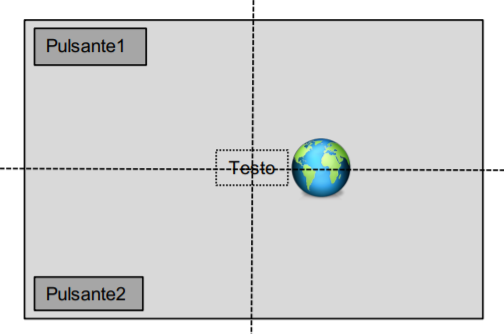
**Prova scritta del 07/2017**

**1 – 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).**

****

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 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:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Testo"  
 android:layout\_centerInParent="true"  
 android:id="@+id/testo" />  
  
 <ImageView  
 android:layout\_width="25dp"  
 android:layout\_height="25dp"  
 android:src="@drawable/right"  
 android:layout\_toRightOf="@id/testo"  
 android:layout\_centerVertical="true" />  
  
  
</RelativeLayout>

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

È un widget specifico per le liste dove divide l’area disponibile in varie posizioni. Gli elementi vengono memorizzati in un array, nel caso in cui abbiamo più elementi rispetto alle posizioni si potrà scorrere la lista.

L’adapter fornisce gli elementi da visualizzare in base allo scorrimento effettuato dall’utente.

L’interazione tra ListView e Adapter avviene associando l’adapter al widget ListView. Per il click sugli elementi bisogna definire un listener, infine occorrono 2 file xml uno contenente il widget ListView l’altro il singolo elemento della lista.

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

In caso di cambio di configurazione il sistema operativo distrugge e ricrea l’attività in esecuzione per meglio permettere all’app di adattarsi alla nuova configurazione.  
In seguita alla onDestroy però lo stato dell’activity viene perso, e quindi è necessario salvarlo nel metodo onSaveInstanceState() che verrà chiamato prima che l’activity venga distrutta. Lo stato viene recuperato nel metodo onCreate.

È però possibile gestire il cambiamento nel manifest tramite il metodo onConfigurationChanged() il quale sarà eseguito al posto della distruzione dell’activity. Quest’ultima strategia non è però consigliata.

Un’altra cosa molto importante che è possibile gestire, è la possibilità di poter creare un layout diverso in base al tipo di visualizzazione dell’activity, infatti, è possibile definire un layout per la visualizzazione portait e uno per la visualizzazione landscape (entrambi i file XML avranno lo stesso nome).

**5 – Un’activity ActivityA ha bisogno di lanciare un’ActivityB passando a tale activity un intero x ed una stringa s. Cosa si deve fare? Dettagliare la risposta con opportuni frammenti di codice.**

class ActivityA extends Activity {  
public void lanciaB() {  
Intent I = new Intent();  
i.setClass(getApplicationContext(), ActivityB.class);  
i.putExtra(“intero”, x);  
i.putExtra(“stringa”, s);  
startActivity(i);  
}  
}

class ActivityB extends Activity {  
public void onCreate(Bundle savedInstanceState) {

...  
Intent I = getIntent();  
int x = i.getIntExtra(“intero”);  
String s = i.getStringExtra(“stringa”);  
}  
}

**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 – 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?**

Il problema può essere che all’animazione non è stata assegnata una durata, poiché assegnando una durata all’animazione l’immagine viene eliminata successivamente all’animazione.

**8 – Per cosa sono utili i Frammenti?**

Un frammento è una porzione di Activity. Non si tratto solo di un gruppo di controlli o di una sezione del layout. Può essere definito più 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 durante 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 riguarda 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.

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

Molti smartphone, tablet hanno sensori di movimento, di ambiente, di posizione etc… I sensori forniscono dati “grezzi” e l’accuratezza dei dati dipende dalla qualità.

Un SensorManager ci dice i sensori disponibili e le caratteristiche del singolo sensore. Inoltre, ci permette di leggere i dati del sensore e usare Listeners sui cambiamenti dei dati.

Per ottenere un riferimento al SensorManager bisogna usare il metodo  
getSystemService(SENSOR\_SERVICE);

Grazie al SensorManager possiamo ottenere riferimenti ai vari sensori. Ad esempio, per ottenere un riferimento all’accelerometro bisognerà fare:  
sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

I listener che gestiscono i cambiamenti di stato del sensore devono essere registrati nel metodo onResume e rilasciati nel metodo onPause. Questo per evitare un consumo di batteria inutile.

Protected void onResume() {  
super.onResume();  
sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR\_DELAY\_UI);  
}

Protected void onPause() {  
super.onPause();  
sensorManager.unregisterListener(this);  
}

Per gesitere gli eventi, la classe deve implementare l’interfaccia SensorEventListener e quindi anche i suoi metodi.

Nel metodo onSensorChanged(SensorEvent event), possiamo gestire il cambiamento di stato del sensore tramite l’oggetto SensorEvent.

**10 – Si descrivano brevemente i Serivces, i BroadcastReceiver ed i ContentProvider.**

Service: svolgono un ruolo opposto all’activity. È un lavoro solitamente svolto in background senza bisogno di interazione diretta con l’utente. Si divide in due tipologie: started, ovvero, quando un’app ha bisogno di svolgere attività in background fino al loro completamento; bounded, ovvero, quando i servivi sono attivati solo nel caso in cui un’altra app abbia bisogno di connettersi a loro.

ContentProvider: sono componenti che permettono la condivisione nell’ambito del sistema di contenuti presenti all’interno di un database, file, o reperibili mediante accessi in rete.

BroadcastReceiver: è un componente che reagisce a invii di messaggi a livello di sistema con cui Android notifica l’avvenimento di un determinato evento. Non utilizzano interfaccia grafica ma possono inoltrare notifiche alla barra di stato per avvisare l’utente. La loro esecuzione dovrebbe essere istantanea.