Widget ed Eventi

Click di un button

```
Button btn= (Button)findViewById(R.id.updateButton);

btn.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View view) {
                  // Gestione dell'evento
            }
      });
```

Listener della classe View

Nome evento	Interfaccia Listener	Descrizione
click	View.OnClickListener	Evento di selezione di un componente.
long click	View.OnLongClickListener	Evento di hold, ovvero di selezione prolungata di un componente.
focus change	View.OnFocusChangeListener	Evento di acquisizione o perdita del focus da parte di un componente.
key	View.OnKeyListener	Evento di selezione di un tasto.
touch	View.OnTouchListener	Evento di touch.
createContextMenu	View.OnCreateContextMenuListener	Evento di creazione del menu contestuale.

Template dei metodi setter per i Listener

public void setOn<Evento>Listener (View.On<Evento>Listener l)

Event Handler

- sono metodi della classe View che vengono invocati in conseguenza di eventi
  - Si possono sovrascrivere

- gestione della pressione dei tasti.
  - boolean onKeyDown (int keyCode, KeyEvent event)
    boolean onKeyUp(int keyCode, KeyEvent event)
    boolean onKeyLongPress (int keyCode, KeyEvent event)
    public boolean onKeyPreIme (int keyCode, KeyEvent event)
    boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event)
    public boolean onKeyShortcut (int keyCode, KeyEvent event)
    public boolean onKeyUp (int keyCode, KeyEvent event)

- gestione del tocco
  - public boolean onTouchEvent (MotionEvent event)

- gestione del focus
  - public void onWindowFocusChanged (boolean hasWindowFocus)

Gestione degli eventi fuori dalla View

- Activity
  - public boolean dispatchTouchEvent (MotionEvent ev)
    - intercettare tutti gli eventi di touch prima che vengano inviati alla finestra contenuta
- ViewGroup/Layout
  - public boolean onInterceptTouchEvent (MotionEvent ev)
    - permette di specificare se gli eventi di touch debbano essere elaborati nel metodo onTouch() di ognuna delle View
- Interfaccia ViewParent
  - public void requestDisallowInterceptTouchEvent (boolean disallowIntercept)
    - per impedire l'invocazione del precedente metodo onInterceptTouchEvent()

Attributo onClick

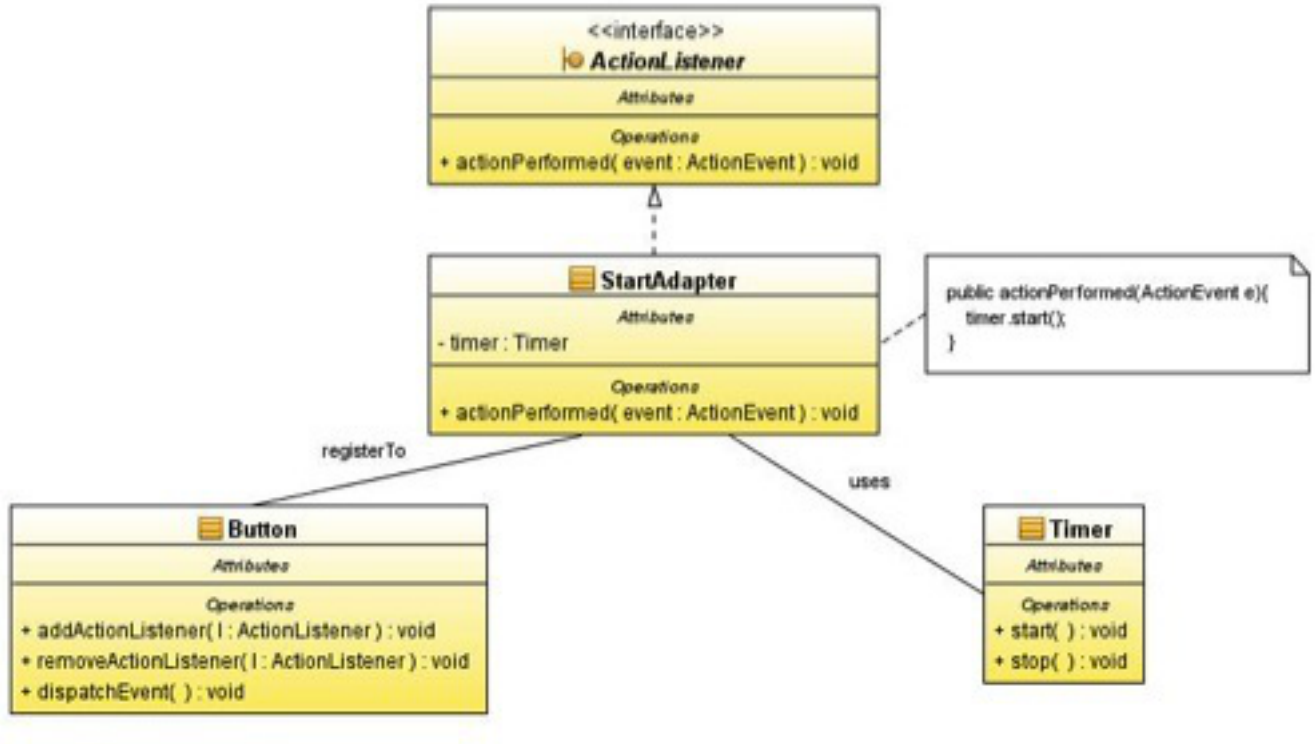
android:onClick="metodoGestioneEvento”

- Si può specificare il nome di un metodo da eseguire nella gestione dell’evento di click direttamente in xml
  - si può applicare a View
  - il metodo va inserito nel nel Context associato alla View
  - Il metodo deve avere la seguente firma
    - void <identificatore>(View v);

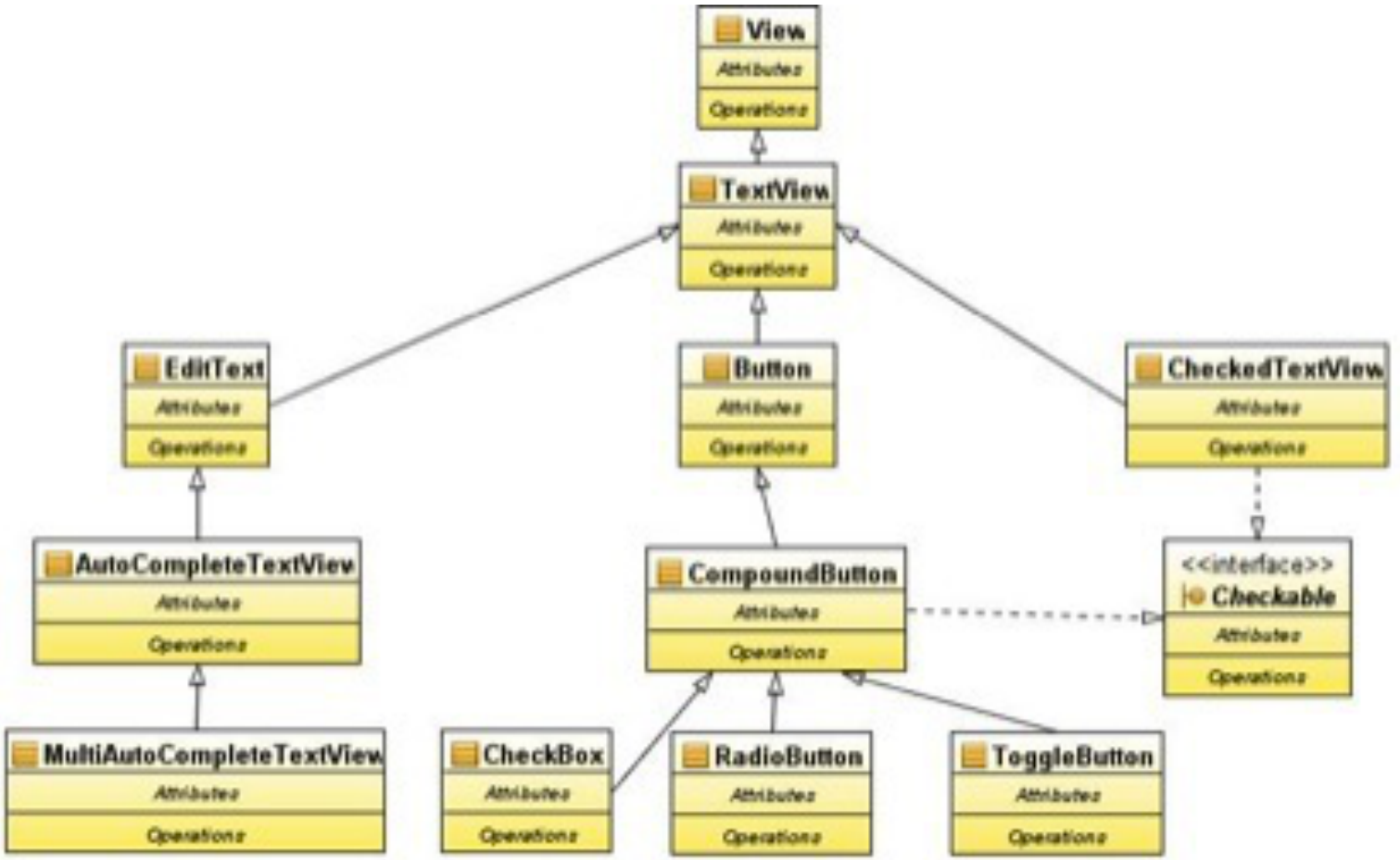
```
<Button
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:text="@string/una_label"
     android:onClick=”cliccami" />


public void cliccami(View view) {

}
```

Delegation Model

- Disaccoppiare il generatore dell'evento dal gestore



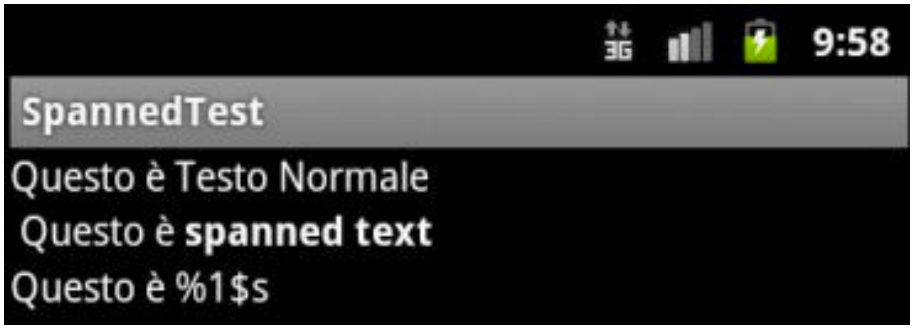
I widget


View
Attributes
Operations
TextView
Attributes
Operations
EditText
Attributes
Operations
Button
Attributes
Operations
CheckedTextView
Attributes
Operations
AutoCompleteTextView
Attributes
Operations
CompoundButton
Attributes
Operations
<<interface>>
Checkable
Attributes
Operations
MultiAutoCompleteTextView
Attributes
Operations
CheckBox
Attributes
Operations
RadioButton
Attributes
Operations
ToggleButton
Attributes
Operations

Textview

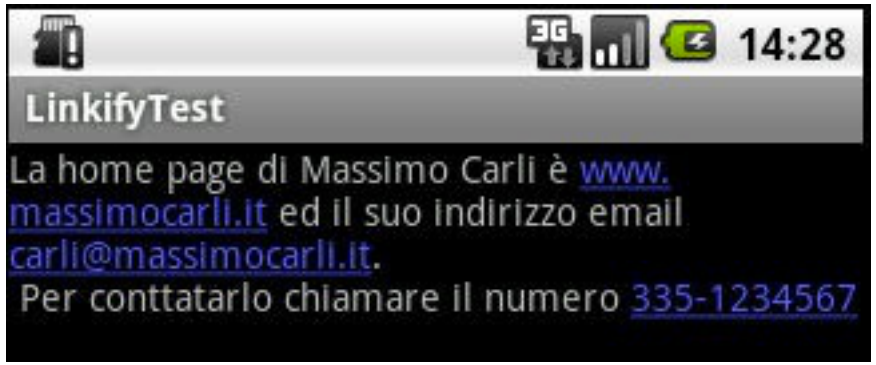
- Metodo principale: setText()
- Gestione dello stato automatica
  - Quando l'activity viene eliminata la Textview salva lo stato
- Transformation Method
  - permette di trasformare ciascun carattere del contenuto in un altro
  - servono a modificare modificare la CharSequence di una TextView
  - Interfaccia TransformationMethod
    - CharSequence getTransformation (CharSequence source, View view)
    - void onFocusChanged (View view, CharSequence sourceText, boolean focused, int direction, Rect previouslyFocusedRect)
- Per impostare un TransformationMethod in una TextView
  - public final void setTransformationMethod (TransformationMethod method)

Testo con tag



- **Spanned e Spannable**

```
<resources>
        <string name="normal_text">Questo è Testo Normale </string>
        <string name="spanned_text”> Questo è <b>spanned text </b></string>
        <string name="parsed_text">Questo è %1$s</string>
</resources>
```



- **Linkify**

```
<resources>
        <string name="linkify_test">La home page di Massimo Carli è
www.massimocarli.it e il suo indirizzo email carli@massimocarli.it.\n
Per contattarlo chiamare il numero 345-1234567</string>
</resources>
```

Ancora sul testo

- **Ellipsizing**
  - è possibile forzare la visualizzazione del testo su un'unica riga
    - public void setSingleLine ()
    - public void setSingleLine (boolean singleLine)
    - public void setHorizontallyScrolling (boolean whether)
  - se il testo è su più righe si può gestire il numero di righe
    - public void setLines (int lines)
  - è possibile introdurre i puntini di sospensione
    - public void setEllipsize (TextUtils.TruncateAt where)
- **Hint (Suggerimento)**
  - public final void setHint (CharSequence hint)
  - public final void setHint (int resid)
  - public final void setHintTextColor (ColorStateList colors)
  - public final void setHintTextColor (int color)
- **Typeface**
  - public void setTypeface (Typeface tf)

EditText

- Box per la modifica del testo
  - public Editable getText ()
  - public void setText (CharSequence text, TextView.BufferType type)
    - TextView.BufferType: EDITABLE, NORMAL e SPANNABLE
- Vedere esercitazioni

Specializzazioni di Button

- Button
  - CompoundBotton
    - CheckBox
    - RadioButton
    - ToggleButton

- La classe CompoundButton aggiunge al Button la possibilità di essere Checkable
  - il Button ha uno stato di selezionato (checked) o non selezionato (unchecked)
    - public void setChecked (boolean checked)
  - è possibile invertire lo stato con il metodo
    - public void toggle ()

CheckBox

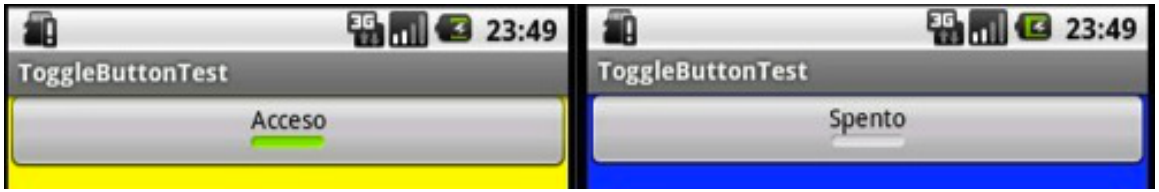

```
CompoundButton.OnCheckedChangeListener checkedListener =
new CompoundButton.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(CompoundButton button, boolean checked) {
                if (button == redBox) {
                        red = (redBox.isChecked()) ? getColorComponent(redText) : 0;
                } else if (button == greenBox) {
                        green = (greenBox.isChecked()) ? getColorComponent(greenText):0;
                } else if (button == blueBox) {
                        blue = (blueBox.isChecked()) ? getColorComponent(blueText): 0;
                } else {
                        Log.w(CHECK_BOX_TEST_TAG,"No checkBox selected. Check the code!");
                }
                container.setBackgroundColor(Color.argb(0x88, red, green, blue));
        }
};
```

Radio Button


```
RadioGroup.OnCheckedChangeListener listener = new RadioGroup.OnCheckedChangeListener() {

        @Override
        public      void onCheckedChanged(RadioGroup radiogroup, int checkedId) {
                    // A seconda del checkId cambiamo il colore di sfondo del container

                    if(checkedId == R.id.redRadio){
                                radioGroup.setBackgroundColor(Color.RED);
                    }else if(   checkedId == R.id.greenRadio){
                                radioGroup.setBackgroundColor(Color.GREEN);
                    }else if(   checkedId == R.id.blueRadio){
                                radioGroup.setBackgroundColor(Color.BLUE);
                    }else{
                                Log.w(RADIO_BUTTON_TEST_TAG, "Source not correct. Check code!");
                    }
        }

};
```

ToggleButton

```
toggle.setOnCheckedChangeListener(new OnCheckedChangeListener(){

                @Override
                public void onCheckedChanged(CompoundButton button, boolean checked) {
                        if(checked){
                                // Colore giallo
                                container.setBackgroundColor(Color.YELLOW);
                        }else{
                                // Colore blue
                                container.setBackgroundColor(Color.BLUE);
                        }
                }
});
```



ListView Checkable

- ListView che contengono View checkable
  - le View ottenute dagli Adapter associati dovranno avere la caratteristica di essere Checkable
  - si può cambiare la modalità di selezione delle View contenute
    - public void setChoiceMode (int choiceMode)
      - CHOICE_MODE_NONE, CHOICE_MODE_SINGLE e CHOICE_MODE_MULTIPLE
  - metodi di utilità della ListView
    - public long[] getCheckItemIds ()
    - public int getCheckedItemPosition ()
    - public SparseBooleanArray getCheckedItemPositions ()

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

- La classe CheckedTextView è la versione Checkable di una TextView