearLayout.LayoutParams lp= new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.*WRAP\_CONTENT*,

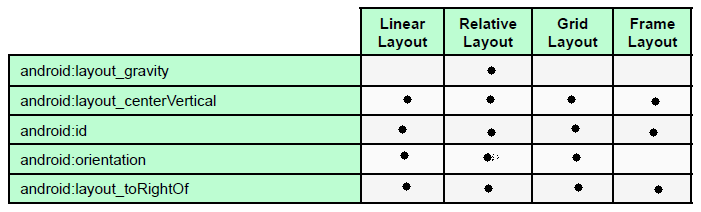
LinearLayout.LayoutParams.*WRAP\_CONTENT*);  
lp.setMargins(30,30,30,30);  
but.setLayoutParams(lp);  
ll.addView(but, lp);

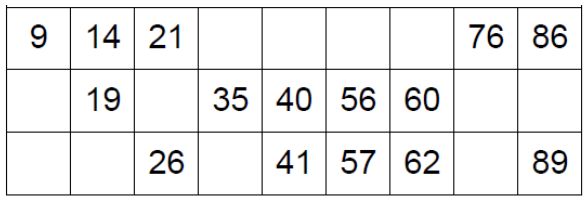
<LinearLayout   
 android:id="@+id/ll1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="15dp"  
 android:background="#DCDCDC"  
 tools:context=".MainActivity">  
  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30px"  
 android:text="Pulsante1" />  
  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30px"  
 android:text="Pulsante2" />  
</LinearLayout>

**Scrivere un file di layout che implementi l’interfaccia riportata in figura. Il rettangolo è un frame di 200x200px e deve essere posizionato al centro del display. I riquadri con etichetti “Pn” sono dei pulsanti la cui pressione deve essere collegata alla funzione “pulsantePremuto”**



**Nella seguente tabella sono riportati alcuni tipi di layout (sulle colonne) ed alcuni attributi (sulle righe). Indicare con una X all’incrocio fra riga e colonna gli attributi che hanno senso per lo specifico layout. Si fornisca anche una breve motivazione per le risposta inserite nella tabella.**

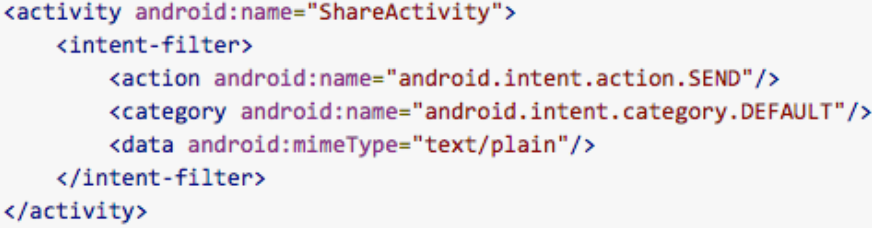




**Si consideri un’app che implementa le cartelle del gioco della Tombola. Un esempio di cartella è riprodotto in figura; la posizione delle casella vuote cambia da cartella a cartella. L’app deve essere in grado di gestire qualunque layout e anche più di una cartella contemporaneamente.**

**Si descriva (senza codice) un modo per implementare il layout giustificano la scelta. Si tenga presente che, plausibilmente, l’app deve avere un riferimento per tutte le caselle che contengono un numero.**

**Nel file manifest.xml è possibile specificare un “intent-filter”, ad esempio:**

****

**A cosa serve? Si dia sia una risposta generale (a cosa serve un intent-filter), sia una risposta per il caso specifico dell’esempio.**

**Si consideri un’app il cui layout prevede 4 pulsanti ai 4 angoli del display. Tale layout può essere implementato in vari modi. Descrivine uno (a parole, senza codice XML, spiegando solo come dovrebbe essere fatto il codice XML) mettendone in evidenza eventuali vantaggi o svantaggi.**

Utilizziamo un RelativeLayout specificando per width e height il valore match\_parent per prendere tutto il display a disposizione. Si comincia a creare il primo button che deve essere posizionato in alto a sinistra. Da questo momento in poi ogni width e height viene eguagliato a wrap\_content. Sfruttando il RelativeLayout impostiamo il “alignParentLeft” uguale a true, in questo modo il bottone verrà posizionato in alto a sinistra. Discorso simile per il bottone in alto a destra con la differenza che verrà usato “alignParentRight” eguagliato a true. Per il bottone che verrà posizionato in basso a sinistra e necessario, oltre a mettere alignParentLeft, anche “alignParentBottom” sempre a true. Per l’ultimo, cioè quello posizionato in basso a destra sarà sempre necessario mettere alignParentRight e alignParentBottom, entrambi a true.

**A cosa serve il file Manifest.xml?**

Tutte le app Android hanno un file *manifest* comunemente noto come “***AndroidManifest.xml***” nella root del sorgente del progetto. Descrive le informazioni essenziale dell’app. È richiesto per dichiarare le seguenti informazioni:

* Dichiara il livello minimo di API richiesto dall’applicazione
* Dichiara le autorizzazioni richieste dall’applicazione, ad es. Videocamera, posizione, etc..
* Fornisce autorizzazioni per le funzionalità hardware e software utilizzate o richieste dall’applicazione
* Elenca le librerie a cui l’applicazione deve essere collegata

Ci sono ulteriori informazioni che vengono riportate come:

* Nome dell’applicazione: titolo dell’app (*android:label*= “*nomeApp*”)
* Nome del pacchetto: è un nome univoco utilizzato per identificare l’app
* Icona dell’applicazione: riferimento all’icona visualizzata nella schermata principale di Android per l’app
* Numero versione: è un singolo numero utilizzato per mostrare una versione dell’app (*android:versionCode*= “*1*”)
* Versione minima di Android: rappresenta la versione minima richiesta di Android per far funzionare l’app

**Gli ambienti di sviluppo app per Android offrono la possibilità di usare un emulatore per eseguire le app. Quali sono i vantaggi e gli svantaggi dell’utilizzo di un emulatore?**

Un emulatore duplica ogni aspetto del comportamento del dispositivo originale, sia hardware che software

L’emulatore Android ha come vantaggio la facilità di creazione di determinate situazioni, quali batteria scarica o arrivo di un messaggio. Gli svantaggi sono:

* Lento (a volte molto), alcune operazioni sono difficoltose
* È comunque un “simulatore”
* Possono esserci dei bug
* Un emulatore di dispositivo mobile non prende in considerazione fattori come surriscaldamento / drenaggio della batteria o conflitti con altre app (predefinite)
* Gli emulatori possono supportare solo determinate versioni del sistema operativo

**EX:** I real device hanno come vantaggio di essere veloci e l’input è facile da gestire (es. rotazioni display). Inoltre l’esecuzione è reale

**A cosa serve la misurazione degli elementi grafici in “dpi” (density independent pixels). Si discuta del perché è necessaria e di come una cattiva gestione possa influire negativamente sull’aspetto grafico di un’app.**

Per preservare la dimensione visibile dell'interfaccia utente su schermi con densità diverse, è necessario progettare l'interfaccia utente utilizzando pixel indipendenti dalla densità (dp) come unità di misura. Un dp è un'unità di pixel virtuale che è all'incirca uguale a un pixel su uno schermo a media densità (160 dpi; la densità "base"). Android traduce questo valore nel numero appropriato di pixel reali per ogni altra densità.

**ES:** Si considerino due schermi della stessa dimensione che hanno numero diversi di pixel. Se dovessi definire una vista larga "100px", apparirà molto più grande su uno dei due dispositivi, a seconda dei pixel a disposizione. Quindi si deve invece usare "100dp" per assicurarsi che appaia la stessa dimensione su entrambi gli schermi.

**Quali sono i widget standard per la costruzione dell’interfaccia utente? E quali sono i principali layout per il posizionamento dei widget? Per ognuno dei layout si descriva brevemente il meccanismo di posizionamento dei widget all’interno del layout.**

I widget standard per la costruzione dell’interfaccia utente sono: ***TextView***, ***Button***, ***TextEdit***, ***ImageView***, ***CheckBox***, ***RadioButton***. I principali layout per il posizionamento dei widget sono:

* ***LinearLayout***: organizza i suoi figli in un’unica riga orizzontale o verticale. Crea una barra di scorrimento se la lunghezza della finestra supera la lunghezza dello schermo.
* ***RelativeLayout***: Consente di specificare la posizione degli oggetti figlio l'uno rispetto all'altro (figlio A alla sinistra del figlio B) o al genitore (allineato alla parte superiore del genitore).

Si possono creare Layout tramite un Adapter:

* ***ListView***: Visualizza un elenco a colonna singola scorrevole.
* ***GridView***: Visualizza una griglia scorrevole di colonne e righe.

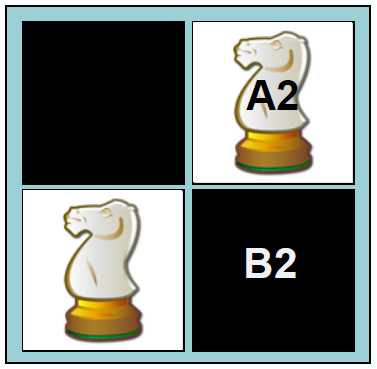
**In un progetto Android come sono organizzati i file per lo sviluppo di un’app? Elencare le principali cartelle/file e descrivere brevemente il loro contenuto.**

L’organizzazione dei file per lo sviluppo di un’app è caratterizzata da varie directory. La prima è la root (app), all’interno devono essere presenti le seguenti (l’ordine delle directory è importante):

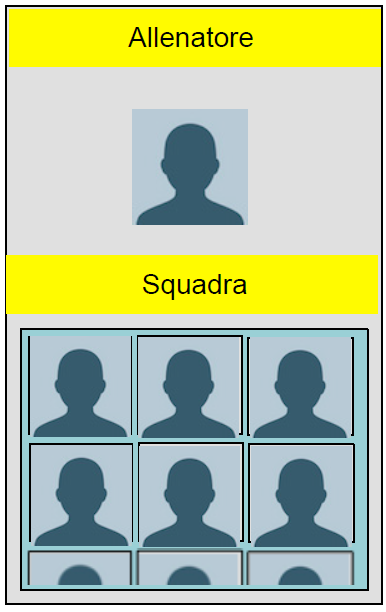
* ***manifests***: Contiene il file di manifesto che contiene le informazioni essenziali dell’app
* ***java***: contiene il codice sorgente per programmare l’app. Fornisce la dinamicità
* ***res***: fornisce le risorse dell’app, ci sono al suo interno altre directory, come layout che fornisce la struttura statica dell’applicazione (formato xml), i drawable all’interno del quale verranno messi immagini e file multimediali, mipmap che contiene le varie icone dell’app a seconda della risoluzione del dispositivo, values contiene al suo interno file xml che forniscono metadati.

**Si scriva un file di layout per una scacchiera per il gioco degli scacchi. Ogni singola cella della scacchiera deve essere un Frame nel quale si sovrappongono sia un TextView che un ImageView. Il colore di background della cella è stabilito nel frame. Il colore del testo nel TextView deve essere l’opposto di quello della cella. Sia l’immagine che il testo possono essere presenti o meno (si assume che verranno gestiti programmaticamente).**

**Per semplicità si usi una scacchiera 2x2, come mostrato nella figura. Si discuta di come ampliare, sempre in modo statico ma evitando di ripetere la singola cella per 64 volte, il layout al caso 8x8.**

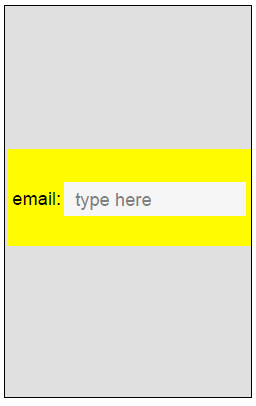


**Si scriva un file di layout che determini l’aspetto grafico mostrato nella figura. Il riquadro esterno rappresenta tutto lo schermo con un sfondo di colore grigio chiaro. La foto dell’allenatore è una singola immagine, mentre le foto dei componenti della squadra sono visualizzate in un gridView. Il colore di sfondo nella parte dei titoli è giallo, mentre il colore di sfondo del gridView è azzurro (per specificare i colori è sufficiente scrivere qualcosa del tipo @color/azzurro).**

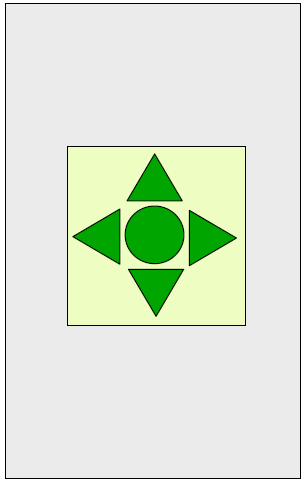


**DESCRIVERE un file di layout per un app che permette di giocare scacchi. Il layout deve costruire la scacchiera e deve permettere di visualizzare sia le caselle vuote sia le caselle con i pezzi del gioco. Si assuma che i pezzi del gioco siano in forma di immagini. Nota: DESCRIVERE significa indicare la struttura ed eventuali dettagli rilevanti su come si deve effettivamente scrivere il file.**

**Si scriva un file di layout che determini l’aspetto grafico mostrato nella figura. Il riquadro esterno rappresenta tutto lo schermo con un sfondo di colore grigio chiaro, mentre il contenitore centrale con uno sfondo giallo ha un’altezza di 200dp. Il testo “email:” è visualizzato in un TextView. Di seguito c’è un EditText con il suggerimento “type here”**



**Si scriva un file di layout statico che riproduca l’interfaccia grafica riportata in figura (si assuma di avere a disposizione 5 file (left.png, right.png, up.png, down.png, circle.png) con gli elementi graficiriportati in figura. Il rettangolo interno è un frame con un background colorato, mentre quello esterno è lo schermo del dispositivo**



**Si completi il textoXML che descrive l’interfaccia utente specificata nel disegno a destra. Il RelativeLayout già presente si riferisce all’intero schermo.**

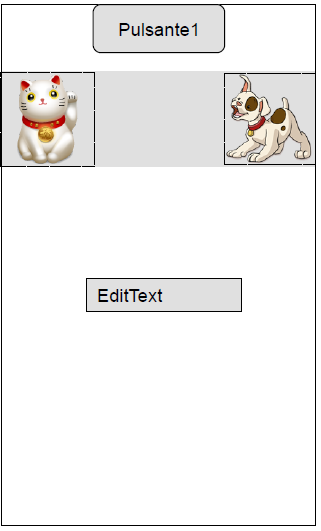
***<?xml version="1.0" encoding="utf-8"?>***

***<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android”***

***android:background=”#FFFFFF"***

***android:layout\_width=”match\_parent"***

***android:layout\_height=”match\_parent”>***



**ListView**

**Un listview prevede on OnItemClickListener che gestisce i click sugli elementi della lista chiamando il metodo onItemClick al quale viene passato un riferimento dell’elemento selezionato. Se usiamo un listview customizzato, in cui ogni elemento della lista è composto da vari sottoelementi (es. una foto, un nome, un numero), il riferimento passato al metodo onItemClick non distingue quale dei sottoelementi è stato selezionato. Come si può fare per reagire in maniera diversa in funzione di quale dei sottoelementi è stato selezionato con il click?**

**Se si usa un listView customizzato con vari campi, ed esempio una foto, un nome ed un numero di telefono per ogni elemento della lista, usando un apposito file di layout, il metodo onItemClickListener del listView non riesce a distinguere i vari campi in quanto è definito per tutto l’elemento. Descrivere un modo per implementare un onClickListener sui singoli campi. Un metodo consiste nel definire un onClickListener per ogni singolo campo. Tuttavia c’è un problema. Quale? Descrivere un modo per risolverlo.**

**Il seguente codice incompleto è un CustomAdapter per una lista customizzata di oggetti Object. Ogni oggetto Object possiede i getter getString() e getInt(). Il file di layout “list\_element” contiene 2 TextView con i seguenti identificativi: “stringa” e “intero”. Completare il CustomAdapter per creare la view di ogni singolo elemento.**

*Public class CustomAdapter extends ArrayAdapter<Object> {*

*private int resource;*

*private LayoutInflater inflater;*

*public CustomAdapter(Context context, int resourceId, List<Objects> objects) {*

*super(context, resourceId, objects);*

*resource = resourceId;*

*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);*

*}*

*//----------------------------------------------------------------------------------------------------------------*

*Object o = getItem(position);*

*TextView tv1;*

*TextView tv2;*

*tv1= (TextView) v.findViewById(R.id.stringa);*

*tv2= (TextView) v.findViewById(R.id.intero);*

*tv1.setText(o.getString());*

*tv2.setText(o.getInt());*

*tv1.setTag(position);*

*tv2.setTag(position);*

*return v;*

*}*

**Descrivere schematicamente tutto ciò che serve per far comparire sullo schermo un ListView (semplice) con una lista di nomi (definiti in un array). Si specifichi per ogni passo l’istruzione, il widget o il file XML, necessari per il ListView.**

* Si definisce nel file xml (activity\_main) un widget ListView, specificando l’id (mylistview) e i vari parametri:

*<ListView*

*android:id=”@+id/mylistview”*

*android:layout\_width=”wrap\_content”*

*android:layout\_width=”wrap\_content”*

*</ListView>*

* Si crea un ulteriore file xml per rappresentare il singolo elemento della lista:

*<TextView*

*android:id=”@+id/textViewList”*

*android:layout\_width=”wrap\_content”*

*android:layout\_height=”wrap\_content”*

*android:text=”” android:padding=”10dp”*

*android:textSize=”22dp”/>*

* All’interno del file Java MainActivity si definisce un Array di stringhe contente nomi di persone:

*String [] array = {“Pasquale”,”Maria”,”Michele”,”Antonella”, “Vincenzo”}*

* Si definisce un ArrayAdapter, dove *context* rappresenta il contesto dell’applicazione, *R.layout.list\_element* rappresenta il file di layout dove viene definito il singolo elemento nella lista, *R.id.textViewList* che rappresenta l’identificativo del TextView nel file di layout e poi c’è *array* che rappresenta le informazioni da visualizzare nel ListView:

*ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(context, R.layout.list\_element, R.id.textViewList, array);*

* Viene individuato il widget ListView:

*listView = (ListView)findViewById(R.id.mylistview);*

* Infine si associa l’adapter al widget:

*listView.setAdapter(arrayAdapter);*

**Che cosa è un ListView? Che cosa è un Adapter? In che modo ListView e Adapter interagiscono?**

***ListView*** è un Widget specifico per le liste, raggruppa diversi elementi e li visualizza in un elenco scorrevole verticale. Gli elementi vengono automaticamente inseriti nell’elenco utilizzando un ***Adapter*** che estrae il contenuto da un’origine come un array o un DB.

L’***Adapter*** è un componente che si occupa della rappresentazione grafica dei dati e dell’interazione con essi, per ogni elemento della ***ListView***.

L’***Adapter*** viene assegnata a ***ListView*** tramite il metodo ***setAdapter*** sull’oggetto ***ListView***: *listView.setAdapter(arrayAdapter)*

**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.**

Indica la posizione della View nell’Adapter

**Che cosa è un Adapter customizzato per un ListView? Si fornisca una spiegazione possibilmente con frammenti di codice.**

Un Adapter customizzato nasce dall’esigenza di creare una visualizzazione personalizzata. Si inizia con l’aggiunta di un Widget ListView all’interno del file xml (activity\_main). Si crea un nuovo file xml dove si definisce il contenuto di ogni riga della lista.

Si crea una classe CustomAdapter che estende ArrayAdapter<Oggetto>. Al costruttore gli si passa un oggetto context, un intero resourceId, e la lista degli oggetti che verranno visualizzati. Si fa l’Override del metodo getView.

*public class CustomAdapter extends ArrayAdapter<Contatto> {*

*private int resource;*

*private LayoutInflater inflater;*

*public CustomAdapter(Context context, int resourceId, List<Contatto> objects) {*

*super(context, resourceId, objects);*

*resource = resourceId;*

*inflater = LayoutInflater.from(context);*

*}*

*@Override*

*public View getView(int position, View v, ViewGroup parent) {*

*if (v == null) {*

*Log.d("DEBUG","Inflating view");*

*v = inflater.inflate(R.layout.list\_elem, null);*

*}*

*Contatto c = getItem(position);*

*TextView nameTextView;*

*TextView telTextView;*

*ImageView fotoImageView;*

*nameTextView = (TextView) v.findViewById(R.id.tvPersona);*

*telTextView = (TextView) v.findViewById(R.id.tvNumero);*

*nameTextView.setText(c.getNome());*

*telTextView.setText(c.getNumero());*

*return v;*

*}*

*}*

**Il seguente frammento di codice serve a gestire un listView con gli elementi di un array. Completare il codice dell’onItemClickListener (riquadro vuoto) in modo tale che quando l’utente clicca un elemento visualizzato nel listView venga mostrato un Toast con il nome cliccato.**

*@Override*

*protected void onCreate(Bundle savedInstanceState) {*

*super.onCreate(savedInstanceState);*

*setContentView(R.layout.activity\_main);*

*String [] nazioni = {”Italia",”Francia",”Spagna",”Germania", ”Portogallo", ”Svizzera", ”Belgio", ”Polonia", ”Grecia", ”Svezia", ”Irlanda"};*

*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 n, long label) {*

*//--------------------------------------------------------------------------------------------------------*

***String str = listView.getItemAtPosition(position).toString();***

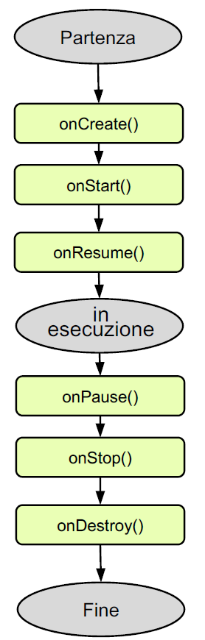
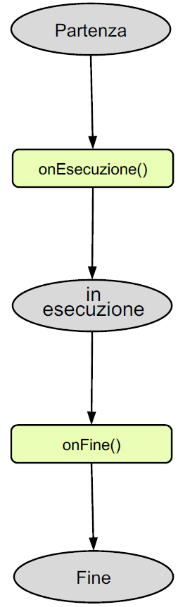
***// Show Toast***

***Toast.makeText(getApplicationContext(), “Nazione :" +str, Toast.LENGTH\_LONG).show();***

*}*

*});*

*}*

**QUESITO 3 (IL CICLO DI VITA)**

**II ciclo di vita delle activity, riportato schematicamente a sinistra, prevede l’esecuzione in successione di 3 metodi (onCreate, onStart, onResume) per far partire l’esecuzione di un’app. Perché? Non sarebbe stato meglio avere un solo metodo, come indicato nella figura a destra, nel quale eseguire tutto ciò che viene fatto nei 3 metodi onCreate, onStart, onResume?**

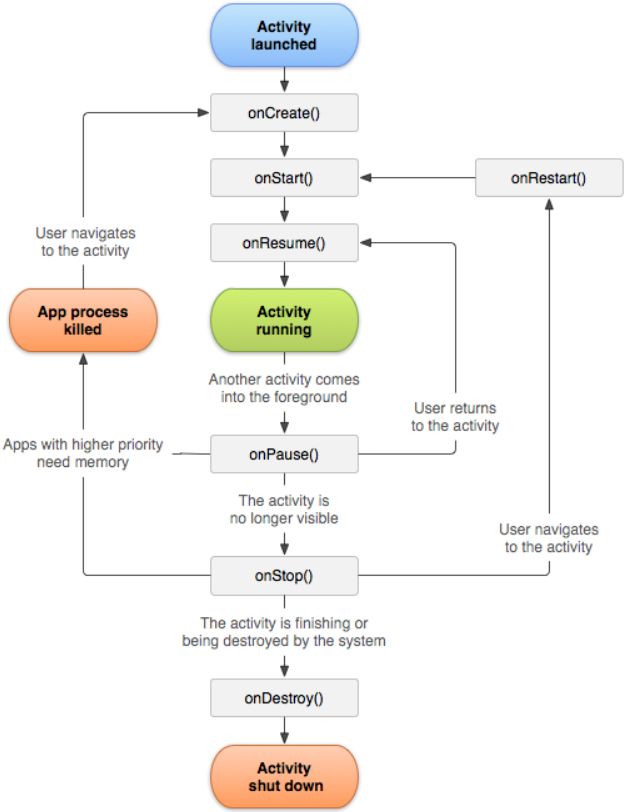
**Analogamente per distruggere un app è prevista l’esecuzione di 3 metodi in successione (onPause, onStop, onDestroy). Non sarebbe stato più semplice avere un solo metodo come indicato nella figura a destra?**

**Motivare la risposta.**

Ogni callback consente di eseguire lavori specifici appropriati per un determinato cambio di stato. Fare il lavoro giusto al momento giusto e gestire correttamente le transizioni rende l’app più solida e performante. Ad esempio, una buona implementazione dei callback del ciclo di vita può aiutare a garantire che l'app eviti:

* Arresto anomalo se l'utente riceve una telefonata o passa a un'altra app durante l'utilizzo dell'app.
* Consumo di preziose risorse di sistema quando l'utente non lo utilizza attivamente.
* Perdere i progressi dell'utente se escono dall'app e ci ritornano in un secondo momento.
* Arresto anomalo o perdita dei progressi dell'utente quando lo schermo ruota tra orientamento orizzontale e verticale.

**Si completi il disegno sottostante che rappresenta il ciclo di vita di una activity. Si descriva un’operazione che è solitamente effettuata in onStart(), con la corrispondente operazione effettuata in onStop(), e un’operazione solitamente effettuata in onResume(), con la corrispondente operazione effettuata in on Pause().**

La callback *onStart()* rende l’attività visibile all’utente, mentre si prepara ad entrare in primo piano e diventare interattiva. È uno stato in cui vengono elaborate le risorse necessarie per l’avvio.

Si entra nella callback *onStop()* quando un’applicazione non viene usata per molto tempo. Quando l’utente naviga tra le app che si trovano nel backstack e decide di riaprire l’app che si trova in *onStop(),* viene invocata la callback *onRestart()* che a sua volta invoca *onStart(),* cosicché l’app passi in foreground e diventi interattiva.

La callback *onResume()*, permette ad un’applicazione di passare in foreground quando le risorse necessarie sono state già caricate precedentemente.

Dallo stato *onPause()* si passa a quello *onResume()* quando si riapre un’applicazione che è stata messa in background da poco tempo, quindi senza necessità di dover rielaborare le risorse necessarie al suo avvio.

**Si spieghi come un’app possa eseguire la seguente sequenza di metodi**

1. onCreate()

2. onStart()

3. onResume()

4. onPause()

5. onStop()

6. onRestart()

7. onStart()

8. onResume()

9. onPause()

10. onResume()

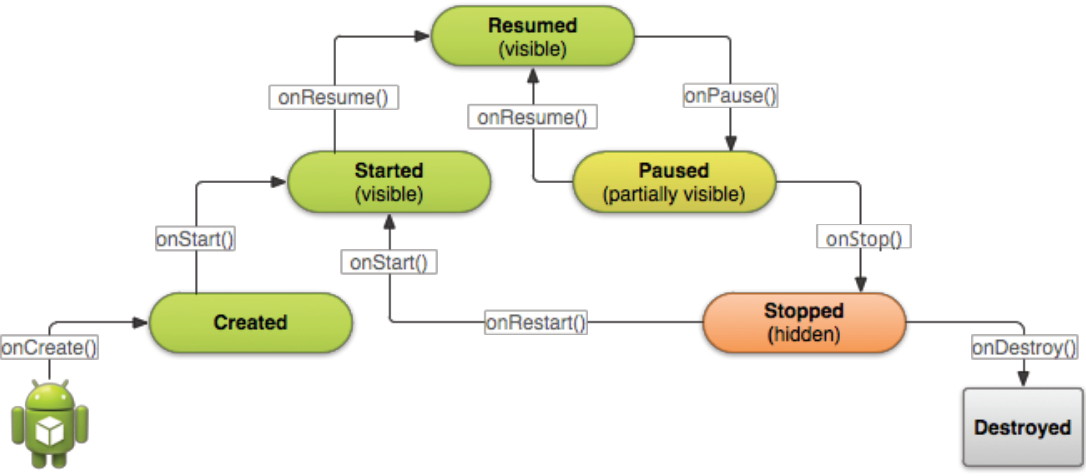
**Che cosa è il backstack? Supponendo che l’activity A sia l’unica in esecuzione (l’unica presente nel backstack), che la stessa app lancia una nuova activity B che a sua volta lancia l’activity C, che lancia l’activity D, quale è il backstack a questo punto? Cosa succede se dall’activity D si preme il pulsante di “back”? Cosa si deve fare se si vuole fare in modo che dall’activity D, si torni direttamente ad A quando si preme il pulsante di back?**

Un task è una raccolta di attività con cui gli utenti interagiscono durante l'esecuzione di un determinato lavoro, disposte in uno stack denominato *backstack*. Lo stack ha una struttura LIFO e memorizza le attività nell’ordine della loro apertura. Le attività nello stack non vengono mai riorganizzate. La navigazione dello backstack viene eseguita con l’aiuto del pulsante *back*.

La situazione nel back è la seguente: D, C, B , A. Se si preme back, l’attività D viene distrutta e C viene messa in foreground.

Impostare le activity B e C come FLAG\_ACTIVITY\_NO\_HISTORY, che non memorizza l’attività nello stack delle attività.

**Dal momento in cui un’Activity viene lanciata si può trovare in 5 diversi stati. Quali sono questi stati? Quali sono i metodi che permettono di passare dall’uno all’altro? Quali di questi stati sono “transienti” (cioè l’activity rimane per pochissimo tempo in questi stati) e quali invece più “duraturi” (cioè l’activity può rimanere anche per molto tempo in questi stati)?**



Transienti: Created, Started, Pauses

Duraturi: Resumed, Stopped

**Un dispositivo Android può funzionare sia in modalità portrait (verticale) landscape (orizzontale). Quando un dispositivo Android viene ruotato si passa dall’una all’altra modalità. Per gestire in maniera appropriata tali passaggi, di cosa si deve preoccupare il programmatore?**

Quando l’utente ruota il dispositivo, l’attività viene prima eliminataeseguendo le callback: *onPause()* 🡪 *onStop()* 🡪 *onDestroy()*, e poi ricreata: *onCreate()* 🡪 *onStart()* 🡪 *onResume().* Dunque si ha una perdita di stato.

Per gestire in maniera appropriata tali passaggi, si salva lo stato in *onSaveInstanceState()* in cui si invocano i put per salvare le informazioni che si vogliono mantenere e alla fine viene richiamato il metodo della superclasse *onSaveInstanceState* in cui si conserva l’oggetto Bundle:

*@Override*

*public void onSaveInstanceState(Bundle savedInstanceState) {*

*// Salvare lo stato dell’app*

*savedInstanceState.putStringArrayList("LISTA\_STRINGHE", array\_di\_stringhe);*

*savedInstanceState.putInt("CONTATORE", counter);*

*// Always call the superclass so it can save the view hierarchy state*

*super.onSaveInstanceState(savedInstanceState);*

*}*

Lo si recupera in *onCreate()* tramite un controllo preliminare e i get.

*@Override*

*protected void onCreate(Bundle savedInstanceState) {*

*…*

*if (savedInstanceState != null) {*

*array\_di\_stringhe = savedInstanceState.getStringArrayList("LISTA\_STRINGHE");*

*counter = savedInstanceState.getInt("CONTATORE");*

*}*

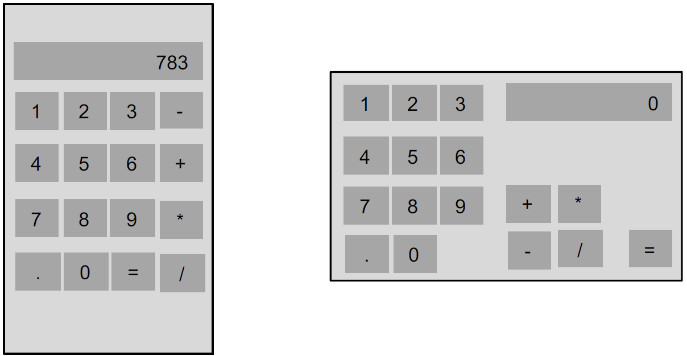
*}*

**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?**

Un’app viene lanciata e l’utente interagisce: Stato, *partenza*, *onCreate()* 🡪 *onStart()* 🡪 *onResume().* Stato: in *esecuzione*.

Rotazione dispositivo: Stato: *in esecuzione, onPause()* 🡪 *onStop()* 🡪 *onDestroy().* Stato: Fine, *onCreate()* 🡪 *onStart()* 🡪 *onResume().* Stato: in *esecuzione*.

**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 ha sbagliato il programmatore dell’app? Cosa avrebbe dovuto fare per ovviare all’errore?**

****

Il dispositivo è stato ruotato da *portrait* a *landscape* di conseguenza ne è cambiato l’orientamento. Dato che il valore 783 è diventato 0, il programmatore non ha salvato in un oggetto Bundle lo stato dell’informazione numerica che viene mostrata sul display. Di conseguenza è andato perso. Avrebbe quindi dovuto … (vedi pagina precedente). Ha invece gestito bene la configurazione del layout nel momento cui è cambiato l’orientamento

**Si descriva il ciclo di vita di un’activity.**

Ogni “Activity” ha un ciclo di vita:

**Attività non esiste**

1. onCreate()

2. onStart()

3. onResume()

**Attività in esecuzione**

4. onPause()

5. onStop()

6. onDestroy()

**Attività non esiste**

Eseguiti secondo un determinato schema.

Quando l'utente inizia a lasciare l'attività, il sistema chiama i metodi per smantellare l'attività. In alcuni casi, questo smantellamento è solo parziale; l'attività risiede ancora nella memoria (come quando l'utente passa a un'altra app) e può comunque tornare in primo piano. Se l'utente ritorna a quell'attività, l'attività riprende da dove era stata interrotta. Con alcune eccezioni, le app non possono avviare attività quando vengono eseguite in background. La probabilità del sistema di uccidere un determinato processo, insieme alle attività in esso contenute, dipende dallo stato dell'attività in quel momento. A seconda della complessità della tua attività, probabilmente non è necessario implementare tutti i metodi del ciclo di vita. Tuttavia, è importante capire ognuno e implementare quelli che assicurano che la tua app si comporti come gli utenti si aspettano.

**Il ciclo di vita di un’activity prevede l’esecuzione dei seguenti metodi: onCreate(),** **onStart(), onResume(), onPause(), onStop(), onDestroy(). Indica per ognuno di questi metodi, o almeno per la maggior parte di essi, un’operazione che viene tipicamente implementata nel metodo.**

In onStart(), onResume(), onPause(), onStop(), onDestroy() viene invocato il metodo della superclasse corrispondente senza parametri.

* *onCreate():* invocazione del metodo *onCreate()* della superclasse con parametro l’oggetto Bundle e setta visibile il layout che viene scritto nell’activity\_main
* *onStart():* iniziare a disegnare elementi visivi, eseguire animazioni, ecc
* *onResume():* provare ad aprire dispositivi ad accesso esclusivo o per accedere a risorse singleton.
* *onPause():* salvare qualsiasi stato persistente dell'attività in fase di modifica, per assicurarsi che non vada perso nulla se non ci sono risorse sufficienti per avviare la nuova attività senza prima averne ucciso e per fermare le cose che consumano una notevole quantità di CPU per rendere il passaggio all'attività successiva il più velocemente possibile.
* *onStop():* per interrompere l'aggiornamento dell'interfaccia utente, l'esecuzione di animazioni e altre cose visive*.*
* *onDestroy():* implementato per liberare risorse come thread associati a un'attività, in modo che un'attività distrutta non lasci tali cose mentre il resto della sua applicazione è ancora in esecuzione.

**Si descriva il ciclo di vita delle Activity. L’intero ciclo di vita va dall’esecuzione del metodo onCreate() all’esecuzione del metodo onDestroy(). Da quale a quale metodo vanno invece il periodo di “visibilità” ed il periodo di “foreground” dell’activity?**

Visibilità: *onStart() 🡪 onResume(), onPause()* 🡪 *onStop()*

Foreground: *onStart()* 🡪 *onResume()*

**Si consideri il seguente frammento di codice usato per salvare lo stato dell’app prima di onDestroy():**

*public void onSaveInstanceState(Bundle savedInstanceState) {*

*savedInstanceState.putString(”NOME", nome);*

*savedInstanceState.putInt(”NUMERO", numero);*

*super.onSaveInstanceState(savedInstanceState);*

*}*

**Si scriva il corrispondente frammento di codice, indicando anche in quale metodo deve essere inserito, per**

**recuperare lo stato quando l’app viene ricreata.**

*Lo si recupera in onCreate():*

*@Override*

*protected void onCreate(Bundle savedInstanceState) {*

*…*

*if (savedInstanceState != null) {*

*nome = savedInstanceState.getString ("NOME");*

*numero = savedInstanceState.getInt("NUMERO");*

*}*

*}*

**A cosa servono i metodi onSaveInstanceState() e onRestoreInstanceState()?**

* ***onSaveInstanceState()*:** viene chiamato prima di posizionare l'attività in tale stato di background, consentendo di salvare qualsiasi stato di istanza dinamico nella propria attività nel pacchetto specificato, per essere successivamente ricevuto onCreate(Bundle) se l'attività deve essere ricreata.
* ***onRestoreInstanceState()***: ulteriore metodo per ripristinare lo stato dell’app, ma non è molto comune, viene chiamato dopo *onStart()*.

**Si disegni il ciclo di vita delle attività spiegando quali transizioni avvengono in modo automatico e quali invece per effetto di un intervento o dell’utente o del sistema.**

Automatico: onStart(), onResume(), onStop(), onDestroy(), onRestart()

Intervento utente/sistema: onCreate(), onPause()

**L’app TreSchermi consta di 3 activity, Activity1, Activity2 e Activity3. Ognuna di queste activity prevede 3 pulsanti: uno per passare all’attività successiva (Next), uno per passare all’attività precedente (Prev) ed una per terminare l’app (Stop). Il pulsante Next dell’Activity3 lancia l’Activity1 ed il pulsante Prev dell’Activity1 lancia l’Activity3. Si indichi la sequenza delle chiamate delle funzioni del ciclo di vita delle 3 attività nei seguenti due casi:**

**1. Avvio app (Activity1) – Pulsante Next – Pulsante Next – Pulsante Next – Pulsante Stop**

**2. Avvio app (Activity1) – Pulsante Prev – Pulsante Next – Pulsante Stop**

**CASO 1 CASO 2**

1. Activity1.onCreate()
2. Activity1.onStart()
3. Activity1.onResume()
4. Activity1.onPause()
5. Activity3.onCreate()
6. Activity3.onStart()
7. Activity3.onResume()
8. Activity3.onPause()
9. Activity1.onResume()
10. Activity1.onPause()
11. Activity1.onStop()
12. Activity1.onDestroy()
13. Activity3.onStop()
14. Activity3.onDestroy()
15. Activity1.onCreate()
16. Activity1.onStart()
17. Activity1.onResume()
18. Activity1.onPause()
19. Activity2.onCreate()
20. Activity2.onStart()
21. Activity2.onResume()
22. Activity2.onPause()
23. Activity3.onCreate()
24. Activity3.onStart()
25. Activity3.onResume()
26. Activity3.onPause()
27. Activity1.onResume()
28. Activity1.onPause()
29. Activity1.onStop()
30. Activity1.onDestroy()
31. Activity2.onStop()
32. Activity2.onDestroy()
33. Activity3.onStop()
34. Activity3.onDestroy()

**Nel contesto dell’esercizio precedente, si mostri un frammento di codice che potrebbe essere per l’onClickListener del pulsante Next dell’Activity2.**

**Si spieghi il meccanismo del backstack. In relazione a tale meccanismo che differenza c’è fra una activity e un frammento?**

**Intent**

**Cosa è un Intent? Si faccia un esempio in cui viene coinvolto un Intent utilizzano opportuni frammenti di codice.**

Un intent è un semplice oggetto messaggio che viene utilizzato per la comunicazione tra componenti Android, come attività, fornitori di contenuti, ricevitori e servizi di trasmissione. Gli intent vengono anche utilizzati per trasferire dati tra attività. Gli intent vengono generalmente utilizzati per iniziare una nuova attività utilizzando startActivity().

**Si forniscano degli spezzoni di codice per il lancio di una nuova activity con un Intent esplicito e con un Intent implicito. Si spieghi cosa è necessario per lanciare l’Intent implicito.**

***Intent esplicito:***

Un intento esplicito è quello che usi per avviare un componente specifico dell'app, come una particolare attività o servizio nella tua app. Per creare un intento esplicito, definire il nome del componente per l' Intentoggetto: tutte le altre proprietà dell'intento sono facoltative.

Ad esempio, se hai creato un servizio nella tua app, denominato DownloadService, progettato per scaricare un file dal Web, puoi avviarlo con il seguente codice:

Intent downloadIntent = new Intent(this, DownloadService.class);

downloadIntent.setData(Uri.parse(fileUrl));

startService(downloadIntent);

***Intent implicito:***

Un intento implicito specifica un'azione che può invocare qualsiasi app sul dispositivo in grado di eseguire l'azione. L'uso di un intento implicito è utile quando l'app non è in grado di eseguire l'azione, ma altre app probabilmente possono e desideri che l'utente scelga quale app utilizzare.

// Create the text message with a string

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION\_SEND);

sendIntent.putExtra(Intent.EXTRA\_TEXT, textMessage);

sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity

if (sendIntent.resolveActivity(getPackageManager()) != null) {

startActivity(sendIntent);

}

Ad esempio, se si dispone di contenuti che l'utente desidera condividere con altre persone, crea un intento con l' ACTION\_SEND azione e aggiungi extra che specificano il contenuto da condividere. Quando chiami startActivity()con quell'intento, l'utente può scegliere un'app attraverso la quale condividere il contenuto.

**Si scrivano degli snippet di codice per lanciare da un activity “Principale” un’altra activity, “Secondaria”, passando un valore di tipo intero dall’activity principale a quella secondaria e facendo in modo che l’activity secondaria restituisca un valore di tipo stringa all’activity principale.**

***Attività Principale:***

public void onClick(View v) {

Intent i = new Intent();

i.setClass(getApplicationContext(), Secondaria.class);

i.putExtra("INTERO", num);

startActivityForResult(i, 0);

}

***Attività Secondaria:***

private void setReturnIntent() {

Intent data = new Intent();

data.putExtra("STRINGA", text);

setResult(RESULT\_OK, data);

}

***Attività Principale:***

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (requestCode != 0) return;

if (resultCode != Activity.RESULT\_OK) return;

if (data == null) return;

word = data.getStringExtra("STRINGA", null);

}

**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 i = getIntent();

nome = i.getStringExtra(“NOME”, null);

valore = i.getIntExtra(“VALORE”, 0);

stato = i.getBooleanExtra(“STATO”, false);

**Il metodo startActivityForResult(Intent, int) permette di lanciare una nuova activity ed alla sua terminazione di recuperare un valore di ritorno ed ulteriori dati inseriti in un intent attraverso il metodo onActivityResult(int, int, Intent). Supponendo che l’intent non sia sufficiente né per passare i dati all’activity chiamata né per memorizzare i dati da restituire all’activity chiamante; per sfruttando i suddetti metodi, come risolveresti il problema?**

**Mostrare un frammento di codice che permette ad un’activity di un’app di lanciare un’altra activity. Mostrare un altro frammento di codice in cui si lancia un’altra activity con l’intenzione di recuperare un valore di ritorno (nota: è richiesto solo il codice per lanciare l’activity, non quello per recuperare il risultato).**

public void onClick(View v) {

Intent i = new Intent();

i.setClass(getApplicationContext(), Secondaria.class);

i.putExtra("INTERO", num);

startActivityForResult(i, 0);

}

public void onClick(View v) {

Intent i = new Intent();

i.setClass(getApplicationContext(), Secondaria.class);

i.putExtra("INTERO", num);

startActivity(i);

}

**QUESITO 5 (Permessi)**

**Si descriva il meccanismo dei permessi spiegando la differenza fra permessi normali e permessi pericolosi. Si metta in evidenza la gestione dei permessi in gruppi spiegando come vengono gestiti tali gruppi.**

Lo scopo di un permesso è proteggere la privacy di un utente Android. Le app Android devono richiedere l'autorizzazione per accedere ai dati sensibili dell'utente (come contatti), nonché a determinate funzionalità del sistema (come videocamera e Internet). A seconda della funzione, il sistema potrebbe concedere l'autorizzazione automaticamente (*Permessi normali*) o richiedere all'utente di approvare la richiesta (*Permessi pericolosi*).

* I ***permessi normali*** riguardano aree in cui l'app deve accedere a dati o risorse al di fuori della sandbox dell'app, ma in cui esiste un rischio minimo per la privacy dell'utente o il funzionamento di altre app. Ad esempio, l'autorizzazione per impostare il fuso orario è un'autorizzazione normale. Se un'app dichiara nel manifest che ha bisogno di una normale autorizzazione, il sistema concede automaticamente tale autorizzazione al momento dell'installazione. Il sistema non richiede all'utente di concedere le autorizzazioni normali e gli utenti non possono revocare tali autorizzazioni (ad esempio BLUETOOTH, FLASHLIGHT, INTERNET, NFC).
* I ***permessi pericolosi*** coprono aree in cui l'app richiede dati o risorse che coinvolgono le informazioni private dell'utente o che potrebbero potenzialmente influire sui dati memorizzati dell'utente o sul funzionamento di altre app. Ad esempio, la capacità di leggere i contatti dell'utente è un'autorizzazione pericolosa. Se un'app dichiara di aver bisogno di un'autorizzazione pericolosa, l'utente deve concedere esplicitamente l'autorizzazione all'app. Fino a quando l'utente non approva l'autorizzazione, l'app non può fornire funzionalità che dipendono da tale autorizzazione. I permessi pericolosi per le API < 23 si approvano quando si installa l’app, mentre da 23 in poi a runtime. Quando l’app richiede un permesso pericoloso, se ha già un permesso per lo stesso gruppo allora viene concesso automaticamente, altrimenti viene richiesto all’utente (dialog box) il permesso per il gruppo (CALENDAR, CAMERA, CONTACTS, LOCATION)

**Un’app utilizza 3 activity, quella principale, MainActivity, che viene lanciata alla partenza dell’app e due activity secondarie, ActivityA e ActivityB. L’activity B ha necessità di accedere al GPS mentre l‘activity B ha necessità di scattare fotografie. Mostrare le parti rilevanti del file Manifest.xml.**

*<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>*

*<uses-permission android:name="android.permission.CAMERA"/>*

**QUESITO 6 (Thread)**

**Si completi il seguente codice assumendo di avere a disposizione la funzione “partialLoad()” che si occupa di caricare in ogni chiamata un 10% dell’immagine img (quindi dopo dieci chiamate a tale funzione img sarà completa). Si renda visibile la ProgressBar all’inzio del caricamento e invisibile alla fine. Si aggiorni la progress bar ad ogni 10% di caricamento e si mostri un Toast di avviso “Caricamento quasi completato” quando si raggiunge l’80% del caricamento. Si mostri l’immagine nell’imageView alla fine del caricamento.**

public class ThreadAsyncTaskActivity extends Activity {

private ImageView imageView;

private ProgressBar progressBar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main\_layout);

imageView = (ImageView)findViewById(R.id.imageView);

progressBar = (ProgressBar)findViewById(R.id.progressBar);

}

class LoadIconTask extends AsyncTask<Integer, Integer, Bitmap> {

private Integer index = 1;

@Override

protected void onPreExecute() {

***progressBar.setVisibility(ProgressBar.VISIBLE);***

}

@Override

protected Bitmap doInBackground(Integer... ids) {

Bitmap img = ***BitmapFactory.decodeResource(getResources(), ids[0]);***

***partialLoad();***

return img;

}

@Override

protected void onProgressUpdate(Integer... values) {

***progressBar.setProgress(values[0]);***

}

@Override

protected void onPostExecute(Bitmap result) {

***progressBar.setVisibility(ProgressBar.INVISIBLE);***

***progressBar.setProgress(0);***

***imageView.setImageBitmap(result);***

}

}

}

**Descrivere i tipi utilizzati per i parametri nella classe AsyncTask<Type1,Type2,Type3> Quale è il ruolo di Type1, Type2, Type3? Fornire un esempio per rendere chiara la risposta.**

I tre tipi utilizzati da un'attività asincrona sono i seguenti:

* ***Params***, il tipo di parametri inviati all'attività al momento dell'esecuzione.
* ***Progress***, il tipo di unità di progresso pubblicate durante il calcolo in background.
* ***Result***, il tipo del risultato del calcolo in background.

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

AsyncTask consente un uso corretto e semplice del thread dell'interfaccia utente. Questa classe consente di eseguire operazioni in background e pubblicare risultati sul thread dell'interfaccia utente senza dover manipolare thread e / o gestori. AsyncTasks dovrebbe essere idealmente usato per operazioni brevi (al massimo pochi secondi).

**Android differenzia il thread principale dagli altri thread. Quali sono le operazioni non permesse nel thread principale (main) e quali quelle non permesse nei thread secondari (background)? Si motivi la risposta.**

<https://developer.android.com/guide/components/processes-and-threads>

**Il seguente abbozzo di classe NetworkTask mostra come viene utilizzato per ricevere da Internet dei dati nella forma di Stringa. Nella parte di ricezione (non dettagliata) viene eseguito un ciclo per 100 iterazioni e ad ogni iterazione viene ricevuta una stringa str mentre la stringa received\_data viene utilizzata per memorizzare tutti i dati ricevuti che alla fine verranno visualizzati nel TextView textResponse. Nel layout c’è una progressBar utilizzata per mostrare in quale iterazione ci troviamo. Questo codice non funziona. Perché? Come possiamo risolvere il problema?**

*class NetworkTask extends AsyncTask<Integer, Integer, String> {*

*@Override*

*protected void onPreExecute() {*

*progressBarDownload.setVisibility(ProgressBar.VISIBLE);*

*}*

*@Override*

*protected String doInBackground(Integer...values) {*

*//Do something*

*String received\_data = “”;*

*for (int i=0; i<100; i++) {*

*progressBarDownload.setProgress(i);*

*//Ricevi i dati nella stringa str*

*received\_data += str;*

*}*

*return received\_data;*

*}*

*@Override*

*protected void onPostExecute(String data) {*

*progressBarDownload.setVisibility(ProgressBar.INVISIBLE);*

*textResponse.setText(data);*

*}*

*}*

**Un’app che utilizza una connessione Internet sfrutta la classe AsyncTask. Tale classe prevede i seguenti metodi: onPreExecute(), doInBackground(), onProgressUpdate(), onPostExecute(). Cosa viene tipicamente implementato in questi metodi? Nella risposta specificare per ogni metodo una possibile operazione da implementare nel metodo.**

Quando viene eseguita un'attività asincrona, l'attività passa attraverso 4 passaggi:

* ***onPreExecute()***, richiamato sul thread dell'interfaccia utente prima dell'esecuzione dell'attività. Questo passaggio viene normalmente utilizzato per impostare l'attività, ad esempio mostrando una barra di avanzamento nell'interfaccia utente.
* ***doInBackground(Params...)***, richiamato sul thread in background immediatamente dopo aver onPreExecute()terminato l'esecuzione. Questo passaggio viene utilizzato per eseguire calcoli in background che possono richiedere molto tempo. I parametri dell'attività asincrona vengono passati a questo passaggio. Il risultato del calcolo deve essere restituito da questo passaggio e verrà passato all'ultimo passaggio. Questo passaggio può anche essere utilizzato publishProgress(Progress...)per pubblicare una o più unità di progresso. Questi valori sono pubblicati nel thread dell'interfaccia utente, nel onProgressUpdate(Progress...)passaggio.
* ***onProgressUpdate(Progress...)***, richiamato sul thread dell'interfaccia utente dopo una chiamata a publishProgress(Progress...). I tempi dell'esecuzione non sono definiti. Questo metodo viene utilizzato per visualizzare qualsiasi forma di progresso nell'interfaccia utente mentre il calcolo in background è ancora in esecuzione. Ad esempio, può essere utilizzato per animare una barra di avanzamento o mostrare i registri in un campo di testo.
* ***onPostExecute(Result)***, invocato sul thread dell'interfaccia utente al termine del calcolo in background. Il risultato del calcolo in background viene passato a questo passaggio come parametro.

<https://www.corsoandroid.it/usare_asynctask_per_il_download_di_una_pagina_web.html>

**Si descriva il funzionamento di AsyncTask.execute, spiegando in che modo i metodi**

1. **onPreExecute()**
2. **doInBackground(Params… params)**
3. **onProgressUpdate(Progress… values)**
4. **onPostExecute(Result result)**

**interagiscono fra loro e con l’oggetto chiamante.**

**Per un oggetto AsyncTask, a cosa serve il metodo onProgressUpdate(Integer… progress)? Viene eseguito nel thread creato per il task asincrono o nel main thread? Perché?**

**Fragment**

**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 il 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 perché favorisce il riuso del codice.

Si assuma di avere un’applicazione che, selezionato un nome di un autore, verrà visualizzata la sua citazione. Per quest’azione vengono utilizzati 2 frammenti (FragmentAutori, FragmentCitazioni). La prima operazione che deve essere effettuata è quella della visualizzazione dei nomi degli autori. Dunque nell’*onCreate()* della MainActivity istanzieremo un riferimento alla classe FragmentAutori. Per non far comunicare direttamente i 2 frammenti viene settato nella classe FragmentAutori, un metodo che setta in una variabile della classe FragmentAutori, un riferimento alla classe MainActivity, in maniera tale che, quando verrà selezionato un autore, la classe FragmentAutori, mediante un onItemClick, invocherà tramite il riferimento alla MainActivity, il metodo della classe MainActivity (respond) che gestirà la richiesta.

**Si descriva il meccanismo dei Frammenti. Si faccia un esempio per illustrarne l’utilità.**

Un frammento rappresenta una “porzione” dell’interfaccia utente. I frammenti possono essere inseriti e rimossi durante l’esecuzione. Si possono creare interfacce con molti frammenti anche in funzione della grandezza dello schermo. Un frammento deve sempre essere ospitato in un'attività e il ciclo di vita del frammento è direttamente influenzato dal ciclo di vita dell'attività dell'host. Ad esempio, quando l'attività viene messa in pausa, lo sono anche tutti i frammenti e quando l'attività viene distrutta, lo sono anche tutti i frammenti.

La porzione dell’interfaccia occupata dal frammento può essere definita nel layout oppure dinamicamente

**Per cosa sono utili i Frammenti?**

Vengono usati i frammenti per costruire interfacce utenti dinamiche. Sono utili per incapsulare la logica dell’applicazione, per migliorare la gestione del ciclo di vita e per riutilizzarli in altre attività. Ad esempio mediante i frammenti si può decidere di inserire un listView che occupa metà dello schermo e un webView che occupa l’altra metà in modo tale che quando si clicca su un elemento del frammento A, viene passata l’informazione alla webView nel frammentoB e immediatamente si vede cosa viene ricercato senza dover effettuare un cambio di attività.

**Scrivere il metodo onCreate di un’app che utilizza due frammenti e che all’avvio faccia partire i due frammenti. I due frammenti dividono lo schermo in due parti uguali. Mostrare anche i file xml per il layout. Si assuma di avere a disposizione le classi FrammentoA e FrammentoB dei due frammenti.**

**Scrivere il frammento di codice da usare nel metodo onCreate() di un’activity che permette di inserire un solo frammento (Frammento1) se la larghezza del display è minore di 300dp e di inserire due frammenti (Frammento1 e Frammento2) se la larghezza del display è uguale o maggiore di 300dp. Si assume che il layout preveda un LinearLayout orizzontale con due frame per inserire i frammenti con id, rispettivamente, frame1 e frame2.**

**State progettando un’app che utilizza dei frammenti. Avete già scritto la classe MioFrammento che implementa un singolo frammento. Adesso dovete scrivere nell’acvtivity principale il codice che inserisce 4 frammenti MioFrammento. Il file di layout prevede 4 frame per ospitare i 4 frammenti e gli id dei 4 frame sono frameFrammento1, frameFrammento2, frameFrammento3, frameFrammento4. Mostrare il codice che serve a creare ed inserire dinamicamente i frammenti. Il costruttore della classe MioFrammento prende in input una stringa che servirà per mostrare il “nome” del frammento; pertanto dovete passare una stringa “Frammento i”, dove i=1,2,3,4, per ognuno dei frammenti da creare ed inserire.**

**Networking**

**Il seguente abbozzo di classe NetworkTask mostra viene utilizzato per ricevere da Internet dei dati nella forma di Stringa. Nella parte di ricezione (non dettagliata) viene eseguito un ciclo per 100 iterazioni e ad ogni iterazione viene ricevuta una stringa str mentre la stringa received\_data viene utilizzata per memorizzare tutti i dati ricevuti che alla fine verranno visualizzati nel TextView textResponse. Nel layout c’è una progressBar utilizzata per mostrare in quale iterazione ci troviamo. Questo codice non funziona. Perché? Come possiamo risolvere il problema?**

***class NetworkTask extends AsyncTask<Integer, Integer, String> {***

***@Override***

***protected void onPreExecute() {***

***progressBarDownload.setVisibility(ProgressBar.VISIBLE);***

***}***

***@Override***

***protected String doInBackground(Integer...values) {***

***//Do something***

***String received\_data = “”;***

***for (int i=0; i<100; i++) {***

***progressBarDownload.setProgress(i);***

***//Ricevi i dati nella stringa str***

***received\_data += str;***

***}***

***return received\_data;***

***}***

***@Override***

***protected void onPostExecute(String data) {***

***progressBarDownload.setVisibility(ProgressBar.INVISIBLE);***

***textResponse.setText(data);***

***}***

***}***

**Quando un’app ha bisogno di comunicare via Internet che cosa è necessario fare? Si faccia un esempio usando uno snippet di codice (senza dettagli ma solo con la struttura generale).**

Per eseguire operazioni di rete nell'applicazione, il manifest deve includere le seguenti autorizzazioni:

*<uses-permission android:name="android.permission.INTERNET" />*

*<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />*

Per evitare di creare un'interfaccia utente che non risponde, non eseguire operazioni di rete sul thread dell'interfaccia utente. Per impostazione predefinita, Android 3.0 (livello API 11) e versioni successive richiede di eseguire operazioni di rete su un thread diverso dal thread dell'interfaccia utente principale; se non lo si fa, viene lanciata una ***NetworkOnMainThreadException***.

Il frammento *Activity* seguente utilizza un headless *Fragment* per incapsulare operazioni di rete asincrone.

**Storage**

**In che modo (o modi) varie activity che fanno parte della stessa app possono condividere dati? Si discuta dei vantaggi e svantaggi di ciascuno dei modi descritti.**

È possibile condividere dati tra varie activity che fanno parte della stessa app in tre modi:

***Shared Preferences***: permette di salvare e recuperare dati usando coppie di chiave-valore, è sufficiente fornire la chiave e recuperare il valore memorizzato.

* *Vantaggi*: sono utili per la memorizzazione delle preferenze dell'utente, non è necessario ricordare i nomi delle chiavi per utilizzare SharedPreferences (getAll()), adatto per mappare valori semplici (int, boolean, String), condividere in modo semplificato dati tra due applicazioni.
* *Svantaggi*: è conveniente per memorizzare solo piccoli bit di dati, non è necessario archiviare grandi quantità di dati in SharedPreferences, se si prova a memorizzare grosse quantità di dati causerà un utilizzo critico della memoria dell’app, velocità ed efficienza bassa per dati di grandi dimensioni.

***File***: Usati quando i dati sono voluminosi ma non strutturati, ottimi per conservare dati temporanei.

* *Vantaggi*: permette un accesso ai dati più efficienti.
* *Svantaggi*: richiede una logica più complessa per la scrittura e la lettura dei dati.

***Database SQLite***: è il linguaggio standard per database relazionali. Si utilizza quindi il concetto di tabella in cui ogni riga rappresenta un elemento e ogni colonna rappresenta un campo

* *Vantaggi*: database molto leggero, facile da usare, operazioni di lettura e scrittura molto veloci, carica solo i dati necessari anziché leggere l’intero file e tenerlo in memoria, si sovrascrivono solo le parti del file che sono state modificate, nessuna installazione richiesta per utilizzarlo, è meno soggetto a bug piuttosto che ai file, è portatile su tutti i sistemi e architetture, può essere utilizzato con tutti i linguaggi di programmazione senza problema di compatibilità.

<https://www.javatpoint.com/sqlite-advantages-and-disadvantages>

* *Svantaggi*: viene utilizzato per gestire richieste HTTP a traffico medio-basso, ma dimensione del database è limitata a 2 GB nella maggior parte dei casi.

**Si descrivano le SharedPreferences. Cosa sono? Come funzionano? Che differenza c’è fra** **SharedPreferences e Preferences?**

Per il suo utilizzo le ***SharedPreferences***, è necessario chiamare un metodo getSharedPreferences() che restituisce un'istanza ***SharedPreference*** che punta al file che contiene i valori delle preferenze.

*SharedPreferences sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE);*

Per salvare qualcosa nelle preferenze condivise usando la classe ***SharedPreferences.Editor***. Chiamerai il metodo di modifica dell’istanza ***SharedPreference*** e lo riceverai in un oggetto editor.

*Editor editor = sharedpreferences.edit();*

*editor.putString(“key”, “value”);*

*editor.commit();*

La differenza fra ***SharedPreferences*** e ***Preferences*** è :

***android.preference*** : è un pacchetto che fornisce classi per la gestione delle preferenze, come PreferenceScreen che contiene elementi di preferenza come CheckBoxPreference, EditTextPreference, ListPreference, PreferenceCategory o RingtonePreference, il che significa che la preferenza è solo gli strumenti dell'interfaccia utente.

Tutte le impostazioni effettuate per una determinata preferenza verranno automaticamente salvate nell'istanza di ***SharedPreferences*** dell'applicazione. L'accesso a SharedPreferences è semplice con getSharedPreferences () ... il che significa che questo è il modo per salvare queste preferenze.

**Si spieghi il ruolo della classe Cursor per l’utilizzo dei database SQL.**

Contiene il set di risultati di una query effettuata su un database in Android. La classe Cursor ha un'API che consente a un'app di leggere (in modo sicuro dal tipo) le colonne restituite dalla query e di scorrere le righe del set di risultati.

Una volta che un cursore è stato restituito da una query del database, un'app deve scorrere il set di risultati e leggere i dati della colonna dal cursore. Internamente, il cursore memorizza le righe di dati restituiti dalla query insieme a una posizione che punta alla riga di dati corrente nel set di risultati. Quando un cursore viene restituito da un metodo query () , la sua posizione punta allo spot prima della prima riga di dati. Ciò significa che prima di poter leggere qualsiasi riga di dati dal cursore, è necessario spostare la posizione in modo che punti a una riga di dati valida. Ci sono vari metodi per la manipolazione della posizione del cursore.

*Cursor resultSet = mydatbase.rawQuery("Select \* from TutorialsPoint",null);*

*resultSet.moveToFirst();*

*String username = resultSet.getString(0);*

*String password = resultSet.getString(1);*

**Si consideri il salvataggio di dati tramite le SharedPreferences. Si forniscano gli opportuni frammenti di codice per memorizzare una stringa (String), un intero (Int) ed un intero lungo (Long) e per recuperarli. Si indichi anche dove i frammenti di codice vanno inseriti.**

SharedPreferences sharedpreferences;

sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE);

b1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String n = ed1.getText().toString();

String ph = ed2.getText().toString();

String e = ed3.getText().toString();

SharedPreferences.Editor editor = sharedpreferences.edit();

editor.putString(Name, n);

editor.putString(Phone, ph);

editor.putString(Email, e);

editor.commit();

}

});

**Si descrivano le varie possibilità offerte da Android per la memorizzazione statica (file) dei dati.**

Per ogni app il sistema operativo prevede una directory privata, solo l’app può accedere a questa directory, all’interno della quale viene scritto un file per la memorizzazione dei dati.

Per creare e scrivere un ***file***

1. Chiamare openFileOutput(fileName, mode), restituisce un FileOutputStream

2. Scrivere nel file (write())

3. Chiudere lo stream (close())

La modalità di accesso può essere:

MODE\_PRIVATE (file accessibile solo all’app)

MODE\_APPEND

Per leggere un file

1. Chiamare openFileInput(fileName), restituisce un FileInputStream

2. Leggere dal file (read())

3. Chiudere lo stream (close())

Per i ***file temporanei*** si può usare una directory cache, Android cancellerà i file in questa directory SE necessario (quando manca spazio).

Android permette l’utilizzo di una ***memoria esterna***, tipicamente una SD card. La memoria esterna può essere rimossa, quindi non si può assumere che i file siano sempre disponibili.

**A cosa servono le classi DatabaseOpenHelper?**

Per la gestione di tutte le operazioni relative al database, è stata data una classe di supporto che si chiama SQLiteOpenHelper. Gestisce automaticamente la creazione e l'aggiornamento del database. La sua sintassi è riportata di seguito

*public class DBHelper extends SQLiteOpenHelper {*

*public DBHelper(){*

*super(context,DATABASE\_NAME,null,1);*

*}*

*public void onCreate(SQLiteDatabase db) {}*

*public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}*

*}*

**L’app Gioco utilizza due attività: L’activity Play per implementare il gioco e l’activity Punteggi per mostrare i punteggi più alti ottenuti dai giocatori. Si spieghi schematicamente, fornendo degli opportuni frammenti di codice, come le due activity possono usare le SharedPreferences per memorizzare/recuperare i punteggi più alti (si supponga di dover ricordare solo i 3 punteggi più alti).**

**Che cosa fa il seguente metodo?**

**private Cursor readSelectedEntries() {**

**String[] projection = {**

**SchemaDB.Tavola.\_ID,**

**SchemaDB.Tavola.COLUMN\_NAME,**

**SchemaDB.Tavola.COLUMN\_VOTO**

**};**

**String sortOrder = SchemaDB.Tavola.COLUMN\_VOTO + " ASC";**

**String selection;**

**selection = SchemaDB.Tavola.COLUMN\_NAME + " LIKE ? "**

**+ " and "**

**+ SchemaDB.Tavola.COLUMN\_VOTO + " >? ";**

**String[] selectionArgs = {"Car%", "25"};**

**Cursor cursor = db.query(**

**SchemaDB.Tavola.TABLE\_NAME,**

**projection,**

**selection,**

**selectionArgs,**

**null,**

**null,**

**sortOrder**

**);**

**return cursor;**

**}**

**ANIMAZIONI**

**Si fornisca un file XML per un’animazione che prima ruota la view di 180°e poi la sposta orizzontalmente di 200dp.**

*<set xmlns:android="http://schemas.android.com/apk/res/android"*

*android:shareInterpolator="false" >*

*<rotate*

*android:startOffset="0"*

*android:duration="4500"*

*android:fromDegrees="0"*

*android:toDegrees="180"*

*android:pivotX="50%"*

*android:pivotY="50%" />*

*<translate*

*android:startOffset="4500"*

*android:duration="2000"*

*android:fromXDelta="0"*

*android:fromYDelta="0"*

*android:toXDelta="200"*

*android:toYDelta="0" />*

*</set>*

**Si scriva un file XML con un’animazione che permette di traslare prima un oggetto prima in orizzontale per 500 pixels in 1 secondo e poi in verticale per 300 pixel in 2 secondi. Normalmente dopo l’animazione l’oggetto ritorna automaticamente nella posizione iniziale; cosa si può fare per farle rimanere nella posizione finale dell’animazione?**

*<set xmlns:android="http://schemas.android.com/apk/res/android"*

*android:shareInterpolator="false"*

*<translate*

*android:startOffset="0"*

*android:duration="1000"*

*android:fromXDelta="0"*

*android:fromYDelta="0"*

*android:toXDelta="500"*

*android:toYDelta="0" />*

*<translate*

*android:startOffset="1000"*

*android:duration="2000"*

*android:fromXDelta="0"*

*android:fromYDelta="0"*

*android:toXDelta="0"*

*android:toYDelta="300" />*

*</set>*

Per far si che un oggetto traslato non ritorni nella sua poizione iniziale dopo l’animazione, è possibile procedere come segue:

1. Aggiungere un AnimationListener all’animazione che effettua la traslazione:

animation.setAnimationListener(new Animation.AnimationListener(){

@Override

public void onAnimationStart(Animation animation){ }

@Override

public void onAnimationEnd(Animation animation){ }

@Override

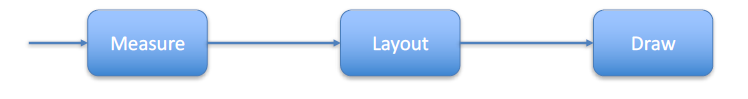
public void onAnimationRepeat(Animation animation){ }

});

1. Nel metodo onAnimationEnd(), chiamare il metodo public void layout(int left, int top, int right, int bottom) sulla view traslata, ed aggiornare i parametri opportuni. Un altro modo è settare i parametri di layout ricavando i LayoutParams, settandoli ed aggiungerli alla View.

**Come funziona il meccanismo di layout e la relativa fase di misurazione delle view?**

Il meccanismo di layout si compone di 3 fasi:



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

Un’interfaccia grafica in Android è costituita da un insieme di View disposte sullo schermo tramite un layout. In Android i bottoni, le caselle di testo, le immagini sono delle View (android.view.View) raggruppate in delle ViewGroup. Le ViewGroup possono essere viste come dei contenitori di View che a loro volta possono essere trattati come delle View e possono essere inseriti in altri ViewGroup. L’interfaccia grafica in Android, costituita da View e ViewGroup può essere vista come un albero costituito dalle View come nodi foglia e le ViewGroup come nodi interni.

**Si spieghi come avviene la misurazione e il posizionamento delle view di un layout. Perché in alcuni casi i metodi v.getWidth e v.getHeight, dove v è una view del layout, usati in onCreate() restituiscono 0?**

* ***Misurazione***: quando si effettua la misurazione di deve distinguere la “measured” size (quanto grande la view vorrebbe essere) e la size reale (quanto grande la view sarà in realtà). Si utilizza un approccio top-down nell’albero delle view: ogni view chiede ai suoi figli quanto vorrebbero essere grandi. Ogni view può esprimere la propria preferenza usando le opzioni della classe ViewGroup.LayoutParams (MATCH\_PARENT, WRAP\_CONTENT o numero di pixel), la misurazione avviene nel metodo onMeasure(). Quando il processo di misurazione finisce ogni view deve aver definito: measuredWidth e measuredHieght, In alcuni casi c’è un processo di `negoziazione’ fra view parent e view figli, se measure() è chiamato più volte.
* ***Posizionamento***: la view parent decide la grandezza e la posizione (in accordo alle misure fatte nella fase precedente), attraverso il metodo onLayout().

In alcuni casi, i metodi getWidth e getHeight usati in onCreate(), non devono essere chiamati troppo presto, perché l'interfaccia utente non è stata ancora dimensionata e impostata sullo schermo.

**Si spieghi il meccanismo di misurazione e posizionamento che avviene sull’albero delle view quando ci sono delle modifiche al layout.**

**Come si può animare un oggetto grafico in Android? Si faccia un esempio scegliendo una specifica animazione.**

Android permette di definire delle animazioni da applicare alle immagini, descritte con file XML (ad esempio rotazione, trasparenza etc…). Una volta definiti i file xml è necessario legge le animazioni fai file XML e applicare alle ImageView mediante un oggetto Animation. Vedi esempio sotto.

**Come funzionano le animazioni? Si faccia un esempio.**

L'animazione è il processo di creazione del cambiamento di movimento e forma. Si assuma un’app in cui è presente un’immagine al centro dello schermo, definita nel layout principale con il Widget “ImageView”. Si vuole fare in modo che quando venga premuto il bottone “rotazione”, l’immagini ruoti di 180 gradi. All’interno della directory res, si crea una directory “anim” all’interno della quale creeremo un file xml che gestirà la rotazione.

*<rotate*

*android:startOffset="0"*

*android:duration="4000"*

*android:fromDegrees="0"*

*android:toDegrees="180"*

*android:pivotX="50%"*

*android:pivotY="50%"*

*/>*

Nella nostra MainActivity verrà istanziato un oggetto Animation che verrà eguagliato nel seguente modo:

*AnimationUtils.loadAnimation(getApplicationContext(), R.anim.rotazione);*

Verrà definito un metodo invocato al click del bottone che starterà l’oggetto con la relativa animazione.

**La seguente classe implementa una view customizzata per un pentagramma. Così come mostrato l’oggetto Pentagramma viene disegnato con una nota in una specifica posizione (alle coordinate 350,78). Descrive le modifiche da fare per fare in modo che inizialmente il pentagramma sia vuota e che successivamente si possano aggiungere delle note al pentagramma.**

*public class Pentagramma extends View {*

*int MARGIN\_TOP= 50;*

*int MARGIN\_H= 40;*

*int LINE\_SPACING = 8;*

*int NOTEHEAD\_SIZE = 6;*

*int STEM\_LENGTH = 30;*

*public class Nota { … }*

*ArrayList<Nota> notesList = new ArrayList<Nota>();*

*public Pentagramma(Context c, int screen\_w, int screen\_h) {*

*super(c);*

*setMinimumWidth(screen\_w);*

*setMinimumHeight(150);*

*Nota nota = new Nota(350, 78);*

*notesList.add(nota);*

*}*

*@Override*

*protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) { ... }*

*@Override*

*protected void onLayout(boolean b, int x1, int y1, int x2, int y2) { … }*

*@Override*

*protected void onDraw(Canvas canvas) {*

*Paint paint = new Paint();*

*paint.setColor(Color.BLACK);*

*int h = MARGIN\_TOP;*

*int x\_start = MARGIN\_H;*

*int x\_end = canvas.getWidth() - MARGIN\_H;*

*for (int i = 0; i<5; i++ ) {*

*canvas.drawLine(x\_start,h,x\_end,h,paint);*

*h = h + LINE\_SPACING;*

*}*

*Nota n = notesList.get(0);*

*int note\_position\_horizzontal = n.getH();*

*int note\_position\_vertical = n.getV();*

*canvas.drawCircle(note\_position\_horizzontal, note\_position\_vertical, NOTEHEAD\_SIZE, paint);*

*canvas.drawLine(note\_position\_horizzontal+5, note\_position\_vertical-STEM\_LENGTH,*

*note\_position\_horizzontal+5, note\_position\_vertical, paint);*

*}*

**Che cosa è e a cosa serve l’oggetto Canvas? Si illustri, con sufficienti dettagli, un esempio in cui è necessario utilizzare un oggetto Canvas.**

Il Canvas svolge un ruolo vitale nella creazione di visualizzazioni personalizzate. Un Canvas è un framework di disegno 2D che ci fornisce metodi per disegnare sulla Bitmap, che funge da superficie su cui è posizionata il Canvas. La classe Paint viene utilizzata per fornire colori e stili.

**Si spieghi come si creano le animazioni grafiche (magari descrivendo tutte le possibilità) in Android possibilmente facendo un esempio concreto (magari usando solo una o due delle varie possibilità descritte).**

Android permette di definire delle animazioni da applicare alle immagini, descritte con file XML:

* Rotazione
* Traslazione
* Scaling
* Trasparenza

Con controlli di vario tipo usufruendo dei parametri, come pivot, velocità, etc….

**Si scriva un file xml per la seguente animazione di un oggetto drawable:**

1. **rotazione di 2 giri completi a destra, dal tempo 0 al tempo 2 sec**
2. **traslazione di 300px a destra, dal tempo 3 sec al tempo 4 sec,**
3. **rotazione di 2 giri completi a sinistra, dal tempo 3 sec al tempo 5 sec.**

*<rotate*

*android:startOffset="0"*

*android:duration="2000"*

*android:fromDegrees="0"*

*android:toDegrees="720"*

*/>*

*<translate*

*android:startOffset="3000"*

*android:duration="1000"*

*android:fromXDelta="0"*

*android:fromYDelta="0"*

*android:toXDelta="300"*

*android:toYDelta="0"*

*/>*

*<rotate*

*android:startOffset="4000"*

*android:duration="2000"*

*android:fromDegrees="0"*

*android:toDegrees="-720"*

*/>*

**Si scriva un file animation.xml che permette di implementare la seguente animazione di una view: spostamento orizzontale di 300 px verso destra, seguito da una rotazione di 2 giri completi in senso orario con pivot al centro dell’oggetto, a sua volta seguita da un nuovo spostamento in orizzontale di 300px verso sinistra che riporta l’oggetto nella posizione originaria.**

*<translate*

*android:startOffset="0"*

*android:duration="2000"*

*android:fromXDelta="0"*

*android:fromYDelta="0"*

*android:toXDelta="300"*

*android:toYDelta="0"*

*/>*

*<rotate*

*android:startOffset="2000"*

*android:duration="5000"*

*android:fromDegrees="0"*

*android:toDegrees="720"*

*android:pivotX= "80%"*

*android:pivotY= "50%"*

*/>*

*<translate*

*android:startOffset="7000"*

*android:duration="2000"*

*android:fromXDelta="0"*

*android:fromYDelta="0"*

*android:toXDelta="-300"*

*android:toYDelta="0"*

*/>*

**Il seguente snippet di codice esegue prima un’animazione e poi rimuove l’oggetto dalla view parent (gli oggetti image, animation e parentView sono stati in precedenza opportunamente inizializzati):**

***…***

***image.startAnimation(animation);***

***parentView.removeViewAt(0);***

***….***

**Tuttavia l’effetto è quello di rimuovere immediatamente l’immagine senza dare il tempo all’animazione di essere eseguita. Perché accade ciò? Come si può ovviare al problema?**

**Si abbozzi (nel senso che si possono tralasciare i dettagli implementativi scrivendo cose del tipo “disegna una linea orizzontale ad un terzo dell’altezza”, oppure “disegna una X nella casella i,j”) la classe TrisView che estende una View e che fornisce dei metodi, addX(i,j) e addO(i,j), per visualizzare una “X” o una “O” in ognuno dei 9 riquadri. Un esempio di utilizzo della classe è riportato di seguito:**

**public class MainActivity extends Activity {**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

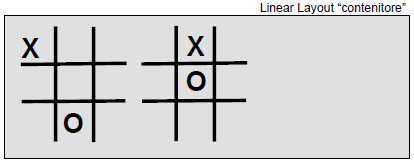
**super.onCreate(savedInstanceState);**

**setContentView(R.layout.activity\_main);**

**LinearLayout contenitore = (LinearLayout) findViewById(R.id.contenitore);**

**TrisView tris1 = new TrisView(300,300);**

**TrisView tris2 = new TrisView(300,300);**

**contenitore.addView(tris1);**

**contenitore.addView(tris2);**

**tris1.addX(1,1);**

**tris1.addO(3,2);**

**tris1.invalidate();**

**tris2.addX(1,2);**

**tris2.addO(2,2);**

**tris2.invalidate();**

**}**

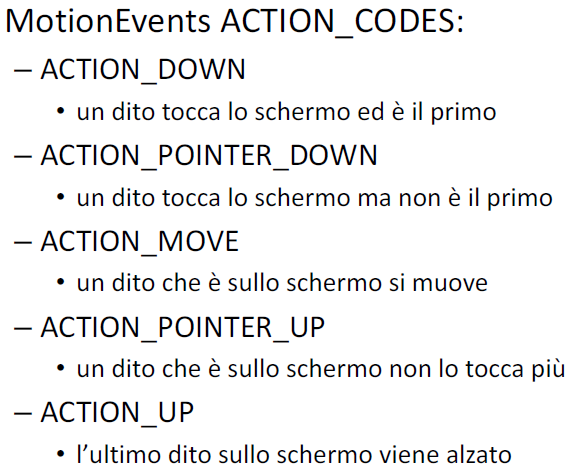
**}**

**public class TrisView extends View {**

**MULTITOUCH**

**Il seguente frammento di codice mostra un OnTouchListener per un MotionEvent. Si completi il codice facendo in modo che la variabile counter (si assuma che tale variabile sia accessibile globalmente) contenga sempre il numero di dita che stanno toccando lo schermo.**

***int counter;***

***public boolean onTouch(View v, MotionEvent event) {***

***switch(event.getActionMasked()) {***

***case MotionEvent.ACTION\_DOWN:***

counter++;

***break;***

***case MotionEvent.ACTION\_POINTER\_DOWN:***

counter++;

***break;***

***case MotionEvent.ACTION\_MOVE:***

---------------

***break;***

***case MotionEvent.ACTION\_POINTER\_UP:***

counter--;

***break;***

***case MotionEvent.ACTION\_UP:***

counter--;

***}***

***return true;***

***}***

**Si descriva il meccanismo del Multitouch. Come vengono rappresentati i movimenti? Che cosa è un “pointer”? Che cosa è un MotionEvent?**

Il Multitouch rappresenta un movimento registrato da una periferica, tocchi multipli sullo schermo. Quando più ***pointer*** toccano lo schermo contemporaneamente, il sistema genera i seguenti eventi di tocco:

* ***ACTION\_DOWN***: un dito tocca lo schermo ed è il primo;
* ***ACTION\_POINTER\_DOWN***: un dito tocca lo schermo ma non è il primo;
* ***ACTION\_MOVE***: un dito che è sullo schermo si muove;
* ***ACTION\_POINTER\_UP***: un dito che è sullo schermo non tocca più;
* ***ACTION\_UP***: ultimo dito sullo schermo viene alzato.

Un ***Pointer*** è il singolo evento che si verifica, esempio un dito che tocca lo schermo. Le singole dita o altri oggetti che generano tracce di movimento sono indicati come pointer.

Ogni ***Pointer*** ha anche una mappatura ***ID*** che rimane persistente negli eventi touch per consentire il tracciamento di un singolo puntatore nell'intero gesto.

Un ***Motion Event*** è un oggetto utilizzato per segnalare eventi di movimento (mouse, penna, dito), rappresenta un singolo pointer e a volte più pointer.

Ogni pointer ha un ID unico per tutto il tempo in cui esiste, l’indice di un MotionEvent multiplo NON è il pointer ID.

Il pointer ID è costante, l’indice può cambiare per eventi successivi.

**Che cosa è un oggetto MotionEvent? Si forniscano dettagli sul suo utilizzo.**

Un ***MotionEvent*** è un oggetto utilizzato per segnalare eventi di movimento (mouse, penna, dito), rappresenta un singolo pointer e a volte più pointer.

Ad esempio, quando l'utente tocca per la prima volta lo schermo, il sistema invia un evento di tocco all'appropriato ***View*** con il codice di azione ***ACTION\_DOWN*** e un insieme di valori degli assi che includono le ***coordinate*** ***X*** e ***Y*** del tocco e informazioni sulla pressione, le dimensioni e l'orientamento di l'area di contatto.

La classe ***MotionEvent*** fornisce molti metodi per interrogare la posizione e altre proprietà di puntatori, come ***getX(int), getY(int), getAxisValue(int), getPointerId(int), getToolType(int),*** e molti altri. La maggior parte di questi metodi accetta l'indice del puntatore come parametro anziché l'id del puntatore. L'indice del puntatore di ciascun puntatore nell'evento varia da 0 a uno in meno del valore restituito da ***getPointerCount().***

Per ottenere un azione di un ***MotionEvent*** si dovrebbe sempre usare il metodo ***getActionMasked()***, è progettato per funzionare con più puntatori. Restituisce l'azione mascherata che viene eseguita, senza includere i bit dell'indice del puntatore.

**L’oggetto MotionEvent viene usato per descrivere i tocchi dell’utente sullo schermo. I tocchi ed i movimenti vengono codificati attraverso dei codici (riportati sotto) e dei “pointer ID”.**

**MotionEvent.ACTION\_DOWN**

**MotionEvent.ACTION\_POINTER\_DOWN**

**MotionEvent.ACTION\_MOVE**

**MotionEvent.ACTION\_POINTER\_UP**

**MotionEvent.ACTION\_UP**

**Quale sequenza di questi codici-pointer ID viene generata nei seguenti casi?**

1. **Pollice tocca lo schermo, indice tocca lo schermo, pollice e indice si muovono, pollice lascia lo schermo, indice lascia lo schermo**

|  |  |  |
| --- | --- | --- |
| Pollice tocca lo schermo | MotionEvent.ACTION\_DOWN | 0 |
| Indice tocca lo schermo | MotionEvent.ACTION\_POINTER\_DOWN | 1 |
| Pollice e indice si muovono | MotionEvent.ACTION\_MOVE | 0,1 |
| Pollice lascia lo schermo | MotionEvent.ACTION\_POINTER\_UP | 0 |
| Indice lascia lo schermo | MotionEvent.ACTION\_UP | 1 |

1. **Pollice tocco lo schermo, medio tocca lo schermo, mignolo tocca lo schermo, medio lascia lo schermo, pollice lascia lo schermo, mignolo lascia lo schermo**

|  |  |  |
| --- | --- | --- |
| Pollice tocca lo schermo | MotionEvent.ACTION\_DOWN | 0 |
| Medio tocca lo schermo | MotionEvent.ACTION\_POINTER\_DOWN | 1 |
| Mignolo tocca lo schermo | MotionEvent.ACTION\_POINTER\_DOWN | 2 |
| Medio lascia lo schermo | MotionEvent.ACTION\_POINTER\_UP | 1 |
| Pollice lascia lo schermo | MotionEvent.ACTION\_POINTER\_UP | 0 |
| Mignolo lascia lo schermo | MotionEvent.ACTION\_UP | 2 |

**State sviluppando un’app che permette di sfruttare il tocco multiplo sullo schermo per visualizzare un cerchio in ogni punto toccato. Avete a disposizione la classe CircleTouch che permette di creare oggetti grafici da visualizzare sullo schermo. La classe ha i seguenti metodi**

* + **costruttore CircleTouch(Context c, int id): crea un nuovo CircleTouch, con identificatore id**
  + **metodo sposta(int x, int y): aggiorna le coordinate dell’oggetto**
  + **metodo onDraw(Canvas c): disegna il cerchio.**

**Assumendo che l’oggetto main\_window sia il layout esterno al quale aggiungere e rimuovere gli oggetti CircleTouch, indicare le principali istruzioni da inserire nel seguente frammento di codice:**

***switch (event.getActionMasked()) {***

***case MotionEvent.ACTION\_DOWN:***

---------------------------------------------------------------

***case MotionEvent.ACTION\_POINTER\_DOWN:***

pointer\_index = event.getActionIndex();

pointer\_id = event.getPointerId(pointer\_index);

if (hashMap.containsKey(pointer\_id)) {

ct = hashMap.get(pointer\_id);

} else {

ct = new *CircleTouch*(getApplicationContext(), pointer\_id);

hashMap.put(pointer\_id, ct);

*main\_window*.addView(ct);

}

ct.***sposta***(event.getX(pointer\_index),event.getY(pointer\_index));

ct.setSize(50);

ct.invalidate();

break;

***case MotionEvent.ACTION\_UP:***

---------------------------------------------------------------

***case MotionEvent.ACTION\_POINTER\_UP:***

pointer\_index = event.getActionIndex();

pointer\_id = event.getPointerId(pointer\_index);

if (hashMap.containsKey(pointer\_id)) {

ct = hashMap.get(pointer\_id);

ct.setSize(0);

ct.invalidate();

*main\_window*.removeView(ct);

hashMap.remove(pointer\_id);

}

break;

***case MotionEvent.ACTION\_MOVE:***

for (int i = 0; i < event.getPointerCount(); i++) {

pointer\_id = event.getPointerId(i);

ct = hashMap.get(pointer\_id);

if (null != ct) {

ct.*sposta*(event.getX(i),event.getY(i));

ct.invalidate();

}

}

break;

***}***

**SENSORI**

**Quando si registra il listener di un sensore è possibile selezionare la velocità di campionamento da utilizzare:**

**– SENSOR\_DELAY\_NORMAL (0,2sec)**

**– SENSOR\_DELAY\_GAME (0,02sec)**

**– SENSOR\_DELAY\_UI (0,06sec)**

**– SENSOR\_FASTEST (0sec )**

**Si discuta dei vantaggi e svantaggi di queste varie possibilità e di quale accortezze deve avere il programmatore per un app che utilizza i sensori.**

**Si consideri l’utilizzo di un sensore, Android permette di specifica re la velocità di campionamento quando si registra il listener del sensore:**

**boolean registerListener (SensorEventListener listener, Sensor sensor, int samplingPeriodUs).**

**Android fornisce anche dei valori predeterminati da usare per il parametro samplingPeriodUs:**

**SensorManager.SENSOR\_DELAY\_NORMAL (0,2sec)**

**SensorManager.SENSOR\_DELAY\_GAME (0,02sec)**

**SensorManager.SENSOR\_DELAY\_UI (0,06sec)**

**SensorManager.SENSOR\_FASTEST (0sec )**

**Si discuta del ruolo di questo parametro spiegando perché si dovrebbe scegliere una frequenza di campionamento più lenta o più veloce e delle conseguenza della scelta.**

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

I sensori consentono di misurare il movimento l'orientamento e varie condizioni ambientali. Questi sensori sono in grado di fornire dati grezzi con elevata precisione e accuratezza e sono utili se si desidera monitorare il movimento o il posizionamento tridimensionale del dispositivo o se si desidera monitorare i cambiamenti nell'ambiente circostante un dispositivo. La piattaforma Android supporta tre grandi categorie di sensori:

* ***Sensori di movimento***, questi sensori misurano le forze di accelerazione e le forze di rotazione lungo tre assi. Questa categoria comprende accelerometri, sensori di gravità, giroscopi e sensori vettoriali rotazionali.
* ***Sensori ambientali***, questi sensori misurano vari parametri ambientali, come temperatura e pressione dell'aria ambiente, illuminazione e umidità. Questa categoria comprende barometri, fotometri e termometri.
* ***Sensori di posizione***, questi sensori misurano la posizione fisica di un dispositivo. Questa categoria comprende sensori di orientamento e magnetometri.

Per identificare i sensori che si trovano su un dispositivo, è innanzitutto necessario ottenere un riferimento al servizio dei sensori. L’attività deve implementare SensorEvenListener. Successivamente si crea un'istanza della classe SensorManager chiamando il metodo getSystemService() e passando l'argomento SENSOR\_SERVICE. Poi si deve controllare se il sensore esiste, per esempio:

*private SensorManager mSensorManager;*

*mSensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);*

*Sensor light= mSensorManager.getDefaultSensor(Sensor.TYPE\_LIGHT);*

*if (light != null){*

*// Success! There's an ambient light sensor.*

*}else {*

*// Failure! No light sensor*

*}*

Una volta dichiarato quel sensore, è necessario registrare il suo listener e sovrascrivere due metodi che sono onAccuracyChanged e onSensorChanged. La sua sintassi è la seguente:

*sMgr.registerListener(this, light,SensorManager.SENSOR\_DELAY\_NORMAL);*

*public void onAccuracyChanged(Sensor sensor, int accuracy) { }*

*public void onSensorChanged(SensorEvent event) { }*

**NOTIFICHE**

**Quali sono i principali modi per fornire notifiche all’utente? Si fornisca una breve descrizione.**

* ***Toast***: Un Toast è una piccola notifica popup che viene utilizzata per visualizzare informazioni sull'operazione che abbiamo eseguito nella nostra app. Toast mostrerà il messaggio per un breve periodo di tempo e scomparirà automaticamente dopo un timeout. Generalmente, la notifica Toast in Android verrà visualizzata con un testo semplice.
* ***Dialog***: una piccola finestra che richiede all'utente di prendere una decisione o inserire informazioni aggiuntive. Una finestra di dialogo non riempie lo schermo e viene normalmente utilizzata per eventi modali che richiedono agli utenti di eseguire un'azione prima di poter procedere.
* ***Notification*** ***Area*** (***Status*** ***Bar***): è un messaggio che Android visualizza all'esterno dell'interfaccia utente dell'app per fornire all'utente promemoria, comunicazioni di altre persone o altre informazioni tempestive dall'app. Gli utenti possono toccare la notifica per aprire l'app o eseguire un'azione direttamente dalla notifica. Le notifiche possono apparire brevemente in una finestra mobile denominata notifica heads-up . Questo comportamento è in genere per le notifiche importanti che l'utente dovrebbe conoscere immediatamente e appare solo se il dispositivo è sbloccato.

**Che cosa è un Toast customizzato? Si spieghi come implementare un Toast customizzato.**

In Android, possiamo personalizzare il layout della nostra notifica toast per cambiare l'aspetto in base a requisiti come includere immagini nella notifica toast o cambiare il colore di sfondo della notifica toast, ecc.

Per implementare un Toast customizzato, si crea un file xml contente la struttura che dovrà avere il Toast. Assumiamo che nella nostra app sia presente un bottone, alla cui pressione dovrà essere visualizzato il Toast mediante un metodo. All’interno del metodo si crea un riferimento ad un oggetto Toast, vengono settare le varie proprietà che dovrà possedere l’oggetto (gravity, duration, etc…), viene associato all’oggetto Toast il file xml creato in precedenza. Infine viene mostrato all’utente.

*public void showCustomToast(View v) {*

*Toast toast = new Toast(getApplicationContext());*

*toast.setGravity(Gravity.CENTER\_VERTICAL, 0, 0);*

*toast.setDuration(Toast.LENGTH\_LONG);*

*toast.setView(getLayoutInflater().inflate(R.layout.custom\_toast,null));*

*toast.show();*

*}*

**Le notifiche in un app android sono brevi messaggi per l’utente visualizzati in vari modi. Si elenchino le tipologie di notifiche conosciute, possibilmente con dei frammenti di codice che mostrano il loro utilizzo.**

* ***Toast***: *Toast.makeText(getApplicationContext(), "Toast!", Toast.LENGTH\_LONG).show();*
* ***Dialog***:

*DialogInterface.OnClickListener dialogClickListener = new DialogInterface.OnClickListener() {*

*public void onClick(DialogInterface dialog, int which) {*

*switch (which) {*

*case DialogInterface.BUTTON\_POSITIVE:*

*// to do*

*case DialogInterface.BUTTON\_NEGATIVE:*

*//to do*

*}*

*}*

*};*

*AlertDialog.Builder builder = new AlertDialog.Builder(this);*

*builder.setMessage("Stai per ripartire da capo. Sei sicuro?")*

*.setPositiveButton("Si", dialogClickListener)*

*.setNegativeButton("No", dialogClickListener).show();*

* ***Notifica***:

*Notification.Builder notificationBuilder = new Notification.Builder( getApplicationContext()).setTicker(“Esempio”)*

*NotificationManager mNotificationManager = (NotificationManager)getSystemService(Context.NOTIFICATION\_SERVICE);*

*mNotificationManager.notify(1,notificationBuilder.build());*

**Content Providers, Broadcast e Services**

**Le activity sono una delle 4 componenti principale di un’app. Quali sono le altre 3? Si fornisca una breve descrizione.**

Le 4 componenti fondamentali di Android sono:

* ***Activity***: //
* ***Broadcasts***: possono essere utilizzate come sistema di messaggistica tra le app e al di fuori del normale flusso di utenti. Tuttavia, è necessario fare attenzione a non abusare dell'opportunità di rispondere alle trasmissioni ed eseguire lavori in background che possono contribuire a rallentare le prestazioni del sistema. Ad esempio, il sistema Android invia trasmissioni quando si verificano vari eventi di sistema, ad esempio quando il sistema si avvia o il dispositivo inizia a caricarsi.
* ***Content*** ***Providers***: fornisce i dati da un'applicazione ad altre su richiesta. Tali richieste sono gestite dai metodi della classe ContentResolver. Un Content Providers può utilizzare diversi modi per archiviare i propri dati ad esempio in un database, in file o persino su una rete.
* ***Services***: è un componente dell'applicazione che può eseguire operazioni di lunga durata in background e non fornisce un'interfaccia utente. Un altro componente dell'applicazione può avviare un servizio e continua a essere eseguito in background anche se l'utente passa a un'altra applicazione. Ad esempio, un servizio può gestire transazioni di rete, riprodurre musica, eseguire operazioni di I / O su file o interagire con un fornitore di contenuti, tutto da sfondo.

**Extra**

**Android permette al programmatore di specificare le dimensioni usando varie unità di misura:**

* **dp, (density-independent pixels)**
* **sp, (scale-independent pixels**
* **pt, points (1/72 di pollice)**
* **px, pixel reali**
* **mm, millimetri**
* **in, pollici (inches)**

**Perché? Si discuta delle differenze fra queste varie possibilità.**

**Cosa è e a cosa serve l’Android Virtual Device Manager?**

**Nel posizionamento di elementi di un’interfaccia grafica che differenza c’è**

**1. fra l’attributo padding e l’attributo margin?**

**2. fra l’attributo gravity e layout\_gravity?**

**Android prevede un meccanismo per il supporto di lingue diverse che rende molto facile cambiare la lingua utilizzata per i messaggi sullo schermo. Come funziona tale meccanismo?**