**Prova scritta del 07/2015**

**1 – 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?**

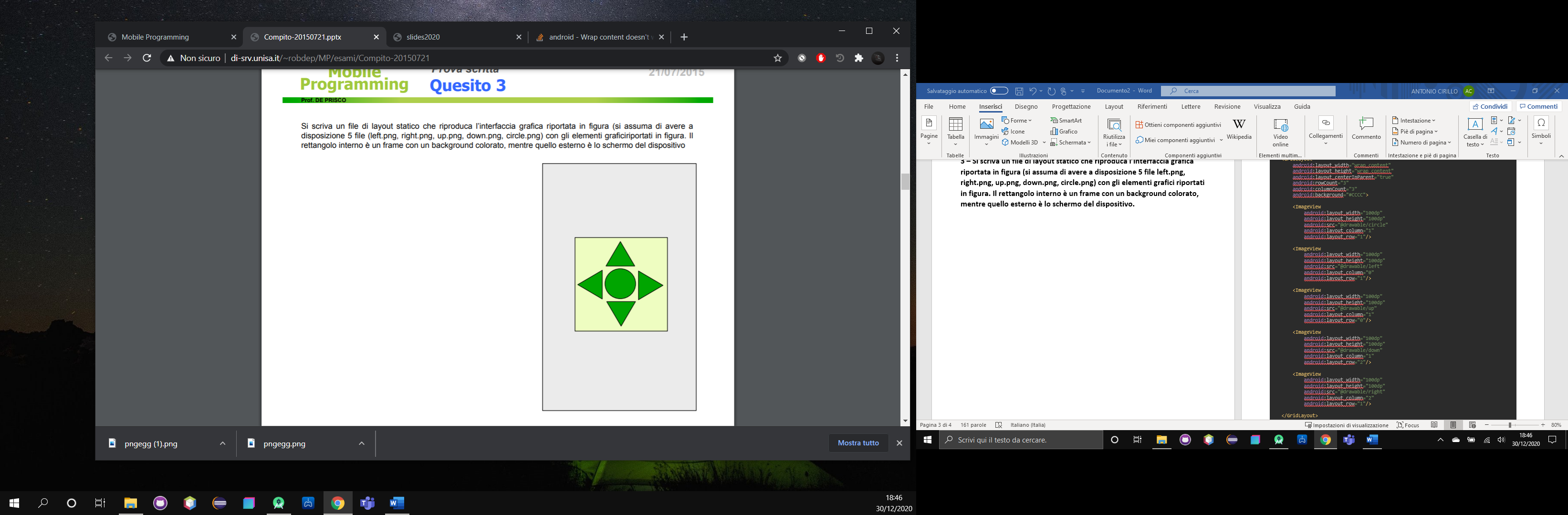
Il vantaggio di utilizzare un emulatore è la possibilità di poter emulare situazioni differenti, come la possibilità di poter far arrivare un messaggio, una chiamata, o semplicemente utilizzare dispositivi diversi e quindi testare la nostra applicazione su schermi piccoli e schermi grandi. L’unica grande pecca di questi emulatori è che possono essere lenti in alcuni casi.

**2 – I layout di un’app possono essere descritti o staticamente usando un file XML oppure dinamicamente eseguendo delle istruzioni a runtime (programmaticamente nel gergo Android). Si discuta di questi due approcci evidenziando vantaggi e svantaggi.**

Con un file XML (metodo dichiarativo) vengono costruiti layout statici. Sono facili da specificare e separano in modo netto la definizione dell’interfaccia dal codice dell’applicazione. Sono quindi facili da modificare. Il loro vero e proprio svantaggio appunto è che è possibile solo costruire layout statici.

In modo programmatico, ossia tramite istruzioni nel programma che saranno eseguite a runtime, è possibile costruire layout dinamici. Il layout è però da gestire nel codice dell’applicazione e quindi di gestione più complicata rispetto a quella con file XML.

**3 – 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.**



<GridLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:rowCount="3"  
 android:columnCount="3"  
 android:background="#CCCC">  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/circle"  
 android:layout\_column="1"  
 android:layout\_row="1"/>  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/left"  
 android:layout\_column="0"  
 android:layout\_row="1"/>  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/up"  
 android:layout\_column="1"  
 android:layout\_row="0"/>  
   
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/down"  
 android:layout\_column="1"  
 android:layout\_row="2"/>  
  
 <ImageView  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/right"  
 android:layout\_column="2"  
 android:layout\_row="1"/>  
  
 </GridLayout>  
  
</RelativeLayout>

**4 – Si spieghi che cos’è un “adapter” per un ListView. Si faccia un esempio scrivendo uno snippet di codice.**

L’adapter è un component collegato ad una struttura dati di oggetti Java (array, Collections, risultati di query) e che incapsula i l meccanismo di trasformazione di questi oggetti in altrettante view da mostrare su layout.  
Quando viene creato un adapter, gli viene assegnata una struttura dati, ad esempio un array, il quale conterrà la lista di oggetti che devono essere visualizzati all’interno della ListView. L’adapter, inoltre, deve conoscere il modo in cui devono essere visualizzati gli oggetti, e quindi deve avere a disposizione un file xml che descrive la struttura delle view per ogni singolo oggetto.

@Override

protected void onCreate(Bundle savedInstancesState) {

super.onCreate(savedInstancesState);

String[] città = new String[]

{"Torino", "Roma", "Milano", "Napoli", "Firenze"};

ArrayAdapter <String> adapter = new ArrayAdapter <String> (this,

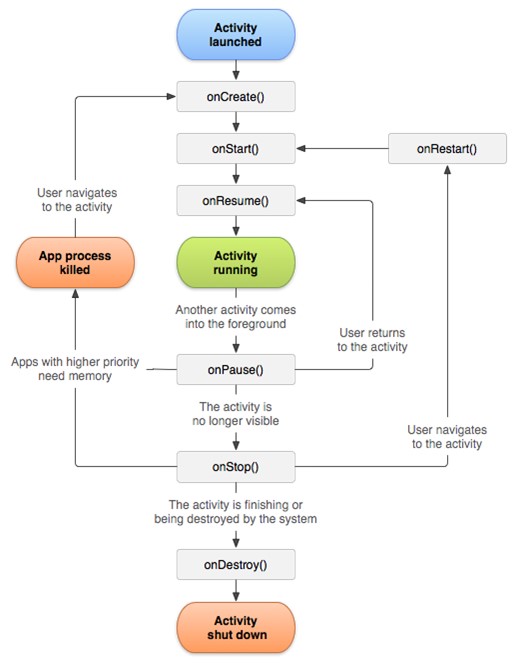
R.layout.row, città);

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

listView.setAdapter(adapter);

}

**5 – 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.**

****

Le transizioni che avvengono in automatico sono i metodi onCreate, onStart, onResume al lancio dell’activity.  
Mentre il metodo onPause viene chiamato quando l’activity passa in foreground, ovvero quando l’activity A lancia un’activity B; il metodo onStop viene chiamato quando l’activity non è più visibile, ad esempio quando passiamo su un’altra applicazione o semplicemente torniamo all’home screen senza però chiudere definitivamente l’applicazione.  
onDestroy invece viene chiamato automaticamente quando viene distrutta l’applicazione.

**6 – Si descriva il meccanismo dei permessi usato da Android per permettere alle app di accedere alle risorse del dispositivo.**

I permessi in Android vengono suddivise in tre famigli in base al livello di protezione:  
- permessi normali, che non metto cioè a rischio la privacy dell’utente;  
- permessi pericolosi, potenzialmente più lesive della riservatezza degli utenti;  
- permessi di firma, gestite a livello di installazione ma utilizzabili solo le l’app che le richiede è firmata con lo stesso certificato di quella che ha definito i permessi.

I permessi pericolosi vanno accettati a runtime dall’utente al momento dell’utilizzo della funzionalità che le richiedono per API >= 23, mentre quando si installa l’app per API < 23, mentre per quelle normali è sufficienti dichiararne l’impiego al momento dell’installazione dell’app. Ciò che cambia sul piano della progettazione è che l’app, in prospettiva potrebbe avere più comportamenti: uno in cui tutte i permessi pericolosi sono stati concessi, altri in cui uno o più di essi sono stati negati o revocati.

**7 – Che differenza c’è fra il main thread di un’app e i thread background? Quali operazioni vengono solitamente svolte nel main thread e quali nei thread in background?**

Le operazioni che dovrebbero essere preferibilmente svolte su thread secondari sono tutte le operazioni “lente”, ma, obbligatoriamente, tutte le operazioni che richiedono accesso in rete devono essere eseguite su thread background.  
I thread background però non possono in nessun modo interagire con l’interfaccia utente poiché quest’ultima è riservata esclusivamente al thread main. Per questo motivo i thread secondari devono comunicare con il thread main quando necessitano di apportare modifiche all’interfaccia grafica dato che non possono farlo in modo diretto,

**8 – Cosa sono i frammenti in un’app? Si descriva una situazione in cui è comodo usare i frammenti per sviluppare un’app.**

Un fragment è una porzione di activity che ha un suo ruolo funzionale molto importante ed un suo ciclo di vita fortemente collegato con quello dell’activity di appartenenza, ad esempio, se l’acitivity è in stato “paused” lo saranno di conseguenza anche tutti i suoi frammenti.  
  
Una situazione in cui è comodo usare i frammenti potrebbe essere quella in cui un’activity deve mostrare più scenari composti dallo stesso layout, ad esempio, un’applicazione che mostra varie cartelle per giocare a tombola. Queste cartelle possono essere vari frammenti che utilizzano tutti lo stesso layout e che a loro volta vanno inserite nell’apposito spazio riservato a loro nell’activity.

**9 – A cosa servono le SharedPreferences? Si faccia un esempio scrivendo degli snippet di codice.**

La SharedPreferences non è nient’altro che una mappa in cui salvare coppie chiave/valore in modo persistente.  
I dati collocati nella SharedPreferences vengono salvati in un file XML contenuto nello storage interno, precisamente nella cartella shared\_prefs. Il file in cui sono contenute può avere un nome di default o assegnato dal programmatore.

SharedPreferences prefs = getSharedPreferences(“File”, MODE\_PRIVATE);  
// PER OTTENERE DATI  
score = prefs.getInt(“maxScore”, 0);  
// PER SCRIVERE DATI  
SharedPreferences.Editor editor = prefs.edit();  
editor.putInt(“maxScore”, 100);  
editor.commit();

**10 – 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).**

Le animazioni in Android possono essere create sia con XML sia in Java.  
In XML per creare delle animazioni ci basterà creare un file xml all’interno della cartella res/anim. Il file xml ha come radice il tag <set> dove al suo interno vi sono le operazioni che verranno effettuate per quell’animazione. Vi sono varie operazioni che possono essere effettuate, e per giunta possono anche essere combinate tra loro per creare nuove operazioni. Alcuni di esse sono rotate, scale, translate, e tant’altre.   
Ognuna di essa ha degli attributi specifici. Uno snippet di codice:  
<set xmlns:android="http://schemas.android.com/apk/res/android">

<rotate

android:fromDegrees="0"

android:toDegrees="2000"

android:pivotX="50%"

android:pivotY="50%"

android:duration="360"

android:repeatMode="restart"

android:repeatCount="3"

android:interpolator="@android:anim/cycle\_interpolator"/>

</set>

In java invece, le animazioni vengono create con l’ausilio della classe ViewPropertyAnimator. Questa classe offre le animazioni più comuni, che possono essere invocate mediante appositi metodi come, rotation, scale, translation, alpha.  
Per ottenere un riferimento ad un oggetto ViewPropertyAnimator è necessario invocare il metodo animate() sulla view.

Uno snippet di codice può essere:  
txt.animate().rotationY(180);