**LAYOUT**

**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 cartella è la root, denominata “app”, all’interno sono presenti le seguenti sottodirectory (l’ordine delle directory è importante):

* ***manifests***: contiene il file ***AndroidManifest.xml*** che racchiude le informazioni essenziali dell’app;
* ***java***: contiene il codice sorgente dell’applicazione, organizzato in vari packages, utilizzato per costruire l’app;
* ***res***: contiene le risorse dell’app, ci sono al suo interno altre directory, come “*layout*” che fornisce la struttura statica dell’app (in 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.

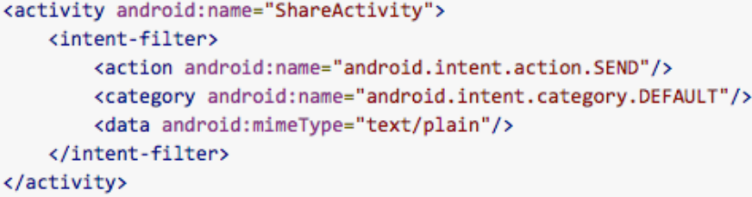
**A cosa serve il file Manifest.xml?**

Tutte le app Android hanno un file ***AndroidManifest.xml*** nella root del sorgente, è un file di risorse che contiene tutti i dettagli necessari dell’applicazione, ci consente di definire:

* ***Packages***;
* ***API***;
* ***Librerie***;
* ***Elementi*** ***base*** dell’app, come activity e servizi;
* Dettagli sulle ***autorizzazioni***;
* ***Set di classi necessarie*** prima del lancio.

Ci sono ulteriori informazioni che vengono riportate come:

* ***Nome dell’applicazione***: titolo dell’app (*android:label*= “*nomeApp*”);
* ***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*”).

**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.**

L’elemento ***<intent-filter>,*** figlio dell’elemento ***<activity>*** in un file manifesto di un’app, viene utilizzato per specificare i tre tipi di informazioni che quell’activity utilizza: ***action***, ***data*** e ***category***; questi tre parametri vengono utilizzati per la risoluzione implicita di un intent.

Per il caso specificato, abbiamo un’activity ***ShareActivity*** che può gestire l’invio (*action: android.intent.action.SEND*) di testo piano (*data: text/plain*); notiamo che tale activity ha la category di ***DEFAULT***: questo indica che, quando viene effettuata la risoluzione implicita di un intent che indica come action e data rispettivamente *SEND* e text/plain, allora l’activity ***ShareActivity*** è quella di default per effettuare tale computazione.

**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?**

All’interno della directory ***res/*** ci sono sottodirectory per vari tipi di risorse, ad esempio file predefiniti come ***res/values/strings.xml***, ma è possibile creare altre directory che aderiscono ad un determinato formato, come impostazioni internazionali dell'interfaccia utente. Quest’ultime conterranno file ***strings.xml*** i quali definiscono testi in lingue diverse. L’utente una volta che sceglie la lingua, verrà caricato il file ***strings.xml*** relativa alla scelta.

**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 dell'interfaccia utente, ad esempio di 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 in modo netto la definizione dell’UI dal codice dell’app (facile modifiche future).

***SVANTAGGI***: elementi statici.

* ***Creare 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.

Un esempio in cui è possibile utilizzare soltanto uno dei due approcci riguarda, generalmente, casi in cui l’applicazione non prevede un’interazione con l’utente al runtime: più nello specifico, un’applicazione che deve mostrare semplicemente del testo all’utente (senza che egli debba interagire con l’interfaccia) può essere creata utilizzando esclusivamente l’approccio statico, attraverso i file XML.

**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***, ***EditText***, ***ImageView***, ***CheckBox***, ***RadioButton***. I principali layout per il posizionamento dei widget sono:

* ***LinearLayout***: organizza gli elementi in un’unica riga orizzontale o verticale ed è possibile crea una barra di scorrimento se la lunghezza della finestra supera la lunghezza dello schermo.
* ***RelativeLayout***: la posizione degli elementi è relativa al layout padre e agli altri elementi del layout, esempio figlio A alla sinistra del figlio B o allineato alla parte superiore del genitore.
* ***AbsoluteLayout***: consente di specificare posizioni esatte, esplicitando coordinate x e y, dei suoi figli. Sono meno flessibili.
* ***FrameLayout***: progettato per bloccare un’area sullo schermo per visualizzare degli elementi.

Si possono creare Layout tramite un Adapter:

* ***ListView***: visualizza un elenco scorrevole verticale.
* ***GridView***: visualizza una griglia scorrevole bidimensionale.

**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, ad esempio:

* “***android:id***”: permette di assegnare un identificativo univoco agli elementi della UI. In questo modo, in una fase programmatica, si potrà ottenere il riferimento a quell’elemento, per poter modificare in modo dinamico lo stato dell’oggetto.
* *“****android:layout\_height”/“android:layout\_weight****”:* specificano altezza/larghezza di base della View, attributi obbligatori per qualsiasi View all’interno del layout. Ci sono varie unità di misurazione disponibili (px, dp, etc…) per entrambi.
* *“****android:background****”*: consente di definire il colore di sfondo di un widget (utile per verificare se è stato inserito nel layout).
* “***android:margin***”: utilizzato per creare spazio attorno agli elementi, al di fuori di qualsiasi bordo definito.
* “***android:padding***”: utilizzato per generare spazio attorno al contenuto di un elemento;
* “***android:gravity***”: specifica come un oggetto deve posizionare il suo contenuto dentro un componente;
* “***android:layout\_gravity***”: specifica come un componente dovrebbe essere posizionato nel suo ViewGroup.

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

Un ***Android Virtual Device*** è una configurazione di un dispositivo Android che si desidera simulare. ***Android Virtual Device Manager*** è un’interfaccia che si può avviare in Android Studio, il quale permette di creare e gestire un ***AVD***.

**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.

I vantaggio sono:

* Facilità di creazione di determinate situazioni, quali batteria scarica o arrivo di un messaggio
* Simulare un qualsiasi stato di rete
* Avviare una chiamata controllata

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.

**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.**

Le varie unità di misura permettono al programmatore di adattare le interfacce utente ai numerosi dispositivi con risoluzioni diverse, ogni unità di misura è utile in diverse situazioni. Ad esempio:

- ***Pixel***, permettono di definire elementi con dimensioni precise, indifferentemente dalla densità dello schermo

- ***Pollici***, definiscono elementi in base alla dimensione dello schermo

- ***Dp***, pixel indipendenti dalla densità, ovvero unità astratta che si basa sulla densità fisica dello schermo

- ***Sp***, pixel indipendenti dalla scala, ovvero le dimensioni saranno regolate sia per la densità dello schermo che per le preferenze dell'utente impostate per il dispositivo.

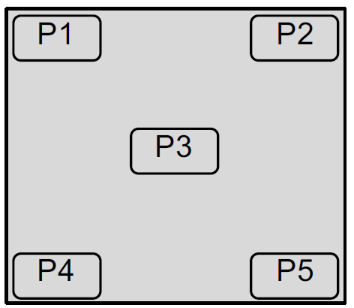
**A cosa serve la misurazione degli elementi grafici in “dpi” (density independent pixels). Si discuta del perché è necessaria e di come una cattiva gestione possa influire negativamente sull’aspetto grafico di un’app.**

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. Android traduce questo valore nel numero appropriato di pixel reali per ogni densità dello schermo.

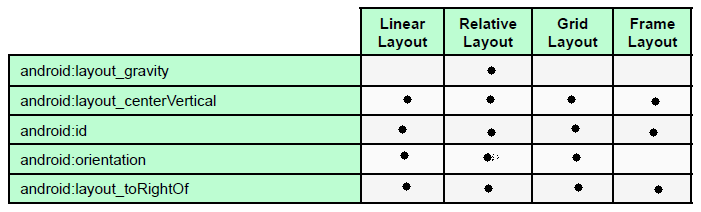
**ES:** Si considerino due schermi della stessa dimensione che hanno numero diversi di pixel. Se dovessi definire una vista larga "100px", apparirà molto più grande su uno dei due dispositivi, a seconda dei pixel a disposizione. Quindi si deve invece usare "100dp" per assicurarsi che appaia la stessa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

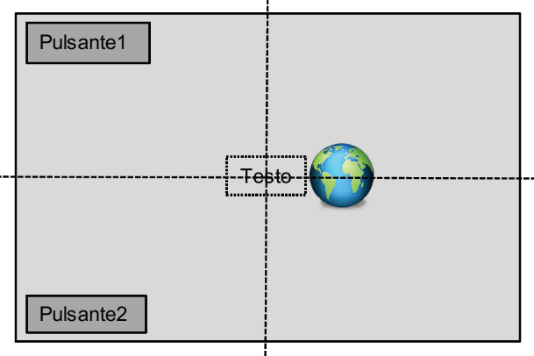
**Si consideri un’app il cui layout prevede 4 pulsanti ai 4 angoli del display. Tale layout può essere implementato in vari modi. Descrivine uno (a parole, senza codice XML, spiegando solo come dovrebbe essere fatto il codice XML) mettendone in evidenza eventuali vantaggi o svantaggi.**

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. 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 ***Button*** che verrà posizionato in basso a sinistra e necessario, oltre a mettere ***alignParentLeft***, anche “***alignParentBottom***” sempre a *true*. Per quello posizionato in basso a destra sarà sempre necessario mettere ***alignParentRight*** e ***alignParentBottom***, entrambi a *true*.

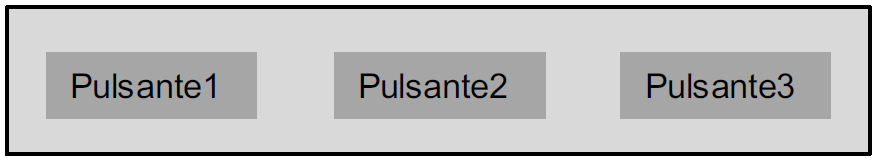
**Scrivere un file di layout che implementi l’interfaccia riportata in figura. Il rettangolo è un frame di 200x200px e deve essere posizionato al centro del display. I riquadri con etichetti “Pn” sono dei pulsanti la cui pressione deve essere collegata alla funzione “pulsantePremuto”**

|  |  |
| --- | --- |
| *<RelativeLayout*  *android:layout\_width="match\_parent"*  *android:layout\_height="match\_parent"*  *tools:context=".MainActivity">*  *<RelativeLayout*  *android:layout\_width="400dp"*  *android:layout\_height="400dp"*  *android:layout\_centerInParent="true"*  *android:background="#A9A9A9">*  *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="P1"*  *android:onClick="pulsantePremuto"*  */>*  *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="P2"*  *android:onClick="pulsantePremuto"*  *android:layout\_alignParentRight="true"*  */>*  *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="P3"*  *android:onClick="pulsantePremuto"*  *android:layout\_centerInParent="true"*  */>* | *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="P4"*  *android:onClick="pulsantePremuto"*  *android:layout\_alignParentBottom="true"*  */>*  *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="P5"*  *android:onClick="pulsantePremuto"*  *android:layout\_alignParentBottom="true"*  *android:layout\_alignParentRight="true"*  */>*  *</RelativeLayout>*  *</RelativeLayout>* |

**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.**

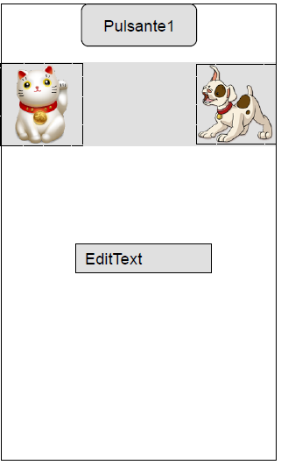
**Si scriva un frammento di codice XML che descriva il layout raffigurato a fianco scegliendo opportunamente gli elementi da utilizzare (le linee punteggiate non sono elementi del layout ma indicano il centro del layout).**

|  |  |
| --- | --- |
| *<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"*  */>*  *<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\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:layout\_toRightOf="@id/TV"*  *android:src="@drawable/world"*  */>*  *</RelativeLayout>* |

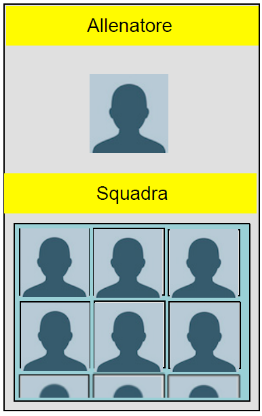
**Si costruisca tale layout utilizzando un file statico che specifica i primi 2 pulsante ed inserendo dinamicamente il terzo pulsante (dettagliare il file XML e lo snippet di codice che serve ad aggiungere il terzo pulsante).**

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

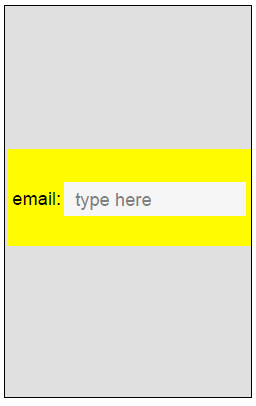
|  |  |
| --- | --- |
| *<LinearLayout*  *android:id="@+id/LL"*  *android:layout\_width="match\_parent"*  *android:layout\_height="match\_parent"*  *android:orientation="horizontal"*  *tools:context=".MainActivity">*  *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="Pulsante1"*  */>*  *<Button*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:text="Pulsante2"*  */>*  *</LinearLayout>* | protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  Button b3 = new Button(this);  b3.setText("Pulsante3");  LinearLayout ll = findViewById(R.id.LL);  ll.addView(b3);  } |

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

|  |  |
| --- | --- |
| **<RelativeLayout**  **android:layout\_width="match\_parent"**  **android:layout\_height="match\_parent"**  **android:background="#FFFFFF"**  *tools:context=".MainActivity">*  *<Button*  *android:id="@+id/B1"*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:layout\_centerHorizontal="true"*  *android:text="Pulsante1"/>*  *<FrameLayout*  *android:id="@+id/FLimage"*  *android:layout\_below="@id/B1"*  *android:layout\_width="match\_parent"*  *android:layout\_height="wrap\_content"*  *android:layout\_marginTop="40dp">*  *<ImageView*  *android:layout\_width="100dp"*  *android:layout\_height="100dp"*  *android:src="@drawable/cat"/>* | *<ImageView*  *android:layout\_width="100dp"*  *android:layout\_height="100dp"*  *android:layout\_gravity="right"*  *android:src="@drawable/dog"/>*  *</FrameLayout>*  *<EditText*  *android:layout\_below="@id/FLimage"*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:layout\_centerHorizontal="true"*  *android:text="EditText"*  *android:layout\_marginTop="100dp"/>*  **</RelativeLayout>** |

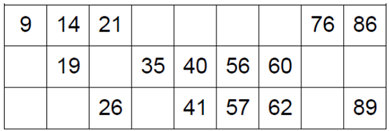
**Si scriva un file di layout che determini l’aspetto grafico mostrato nella figura. Il riquadro esterno rappresenta tutto lo schermo con uno sfondo di colore grigio chiaro. La foto dell’allenatore è una singola immagine, mentre le foto dei componenti della squadra sono visualizzate in un gridView. Il 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).**

|  |  |
| --- | --- |
| *<RelativeLayout*  *android:layout\_width="match\_parent"*  *android:layout\_height="match\_parent"*  *tools:context=".MainActivity"*  *android:background="#808080">*  *<TextView*  *android:id="@+id/TVallenatore"*  *android:layout\_width="match\_parent"*  *android:layout\_height="wrap\_content"*  *android:text="Allenatore"*  *android:padding="10dp"*  *android:textSize="20dp"*  *android:background="#FFFF00"*  *android:gravity="center"*  */>*  *<ImageView*  *android:id="@+id/IVallenatore"*  *android:layout\_width="100dp"*  *android:layout\_height="100dp"*  *android:src="@drawable/baby\_yoda"*  *android:layout\_below="@id/TVallenatore"*  *android:layout\_margin="40dp"*  *android:layout\_centerHorizontal="true"*  */>* | *<TextView*  *android:id="@+id/TVsquadra"*  *android:layout\_below="@id/IVallenatore"*  *android:layout\_width="match\_parent"*  *android:layout\_height="wrap\_content"*  *android:text="Squadra"*  *android:padding="10dp"*  *android:textSize="20dp"*  *android:background="#FFFF00"*  *android:gravity="center"*  */>*  *<GridView*  *android:id="@+id/GVsquadra"*  *android:layout\_below="@id/TVsquadra"*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:layout\_margin="10dp"*  *android:background="#007fff"*  *android:numColumns="3"*  *android:gravity="center"*  */>*  *</RelativeLayout>* |

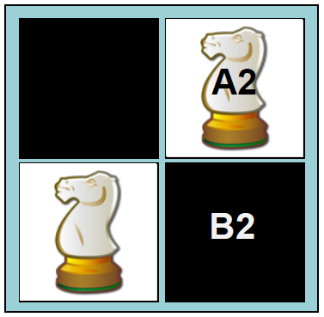
**Si scriva un file di layout che determini l’aspetto grafico mostrato nella figura. Il riquadro esterno rappresenta tutto lo schermo con uno sfondo di colore grigio chiaro, mentre il contenitore centrale con uno sfondo giallo ha un’altezza di 200dp. Il testo “email:” è visualizzato in un TextView. Di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

|  |
| --- |
| *<RelativeLayout*  *android:layout\_width="match\_parent"*  *android:layout\_height="match\_parent"*  *android:background="#A9A9A9"*  *tools:context=".MainActivity">*  *<RelativeLayout*  *android:layout\_width="match\_parent"*  *android:layout\_height="200dp"*  *android:layout\_centerInParent="true"*  *android:background="#FFFF00">*  *<TextView*  *android:id="@+id/TV"*  *android:layout\_width="wrap\_content"*  *android:layout\_height="wrap\_content"*  *android:layout\_centerVertical="true"*  *android:text="email: "/>*  *<EditText*  *android:layout\_toRightOf="@id/TV"*  *android:layout\_width="match\_parent"*  *android:layout\_height="wrap\_content"*  *android:layout\_centerVertical="true"*  *android:background="#FFFFFF"*  *android:hint="type here"/>*  *</RelativeLayout>*  *</RelativeLayout>* |

**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.**

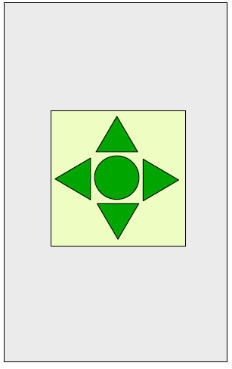


**Si descriva (senza codice) un modo per implementare il layout giustificano la scelta. Si tenga presente che, plausibilmente, l’app deve avere un riferimento per tutte le caselle che contengono un numero.**

**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.**

**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 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 grafici riportati in figura. Il rettangolo interno è un frame con un background colorato, mentre quello esterno è lo schermo del dispositivo**