**Prova scritta del 09/2017**

**1 – 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 più significativi o comunque i più utilizzati? Motivare la risposta.**

Ogni widget supporta degli attributi che:  
- Specificano l’aspetto grafico;  
- Specificano dove visualizzare l’elemento;  
- Forniscono informazioni.  
Alcuni attributi sono comuni a tutti gli elementi tra i più utilizzati troviamo:  
- Id (android:id=”@+id/text”) che specifica che stiamo aggiungendo un nuovo identificato al fine di poter ottenere un riferimento a quel widget dentro il codice sorgente.  
- layout\_width e layout\_height che sono usati per indicare l’altezza e la larghezza del widget. Di solito vengono utilizzati come valori mach parent (che indica che il widget prenderà tutto lo spazio disponibile del contenitore in cui si trova) o wrap\_content (che indica che prenderà solo lo spazio necessario).  
Altri attributi comuni a tutti i widget sono: margin, padding e molti altri…

**2 – 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.**

Abbiamo bisogno di 2 file XML, uno in cui viene rappresentata la view del singolo item della lista, e l’altro file XML (quello dell’activity che mostrerà la ListView) dove avremo appunto la visualizzazione vera e proprio della ListView.  
Successivamente definiamo un adapter a cui passeremo l’array dei nomi, la view del singolo item e l’assoceremo al widget ListView.  
Infine definiamo (se necessario) un listener per i click sugli elementi.

Schematicamente sarebbe:  
1 – Definire il widget ListView nel XML dell’activity che l’ospiterà.  
2 – Definire la view del singolo item in un file XML.  
3 – Definire l’array con gli elementi.  
4 – Definire un adapter.  
5 – Ottenere un riferimento al widget ListView.  
6 – Associare l’adapter al widget.  
7 – Definire un listener per i click sugli elementi.

**3 – Che cos’è il backstack? Supponendo che l’activity A sia l’unica in esecuzione (‘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, qual è il backstack a questo punto? Cosa succede se dall’activity D si preme il pulsante di back?  
Cosa si dive fare se si vuole fare in modo che dall’activity D, si torni direttamente ad A quando si preme il pulsante di back?**

Più activity possono coesistere tra loro e sono organizzate in un Backstack. Esempio: solitamente un task (insieme di Activity con le quali l’utente interagisce) parte dalla Home. Quando l’utente clicca su un’icona e lancia una nuova Activity, l’app è mostrata a schermo (foreground). Nel momento in cui vengono lanciate nuove Activity, la corrente è messa nel backstack e ci si può tornare col pulsante back.

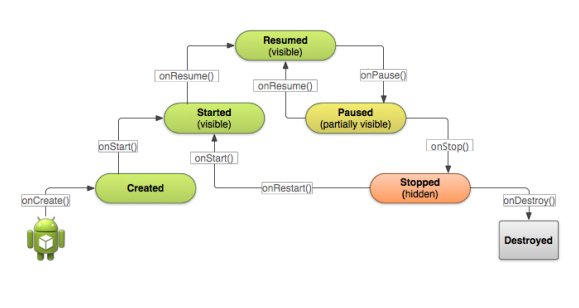
Un task con le sue activity può essere spostato in background (quando l’utente inizia un Task o clicca su home per esempio) e quindi le sue activity verranno messe in stato di Stop ma il loro backstack rimane intatto.

Il backstack considera solo le activity, per i fragments dobbiamo gestirli manualmente. Per inserire i cambiamenti nel backstack bisogna creare un metodo addToBackStack(). Nel caso in cui non chiameremo il metodo addToBackStack() i cambiamenti andranno persi.

Se l’activity D preme il pulsante back, D viene distrutta e viene rimossa dal backstack l’ultima activity che è stata inserita e viene riportata in stato di foreground.

Per passare dall’activity D direttamente ad A quando si preme il pulsante di back possiamo, o sovrascrivere il metodo onBackPressed in cui dichiariamo un intent che lancia l’activity A, oppure, quando passiamo dall’activity B all’activity C e dall’activity C all’activity D, dobbiamo settare un flag all’intent che ci permettere di cambiare activity ovvero, Intent.FLAG\_ACTIVTY\_NO\_HISTORY, e dopo aver chiamato il metodo startActivity(), chiameremo il metodo this.finish(), in questo modo l’activity B e C non verranno inserite nel backstack e verranno distrutte.

**4 – Si descriva il ciclo di vita delle attività attraverso un grafo che mostri gli stati nei nodi e i possibili passaggi con degli archi. Sugli archi si specifichi il metodo che fa passare da uno stato all’altro.**



**5 – Un’activity ActivityA ha bisogno di lanciare un’ActivityB dalla quale poi dovrà avere un valore (un intero) come risposta. Cosa si deve fare? Dettagliare la risposta con opportuni frammenti di codice.**

Class ActivityA extends Activity {  
 …  
 public void chiamaB() {  
 Intent I = new Intent();  
 i.setClass(getApplicationContext(), ActivityB.class);  
 startActivityForResult(i, 0);  
 }  
 …  
}

Class ActivityB extends Activity {  
 …  
 public void tornaA() {  
 Intent I = new Intent();  
 i.setClass(getApplicationContext(), ActivityA.class);  
 i.putExtra(“intero”, intero);  
 setResult(RESULT\_OK, i);  
 }

…  
}

**6 – 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) {  
 String recevide\_data = “”;  
 for(int I = 0; I < 100; i++) {  
 progressBarDownload.setProgress(i);  
 receovde\_data += str;  
 }

@Override  
 protected void onPostExecute(String data) {  
 progressBarDownload.setVisibility(ProgressBar.INVISIBLE);  
 textResponse.setText(data);  
 }

}

L’errore è che non vi è il metodo onProgressUpdate e che il metodo doInBackground modifica l’interfaccia utente, questo non è possibile poiché l’interfaccia utente può essere modificata solo dal thread main.  
Per questo motivo, l’operazione progressBarDownalod.setProgress(i), deve essere effettuata nel metodo onProgreessUpdate, e quindi dal main thread, inoltre al posto di progressBarDownload.setProgress(i) nel metodo doInBackGround bisogna chiamare il metodo publishProgress(i);

Modifiche:  
@Override  
protected String doInBackground(Integer… values) {  
 String recevide\_data = “”;  
 for(int I = 0; I < 100; i++) {  
 publishProgress(i);  
 receovde\_data += str;  
}

@Override  
protected void onProgressUpdate(Integer… values) {  
 progressBarDownload.setProgress(values[0]);  
}

**7 – Si scriva un file XML con un’animazione che permette di translare prima un oggetto prima in orizzontale per 500 pixel in 1 secondo e poi in verticale per 300 pixel in 2 secondi. Normalmente dopo l0animazione l’oggetto ritorna automaticamente nella posizione iniziale; cosa si può fare per farle rimanere nella posizione finale dell’animazione?**

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">  
  
 <translate  
 android:startOffset="0"  
 android:fromXDelta="0"  
 android:toXDelta="500"  
 android:duration="1000" />  
   
 <translate  
 android:startOffset="1000"  
 android:duration="2000"  
 android:fromYDelta="0"  
 android:toYDelta="300" />  
  
</set>

Per poter rimanere nella posizione finale basta settare all’animazione setFillAfter(true);

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

Il multitouch permette il rilevamento di uno o più tocchi sullo schermo.  
Il tocco è rappresentato dal pointer (singolo evento) e il movimento dal MotionEvent.  
Il MotionEvent rappresenta un movimento registrato da una periferica penna, trackball, mouse e dita sul display. Il movimento è rappresentato con un ACTION\_CODE (cambiamento avvenuto) e ACTION\_VALUES (posizione e proprietà del movimento).  
Gli ACTION\_CODE possono essere:  
- 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 lo tocca più.  
- ACTION\_UP: l’ultimo dito sullo schermo viene alzato.

**9 – Si descriva il meccanismo dei Frammenti.**

Un frammento è una porzione di activity. Non si tratta solo di un gruppo di controlli o di una sezione del layout. Può essere definito come una specie di sub-activity con un suo ruolo funzionale molto importante ed un suo ciclo di vita.

Un frammento non può vivere senza un’activity. Come detto il frammento ha il suo ciclo di vita fortemente collegato con quello dell’activity di appartenenza.

Un’activity può ospitare vari frammenti. I frammenti possono essere inseriti e rimossi duranti l’esecuzione. Si possono creare UI con molti frammenti anche in funzione della grandezza dello schermo.

La filosofia di progettazione che risiede dietro i frammenti riguardo lo sviluppo di un’interfaccia utente dinamica, in particolare capace di adattarsi sia a schermi grandi che a schermi piccoli.

Un frammento può comunicare con l’activity creando dei metodi di callback.

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

SharedPreferences vengono utilizzate per salvare dati all’interno di un file XML tramite una coppia chiave-valore. Per scrivere nel file xml abbiamo bisogno di un editor e successivamente possiamo usare il put, mentre per leggere i dati si usa la get.

Preferences è un pacchetto che fornisce classi per la gestione delle preferenze che riguardano solo gli strumenti dell’interfaccia utente.