# Corso Interazione Uomo-Macchina

a.a. 2019-20

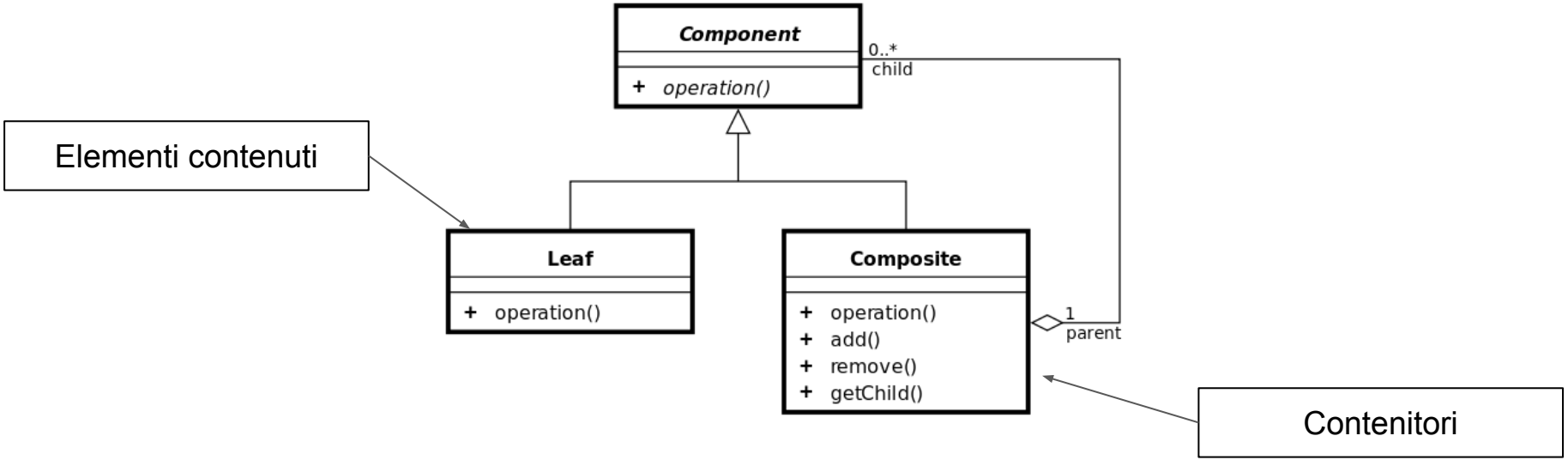

Francesco Cutugno
Enrico Leone
Vincenzo Norman Vitale
Marco Grazioso

# Outline

- View e Layout
- Creazione di UI
- Integrazione in Java

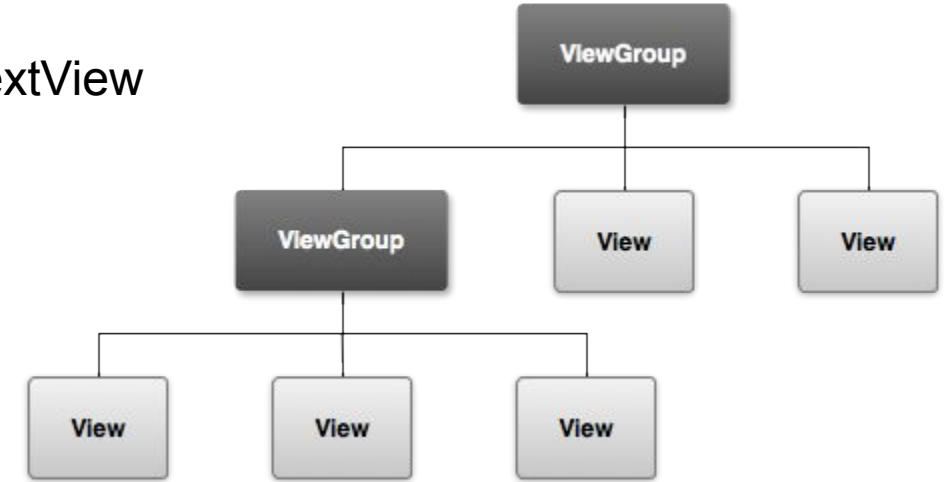
# Modello delle GUI

## Implementazione del Composite Pattern



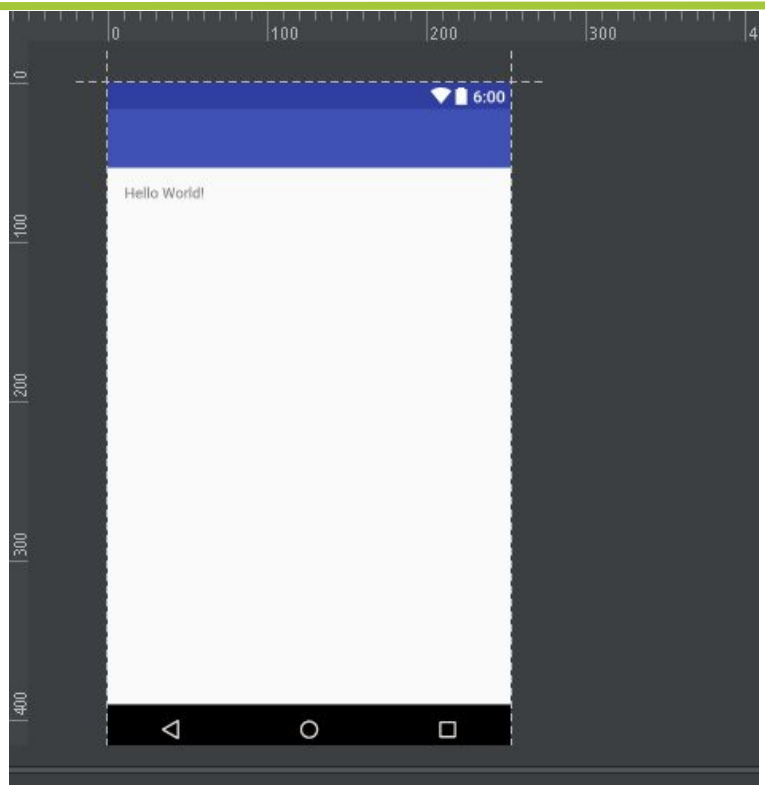
# View e ViewGroup

- **ViewGroup**: Contenitore invisibile che può contenere altri ViewGroup e View
- **View**: elemento base della UI es. textView



# I Layout

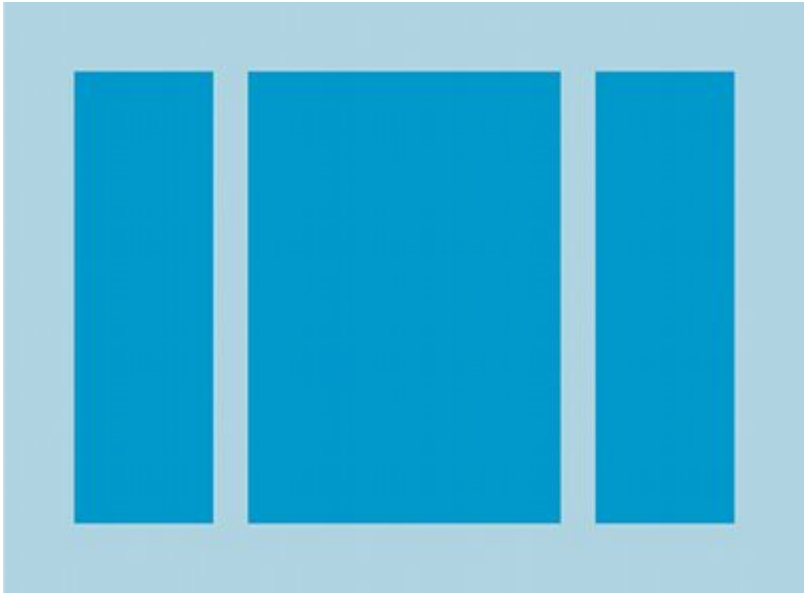
- **LinearLayout**: Allinea ogni View in orizzontale o in verticale
- **RelativeLayout**: La posizione delle View è definita in relazione alle View presenti
- **GridLayout**: Usa una griglia per posizionare le View



# LinearLayout

android:orientation

- horizontal
- vertical



android:layout_weight

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.LinearLayout

# RelativeLayout

android:layout_alignParentStart (end, top, bottom)

- true/false

android:layoutCenterHorizontal (vertical)

- true/false

android:layout_to(Start/End)Of

android:layout_above/below

- ID della view

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.RelativeLayout

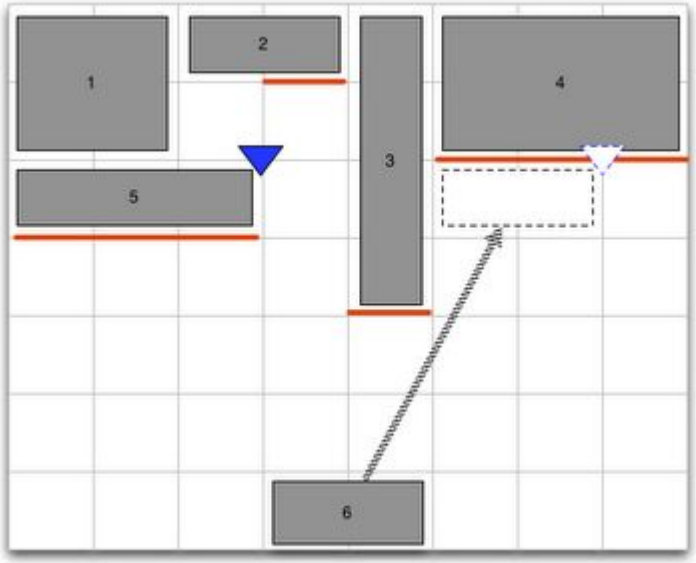
# GridLayout

android:columnCount

Negli elementi:

- android:layout_columnSpan
- android:layout_column/row

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.GridLayout



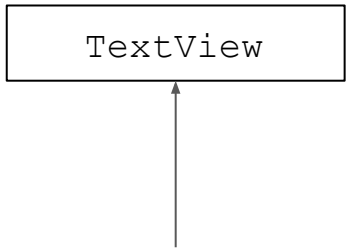
# ScrollView

Contiene una lista di View che possono essere “*scrollate*” dall’utente

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.FrameLayout
↳ android.widget.ScrollView

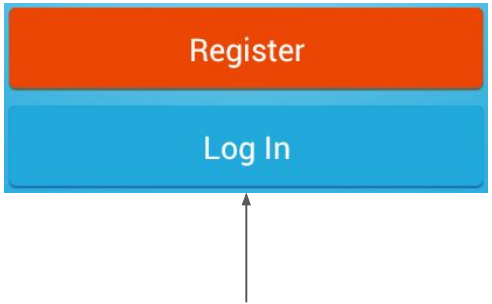
# Elementi Grafici

- Riempiono i Layout
- Permettono l'interazione con l'utente
- Possono generare eventi




Visualizzazione: ref


Inserimento: ref



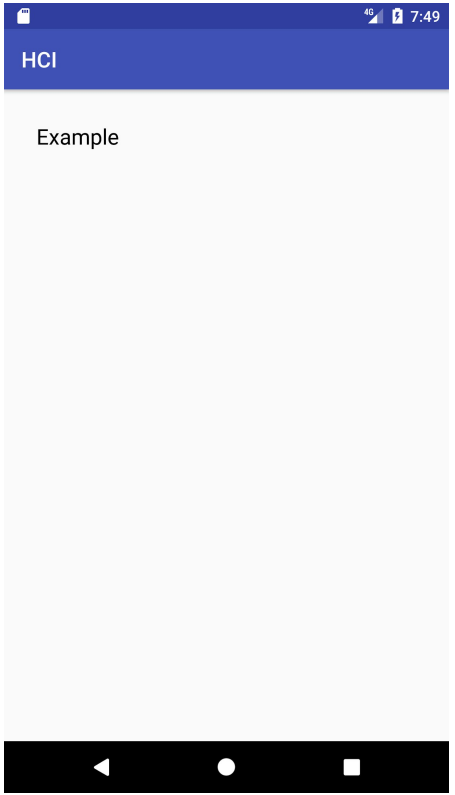

Azioni: ref

# TextView

- Visualizza stringhe sulla schermata
- Utili a spiegare l'interfaccia
- Il testo può essere selezionabile o meno

java.lang.Object
↳ android.view.View
↳ android.widget.TextView

```
<TextView
    android:id="@+id/myTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/string_code"
    android:textColor="@android:color/black"
    android:textSize="20sp" />
```

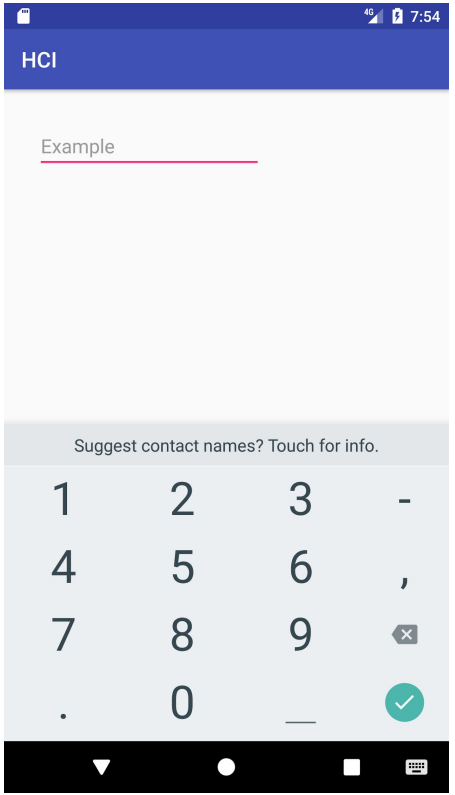


# EditText

- Permette l'inserimento di dati
- Eredita da TextView

java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.EditText

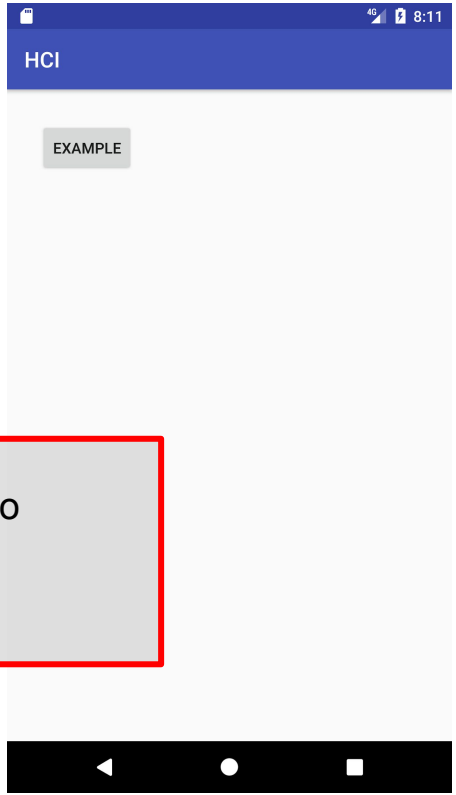
```
<EditText
    android:id="@+id/editText3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="@string/string_code"
    android:inputType="number" />
```



# Button

- Realizza l'interazione con l'utente
- Eredita da TextView

java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.Button

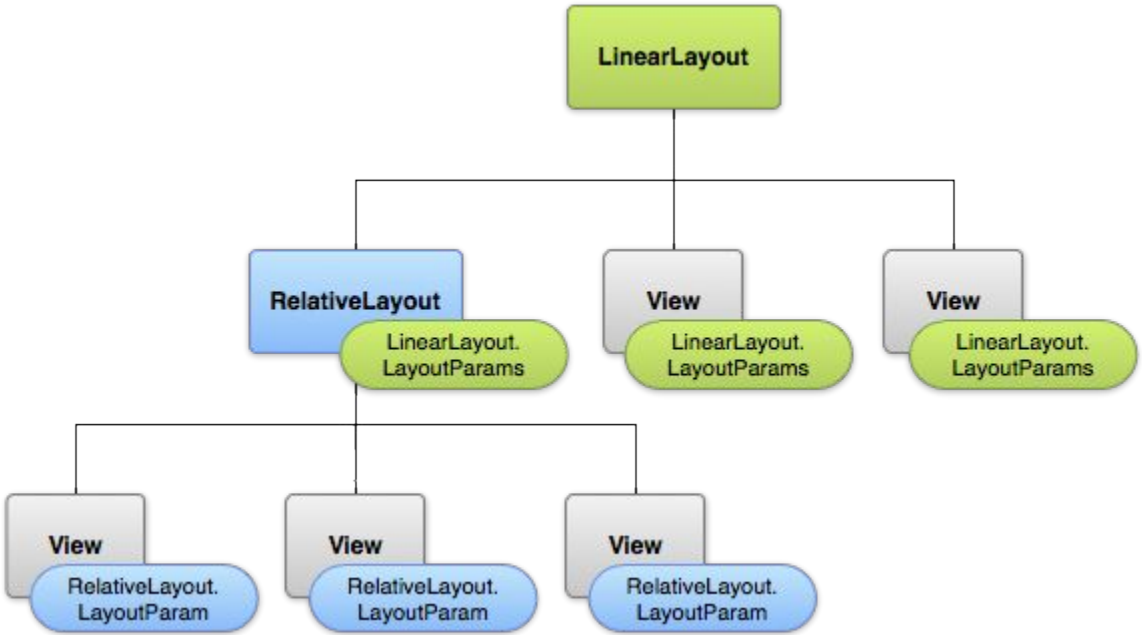



```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/string_code"
    android:onClick="onClick" />
```

- Richiede di specificare il contesto
- Riduce la portabilità

# Layout Parameters

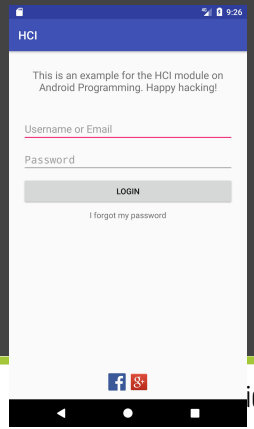
## match_parent *vs* wrap_content



# GUI in Android

```
<?xml version="1.0" encoding="utf-8"?>
<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">
   <TextView />
   <LinearLayout>
      <EditText />
      <EditText android:inputType="textPassword"/>
      <Button />
      <TextView />
   </LinearLayout>

</RelativeLayout>
```




```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:orientation="horizontal">
    <ImageView
        android:id="@+id/fbLogin"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:contentDescription="@string/fb_login"
        android:background="@drawable/fb"/>
    <ImageView
        android:id="@+id/googleLogin"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_marginStart="@dimen/internalMargin"
        android:contentDescription="@string/google_login"
        android:background="@drawable/gplus"/>
</LinearLayout>
```

# Assegnare UI

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
}
```

Metodo per invocare una view o un layout

ID del layout da caricare

# findViewById

Recuperare gli elementi grafici dal layout nelle Activity

```
View findViewById(int reference)
```

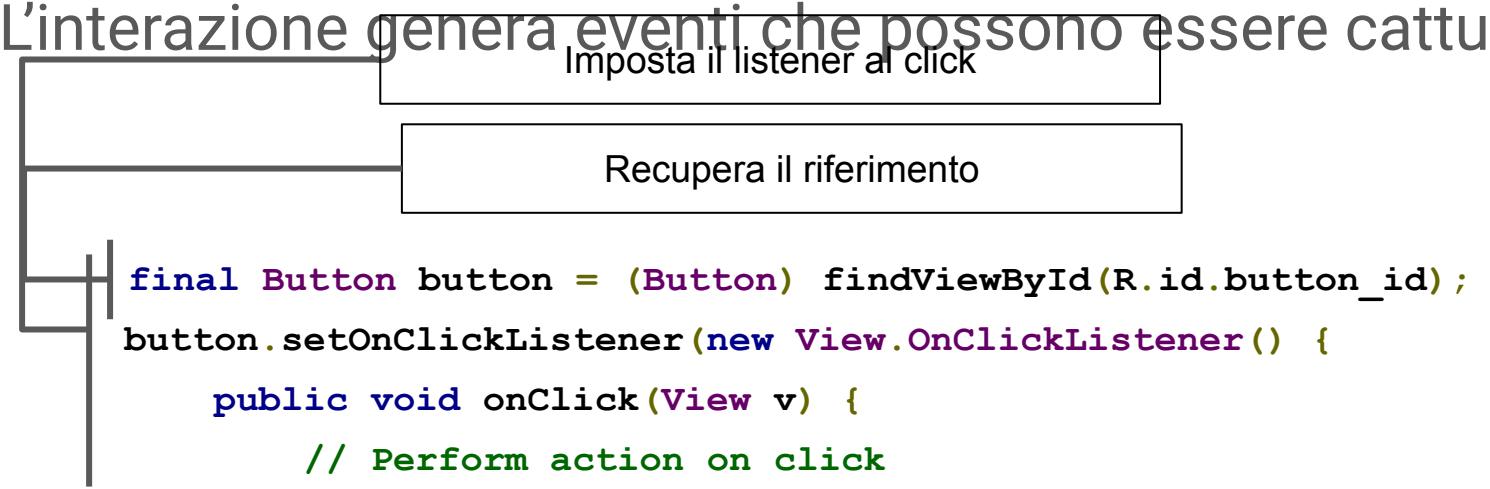
Superclasse

ID (*R.id.submit_button*)

Stesso metodo per tutti i tipi di elementi

# Listener

L'interazione genera eventi che possono essere catturati




```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Perform action on click
    }
});
```

# Esercizi

**Es. 1**

- Creare un LinearLayout con 5 TextView
- Sono tutte vuote tranne una
- Al click di un Button, il testo si sposta in alto (a rotazione)

**Es. 2**

- Creare un RelativeLayout con 2 TextView T1 e T2
- T1 dipende dal layout, T2 dipende da T1
- Al click di un Button, muovere T1 (cambiare marginTop e marginStart)

```
LinearLayout ll;
ll.getChildCount() // Restituisce il numero di View contenute
ll.getChildAt(i)   // Accede all'i-esima View di ll
```

# Esercizio II

- Aggiungere un RadioGroup nell'XML
- Al click di un Button, aggiungere un RadioButton al gruppo, lasciando solo l'ultimo selezionato (*checked*)

```
RadioGroup rg;

rg.getChildCount() // Restituisce il numero di View contenute
rg.getChildAt(i)   // Accede all'i-esima View di rg
```