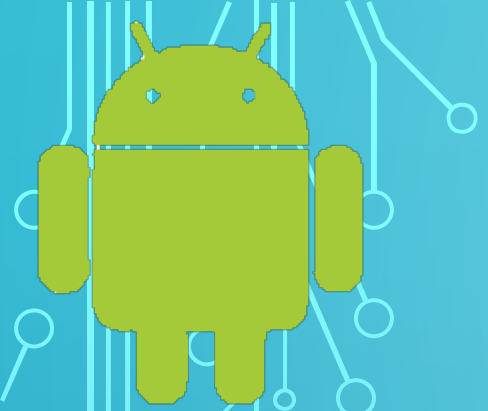


SZOFTVERFEJLESZTÉS ANDROID PLATFORMRA

DR. ISZÁLY GYÖRGY BARNA

FELHASZNÁLÓI FELÜLETEK

- Kialakításuk az egyik legfontosabb lépés az Android alkalmazásunk fejlesztése során.
- Vonzó, jól áttekinthető, jól kialakított felhasználóbarát
- A Google Play-en is először csak a felhasználói felülettel találkozunk
- Számos gyártó, számos készülékkel – eltérő hardveres lehetőségek
- A sok eltérő képernyőméret és felbontás ellenére mindegyik eszközön ugyanúgy jelenjen meg alkalmazásunk!!!!

- LEHETETLEN!!! VAGY MÉGIS?????

- Elegendő elkészítenünk a megfelelő felületeket és grafikákat a megfelelő méretekben, és a rendszer a felületi elemeket az adott eszköz kijelző tulajdonságai szerint képes skálázni.

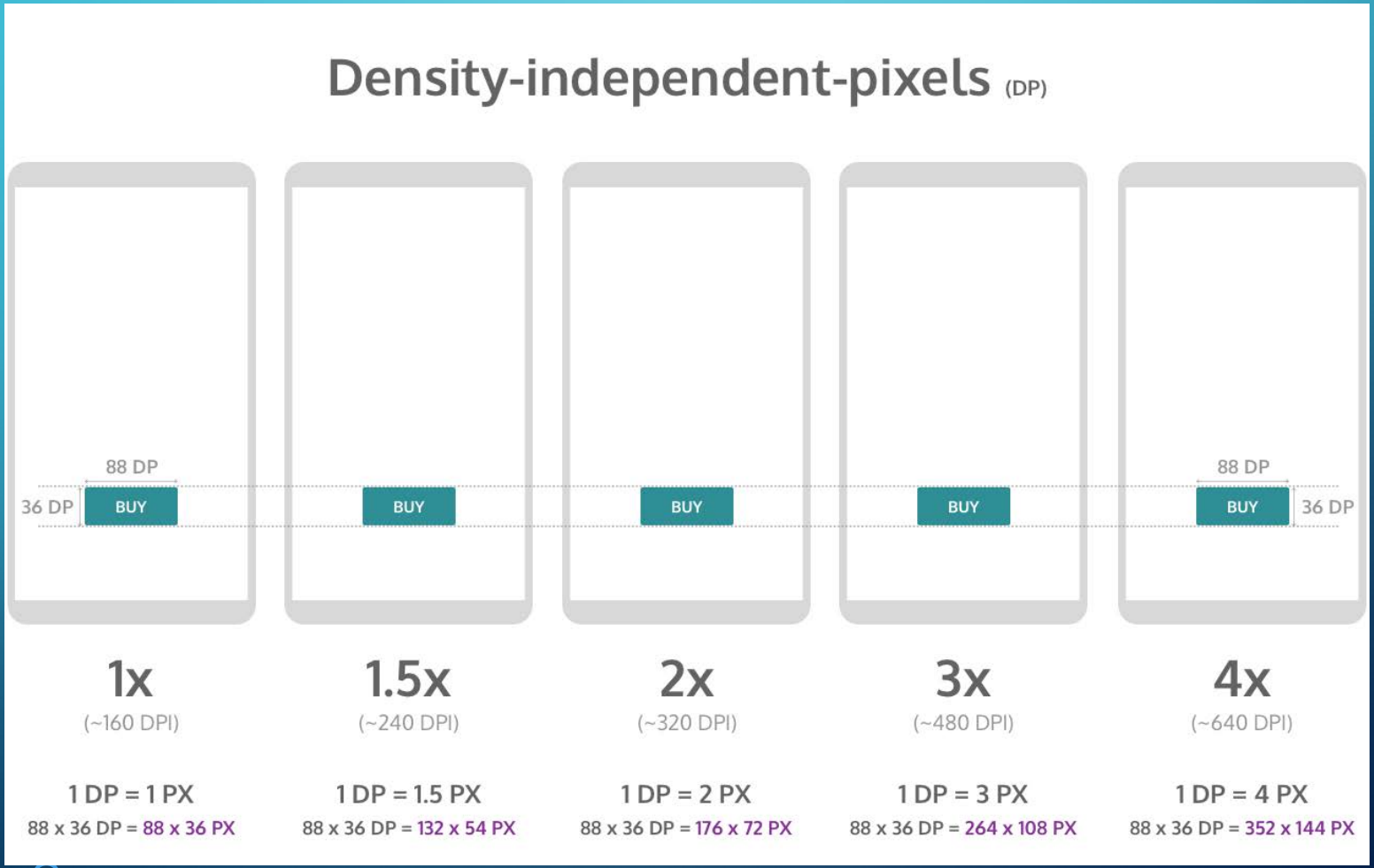
FELHASZNÁLÓI FELÜLETEK FONTOSABB FOGALMAK

- **Képernyő méret –** A tényleges fizikai mérete képátlóban megadva.
  - small,
  - normal,
  - large,
  - extra large
- **Képernyősűrűség –** Egy fizikai területen található pixelek sűrűsége dot per inch-ben (dpi)
  - low,
  - medium,
  - high,
  - extra high

FELHASZNÁLÓI FELÜLETEK FONTOSABB FOGALMAK

- **Orientáció** – Ez lehet álló (portrait), vagy fekvő (landscape). Az orientáció előírható fixen, vagy futási időben megváltoztatható.
- **Felbontás** – Az összes pixel száma a fizikai képernyőn. Az alkalmazások nem dolgoznak ezzel az értékkel, hanem helyette a képernyő méret és a képernyősűrűség értékeivel számolnak.
- **Sűrűségfüggetlen pixel (dp)** – virtuális pixel egység
  - Egy 160dpi-s képernyőn egy fizikai pixellel egyezik meg.
  - A rendszer futási időben gondoskodik a dp egységek mértezéséről az aktuális képsűrűség függvényében.
  - Az átváltás a dp egység és a pixel között: $px=dp*(dpi/160)$.
    Pl.: egy 200 dpi-s képernyőn 1 dp 1,25 fizikai pixelnek felel meg.

SŰRŰSÉGFÜGGETLEN PIXEL

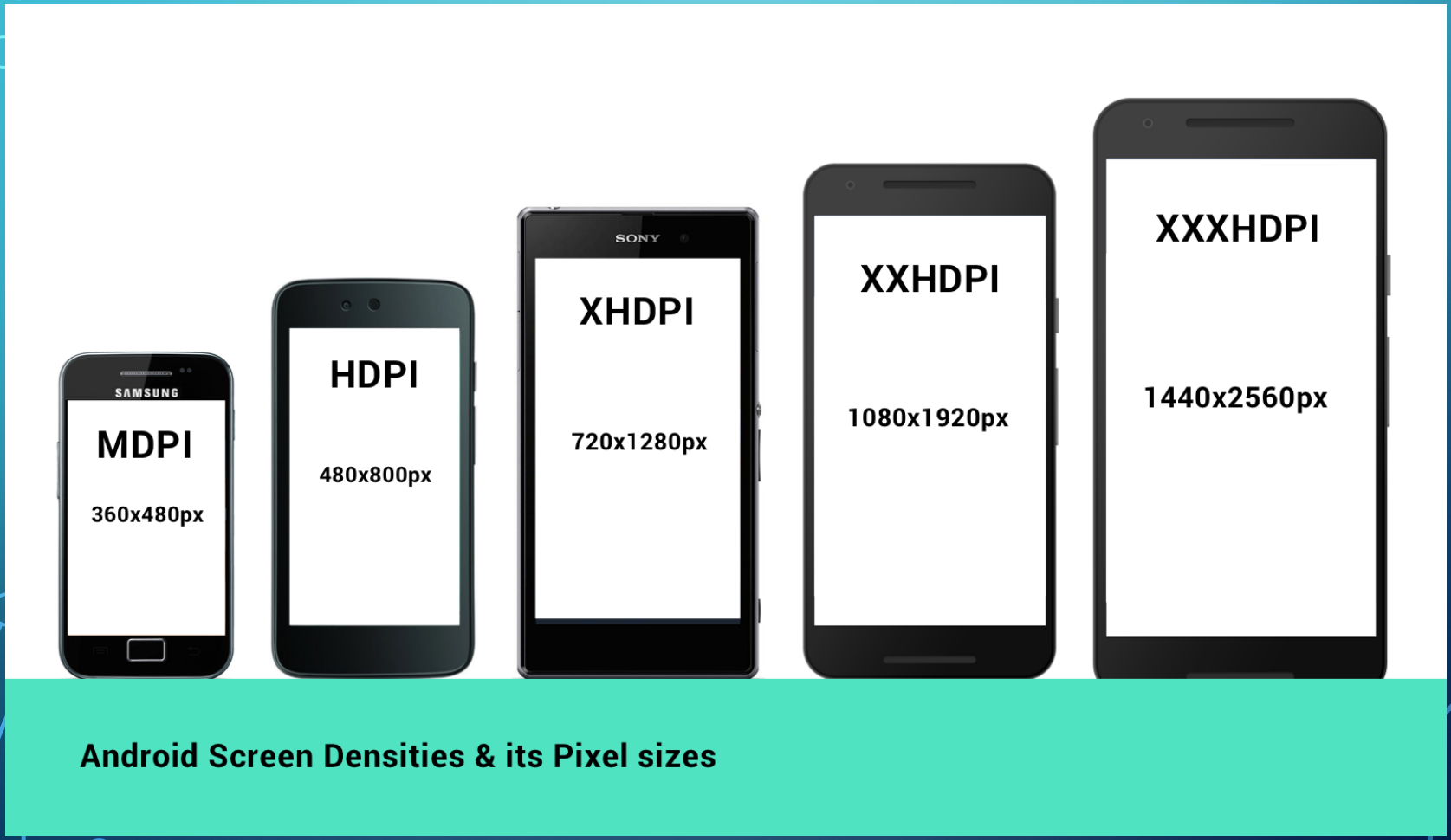


KÉPERNYŐMÉRET, KÉPERNYŐSŰRŰSÉG

- Minimális méretek:
  - xlarge : 960dp x 720dp
  - large: 640dp x 480dp
  - normal: 470dp x 320dp
  - small: 426dp 320dp

- A rendszer az aktuális képernyősűrűségnek megfelelően a dp kiszámítása alapján skálázza a felhasználói felületet
- A rendszer a képernyősűrűség felhasználásával automatikusan átméretezi a képerőforrásokat

KÉPERNYŐSŰRŰSÉG, PIXEL MÉRET



Android Screen Densities & its Pixel sizes

MÉRETEZÉS, MINŐSÍTŐK

- A grafikus erőforrások átméretezését nem célszerű a rendszerre hagyni
- Célszerű a grafikus elemeket is négyfajta méretben elkészíteni a 3:4:6:8 aránynak megfelelően
- A képek a res/drawable alkönyvtárba kerülhetnek
- Minősítőket használhatunk

  <erőforrás_azonosító> - <módosító>

  Pl.: drawable-small, drawable-small-mdpi

- Fontosabb minősítők:
  - Képernyőméret: small, normal, large, xlarge
  - Sűrűség: ldpi, mdpi, hdpi, xhdpi, nodpi (sűrűség független), tvdpi (megközelítőleg 213dpi)
  - Orientáció: land, port
  - Képarány: long, notlong

GRAFIKUS ELEMEK MEGJELENÍTÉSE

1. Az Android először az eszköz paramétereinek megfelelő módosítókkal ellátott erőforrásokat próbálja használni
2. Ha nem talál ilyet, akkor az alapértelmezett erőforrást használja, és az adott képernyő mérethez és sűrűséghez igazodva nagyítja, vagy kicsinyíti az adott elemet.
3. A legjobb megjelenés elérése érdekében a rendszer néha az alapértelmezett erőforrás helyett egy másik felbontás specifikus erőforrást használ.
   PL.: ha az alapértelmezett elemnél nagyobb felbontású elem létezik az alkalmazásban, akkor a rendszer előnyben fogja részesíteni a nagy felbontású képet, mert annak kicsinyítése sokkal szebb megjelenést tesz lehetővé.

KÉPERNYŐMÉRET ELŐÍRÁSOK A MANIFEST ÁLLOMÁNYBAN

- *android:requiresSmallestWidthDp* – Az alkalmazás futtatásához szükséges legkisebb szélességet írja elő.
- *android:compatibleWidthLimitDp* – Ha az itt megadott értéknél nagyobb a képernyő szélessége, akkor a rendszer felajánlja, hogy az alkalmazás kompatibilis megjelenési módban fusson. Ez a kompatibilitási mód kikapcsolható a rendszer menüből bármikor.
- *android:largestWidthLimitDp* – Az előzővel egyezik meg, azaz az alkalmazás itt is kompatibilis módban fog futni, azonban ebben az esetben ez a mód nem kapcsolható ki.

```
<manifest ... >
        <supports-screens android:requiresSmallestWidthDp="400" />
        ...
</manifest>
```

SUPPORT-SCREENS

```
<supports-screens android:resizeable=["true"| "false"]
                  android:smallScreens=["true" | "false"]
                  android:normalScreens=["true" | "false"]
                  android:largeScreens=["true" | "false"]
                  android:xlargeScreens=["true" | "false"]
                  android:anyDensity=["true" | "false"]
                  android:requiresSmallestWidthDp="integer"
                  android:compatibleWidthLimitDp="integer"
                  android:largestWidthLimitDp="integer"/>
```

JÓ TANÁCSOK GRAFIKUS ERŐFORRÁSOKHOZ

- Kerüljük a *pixel (px)* mértékegység használatát
- Használjunk a *density-independent pixelt (dp)*, a *wrap_content*, *fill_parent* jellemzőkkel.
- Szövegek esetében a *scale-independent pixel (sp)* mértékegységet használjuk
- Kerüljük az *AbsoluteLayout* használatát
- Használjunk képernyőméret és sűrűség specifikus erőforrásokat

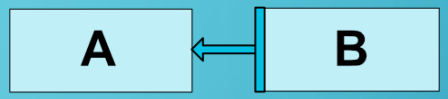
LAYOUT

- A felhasználói felület struktúráját definiálja
- Layout állomány gyökér eleme lehet
  - View – négyszögletes felület, amely rajzolásáért és a rajta bekövetkező eseményekért felelős
  - ViewGroup – A View osztály leszármazottja, és tartalmazhat további View osztálybeli elemeket. Felelős a közvetlen gyermekelemei kirajzolásáért és elhelyezéséért
- Az Activity a gyökérelemet utasítja a megjelenítésre, amely rekurzívan utasítja a gyermekelemeit ugyanerre

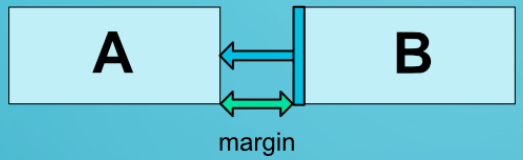
CONSTRAINTLAYOUT

- Rugalmas módon lehet az elemek méretét és pozícióját megadni
- Számos lehetőség van az elemek helyzetének megadására:
  - Relative positioning
  - Margins
  - Centering positioning
  - Circular positioning
  - Visibility behavior
  - Dimension constraints
  - Chains
  - Virtual Helpers objects
  - Optimizer

CONSTRAINTLAYOUT

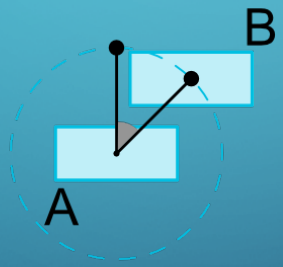


- Relative pisitioning
- Margins

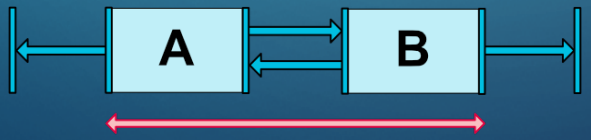


- Centering Positioning, Centering Positioning with Bias

- Circular pisitioning



- Chains



LINEARLAYOUT

- horizontálisan, vagy vertikálisan kerülnek egymás után elhelyezésre a View elemek *android:orientation* tulajdonság :
  - *horizontal*
  - *vertical*
- Az elemeket ezen belül igazíthatjuk az *android:gravity*
- *Az android:layout_weight* tulajdonsággal egy „fontossági” értéket adhatunk meg az egyes elemeknek.
  - a nagyobb súllyal rendelkező elem fogja kitölteni a szülő elem maradék részeit.
  - alapértelmezett értéke 0.

RELATIVELAYOUT

- Az elemeket elhelyezkedését az egymáshoz viszonyított helyzetükkel lehet meghatározni
- Pl.: egy elem egy másik alatt jelenjen meg, akkor használjuk a *android:layout_below* tulajdonságot, amely értékéül annak az elemnek az azonosítóját kell megadni, amelyik alá szeretnének az új elemünket elhelyezni.

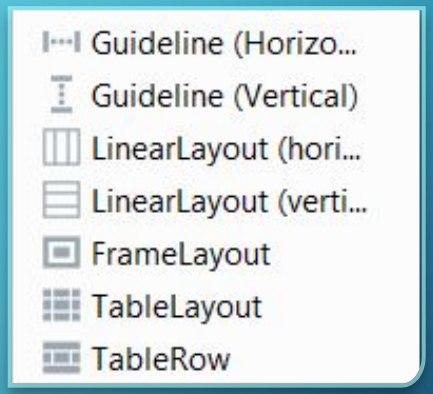
android:layout_alignRight="@+id/checkBox1"

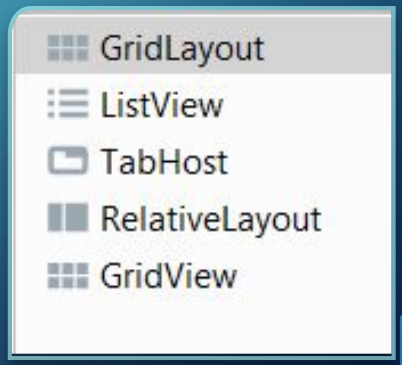
android:layout_below="@+id/checkBox1"

- Több, mint 20 különböző *RelativeLayout* paraméter áll rendelkezésünkre

LEGGYAKRABBAN HASZNÁLT LAYOUT-OK

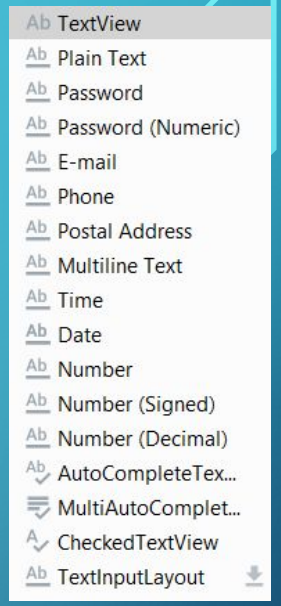
- AbsoluteLayout: az elemeket abszolút pozíciójuk alapján helyezhetjük el benne
- GridView: két dimenziós gördíthető elrendezést ad
- ListView: listanézetben jelennek meg az elemek
- MapView: térképnézeteket helyezhetünk el vele
- TableLayout: táblázatos elrendezést biztosít
- WebView: weboldalak megjelenítésére alkalmas
- RelativeLayout: az elemeket egymáshoz viszonyítottan lehet elhelyezni

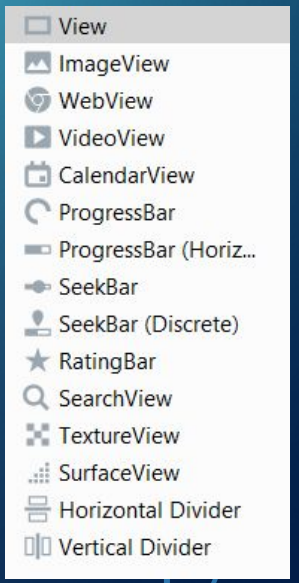


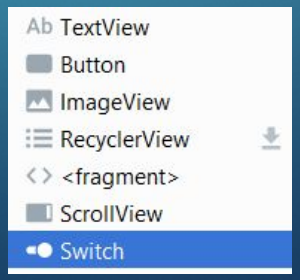


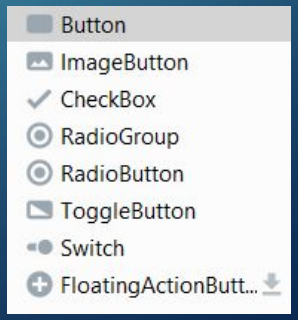
INTERAKTÍV VEZÉRLŐ ELEMEK

- Button (Button osztály) –A grafikus felhasználói felületeken már megszokott tulajdonsággal rendelkező nyomógomb.
- Checkbox (CheckBox osztály) – Kijelölő négyzet. lehet be, illetve kikapcsolt.
- Radio button (RadioButton és RadioGroup osztályok) – Az egy csoportba tartozó választási lehetőségek közül csak egy jelölhető ki.
- Spinner (Spinner osztály) – Egy legördülő lista elemeiből választhat ki egyet.
- Textfield (EditText osztály) – Szerkeszthető szöveges beviteli mező. Az AutoCompleteTextView osztály segítségével olyan beviteli mező készíthető, mely a begépelt betűk alapján egy listát ajánl fel a lehetséges értékekről.
- Toggle button (ToggleButton és Switch osztály) – Egy ki- és bekapcsolható gomb, mely a bekapcsolt állapotát az On feliraton kívül egy kis fénycsíkkal is jelzi. Az Android 4.0 verziótól rendelkezésünkre áll egy másik csúsztatható fajtája is.
- Pickers (DatePicker és TimePicker osztályok) –Valós dátumot és időpontot lehet beállítani a beállított nyelvi környezetnek megfelelően.









VEZÉRLŐ ELEMEK ALAPTULAJDONSÁGAI

- *android:layout_height* és *android:layout_width* – a ViewGroup osztály tulajdonságai, és az adott elem magasságát és szélességét állíthatjuk be velük. Értéke lehet valamilyen konkrét érték, vagy a követező konstansok egyike:
  - *fill_parent* – Az elem olyan nagy lesz, mint a szülő eleme. (Korábban *match_parent*).
  - *match_parent*
  - *wrap_content* – a tartalomhoz igazítja a méretet.
- *android:textSize* – A szöveg mérete állítható be vele, amit célszerű "sp", azaz scaled-pixels mértékegységben megadni.
- *android:textStyle* – A szöveg stílusa: normal, bold vagy italic. Ha egyszerre szeretnék félkövér és dőlt betűt, akkor a két konstanst a „|” (bitenkénti vagy) művelettel kell összekötni.
- *android:autoLink* – a szövegünkben előforduló url címek, vagy e-mail címek automatikusan átkonvertálódjanak-e klikkelhető linkké. Konstansai lehetnek a következők: none 0x00 Match no patterns (default).
  - web – webcímeket illeszt.
  - email - email címeket illeszt.
  - phone – telefonszámokat illeszt.
  - map – térkép címeket illeszt.
  - all – mindent illeszt. Megegyezik a web|email|phone|map megadással.

TULAJDONSÁGOK MÉRTÉKEGYSÉGEI

- dp (Density-independent Pixels) – sűrűség független pixel.
- sp (Scale-independent Pixels) – Hasonló, mint a dp, csak ezt a betűk méretének megadására használhatjuk.
- pt – nyomdászati pont, ami a méter 2660-ad része (1/72 inch).
- px – Pixel.
- mm – Milliméter.
- In – Hüvelyk.

- Az utolsó négy mértékegység használata kevésbé ajánlott
  - Ezek az aktuális képernyő fizikai tulajdonságaitól függnek
  - Nem garantálják a megfelelő megjelenést a különböző méretű és tulajdonságú kijelzőkön.

A VIEW OSZTÁLY ESEMÉNYFIGYELŐ INTERFÉSZEK

- Callback metódusok:
  - *onClick()* (View.OnClickListener interfész)– akkor hívódik meg, ha a felhasználó megérinti az adott elemet
  - onLongClick() (View.OnLongClickListener interfész) – akkor hívódik meg, ha a felhasználó megérinti az adott elemet, és továbbra is nyomva tartja azt.
  - *onFocusChange()* (View.OnFocusChangeListener interfész) – akkor hívódik meg, ha a felhasználó az adott elemre, vagy arról elnavigál.
  - *onKey()* (View.OnKeyListener interfész) - akkor hívódik meg, ha a fókusz az adott elemen van, és a felhasználó megnyom, vagy felenged egy hardveres gombot
  - *onTouch()* (View.OnTouchListener) – akkor hívódik meg, ha a felhasználó végrehajt egy érintési eseményt, ami lehet lenyomás, felengedés, vagy egyéb mozgatással járókézmozdulat

STÍLUSOK ÉS TÉMÁK

- Hasonló a CSS stíluslapokhoz
- Stílus – a létrehozott stílus a felhasználói felület egyetlen *View* elemére vonatkozik csak.
- Téma – az elkészített stílust egy *Activity*-re, vagy akár a teljes alkalmazásra vonatkozik.
- A *res/values* alkönyvtárba kell elhelyeznünk
- Az állomány gyökéreleme: <*resources*>
  - stílust definiáló elemek: <*style*> - kötelező attribútuma a *name*
  - Stílustulajdonságok: <*item*> - *kötelező name* tulajdonság. Ez tartalmazza a stílus tulajdonság értékét.
- Stílusok öröklődhetnek.
  - Az összes tulajdonság átöröklődik a szülő stílusból a gyerek stílusba.
  - Az öröklődés megadása a <*style*> elem *name* attribútumában történhet, a következő módon:
    - *szülőstílus_azonosítója.gyermekstílus_azonosítója*.

PÉLDA SAJÁT STÍLUSRA

styles.xml

```
<resources>
…
<style name="piroska"
parent="TextAppearance.AppCompat">
        <item
name="android:textColor">#FF0000</item>
    </style>
</resources>
```

activity_main.xml

```
…
 <TextView
        style="@style/piroska"

android:id="@+id/TextView_koszontes"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="Ki vagy?"
        android:textSize="24sp" />
```

STÍLUS BEÁLLÍTÁSA

- A felhasználandó témát az *AndroidManifest.xml* állományban állíthatjuk be
  - *android:theme* tulajdonságnál megadva a téma azonosítóját.
  - Ha az *android:theme* tulajdonság az <*activity*> elemen belül szerepel, akkor az adott Activity-re fog vonatkozni a téma
  - Ha az <*application*> elemen belül szerepel, akkor a teljes alkalmazásra.
- Forráskódból is beállítható:
  - *setTheme()* metódust kell meghívnunk az *onCreate()* metódusban
  - a hívás még a felhasználói felület elemeinek létrehozása előtt kell megtenni.

LOKALIZÁCIÓ

- Több nyelven is elérhetővé érdemes tennünk az alkalmazásunkat
- A minősítők használatával oldható meg.
- Az alkalmazásunkban található szöveges elemet alapértelmezetten a *res/values/strings.xml* állományba kell elhelyezni.
- Az Android azonban lehetővé teszi, hogy ezen felül a *values* könyvtárhoz minősítőket tegyünk, és így ott az adott nyelvnek megfelelő változatban hozzuk létre a *strings.xml* állomány tartalmát.
  - angol verzió esetén – hozzuk létre a *values-en* alkönyvtárt
  - magyar verzió esetén – hozzuk létre a *values-hu* alkönyvtárt
  - ezekbe tároljuk a nyelv specifikus *strings.xml* erőforrás állományt.
- A values alkönyvtárban lévő *strings.xml* állomány az alapértelmezett
- Ennek hiánya hibát okozhat, ha olyan nyelven szeretné megnézni valaki az alkalmazásunkat, melyhez nem hoztunk létre nyelv specifikus részt.