Zkouška PAA – otázky ke zkoušce

1. **Platforma Android, verze, architektura, nástroje pro vývoj**
   - Platforma Android poskytuje nejen OS, ale také GUI, specifikace ovladačů a SDK
   - Verze Android – první Cupcake (1.5), poslední Oreo (8.0), od verze 6 lepší práce s právy aplikací.
   - Architektura má 5 vrstev – **Linux kernel (jádro OS):** obstarává správu paměti, ovladače zařízení, power manegement, správa procesů; **knihovny v C++:** pro komponenty systému, poskytují funkce pro Android App Framework (systémová knihovna libc, media framework, sqlite,FreeType rendering fontů, WebKit, SSL, OpenGL,…); **Android Runtime (aplikační VM):** původně Dalvik VM, .java->.class->Dalvik code->DVM, od 4.4 Android RunTime ART, používá dopřednou kompilaci, rychlejší aplikace, delší výdrž baterie; **App Framework – aplikační vrstva:** nejdůležitější pro vývojáře, služby a funkce knihoven a nižších vrstev, např.: UI aplikací, Content providers – sdílení dat, Resource manager – lokalizace, gui, design, Notification manager – zasílání notifikací, Activity manager – životní cyklus aplikací; **aplikace:** přeinstalováno systémem, poskytovatelem, Google play, 3. Strana, vlastní vývoj
   - Nástroje pro vývoj – tvoří Android SDK pro jednotlivé verze, SDK Tools – debugging, android virtual device, android emulátor, usb drivery, doplňky 3. Stran, IDE – Android studio, donedávna Eclipse, od 5/2017 Kotlin pro vývoj
2. **Základní komponenty aplikace, manifest, oprávnění**
   - Komponenty aplikace: Activity, Fragments, Views, Service, Content Provider, Intents, Broadcast Receiver, Widgets, Resources (layout, menu,…), AndroidManifest.xml
   - Manifest – informace o package a verzi aplikace; název, ikona, téma aplikace; komponenty aplikace (aktivity, service); práva; min a max API level; použité knihovny a další konfigurace
   - Oprávnění – uvedená v manifestu, do verze 6 schvalování práv při instalaci aplikace
3. **Aktivita, životní cyklus, důležité metody k překrytí**
   - Aktivita – základní komponenta pro zobrazení aplikace, bez aktivity nelze aplikaci spustit, aplikace může mít více aktivit, vždy potomkem android.app.Activity
   - Životní cyklus – řídí ActivityManager, 4 fáze životního cyklu: **spuštěna** – spuštění app; **běží** – spuštěna a v popředí; **v pozadí** – je vidět, ale překryta jinou aplikací (příchod notifikace, hovor); **zastavená** – není vidět, bez přístupu, není ukončena (stisknutí home); **ukončená** – úplné ukončení aktivity
   - Metody k překrytí – onCreate(), onPause(), onResume(), onStop(), onDestroy()
4. **Android resources - co to je, význam jednotlivých resources, modifikátory, jednotky, displeje atd.**
   - Díky oddělení designu, zdrojů od aplikační logiky je možné tvorbu aplikací pro různá rozlišení, jazykové verze atp. Kořen v /res, dále pak dělené na jednotlivé kategorie, *.xml, přístup přes ID.
   - Jednotlivé kategorie – **drawable/** (obrázky), **layout/** (rozložení aktivit), **menu/** (definice jednotlivých menu), **raw/** (libovolná data nezapadající do uvedených kategorií), **values/** (definice textů, stylů, polí, barev, atd.), **xml/** (další xml soubory pro aplikaci)
   - Modifikátory – aplikace pro různá rozlišení, lokalizace; slouží pro rozšíření názvů složek, priorita nejspecifičtější->nejobecnější; priorita jednotlivých modifikátorů (region->šířka->orientace displeje->hustota pixelů->api lvl)
   - Jednotky – displej v palcích; rozlišení v pixelech; pixel density (dpi, ppi) -> nezávislá jednotka dp (density-independent pixel); 1dp = 160px/dpi
   - Displeje – dle velikosti; small, normal, large, xlarge
   - Hustota pixelů dpi (3:4:6:8:12:16) – ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi

5. **UI - View, ViewGroup, zachyvcení události UI**
   - View – třída, rodičem všech UI prvků, reprezentuje jeden konečný prvek UI (metody a atributy)
   - ViewGroup – zapouzdřuje další View a ViewGroup; stará se o pořadí, rozmístění a vykreslení prvků UI
   - Zachycení události – překrytí specializované metody zděděné z Activity, listener (setOnXXXListener), vlastní metody + deklarace události v xml layoutu.
6. **UI - Layouty, co to je, navázání na resources, typy layoutů**
   - Layout – potomek ViewGroup, rozložení a vzhled obrazovky
   - Navázání na resources – R.layout.id, res/layout
   - Typy layoutů – **AbsoluteLayout** (absolutní souřadnice, k ničemu), **FrameLayout** (nejjednodušší layout, prvky umisťovány na sebe, zarovnání pomocí gravity), **LinearLayout** (prvky vedle/pod sebe, rozložení pomocí gravity a orentation), **RelativeLayout** (prvky relativně vůči ostatním, A nad C, C vedle B,…), **TableLayout, GridView** (možno rolovat)
7. **Intenty - co to je, typy intentů, použití s jednotlivými komponenty aplikace**
   - Intent je základní asynchronní komunikační nástroj mezi prvky aplikací. Je to třída obsahující popis a data nějakého záměru. Objekt s definicí cílového procesu s možností zaslat mu data. Zpravidla obsahuje: **Action** (záměr, který je třeba vyvolat), **Category** (BROWSABLE, LAUNCHER, HOME), **Extras** (key/value páry hodnot předaných intentu)
   - Typy indentů – explicitní (má informaci o konkrétní třídě, kterou chce spustit), implicitní (má pouze info o záměru (chci psát email) a případná data k předání. Nechá na systému, kterou aplikaci spustí nebo nabídne.)
   - Použití s komponentami – intent filter – informace o tom, na jaký intent umí komponenta reagovat
8. **UI - Fragmenty, co to je, použití, asociace s Activity**
   - Fragment je jakási podaktivita, rozdělí jednu aktivitu na různé části, které obsahují různé informace. Např. zobrazení detailu položky vybrané v ListView.
   - Slouží pro optimalizaci UI na různých displejích (telefon/tablet).
   - Rodičem Fragment, nikoliv Activity. Mírně odlišné metody životního cyklu. Instance fragmentu vytvářena přímo, bez indentu. Předání bundle se provádí jinou metodou.
   - Asociace 2 způsoby. Přidáním <fragment> do layoutu aktivity, nebo programově pomocí FragmentManageru (potvrzování transakcí,…).
9. **Persistentní ukládání dat, SharedPreference**
   - Slouží pro ukládání primitivních dat typu key/value. Původně určeno pro ukládání nastavení. Pouze pro typy boolean, float, int, long, String. Data jsou přístupná pro všechny komponenty aplikace. Získání instance pomocí PreferenceManager, Context nebo v rámci aktivity pomocí getPreference. K editaci a uložení se používá SharedPreferences.Editor, instance editoru metoda edit(), uložení pomocí commit(). Získávání dat pomocí příslušných getterů, má dva parametry – název klíče a výchozí hodnotu.
10. **Persistentní ukládání dat, práce se soubory**
    - Ukládání soukromých souborů pro aplikaci. Využívá interní paměť zařízení, vhodná pro malé soubory. Metody openFileInput/openFileOutput. Možnost použití soukromé souborové cache. Po skončení aplikace může být obsah cache vymazán, pokud dochází místo v interní paměti. Reference na složku chache pomocí getCacheDir().
    - Ukládání na veřejné úložiště v zařízení nebo na SD kartu. Vyžaduje příslušná oprávnění (W/R_external_storage). getExternalFilesDir. Obdobně možnost souborové cache

11. **Persistentní ukládání dat, SQLite**
    - Slouží k ukládání strukturovaných dat v soukromé databázi. Kompletní databáze je uložena v 1 souboru (*.db). Transakce jsou atomické, trvanlivé a konzistentní i po pádu systému. Podporuje většinu standardů SQL92. V Androidu se k SQLite DB nepřistupuje přímo, ale pomocí specializovaných tříd. Datové typy v třídě Cursor.
    - Třída SQLiteOpenHelper je rodič pro konkrétní DB. Potomek obsahuje (jméno DB souboru, verzi DB, definici struktury DB, překrytí metody onCreate(), onUpgrade(). Díky rodiči poskytuje metody pro získání instance (getReadableDatabase(), getWriteableDatabase()).
    - Pro přístup je nutné vytvořit instanci potomka SQLiteOpenHelper, získat DB pomocí helper.getWritableDatabase() a pak uzavřít db.close().
    - Vkládání pomocí metody insert; k vyhledávání slouží metoda query, která vrací Cursor, který je na konci úkonů potřeba uzavřít;…
    - Možnost vkládání sql dotazů (rawQuery) nebo pomocí query builderu.
12. **Práce na pozadí, Handlery**
    - Aplikace musí na akci odpovědět do 5 (10) sekund od zahájení požadavku, jinak OS nabídne ukončení aplikace. Každá aplikace obsahuje hlavní vlákno (UI thread) ve kterém běží všechny UI procesy (aktivity), services a broadcast receivers.
    - Handler umožňuje asynchronní vykonání operace + promítnutí změn do UI. Funguje na principu zasílání a příjmu zpráv. V hlavním vlákně je vytvořena instance třídy Handler. Proces zaslání zprávy v jiném vlákně.
    - Zpráva zaslaná pomocí handleru je datového typu Message. Získání pomocí new nebo obtainMessage. Obsahuje vlastnosti (arg.. – pouze int)
13. **Práce na pozadí, vlákna**
    - Android poskytuje standardní Java mechanismy pro vytváření, běh a správu vláken (extends Thread, implements Runnable). Lze využít ThreadPool, Executor. Nikdy nesmí zasahovat do UI, včetně toast.
14. **Práce na pozadí, AsyncTask**
    - Zjednodušuje volání metod v hlavní vlákně. Kombinace metod, z nichž jedna běží ve vedlejším vlákně a průběžné výsledky se spouští v hlavním.
    - 4 metody onPreExecute, doInBackground, onProgressUpdate
15. **Výměna dat, XML**
    - Výměna dat není omezena pouze na SQLite. Data v textově čitelné podobě. Rozhraní XmlPullParser. Pro vytváření XML dat slouží XMLSerializer
16. **Výměna dat, JSON**
    - Zapouzdřuje objekt, obecně dvojice klíč/hodnota. Klíčem je string. Pro jednotlivé datové typy příslušné settery a gettery. Užitečné metody(string key), keys(), names() – JSONArray klíčů,…
    - JSONArray reprezentuje indexované JSON pole.
    - Nápomocné třídy JsonWriter a JsonReader pro usnadnění parsování a tvorby JSON dokumentu.
17. **Content Provider / Resolver, UriMatcher**
    - Prostředek pro poskytování dat. Data uložena v SQLite DB, souborech, na webu. Jediný způsob, jak sdílet data napříč aplikacemi. Vyjímkou je Mode a IPC. Možné využívat i pro přístup k datům z vlastní aplikace. Každá aplikace může poskytovat svoje data přes Content Provider definované pomocí URI. ContentResolver slouží pro přístup k datům poskytovaným Content Providerem.
    - Kontakty, SMS, zmeškané hovory, multimedia library
    - Pokud Provider poskytuje více typů dat, pro každý typ je nutné vytvořit proměnnou a nastavit UriMatcher.

18. **Content Provider / Resolver, integrace s SQLite, Cursor**
    - Query vrací Cursor s výsledky dotazu. Analogicky se chovají ostatní metody (insert, update, delete). Cursor lze získat použitím SQLiteQueryBuilderu.
    - Content Resolver se chová podobně jako DB.
19. **Service - co to je, rozdíl oproti ostatním komponentám aplikace, životní cyklus, integrace do manifestu**
    - Služby se v principu podobají aktivitám, akorát bez UI. Určeny hlavně pro dlouhou kontinuální činnost. Přehrávání hudby, dlouhé stahování dat, atd. Nevadí jim přesunutí do pozadí a práce s jinou aplikací. Vlastní životní cyklus, nezávislý na aktivitě.
    - Jsou méně náchylné k násilnému ukončení systémem než aktivity na pozadí.
    - Životní cklus – onCreate(), onStartCommand() (metoda startService()), onBind() (bindService()), onDestroy()
20. **Service - důležité metody, spouštění, návratové hodnoty, flags, provázání s Activity**
    - onStartCommand – vrací konstanty, které určují, jak se bude služba chovat, když jí systém ukončí při nedostatku prostředků. START_STICKY – výchozí chování, START_NON_STICKY – restart pouze když čekají další požadavky, START_REDELIVER_INTENT – podobné START_NON_STICKY.
    - Na rozdíl od aktivity má daná služba maximálně jednu instanci.
    - Flags – slouží pro zjištění jak se service spustila. START_FLAG_REDELIVERY – znovu doručení, START_FLAG_RETRY – restart systémem.
    - Provázání pomocí startService(), parametr Intent s Extras, asynchronní volání, služba běží, dokud není zastavena; bindService()
21. **Notifikace - co to je, vytvoření, PendingIntent, priority**
    - Notifikace slouží k upozornění uživatele na nějakou událost bez použití aktivity. Indikují také probíhající služby.
    - Vytváření pomocí NotificationManager. Ten umožňuje nejen vytváření, ale i úpravu či odstranění i nepotřebných.
    - Pending intent – zapouzdří intent, který pak může být použit systémem nebo jinými aplikacemi.
    - Priority – 5 různých – Foreground process, Visible process, Service process, Background process, Empty process.
22. **Broadcast - co to je, vazba na manifest, princip činnosti**
    - Intent může sloužit k zasílání anonymních zpráv mezi komponentami (reakce na změny v systému, naprosto oddělený mechanismus). Při odeslání broadcastu se pošle daný intent všem BroadcastReceiverům zaregistrovaným na danou akci v intentu.
    - Potřeba implementovat pouze metodu onReceive(), není vázaný na UI.
    - Vazba na manifest staticky (tag receiver) nebo pomocí registerReceiver.
    - Odeslání pomocí Context.sendBroadcast – kompletně asynchronní, není možné ho zrušit; Context.sendOrderedBraodcast – obdrží maximálně jeden BroadcastReceiver, pořadí určené pomocí priority v intent filteru
23. **BroadcastReceiver - oč jde, jak se předávají / zachycují data, oprávnění**
    - Startují se automaticky obržením broadcastu. Metoda onReceive musí skončit do deseti sekund. Pro delší operace je vhodnější služba. Nedědí z kontextu, je předán metodě onReceive.
    - Oprávnění slouží jako jakýsi filtr – daný broadcast přijmou pouze receivery, které mají daná oprávnění definovaná v manifestu