Dokumentácia k semestrálnej práci

z predmetu vývoj aplikácií pre mobilné zariadenia

Vypracoval: Andrej Michalek

Téma semestrálnej práce

Témou mojej semestrálnej práce je Android aplikácia, vďaka ktorej si bude užívateľ vedieť zahrať kartovú hru sedma proti mobilu.

Spracovanie aplikácií podobného zamerania

V Obchode Play sa aktuálne nachádzajú dve aplikácie s podobným zameraním, pričom obe z nich majú rovnaký názov – Sedma.

Aplikácia sedma od vydavateľa Honzales vyzerá z pohľadu grafického užívateľského rozhrania nasledovne:

Karty na ruke sa nachádzajú v dolnej časti obrazovky a pri pridaní kariet na kopu je potrebné kartu pomocou drag and drop chytiť a premiestniť na kopu. Hra v natívnom nastavení neustále zobrazuje rady ako napríklad „Desiatky a esá majú hodnotu 10 bodov a sedma berie štich“, a to aj po niekoľkých kolách. Zároveň som si všimol, že bublinky s radami sa prekrývajú s niektorými tlačidlami. Karty v kole hra pridáva nad seba, takže po tom, ako každý hráč vyloží niekoľko kariet už nemusí byť zrejmé, aká karta bola vložená prvá a koľko obsahuje aktuálne kolo es a desiatok, a teda aj o koľko bodov sa v danom kole hrá.

Hra od vydavateľa Nelu Cimpean funguje podobným spôsobom, ale na rozdiel od prvej už nie je potrebné karty ťahať na hraciu plochu, ale reagujú aj na kliknutie. Dizajn hry je prepracovanejší, ale nenesie sa v štýle Material dizajnu, ktorý Google odporúča pre svoje aplikácie. Takisto aj v tomto prípade chýba informácia o tom, koľko kariet sa ešte nachádza na kope, podľa ktorej by sa dalo odhadnúť, koľko kôl zostáva do konca hry. Aplikácia zároveň obsahuje reklamy, a to v menu aj v aktivite hry, čo môže na hráča pôsobiť rušivým dojmom. V hre sa nachádza zvláštny bug, kedy nedopĺňa po každom kole do ruky 4 karty ako je v sedme zvykom, ale na ruke necháva 3 karty. Po dohratí sa zobrazia karty, ktoré užívateľ v danej hre získal.

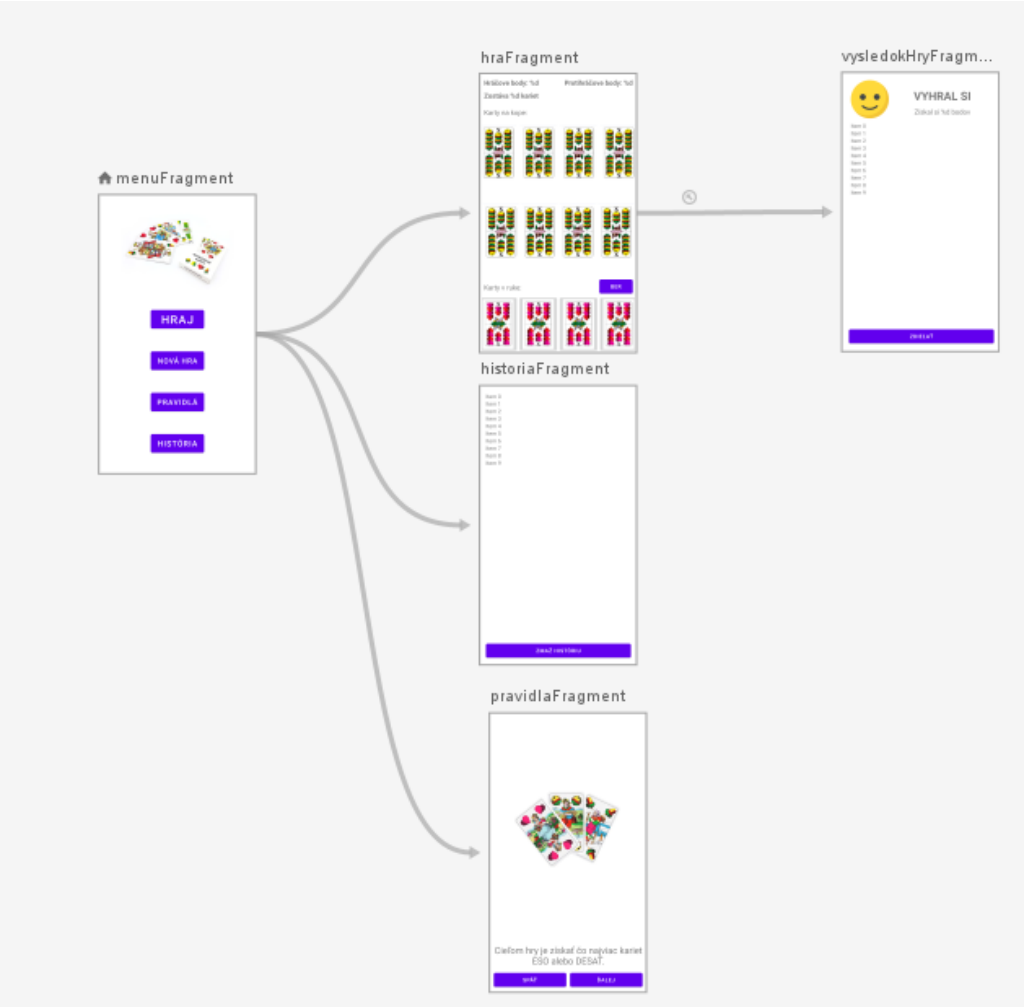
Analýza navrhovanej aplikácie

Moja aplikácia by mala byť čo najjednoduchšia na použitie, keďže sedmové karty hrajú ľudia všetkých vekových kategórií. Navigácia v aplikácii by mala byť jednoduchá na pochopenie a z dizajnového hľadiska by som sa chcel priblížiť Material dizajnu.

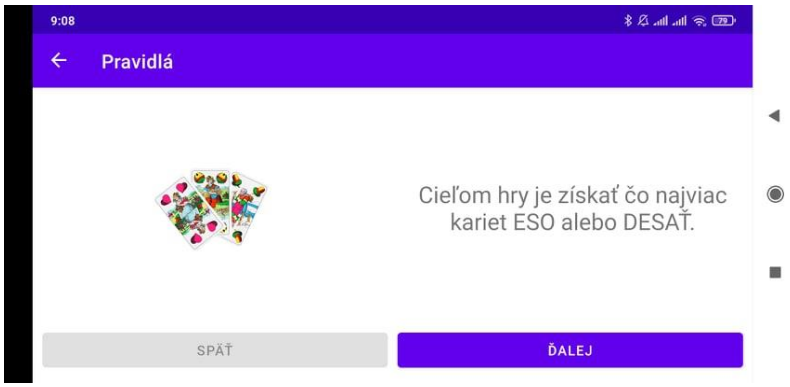
Hráč by mal mať informáciu o tom, aké karty sa nachádzajú v danom kole na stole, a preto by bolo rozumné ich zobrazovať vedľa seba a nie nad sebou. Pre ovládanie bude lepšie použiť klikanie na karty ako drag and drop, čo prispeje k zjednodušeniu ovládania. Karty, ktoré sú aktuálne v ruke by sa mali zobrazovať v spodnej časti herného okna, aby boli jednoducho dostupné palcom aj na smartphonoch s väčšou obrazovkou. Okrem kariet na stole a v ruke by mala hra užívateľa informovať o tom, koľko kariet ešte zostáva a aké je aktuálne skóre – tj. koľko bodov získal on a koľko protihráč (smartphone).

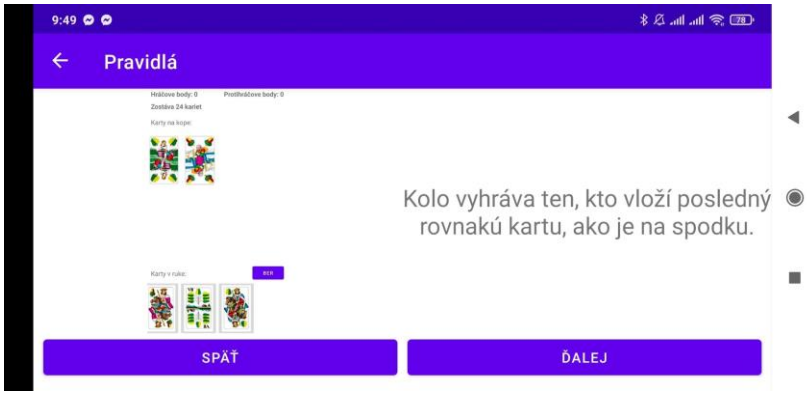
Po dohratí hry (či už po výhre alebo prehre) by sa mal hráč dozvedieť, aké karty sa mu podarilo získať a ako hra dopadla (tj. koľko získal bodov). Svoj výsledok by mal byť schopný z tohto fragmentu cez príslušné tlačidlo zdieľať a pochváliť sa tak svojmu okoliu. Aplikácia by mala ukladať výsledok každej hry spolu s dátumom, kedy hra prebehla, aby si užívateľ vedel neskôr pozrieť históriu svojich hier. Užívateľ by mal vedieť tento zoznam premazať.

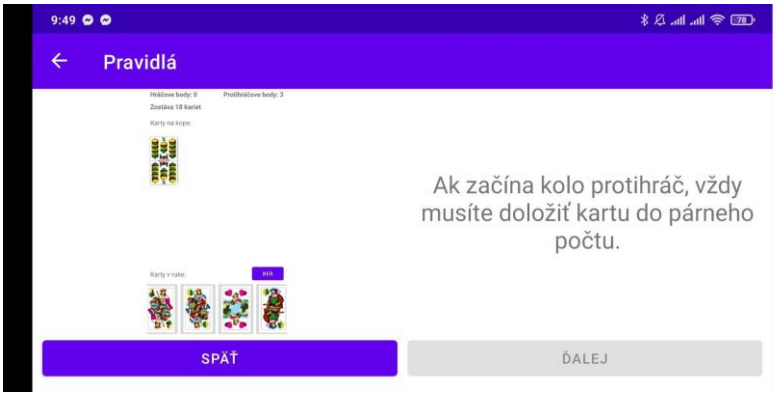
Návrh architektúry a ukážky obrazoviek



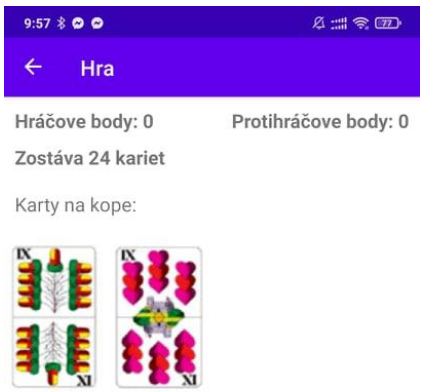
Po otvorení aplikácie sa zobrazí menu fragment, kde si užívateľ môže vybrať spomedzi viacerých možností – hraj (pokračuje v rozohranej hre, ak nejaká bola, inak sa otvorí nová hra), nová hra (vždy otvorí novú hru) pravidlá a história. V pravidlách hry sa môže preklikávať medzi viacerými ukážkami z hry spolu s popisom pravidla, pričom tlačidlá späť a ďalej sa automaticky aktivujú /deaktivujú, ak je na poslednej/prvej strane.

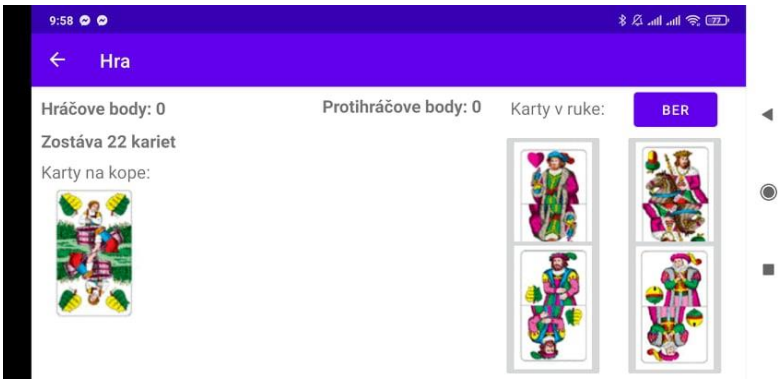


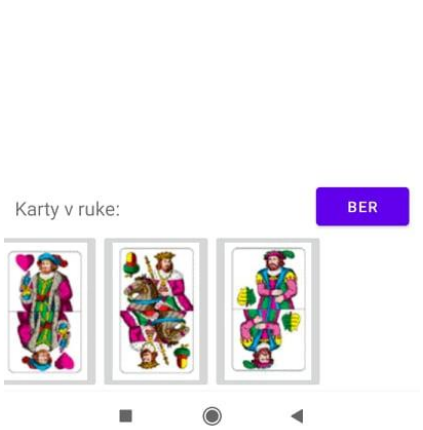


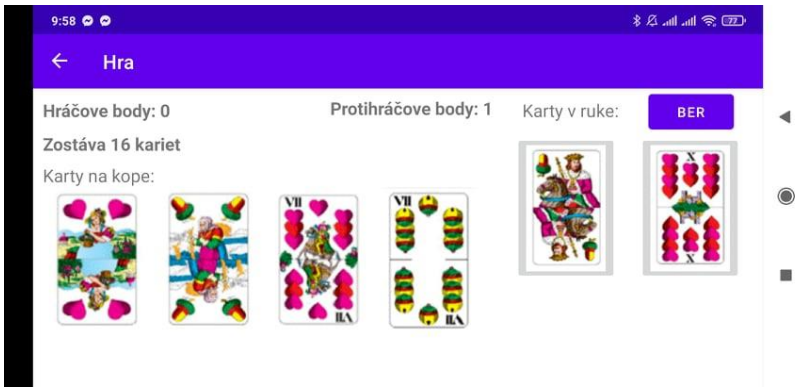


Samotná hra obsahuje 4 ImageButtony reprezentujúce karty v ruke a 8 ImageViewov, ktoré predstavujú karty položené na stôl. Hráč hru ovláda klikaním na karty v svojej ruke, prípadne môže použiť aj tlačidlo ber.

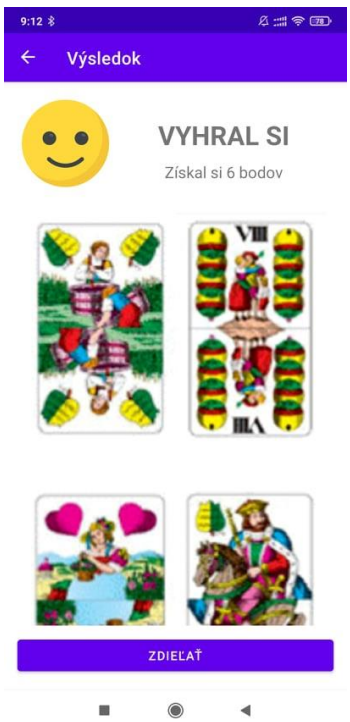


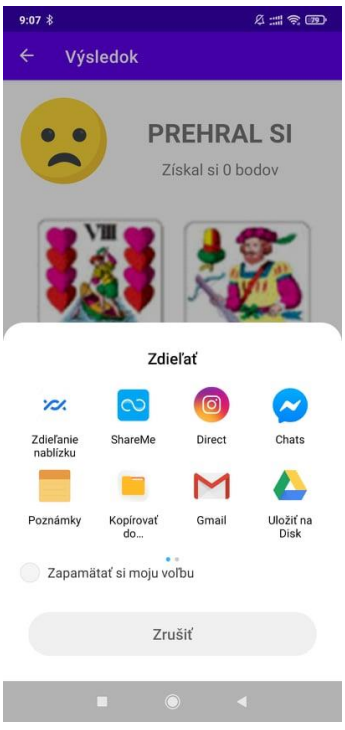




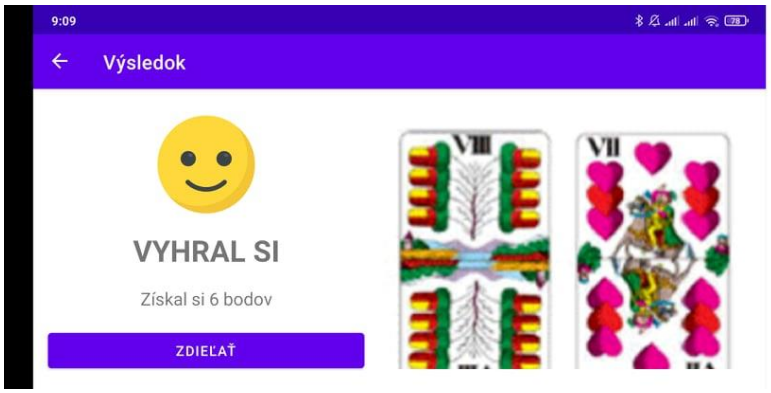


Po skončení hry sa hráčovi ukáže, aké karty získal, koľko bodov sa mu podarilo získať a svoj výsledok môže cez tlačidlo zdieľať so svojím okolím.





História hry zobrazuje históriu všetkých hier spolu s počtom bodov, dátumom a časom, kedy sa hra uskutočnila. Cez tlačidlo sa v prípade potreby dá história hier vymazať.



Popis implementácie

Celá aplikácia je postavená na fragmentoch a namiesto hľadania viewov a iných položiek cez findViewById som sa rozhodol použiť komponent data binding. Prepínanie fragmentov je sprostredkované cez navigation model. Vzhľadom na typ aplikácie väčšinou nebolo možné nastaviť univerzálne constrainty, preto som pre väčšinu fragmentov spravil dva XML súbory – jeden pre portrait a jeden pre landscape režim, aby aj po otočení obrazovky zostalo ovládanie intuitívne a prostredie aplikácie prehľadné.

MainActivity

Aby bola pre užívateľa orientácia v aplikácii čo najjednoduchšia, rozhodol som sa použiť navigation component. V action bare je vždy (samozrejme okrem menu) k dispozícii tlačidlo späť spolu s názvom fragmentu. V main activity je hneď na začiatku zobrazený menuFragment.

MenuFragment

Trieda menu fragment je jednoduchá a využívam v nej len overridnutú metódu onCreateView. V nej sa spúšťajú onClickListenery všetkých tlačidiel. Ak užívateľ klikne na tlačidlo HRAJ, cez findNavController().navigate sa otvorí fragment s hrou. Ak klikne na tlačidlo nová hra, do Bundle sa zabalí booleanovský atribút pod kľúčom „nova_hra“, vďaka čomu herný fragment zistí, že má namiesto pokračovania v starej hre otvoriť novú. Kliknutie na tlačidlo pravidlá hry alebo história hry vyvolá otvorenie príslušného fragmentu.

HraFragment

HraFragment spolupracuje s triedou Hra a dá sa povedať, že obstaráva len prácu s týmto objektom v rámci životného cyklu fragmentu (aby nedošlo k strateniu údajov pri otočení obrazovky alebo pri popnutí tohto fragmentu zo stacku). V onCreateView si na začiatku vypýta booleanovskú premennú pod kľúčom nova_hra, aby fragment vedel, či bol otvorený cez kliknutie na tlačidlo hrať alebo nová hra. Ak bol otvorený cez nová hra, a teda sa podarí získať túto premennú a jej hodnota je true, potom sa stará hra vymaže z pamäte a do atribútu hra sa vytvorí nová inštancia objektu Hra.

Booleanovská premenná hraUlozena, ktorá sa získava zo savedInstanceState pod kľúčom „hraUlozena" kontroluje, či ide o volanie onCreateView po otočení obrazovky. V takomto prípade volá metódu načítaj hru, ktorá inicializuje atribút hra zo savedInstanceStatu, kam ju ukladám v metóde ulozHru, ktorá sa volá z overridnutej metódy onSaveInstanceState.

Ak sa onCreate v tomto fragmente nevolá po otočení obrazovky a ani po stlačení tlačidla nová hra, potom sa zavolá metóda nacitajHruZoSuboru, ktorá sa pokúsi načítať uloženú hru z internej pamäte z binárneho súboru. Ak sa jej to podarí, automaticky inicializuje atribút a vráti true, ak je vyvolaná výnimka (samozrejme zachytená) a teda sa jej to nepodarí, vráti false a v takomto prípade sa vytvorí nový objekt typu Hra. Metóda ulozHruDoSuboru je zavolaná z onDestroyView a ukladá objekt hry cez ObjectOutputStream do súboru, keďže trieda hra spĺňa podmienku, že všetky jej atribúty sú Serializable. Referencia na binding sa však z pochopiteľných dôvodov nedá spraviť Serializable a ani nemá zmysel tento objekt ukladať, preto som sa rozhodol tento atribút nastaviť v triede hra na nullable. Tesne pred zápisom do súboru ho nastavujem na null a ihneď po prečítaní hry zo súboru ho inicializujem na správnu hodnotu.

Použiť v jednej triede dva typy ukladania som sa rozhodol pre rýchlosť a efektivitu. Pri otáčaní obrazovky nie je dôvod používať relatívne pomalý zápis do súboru, preto sa používa rýchlejší spôsob ukladania a načítania cez Bundle a naopak Bundle nedrží dáta po popnutí fragmentu alebo vypnutí aplikácie, preto vtedy ukladám hru do binárneho súboru a hráč môže pokračovať v hre tam kde skončil aj v prípade, že dôjde k úplnému vymazaniu aplikácie z operačnej pamäte.

Popis triedy Hra

Hernú logiku riadi trieda Hra. Obsahuje nasledovné atribúty:

binding – referencia na FragmentHraBinding z herného fragmentu, aby bolo možné jednoducho a efektívne pristupovať ku konkrétnym textViewom a imageViewom a nastavovať ich podľa potreby.

nacitanaHra – ak je true, konštruktor vie, že ide o hru, ktorá už bola predtým rozohraná, a teda nemusí inicializovať karty nanovo, ak je false, nejde o načítanú hru, ale o úplne novú hru, a preto naplní ArrayList novými kartami a rozdá príslušný počet hráčovi a protihráčovi.

hrac – reprezentuje hráča. Ide o objekt dátovej triedy Hrac, ktorá obsahuje atribúty karty ruka – tj. karty, ktoré hráč drží v ruke a atribút body typu int (získané body v danej hre). Triedu hrac som zaviedol kvôli tomu, aby sa znížila duplicita kódov a zvýšila prehľadnosť, keďže rovnaké atribúty má v danej hre aj protihráč. Táto trieda je zároveň Parcelable, aby ju bolo možné vkladať do Bundle a Serializable, aby bolo možné v inej časti kódov celú hru ukladať do súboru.

karty – ArrayList kariet, z ktorých sa po každom kole rozdáva.

ziskaneKartyHrac – Tento atribút obsahuje všetky karty z kôl, ktoré vyhral hráč, aby si mohol po skončení hry pozrieť, aké karty sa mu podarilo v danej hre získať. Mohlo by sa zdať, že tento atribút by

mal patriť do triedy Hrac, ale pre protihráča nemá zmysel ukladať jeho získané karty, keďže sa s nimi v budúcnosti nebude nijako narábať.

hracVyhralKolo - Atribút sa dynamicky mení v závislosti od toho, kto posledný „zabil“, respektíve by vyhral dané kolo v prípade, že by skončilo. Ak by ho vyhral hráč, atribút je nastavený na true, ak protihráč, tak na false.

kartyNaStole – Obsahuje referencie na karty, ktoré sú vyložené na stole (tj. sú to karty, o ktoré sa hrá v danom kole).

Konštruktor – Skontroluje, či ide o už načítanú hru, ak áno, znamená to, že atribúty kariet už sú naplnené z parametrov a nie je potrebné rozdávať a inicializovať karty. Ak nie, zavolá metódy inicailizujKarty a rozdajKarty. Metóda inicializuj karty priradí do ArrayListu kariet 32 unikátnych kariet, pričom každej nastaví typ a referenciu na jej obrázok. Metóda rozdajKarty je využívaná aj počas hry. V cykle rozdáva hráčovi a protihráčovi po jednej karte až pokiaľ neplatí, že majú v ruke 4 karty alebo sa karty na rozdanie minuli. Konštruktor na koniec zavolá metódy aktualizujZobrazenieKariet, ktorá sa stará o prekreslenie kariet na stole a v hráčovej ruke podľa hodnôt príslušných ArrayListov. Aktualizuj zobrazenie textu prekreslí textViewy s informáciami o zostávajúcom počte kariet, o bodoch hráča a o bodoch protihráča. Nakoniec je zavolaná metóda nastavListenery.

nastavListenery - nastaví onClickListener na štvorici ImageButtonov, ktoré reprezentujú hráčove karty a na tlačidle ber. Ak hráč klikne na jednu z kariet vo svojej ruke, onClickListener zavolá metódu hracovTah, kde ako parameter vstupuje index karty v jeho ruke, ktorú sa rozhodol použiť. Hneď nato je zavolaná metóda aktualizujZobrazenieKariet, aby užívateľské prostredie poskytovalo hráčovi vždy aktuálne informácie o stave, v akom sa hra nachádza. Tlačidlo ber skontroluje, či sa na stole aktuálne nachádza 2, 4, 6 alebo 8 kariet, pretože to je jediná situácia, keď je možné toto tlačidlo použiť (protihráčovi sa podarilo zabiť a čaká sa na rozhodnutie hráča). Ak táto podmienka platí, zavolá sa metóda koloSkoncilo a aktualizujZobrazenieKariet

Pomocné metódy potrebné pre vnútorné fungovanie hry:

dajIndexKartyZRukyProtihraca – preberá ako parameter typ karty a skontroluje, či má protihráč v ruke kartu daného typu. Ak áno, vráti jej index, v opačnom prípade vráti -1.

jeNaStoleDesatAleboEso – prezrie karty vyložené na stole. Ak sa medzi nimi nachádza desať alebo eso (tj. karty s bodovým ohodnotením), vráti true, inak vráti false.

maProtihracObycajnuKartu – prezrie karty v ruke protihráča a ak má kartu, ktorá nie je ani desať, ani eso a ani sedmička, potom vráti jej index. V opačnom prípade vráti -1.

Princíp fungovania hry

Metóda hráčov ťah sa na začiatku presvedčí, či toto kolo začínal hráč alebo protihráč. Ak je počet kariet na stole nepárny (1, 3, 5 alebo 7), znamená to, že prvú kartu vykladal protihráč. Karta sa podľa indexu z parametra odoberie z hráčovej ruky a pridá sa na stôl. Ak sa typ karty rovná typu karty, ktorá je položená ako prvá alebo ak je typ karty sedem, atribút hráč vyhral kolo sa nastaví na true. Ak je už na stole 8 kariet, znamená to, že hráč ani protihráč už nemajú v ruke žiadne karty, a preto nemá zmysel ďalej pokračovať a automaticky sa volá metóda koloSkoncilo. Táto metóda sa hneď zavolá aj v prípade, že hráč nevyložil kartu, ktorá by dokázala v tomto kole vyhrať.

Ak je počet kariet na stole párny, znamená to, že v danom kole vykladal prvú kartu hráč. Ak vykladá prvú kartu, karta sa pridá na stôl a zavolá sa metóda protihracovTah. Ak už na stole sú karty a hráč vykladal párnu kartu v tomto kole, znamená to, že protihráčovi sa podarilo zabiť. V takomto prípade

môže hráč vyložiť kartu len ak ide o kartu aká je na spodku alebo kartu typu sedem. Ak to platí, karta sa vyloží, hráč vyhral kolo sa nastaví na true a zavolá sa protihráčov ťah. Ak to neplatí, hráč je cez toast.makeText informovaný o tom, že karta, ktorú sa snažil vyložiť, nemôže zabiť a karta zostáva v jeho ruke. Ak hráč nemá v ruke príslušnú kartu, musí použiť tlačidlo ber.

Metóda protihráčov ťah hneď na začiatku skontroluje, koľko kariet je na stole. Ak je ich nepárny počet, znamená to, že prvú kartu vykladal v tomto kole hráč (a teda je pre ukončenie kola potrebné vždy vyložiť kartu). Cez metódu dajIndexKartyZRukyProtihraca skontroluje, či má protihráč v ruke kartu daného typu. Ak áno, vyloží ju na stôl a nastaví hodnotu hráč vyhral kolo na true. Ak nie, skontroluje, či má hráč v ruke sedmičku. Ak áno, skontroluje, či je na stole desať alebo eso (tj. či sa danú sedmičku oplatí v tomto kole použiť alebo sa viac oplatí ju odložiť do budúcna, kde by mohla získať body). Ak je na stole desať alebo eso, vyloží na stôl sedmičku. Ak nie je, skontroluje, či má v ruke obyčajnú kartu. Ak áno, vyloží ju, ale ak nie, neoplatilo by sa dávať hráčovi cennú kartu, a preto použije sedmičku.

Ak neplatila ani prvá podmienka a hra je v situácii, že protihráč nemá kartu, aká je na spodku a ani sedmičku, potom skontroluje, či má obyčajnú kartu. Ak áno, vyloží ju, ak nie, vyloží prvú kartu z ruky. V prípade, že sa hráčovi podarilo zabiť a je na stole už 8 kariet, automaticky sa zavolá metóda kolo skončilo. V opačnom prípade hra čaká na hráčov ťah.

Druhá časť metódy počíta so situáciou, že na stole je pred protihráčovým ťahom párny počet kariet, a teda že kolo začínal protihráč, ale hráčovi sa podarilo zabiť. V takomto prípade skontroluje, či má index rovnakej karty, ako je na spodku, ak áno, použije ju. Ak nie, skontroluje, či má sedmičku a ak áno, skontroluje, či sa hrá o desiatku alebo eso. Ak áno, použije sedmičku. Ak nemá sedmičku alebo sa nehrá o desiatku alebo eso, je automaticky zavolaná metóda kolo skončilo.

Funkcia koloSkoncilo aktualizuje zobrazenie kariet. Následne sa cez Handler dosiahne oneskorené vyvolanie kódu, ktorý spočíta body v kole podľa počtu desiatok a es v kartách na stole. Oneskorenie o 1 sekundu zaručí, že hráč vidí, akú kartu vyložil protihráč a až potom sa karty zo stola upracú a môže začať nové kolo. Ak kolo vyhral hráč, pripočíta mu body, aktualizuje zobrazenie textov a karty zo stola pridá k jeho získaným kartám. Potom vyčistí karty na stole a rozdá nové karty.

Ak kolo vyhral protihráč, body sa pripočítajú jemu a ak má ešte nejaké karty v ruke, jednu z nich vyloží na stôl, keďže nasledujúce kolo začína on. Ak už hráč nemá na ruke žiadnu kartu (aj po vyvolaní metódy rozdajKarty), znamená to, že sa karty minuli a hra končí.

Do Bundle sa cez putParcellableArrayList zabalí arraylist získaných kariet a cez putInt získané body. Prostredníctvom findNavController().navigate sa otvorí nový fragment a balíček s príslušnými dátami sa doňho odošle. Ešte predtým sa do objektu typu VysledokHry uloží počet získaných bodov a aktuálny dátum a čas hry získaný z Calendara vo forme stringového reťazca. Tento objekt sa pošle ako parameter do metódy zapisNovy objektu triedy UkladanieVysledkovHry, ktorá ho zapíše ku ostatným výsledkom hier do súboru. Aplikácia si tak uchováva informácie o tom, ako sa hráčovi darilo a hráč si môže svoje výsledky pozrieť v sekcii História.

VysledokHryFragment

V metóde onCreateView sa hneď na začiatku získa referencia na DataBinding tohto fragmentu. Získané body sa spolu s referenciou na ArrayList získaných kariet vytiahnu z balíčka, ktorý do tohto fragmentu poslala hra. Následne sa zavolá metóda zobrazBodyAVyhru.

Tá nastaví textView, ktoré informuje o počte získaných bodov na správne naformátovaný text, ktorý vracia metóda getZiskaneBodyText. Vďaka tejto metóde je text vždy v správnom tvare vo vzťahu k počtu bodov. V tomto prípade ide o špecifickosť slovenského jazyka. Ak je bodov viac ako 4 alebo 0,

metóda vráti text „Získal si {8/7/6/5/4/0} bodov“, ak je jeden, vráti „Získal si 1 bod“ a ak sú 3 alebo 2, potom vráti „Získal si {2/3} body“. Referencie na príslušný stringový resource získavam pomocou binding.root.resources.getString(R.string.nazovStringu).

Metóda getVyhralPrehralText formátuje text pre druhý textView a samozrejme aj pre zdieľanie. Ak je bodov viac ako 4, vráti text (samozrejme opäť z resourcov) „VYHRAL SI“, ak sú body vyrovnané, vráti „REMÍZA“ a ak je bodov menej ako 4, vráti „PREHRAL SI“.

Po inicializácii dvoch textViewov je nastavený obrázok smajlíka. Ak je bodov 8, nastaví sa najšťastnejší smajlík, ak ich je viac ako 4, nastaví sa veselý, ak 4, tak neutrálny, ak 0, tak plačúci a ak menej ako 4, tak smutný.

Následne sa inicializuje recycler view získaných kariet s využitím príslušnej triedy ZiskaneKartyAdapter. Cez binding sa aktivuje onClickListener na tlačidlo zdieľať, ktorý volá metódu zdieľajVýsledok. Prostredníctvom tohto tlačidla môže hráč jednoducho zdieľať výsledok hry so svojím okolím. V metóde dajMiZdielaciIntent sa vytvorí príslušný intent na odoslanie textovej správy s parametrom getHracVyhralText a getZiskaneBodyText. Metóda zdielaj vysledok zavolá startActivity s týmto vytvoreným intentom.

Vďaka nastaveniu v navigation modeli stlačenie tlačidla späť z tohto fragmentu nevráti hráča späť do prázdnej hry, ale vráti ho to do menu, odkiaľ môže spustiť novú hru alebo si pozrieť históriu hier a podobne.

HistoriaFragment

V onCreateView sa vytvorí inštancia objektu UkladanieVysledkovHry, prostredníctvom ktorého načítam cez metódu dajVsetky všetky doteraz uložené hry. Tie pošlem ako parameter do konštruktora triedy HistoriaHryAdapter, čo je adaptér pre RecyclerView zobrazujúci históriou hier. Nakoniec sa aktivuje onClickListener na tlačidle vymaž históriu, ktorý v prípade stlačenia zavolá zmazVsetky nad objektom načítavača hier (to vymaže uložené hry v pamäti), potom zavolá clear nad výsledkami hier (to vymaže hry z ArrayListu, z ktorého čerpá aj recyclerView) a následne zavolá nad historiaAdapterom recycler viewu notifyDataSetChanged, vďaka čomu sa prekreslí história na vymazanú aj v recyclerViewe na obrazovke.

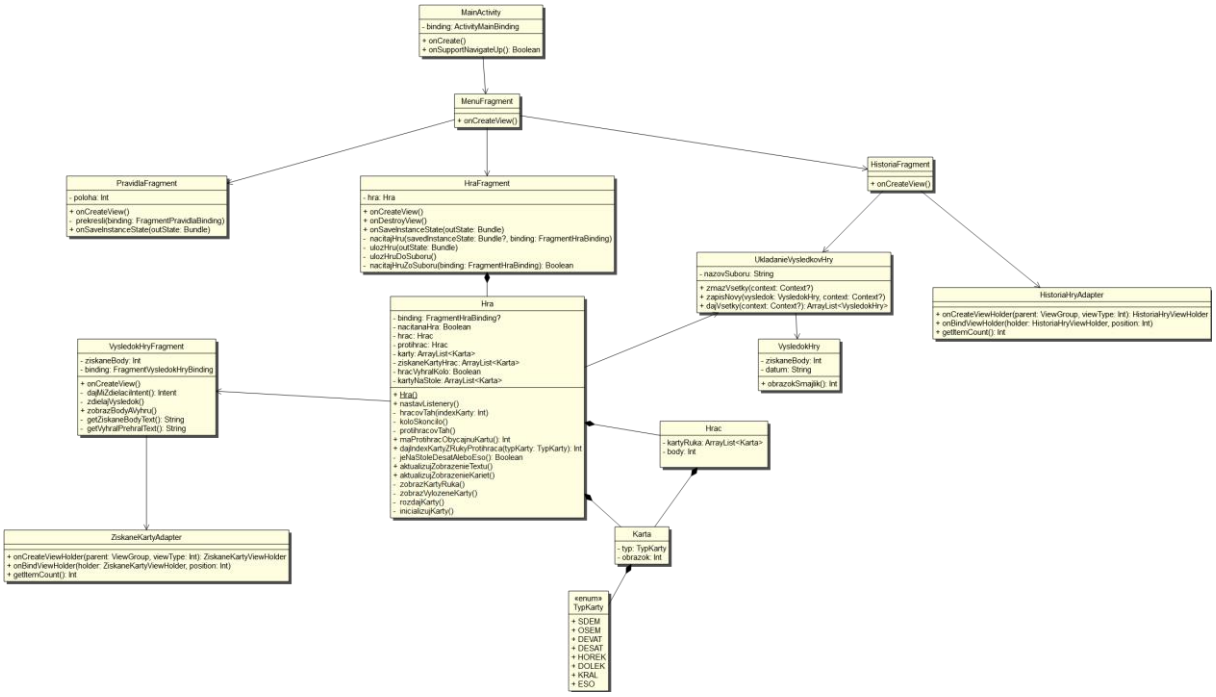
PravidlaFragment

Vo fragmente pravidiel sa dá prepínať medzi niekoľkými „obrazovkami“, ktoré sú reprezentované zmenou resourcu v imageViewe a textViewe - aplikácia ukáže danú situáciu v hre s popisom a po stlačení tlačidla treba tieto hodnoty zmeniť. Aby aplikácia vedela, ktoré pravidlo je práve zobrazované, trieda PravidlaFragment obsahuje atribút poloha. Ten v onSaveInstanceState ukladám pod kľúčom „poloha“ do balíčka, aby sa dal v onCreateView opäť vytiahnuť, ak došlo k otočeniu obrazovky. Tj. ak je null a pravidlá boli ešte len otvorené, poloha je automaticky na 0 a ak nie je null, poloha sa nastaví na stav, v akom bola pred otočením obrazovky.

Funkcia prekresli nastavuje podľa polohy príslušné image a text resourcy do viewov. Ak je poloha 0 (tj. hráč je na prvej strane), nastaví ľavému tlačidlu isEnabled na false (vľavo už sa nie je kam prepnúť), takisto nastaví isEnabled na false pravému tlačidlu, ak poloha signalizuje, že už sa zobrazuje posledná strana.

OnClickListenery na tlačidlách fungujú tak, že sa poloha zvýši (pravé tlačidlo) alebo zníži o 1 (ľavé tlačidlo) a zavolá sa metóda prekresli.

UML diagram


MainActivity
- binding: ActivityMainBinding
+ onCreate()
+ onSupportNavigateUp(): Boolean
MenuFragment
+ onCreateView()
PravidlaFragment
- poloha: Int
+ onCreateView()
- prekresli(binding: FragmentPravidlaBinding)
+ onSaveInstanceState(outState: Bundle)
HraFragment
- hra: Hra
+ onCreateView()
+ onDestroyView()
+ onSaveInstanceState(outState: Bundle)
- nactajHru(savedInstanceState: Bundle?, binding: FragmentHraBinding)
- ulozHru(outState: Bundle)
- ulozHruDoSuboru()
- nactajHruZoSuboru(binding: FragmentHraBinding): Boolean
HistoriaFragment
+ onCreateView()
UkladanieVysledkovHry
- nazovSuboru: String
+ zmazVsetky(context: Context?)
+ zapisNovyVysledok(vysledok: VysledokHry, context: Context?)
+ dajVsetky(context: Context?): ArrayList<VysledokHry>
HistoriaHryAdapter
+ onCreateViewHolder(parent: ViewGroup, viewType: Int): HistoriaHryViewHolder
+ onBindViewHolder(holder: HistoriaHryViewHolder, position: Int)
+ getItemCount(): Int
Hra
- binding: FragmentHraBinding?
- nactanaHra: Boolean
- hrac: Hrac
- protihrac: Hrac
- karty: ArrayList<Karta>
- ziskaneKartyHrac: ArrayList<Karta>
- hracVyhralKolo: Boolean
- kartyNaStole: ArrayList<Karta>
+ hra()
+ nastavListenery()
- hracovTah(indexKarty: Int)
- koloSkoncilo()
- protihracovTah()
+ maProtihracCitycajnuKartu(): Int
+ dajIndexKartyZRukyProtihraca(typKarty: TypKarty): Int
- jeNaStoleDesatAleboEso(): Boolean
+ aktualizujZobrazenieTextu()
+ aktualizujZobrazenieKariet()
- zobrazKartyRuka()
- zobrazVylozeneKarty()
- rozdajKarty()
- inicializujKarty()
VysledokHry
- ziskaneBody: Int
- datum: String
+ obrazokSmajlik(): Int
VysledokHryFragment
- ziskaneBody: Int
- binding: FragmentVysledokHryBinding
+ onCreateView()
- dajMiZdielaciIntent(): Intent
- zdielajVysledok()
+ zobrazBodyAVyhru()
- getZiskaneBodyText(): String
- getVyhralPrehralText(): String
Hrac
- kartyRuka: ArrayList<Karta>
- body: Int
ZiskaneKartyAdapter
+ onCreateViewHolder(parent: ViewGroup, viewType: Int): ZiskaneKartyViewHolder
+ onBindViewHolder(holder: ZiskaneKartyViewHolder, position: Int)
+ getItemCount(): Int
Karta
- typ: TypKarty
- obrazok: Int
«enum»
TypKarty
+ SDEM
+ OSEM
+ DEVAT
+ DESAT
+ HOREK
+ DOLEK
+ KRAL
+ ESO