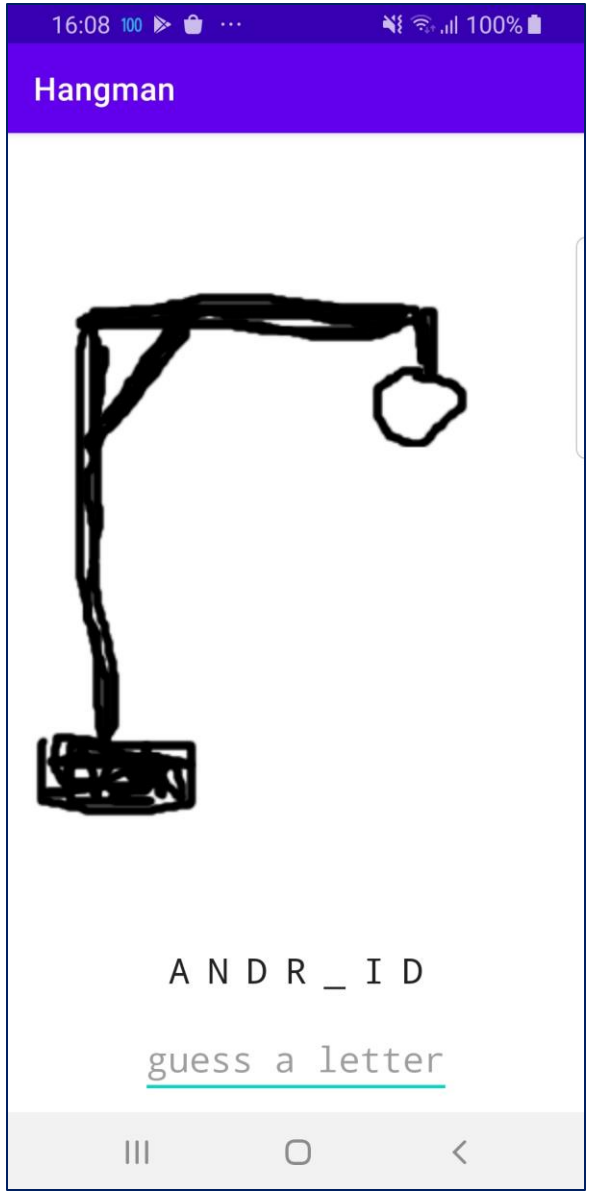
Hangman

Aplikácia, ktorá pomôže porozumieť základom životného cyklu aktivity, tvorbe layoutu a ďalším základným činnostiam.



#1 Layout a resources

Preskúmať predpripravený kód:

- Obrázky šibenice v **`res/drawable`**
- Interface hernej logiky v **`Game.java`**
- Layout s troma widgetmi (obrázok, hádané slovo a zadávané písmeno)

- Vyrobiť zoznam slov (**`string-array`**), ktoré sa budú hádať v **`res/values/strings.xml`**

String ako resource

- /res/values/strings.xml
- **Podpora viacjazyčnosti**
- **String array** - skúsme si zoznam slov na hádanie
- Ďalšie možnosti pri prekladoch do iných jazykov
  - **Quantity strings** (zero, one, two, few, many, and other.)
  - Zvýrazňovanie textu
  - Formátovanie, špeciálne znaky

- `getResources().getStringArray(R.array.name);`
  - Túto metódu má aktivita

ConstraintLayout

- Musí byť pridaná závislosť (`build.gradle`)
- Kľúčový princíp - **constraints**
- Veľkosť widgetu
  - Fixed
  - Wrap Content
  - Match Parent
  - Match Constraints (`layout_width="0dp"`...)

Android Jetpack

Jetpack is a **suite of libraries** to help developers follow best practices, reduce boilerplate code, and write code that works consistently across Android versions and devices so that developers can focus on the code they care about.

- dependency - `build.gradle (Module:app)`
- **ConstraintLayout** a mnohé iné

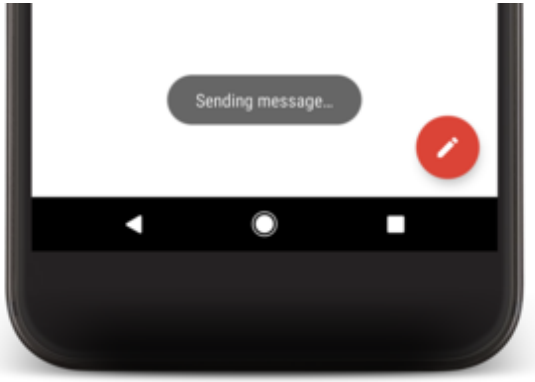
#2 Logika hry

Po kliknutí na šibenicu (v aktivite):

1. zistiť zadané písmenko
2. ošetriť situáciu, že nebolo zadané žiadne písmeno alebo nejaký nevhodný znak (ten môžeme priamo aj zmazať)
3. prispôsobenie písmena (aby bolo jedno, či sú zadávané písmena a uložené slová tvorené veľkými alebo malými písmenami),
4. zavolať metódu `guess`
5. vyčistiť `EditText`
6. podľa odpovede z `guess` aktualizovať slovo (`textView.setText`) alebo šibenicu (`imageView.setImageResource`)

+ **Implementovať interface**

Toast



- ostáva viditeľné aj keď aktivita skončí
- `Toast.makeText(context, text, duration)`
  `.show();`
- **context** = aktivita (nie vždy, ale pre nás zatiaľ áno)
- **duration** = `Toast.LENGTH_SHORT` (alebo `_LONG`)
- setGravity, vlastný view...
- alternatíva – *snackbar*

#3 Doplnenie funkcionality

- Opakovanie hry - pomocná metóda `restartGame` zavolaná v `onCreate` a po skončení hry
- Farebné filtre pomocou `LightingColorFilter`

Čo sa stane po otočení zariadenia?

Životný cyklus aktivity

onCreate()

onStart()

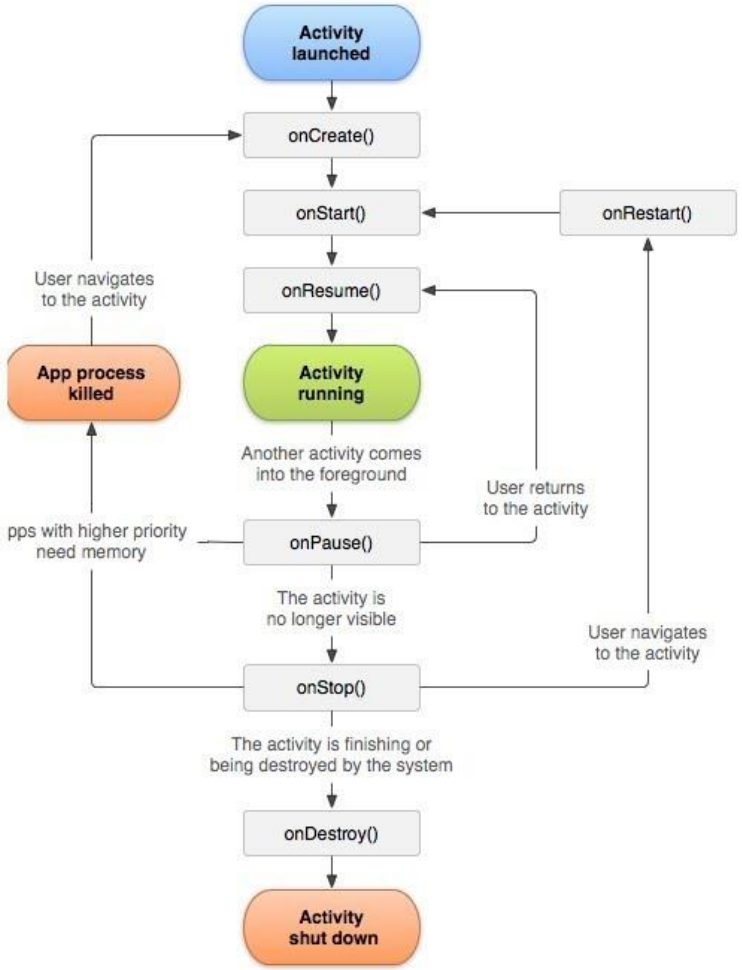
onRestart()

onResume()

onPause()

onStop()

onDestroy()



Životný cyklus aktivity - stav

- **viditeľná** - proces v popredí, má *focus*
- **čiastočne viditeľná** - proces v pozadí, ak aktivita stratila *focus* napr. pri prekrytí lišty s prichádzajúcim hovorom alebo pri multiview zobrazení (focus má iná aktivita, ale daná pozastavená aktivita je stále viditeľná). Po zavolaní metódy `onPause()`
- **skrytá** - proces v pozadí, nie je viditeľná. Po zavolaní metódy `onStop()`
- **zničená** - proces je prázdny. Po zavolaní metódy `onDestroyed()`

Bundle

- reštart aktivity - otočenie obrazovky a iné
- mapa, key=`String`, value=`Parcelable`
- **save** - `onSaveInstanceState()`
  - parameter Bundle
  - volá sa pred zaniknutím aktivity – pred `onStop()`
  - špeciálne prípady v dokumentácii (prekrytie inou akt.)
- **load** – `onCreate()`
  - cez parameter Bundle (môže byť **null**)

Rozhrania na zabalenie objektov

- **Serializable**
  - jednoduchá implementácia (žiadne metódy)
  - object -> byte stream
  - všetky inštančné premenné musia byť Serializable
- **Parcelable**
  - rýchlejšie ako Serializable (nevytvára temp objekty)
  - object -> byte stream
  - vyžaduje implementáciu

#4 Meranie času, uloženie a alert

- Vypočítať čas hrania (bez upravovania rozhrania)
- Pri lepšom čase aktualizovať doterajší rekord – Kde bude uložený?
- AlertDialog - informovať používateľa

Ako merať čas?

- **System.currentTimeMillis**
  - čas od 1.1.1970
  - nastaviteľný používateľom/sieťou – nevhodný na meranie uplynutého času
- **SystemClock.uptimeMillis** (System.nanoTime)
  - čas od bootovania systému
- **SystemClock.elapsedRealtime**
  - rovnako ako uptimeMillis + deep sleep

Kde uložiť dáta?

- súbor vs. databáza

- **app-specific** - internal/external storage
- **shared storage** – prístup aj z iných aplikácii
- **preferences** – key-value
- **database** - Android Room (súčasť Jetpack), SQLite lokálna databáza

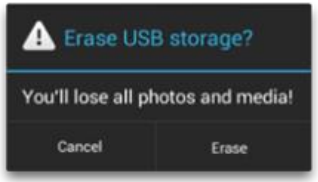
Shared Preferences

- mapa – kľúč & hodnota
- `getPreferences()` – ak stačí 1 súbor pre aktivitu
  - `getSharedPreferences()` – súbory s menom

- READ – getX() + default hodnota
- WRITE – Editor + putX() + commit/apply
  - editor zapisuje po dávkach
  - **apply** - v pamäti ihneď, na disku asynchrónne
  - **commit** – synchrónny zápis (nevhodné do UI vlákna)

Dialóg

- **`AlertDialog.Builder`**
  - Stratégia (využívaná v Androide častejšie) - Builder vytvorí dialóg podľa požiadaviek, s ním sa potom pracuje (=zobrazí sa)
- Možné pridať tlačidlo na zatvorenie, resp. prekryť metódu **`onCancel()`**
- v dokumentácii je spojené s fragmentmi – budeme o tom hovoriť neskôr



Easter eggs

- Kliknite viackrát na verziu Androidu v nastaveniach telefónu

Zaujímavé premenné/metódy v triedach:

- Log
- ActivityManager
- UserManager
- SensorManager