ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

VÝVOJ APLIKACÍ PRO ANDROID V PROSTŘEDÍ INTELLIJ

BAKALÁŘSKÁ PRÁCE

Martin Hofman
Informatika se zaměřením na vzdělávání

Vedoucí práce: PhDr. Denis Mainz, Ph.D.

Plzeň, 2017

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 18. dubna 2017

.....

vlastnoruční podpis

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce, kterým byl PhDr. Denis Mainz, Ph.D., a to za odborné vedení práce, mnoho věcných rad a doporučení, které mi velmi pomohly při zpracování této práce.

ZDE SE NACHÁZÍ ORIGINÁL ZADÁNÍ KVALIFIKAČNÍ PRÁCE.

OBSAH

SEZNAM ZKRATEK

API	- rozhraní pro programování aplikací
ART	- virtuální stroj
DOM Parser	- analyzátor dokumentu podle objektů
GUI	- grafické uživatelské rozhraní
HAL	- hardwarová abstraktní vrstva
JDK	- balíček základních nástrojů pro vývoj Java aplikací
OHA	- Open Handset Alliance
SDK	- sada vývojových nástrojů
UML	- standartní grafický způsob návrhu aplikace
XML	- rozšiřitelný značkovací jazyk

ÚVOD

Tato práce je zaměřena na vývoj Android aplikací a prostředí IntelliJ. V současné době je vývoj aplikací pro chytré telefony, tablety a další zařízení velmi rozšířený. Denně se objevují nové aplikace, které si může kdokoliv většinou zdarma stáhnout a nainstalovat. Ovšem ne každá aplikace koncovému uživateli vyhovuje, a proto můžeme v Google Play Store najít celou řadu volně dostupných nebo placených alternativ. Cílem práce je seznámit čtenáře s možnostmi vývoje Android aplikací pomocí volně dostupných programových nástrojů, zdůvodnit výběr vhodného prostředí pro vývoj Android aplikací a popsat jeho součásti v kontextu vývoje programových aplikací. Dalším cílem je ukázat na vhodných příkladech funkce vybraného vývojového prostředí využívaného při tvorbě programových aplikací pro operační systém Android a navrhnout postup, který by mohl být uplatněn při vývoji.

První kapitola práce je věnována seznámení s obecnými informacemi o platformě Android. Dále se nachází popis několika nejpoužívanějších vývojových prostředí pro Android. Kapitola je ukončena pojednáním o kritériích pro výběr optimálního vývojového prostředí.

Druhá kapitola tvoří jedno z ústředních témat práce a zabývá se především podrobným popisem a charakteristikou vývojového prostředí IntelliJ. Čtenář bude seznámen s hlavními výhodami, které přináší používání tohoto prostředí, a jakým způsobem prostředí získat a zprovoznit. Poslední částí kapitoly je seznámení s grafickým uživatelským rozhraním (GUI) a jeho možnými úpravami pro individuální požadavky.

Následující kapitola ukazuje na vlastní sadě příkladů možnosti prostředí. Sada příkladů byla založena na úlohách řešených v rámci předmětu programování, který se ovšem orientoval na jiný programovací jazyk. Hlavním důvodem převzetí příkladů bylo, aby příklady odpovídaly didaktickým zásadám výuky programování.

Poslední kapitola je zaměřena na doporučený postup při vývoji Android aplikací. Uvedený postup je dílem osobního názoru autora práce vycházející také z konzultací na uvedené téma s odbornou veřejností. Závěrem čtenář zjistí, jaké jsou možnosti při migraci Java aplikací na Android aplikace.

1 ANDROID APLIKACE A DOSTUPNÉ NÁSTROJE

1.1 ANDROID

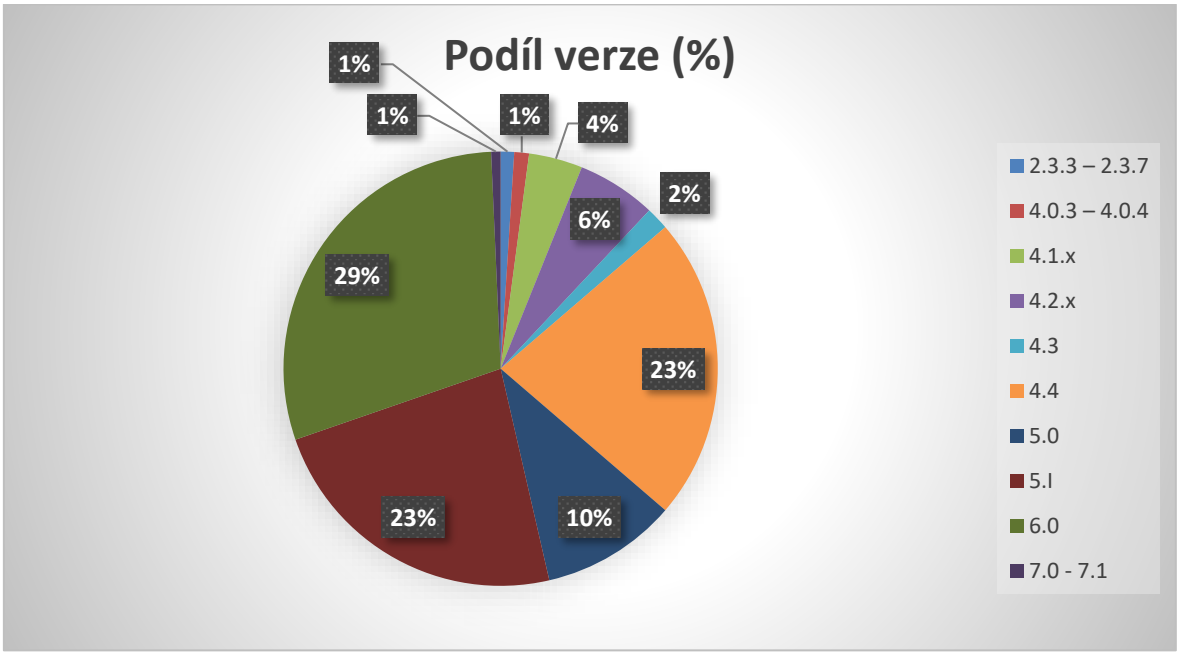
Jedná se v současnosti o nejrozšířenější operační systém pro mobilní zařízení. Nejčastěji se s ním můžeme setkat na tabletech či smart (neboli chytrých) zařízeních, jako jsou telefony, hodinky a televize. V blízké budoucnosti se s tímto systémem pravděpodobně budeme setkávat i v automobilech. Systém je vytvořený tak, aby fungoval na hardwaru co nejvíce přístrojů. V sekci chytrých telefonů se dlouhodobě Android drží nad 70 % podílu na trhu[1]. Koncem roku 2016 tento podíl činil dokonce 81,7 %[2]. Ovšem málokterý výrobce telefonů dodává ve svých telefonech neupravený Android. Většina z nich má svoji nadstavbu. Například společnost Samsung využívá TouchWiz, HTC nazývá svoji nadstavbu Sense a LG Optimus UI.

Android je vyvíjen společnostmi Google a Open Handset Alliance. OHA je společenstvo téměř 100 společností, které vyvíjejí standardy pro mobilní zařízení, a jeho cílem je progresivní vývoj mobilních technologií. Google je jedna z největších společností, která v oblasti softwaru a služeb existuje. Společnost má v současnosti více než 57 tisíc zaměstnanců[3] v 70 pobočkách po celém světě[4]. Jedna z poboček se nachází v Praze.

První verze Androidu byla uvedena na trh 23. 9. 2008. Android 1.0 byl na rozdíl od současné verze závislý na hardwaru. Postupem času vycházely nové a pořád se zlepšující verze. Například již začátkem roku 2011 byl vydán Android 3.0, který byl určen jen pro tablety. Poté následovala verze 4 a její deriváty. Tyto verze ještě dodnes běží na velkém množství zařízení. Předposlední verzí je verze 6.0 (s označením Marshmallow), která podporuje USB typu C, rozpoznává otisky prstů, nebo výrazně prodlužuje výdrž telefonu na jedno nabití. USB typu C[5] je univerzální a lze ho využít pro více činností naráz. V jeden okamžik můžete jeho pomocí nabíjet potřebné zařízení a přenášet obraz. Přenos dat pomocí tohoto

[1] Android vládne mobilnímu trhu. *Adsl.cz* [online]. 2017 [cit. 2017-03-19]. Dostupné z: http://www.adsl.cz/clanky/android-vladne-mobilnimu-trhu
[2] Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. *Gartner* [online]. Egham, UK, 2017 [cit. 2017-03-19]. Dostupné z: http://www.gartner.com/newsroom/id/3609817
[3] Google's hiring may have slowed, but it's still adding thousands of new employees. *Business Insider* [online]. [cit. 2017-03-30]. Dostupné z: http://www.businessinsider.com/google-has-57000-employees-2015-7
[4] Our offices. *Google* [online]. [cit. 2017-03-30]. Dostupné z: https://www.google.com/intl/en/about/locations/
[5] USB Type-C™ Cable and Connector Specification. *Universal Serial Bus* [online]. [cit. 2017-03-19]. Dostupné z: http://www.usb.org/developers/usbtypec/

konektoru dosahuje až 10Gb/s. V současné době je nejnovější verze 7.0 (s označením Nougat), která umožňuje práci s okny a má API pro virtuální realitu.



Graf 1: Podíl verzí Androidu na trhu, data z ledna 2017 (zdroj: [6])

1.2 VNITŘNÍ STRUKTURA ANDROIDU

Vnitřní architektura je tvořena pěti vrstvami, které jsou částečně propojeny, ale každá má jiný účel.

Linuxové jádro je nejnižší vrstvou architektury. Jeho hlavním úkolem je propojení hardwaru a softwaru. Jádro tedy obsahuje zejména ovladače, správu paměti nebo řízení spotřeby energie. Velkou výhodou linuxového jádra je, že Android může využívat jeho bezpečnostní prvky.

Nad jádrem se v hierarchii nachází hardwarová abstraktní vrstva, která poskytuje jednotné API pro ovládání různého hardwaru. HAL je tedy tvořena knihovními moduly, které implementují rozhraní pro určitou hardwarovou součástku, například pro fotoaparát, bluetooth a reproduktory.

Další vrstva se skládá ze dvou částí. První je virtuální stroj (ART). Každá aplikace má tedy vlastní proces, který má vlastní instanci ART. ART je naprogramován tak, aby zvládl běh několika svých virtuálních instancí a také se staral o potřebnou paměť. Dále je vrstva

[6] Dashboards. *Android Developers* [online]. [cit. 2017-02-20]. Dostupné z: https://developer.android.com/about/dashboards/index.html

tvořena nativními knihovnami v C/C++. Například ART a HAL jsou postaveny na těchto knihovnách. I další vrstva, tedy Java API Framework, je založena na těchto knihovnách.

Již zmiňovaný Java API Framework je nejdůležitější vrstvou pro samotný vývoj aplikací v Javě. API je základ pro vývoj Android aplikací a poskytuje zjednodušené použití jádra systému. Tímto tedy programátor získává přístup k mnoha službám a funkcím, které může v aplikaci využít. Mezi základní služby patří například sada prvků View, které slouží k sestavení UI aplikace. Obsahem tedy jsou textová pole, tlačítka atd. Jako dalšího zástupce lze uvést Notification manager, který umožňuje aplikacím zobrazit vlastní upozornění ve stavovém řádku.

Poslední vrstvou v architektuře jsou systémové aplikace. Jedná se o sadu základních aplikací, například webový prohlížeč, e-mailový klient nebo kalendář. Tyto aplikace nejsou běžně využívány a uživatel přístroje je nahradí jinými z obchodu Google Play. Tyto aplikace však mohou usnadnit práci při vývoji. Programátor již nemusí ve své aplikaci vše vymýšlet sám, ale může využít funkcionalitu jedné ze systémových aplikací.[7]

1.3 DOSTUPNÁ VÝVOJOVÁ PROSTŘEDÍ

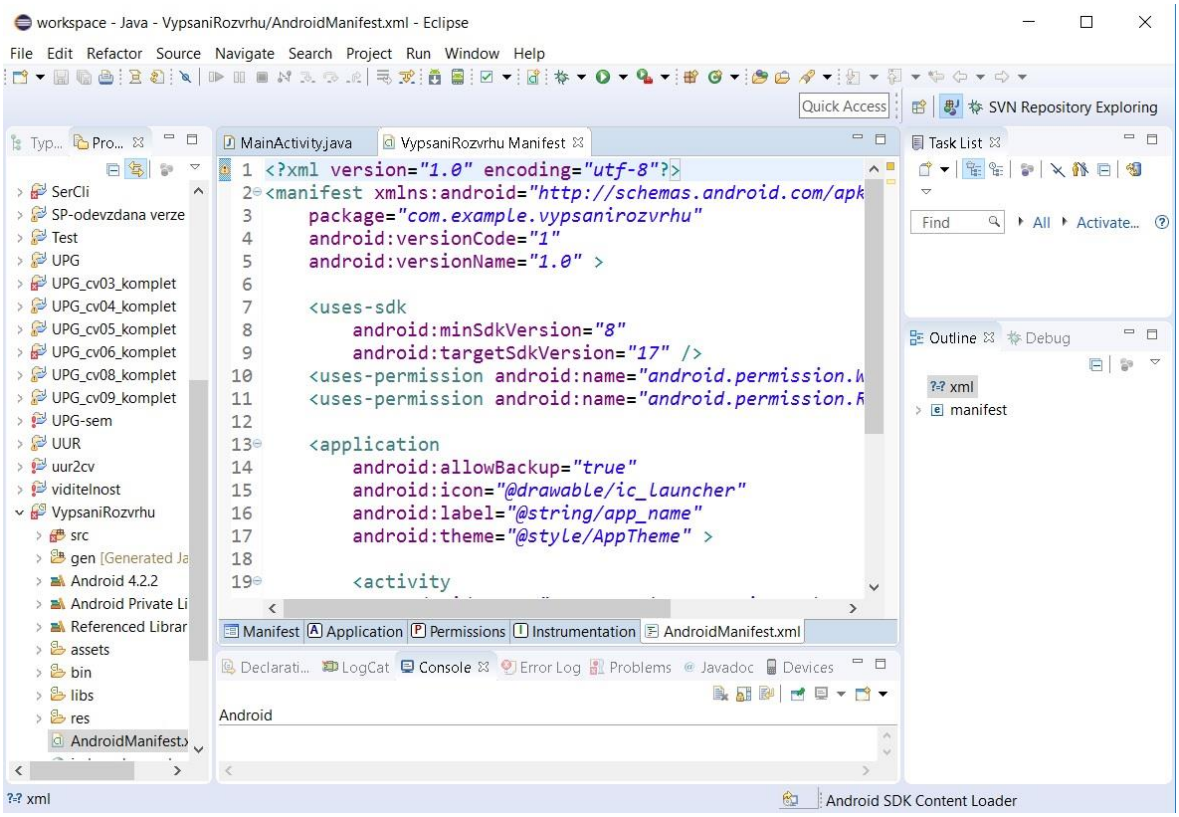
Pro vývoj Android aplikací existuje mnoho prostředí, proto jsem vybral ta v dnešní době nejznámější a nejvíce používaná. To dokládá žebříček uvedený na stránkách pypl.github.io/IDE.html. Jsou to Android Studio, Corona SDK, IntelliJ a Eclipse. Ve všech uvedených prostředích kromě Corona SDK lze programovat aplikace jak v Javě, tak v C/C++. Corona SDK je prostředí zaměřené na jazyk Lua. Lua je procedurální skriptovací jazyk, který je také nejrychlejší z interpretovaných jazyků v překladu programu do bajtkódu[8]. V praxi se také používá LuaJIT, která je ještě výkonnější než klasická Lua, a to je zajištěno tím, že ke kompilaci úseku kódu dojde, až když je ho potřeba.

1.3.1 ECLIPSE

Jedná se o jeden z prvních vývojových nástrojů, ve kterém lze programovat Android aplikace, i když k tomu nebyl primárně určen. Primárně byl určen pro vývoj Java aplikací. Pro Eclipse byl vytvořen plugin, který umožňuje tvorbu Android aplikací. Obecně díky

[7] Platform Architecture. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/guide/platform/index.html
[8] Programovací jazyk Lua. *ROOT.CZ* [online]. 2009 [cit. 2017-03-19]. Dostupné z: https://www.root.cz/clanky/programovaci-jazyk-lua/

pluginům lze v Eclipse programovat ve velkém množství programovacích jazyků. Jeho hlavní výhodou je, že programátor nemusí měnit prostředí a může začít s vývojem Android aplikací přímo v Eclipse, i když koncem roku 2015 skončila oficiální podpora Google[9]. Hlavně díky tomu se i v dnešní době toto prostředí používá, ačkoliv už existují specializovaná prostředí přímo pro Android aplikace. V současnosti je k dispozici už 11. verze tohoto prostředí a v polovině roku 2017 je plánovaná další verze.



Obrázek 1: Vývojové prostředí Eclipse (zdroj: vlastní)

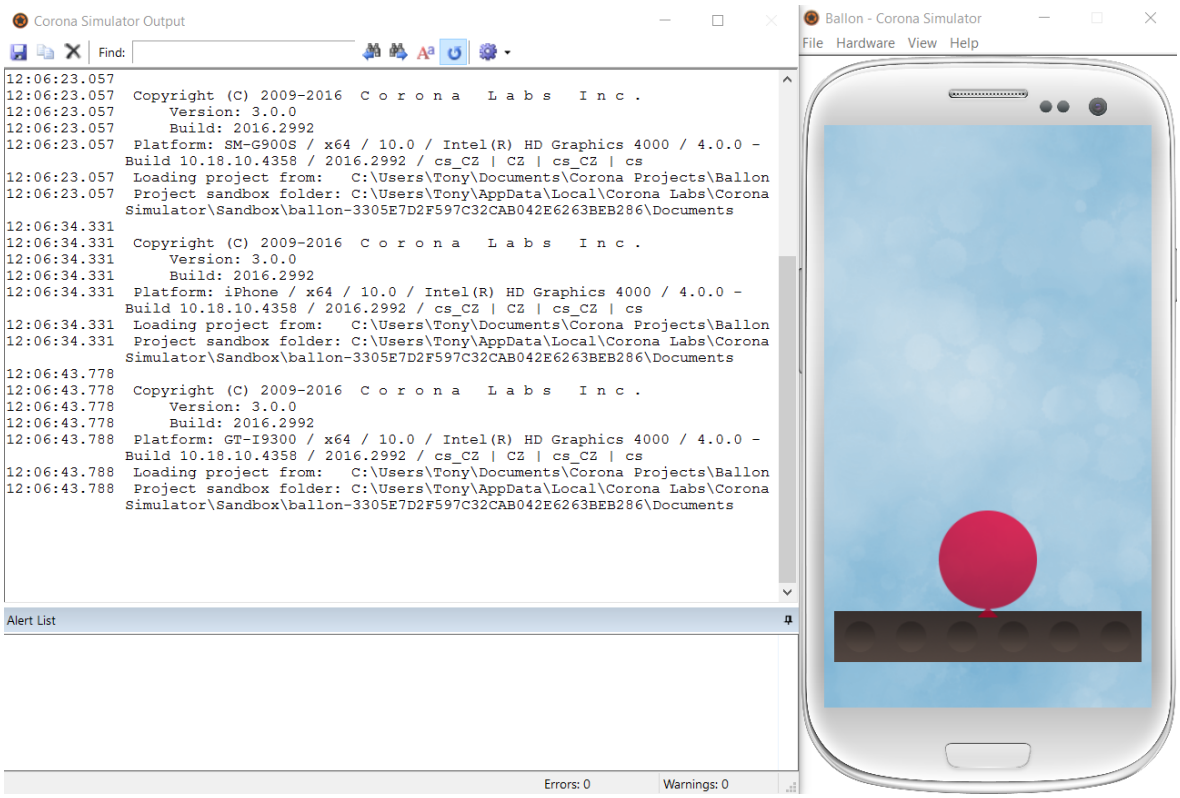
1.3.2 CORONA SDK

Jedná se o jeden z mála multiplatformních vývojových nástrojů. Programátor tedy vytvoří aplikaci, a poté jednoduše vytvoří verzi jak pro Android, tak třeba i pro Windows. Aplikace v tomto prostředí jsou vytvářeny pomocí jazyka Lua[10]. Uživatel, který je zvyklý spíše na jiné typy vývojových prostředí a nemá s Coronou dřívější zkušenost, může být při jejím prvním spuštění poměrně zmaten. Ihned po vytvoření projektu se zobrazí jak simulátor textový,

[9] CRIDER, Michael. *Google officially ends support for Eclipse Android Developer Tools in favor of Android Studio* [online]. 2016 [cit. 2017-03-19]. Dostupné z: http://www.androidpolice.com/2016/11/02/google-officially-ends-support-for-eclipse-android-developer-tools-in-favor-of-android-studio/

[10] WHY CHOOSE CORONA SDK ? *Corona Labs* [online]. [cit. 2017-03-19]. Dostupné z: https://coronalabs.com/corona-sdk/

tak i grafický, a otevře se složka s projektem. Ovšem v tuto chvíli není jasné, kam se má začít psát kód budoucí programové aplikace. Programátor je tedy nucen využít webových stránek dodavatele vývojového nástroje. Na webu www.docs.coronalabs.com se nachází vzorový první projekt, který si může každý vytvořit na svém prostředí, a tím pochopit fungování vývoje pomocí Corona Simulator. Po dokončení první části vzorového projektu byla objasněna většina z otázek. Po dokončení celého vzorového projektu je patrné, že vývoj aplikací v tomto prostředí je vcelku jednoduchý. Ovšem hlavní nevýhodou prostředí je absence kontextové nápovědy, která by byla pro nováčky při programování v jazyce Lua velmi nápomocná. Samotný kód se píše v běžném textovém editoru, tím pádem je i případná chyba v názvu zjištěna až při kompilaci kódu a neprobíhá tedy kontrola v reálném čase.

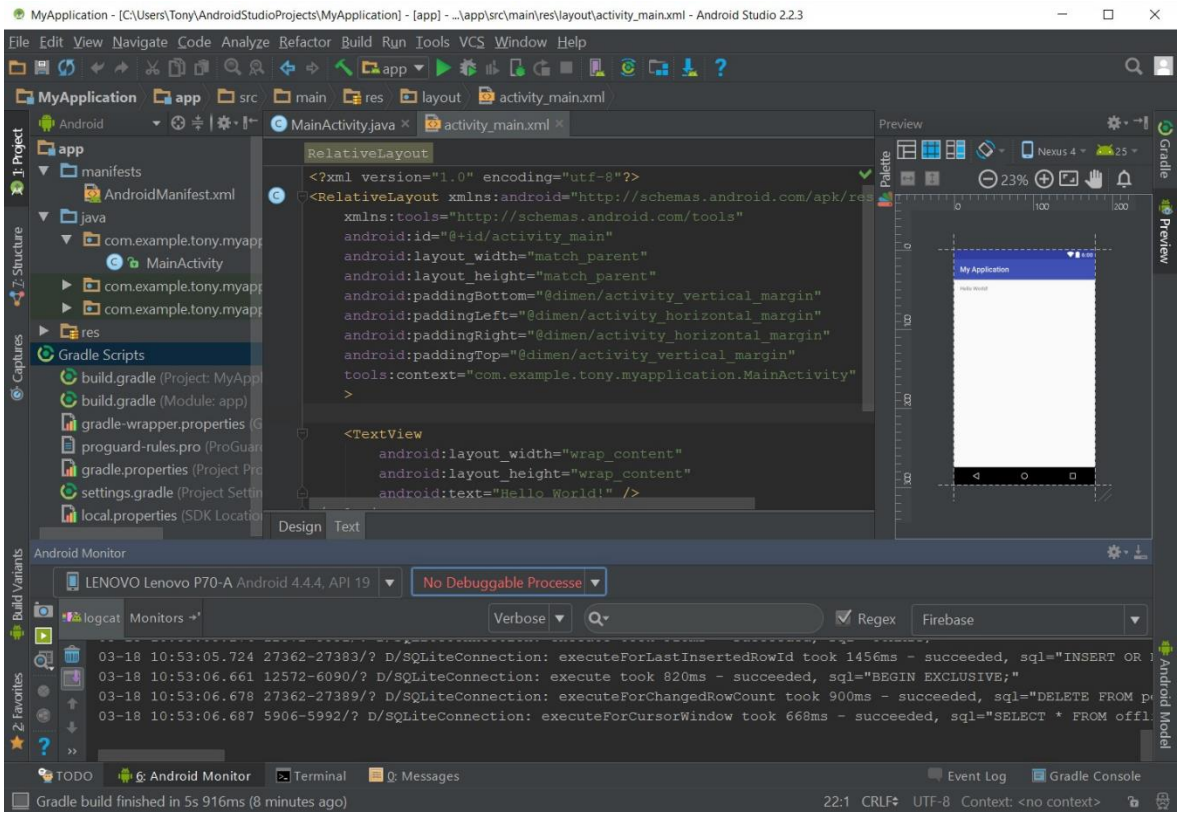


Obrázek 2: Vývojové prostředí Corona SDK (zdroj: vlastní)

1.3.3 ANDROID STUDIO

Jedná se o oficiální vývojový nástroj pro Android aplikace a s tím je spojena jedna výhoda. Některé novinky se mohou v tomto prostředí projevit dříve než v ostatních prostředích. Tuto platformu pro vývoj Android aplikací poprvé představila společnost Google v květnu roku 2013 na konferenci Google I/O. Tento nástroj je k dispozici v jedné edici, která je

každému volně dostupná jak pro operační systémy Windows, tak i Linux a MAC OS. Současná verze je 2.3.0.8, která byla vydána v březnu 2017[11]. Samotný nástroj je napsaný v jazyce Java, stejně jako Eclipse a IntelliJ. Oproti IntelliJ je start Android Studia nepatrně rychlejší. Samotné prostředí působí přehledně a je vhodně uspořádané. Po přepnutí prostředí do tzv. Darcula módu se překvapivě objevily určité nedokonalosti. Temné prostředí je narušováno bílými plochami a určité části textu jsou hůře čitelné. Pro odstranění nedokonalostí stačí prostředí restartovat. Připojení vlastního zařízení pro testování aplikací proběhlo automaticky, bez jakéhokoli nastavování v prostředí.



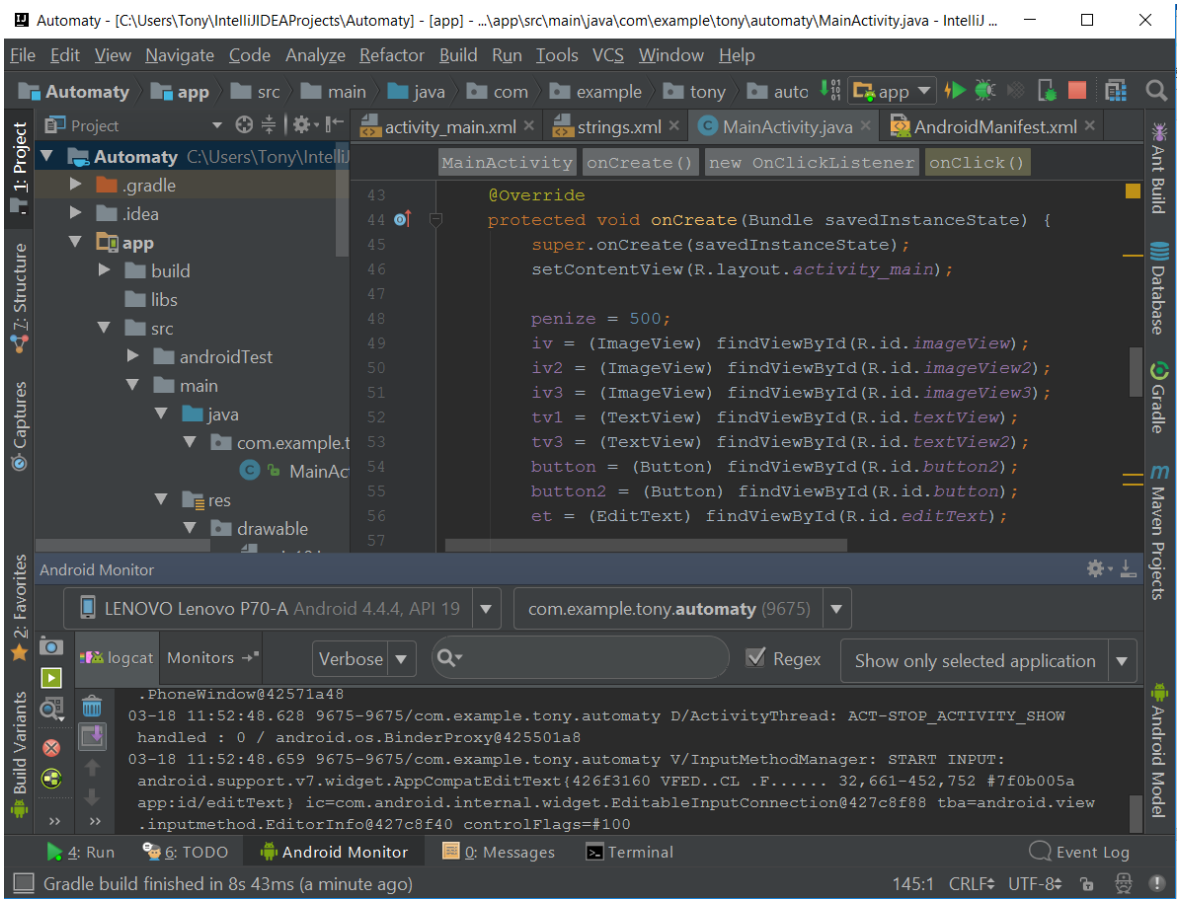
Obrázek 3: Vývojové prostředí Android Studio (zdroj: vlastní)

1.3.4 INTELLIJ

Vývojový nástroj IntelliJ je produktem společnosti JetBrains. Tato společnost již vyvinula vývojová prostředí pro mnoho jazyků. Konkrétně prostředí IntelliJ je primárně určeno pro vývoj Java aplikací. Za tento produkt již společnost získala během posledních 15 let více než 20 prestižních ocenění[12]. Při prvním spuštění je téměř k nerozeznání od Android Studia.

[11] Android Studio. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/studio/index.html
[12] Awards. *JET BRAINS* [online]. [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/company/press/awards.html#product=Intellij%20IDEA

Tudíž i toto prostředí je vhodně uspořádané. Především oceňuji i Darcula mód, který opravdu šetří oči. V současné době je k dispozici verze 2016.3.5., která byla uvolněna ke stažení 7. 3. 2017. Tato verze nepřinesla téměř nic nového, ale opravovala několik vyskytujících se chyb. Společnost JetBrains vydává aktuální verzi svého prostřední poměrně často. Za první dva měsíce roku 2017 již bylo poskytnuto 7 aktualizací produktu.



Obrázek 4: Vývojové prostředí IntelliJ (zdroj: vlastní)

1.4 KRITÉRIA PRO VÝBĚR OPTIMÁLNÍHO VÝVOJOVÉHO PROSTŘEDÍ PRO VÝVOJ ANDROID APLIKACÍ

Pokud chceme, aby byl vývoj aplikací co nejefektivnější, je vhodné si vybrat takový vývojový nástroj, který vyhovuje našim potřebám. Pro výběr je podstatné si sestavit seznam kritérií, která mohou vývoj usnadnit. Za tímto účelem je možné se nechat inspirovat několika porovnáními vývojových nástrojů, kde uživatelé hodnotí jejich výhody, ale také

nevýhody[13,14]. Je velmi pravděpodobné, že jedno prostředí nebude nejlepší ve všech kritériích, ale bude mít lepší výsledky v několika parametrech.

Sestavený seznam kritérií:

- intuitivita prostředí,
- kontrola v reálném čase,
- našeptávač – kontextová nápověda,
- možnosti jazyků,
- rychlost zpracování,
- stabilita,
- podpora,
- pluginy,
- licence.

Již po přečtení těchto stanovených kritérií je jasné, že jako nejméně vyhovující prostředí se pro vývoj Android aplikací může jevit Corona SDK. Prostředí má nevyhovující výsledky hned v prvních třech kritériích. Prostředí není intuitivní, kontrolu v reálném čase ani kontextovou nápovědu toto prostředí vůbec neobsahuje. Podstatnou nevýhodou je také potřebná placená licence pro rozšíření možností vývojového nástroje.

Na pomyslné předposlední místo bychom mohli zařadit vývojový nástroj Eclipse. Předností Eclipse je velké množství zásuvných modulů, které jsou dostupné bezplatně. Hlavní nevýhodou nástroje je ukončení jeho oficiální podpory pro vývoj Android aplikací. Pravděpodobně se budou objevovat nové zásuvné moduly pro vývoj Android aplikací, ale není jisté, zda budou stabilní. Jako další problém bych označil jeho horší GUI a menší intuitivitu ovládání nástroje.

[13] What are the best IDEs for Android development in Java? *Slant* [online]. [cit. 2017-03-19]. Dostupné z: https://www.slant.co/topics/4649/versus/~intellij-idea_vs_android-studio_vs_eclipse-android-development-tools-plugin

[14] SHARMA, Asankhaya. How do developers choose their IDE? *Linkedin* [online]. 2014 [cit. 2017-03-19]. Dostupné z: https://www.linkedin.com/pulse/20140522113335-16837833-how-do-developers-choose-their-ide

Zbývá již jen porovnání Android Studia a IntelliJ. Samotné Android Studio je založeno na IntelliJ. Ovšem Android Studio je zaměřeno výhradně na vývoj Android aplikací a je zcela zdarma. V mnoha kritériích jsou tato prostředí totožná. Hlavní výhodou IntelliJ je propracovanější kontextová nápověda. Výhoda Android Studia spočívá v oficiální podpoře společností Google a v jeho efektivnějším nastavení pro kompilaci Android aplikací. Tudíž pokud bude programátor tvořit výhradně Android aplikace, bude mu pravděpodobně více vyhovovat Android Studio, ale pokud někdy tvoří klasické desktopové aplikace, může mu práci více usnadnit vývojový nástroj IntelliJ.

2 PROSTŘEDÍ INTELLIJ

2.1 ZDŮVODNĚNÍ VÝBĚRU VÝVOJOVÉHO NÁSTROJE INTELLIJ

Hlavním důvodem pro výběr vývojového prostřední IntelliJ od JetBrains bylo, že velmi dobře splňuje dříve uvedená kritéria pro výběr optimálního vývojového prostředí. Dalším důvodem bylo to, že toto vývojové prostředí používá čím dál více programátorů. To dokládají informace na webu pypl.github.io/IDE.html, kde je toto prostřední aktuálně na 7. pozici v žebříčku oblíbenosti vývojových prostředí. V porovnání s některými jinými vývojovými prostředími zároveň vykazuje výrazněji se zvyšující tendenci v oblíbenosti jeho používání. Posledním zásadním důvodem bylo to, že tento produkt získal již mnoho ocenění, a to jak ze strany komerční, tak i díky hlasování samotných programátorů. Jedna z posledních cen byla udělena na 2015 Technology of the Year Awards[15].

2.2 ZÍSKÁNÍ VÝVOJOVÉHO NÁSTROJE INTELLIJ A JEHO INSTALACE

IntelliJ je distribuováno ve 2 verzích, a to Community a Ultimate. Verze Community je bezplatná a je určena především na vývoj Android aplikací v Javě. Verzi Ultimate lze získat ve formě 30-ti denního trialu. Verze Ultimate nabízí téměř veškeré služby, které lze v praxi využívat. Oproti základní verzi obsahuje navíc podporu JavaScriptu, databází a mnoha frameworků. Tuto verzi mohou získat studenti nebo učitelé bezplatně na rok, ale pouze pro vzdělávací účely. Pokud student či učitel vlastní školní e-mail, tak mu společnost zašle aktivační kód. Všechny verze lze stáhnout ze stránek www.jetbrains.com.

Společnost JetBrains v současnosti nabízí 12 prostředí pro práci s různými programovacími jazyky. Za zmínku stojí např. vývojové nástroje pro programování v PHP, které je společností nazýváno PhpStrom či PyCharm pro jazyk Python. Společnost se na trhu objevila poprvé v roce 2001, kdy otevřela svoji první pobočku a jediným produktem bylo prostředí IntelliJ. První pobočka se nacházela v Praze. V současnosti má společnost 6 poboček a poskytuje 20 produktů[16].

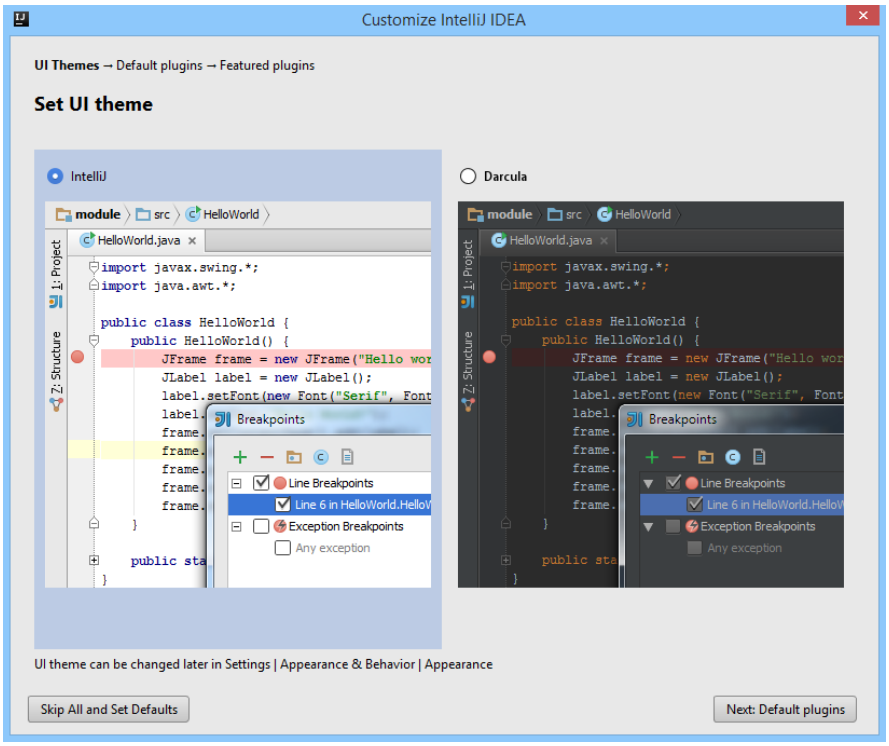
Prostředí je možné získat jak pro operační systém Windows, tak i pro Mac OS a Linux. Hardwarové nároky prostředí jsou minimální a v dnešní době je splňuje téměř každý

[15] InfoWorld's 2015 Technology of the Year Award winners. *InfoWorld* [online]. 2015 [cit. 2017-03-19]. Dostupné z: http://www.infoworld.com/article/2871935/application-development/infoworlds-2015-technology-of-the-year-award-winners.html?nsdr=true

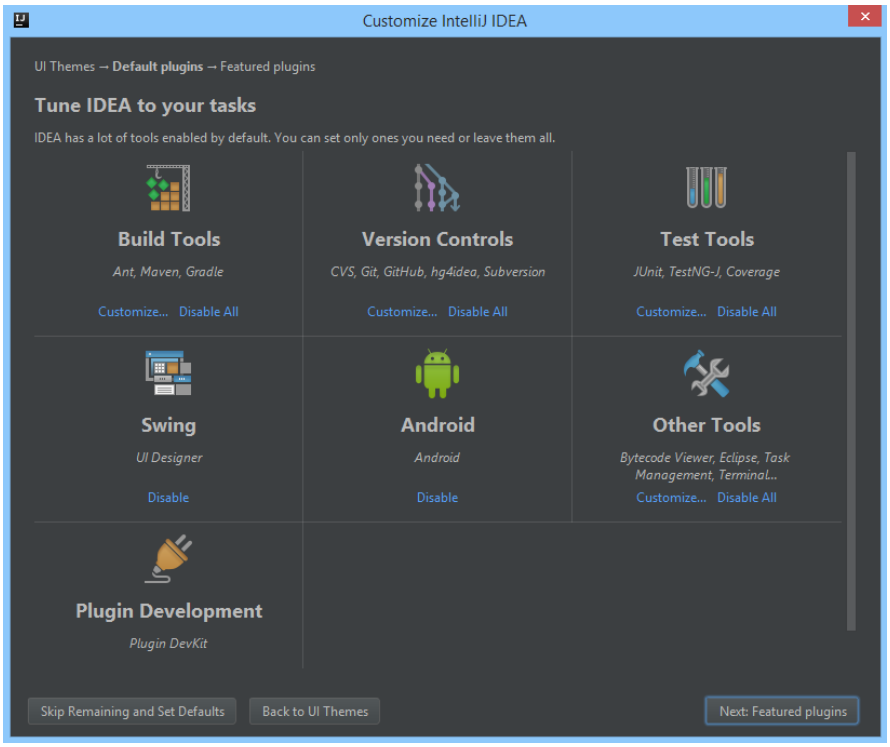
[16] *JetBrains* [online]. [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/

vyráběný PC. Verze Community má velikost 342 MB a při instalaci vyžaduje prostor o velikosti téměř 800 MB. Instalace prostředí trvá na běžném PC méně než 5 minut.

Při prvním spuštění aplikace je uživatel dotázán, zda nevlastnil předchozí verzi prostředí. Pokud ano, je uživateli umožněno, aby určil složku, ze které má nová verze převzít předchozí konfiguraci prostředí. V dalším kroku se zobrazí podmínky použití softwaru, které uživatel musí potvrdit. Následně si může uživatel vybrat verzi UI, základní pluginy, které bude využívat, a také i rozšiřující pluginy. Po dokončení tohoto průvodního nastavení se již spustí prostředí a je připraveno k použití. Pokud ovšem chceme vyvíjet aplikace pro Android, musíme mít v PC nainstalované JDK a SDK.



Obrázek 5: Výběr uživatelského rozhraní (zdroj: vlastní)



Obrázek 6: Nastavení zásuvných modulů (zdroj: vlastní)

JDK

Jedná se o balíček základních nástrojů pro vývoj aplikací. Součástí je například překladač kódu z Javy do bajtkódu, debugger pro ladění aplikací a JRE neboli Java Runtime Enviroment, tj. virtuální stroj pro spouštění Java aplikací. Balíček lze získat na stránkách Oracle – www.oracle.com

SDK

SDK představuje sadu nástrojů potřebných k vytváření a testování aplikací pro Android. Sada má 2 části. První část obsahuje základní nástroje a druhá komponenty pro konkrétní verzi Android. Pro udržování aktuálního SDK slouží aplikace Android SDK Manager, díky níž můžeme stahovat aktuální platformy a následně je pro určené verze aplikace vyvíjet a testovat. Nová verze Android SDK vychází s aktualizacemi systému. Na základě toho může vývojář nebo vývojářský tým vždy přizpůsobit aplikaci aktuální verzi či využít něco nového, co bylo aktualizací přidáno. Sada nástrojů je volně ke stažení na stránkách www.developer.android.com.

2.3 VLASTNOSTI GUI

Grafické rozhraní vývojového prostřední je velmi přehledné a intuitivní. Zejména propracované je členění prostředí. Programátor si může prostředí téměř libovolně

upravovat. Ve výchozím nastavení je na levé straně struktura projektu, uprostřed okno pro kód otevřeného souboru, a ve spodní části obrazovky je vyhrazena část pro vypisování událostí. Všem těmto částem lze změnit velikost, či je úplně skrýt. Jediné pevné části tvoří spodní lišta a pás karet hlavní nabídky, pod nímž je zobrazena cesta k otevřenému souboru k projektu. Největší vliv na zobrazení částí projektu má tlačítko umístěné na spodní liště. Po jeho stisknutí se na bočních stranách a spodní straně zobrazí tlačítka pro zobrazení různých částí projektu. Po stisknutí tlačítek se zobrazí požadované podokno. Většinou se jedná o skupinu tlačítek, která sdílejí jedno podokno. V podokně se zobrazí vybraná položka, případně i více vybraných. Díky tomu uživatel nemusí dlouze hledat v hlavní nabídce, kde je možnost si danou část zobrazit, a tím šetří čas při vývoji aplikací. Samotná část pro strukturu projektu je přehledně uspořádaná a části projektu jsou od sebe snadno rozeznatelné. Podstatné části aplikace mají různé ikony.

Velmi dobře je také propracované podokno pro psaní zdrojového kódu. Barevně jsou zde odděleny skupiny klíčových slov. Pokud má uživatel zapnutý Darcula mód, tak jsou klíčová slova jazyka zobrazována oranžovou barvou, proměnné fialovou, názvy metod okrovou barvou, anotace žlutou barvou a ostatní kód šedomodrou barvou.

Prostředí také obsahuje funkci pro zobrazení kódu dalších částí otevřeného souboru. Funkce je součástí posuvné lišty. Programátor si na běžném displeji pohodlně zobrazí maximálně 50 řádek kódu, ovšem pokud v prostředí IntelliJ najede myší na posuvnou lištu, tak se zobrazí podokno s 10 řádky kódu, který se nachází na místě, kde je umístěna myš na liště. Snadnými pohyby si uživatel může prohlédnout část kódu, kterou má napsanou na jiném místě v souboru. Nejen že se zobrazí kód z jiné části, ale jsou tam zobrazena také doporučení pro provedení úprav.

3 FUNKCE INTELLIJ A ANDROID APLIKACE

Vývojový nástroj obsahuje mnoho využitelných funkcí. Některé z nich jsou patrné již při vývoji aplikací a další slouží jako kontrola kódu, která programátorovi umožňuje ladit svůj kód. Již při vývoji jednoduché aplikace programátor zjistí, jak je IntelliJ o mnoho propracovanější než některá jiná vývojová prostředí, jako např. Eclipse. To je ovšem samozřejmě dáno tím, že IntelliJ i Android studio jsou přímo připravené na vytváření Android aplikací. Při práci v IntelliJ je k dispozici více než 120 klávesových zkratek, avšak asi nebude mnoho programátorů, kteří by byli schopni si takové množství pamatovat. Běžný programátor v praxi využije přibližně 20 klávesových zkratek.

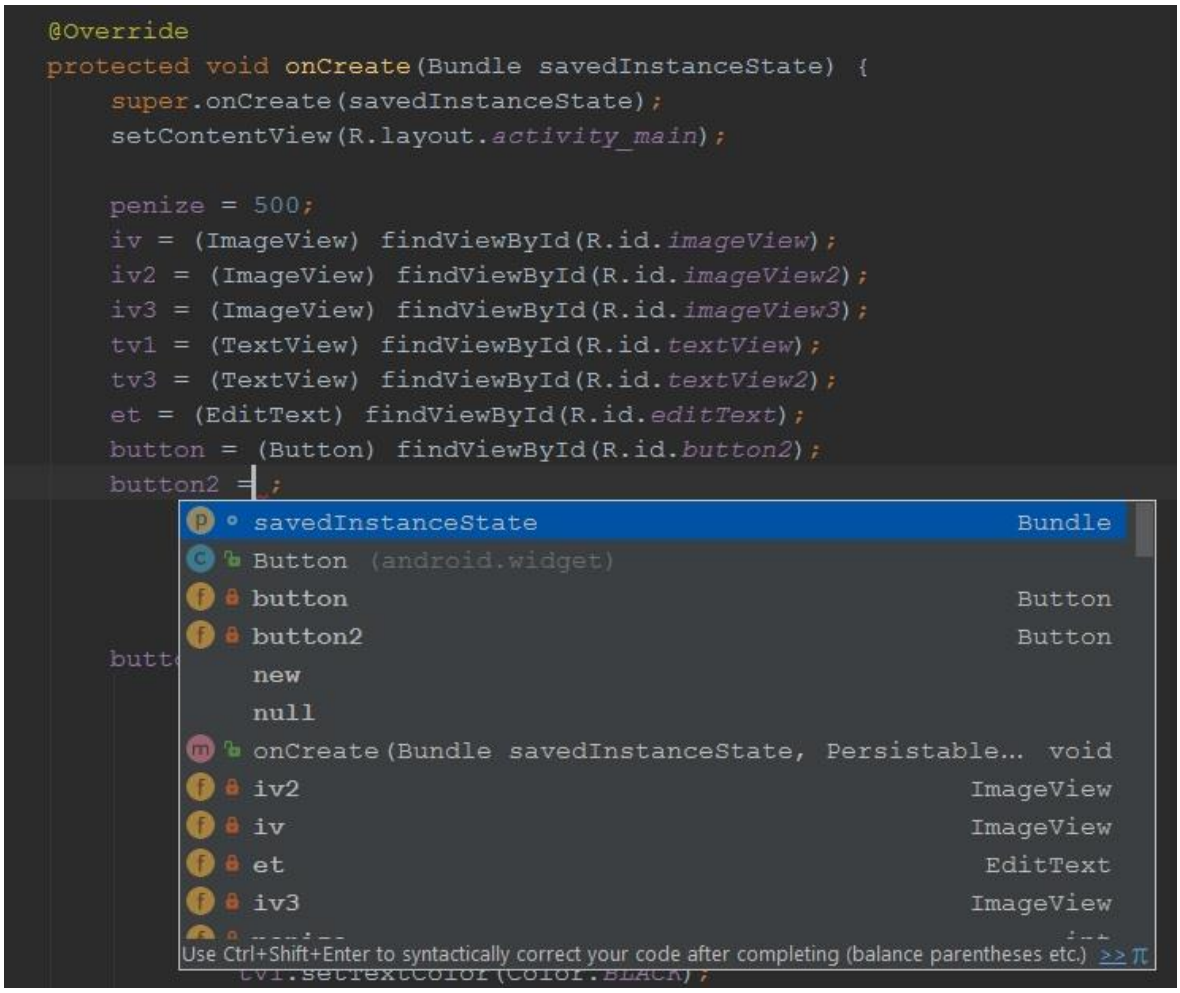
3.1 ZÁKLADNÍ KONTEXTOVÁ NÁPOVĚDA

Tato nápověda je vyvolána automaticky nebo klávesovou zkratkou Ctrl + mezerník. Pokud programátor nestiskne příslušnou klávesovou zkratku, tak je mu automaticky poskytnuta nápověda k dokončování názvů tříd, metod a klíčových slov. Pokud ovšem programátor příslušnou klávesovou zkratku použije, IntelliJ analyzuje kontext a navrhne možnosti dostupné v dané části programu. Další změna nastane, pokud je uvedená klávesová zkratka stisknuta podruhé. V takovém případě se zobrazí názvy tříd dostupné skrze moduly. Změna nastane i po třetím použití této klávesové zkratky. V tuto chvíli už jsou nabízeny všechny třídy v projektu bez ohledu na závislosti[17]. Následně jsou tedy doplňovány názvy tříd, rozhraní a po potvrzení jsou třídy i importovány.

PŘÍKLAD POUŽITÍ FUNKCE

Funkci základní kontextové nápovědy lze vidět na následujícím obrázku. V této části kódu dochází k definici proměnné. Základní kontextová nápověda ovšem nenabízí požadované řešení.

[17] Basic Code Completion. Completing Names and Keywords. *JetBrains* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/help/idea/2016.3/basic-code-completion-completing-names-and-keywords.html



Obrázek 7: Ukázka funkce základní kontextové nápovědy (zdroj: vlastní)

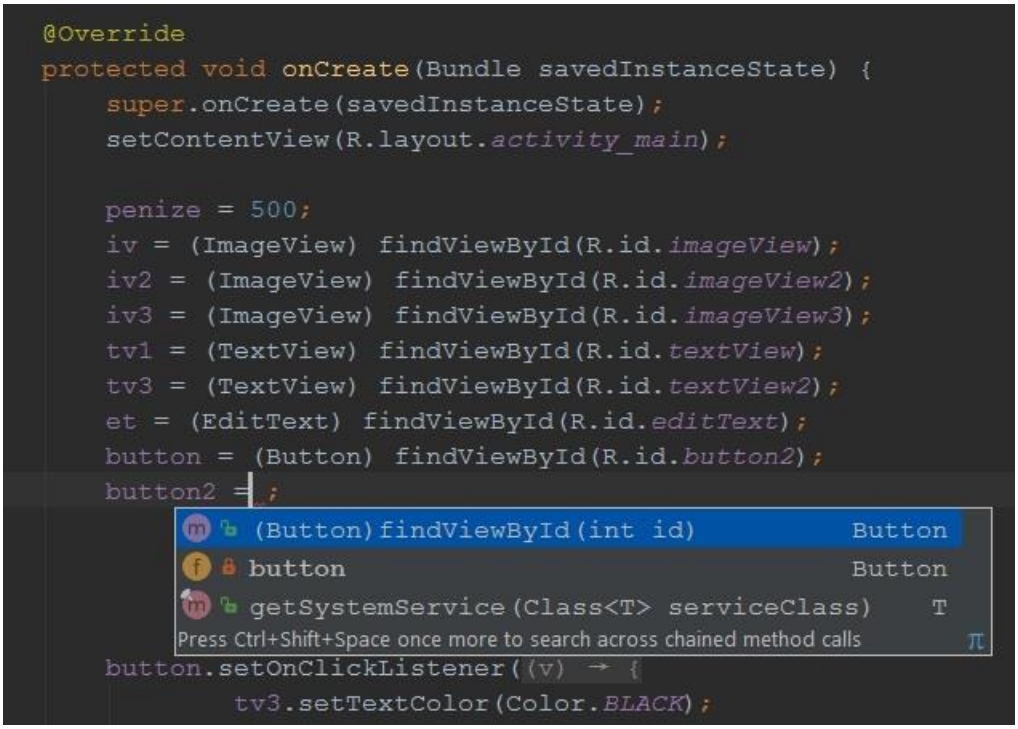
3.2 CHYTRÁ KONTEXTOVÁ NÁPOVĚDA

Pro použití této nápovědy je zapotřebí použít klávesovou zkratka Ctrl + Shift + mezerník. Chytrá nápověda filtruje seznam návrhů a zobrazuje pouze ty, které se vztahují k aktuálnímu kontextu. Využití této nápovědy se doporučuje v pěti základních případech. Prvním případem je použití přiřazení na pravé straně (obecně řečeno za =), druhý případ je při inicializaci proměnných, třetí případ při deklaraci návratových hodnot, předposlední případ pro zobrazení parametrů volané metody a poslední po deklaraci nového klíčového slova v deklaraci objektu[18].

[18] Smart Type Code Completion. Completing Code Based on Type Information. *JetBrains* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/help/idea/2016.3/smart-type-code-completion-completing-code-based-on-type-information.html

PŘÍKLAD POUŽITÍ FUNKCE

Pro příklad ukázání funkce chytré kontextové nápovědy byla vybrána identická situace jako v předchozím případě. Chytrá kontextová nápověda nabízí mnohonásobně kratší seznam vhodného doplnění kódu. Ze stovek položek byl seznam zkrácen na 3 položky.



Obrázek 8: Ukázka funkce chytré kontextové nápovědy (zdroj: vlastní)

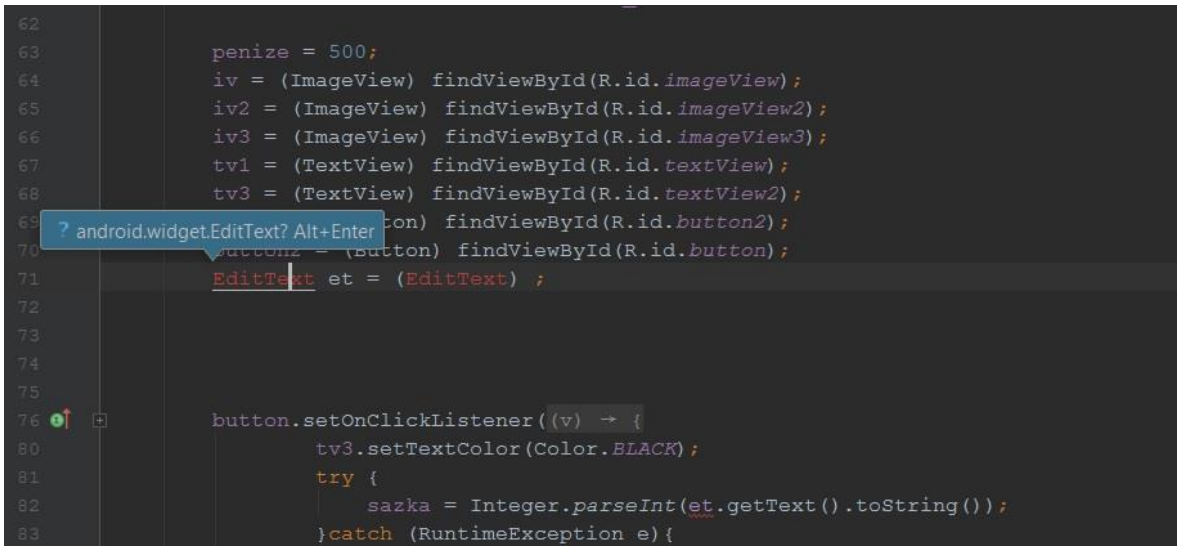
3.3 VYTVÁŘENÍ IMPORTŮ

Importování slouží k získání určité definice nějakého prvku, nebo k možnosti využití knihovních tříd, kde získáme metody, které už jsou v Javě obsaženy a nemusíme je sami implementovat. Metody, které Java již implementovány má, jsou optimalizované pro co největší výkon. Importované knihovní třídy jsou v každé třídě uváděné nad deklarací třídy. Ovšem programátor nemusí importovat jen jednu třídu, ale může importovat celý balíček. Jeho obsahem může být i více než 100 knihovních tříd. V IntelliJ se postupně importují knihovní třídy, po pátém importu z identické třídy se však definice importu změní z jednotlivých tříd na import balíčku. Pro importování knihovních tříd a balíčků je v IntelliJ určena klávesová zkratka Alt + Enter. Pokud tedy píšeme kód a chceme vložit nějaký prvek z knihovní třídy, tak se při psaní jeho názvu zobrazí prvek v automatické kontextové nápovědě, a pokud potvrdíme nějaký návrh z kontextové nápovědy, tak se automaticky provede importování. Pokud ovšem napíšeme kompletní název, tak se import neprovede.

Poté je vhodné použít klávesovou zkratku, která importování následně provede. Další možností je, že existuje definice prvku ve více knihovních třídách. Pokud ano, tak je po použití klávesové zkratky zobrazen seznam možných importů.

PŘÍKLAD POUŽITÍ FUNKCE

Ukázka funkce pro importování je zobrazena na následujícím obrázku v příkladu, kdy chce programátor využít grafického prvku EditText, ale vývojové prostředí hlásí chybu. Daný prvek nelze použít, dokud nebude naimportována příslušná knihovna. Po stisku klávesové zkratky se knihovna objeví mezi ostatními importy a lze ji plně využívat.



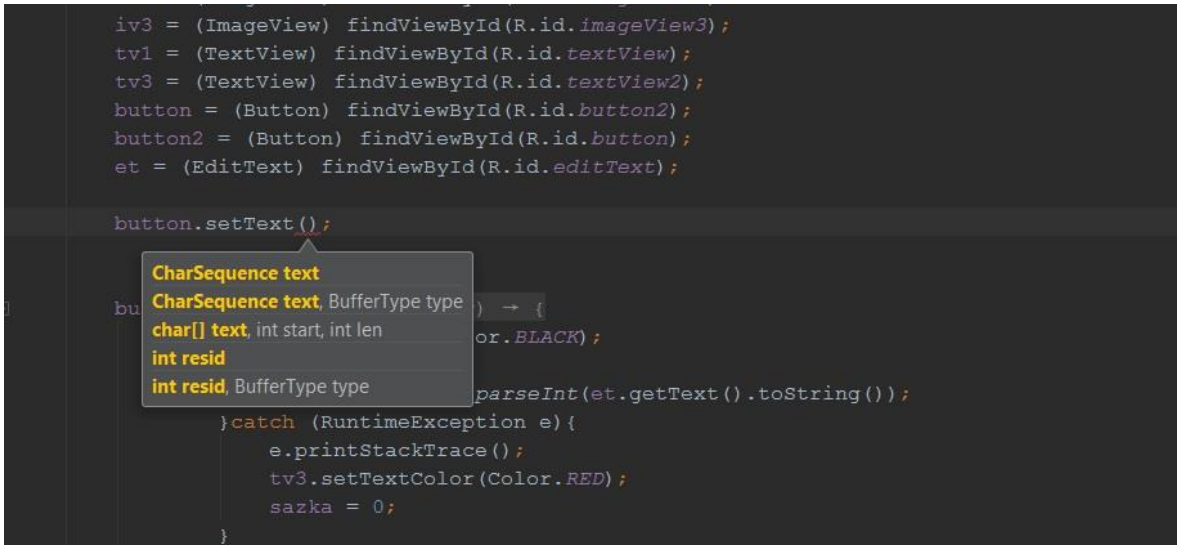
Obrázek 9: Ukázka funkce pro vytváření importů (zdroj: vlastní)

3.4 ZOBRAZOVÁNÍ PARAMETRŮ METOD

Téměř každá instance objektu má na sebe navázány určité metody. Těchto metod je v rámci jedné třídy obvykle mnoho, některé mají stejný název a jsou tzv. přetížené. To znamená, že daná metoda má více definic. Definice se může lišit počtem parametrů, jejich typem či pořadím. Pro zobrazení parametrů je použita klávesová zkratka Ctrl + p. Funkčnost je obdobná jako v předchozím případě. Pokud necháme doplnit název metody kontextovou nápovědu, tak se seznam parametrů automaticky zobrazí, ale pokud v danou chvíli metodu opustíme, tak se při vrácení na dané místo parametry v kódu nezobrazí. Poté je tedy můžeme vyvolat pomocí klávesové zkratky, která parametry zobrazí pro všechny přetížené metody.

PŘÍKLAD POUŽITÍ FUNKCE

V Javě existuje mnoho přetížených metod. Například pro definování textu na tlačítku je pět možností. Tyto definice si ovšem nemusí programátor pamatovat, tudíž mu funkce pro zobrazení definic metod toto umožňuje. Popsaný příklad je vidět na následujícím obrázku.

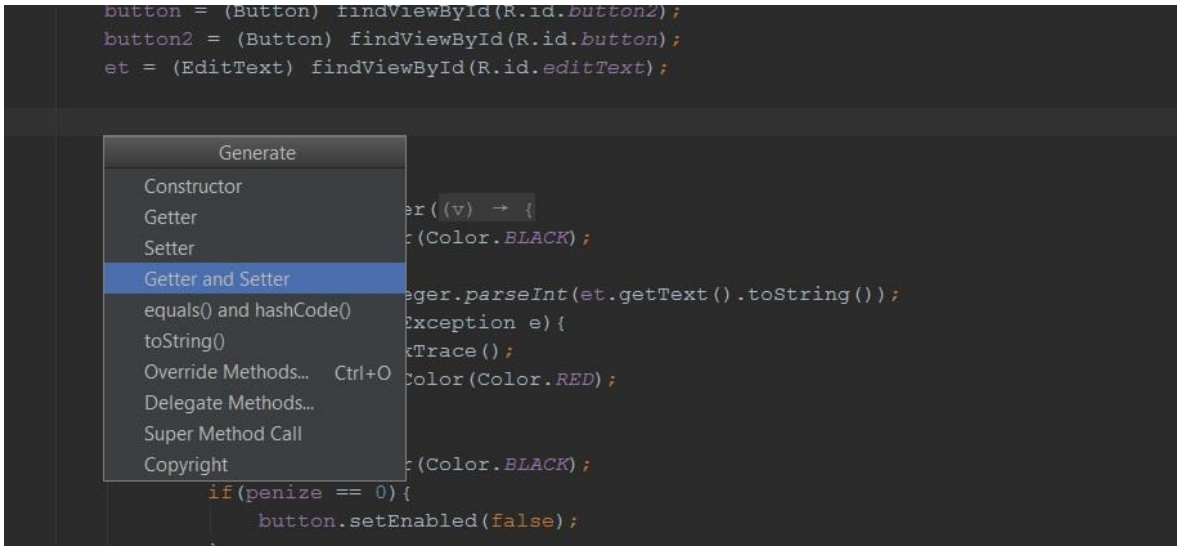


Obrázek 10: Ukázka funkce pro zobrazování parametrů metod (zdroj: vlastní)

3.5 GENEROVÁNÍ KÓDU

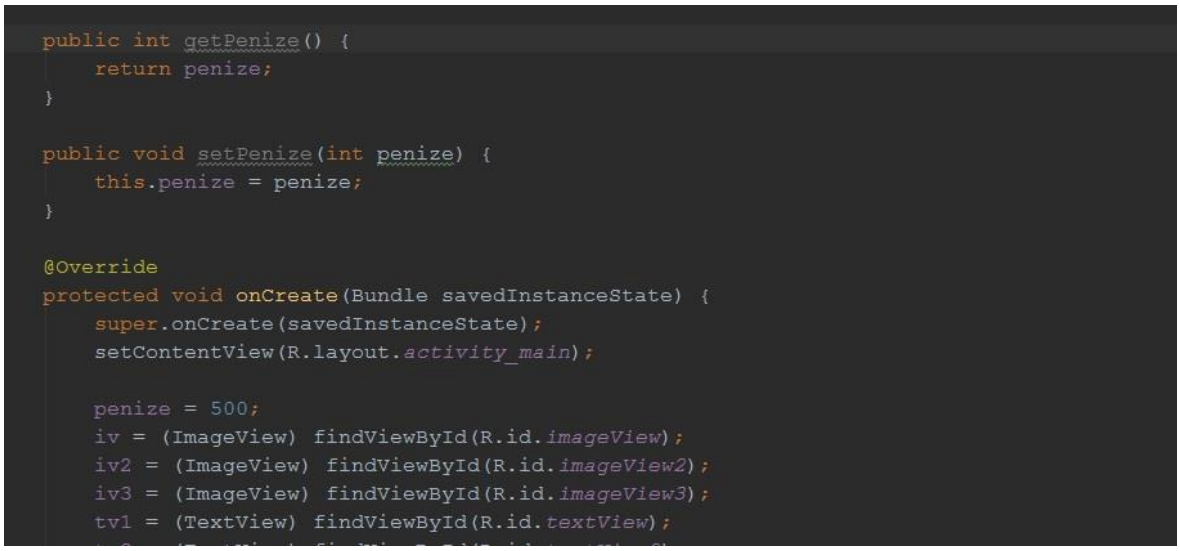
Vývojový nástroj obsahuje i generování základních metod, jako např. konstruktor (metoda, která se označuje stejně jako název třídy a je vyvolána při vytváření instance), getry a setry, toString a překrývané metody. K vyvolání nabídky pro generování je určena klávesová zkratka Alt + Insert. Tyto generované metody, zejména getry a setry, mívají základní tvar, který se nemění. Změnit se může program obsažený v těchto metodách. Generování slouží zejména k urychlení práce, protože psaní getrů a setrů by pro mnohé proměnné bylo časově náročné. Velmi užitečná je ovšem i funkce pro překrývané metody. Pokud by si programátor myslel, že danou metodu překryl, ale špatně napsal její název, tak by program nefungoval a taková chyba se pak obtížně hledá. IntelliJ nám přímo pomůže danou metodu najít. Pro generování překrývané metody slouží i vlastní klávesová zkratka Ctrl + o.

PŘÍKLAD POUŽITÍ FUNKCE



Obrázek 11: Ukázka funkce pro generování kódu, část 1 (zdroj: vlastní)

Na předchozím obrázku je vidět, co se uživateli zobrazí po použití klávesové zkratky Alt + Insert. Při vyvolání této nabídky nezáleží na tom, kde se programátor v kódu nachází. Potřebný kód je vygenerován a umístěn na místo, kam patří. Pokud je tedy vybráno generování kódu pro getry a setry, tak následně programátor vybere, pro kterou proměnnou chce daný kód získat. Na následujícím obrázku je vidět vygenerovaný kód.



Obrázek 12: Ukázka funkce pro generování kódu, část 2 (zdroj: vlastní)

3.6 ROZŠÍŘENÝ VÝBĚR

Pokud chceme označit určitou část kódu, tak stačí umístit na dané místo kurzor a opakovaným stisknutím klávesové zkratky se označená část zvětšuje. Rozšíření postupuje po uzlech. Pro tuto funkci je defaultně nastavena klávesová zkratka Ctrl + w.

PŘÍKLAD POUŽITÍ FUNKCE

Na následujících dvou obrázcích je vidět funkce rozšířeného výběru. Rozdí mezi obrázky ukazuje jedno použití klávesové zkratky, aby bylo patrné rozšiřování se označení po uzlech. Postupně je možné označit celou metodu, nebo dokonce obsah celého souboru.

```
tv1.setTextColor(Color.BLACK);
if(penize == 0){
    button.setEnabled(false);
}
if(sazka > penize){
    tv1.setTextColor(Color.RED);
}else {
    vyberObrazekMice(iv);
    vyberObrazekMice(iv2);
    vyberObrazekMice(iv3);
    if (iv.getId() == iv2.getId() && iv2.getId() == iv3.getId()) {
        penize = penize + sazka * 5;
        tv1.setText(format("%s%d", new Object[]{"Kredit: ", penize}));
    } else {
        penize = penize - sazka;
        tv1.setText(format("%s%d", new Object[]{"Kredit: ", penize}));
    }
}
```

Obrázek 13: Ukázka funkce pro rozšířený výběr, část 1 (zdroj: vlastní)

```
tv1.setTextColor(Color.BLACK);
if(penize == 0){
    button.setEnabled(false);
}
if(sazka > penize){
    tv1.setTextColor(Color.RED);
}else {
    vyberObrazekMice(iv);
    vyberObrazekMice(iv2);
    vyberObrazekMice(iv3);
    if (iv.getId() == iv2.getId() && iv2.getId() == iv3.getId()) {
        penize = penize + sazka * 5;
        tv1.setText(format("%s%d", new Object[]{"Kredit: ", penize}));
    } else {
        penize = penize - sazka;
        tv1.setText(format("%s%d", new Object[]{"Kredit: ", penize}));
    }
}
```

Obrázek 14: Ukázka funkce pro rozšířený výběr, část 2 (zdroj: vlastní)

3.7 ZMĚNA NÁZVU

Jedná se o funkci, která pracuje v rámci refaktoringu. Pokud programátor nevhodně zvolí název nějaké proměnné nebo metody, tak by bylo náročné danou proměnnou či metodu hledat v celém projektu a všude upravovat její název. Proto byla tato funkce navržena tak, že se po stisknutí příslušné klávesové zkratkyumožní editace názvu a po jeho potvrzení se název automaticky změní v celém projektu. Klávesová zkratka vedoucí k této funkci je Shift

+ F6. Před vyvoláním funkce musí být kurzor umístěn na místo názvu proměnné nebo metody.

PŘÍKLAD POUŽITÍ FUNKCE

Pokud programátor zvolí nevhodný název proměnné či metody, je možné ho změnit. Na následujícím obrázku je vidět červený rámeček, který označuje proměnnou, nad kterou byla vyvolána funkce pro změnu názvu. V danou chvíli je možné buď název upravit, nebo ho celý smazat a napsat nový. Poté, co je název upraven, tak se změna potvrdí tlačítkem Enter. Samotná funkce nabízí změnu názvu, který obsahuje pouze malá písmena.

```
tv1.setTextColor(Color.BLACK);
if(penize == 0){
    button.setEnabled(false);
}
if(sazka > penize){
    tv1.setTextColor(Color.RED);
}else {
    vyberObrazekMice(iv);
    vyberObrazekMice
    vyberobrazekmice
    Press Shift+F6 to show dialog with more options
                                    && iv2.getId() == iv3.getId()) {
        penize = penize + sazka * 5;
        tv1.setText(format("%s%d", new Object[]{"Kredit: ", penize}));
    } else {
        penize = penize - sazka;
        tv1.setText(format("%s%d", new Object[]{"Kredit: ", penize}));
    }
}
```

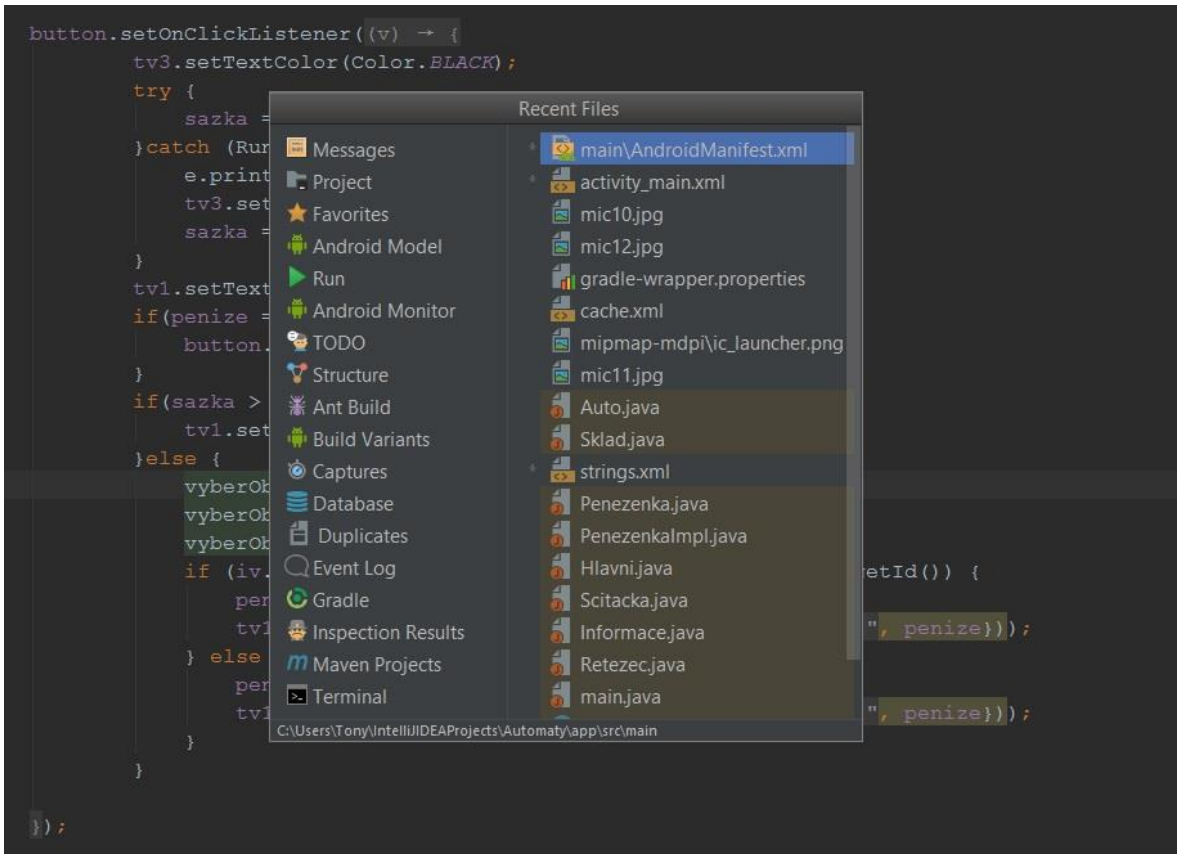
Obrázek 15: Ukázka funkce pro změnu názvu (zdroj: vlastní)

3.8 NEDÁVNÉ SOUBORY

Jedná se o funkci, která zobrazí nedávno otevřené soubory a umožní jejich opětovné otevření. Největší uplatnění bude mít v rozsáhlých projektech. V prostředí IntelliJ může být otevřeno současně maximálně deset oken, mezi nimiž lze přepínat. Ovšem paměť funkce sahá hlouběji a programátor si může snadněji najít a otevřít potřebný soubor. K tomuto slouží klávesová zkratka Ctrl + e. Pomocí této funkce si uživatel může také zobrazit určitou část grafického rozhraní, například strukturu projektu či terminál.

PŘÍKLAD POUŽITÍ FUNKCE

Funkci pro zobrazení nedávno otevřených souborů je možné vyvolat z jakéhokoli místa v projektu. Jak je vidět na níže vloženém obrázku, soubory, které se nenachází v otevřeném projektu, jsou zvýrazněny oranžovým podkresem.



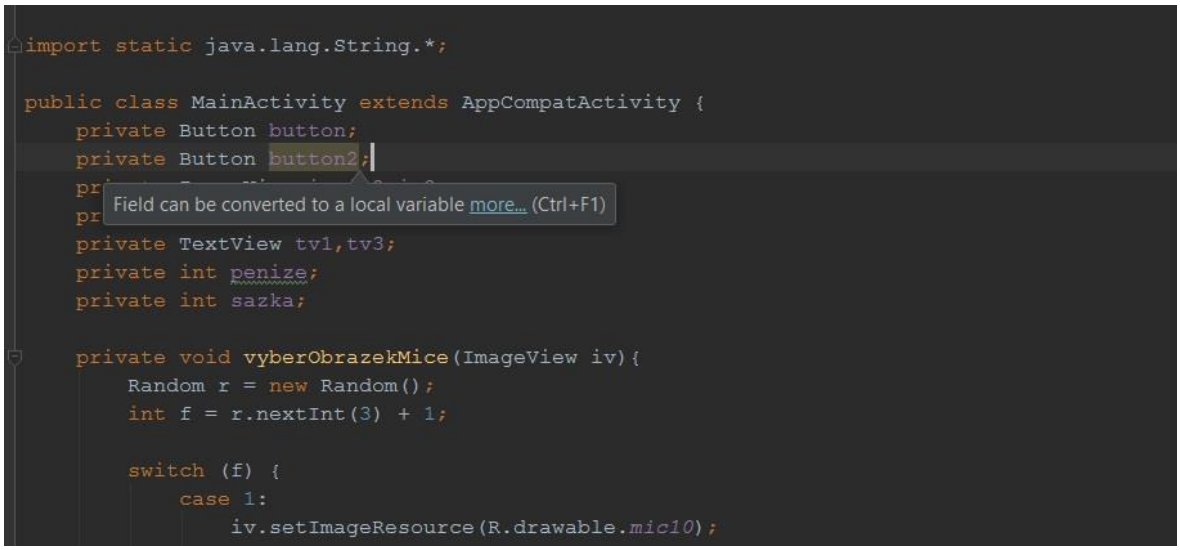
Obrázek 16: Ukázka funkce pro zobrazení nedávných souborů (zdroj: vlastní)

3.9 ZOBRAZENÍ CHYB A UPOZORNĚNÍ

Programátor při vytváření kódu nemusí vědět, zda některé položky bude dále potřebovat i v jiných třídách, a často nedokončí nějakou deklaraci úplně, nebo něco opomene. Ve vývojovém prostředí ovšem běží kontrola kódu v reálném čase. Pokud již programátor považuje danou část za dokončenou, tak se pomocí klávesové zkratky může pohybovat v kódu po částech vyhodnocených jako chyba, nebo obsahujících nějaké upozornění. K tomuto slouží klávesová zkratka F2, která však umožňuje pohyb jen v daném souboru.

PŘÍKLAD POUŽITÍ FUNKCE

Po stisknutí příslušné klávesové zkratky je kurzor přesunut na následující chybu či upozornění v kódu. Po najetí myší na označený prvek v kódu je zobrazen důvod, proč k označení došlo. Poté je již na programátorovi, zda doporučení přijme a provede vhodnou operaci či nikoli.



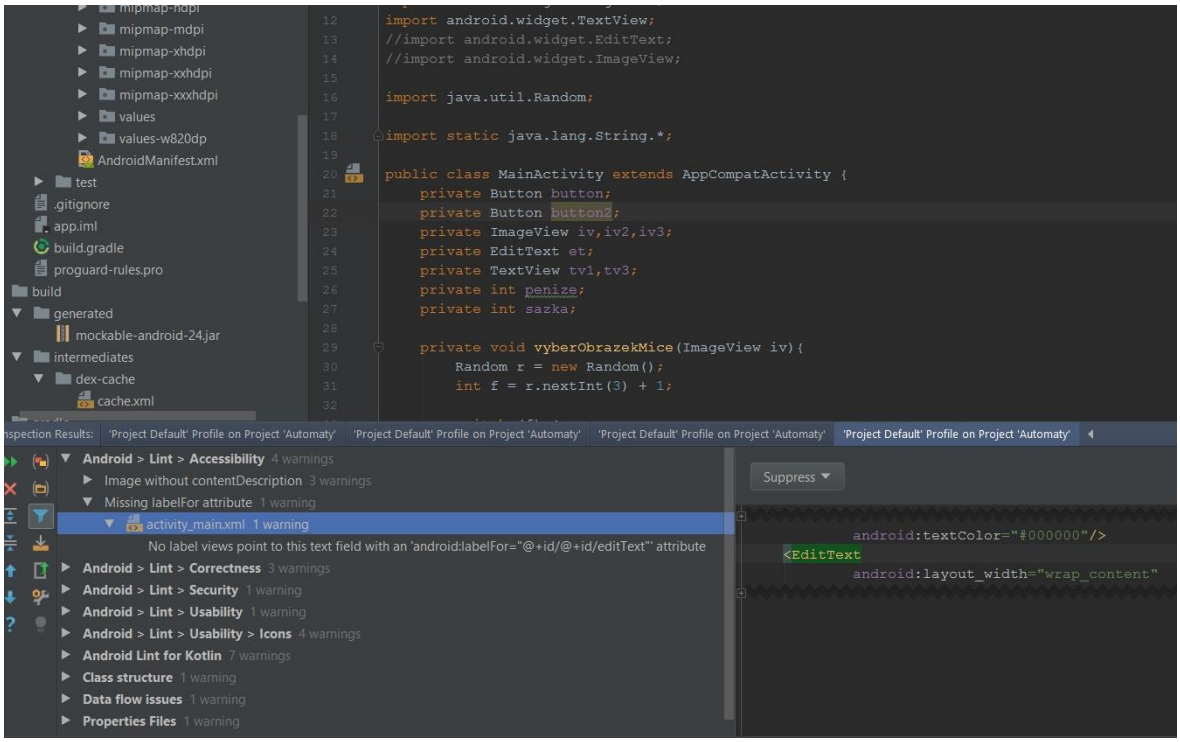
Obrázek 17: Ukázka funkce pro zobrazení chyb a upozornění (zdroj: vlastní)

3.10 KONTROLA KÓDU

Jedná se o nadstavbu předchozí funkce. Tato funkce kontroluje veškerý kód obsažený v projektu. Funkce je spuštěna přes položku v menu Analyze a následně Inspect Code. Zde potvrdíme, že chceme provést kontrolu celého projektu, a následně se zobrazí seznam chyb a upozornění. Stisknutím tlačítka myši na konkrétní položce seznamu se dostaneme na místo, kde se chyba nachází, a je zde i uvedeno, o jakou chybu se jedná. Tato funkce může být velmi užitečná a pomůže vylepšit programátorův kód. Nabízená funkce může být vhodná především pro začínající programátory, kteří ještě nemají s programováním pro Android potřebnou zkušenost.

PŘÍKLAD POUŽITÍ FUNKCE

Na následujícím obrázku je vidět, co se programátorovi po spuštění funkce pro kontrolu kódu zobrazí. V levé spodní části je zobrazeno rozbalovací menu s nalezenými nedostatky. Každý nedostatek obsahuje důvod, proč byl do seznamu zahrnut. Po přímém kliknutí na chybu je v pravé spodní části přímo zobrazena část kódu, kde se problém nachází. Ovšem dané podokno slouží pouze k náhledu a kód tam opravit nelze.



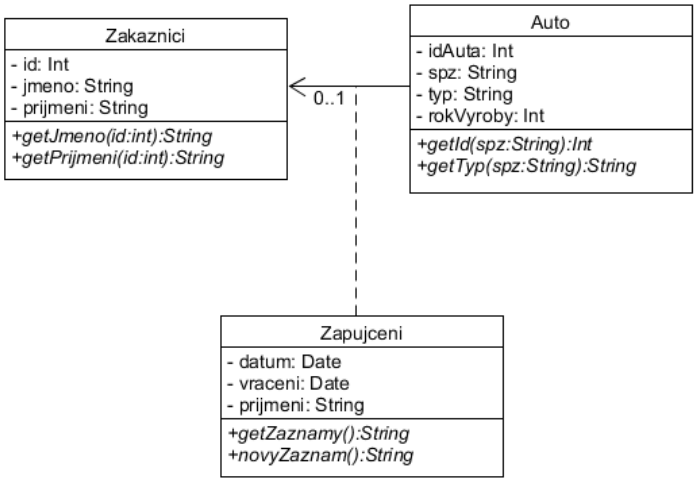
Obrázek 18: Ukázka funkce pro kontrolu kódu (zdroj: vlastní)

4 VHODNÝ POSTUP VÝVOJE

4.1 EFEKTIVNÍ VÝVOJ APLIKACE

Vývoj aplikací nezačíná tím, že rovnou začneme psát kód aplikace. Prvním krokem by mělo být zjištění, co vše aplikace musí umět a jak by měla vypadat. Dále také záleží na tom, zda bude aplikaci vyvíjet jeden programátor, nebo celý tým.

Pokud má programátor celkový obraz o aplikaci, tak by mohlo být prvním krokem sestavení UML diagramu. Diagram dobře ukazuje různé vazby v programu a mezi soubory (konkrétně v jazyce Java mezi třídami), do diagramu se také umisťují proměnné dané třídy a její metody. Jinými slovy jde o zjednodušený návrh celé aplikace. Pokud tedy danou aplikaci vytváří tým programátorů, musí být zajištěna dělba práce. UML diagram zajistí i konzistenci názvů proměnných a metod.



Obrázek 19: Ukázka vývojového diagramu (zdroj: vlastní)

4.1.1 VLASTNÍ NÁVRH POSTUPU PŘI VÝVOJI ANDROID APLIKACÍ

Samotný vývoj aplikací může být rozdělen do pěti částí. První fáze je **příprava**. Ta zahrnuje zjištění kompletních informací o cílové aplikaci. Programátor tedy během přípravy získá představu o tom, jak by měla aplikace vypadat, k čemu by měla sloužit a co všechno by měla poskytovat svému budoucímu uživateli. Tyto informace pak programátorovi pomohou objasnit postup samotného vývoje aplikace. Pro snadný vývoj může být vytvořen i algoritmus aplikace a neméně důležitou informací je, pro jaká zařízení bude aplikace určena.

Druhým krokem by mělo být **vytvoření uživatelského rozhraní aplikace**. Tvůrce určí základní komponenty, na které následně bude navazovat funkčnost programu. Při vkládání grafických komponent vzniká provázání s ostatními komponentami. Za účelem vytvoření grafického prostředí může uživatel využívat buď Drag and drop technologie, nebo formu zápisu XML kódu. V praxi se jeví jako nejvýhodnější kombinace obou možností. Např. nastavování pozic a umístění formulářových komponent v rámci formuláře aplikace nemusí končit vždy jen přetažením komponenty na vizuálně vhodnou pozici na formuláři. Pokud programátor použije nějaký prvek, který se bude překreslovat, např. ImageView, tak se při překreslení může jeho pozice změnit. Zásadním prvkem Android aplikace je také rozhodnutí, zda bude aplikace používána v orientaci portrait (na výšku), landscape (na šířku), nebo zda bude podporovat oba uvedené režimy. V praxi se většinou využívají všechny možnosti a finálně se vybere především ta, která se k dané aplikaci nejvíce hodí. Pokud programátor zvolí možnost třetí, tedy umožní obě orientace, bude tak mít dvojí práci s pozicováním formulářových komponent.

Třetím krokem při vývoji je **psaní samotného kódu aplikace**. Programátor může aplikaci psát jak v jazyce Java, tak v C++. Zde záleží na preferencích programátora, protože cíloví uživatelé nepoznají, jak je aplikace tvořena. Uživatelům jde pouze o její plnou funkčnost. Nejdříve je vhodné promyslet, jaké proměnné budou používány a následně deklarovány. Při vytvoření projektu se v třídě MainActivity (pokud není jméno při vytváření projektu změněno) automaticky vytvoří metoda OnCreate, kde programátor určuje, co se má při zapnutí aplikace stát. V této metodě je nastaveno počáteční zobrazení potřebných formulářových prvků (komponent) a běžně se zde nastavuje obsah proměnných. Dále je již vytvářen kód samotné aplikace.

Předposledním krokem je **testování vytvořené aplikace**. Nejdříve se testuje, jestli při zadaných datech aplikace uživateli vrátí očekávané výsledky. Pokud ano, tester se pak zaměří na činnosti, díky nimž se snaží docílit např. pádu aplikace. Příkladem mohou být špatná vstupní data, kdy místo číselných hodnot uživatel zadá text a aplikace musí adekvátně reagovat. Některým těmto případům lze předejít tak, že vybereme vhodný typ editačního pole pro vstupní data. V Android aplikaci můžeme využít aktuálně 12 typů editačního pole, například pole pro text, číslo či mailovou adresu.

Závěrem již stačí **aplikaci vyladit**, ošetřit chyby, které byly při testování zjištěny. Dále je také vhodné využít možnosti prostředí IntelliJ a nechat aplikaci analyzovat. Po analýze a zohlednění jejích výsledků již lze aplikaci považovat za dokončenou. Dokončená aplikace může být také umístěna na Google Play.

4.1.2 GRAFICKÉ PRVKY POUŽITÉ PŘI VÝVOJI APLIKACÍ

Při vývoji sady příkladů bylo použito několik grafických prvků. Použité prvky jsou TextView, EditText, Spinner, SeekBar, CheckBox, ImageView a Button. Parametry prvků lze vždy změnit třemi způsoby. První je nastavení grafického prvku pomocí zobrazeného panelu vlastností, druhou možností je upravení XML kódu a poslední je úprava pomocí kódu samotné aplikace.

TextView je grafický prvek, který slouží pro zobrazování textu. Tento prvek patří mezi ty nejzákladnější a nachází se v řadě aplikací. TextView je textový editor jako EditText, ale je nakonfigurován tak, aby editace nebyla možná. Hlavní metodou tohoto prvku je tedy setText()[19]. Programátor může tomuto prvku nastavit, aby si uživatel mohl zobrazený text uložit do schránky.

Prvkem pro vstup dat je EditText. Tento prvek má 14 variant. Podle varianty je uživateli umožněno vkládání dat. Prvním typem je PlainText, který slouží pro zadávání textu. Druhým typem je Password, který se dále dělí na klasické textové nebo číselné heslo. Při použití tohoto prvku se zadaný znak zobrazí na krátký čas a pak je nahrazen puntíkem. Dalším typem je E-mail, který slouží k zadání e-mailové adresy. Ostatními typy jsou Phone (telefonní číslo), Postal Address (poštovní adresa), MultilineText (víceřádkový text), Time (čas), Date (datum), Number (číslo), Number (Signed) – číslo se znaménkem, Number (Decimal) – číslo desetinné, AutoCompleteTextView a MultiAutoCompleteTextView. Na první pohled jsou tyto prvky od sebe k nerozeznání, změna se projeví, pokud chce uživatel zadat text. Podle prvku je mu zobrazena příslušná varianta klávesnice, která usnadňuje zadání vhodných dat. Někdy je zobrazena i klávesnice, u které nelze použít všechny zobrazené prvky. Příkladem je klávesnice pro zadávání čísel. Pokud použijeme typ pro celá kladná čísla, nelze použít tlačítka pro znak mínus a desetinnou čárku. Při použití typu Time je uživateli zobrazena číselná klávesnice doplněná o dvojtečku. Pokud je použit typ E-mail,

[19] TextView. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/reference/android/widget/TextView.html

tak se uživatel nemusí proklikávat do nabídky speciálních znaků a zavináč je mu zobrazen mezi tlačítky písmen. Některé prvky tedy úplně eliminují zadání špatných vstupních hodnot a programátor nemusí kontrolovat správnost vstupních dat. Pokud ovšem nelze zadání špatných hodnot zamezit, programátor musí využít například výjimek, danou chybu odchytit a adekvátně na ni zareagovat.

Pro výběr ze sady hodnot se používá komponenta nazvaná Spinner. Programátor vytvoří sadu hodnot, z níž si uživatel může vybrat. Jedna z hodnot je výchozí a další se zobrazí po stisknutí prvku. Rozbalovací menu poznáme podle šipky orientované směrem dolů na konci prvku. Spinner lze využít jak pro získání konečné hodnoty, tak vybranou hodnotu použít jako rozhodovací prvek[20].

Dalším použitým prvkem byl SeekBar, který představuje rozšíření prvku ProgressBar. Prvek představuje úsečku s bodem, který lze táhnout, a tím je nastavena hodnota. Při deklaraci prvku je nutné nastavit, jaké hodnota odpovídá maximu. Tento prvek je téměř vždy doplněn prvkem TextView, kde je uživatel informován, jakou hodnotu nastavil. Pro zobrazení aktuální nastavené hodnoty slouží překrytí metody onStopTrackingTouch(). Dané metodě určíme, že po skončení posunu má na prvku TextView aktualizovat text, a tím informovat uživatele o úrovni nastavení hodnoty.

Mezi prvky použité v aplikacích vytvořené sady také patří CheckBox, který může nabývat pouze dvou hodnot. CheckBox je zatržený či nikoli, pak se tedy jedná o prvek typu boolean a návratová hodnota je true nebo false. Pro zjištění aktuálního stavu slouží metoda isChecked(). Příkladem použití uvedeného prvku může být otázka, zda chce uživatel zasílat nějaké informace.

Dalším použitým prvkem v aplikacích vytvořených v rámci této práce je prvek ImageView, prvek sloužící k zobrazování obrázků. Zobrazovaný obrázek se nemusí nacházet přímo v projektu, ale může se načítat z různých zdrojů. Zvolený obrázek ovšem nemusí mít konečnou podobu, existuje mnoho metod, kterými může být obrázek upraven.

[20] Spinners. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/guide/topics/ui/controls/spinner.html

Programátor může upravit nejen velikost a pootočení obrázku, ale může nastavit i jeho průhlednost, rámeček a mnoho dalšího[21].

Jedním z použitých formulářových prvků vytvořené vlastní sady aplikací byl Button. Jedná se tedy o tlačítko, jeden z nejdůležitějších prvků vůbec. Tento prvek se nachází v téměř každé aplikaci a po jeho stisknutí je vyvolána metoda onClick(), kde programátor určuje, co se má stát. Metoda onClick() není jediná, která může být vyvolána. Při stisku a podržení tlačítka dochází k vyvolání metody onLongClick(). Tyto metody často slouží ke zpracování získaných vstupních dat a podle nich aplikace reaguje. Uvedené metody patří do rozhraní View.onClickListener. Další metody obsahuje rozhraní View.onTouchListener, kde může programátor definovat akce například po uvolnění tlačítka.

Pokud nějaký prvek do aplikace vložíme, bude jeho design upraven podle výchozího nastavení daného zařízení. Design také závisí na aktuálně nainstalované verzi systému Android. Proto lze říci, že prvky budou na různých zařízeních zobrazovány jinak.

4.1.3 VYZKOUŠENÍ POSTUPU VÝVOJE NA KONKRÉTNÍ APLIKACI

Postup bude aplikován na aplikaci Automaty s tímto zadáním: Vytvořte aplikaci, kde bude mít uživatel určitý kredit, který bude moct postupně sázet. Zda vyhraje další kredit, rozhodne náhodné generování ze tří obrázků. Počet okének automatu bude také tři. Pokud se vygenerují tři stejné obrázky, uživatel získá pětinásobek vsazené částky.

V aplikaci bude potřeba zobrazovat text, zobrazovat políčko pro zadání sázky, popisek k políčku, tři vykreslovací prvky, tlačítko pro spuštění automatu a tlačítko pro spuštění nové hry.

Obecný algoritmus aplikace by tedy mohl být následovný. Pokud uživatel stiskne tlačítko pro losování symbolů, tak se do proměnné načte vsazená částka. Dále se ověří, jestli má na danou sázku dostatek kreditu. Poté se vygenerují obrázky a podle výsledku se kredit hráče buď zvýší, nebo sníží. Poslední úkol je nastavení druhého tlačítka. Po jeho stisknutí se kredit nastaví na výchozí hodnotu a hráč může hrát dále.

Nyní je nutné vhodně zvolit vizuální komponenty. Vzhled aplikace se určuje v XML souboru aktivity_mail.xml. Cesta k souboru je NazevProjektu/app/src/main/res/layout. Pro

[21] ImageView. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/reference/android/widget/ImageView.html

vypisování textu slouží komponenta TextView. Pro zadávání sázky bude vhodné použít EditText typu Number, tím bude zamezeno i nevhodným vstupním datům. Nejdůležitější komponenta bude ImageView, která bude sloužit k vykreslování losovaných obrázků. Pro tlačítka bude použita komponenta Button. Aktuální otázkou je vhodné zvolení orientace aplikace. Pokud zvolíme orientaci na výšku, je vhodné zakázat překlápění aplikace na šířku. Zakázání probíhá v souboru AndroidManifest.xml. Cesta k souboru je NazevProjektu/app/arc/main. V tomto souboru přidáme parametry tagu aktivity. Prvním parametrem je android:configChanges="orientation" a následující android:screenOrientation="portrait". V konkrétní aplikaci Automaty tedy bude AndroidMainfest.xml vypadat následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.tony.automaty">
     <application
          android:allowBackup="true"
          android:icon="@mipmap/ic_launcher"
          android:label="@string/app_name"
          android:supportsRtl="true"
          android:theme="@style/AppTheme">
          <activity android:name=".MainActivity"
               android:configChanges="orientation"
               android:screenOrientation="portrait">
               <intent-filter>
                    <action
                android:name="android.intent.action.MAIN"
                    />
               <category
          android:name="android.intent.category.LAUNCHER"/>
               </intent-filter>
          </activity>
     </application>
</manifest>
```

V tuto chvíli se může zdát, že je vše připraveno k tvorbě uživatelského prostředí aplikace, ale to není úplně pravda. Při vkládání komponenty ImageView na plochu jsme ihned dotázáni, jaký obrázek se má zobrazit. Tím pádem potřebujeme v tuto chvíli získat tři různé obrázky. Pro jednodušší manipulaci s obrázky je vhodné, aby měly všechny tři obrázky

stejné rozměry. Po vytvoření či získání obrázků je nutné vložit je do projektu na určené místo. Cesta do příslušné složky je NazevProjektu/app/src/main/res/drawable.

Dalším krokem je vytvoření uživatelského prostředí. Postupně pomocí technologie DragAndDrop v režimu Design vhodně rozmístíme komponenty na plochu. V tuto chvíli je vhodné upravit texty zobrazované v TextView a na tlačítkách, jedná se o položky android:text="" v textovém režimu. Další položky hodné změny jsou android:textSize="" a android:textColor="", protože text ve výchozím nastavení je poměrně malý a špatně čitelný, také je značně menší než čísla zapisovaná do políčka EditText. Kód aktivity_main.xml by tedy mohl vypadat následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
     xmlns:android="http://schemas.android.com/apk/res/androi
d"
     xmlns:app="http://schemas.android.com/apk/res-auto"
     xmlns:tools="http://schemas.android.com/tools"
     android:id="@+id/activity_main"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:paddingLeft="@dimen/activity_horizontal_margin"
     android:paddingRight="@dimen/activity_horizontal_margin"
     android:paddingTop="@dimen/activity_vertical_margin"
     android:paddingBottom="@dimen/activity_vertical_margin"
     tools:context="com.example.tony.automaty.MainActivity">
     <TextView
          android:text="@string/kredit_500"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignParentTop="true"
          android:layout_centerHorizontal="true"
          android:layout_marginTop="18dp"
          android:id="@+id/textView"
          android:textSize="30sp"
          android:textColor="#000000"/>
     <ImageView
          android:layout_width="100dp"
          android:layout_height="200dp"
          app:srcCompat="@drawable/mic11"
          android:id="@+id/imageView2"
          android:layout_alignTop="@+id/imageView"
          android:layout_below="@+id/textView"
```

```
        android:layout_centerHorizontal="true"/>
    <ImageView
        android:layout_width="100dp"
        android:layout_height="200dp"
        app:srcCompat="@drawable/mic12"
        android:id="@+id/imageView3"
        android:layout_alignTop="@+id/imageView2"
        android:layout_below="@+id/textView"
        android:layout_alignParentEnd="true"/>
    <TextView
        android:text="@string/s_zka"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_alignParentStart="true"
        android:id="@+id/textView2"
        android:textSize="30sp"
        android:textColor="#000000"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="number"
        android:ems="10"
        android:layout_below="@+id/textView2"
        android:layout_alignParentStart="true"
        android:layout_marginTop="31dp"
        android:id="@+id/editText"/>
    <Button
        android:text="@string/nov_hra"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText"
        android:layout_alignParentStart="true"
        android:layout_marginTop="49dp"
        android:id="@+id/button"/>
    <Button
        android:text="@string/losuj"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button"
        android:layout_alignParentEnd="true"
        android:id="@+id/button2"/>
    <ImageView
        android:layout_width="100dp"
```

```
            android:layout_height="200dp"
            app:srcCompat="@drawable/mic10"
            android:id="@+id/imageView"
            android:layout_below="@+id/textView" />
</RelativeLayout>
```

Po dokončení úprav vzhledu je na řadě psaní kódu samotné aplikace. Cesta k hlavnímu Java souboru je

NazevProjektu/app/src/main/java/com/example/UzivatelskeJmeno/nazevProjektu/Main Activity. Ihned pod název třídy je vhodné deklarovat proměnné. Jelikož se jedná o jednoduchý projekt, jeví se jako vhodné všechny proměnné deklarovat jako privátní. Následně mohou být v metodě onCreate() všechny proměnné definovány. Proměnné komponent nemají běžné definování, např. pro Button b je definování následující: b = (Button) findViewById(R.id.button);. Momentálně bude IntelliJ hlásit chybu, protože prvek Button nezná, ovšem stačí jen pomocí klávesové zkratky Alt + Enter importovat příslušné knihovny.

Nyní je v pořadí nastavit chování aplikace po stisknutí tlačítka losování. Proto musíme deklarovat metodu, která se provede po stisku tlačítka, např: b.setOnClickListener(New View.OnClickListener(){});. Prvním krokem v metodě je vytvoření instance třídy Random, abychom mohli náhodně generovat čísla obrázků. V tuto chvíli do naší proměnné načteme hodnotu zadanou uživatelem a podmínkou ověříme, zda sázka není vyšší než kredit hráče. Pokud je zadaná hodnota v pořádku, vygenerujeme čísla obrázků a pomocí následně implementované metody obrázky vykreslíme. Poslední částí metody při stisknutí tlačítka je podmínka, která podle obrázků uživateli buď kredit zvýší, nebo sníží a aktuální stav kreditu zobrazí hráči. Metoda pro vykreslení obrázků obsahuje rozhodovací prvek switch, který podle vygenerované hodnoty zobrazí příslušný obrázek. Teď zbývá jen určit, podle jakého parametru se bude porovnávat, zda jsou obrázky stejné či nikoli. Za tímto účelem můžeme při vekreslování obrázků nastavit danému ImageView jeho identifikátor Id. Podmínka pro zvyšování či snižování bude tedy porovnávat Id všech tří komponent ImageView. Pro restart hry nám poslouží druhé tlačítko, kterému stačí nastavit, že při jeho stisknutí se hodnota kreditu obnoví na výchozí. Výsledný soubor MainActivity.java by mohl vypadat následovně:

```
Package com.example.tony.automaty;
```

```
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.Random;
import static java.lang.String.*;

public class MainActivity extends AppCompatActivity {
     private Button button;
     private Button button2;
     private ImageView iv,iv2,iv3;
     private EditText et;
     private TextView tv1,tv3;
     private int penize;
     private int sazka;

     private void vyberObrazekMice(ImageView iv){
          Random r = new Random();
          int f = r.nextInt(3) + 1;

          switch (f) {
               case 1:
                    iv.setImageResource(R.drawable.mic10);
                    iv.setId(1);
                    break;
               case 2:
                    iv.setImageResource(R.drawable.mic11);
                    iv.setId(2);
                    break;
               case 3:
                    iv.setImageResource(R.drawable.mic12);
                    iv.setId(3);
                    break;
          }
     }

     @Override
     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        penize = 500;
        iv = (ImageView) findViewById(R.id.imageView);
        iv2 = (ImageView) findViewById(R.id.imageView2);
        iv3 = (ImageView) findViewById(R.id.imageView3);
        tv1 = (TextView) findViewById(R.id.textView);
        tv3 = (TextView) findViewById(R.id.textView2);
        button = (Button) findViewById(R.id.button2);
        button2 = (Button) findViewById(R.id.button);
        et = (EditText) findViewById(R.id.editText);

        button.setOnClickListener(new
    View.OnClickListener(){
              @Override
              public void onClick(View v) {
                   tv3.setTextColor(Color.BLACK);
                   try {
                   sazka =
              Integer.parseInt(et.getText().toString());
              }catch (RuntimeException e){
                   e.printStackTrace();
                   tv3.setTextColor(Color.RED);
                   sazka = 0;
              }
              tv1.setTextColor(Color.BLACK);
              if(penize == 0){
                   button.setEnabled(false);
              }
              if(sazka > penize){
                   tv1.setTextColor(Color.RED);
              }else {
                   vyberObrazekMice(iv);
                   vyberObrazekMice(iv2);
                   vyberObrazekMice(iv3);
                   if (iv.getId() == iv2.getId() &&
              iv2.getId() == iv3.getId()) {
                        penize = penize + sazka * 5;
                        tv1.setText(format("%s%d", new
                        Object[]{getString(R.string.kredit),
                        penize}));
                   } else {
                        penize = penize - sazka;
```

```
                    tv1.setText(format("%s%d", new
                Object[]{getString(R.string.kredit),
                penize}));
                }
            }
        }
    });
        button2.setOnClickListener(new
    View.OnClickListener(){
            @Override
            public void onClick(View v) {
                penize = 500;
                tv1.setTextColor(Color.BLACK);
                tv1.setText(String.format("%s500",
            getString(R.string.kredit)));
            button.setEnabled(true);
            }
        });
    }
}
```

Momentálně je aplikace připravena na testování. Nejdříve bychom měli ověřit správnost chování a následně ji nejlépe nechat otestovat další osobu. Pokud návrh naší aplikace funguje správně, lze ji považovat za dokončenou. Díky vhodnému zvolení pole pro vstupní data nelze očekávat ani chybu vstupních dat, která by se dala ošetřit pomocí výjimek. V tuto chvíli je již možné začít aplikaci používat a případně se o ni dále podělit s veřejností.

4.2 KONVERZE JAVA APLIKACÍ NA ANDROID

Konverzí je myšlena především nějaká aplikace, která by dokázala modifikovat Java aplikaci na Android aplikaci. Ovšem i když v dnešní době existuje mnoho sofistikovaného programového vybavení, při realizaci této práce nebyl dohledán nástroj, který by tuto problematiku pohodlně řešil. Takový nástroj by v dnešní době určitě uvítala celá řada společností a programátorů.

4.2.1 DOPORUČENÝ POSTUP

Pokud tedy chceme vytvořit Android aplikaci na základě Java aplikace, stačí si do vytvořeného Android projektu zkopírovat třídy z Java projektu, následně v příslušném XML souboru nastavit požadované uživatelské rozhraní a na něj navázat původní metody. Toto

ovšem určitě nebude vše, co bude muset programátor upravit. Pokud se jedná o nějakou aplikaci například pro práci s databází, bude nutné v souboru AndroidManifest.xml zadat požadovaná práva aplikace, která uživatel při instalaci aplikace schvaluje. Při testování na jednoduchých aplikacích došlo k obměně přibližně 75 % kódu. Toto procento by se ale mohlo u větších projektů snižovat, protože jde zejména jen o změnu uživatelského rozhraní. Pokud má ale aplikace za úkol vykreslovat grafy, tento kód bude značně jiný. Pokud je cílem programátora vytvoření Android aplikace, bude se muset zaobírat otázkou, zda přizpůsobování kódu nezabere více času než vytváření kompletní nové aplikace.

Zde tedy vidíme, kolik problematiky se může za konverzí projektů skrývat. Obecně to tedy vypadá, že aplikace na konverzi se nejspíše v dohledné době nedočkáme, a pokud ano, její volné šíření není příliš pravděpodobé.

ZÁVĚR

V současné době je Android nejrozšířenější mobilní platformou. Najdeme ji jak v chytrých telefonech, tabletech, chytrých televizích, tak se v současné době už začíná objevovat i v automobilech. Platforma se stále rozšiřuje a vylepšuje, a proto lze očekávat, že vývoj aplikací pro Android bude i v budoucnu žádanou službou.

V této práci je stručně popsána platforma Android, důraz byl kladen především na její architekturu. Představeno bylo několik volně dostupných vývojových prostředí. Také proběhlo stanovení kritérií pro výběr vhodného IDE pro vývoj Android aplikací. Práce je zaměřena na vývojový nástroj IntelliJ. Potenciální začínající programátor zde zjistí, jak vývojový nástroj nainstalovat a poprvé spustit.

V další části práce jsou představeny hlavní funkce prostředí IntelliJ. Tyto funkce značně zjednoduší vývoj Android aplikací. U každé z funkcí je uveden konkrétní příklad použití pro jednodušší pochopení dané problematiky. Pokud se tedy programátor rozhodne pro vývoj Android aplikací v prostředí IntelliJ, mohl by být po přečtení dané části této práce schopen vývojový nástroj používat a následně přispět k urychlení samotného vývoje aplikací.

V poslední kapitole práce je uveden obecný postup, který by mohl být aplikován k vytváření Android aplikací. Postup je rozčleněn do pěti částí a mohl by pomoci začínajícím programátorům a vývojářům Android aplikací.

Obsahem poslední kapitoly práce je představení základních grafických komponent formulářových aplikací, které byly v rámci této bakalářské práce vytvořeny. Vytvořené aplikace byly založeny na příkladech z předmětu programování 2 vyučovaném na Katedře výpočetní a didaktické techniky Fakulty pedagogické Západočeské univerzity v Plzni.

Navržený postup vývoje aplikací byl úspěšně prakticky ověřován na všech vytvořených příkladech a zdokumentován na vybrané příkladové aplikaci. V zájmu ukázat jedno z možných řešení, tvoří součást dokumentace k aplikaci i její zdrojový kód.

Součástí práce je také pojednání o možnostech konverze Java aplikací na Android aplikace.

RESUMÉ

Práce je zaměřena na vývoj Android aplikací ve vývojovém prostředí IntelliJ od společnosti JetBrains. Nejvíce je kladen důraz na funkce vývojového nástroje a návrh postupu vývoje aplikace.

Nejdříve jsou uvedeny základní informace o platformě Android a její architektuře. Dalším obsahem je jednoduchý popis vybraných vývojových prostředí. První kapitola je uzavřena stanovením kritérií pro výběr optimálního vývojového prostředí.

Další část obsahuje zdůvodnění výběru vývojového prostředí IntelliJ. Na tuto část navazuje popis získání nástroje a způsob jeho instalace. Druhou kapitolu uzavírá seznámení s grafickým uživatelským prostředím a jeho možnostmi.

Ústřední částí práce je třetí kapitola, ve níž jsou popsány hlavní funkce vývojového prostředí a příklady jejich použití na jedné z programovaných Android aplikací.

Poslední část je věnována efektivnímu vývoji Android aplikací. Hlavní částí je vlastní návrh postupu při vývoji Android aplikací. Dále kapitola obsahuje stručný popis grafických prvků používaných při programování sady příkladů. Závěrem je vyzkoušení doporučeného postupu na jednom z vytvořených příkladů včetně zdrojového kódu.

Cílem práce je seznámit začínající vývojáře s prostředím IntelliJ a způsobem, jak vhodně vytvářet Android aplikace.

SUMMARY

The Bachelor Thesis deals with Android applications‘ develoment in the development environment of intelliJ by the JetBrains company. It emphesizes the function of the development instrument and the draft of the application’s progress.

First, basic information on the Android platform and its architecture is introduced. Next content is a simple description of development environment selected by the author. The first chapter is concluded by stating of criteria how to choose an optimal development environment.

The next chapter contains reasoning why the development environment of IntelliJ was selected by the author of this Thesis. This part is followed by the description how to gain the instrument and the method of its installation. The end of this chapter deals with the graphical user interface and its options.

The main part is chapter number three, which is focused on the development environment’s main functions and contains examples of their use in one of the programmed Android applications.

The last chapter deals with efficient development of Android applications. It comprises also the athor’s design for the process of the Android applications‘ development. Next a simple description of graphic elements used for programming of the example set is characterized. In the final part of this chapter the recommended process is also tried on one of the examples created including the source code.

The aim of this Thesis is to introduce the environment of IntelliJ to a developer - beginner and the way how to create Android applications appropriately.

SEZNAM LITERATURY

LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

KROCHMALSKI, Jarosław. *IntelliJ IDEA Essentials*. Birmingham: Packt Publishing Ltd., 2014. ISBN 978-1-78439-693-0.

LEE, Wei-Meng a Chaim Krause. TECHNICAL EDITOR. *Beginning Android 4 application development*. Indianapolis, Ind: Wrox/John Wiley & Sons, 2012. ISBN 9781118265383

MEDNIEKS, Zigurd R. *Programming Android*. 2nd ed. Beijing: O'Reilly Media, c2012. ISBN 978-1-4493-1664-8.

Android vládne mobilnímu trhu. *Adsl.cz* [online]. 2017 [cit. 2017-03-19]. Dostupné z: http://www.adsl.cz/clanky/android-vladne-mobilnimu-trhu

Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. *Gartner* [online]. Egham, UK, 2017 [cit. 2017-03-19]. Dostupné z: http://www.gartner.com/newsroom/id/3609817

Google's hiring may have slowed, but it's still adding thousands of new employees. *Business Insider* [online]. [cit. 2017-03-30]. Dostupné z: http://www.businessinsider.com/google-has-57000-employees-2015-7

Our offices. *Google* [online]. [cit. 2017-03-30]. Dostupné z: https://www.google.com/intl/en/about/locations/

USB Type-C™ Cable and Connector Specification. *Universal Serial Bus* [online]. [cit. 2017-03-19]. Dostupné z: http://www.usb.org/developers/usbtypec/

Dashboards. *Android Developers* [online]. [cit. 2017-02-20]. Dostupné z: https://developer.android.com/about/dashboards/index.html

Platform Architecture. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/guide/platform/index.html

Programovací jazyk Lua. *ROOT.CZ* [online]. 2009 [cit. 2017-03-19]. Dostupné z: https://www.root.cz/clanky/programovaci-jazyk-lua/

CRIDER, Michael. *Google officially ends support for Eclipse Android Developer Tools in favor of Android Studio* [online]. 2016 [cit. 2017-03-19]. Dostupné z: http://www.androidpolice.com/2016/11/02/google-officially-ends-support-for-eclipse-android-developer-tools-in-favor-of-android-studio/

WHY CHOOSE CORONA SDK ? *Corona Labs* [online]. [cit. 2017-03-19]. Dostupné z: https://coronalabs.com/corona-sdk/

Android Studio. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/studio/index.html

Awards. *JET BRAINS* [online]. [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/company/press/awards.html#product=Intellij%20IDEA

What are the best IDEs for Android development in Java? *Slant* [online]. [cit. 2017-03-19]. Dostupné z: https://www.slant.co/topics/4649/versus/~intellij-idea_vs_android-studio_vs_eclipse-android-development-tools-plugin

SHARMA, Asankhaya. How do developers choose their IDE? *Linkedin* [online]. 2014 [cit. 2017-03-19]. Dostupné z: https://www.linkedin.com/pulse/20140522113335-16837833-how-do-developers-choose-their-ide

InfoWorld's 2015 Technology of the Year Award winners. *InfoWorld* [online]. 2015 [cit. 2017-03-19]. Dostupné z: http://www.infoworld.com/article/2871935/application-development/infoworlds-2015-technology-of-the-year-award-winners.html?nsdr=true

JetBrains [online]. [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/

Basic Code Completion. Completing Names and Keywords. *JetBrains* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/help/idea/2016.3/basic-code-completion-completing-names-and-keywords.html

Smart Type Code Completion. Completing Code Based on Type Information. *JetBrains* [online]. 2017 [cit. 2017-03-19]. Dostupné z: https://www.jetbrains.com/help/idea/2016.3/smart-type-code-completion-completing-code-based-on-type-information.html

TextView. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/reference/android/widget/TextView.html

Spinners. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/guide/topics/ui/controls/spinner.html

ImageView. *Android Developers* [online]. [cit. 2017-03-19]. Dostupné z: https://developer.android.com/reference/android/widget/ImageView.html

SEZNAM OBRÁZKŮ, TABULEK, GRAFŮ A DIAGRAMŮ

PŘÍLOHY

Příloha 1 – CD

Příloha 1 - CD

Na CD se nachází:

- Adresář s aplikacemi vytvořenými v prostředí IntelliJ.
- Vlastní text bakalářské práce ve formátu docx a pdf.