UNIVERSITÀ DI SALERNO

Corso di laurea in Informatica

Mobile Programming

Prof. De Prisco

Prova scritta del

esempio 1

NOME: _____

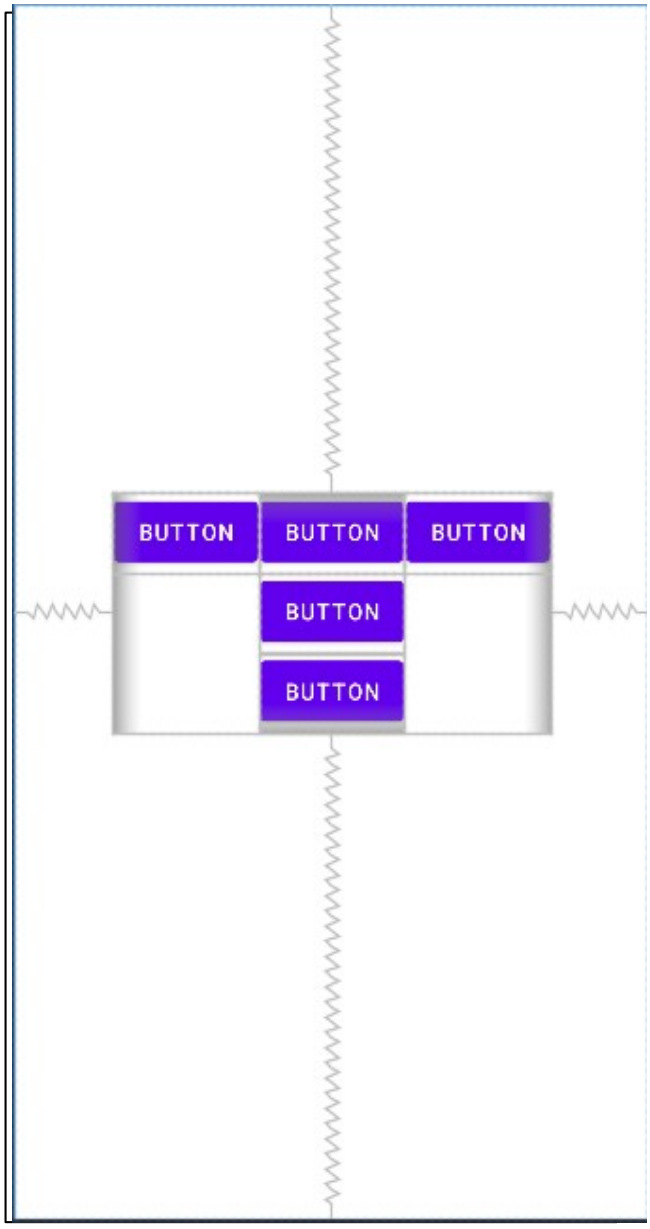
COGNOME:_____

MATRICOLA: _____

Domande	Punti
1	/10
2	/10
3	/10
4	/10
5	/10
6	/10
7	/10
8	/10
9	/10
10	/10
TOTALE	/100

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>
```



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 object = getItem(position);

      TextView stringTextView = (TextView) v.findViewById(R.id.stringa);
      TextView intTextView = (TextView) v.findViewById(R.id.intero);

      stringTextView.setText(object.getString());
      intTextView.setText(String.valueOf(object.getInt()));

      return v;
      }
      }
```

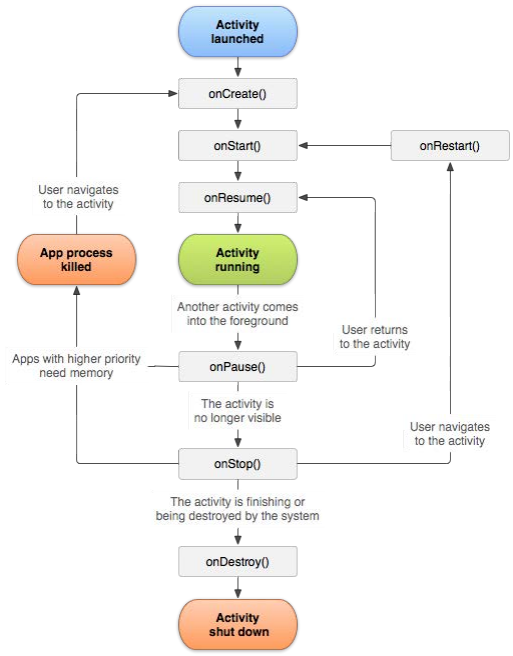
Si spieghi il meccanismo del backstack.
In relazione a tale meccanismo che differenza c'è fra una activity e un frammento?

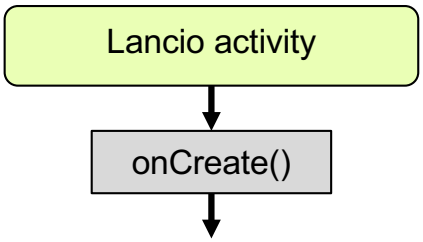
Il back stack (o "stack di navigazione") è una struttura di dati che tiene traccia delle attività aperte all'interno dell'applicazione. Quando un'attività viene avviata, viene aggiunta alla cima dello stack; quando l'utente esce da un'attività tornando alla schermata precedente, l'attività viene rimossa dallo stack.

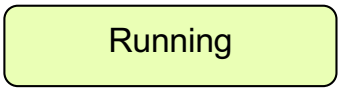
Un frammento è una porzione di un'interfaccia utente in un'attività. Un'attività può contenere uno o più frammenti e può essere utilizzata per raggruppare elementi dell'interfaccia utente in modo da poterli gestire facilmente. Ad esempio, se si vuole creare una schermata di dettaglio con una lista di elementi e una sezione di dettaglio che mostra i dettagli di un elemento selezionato, si può utilizzare un frammento per la lista e un altro frammento per la sezione di dettaglio.

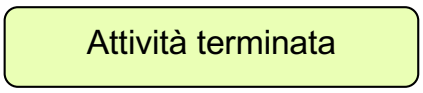
La differenza principale fra un'attività e un frammento è che un'attività rappresenta un'intera schermata mentre un frammento rappresenta solo una parte della schermata. Inoltre, i frammenti possono essere riutilizzati all'interno di diverse attività, mentre ogni attività ha una schermata indipendente.

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().









onStart() è un metodo della classe Activity che viene chiamato quando l'attività viene avviata.
In genere, in onStart() si svolgono le seguenti operazioni:

Inizializzare i componenti dell'interfaccia utente, come ad esempio TextView, Button, etc.
Collegare gli eventi dei componenti dell'interfaccia utente ai relativi listener.
Inizializzare le variabili globali dell'attività.
onStop() è un metodo della classe Activity che viene chiamato quando l'attività viene fermata. In genere, in onStop() si svolgono le seguenti operazioni:

Chiudere tutte le connessioni aperte, come ad esempio quelle a database o a servizi web.
Liberare le risorse acquisite dall'attività, come ad esempio file aperti o connessioni di rete.
Salvare lo stato dell'attività, ad esempio in modo da poter riprendere l'attività esattamente dove l'utente l'ha lasciata quando verrà ripresa in seguito.
onResume() è un metodo della classe Activity che viene chiamato quando l'attività viene ripresa. In genere, in onResume() si svolgono le seguenti operazioni:

Riprendere le attività sospese, ad esempio riprendere un video o una musica che stavano riproducendo prima che l'attività venisse messa in pausa.
Aggiornare i dati visualizzati sullo schermo, ad esempio scaricando nuovi dati da un server o dal database.
onPause() è un metodo della classe Activity che viene chiamato quando l'attività viene messa in pausa. In genere, in onPause() si svolgono le seguenti operazioni:

Sospendere le attività in corso, ad esempio interrompendo la riproduzione di un video o di una musica.
Salvare lo stato dell'attività, ad esempio per poter riprendere l'attività esattamente dove l'utente l'ha lasciata quando verrà ripresa in seguito.
Rilasciare le risorse acquisite dall'attività, come ad esempio file aperti o connessioni di rete, in modo da liberare memoria e risorse di sistema per le altre attività in esecuzione.

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:

```
Intent intent = new Intent(this, DestinationActivity.class);
startActivity(intent);
```

Intent implicito:

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(intent);
```

Per lanciare un Intent implicito, è necessario specificare un'azione e un URI. L'azione specifica il tipo di operazione che si desidera eseguire (ad esempio, visualizzare una pagina web), mentre l'URI fornisce i dettagli sull'oggetto su cui si desidera eseguire l'azione (ad esempio, l'indirizzo della pagina web da visualizzare). L'Intent viene quindi inviato al sistema, che individuerà le app che possono gestire l'Intent e mostrerà all'utente un elenco di opzioni per scegliere l'app da utilizzare.

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

AsyncTask è una classe che viene utilizzata per eseguire operazioni in background in un'applicazione Android. La classe AsyncTask prende tre tipi di parametri generici: Type1, Type2 e Type3. Ecco cosa rappresentano questi tipi:
Type1: il tipo di dati utilizzato come input per la task in background. Ad esempio, se la task richiede dei parametri per essere eseguita, Type1 sarà il tipo di dati di questi parametri.
Type2: il tipo di dati utilizzato per indicare il progresso della task. Ad esempio, se si desidera mostrare una barra di avanzamento durante l'esecuzione della task, Type2 sarà il tipo di dati utilizzato per indicare il valore della barra di avanzamento.
Type3: il tipo di dati utilizzato per il risultato della task. Ad esempio, se la task restituisce un risultato a seguito dell'esecuzione, Type3 sarà il tipo di dati di questo risultato.
Ecco un esempio di utilizzo di AsyncTask per scaricare un file da Internet:

```
private class DownloadTask extends AsyncTask<String, Integer, String> {
    @Override
    protected String doInBackground(String... params) {
        String url = params[0];
        String result = "";
        try {
            URL downloadUrl = new URL(url);
            HttpURLConnection connection = (HttpURLConnection) downloadUrl.openConnection();
            connection.connect();
            int fileLength = connection.getContentLength();
            InputStream inputStream = connection.getInputStream();
            BufferedInputStream bis = new BufferedInputStream(inputStream);
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            byte[] data = new byte[1024];
            long total = 0;
            int count;
            while ((count = bis.read(data)) != -1) {
                total += count;
                publishProgress((int) (total * 100 / fileLength));
                baos.write(data, 0, count);
            }
            result = baos.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return result;
    }
    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        progressBar.setProgress(values[0]);
    }
    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);
        textView.setText(result);
    }
}
```

In questo esempio, Type1 è String (perché il costruttore di AsyncTask riceve un array di stringhe come parametro), Type2 è Integer (perché viene utilizzato per indicare il progresso della task) e Type3 è String (perché il metodo doInBackground restituisce una stringa come risultato).

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.

Ci sono diversi modi per condividere dati tra diverse Activity (che sono componenti dell'interfaccia utente di un'applicazione Android) che fanno parte della stessa app:

Intent: puoi utilizzare gli Intent per passare dati da una Activity all'altra. Gli Intent sono messaggi che possono essere inviati dalle componenti dell'applicazione (come Activity, Service, BroadcastReceiver) per richiedere l'esecuzione di un'azione da parte di un'altra componente dell'applicazione. Gli Intent possono contenere dati sotto forma di Extra, che possono essere aggiunti utilizzando il metodo putExtra(). Quando si passano dati di grandi dimensioni (come un'immagine) tramite gli Intent, è consigliabile utilizzare un FileProvider per evitare problemi di sicurezza.
Vantaggi:

Facile da usare e veloce da implementare
Non richiede l'utilizzo di altre risorse (come il database o le SharedPreferences)
Svantaggi:

I dati condivisi tramite gli Intent sono disponibili solo per il periodo di tempo in cui l'Intent viene gestito. Una volta che l'Intent è stato gestito, i dati non sono più disponibili.
Gli Intent possono essere utilizzati solo per passare dati tra componenti dell'applicazione (ad esempio, da una Activity a un'altra Activity). Non è possibile utilizzare gli Intent per condividere dati con altre app.
Database: puoi utilizzare un database per condividere dati tra diverse Activity della tua app. Un database è una raccolta di dati organizzati in modo da consentirne l'accesso, la gestione e l'aggiornamento. Android offre diversi tipi di database, come il SQLite e il Room, che possono essere utilizzati per creare e gestire un database all'interno dell'app.
Vantaggi:

I dati sono persistenti, ovvero rimangono disponibili anche dopo che l'app è stata chiusa o il dispositivo è stato spento.
È possibile utilizzare un database per condividere dati con altre app utilizzando un ContentProvider.
Svantaggi:

L'implementazione di un database può essere più complessa rispetto ad altre soluzioni.
L'utilizzo di un database può aumentare la dimensione dell'app e ridurre le prestazioni se utilizzato in modo inappropriato

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:



               break;
         case MotionEvent.ACTION_POINTER_DOWN:



               break;
         case MotionEvent.ACTION_MOVE:



               break;
         case MotionEvent.ACTION_POINTER_UP:



               break;
         case MotionEvent.ACTION_UP:

                        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:
                                 // non c'è bisogno di fare nulla qui, poiché il numero di dita che
                        stanno toccando lo schermo non cambia durante il movimento
                                 break;
                              case MotionEvent.ACTION_POINTER_UP:
                                 counter--;
                                 break;
                              case MotionEvent.ACTION_UP:
                                 counter--;
                                 break;
                           }
                           return true;
                        }
      }
      return true;
}
```

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?

Quando viene creato un layout in un'applicazione Android, le view vengono posizionate e dimensionate sullo schermo utilizzando un sistema di coordinate basato su pixels. Per prima cosa, il sistema Android misura le dimensioni dello schermo del dispositivo (in pixels). Successivamente, assegna una posizione e una dimensione alle view del layout in base alle regole di layout specificate (ad esempio, utilizzando layout manager o regole di layout definite in file XML).

Il metodo getWidth() di una view restituisce la larghezza della view in pixels. Il metodo getHeight() restituisce l'altezza della view in pixels. Se questi metodi sono utilizzati in onCreate(), potrebbero restituire 0 se la view non è stata ancora misurata e posizionata. Ciò può accadere perché la view potrebbe essere ancora in fase di rendering o perché potrebbe dipendere da altre view che ancora non sono state completamente create.

Per assicurarsi che la view sia stata correttamente misurata e posizionata, è consigliabile utilizzare i metodi getWidth() e getHeight() in un metodo come onWindowFocusChanged(), che viene chiamato quando l'activity ottiene o perde il focus della finestra. In questo modo, è garantito che la view sia stata completamente creata e posizionata prima di tentare di accedere alle sue dimensioni.

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

Un Toast customizzato è un Toast (una piccola notifica temporanea che viene visualizzata sullo schermo) che viene modificato dal suo aspetto predefinito per soddisfare le esigenze dell'applicazione. Ciò può includere il cambiamento del colore di sfondo, del testo o dell'icona visualizzata.

Per implementare un Toast customizzato, è necessario seguire i seguenti passaggi:

Creare un layout personalizzato per il Toast. Questo può essere fatto creando un file XML nella cartella res/layout dell'applicazione e definendo la struttura del Toast utilizzando elementi di layout come TextView o ImageView.

Caricare il layout personalizzato in una vista. Questo può essere fatto utilizzando il metodo LayoutInflater.inflate() e passando il layout personalizzato come argomento.

Creare un Toast utilizzando il metodo Toast.makeText() e passando il contesto dell'applicazione e la vista personalizzata come argomenti.

Modificare le proprietà del Toast, come il colore di sfondo o il testo, utilizzando i metodi appropriati (ad esempio, setBackgroundColor() o setText()).

Visualizzare il Toast utilizzando il metodo show().

```
LayoutInflater inflater = getLayoutInflater();
View customToastLayout = inflater.inflate(R.layout.custom_toast,
(ViewGroup) findViewById(R.id.custom_toast_container));
TextView text = (TextView) customToastLayout.findViewById(R.id.text);
text.setText("This is a custom toast!");

Toast customToast = Toast.makeText(getApplicationContext(), "",
Toast.LENGTH_LONG);
customToast.setView(customToastLayout);
customToast.show();
```

Pagina aggiuntiva

Pagina aggiuntiva

Pagina aggiuntiva