Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

MobChar - balíček pro Gurps

Ondřej Lakomý

Vedoucí práce: Ing. Zdeněk Rybola

18. května 2017

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Zdeňku Rybolovi za ochotu při konzultacích a za poskytování rad zejména při návrhu balíčku. Dále bych chtěl poděkovat Matěji Shánělovi, za velkou pomoc v začátku tvorby balíčku, za nesčetné rady k programování pro Android a rady všeho druhu v průběhu tvorby práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 18. května 2017 .....

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

Lakomý, Ondřej. *MobChar - balíček pro Gurps*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Aplikace MobChar slouží ke správě postav pro hráče her na hrdiny a je dostupná pro operační systém Android. v současnosti aplikace podporuje tvorbu a správu postav pro hry Dračí doupě a Dungeons and dragons. Tato práce se bude zabývat tvorbou rozšiřujícího balíčku pro současnou aplikaci MobChar, který umožní také hráčům hrajícím ve světě podle pravidel GURPS využívat tuto aplikaci. Budu se zabývat analýzou a porovnáním odlišností her Dungeons and Dragons, Dračí doupě a GURPS, dále navrhnu nový balíček, implementuji jej a provedu testování. pro lepší orientaci hráčů dále vytvořím uživatelskou příručku s popsáním funkcionality.

Klíčová slova MobChar, GURPS, Java, Android, balíček, příručka

Abstract

MobChar application is used to manage characters by players playing role-playing games. It is available for Android operating systems. Application currently provides creation of characters for Draci doupe and Dungeons and Dragons. This thesis is supposed to make another package for MobChar which

will allow the players playing by GURPS rules to make and manage characters as well. I will be concerned about analysis and comparing of the differences of games Dungeons and Dragons, Dračí doupě and GURPS. Next step will be to design a new package, implement it and test it. For better understanding, I will make a user manual for the players where functionality will be described.

Keywords MobChar, GURPS, Java, Android, package, user manual

Obsah

Seznam obrázků

Úvod

Co je to vlastně hra na hrdiny? Hra na hrdiny je typ hry, kterou hraje skupina hráčů ať již ve formě deskové či např. počítačové hry. Každý hráč si vytvoří svoji postavu, za kterou pak následně prožívá dobrodružství a v podstatě tak přejímá její osobnost, povahu a historii. Jeden z hráčů si postavu netvoří, ale představuje vypravěče. Hra probíhá podle následujícího schématu: Vypravěč si připraví nějaké dobrodružství. Poté se hráči sejdou a dané dobrodružství hrají. Hra se skládá z kol, přičemž každé kolo probíhá tak, že hráči sdělí vypravěči, co chtějí udělat v následujícím kole a vypravěč jim odpoví, jestli se jim daná činnost povedla, jestli se odehrálo něco nečekaného, také jim odhalí, jak na jejich činy reagovaly ostatní postavy nebo prostředí, ve kterém se nacházejí.

Příběhy se můžou odehrávat v rozličných světech. Hráči mohou sbírat předměty jako zbraně a zbroje, které jim pomohou porazit nejnebezpečnější protivníky, mohou také vyrábět lekvary a jiné užitečné věci. Můžou se rozhodnout pro kariéru lupiče, zloděje nebo dokonce vraha, můžou se stát obchodníkem, řemeslníkem, anebo můžou konat dobro a stát se bojovníkem či ochráncem. Mohou dokonce také ovládnout magii a být čarodějem, černokněžníkem nebo třeba i knězem.

Takových her už vznikla spousta. Mezi nejznámější z deskových her patří Dungeons and Dragons (dále DnD), české Dračí doupě (dále DrD), ale také GURPS. ve všech těchto hrách je potřeba zaznamenávat vlastnosti, schopnosti a vybavení každé postavy. k takovýmto účelům slouží hráčům zpravidla papír a tužka. s postupem času se však tento princip uchovávání postavy ukáže jako nevhodný, neboť hráči musí každou chvíli informace upravovat. Kvůli tomu vznikla aplikace MobChar na mobilní telefony, která umožňuje velmi jednoduchou úpravu a uschování údajů právě na telefonu. Vše se tak stane přehlednější a příjemnější, hráči se mohou více soustředit na samotné hraní hry než na záznam údajů.

V současné chvíli aplikace obsahuje balíčky pro tvorbu postavy podle pravidel DnD a DD. Úkolem této práce je navrhnout a implementovat balíček na zmiňovaná pravidla GURPS.

Struktura práce

Nejprve budu analyzovat požadavky na samotný balíček. Detailně prostuduji a popíšu pravidla GURPS a zjistím rozdíly těchto pravidel od pravidel již implementovaných balíčků. Poté provedu analýzu aplikace MobChar. Zjistím, co z funkcionality potřebné pro můj balíček je v aplikaci již implementováno a co musím implementovat. ze zjištěných informací navrhnu strukturu vlastního balíčku, popíšu jednotlivé vrstvy - např. zvolenou databázi.

Po navrhnutí balíčku bude následovat implementace, propojení balíčku s existující aplikací. Jedním z posledních kroků pak bude otestování aplikace. Nakonec sepíšu uživatelskou příručku pro snadnou orientaci uživatelů.

Cíl práce

Cílem mé práce je navrhnout rozšiřující balíček pro stávající aplikaci MobChar, která bude hráčům umožňovat spravovat postavy podle pravidel GURPS. Balíček bude sloužit k usnadnění administrace postavy - např. zaznamenávání údajů o postavě nebo evidování předmětů, které postava na dobrodružstvích nalezla. Dále bude aplikace poskytovat vypočtené hodnoty vedlejších atributů.

Stanovil jsem si průběžné menší cíle:

- Analyzovat pravidla pro hraní her na hrdiny GURPS
- Navrhnout rozšiřující balíček pro stávající aplikaci MobChar
- Implementovat rozšiřující balíček pro stávající aplikaci MobChar
- Integrovat balíček do stávající aplikace
- Vytvořit uživatelskou příručku umožňující uživatelům snadnější orientaci v aplikaci

Co není cíl práce

Cílem práce není implementovat funkcionalitu samotné hry. Aplikace neslouží k samotnému hraní hry GURPS - používá se sice při hraní, ale nevykonává herní logiku ani nezastupuje hráče či GM. Slouží pouze jako záznamové zařízení, které zjednodušuje hráčům evidenci postavy.

Kapitola 1

Analýza

V kapitole Analýza bych nejprve představil samotnou hru GURPS. Nastíním základní princip hraní, průběh herního kola, pravidla hry a další neopomenutelné součásti GURPS. Dále bych pak popsal stávající aplikaci MobChar. Rozeberu společné části aplikace, které jsou v současnosti dvě: jádro aplikace a sdílená knihovna. ve třetí části bych pak specifikoval požadavky na balíček - co všechno potřebuje hráč hry GURPS v aplikaci mít, aby byla aplikace skutečně využitelná v praxi a usnadnila hráči samotné hraní.

1.1 GURPS

Hra GURPS je jednou z mnoha variant hry na hrdiny. Zkratka GURPS v angličtině znamená Generic Universal RolePlaying System. Znamená to, že pravidla GURPS jsou jednoduchá, avšak volitelně rozšiřitelná o více specifická pravidla. Začátečníci proto mohou použít základní verzi pravidel, zatímco pokročilejší hráč mohou použít pravidla více specifická a vytvořit tak daleko více věrohodný zážitek ze hry. Pravidla jsou dále univerzální - mohou být použita jak pro příběh z dob dávno minulých, tak i pro příběhy z budoucnosti.

Podobně jako u většiny her na hrdiny, ani v GURPS nechybí vypravěč, tzv. "Game Master"(dále GM). GM připravuje dobrodružství, sděluje hráčům reakce okolí na jejich akce nebo jedná za postavy neovládané hráči (dále NPC - z anglického "Non Player Character").

Existuje nespočetné množství knih a jiných publikací, rozšiřujících herní světy a pravidla GURPS. Některé jsou zdarma k dispozici ve formátu PDF, některé jsou placené. Některé rozšiřují obecná pravidla, některé poskytují konkrétní informace a rady jak hrát GURPS v určitém časovém období nebo v určité situaci (příkladem publikace GURPS Zombies, poskytující rady a návody, jak založit dobrodružství ve světě ovládaném zombiemi).

Moje bakalářská práce čerpá pouze z publikace GURPS Lite [1], což je základní soubor pravidel pro hraní ve světech GURPS. Obsahuje základní pravidla, ale pouze v takovém množství, aby umožnila hratelnost hry, ale zároveň zbytečně hru nekomplikovala pro nové hráče. v následujících podkapitolách stručně shrnu základní pravidla a principy GURPS čerpané právě z aplikace GURPS Lite.

1.1.1 Základní principy

Herní cyklus GURPS je podobný jako u většiny ostatních her na hrdiny. GM hráčům popíše situaci, ve které se nacházejí. Popíše prostředí, v němž se postavy nacházejí, sdělí jim, zda jsou s nimi nějaké další postavy, atd. Hráči pak sdělí GM svoji akci, GM vyhodnotí, jestli se akce povedla, pokud je to relevantní, vyhodnotí všechny následky akce hráčů a sdělí jim výsledek jejich jednání, např. reakce ostatních postav. Poté se opět vrací do prvního bodu a popisuje prostředí. Jedna hra může trvat dny, týdny, ale i několik let - vše záleží na hráčích a GM.

V pravidlech GURPS se pro výpočty hodnot používá šestistěnná kostka. Základním mechanismem je pak tzv. "dice+adds"pojem, který znamená "hoď kostkou + přičti bonus". v praxi pak následující mechanismus vypadá následovně: hráč hází podle formule "4d+2", což znamená 4-krát hodnota na kostce plus 2. Hráč hodí 3, konečný výsledek tedy bude 4 * 3 + 2 = 14. Navazující herní mechanismy jsou tzv. "Success rolls", "Reaction rolls"a "Damage rolls". První dva mechanismy rozeberu v dalších odstavcích, Damage roll bude vysvětlen později v pravidlech boje.

1.1.1.1 Success rolls

Success roll (dále SR) je hod třemi kostkami proti některé z vlastností hráče, zpravidla když je potřeba otestovat, jestli postava zvládla nějakou činnost či nikoliv. Například když chce postava zvednout těžký předmět, provede SR proti své síle, aby zjistila, jestli předmět dokáže nebo nedokáže zvednout. Někdy místo hráče za postavu hází GM - například v případě, zda si postava všimne temné stopy na podlaze. GM si hodí pro sebe a pokud hod uspěje, informuje hráče, že jeho postava si stopy všimla.

Někdy se do hodnoty hodu započítají modifikátory. Například když předmět, který se postava snaží zvednout, je enormně těžký, sníží se hodnota atributu, proti kterému postava hází, o modifikátor (např. Strength - 3). Postava má tak menší šanci uspět při zvedání předmětu. Modifikátor může být jak kladný, tak záporný, a zpravidla záleží na rozhodnutí GM.

Při hodech se můžou vyskytnout "kritický úspěch"(critical success) nebo "kritický neúspěch"(critical failure). Tyto jevy zpravidla znamenají, že se čin-

nost, o kterou se postava snaží, povedla lépe, než se očekávalo (resp. vůbec nepovedla). GM určí, co v dané situaci kritický úspěch či kritický neúspěch znamená. Jako kritický úspěch se označují velmi nízké hody, naopak jako kritický neúspěch velmi vysoké hody.

1.1.1.2 Reaction rolls

Podobným hodem, jako je SR, je Reaction roll (dále RR). RR se používá při vyhodnocování, jak se k postavě zachová NPC. Zde platí, že čím je výsledná hodnota po započítání modifikátorů vyšší, tím lépe na hráče NPC reaguje. Reakce NPC můžou být při velmi nízkém hodu silně negativní vůči postavě hráče, nebo naopak velice kladné až ochranářské při velmi vysokém hodu - záleží na interpretaci GM. Zde je vhodné poznamenat, že GM nemusí provést hod, ale může reakci zvolit podle svého uvážení.

1.1.2 Postava

Hlavní myšlenkou her na hrdiny je samotné hraní role postavy (role-playing), tedy vcítění se do jiné postavy a jednání tak, jako by samotný hráč byl danou postavou. z toho vyplývá, že tvorba postavy je jedním z nejdůležitějších, neli úplně nejdůležitějším faktorem hraní. Postava by neměla být "plochá", což lze vysvětlit tak, že by neměla mít samé dobré nebo samé špatné vlastnosti. Postava, která bude zajímavá ke hraní, by měla být v něčem dobrá a naopak v něčem horší, mít své silné stránky, ale také svoje slabiny. Doporučuje se dobře promyslet a sepsat historii postavy - odkud pochází, jaká byla její minulost, kde vyrůstala, jaké vlastnosti a schopnosti z toho pramení. Všechny tyto faktory by měly hráči pomoct pochopit nejen jak jeho postava jedná a reaguje, ale hlavně jak přemýšlí.

1.1.2.1 Osobnostní body

Osobnostní body jsou speciální měna ve světě GURPS. Pomocí těchto bodů může hráč vylepšovat postavu - učit ji nové schopnosti, vylepšovat je na vyšší úroveň nebo vylepšovat atributy postavy. Pokud je k tomu postava způsobilá, pak lze osobnostní body využít také pro učení nových kouzel. pro většinu kampaní je doporučená základní hodnota kolem 100 bodů pro každého hrdinu. Tyto body se však nepoužívají jen pro tvorbu a rozvoj postav hráčů, GM pomocí nich tvoří také NPC. Například průměrný vesničan, kterého může hráčova postava potkat ve vesnici, může mít 25 osobnostních bodů, zatímco hrdina, který zachrání celé město, může disponovat až 300 osobnostními body.

Vždy, když hráči dokončí dobrodružství, může jim být od GM přidělen obnos osobnostních bodů. Každý hráč dostane body za to, že dokončil samotné dobrodružství, ale druhá část bodů je přidělována za to, jak dobře hráč za postavu hrál. Toto osobně vidím jako velmi vhodné pravidlo, protože hráči

můžou dobrým výkonem až zdvojnásobit obnos svých bodů. Může se však stát, že hráč nedostane body vůbec žádné (viz. TODO ODKAZ na KAPITOLU "PŘÁTELÉ A NEPŘÁTELÉ").

Každá postava si může své dostupné osobnostní body rozdělit mezi libovolné vlastnosti, schopnosti, kouzla a charakteristiky. Body navíc, které hráč obdrží po dokončení každého dobrodružství, může investovat do schopností, kouzel nebo atributů (do oblastí postavy, které jsou povoleny měnit po zahájení kampaně). Hráč má k dispozici maximální počet bodů, které může na postavě utratit. na konci rozdělování bodů musí být součet všech hráčem investovaných bodů nižší nebo stejný jako je jeho maximální počet bodů.

1.1.2.2 Základní atributy

Každá postava disponuje čtyřmi základními atributy - síla (Strength = ST), obratnost (Dexterity = DX), inteligence (Intelligence = IQ) a zdraví (Health = HT). Čím více má postava síly, tím snadněji pohne s těžkými dveřmi, přenese obří artefakt nebo přemůže slabšího protivníka v neozbrojeném boji. Dále síla také zvyšuje poškození zbraní nebo zvětšuje počet kouzel, které může postava vykouzlit (prodlužuje čas, než se vyčerpá). Obratnost se používá k vypočítávání pohybových činností - například lezení na skály nebo přeskakování propastí. Postavy s vysokou inteligencí se lépe učí kouzla, mentální schopnosti a snadněji odhalí například podvod při obchodu. Zdraví pak slouží především k stanovení počtu zásahových bodů postavy, ale vyšší zdraví také poskytuje vyšší odolnost vůči nákazám. Výchozí hodnotou každého atributu u průměrného člověka je 10. Nižší hodnota znamená osobnostní body navíc, které může hráč investovat jinde, vyšší hodnota naopak vyžaduje bodovou útratu.

1.1.2.3 Vzhled postavy

Definuje vzezření postavy. Postava může být obézní, vychrtlá, charismatická. Může mít mateřské znaménko na levém spánku, jizvu na koleni nebo nesouměrný obličej - cokoliv co hráče napadne. Základní možnosti definované v GURPS Lite mají stanoveny své ceny osobnostních bodů, pokud hráč vymyslí jinou vlastnost, diskutuje cenu s GM.

1.1.2.4 Společenské postavení

Společenské postavení obsahuje spoustu ať už kladných či záporných vlastností, které může postava mít. Může být například zatížena povinností (např. z povolání). Postava která pracuje jako policista bude zatížena povinností pomoct slabším v nouzi. Postava musí povinnost dodržovat, i kdyby ji to mělo dostat do složité situace nebo dokonce ohrozit na majetku či životě! Pokud po-

stava nebude dodržovat svoji povinnost, GM je oprávněn udělit hráči snížený nebo i žádný počet bodů za hraní postavy na konci dobrodružství.

1.1.2.5 Bohatství postavy

Bohatství postavy definuje, s kolika penězi postava disponuje na začátku dobrodružství, kolik si vydělá za každý herní měsíc, ale i jak hodně času musí postava trávit tím, aby si peníze vydělala. v pravidlech GURPS je popsáno mnoho úrovní bohatství. Každá postava začíná na standardním bohatství, pokud si nezvolí chudobu nebo naopak nadměrné bohatství.

1.1.2.6 Přátelé a nepřátelé

Velice zajímavou kategorií GURPS postavy jsou přátelé a nepřátelé postavy. Jak přátelé, tak nepřátelé mají zavedenou frekvenci výskytu (Frequency of Appearance). Tato vlastnost nám říká, s jak velkou šancí se v dobrodružství vyskytnou. Mocní přátelé jsou bráni jako výhoda, mocní nepřátelé a slabí spojenci, které musí postava bránit, nebo jí jinak komplikují život, jsou bráni jako nevýhoda, a oceněni podle toho osobnostními body. Navíc čím vyšší mají frekvenci výskytu, tím je cena vyšší.

Mezi základní druhy přátel patří spojenci (Allies) a patroni (Patrons). Spojenci jsou tvořeni maximálně 150 charakterovými body. Doprovází hráče na jejich dobrodružství a pomáhají jim. Avšak hráč nedostane od GM žádné osobnostní body na konci dobrodružství, pokud svého spojence v průběhu napadl, zradil nebo zbytečně ohrozil. Patroni jsou nejmocnější spojenci. Mohou disponovat lepším vybavením, obrovským bohatstvím, speciálními schopnostmi. Patronem může být například i obrovská nadnárodní organizace. Hráč, který si vymyslí svého patrona, mu pak také vybere adekvátní frekvenci výskytu.

Nepřítelem postavy může být jak NPC, tak např. organizace. Takoví nepřátelé se snaží hráči uškodit, ublížit mu, nebo ho dokonce zabít. Hráč by měl u každého nepřítele dobře vysvětlit, jaký příběh se ukrývá za jejich špatným vztahem.

1.1.2.7 Výhody

Výhody jsou skupinou vlastností hráče, které jsou obecně považovány za pozitivní. za normálních okolností mohou být zvoleny pouze při vytváření postavy. Výhod postav je opět v GURPS Lite spousta, jako například neuvěřitelná ohebnost (Double-Jointed) nebo silná vůle (Strong Will). Některé výhody mají fixní cenu osobnostních bodů, některé se dají získat na několika stupních.

1.1.2.8 Nevýhody

Opakem výhod jsou nevýhody. Je nepsaným pravidlem, že výhody si může hráč vzít pouze při vytváření postavy. Je ale možné v pozdější fázi hry určitou nevýhodu vykoupit (za její původní zápornou cenu). Některé vlastnosti, jako například pravdomluvnost, sice ve skutečnosti nejsou přímo nevýhody, ale pro účely hraní GURPS jsou mezi ně zahrnuty - převážně proto, že omezují svobodu hráče. Mezi nevýhody patří zbabělost (Cowardice) nebo i například to, že má postava pouze jedno oko (One Eye).

1.1.2.9 Výstřednosti

Poslední poněkud méně významnou kategorií vlastností postavy jsou výstřednosti (Quirks). Jedná se o vlastnost, která není považována ani za výhodu, ani za nevýhodu. Většinou se jedná o velice konkrétní specifickou vlastnost - např. postava trvá na tom, že chce být placena pouze penězi.

1.1.2.10 Schopnosti

Obsáhlou kategorií specifikace postavy, kde může každý hráč dát prostor své fantazii, je položka Schopnosti. Schopnost je určitá znalost, kterou postava ovládá. Každá schopnost má svůj atribut, od kterého se odvíjí (Controlling attribute). Schopnosti rozdělujeme do dvou hlavních kategorií: fyzické (Physical) a mentální (Mental). Pravidlem je, že fyzické schopnosti závisí na síle postavy a mentální na inteligenci. Některé schopnosti se však tomuto pravidlu vymykají. Celkový level dané schopnosti se spočítá součtem aktuálního levelu řídícího atributu a modifikátoru. Modifikátor pro fyzické schopnosti se pohybuje v hodnotách od -3 do +4, pro mentální schopnosti od -4 do +2. Dále schopnosti rozdělujeme na čtyři kategorie podle náročnosti - lehké (Easy), průměrné (Average), těžké (Hard) a velmi těžké (Very Hard). Čím těžší je schopnost, kterou se hráč chce naučit, tím více bodů osobnosti bude další level stát.

Většina schopností má výchozí hodnotu, na které je každá postava schopna danou schopnost vykonávat. Jedná se většinou o méně náročné schopnosti, jako například šplhání (Climbing) nebo gamblerství (Gambling). Těžší a náročnější schopnosti pak výchozí hodnotu mít nemusí.

1.1.2.11 Rasy

V mnoha světech, kde se dobrodružství odehrávají, se nevyskytují pouze lidé. Může jít o jiné rasy z fantasy, nebo dokonce i sci-fi světů. Pokud se hráč rozhodne vytvořit si postavu jiné rasy, je potřeba definovat takovou rasu a sepsat historii a kulturu.

Pro každou rasu je vhodné si vytvořit šablonu rasy - soubor modifikátorů atributů, výhod, nevýhod a výstředností postav, které platí obecně pro všechny členy dané rasy. Tento úkol má obvykle na starosti GM.

Modifikátor atributů Zatímco pro každou postavu lidské rasy je průměrná hodnota každého atributu 10, u jiných ras se tato hodnota může lišit. z toho důvodu definujeme u rasy modifikátor atributů. Počítáme, že výchozí hodnota každého atributu je 10, hráč si pak vybere například +2 ST (což stojí 20 osobnostních bodů), tzn. jeho postava by měla mít sílu 12. Jenže modifikátor síly pro danou rasu je +3, takže výsledná hodnota síly postavy je 15 (modifikátor atributu se připočítá až jako poslední).

Výhody, nevýhody, výstřednosti, schopnosti Tato část tvorby rasy závisí zcela na uvážení a logickém zdůvodnění tvůrce rasy. Každá postava této rasy pak tyto vlastnosti, resp. schopnosti "zdědí".

Popis rasy Autor rasy by měl po definování základních vlastností dané rasy také vymyslet její původ, politickou situaci, vztahy k ostatním, přirozené prostředí, ve kterém členové rasy žijí a mnoho dalších - představivosti se meze nekladou.

1.1.2.12 Konec výpravy

Každý hráč na konci dobrodružství může obdržet od GM obnos osobnostních bodů. Jeho výše záleží především na výkonu hráče při výpravě a na tom, jak dobře hráč za postavu hrál - maximum bodů je 5. Nemá ale nárok na žádné body, pokud byla během dobrodružství jeho závislá postava zabita, vážně raněna nebo unesena a nevysvobozena.

Hráč může dané body investovat stejně jako při tvorbě postavy, až na některé výjimky:

- Základní atributy stojí dvojnásobný počet bodů než při tvorbě postavy (všechny schopnosti závisející na daném atributu se také o level vylepší)
- Většina výhod je daná a nejde "přikoupit"později - existují i výjimky, rozhodne GM.
- Nelze si "koupit"nevýhodu, pouze je možno naopak se nějaké nevýhody zbavit, pokud hráč logicky odůvodní, proč se tak stalo. v takovém případě zaplatí tolik osobnostních bodů, kolik za danou nevýhodu dostal při tvorbě postavy.
- Pokud chce hráč vylepšit schopnost, musí se jednat o schopnost, kterou při dobrodružství využíval.

1.1.3 Vybavení

Hráči se při svých dobrodružstvích budou potkávat s různými předměty. Zatímco předměty, které postava pouze nosí s sebou, nemusí mít přesně definované vlastnosti, u zbrojí, štítů a zbraní jsou bojové statistiky velice důležité. Pravidla GURPS na ně nezapomínají a obsahují důležité informace o základních typech bojového vybavení.

1.1.4 Herní mechanismy

GURPS mají spoustu herních mechanismů, přičemž téměř všechny jsou důležité. Avšak nechci se v této práci zabírat detailním rozborem přepočtů vedlejších atributů a čísel, proto představím pouze z mého hlediska důležité herní mechanismy.

Pohyblivost Základní pohyblivost postavy (Basic speed) určuje rychlost postavy a jejích reakcí. Závisí na zdraví postavy a obratnosti a představuje rychlost postavy, která nemá penalizaci za nadměrné naložení. Průměrná rychlost pro lidskou rasu je 5 yd za sekundu.

Zátěž Čím větší zátěž (Eucumbrance) má postava u sebe, tím je pomalejší. Snižuje se také její rychlost cestování. Postava navíc pomaleji plave a běhá. k tomu, abychom mohli přesně vypočítat, kdy je zatížená postava penalizována a jak moc, slouží tzv. stupeň zátěže (Encumbrance level). Tato hodnota je získána porovnáním zatížení a síly postavy.

Pohyb Oproti pohyblivosti, samotná hodnota pohybu (Move) vznikne od pohyblivosti odečtením stupně zátěže. Tato hodnota je velice důležitá, protože určuje, kolik yardů za sekundu skutečně postava uběhne, kdy se přijde na tah (např. v souboji první vždy hraje postava, která má nejvyšší pohyb), ale i výši aktivní obrany (Dodge).

1.1.5 Boj

Neodmyslitelná složka každého dobrodružství je boj. Boj v GURPS probíhá v několika fázích. Každá postava jedná, když se dostane k tahu (pořadí určeno podle pohyblivosti). Každé kolo začíná výběrem akce postavy - pohyb (a pouze pohyb), změna pozice, nabití zbraně, zamíření a zaútočení. Dále může postava zvolit obranu nebo nějakou dlouhodobější akci.

Vedle ozbrojeného boje existuje ještě neozbrojený boj, který disponuje celou řadou vlastních pravidel, která zde rozebírat nebudu. Chtěl bych pouze nastínit obecný princip boje a detaily nechat již na hráčích GURPS.

Útok Postava může vést útok, pokud má připravenou zbraň. Každý útok sestává ze tří hodů: útočný hod, obranný hod a hod zranění. Útočník nejprve hodí kostkou, zda se jeho útok povedl. Pokud se povedl, obránce provede svůj hod pro zjištění, jestli se útoku vyhnul nebo ho např. vykryl štítem. Pokud útok prošel a nebyl vykryt, útočník hází znovu, tentokrát na velikost zranění, které způsobil. Útočný hod může být redukován (popř. navýšen) o různé modifikátory, např. špatná viditelnost.

Obrana Pasivní obrana je obrana poskytovaná štítem a zbrojí. Tato složka obrany je aktivní vždy - i když postava o příchozím útoku neví. Obránce může vedle pasivní obrany zahrnout do svého bránění aktivní obranu. Může si vybrat ze tří druhů aktivních obran: úhyb (Dodge), blok (Block) a odvrácení útoku zbraní (Parry). Vybraný typ obrany může přidat ke své obraně v daném kole.

1.1.6 Magie

Ani ve světě GURPS nechybí magie - neodmyslitelná složka téměř každého dobrodružství. Kouzla jsou získávána podobně jako schopnosti za osobnostní body. Každý svět má vlastnost Mana, která určuje, sílu magie v daném světě. Pokud je Mana nízká, každý čaroděj obdrží ke svým kouzlům penalizaci, pokud mana chybí, kouzla vůbec nefungují.

Čaroděj jako palivo pro svá kouzla používá svoji sílu. za každé kouzlo, které vykouzlí, si musí odečíst jeho cenu (Cost) jako body únavy (Fatique). Pokud kouzlo selže, zaplatí však pouze jedním bodem únavy.

Kouzla U každého kouzla trvá určitou dobu, než jeho efekt začne působit (Cast time). po tuto dobu se musí čaroděj na vyvolávání soustředit. Některá kouzla lze udržovat déle než je jejich doba trvání (Duration) - takové kouzlo pak stojí další body únavy.

Ve světě GURPS rozlišujeme 4 základní druhy kouzel - normální (Regular), projektily (Missile), plošné (Area) a přímé (Resisted). Projektily jsou kouzla, která čaroděj vytvoří mezi svýma rukama a pak je vrhá. Účinek plošných kouzel pokrývá oblast, kterou si může hráč libovolně zvětšit (čím větší oblast, tím větší cena kouzla). Přímá kouzla jsou čarována přímo na určitý cíl a pro svůj úspěch testují jeden z atributů cíle.

1.1.7 Ostatní

Ze všech ostatních aspektů hry podle pravidel GURPS jsem vybral ještě dvě samostatné kapitoly, o kterých bych se chtěl zmínit.

1.1.7.1 Práce

Většina postav, které se účastní dobrodružství, potřebují nějaké peníze pro financování výprav a pro živobytí. Postavy proto mají práce, které pravidelně každý týden vykonávají a mezi jednotlivými výpravami může být vsunuta např. dvouměsíční prodleva, ve které dané postavy chodily do práce a žily normálním životem. Podle stupně bohatství dané postavy se také určí, kolik hodin týdně postava pracuje a jak moc peněz je jí vypláceno.

1.1.7.2 Technologické levely

Technologický level (Tech level) je jednotka, která určuje, v době jaké technologie se odehrává příběh nebo pochází postava. Technologické levely začínají stupněm 0 (doba kamenná) a v GURPS Lite končí stupněm 15 (vynalezena teleportace). Hráči si mohou vytvořit další stupně podle svých představ. Náš reálný svět se nachází v technologickém stupni 8.

1.2 MobChar

MobChar je aplikace, jejíž účelem je usnadnit hráčům her na hrdiny správu jejich postav. Umožňuje evidenci a snadnou editaci informací o těchto postavách. Aplikace je vyvíjena studenty Fakulty informačních technologií a je zaměřená na operační systém Android.

Práce na této aplikaci začala v roce 2014, kdy v rámci předmětu Softwarový projekt 1 a Softwarový projekt 2 skupina studenů vedená Ing. Zdeňkem Rybolou vytvořila základ aplikace MobChar. Tato prvotní verze sloužila ke správě postav pouze pro hru Dračí Doupě. Poté se aplikace chopilo několik dalších studentů a v rámci bakalářských prací postupně MobChar modifikovali a dále vyvíjeli. Jakub Nižaradze oddělil implementaci Dračího Doupěte od základu, který nyní představuje jádro aplikace MobChar, ve své práci „MobChar - jádro aplikace“ [2]. Dalším studentem, který svou prací přispěl k rozvoji MobCharu byl Tomáš Pastorek, který ve své práci „Mobchar - balíček pro Dungeons and Dragons“ [3] přidal balíček pro hru Dungeons and Dragons. v současné době k aplikaci vznikají i další balíčky.

1.2.1 Jádro aplikace MobChar

Jádro aplikace MobChar slouží jako základní skelet pro ostatní balíčky. Umožňuje zobrazení seznamu postav, tvorbu nové postavy a výběr existující postavy.

Když uživatel spustí aplikace MobChar, je to právě jádro aplikace, které zprostředkovává prvotní menu. Až když uživatel vybere konkrétní balíček,

ve kterém chce vytvořit postavu, popřípadě vybere postavu k zobrazení, spustí se konkrétní balíček a převezme tvorbu obrazovek.

Pro integraci nového balíčku do celé aplikace je potřeba definovat základní aktivity, které pak jádro aplikace volá. Jmenovitě jsou to aktivity pro zobrazení seznamu postav daného balíčku, pro získání konkrétní postavy po kliknutí na postavu a aktivita pro vymazání postavy z balíčku.

1.2.2 Společná knihovna

Základní strukturou pro budování balíčku a jeho částí je knihovna MobCharu. Zde se nachází abstraktní základ pro jednotlivé části balíčku, základ pro inventář, ale i některé fragmenty a aktivity pro tvorbu uživatelského rozhraní. z této knihovny jsem se ve své práci snažil vycházet co nejvíce, pokud to okolnosti umožňovaly a pokud by v dané situaci nedocházelo ke zbytečnému nevyužití funkcionality balíčku.

1.3 Požadavky na balíček

Tato kapitola by měla nastínit funkcionality a požadavky, které jsou na balíček GURPS aplikace MobChar kladeny. Požadavky v následujících odstavcích mají původ v online pravidlech Gurps Lite [1].

1.3.1 Tvorba postavy

Balíček GURPS by měl umožňovat vytvoření nové postavy podle pravidel GURPS. Hráč je schopen zvolit si jméno postavy a její rasu. Má na výběr z přednastavené rasy člověk a vlastní rasy, u které si může zvolit libovolné jméno, popis a výchozí hodnoty čtyř primárních atributů postavy. Dále bude hráči umožněno definovat vzhled postavy, její původ a počet bodů osobnosti, se kterými bude postava začínat.

V neposlední řadě bude hráči umožněno zvolit si výchozí hodnoty atributů postavy a zároveň bude hráči pro informaci zobrazena cena těchto úprav v bodech osobnosti. na závěr bude umožněno k postavě uložit libovolný popis.

Daná postava se poté vytvoří, vložené informace se uloží a postava bude k dispozici v seznamu postav, kam se také aplikace přepne po dokončení tvorby postavy.

Při vytváření postavy je vyžadováno pouze jméno, ostatní hodnoty mohou zůstat nevyplněny. Pokud je vyžadována číselná hodnota, má každá taková definovánu výchozí hodnotu, která se použije. Pokud se jedná o textový záznam, zůstane nevyplněný. Výchozí hodnoty atributů stejně jako rasa postavy

po vytvoření postavy změnit nejdou, všechny ostatní informace jsou libovolně měnitelné i po vytvoření.

1.3.2 Zobrazení detailu postavy

Aplikace bude umožňovat zobrazení všech dostupných informací o vybrané postavě a až na výjimky editaci těchto informací. pro přehlednost budou informace sdruženy do logických celků a po těchto celcích zobrazovány na jednotlivých záložkách.

1.3.2.1 Základní informace o postavě

Aplikace umožní zobrazit základní přehled postavy, ve kterém hráč uvidí všechny informace nastavené při vytvoření postavy. Navíc budou zobrazeny hodnoty sekundárních atributů - únavy a obdrženého poškození. Dále přehled zobrazí množství peněz, kterým postava disponuje. v přehledu postavy se také bude nacházet přehled naložení a pohybu postavy, obsahující hodnoty rychlosti, aktuální naložení, stupeň zátěže postavy a hodnoty rozmezí jednotlivých stupňů zátěže.

Všechny tyto informace budou měnitelné, kromě sekce zátěže a pohybu. Všechny hodnoty této sekce budou vypočítány z atributů postavy a jiných informací, které samy o sobě budou editovatelné.

1.3.2.2 Přehled bodů osobnosti

Hráč bude mít k dispozici přehled osobnostních bodů jeho postavy. v přehledu bude zahrnutá celková bilance bodů - kolik bodů je utraceno, kolik bodů je k dispozici a kolik bodů má postava celkem. Dále bude aplikace evidovat, kolik osobnostních bodů má postava utraceno v kterých sekcích - například kolik je utraceno za kouzla, schopnosti atd.

Na stejné kartě bude aplikace obsahovat tlačítko sloužící pro přidání bodů osobnosti. Bude možné body přičíst, ale i odečíst - například když hráč zadá větší počet bodů, než původně chtěl.

1.3.2.3 Přehled rysů postavy, setů, kouzel a schopností

Aplikace bude dále evidovat seznam rysů postavy, setů, kouzel a schopností. Seznamy budou reprezentované každý vlastní záložkou, avšak jejich funkcionalita bude obdobná. u každého ze seznamů se zobrazí všechny objekty vyžadovaného typu, které jsou u postavy evidované. Každý záznam bude obsahovat základní informace o něm uvedené. pro kompletní informace o objektu bude aplikace umožňovat každý záznam v seznamu rozkliknout a zobrazit tak jeho detail.

Hráči bude umožněno objekty vytvářet. po stisknutí tlačítka pro přidání nového rysu se v každém seznamu otevře obrazovka, která uživateli umožní zvolit libovolně všechny parametry nového objektu. Pouze jméno bude vyžadováno, ostatní údaje nemusí být vyplněny (jsou libovolně editovatelné po vytvoření).

U rysů bude navíc možné evidovat typ rysu - výhoda, nevýhoda či výstřednost. při vytváření bude tato hodnota nastavitelná a i po vytvoření bude změnitelná.

Sety budou navíc v seznamu obsahovat ještě tlačítko, pomocí kterého bude moct hráč set obléknout/svléknout. po rozkliknutí konkrétního setu bude zobrazen kromě jména a popisu setu také seznam všech nalezených předmětů dané postavy, u každého z nich se bude nacházet tlačítko pro zahrnutí/vyloučení předmětu ze setu.

V seznamu schopností bude každé schopnosti přiřazena aktuální hodnota. Ta se zjistí jako nejvyšší hodnota z množiny všech výchozích hodnot dostupných pro danou schopnost a aktuální vycvičené hodnoty schopnosti. Jak výchozí hodnoty, tak aktuální hodnota budou volně editovatelné (u editace aktuální hodnoty se bude navíc zobrazovat cena v bodech osobnosti). po otevření detailu schopnosti bude navíc možné nastavit, zda má schopnost nastavenou aktuální hodnotu, či nikoli.

Seznam kouzel bude zobrazovat všechna kouzla spolu s ikonou reprezentující jejich typ. Typ lze měnit i po vytvoření kouzla. Speciálně kouzla projektilového typu (Missile) budou v seznamu zobrazovat hodnotu schopnosti vrhání kouzel, pokud bude taková schopnost nalezena v seznamu schopností postavy (schopnost je vyhledána pomocí přesné shody jména, které je „Spell throwing“). Kouzla typu „Resisted“ budou navíc zobrazovat atribut/schopnost, ze kterého se při aplikaci kouzla bude brát hodnota, proti níž bude úspěch kouzla testován.

1.3.2.4 Inventář

Další funkcionalitou, kterou bude balíček implementovat, je inventář. Hráči bude umožněno přidávat věci, batohy, zbraně, brnění a štíty. Dále bude možné přepínat mezi batohy a zanořovat je do sebe, což vytvoří něco jako „adresářovou strukturu“. Hráč bude moct umisťovat předměty do libovolných batohů.

Každý předmět bude v seznamu předmětů zobrazovat základní údaje a bude doplněn ilustrací. po rozkliknutí konkrétního předmětu aplikace nabídne detailní obrazovku, ve které budou k vidění všechny informace o předmětu. Bude zde také možnost nastavit, zda je daný předmět postavou nošen či nikoliv. k dispozici bude také tlačítko v menu, které vyvolá novou obrazovku,

ze které bude možné změnit jakoukoli informaci o předmětu. Předměty budou seřazeny podle typu, kde batohy budou pro snadnou orientaci ve struktuře inventáře řazeny jako první, dále budou obecné věci, brnění a štíty a nakonec zbraně.

1.3.2.5 Přehled bojových statistik

Balíček GURPS bude také zobrazovat přehled bojových statistik postavy. Přehled bude nejprve zobrazovat nejdůležitější hodnoty celého přehledu, což je hodnota schopnosti používání vybrané zbraně, pasivní obrana a hodnoty aktivních obran (spolu s číselnou reprezentací hodnoty bude také zobrazen vzorec, podle kterého byla hodnota vypočtena). v sekci útoků pak bude rozbalovací seznam zbraní, ze kterých si hráč může vybrat tu, pro kterou chce zobrazit útočné informace. Tyto se po vybrání zobrazí pod boxem. Stejným principem se budou vybírat a zobrazovat i informace o brněních a štítech.

1.4 Doménový model

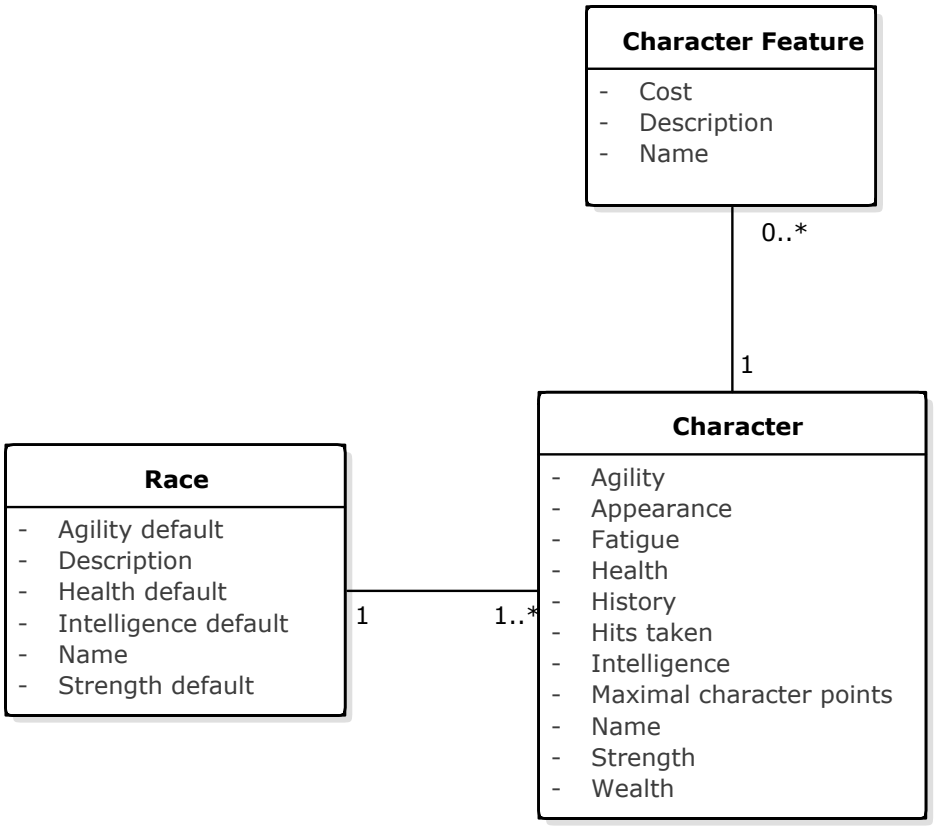
Doménový model slouží k zachycení základních entit aplikace a jejich vzájemných vztahů. Je platformově nezávislý, tzn. že není určen pro konkrétní programovací jazyk, ale slouží spíše jako schéma. Entity v doménovém modelu nemají metody ani datové typy u atributů. Svůj doménový model jsem rozdělil do třech ucelených částí.

1.4.1 Profil postavy

První část ukazuje reprezentaci postavy a jejích vlastností 1.1.

Postava Nejzákladnější entitou v celé aplikaci MobChar je herní postava (Character). u postavy zaznamenáváme její základní informace - jméno (Name), vzhled (Appearance), původ (History) a popis postavy (Description). Dále evidujeme čtveřici primárních atributů: sílu (Strength), obratnost (Agility), inteligenci (Inteligence) a zdraví (Health). s těmito atributy jsou spojeny dva druhotné atributy, které vyjadřují úbytek jednoho z primárních atributů. Únavu (Fatique) postava získá náročnou fyzickou aktivitou, přičemž snižuje Sílu. Druhým atributem je obdržené poškození (Hits taken), který snižuje Zdraví. v poslední řadě je u každé postavy uveden maximální počet bodů osobnosti (Maximal character points), který udává, kolik maximálně může daná postava utratit osobnostních bodů, a bohatství postavy.

Rasa Každá postava je členem rasy - k zaznamenání této skutečnosti slouží entita rasa (Race). Tato entita obsahuje název (Name), popis rasy (Description) a informace o tom, jaké jsou výchozí hodnoty základních atributů pro každou postavu dané rasy.



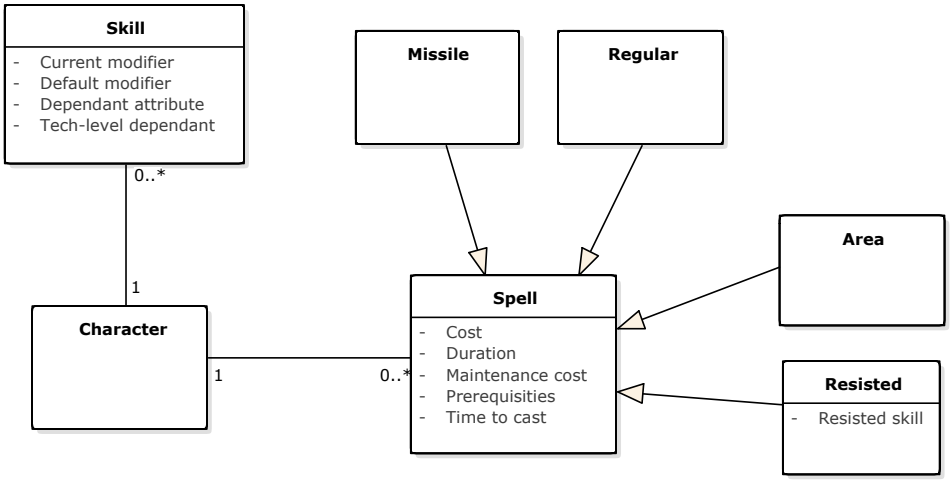
Obrázek 1.1: Doménový model postavy.

Rysy postavy Rys postavy (Character feature) je entita obsahující informace o názvu rysu (Name) a cenu osobnostních bodů (zpravidla kladná pro výhody a záporná pro nevýhody, různá pro výstřednosti) (Cost). v poslední řadě obsahuje detailní popis vlastnosti (Description).

Rys postavy se dělí na tři podkategorie - výhody (Advantage), nevýhody (Disadvantage) a výstřednosti (Quirk). v balíčku však nehraje zásadní roli rozlišování mezi jednotlivými typy rysu, proto budou všechny rysy reprezentovány pouze jedním objektem a typ rysu bude evidování pomocí výčtového typu enum.

1.4.2 Schopnosti a kouzla

Druhá část popsána na obrázku 1.2 znázorňuje základní reprezentaci herní postavy společně se schopnostmi a kouzly, které se může postava naučit. Entita Character představuje entitu z minulého modelu.

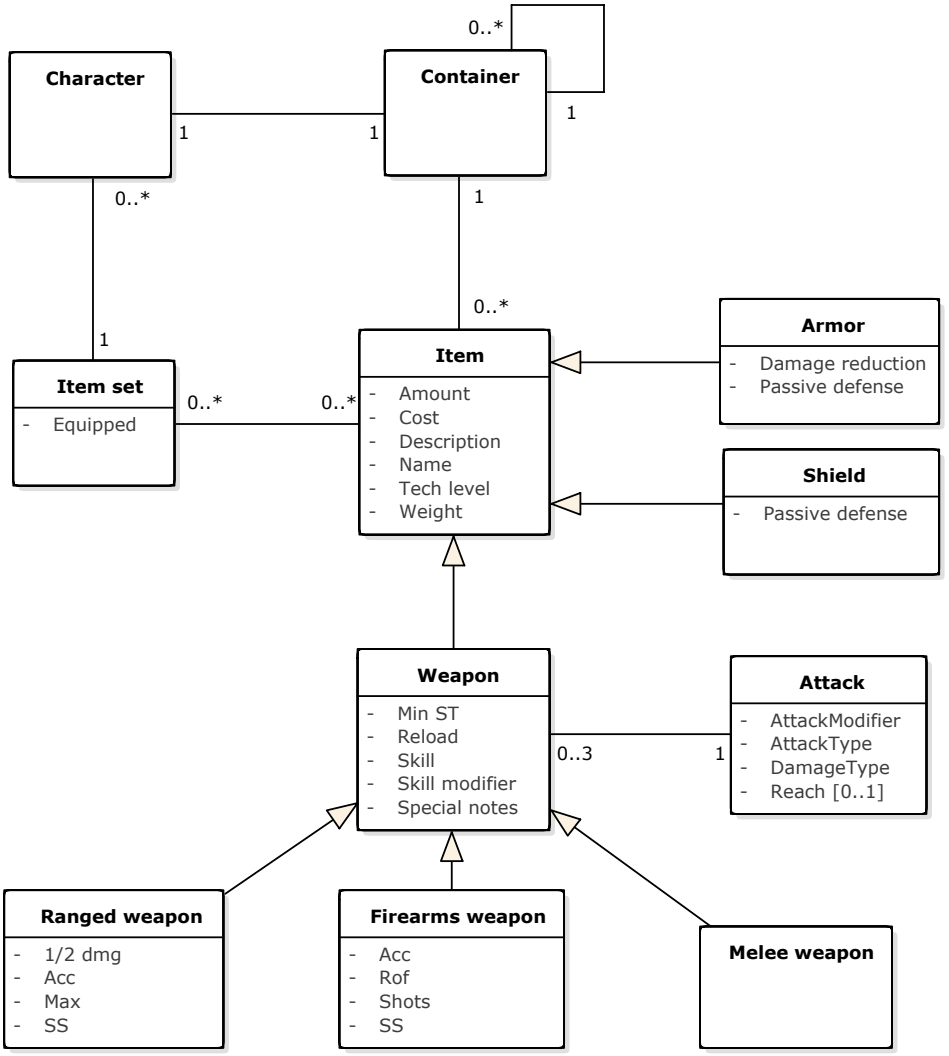


Obrázek 1.2: Doménový model schopností a kouzel

Schopnost Další důležitou entitou je schopnost (Skill). Vzhledem k pravidlům GURPS, které jsou velmi obecné, lze vymyslet nespočetné množství různých schopností, které může postava ovládat. Schopnosti rozdělujeme na 4 kategorie dle obtížnosti (Difficulty) - lehké, průměrně těžké, těžké a velmi těžké. Naučení vyššího levelu schopnosti s lehkou obtížností stojí méně bodů osobnosti než naučení stejného levelu těžké schopnosti. Rozlišujeme dva druhy - fyzické (Physical) a mentální (Mental) schopnosti.

Úroveň znalosti (Dependant atribute) určité schopnosti závisí na primárním atributu postavy - zpravidla na obratnosti u fyzických a inteligenci u mentálních schopností. Aktuální level schopnosti se vypočítá jako součet atributu, na kterém je schopnost závislá, a modifikátoru (Current modifier). Některé schopnosti jsou běžné, což znamená, že i postava, která nikdy danou schopnost specificky netrénovala, ji ovládá na základní úrovni. z tohoto důvodu mají takové schopnosti definovaný výchozí modifikátor (Default modifier). Každá postava pak automaticky takovou schopnost ovládá na úrovni závislý atribut + výchozí modifikátor. Některé schopnosti jsou pak závislé na levelu technologické vyspělosti (Tech level dependant).

Kouzlo Jako v každém dobrém fantasy prostředí, i ve světě GURPS mohou postavy ovládat magii. u kouzla vedeme dobu trvání (Duration) jeho efektu, dobu, za kterou je kouzlo vykouzleno (Time to cast), dále pak seznam kouzel, která je nutné znát, aby se postava mohla dané kouzlo naučit (Prerequisities). Každé kouzlo má také svoji cenu (Cost), která se vyhodnotí přidáním bodů Únavy. Některá kouzla lze udržovat delší dobu, než je jeho doba trvání - to však znamená, že kouzlo bude způsobovat více bodů únavy, čím



Obrázek 1.3: Doménový model předmětů a inventáře

déle ho bude daná postava udržovat. Tato informace je ukládána jako Cena pro udržení (Maintenance cost).

Rozlišujeme 4 kategorie kouzel: normální (Regular), dále projektily (Missile), které jsou postavou vrhány, a plošná kouzla (Area), která postihují zvolenou oblast. Plošná kouzla mají tím větší cenu, čím větší plochu kouzlo zasáhlo. Poslední kategorií kouzel jsou přímá kouzla (Resisted), kterým se může cíl kouzla ubránit.

1.4.3 Inventář a předměty

Poslední ucelenou kategorií jsou inventář a předměty na obrázku 1.3. Postava může na svých cestách nacházet různé předměty, od obyčejných, které nenajdou využití, až po různá brnění či artefakty nevyčíslitelných hodnot. Postava může maximálně nést třicetinásobek své Síly, a to jen krátkodobě. Maximální dlouhodobější zátěž postavy je dvacetinásobek síly. v závislosti na současném naložení také dostává body zátěže (Encumbrance). Čím větší váhu postava nese, tím nižší je pak její pohyblivost.

Jedna z důležitých informací o předmětu je informace, jestli je předmět právě nošený postavou či nikoli. Pokud postava daný předmět nese, znamená to, že může využívat jeho bonusy, pokud se předmět nachází v setu, který je nasazen. Pokud naopak nesen není, nezatěžuje postavu svou hmotností (ale postava ho má stále evidovaný, což se uplatní například když si postava ve městě nechá opravit své brnění a mezitím se pohybuje bez něj; stále je však evidováno v balíčku, aby mohlo dojít k jeho jednoduchému navrácení a případnému používání).

Entita Character opět zastupuje entitu popsanou v prvním doménovém modelu.

Kontejner Entita kontejner (Container) reprezentuje možnost postavy mít u sebe předměty. u každé postavy je automaticky evidován kořenový kontejner, který sdružuje všechny evidované předměty. Hráč si může vytvořit další kontejnery, které však mají spíše funkci organizační - pravidla GURPS Lite neberou v úvahu kapacitu kontejnerů.

Předmět Základní entitou v této části doménového modelu je entita Předmět (Item). Vše, co postava získá z výprav i kdekoli jinde je instancí entity Předmět. Tato entita představuje nejobecnější vrstvu systému předmětů, mezi konkrétnější druhy předmětů patří zbroje, štíty a zbraně.

U každého předmětu zaznamenáváme jeho název (Name), cenu (Cost), váhu (Weight) a level technologické vyspělosti (Tech level), od které je předmět dostupný, což je nepovinný údaj. Dále pak nechávám prostor hráči pro vyplnění jakéhokoli popisu a přiblížení předmětu jako atribut popis (Description).

Zbroj Prvním konkrétnějším druhem předmětů jsou zbroje (Armor). Zbroj slouží k obraně proti většině druhů útoků vedených proti postavě. Může se jednat od kožené lehké zbroje až po těžkou plátovou zbroj nebo dokonce zbroje z budoucnosti. u každé zbroje je uvedena pasivní obrana (Passive defense) a redukce zranění (Damage reduction).

Štít Dalším předmětem, se kterým se pravděpodobně každá postava světa GURPS setká, je štít (Shield). Jedná se o další důležitou složku obrany postavy proti příchozím útokům. Avšak narozdíl od zbroje neposkytuje redukci zranění, ale výrazně snižuje šanci zasáhnout jeho nositele (prostřednictvím pasivní obrany).

Zbraň Rozsáhlou podmnožinou předmětů jsou zbraně (Weapon). Každá zbraň vyžaduje ke svému použití nějakou schopnost na určitém levelu. Tyto hodnoty jsou reprezentovány atributy schopnost (Skill) a modifikátor schopnosti (Skill modifier), o který je cílový atribut zvýšen/snížen a tím získána výsledná požadovaná hodnota pro použití zbraně. Každá zbraň dále vyžaduje určitý level síly (Min ST = minimal strength), aby mohla postava se zbraní zacházet bez postihů (obecné pravidlo zní čím těžší zbraň, tím větší síla je potřeba, ale neplatí vždy). Atribut nabití (Reload) pak značí, kolik tahů je třeba nabíjet zbraň, než může být znovu použita. Jelikož mají zbraně různé speciální schopnosti, podmínky a poznámky, vytvořil jsem k tomuto účelu jednotný atribut speciální poznámky (Special notes).

Zbraň pro boj na blízko Základní typ zbraně jsou zbraně pro boj na blízko (Melee weapon). Jediným atributem je dosah zbraně (Reach). Ten se však v pravidlech GURPS lite nevyužívá, je u zbraní uveden pouze za účelem budoucí kompatibility s rozšiřujícími pravidly. Příkladem takových zbraní jsou meče, kopí, halapartny, díky, nože a spoustu dalších.

Střelné zbraně Druhým typem zbraní jsou střelné zbraně (Ranged weapon). Mezi tyto zbraně patří luky, kuše, praky, nikoliv však zbraně palné, které tvoří samostatnou skupinu. u střelných zbraní zaznamenáváme 4 druhy dostřelů, avšak v pravidlech GURPS lite má význam pouze dostřel Max - maximální dostřel zbraně.

Palné zbraně Poslední skupinou zbraní jsou zbraně palné (Firearms weapon). u palných zbraní evidujeme nové atributy - kadenci (RoF - rate of fire) a počet nábojů (Shots). Řadíme mezi ně pistole, pušky, brokovnice, ale i lazery.

Sety Jelikož ve světě GURPS jsou věci, které má postava na sobě, a věci, které pouze přenáší, evidované pouze v inventáři, musel jsem zavést způsob, jak odlišit právě nošené věci od ostatních (např. pro vypočítání obraných a útočných bonusů). k tomu využiji setů. Set sestává z jedné nebo více věcí a zaznamenáváme u něj informaci o tom, jestli je právě nošen nebo ne. Postava může nosit libovolný počet setů najednou - zde je nechán prostor hráči, aby sám ohlídal, ze kterých předmětů se mu bonusy započítávají. z nošeného setu pak postava získává obrané a útočné bonusy.

Kapitola 2

Návrh

Tato kapitola se zabývá návrhem modulu aplikace MobChar pro hru GURPS. Kapitola bude pojednávat o zvoleném databázovém systému, dále o architektuře balíčku a bude popisovat jednotlivé vrstvy balíčku.

2.1 Databázový model

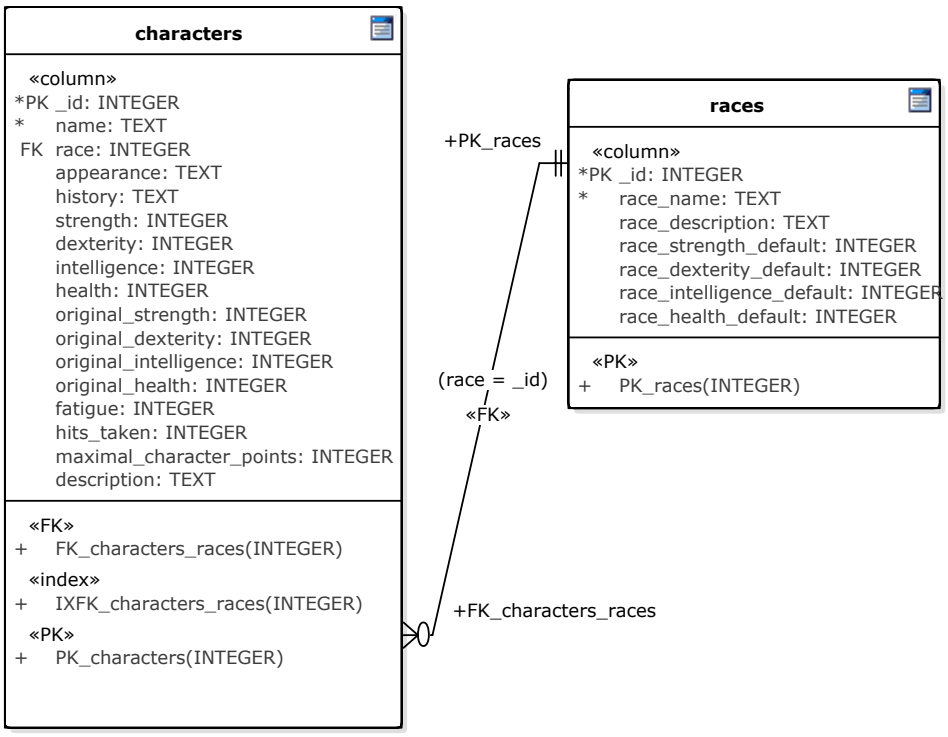
Databázový model byl navrhnut s ohledem na jednoduchost a nízkou provázanost databází. Celý databázový model jsem rozdělil na několik souvislých celků. v modelech, kde je uvedena pouze zkrácená verze tabulky "characters", se vždy jedná o plnou tabulku, popsanou na diagramu 2.1. Cizí klíče databází jsou realizovány v rámci programu, avšak pro úplnost a smysl diagramů jsem tyto cizí klíče zaznamenal do diagramů též. Každá tabulka má k dispozici sloupec _id, který tvoří její primární klíč.

2.1.1 Postava

První a zároveň stěžejní částí databázového modelu je část popisující detail postavy a její rasy 2.1. Pokud není řečeno jinak, každá tabulka představuje stejnojmennou entitu doménového modelu.

Tabulka characters Tato tabulka slouží k uložení základních informací o postavě. Jejím primárním klíčem je sloupec id, jehož hodnota slouží v dalších tabulkách jako základní informace, na základě které se přiřazují jednotlivé informace ke konkrétní postavě.

Do tabulky jsou ukládány základní informace o postavě - jméno a popis. Dále lze do tabulky uložit vzhled a historie postavy. Tabulka také obsahuje sloupec do kterého je ukládáno id rasy postavy.



Obrázek 2.1: Databázový model - část 1

Převážnou část tabulky však zaujímají informace o jednotlivých atributech postavy. Přestože jsou atributy postavy v doménovém modelu reprezentovány vlastní třídou, rozhodl jsem se nevytvářet pro ně specifickou tabulku. Místo toho jsem zvolil uložení hodnot přímo do tabulky o postavě.

Tabulka tedy eviduje set aktuálních hodnot atributů (Strength, Dexterity, Intelligence, Health), dále také druhotné atributy (Fatigue a Hits taken). Nakonec je nutné u postavě ukládat také původní hodnotu čtyř základních atributů, kterou si uživatel nastavil při vytváření postavy (z těchto hodnot se později vypočítávají ceny atributů v bodech postavy.

Na závěr se do tabulky ukládá hodnota maxima bodů postavy, tzn. kolik maximálně může hráč do své postavy investovat bodů postavy.

Tabulka races Tato tabulka reprezentuje rasu a uchovává základní informace, které o rase potřebuje aplikace evidovat. Tabulka obsahuje sloupec _id, který představuje primární klíč tabulky, dále pak id postavy. v poslední řadě pak obsahuje výchozí hodnoty čtyř primárních atributů.

Zatímco standardní reprezentace by byla uložit do tabulky rasy jako cizí klíč id postavy, ve svém balíčku jsem zvolil reprezentaci opačnou. Důvodem byla myšlenka, že většina postav vytvořených v balíčku GURPS bude pravděpodobně stejné rasy - člověk. Pokud bych ukládal cizí klíč do tabulky rasy, musel bych pro každou postavu vytvořit rasu znovu, což sem považoval při vysoké pravděpodobnosti pouze jedné rasy zbytečné. Reprezentace cizím klíčem v tabulce postavy mi umožňuje vytvořit pouze jeden záznam rasy člověk, přičemž pokud bude tuto rasu sdílet více postav, zvolená reprezentace umožní udržovat v databázi pouze jeden záznam o této rase.

Při vytváření rasy se aplikace podívá do databáze, jestli již zvolená rasa existuje, a pokud ano, pouze přiřadí její id. v opačném případě rasu do databáze zapíše. při mazání postavy aplikace nejprve zjistí, u kolika postav se daná rasa nachází, a smaže se pouze pokud nebyla nalezena u více než jedné postavy.

Aplikace umožňuje tvorbu vlastní rasy s nastavitelnými parametry. pro vlastní rasy platí totéž co pro rasu přednastavenou - v databázi se vždy bude nacházet maximálně jeden záznam dané rasy sdílený mezi všemi postavami této rasy.

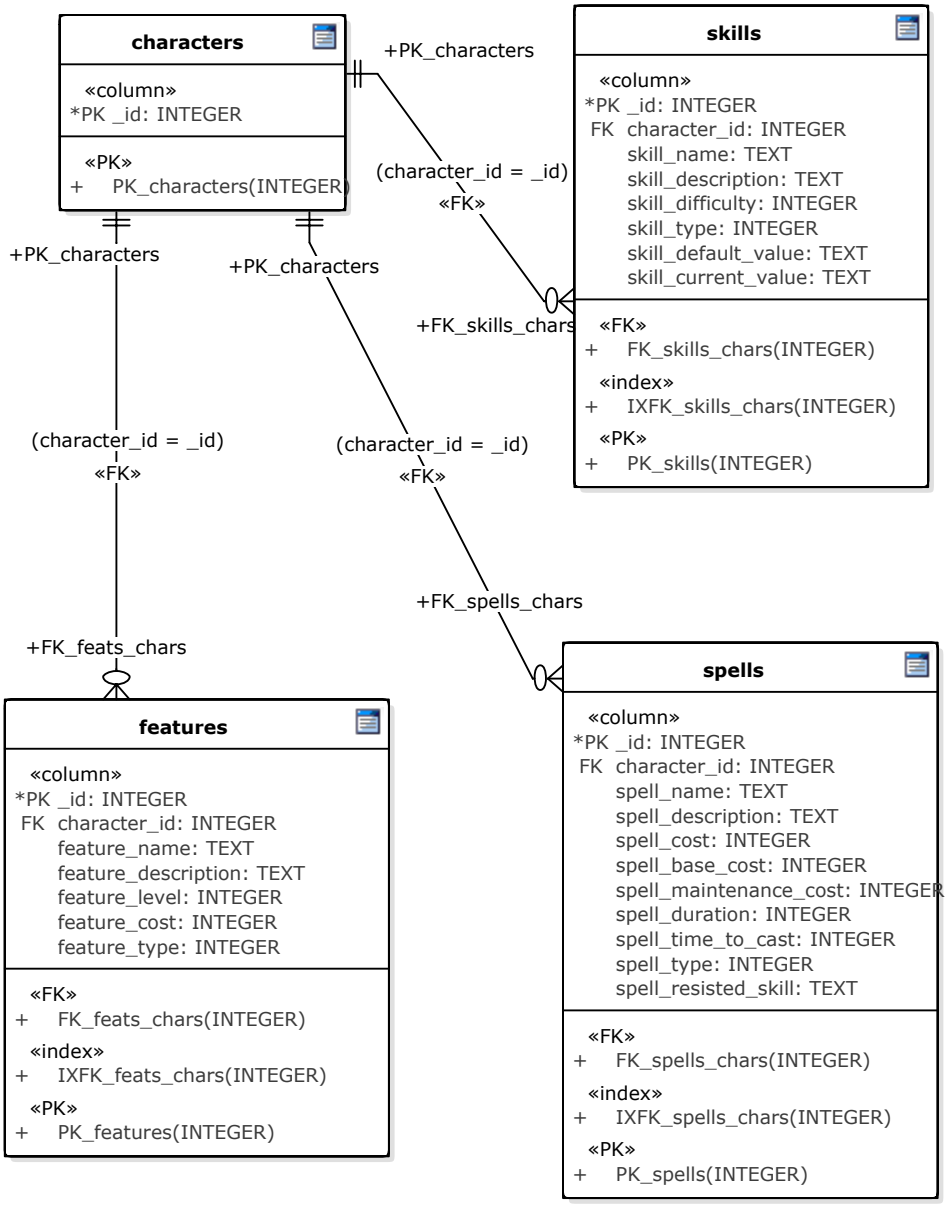
Rasa je vyhodnocena s jinou rasou jako stejná, pokud se shodují všechny informace obou ras.

2.1.2 Rysy postavy, schopnosti a kouzla

Do druhé části databázového modelu spadají další informace o postavě - rysy, schopnosti a kouzla - zobrazené na obrázku 2.2. Ačkoli jsou tyto kategorie navzájem nezávislé, jejich vztah k entitě postavy je podobný, proto jsou uvedeny jako samostatný diagram.

Tabulka features Tabulka reprezentuje skupinu rysů postavy - výhody, nevýhody a výstřednosti. o každém rysu je v tabulce uchovávána informace o jménu rysu, jeho popisu, levelu daného rysu, jeho ceně a typu. Typ rysu (výhoda, nevýhoda, výstřednost) je reprezentován pomocí čísla a po načtení z databáze převeden na výčtový typ enum. Cena rysů je ukládána ve formě čísla, je tedy zcela na uživateli, aby ohlídal, že zadává korektní hodnoty - aplikace ho v těchto ohledech nijak neomezuje.

Tabulka skills V této tabulce jsou ukládány schopnosti postavy. Jsou ukládány informace o id postavy, ke které schopnost patří, dále jméno schopnosti a popis. Další uchovávanou hodnotou je obtížnost schopnosti a typ schopnosti. Tyto dvě hodnoty jsou také uchovávány číselně, ale v programu jsou reprezentovány výčtovým typem. Posledními dvěma sloupci v tabulce jsou skill_default_value a skill_current_value. v GURPS se tyto hodnoty skládají ze dvou částí - atribut, ke kterému se vztahují, a modifikátor, který se



Obrázek 2.2: Databázový model - část 2

připočte k hodnotě vazebního atributu a tím se získá výsledná hodnota schopnosti. Abych tyto informace zaznamenal, ale zároveň aby nebyla databázová struktura zbytečně složitá, zvolil jsem textovou reprezentaci těchto hodnot.

Textová hodnota se skládá ze zkratky vazebního atributu a modifikátoru. na tento formát jsou při ukládání hodnoty převedeny a také zpět převedeny při čtení z databáze. Navíc výchozích hodnot dané schopnosti může být víc. v rámci udržení jednoduchosti struktury dat v databázi a s přihlédnutím k faktu, že zřídkakdy má schopnost více než dvě výchozí hodnoty, rozhodl jsem se, že nebudu vyčleňovat další tabulku pro vztah Schopnost - Výchozí hodnota, ale pouze hodnoty v textovém formátu spojím čárkou a zapíšu jako jeden textový řetězec.

Tabulka spells V tabulce spells jsou uchovávány informace o kouzlech postavy. Jako v ostatních tabulkách se zde nachází sloupec _id označující id kouzla a character_id označující id postavy, které kouzlo náleží. Sloupce name a description uchovávají informace o jménu a popisu kouzla. Ostatní sloupce obsahují další informace o kouzlu v číselném formátu.

2.1.3 Inventář

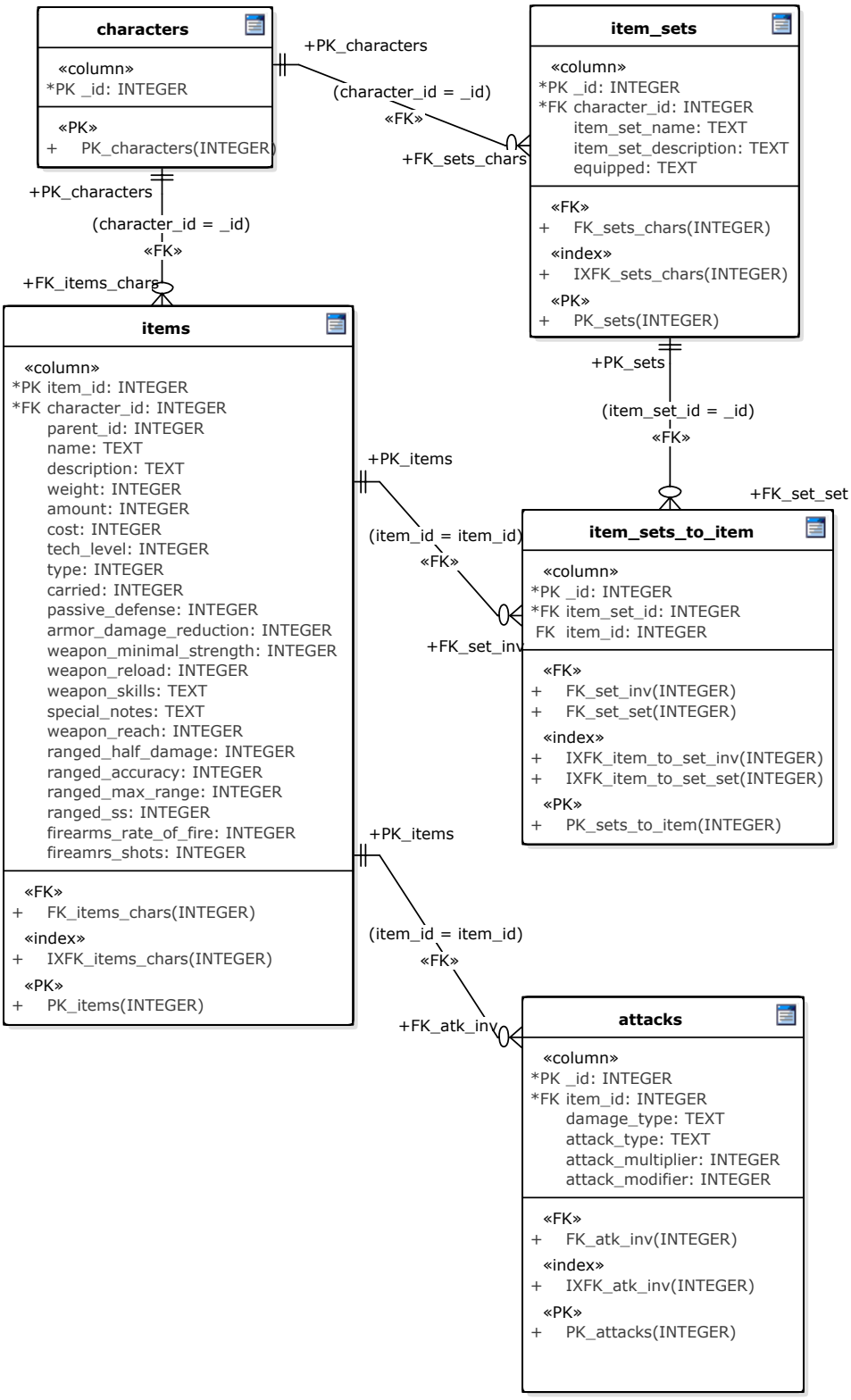
Poslední sekcí databázového modelu je inventář 2.3. Tato část zahrnuje tabulky předmětů, setů a útoků zbraní.

Tabulka items Do tabulky items jsou ukládány informace o předmětech postavy. Balíček eviduje více druhů předmětů, ale v rámci jednoduchosti databázového modelu jsou všechny informace sdruženy do jedné robustní tabulky. Každý předmět, který bude do tabulky uložen, pak využije pouze ty sloupce, které se vztahují k danému typu předmětu.

Každý předmět obsahuje id kontejneru, ve kterém se nachází; pokud se nachází v kořenovém kontejneru, bude do tohoto sloupce uloženo id -1. u předmětu evidujeme jméno, popis, hmotnost, množství, cenu, typ, technologický level a informaci, zda jej má postava u sebe, či nikoli.

Dále pak tabulka obsahuje informace specifické pro zbroje, štíty a zbraně tak, jak jsou popsány v doménovém modelu. za zmínku stojí sloupec weapon_skill, ve kterém jsou uloženy atributy, ale také schopnosti, jejichž hodnota je použita k vypočtení schopnosti ovládat danou zbraň. Tyto údaje jsou podobně jako v tabulce schopností evidovány ve formátu textového řetězce, kde jsou jednotlivé atributy/schopnosti odděleny čárkou.

Tabulka item_sets Tabulka item_sets obsahuje informace o setech věcí. Klíčovou informací u setu je, zda je nošen, či není.



Obrázek 2.3: Databázový model - část 3

Tabulka item_sets_to_item Jelikož jsou předměty a sety ve vztahu m:n, bylo potřeba vytvořit tabulku, která rozloží vztah m:n na dva vztahy 1:n. Touto tabulkou je item_sets_to_item. Zde je zaznamenáno, které předměty se nachází ve kterých setech.

Tabulka attacks Každá zbraň disponuje jedním nebo více druhy útoků. Každý útok se skládá z typu útoku a typu poškození (reprezentovány číslem v databázi, v programu jsou pak tyto čísla převedeny na výčtový typ enum). Poslední tabulka slouží právě k evidenci útoků jednotlivých zbraní. Cizím klíčem zde je id zbraně. Dále si u útoku držíme informace o modifikátoru a multiplikátoru poškození.

2.2 Architektura balíčku

Pro balíček GURPS aplikace MobChar jsem se rozhodl použít třívrstvou architekturu, jejíž model je zobrazen na obrázku 2.4. Mezi důvody patři například implementace knihovny, která obsahuje základní DAO objekty, na které je vhodné navázat. Dalším důvodem je oddělení souborů pro práci s datovou vrstvou, logickou (business) vrstvou a samotnou prezentační vrstvou.

Přestože některé objekty balíčku jsou pouze ukládány do databáze a načítány zpět bez přílišného upravování, každý objekt doménového modelu (kromě GURPSAttack) má svůj objekt v logické vrstvě, který zprostředkovává jeho komunikaci s datovou vrstvou.

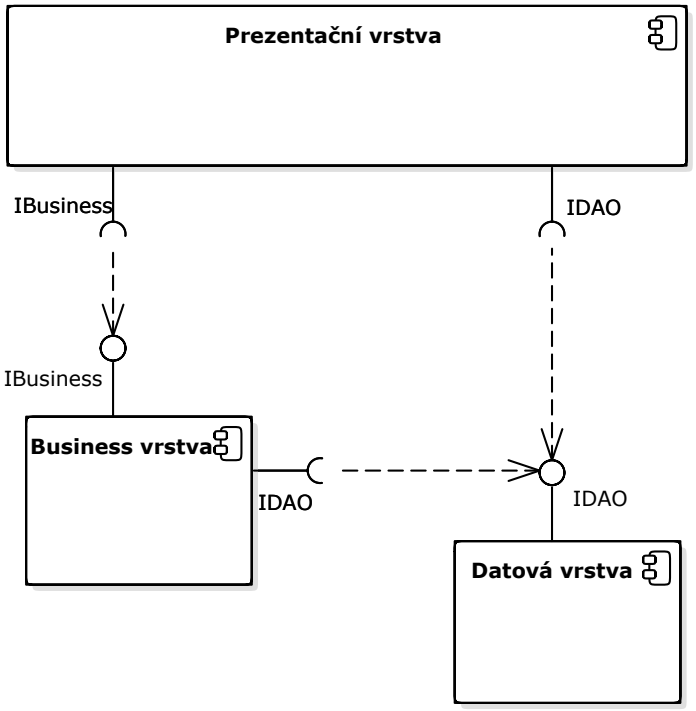
2.3 Datová vrstva

Datová vrstva slouží k pracování s datovými strukturami a k zajištění perzistence dat datových struktur do databáze. v datové vrstvě najdeme datové struktury a třídy, které s nimi manipulují.

2.3.1 Datové struktury

V kapitole Datové struktury popíšu třídy pracující s daty balíčku. Většina datových struktur použitých v balíčku GURPS rozšiřuje struktury poskytnuté knihovnou, některé struktury by však nevyužily napojení na knihovnu. Popis datových struktur jsem rozdělil do dvou částí - struktury rozšiřující knihovnu a struktury vlastní.

Datovou strukturou se rozumí třída, která popisuje některou z entit aplikace spolu s jejími atributy a metodami. Atributy jsou zde reprezentovány na úrovni konkrétních datových typů či kolekcí.



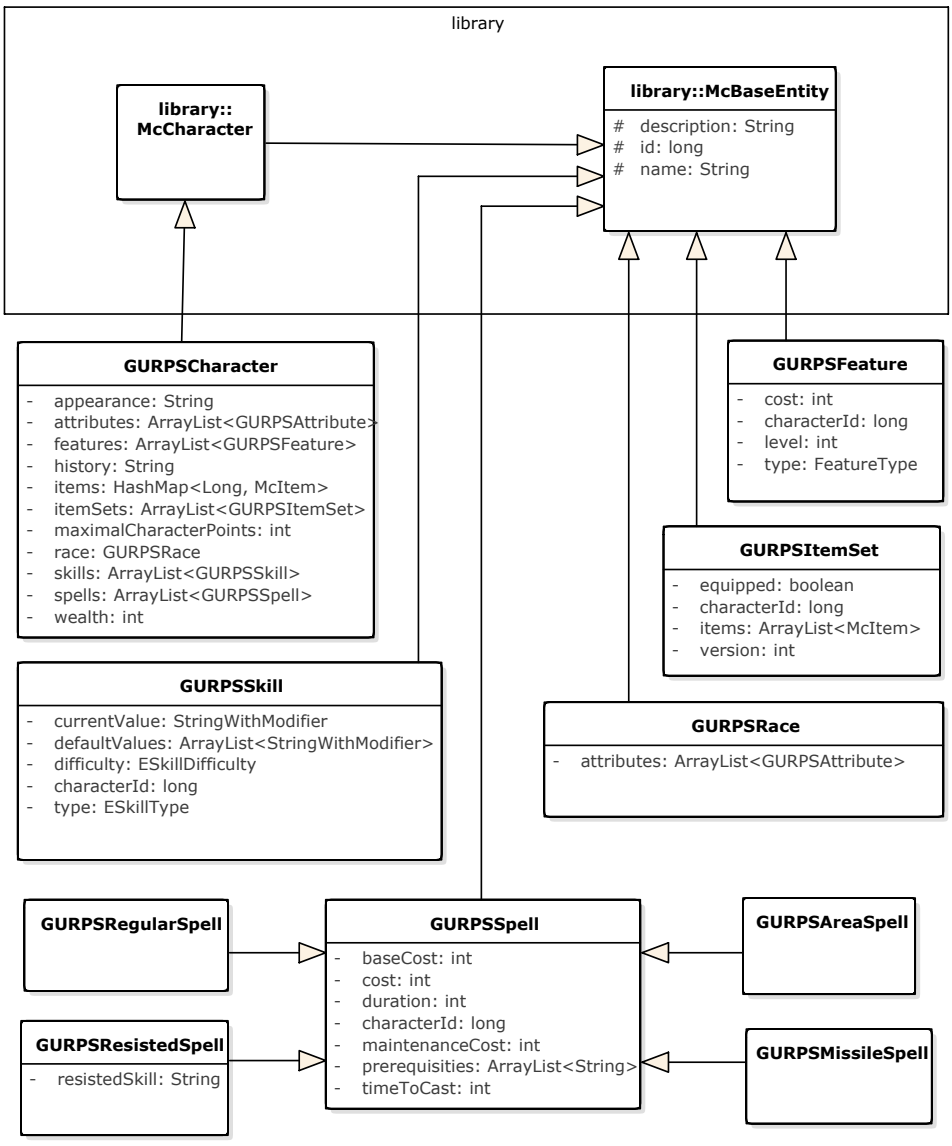
Obrázek 2.4: Model třívrstvé architektury

Všechny entity vytvořené pro balíček GURPS začínají prefixem "GURPS", aby se daly lehce odlišit od ostatních objektů v aplikaci MobChar.

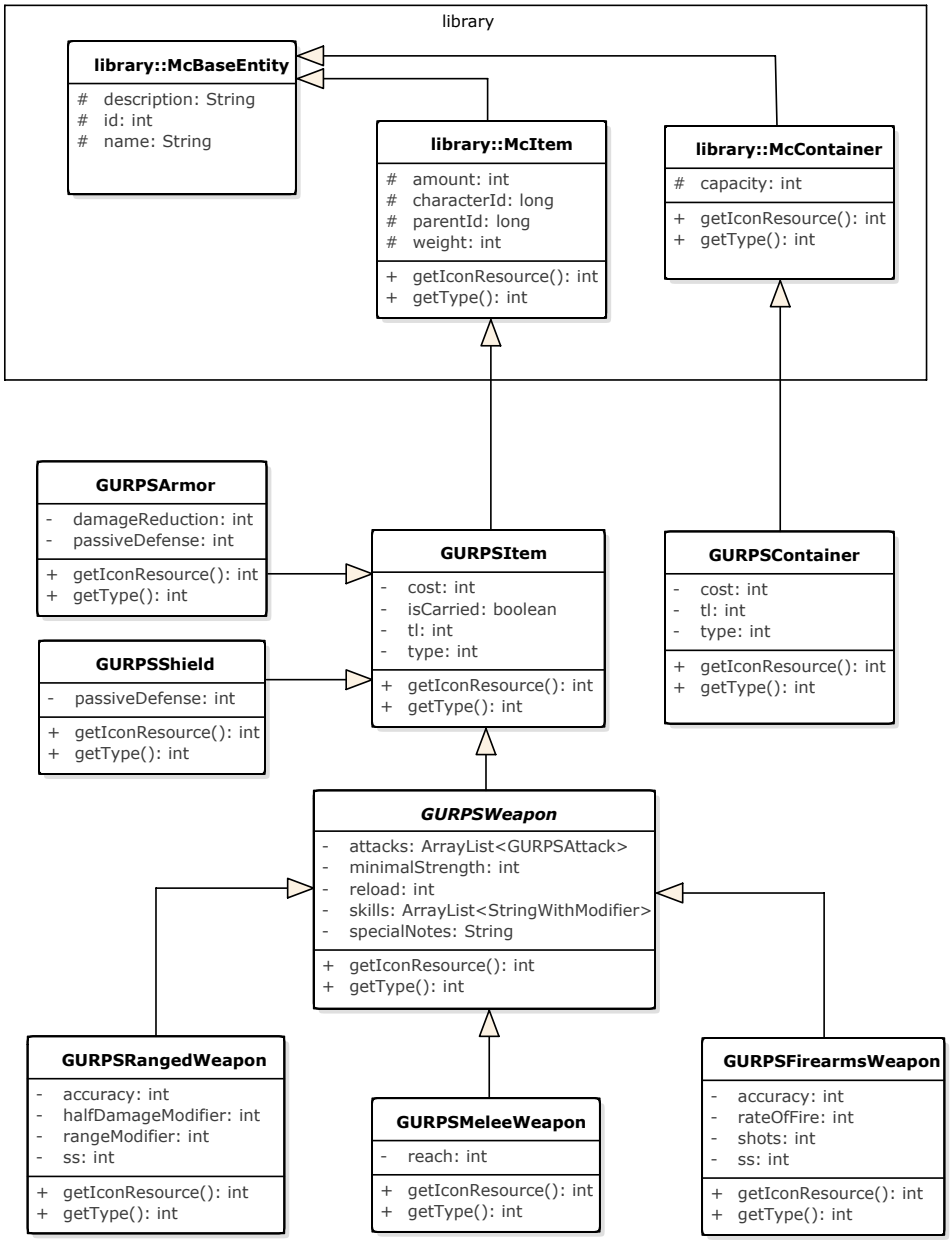
2.3.1.1 Struktury rozšiřující knihovnu

Základní entitou celé aplikace MobChar je entita McBaseEntity. Obsahuje atributy jméno, popis a id (které se také zpravidla používá k uložení entity do databáze). Většina entit balíčku GURPS vychází právě z této entity, avšak v některých případech by entita nevyužila kombinaci názvu a popisu, proto z McBaseEntity nevychází. Dalšími entitami v knihovně jsou McItem a McContainer, které rozšiřují McBaseEntity. Tato dvojice tvoří základ pro implementaci inventáře. Základ pro reprezentaci postavy pak tvoří McCharacter. Vztahy mezi entitami knihovny a balíčku GURPS popisují diagramy 2.5 a 2.6.

GURPSCharacter GURPSCharacter je první třídou na diagramu 2.5. Třída eviduje základní informace o postavě. Hlavní částí těchto informací je čtveřice primárních a dvojice sekundárních atributů.



Obrázek 2.5: Datové struktury rozšiřující knihovnu MobChar - 1. část



Obrázek 2.6: Datové struktury rozšiřující knihovnu MobChar - 2. část

GURPSRace GURPSRace eviduje informace o rase postavy. u každé rasy evidujeme především čtveřici výchozích hodnot primárních atributů - každá postava dané rasy pak přejímá tyto výchozí hodnoty jako své vlastní výchozí.

GURPSFeature Třída GURPSFeature ukládá informace o rysu postavy - výhodě, nevýhodě či výstřednosti. Tento typ rysu je reprezentován výčtovým typem, avšak ukládán je v číselném formátu.

GURPSSkill Další třídou rozšiřující knihovní třídu McBaseEntity je třída GURPSSkill. Třída slouží k evidování schopností postavy a jejich výchozích a aktuální hodnoty.

GURPSItemSet Třídou sloužící ke správě nošených věcí je třída GURPSItemSet. Hlavní informací v této třídě je boolean flag equipped, který značí, zda je set používán či nikoli.

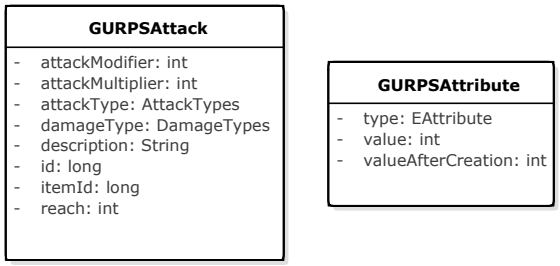
GURPSSpell Poslední třídou diagramu 2.5 je třída GURPSSpell. Tato třída reprezentuje abstraktní entitu kouzla, které je dále rozšířeno čtyřmi podtřídami - GURPSRegularSpell, GURPSResistedSpell, GURPSMissileSpell a GURPSAreaSpell. Každá z podtříd implementuje abstraktní metodu getIconResource() nacházející se ve třídě GURPSSpell.

GURPSItem První a základní třídou spadající do kategorie inventář je třída GURPSItem zachycená na diagramu 2.6. Představuje výchozí třídu pro reprezentaci předmětu v balíčku GURPS. Obecný blíže nespecifikovaný předmět může být vytvořen a reprezentován jako instance této třídy. Tato třída také přetěžuje metody getIconResource() a getType(). První jmenovaná metoda slouží k vykreslení odpovídajícího obrázku v inventáři, druhá identifikuje, o jaký typ předmětu se jedná. Zároveň tyto metody přetěžují všechny třídy vycházející z této.

GURPSContainer Základní třídou pro správu předmětů je třída GURPSContainer vycházející z knihovní třídy McContainer. Třída slouží pro správu a sdružování věcí v inventáři. v GURPS balíčku není atribut capacity využit, z objektu GURPSCoitainer se tedy stává spíše administrační prostředek pro hráče.

GURPSArmor a GURPSShield GURPSArmor a GURPSShield jsou třídy vycházející z GURPSItem a reprezentující brnění a štíty.

GURPSWeapon Abstraktní třída GURPSWeapon slouží jako základ poměrně rozsáhlého systému zbraní ve světě GURPS.



Obrázek 2.7: Vlastní datové struktury

GURPSMeleeWeapon, GURPSRangedWeapon a GURPSFirearmsWeapon Třída GURPSMeleeWeapon reprezentuje zbraň pro boj na blízko. Další dvě třídy popisují zbraně na dálku a ukládají informace o jejich dostřelech.

2.3.1.2 Vlastní datové struktury

V balíčku zbývají ještě dvě nepopsané třídy, které se mi nepodařilo napojit na rozhraní poskytnuté knihovnou. Hlavní motivací pro odchýlení se od napojení na knihovnu bylo nevyužití poskytovaných atributů jména a popisu. Entity nemají jednoznačný název nebo nepotřebují evidovat popis. Jedná se konkrétně o třídy GURPSAttribute a GURPSAttack, zobrazené na obrázku 2.7. Tyto třídy tedy nevycházejí z knihovní entity McBaseEntity.

GURPSAttribute Tato třída reprezentuje atribut - primární či sekundární. Uchovává informaci o aktuální hodnotě atributu a hodnotě, kterou si hráč nastavil při tvorbě postavy.

GURPSAttack Poslední třídou je GURPSAttack. Tato třída reprezentuje a ukládá útoky zbraní.

2.3.2 Manipulace s daty

V balíčku GURPS je manipulováno s daty pomocí DAO (Data Access Object) tříd. Tento přístup byl připraven již v knihovně, proto byl logickou volbou i pro rozšiřující balíček.

DAO DAO soubory slouží k obalení práce s databází. Cílem DAO je poskytnout aplikaci rozhraní pro práci s daty, ale zapouzdřit implementaci těchto služeb. DAO tedy zajišťuje ukládání dat do úložiště a jejich opětovnou extrakci. DAO jako celek se skládá z interface, které definuje metody, jež DAO bude po-

skytovat aplikaci, a dále ze samotné třídy, která daný interface implementuje [4].

Singleton Využití návrhového vzoru singleton spočívá v tom, že se za běh aplikace vytvoří maximálně jedna instance třídy považované za singleton. Taková implementace přináší výhody ve formě ušetřené paměti, speciálně u tříd, jejichž instance je vyžadována často. v balíčku je využita "Lazy initialization", což je způsob inicializace, při které se instance třídy vytvoří až při prvním volání metody pro vytvoření [5].

2.3.2.1 Základ z knihovny

Knihovna nabízí základní implementaci DAO znázorněnou diagramem 2.8, na kterou jsem ve svém balíčku navázal. k dispozici je základní DAO implementující základní interface, dále pak specifický DAO základ k implementaci DAO pro inventář. ve třídách rozšiřujících knihovní DAO je nutné implementovat abstraktní metody get(Cursor) a getValues(T), které transformují objekt na datový záznam a obráceně.

McBaseDAO je základním DAO aplikace MobChar implementující interface McIBaseDAO. Interface poskytuje metody pro přidání, aktualizaci i odebrání objektů, v neposlední řadě pak také metody pro získání objektů z databáze. Parametr navíc musí rozšiřovat knihovní třídu McBaseEntity, což je důvod, proč ne všechny DAO použité v balíčku vycházejí z knihovního DAO.

Třída McInventoryDAO dále obohacuje původní DAO o metody pro získání předmětu/předmětů.

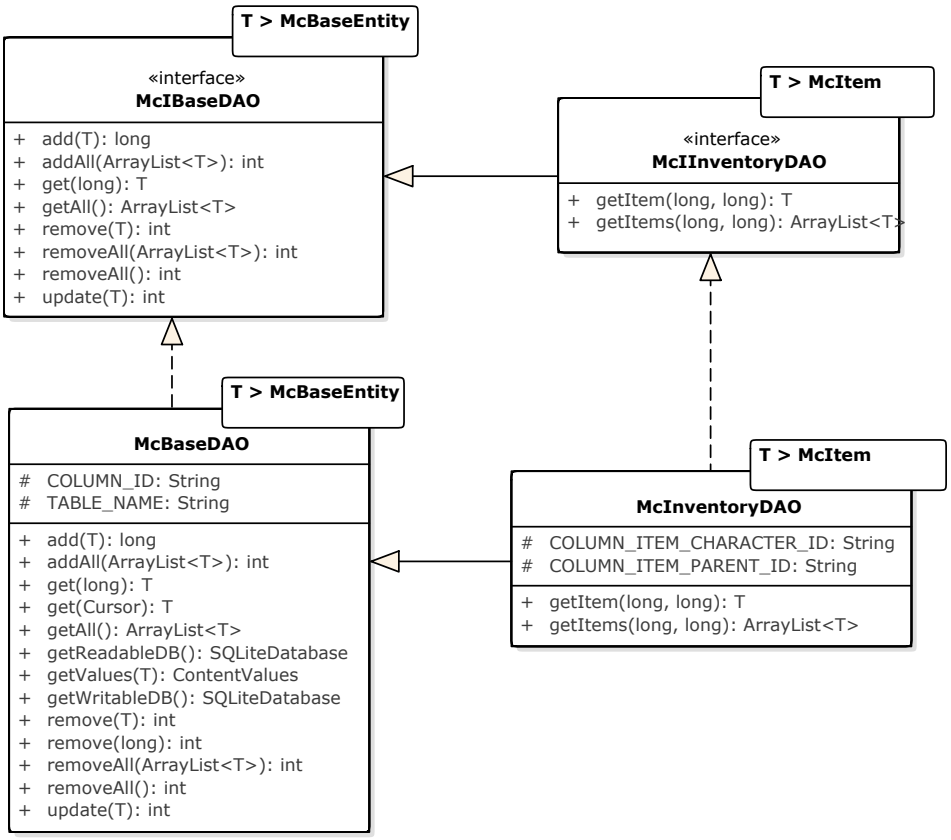
2.3.2.2 Rozšíření knihovní funkcionality

DAO popsané v předchozí podkapitole jsou vhodným výchozím bodem pro implementaci konkrétních DAO použitých v balíčku GURPS. Tento proces popisuje diagram 2.9.

BaseDAO Základním DAO pro práci s daty v balíčku je BaseDAO. BaseDAO přidává možnost získat a smazat všechny záznamy daného typu z jedné postavy. Jedná se o vztah objektů 1:n a takové operace knihovna neumožňuje.

DAO rozšiřující BaseDAO Čtyřmi objekty rozšiřujícími BaseDAO je FeatureDAO, SpellDAO, SkillDAO a ItemSetDAO. Každý z těchto objektů zapouzdřuje korespondující entitu a implementuje operace nad touto entitou.

CharacterDAO a RaceDAO Jak již bylo zmíněno, BaseDAO umožňuje získat všechny záznamy dané entity vztahující se ke konkrétní postavě. Ne



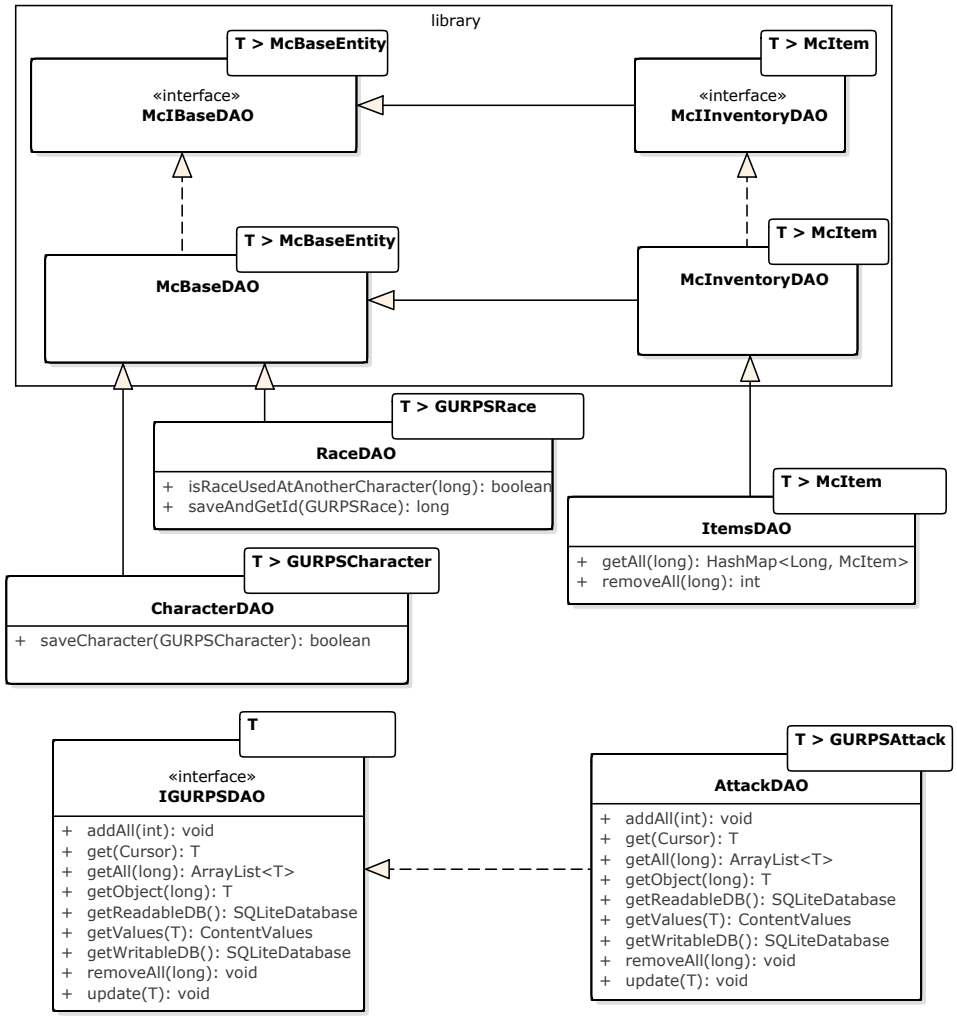
Obrázek 2.8: Přístup k datům - knihovna

pro všechny entity je však tato funkcionalita využitelná. Příkladem je samotná entita GURPSCharacter. Druhou takovou entitou je GURPSRace - každá postava má přesně jednu rasu. DAO pro tyto dvě entity tedy přetěžují přímo knihovní McBaseDAO.

ItemsDAO Posledním DAO, který ještě nebyl zmíněn, je ItemsDAO. Tento rozšiřuje McInventoryDAO dostupný v knihovně o metodu umožňující získat všechny předměty dané postavy.

2.3.2.3 Vlastní DAO

Protože entita GURPSAttack nerozšiřuje knihovní entitu McBaseEntity, nelze pro její ukládání a načítání z databáze využít knihovní DAO základ. AttackDAO tedy není rozšířením McBaseDAO, ale pouze implementuje interface IGURPSDAO. IGURPSDAO poskytuje podobné metody pro práci s objekty



Obrázek 2.9: Přístup k datům - balíček 1. část

a databázovými záznamy jako McBaseDAO, avšak nepožaduje, aby parametr rozšiřoval McBaseEntity. Jednoduchý vztah je znázorněn na diagramu 2.10.

2.3.3 Konstanty

Nezanedbatelným médiem pro uchovávání dat v balíčku GURPS jsou také konstanty. Prvním způsobem použití konstant je uchování konstantních hodnot. pro celý balíček platí stejná pravidla vzhledem k ošetřování vstupů uživatele: Jméno vytvářeného objektu je vyžadováno, ostatní vstupní informace jsou volitelné (pro tento způsob jsem se rozhodl, protože téměř všechny informace jsou v průběhu používání aplikace editovatelné). v souborech konstant se nacházejí výchozí hodnoty pro čísla, která uživatel nevyplní.

Druhý případ, ve kterém jsou využívány soubory s konstantami, jsou výčtové typy. Všechny výčtové typy jsou vytvářeny s identifikátorem id, pomocí kterého je možné převést enum na číslo a zpět. v aplikaci jsou tedy informace jako například druh útoku reprezentovány konkrétní hodnotou enumu a při ukládání do databáze se enum převede na číslo. Tyto enumy jsou deklarovány právě v souborech s konstantami.

CharacterConstants Soubor CharacterConstants sdružuje výchozí hodnoty použité při nevyplnění číselných hodnot při tvorbě postavy.

AttributeConstants V AttributeConstants se nachází funkce, která je používána k vypočítání ceny modifikace hodnoty atributu, a také enum reprezentující typ atributu.

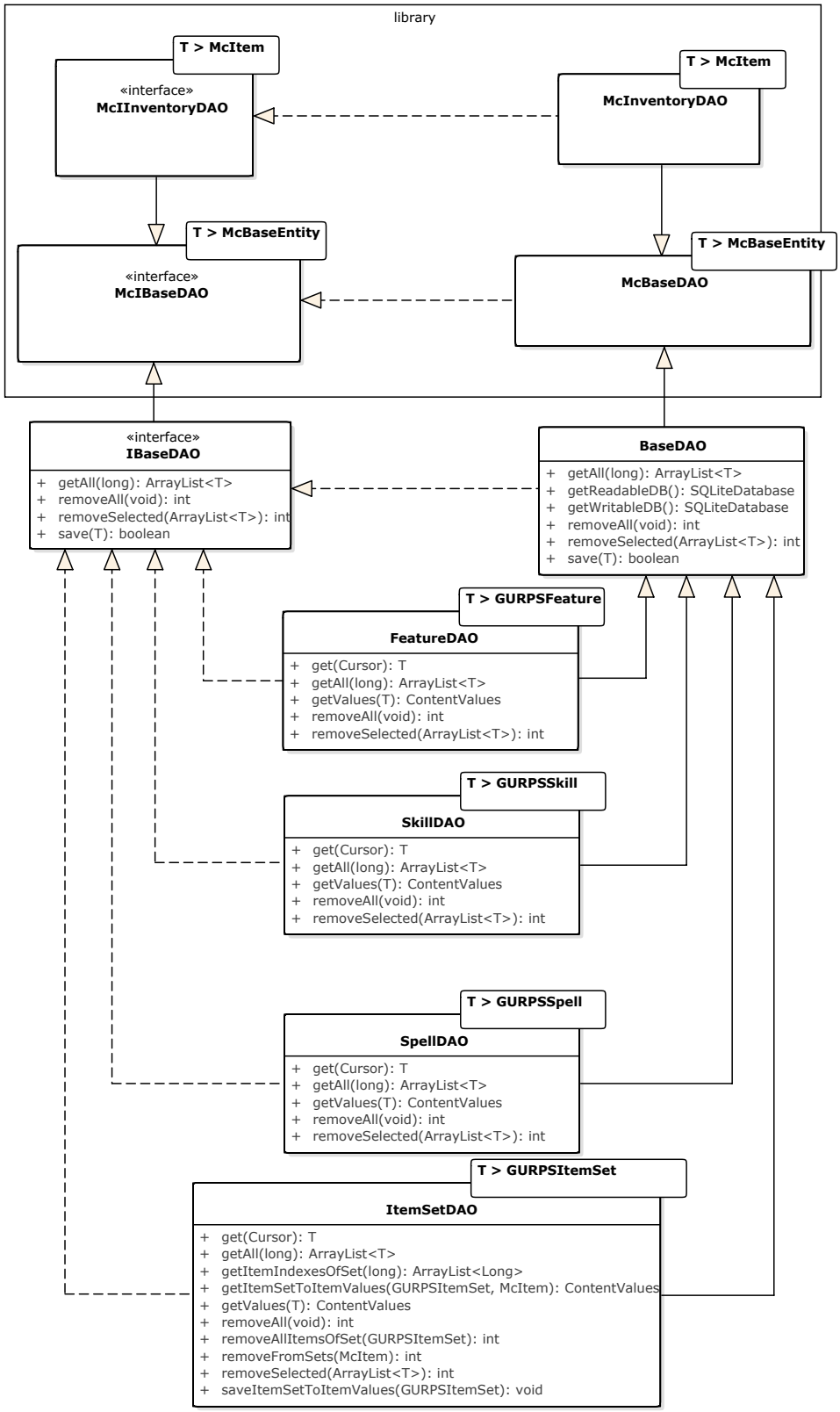
RaceConstants Třída RaceConstants obsahuje enum, který do aplikace poskytuje šablonu rasy Human. Jsou zde také uloženy výchozí hodnoty atributů pro tuto rasu.

FeatureConstants Jednoduchá třída uchovávající výchozí hodnoty pro tvorbu rysu postavy.

SkillConstants V souboru SkillConstants jsou uloženy hodnoty používané k výpočtu ceny vylepšení hodnoty schopnosti na vyšší úroveň, stejně tak jako funkci pro její získání. Dále pak v souboru jsou dva enumy reprezentující druh a obtížnost schopnosti.

SpellConstants Třída obsahující výchozí hodnoty pro vytváření nových kouzel a enum reprezentující typ kouzla.

ArmorConstants Třída obsahující šablony několika přednastavených brnění.



Obrázek 2.10: Přístup k datům - balíček 2. část

ShieldConstants Třída obsahující šablony několika přednastavených štítů.

MeleeWeaponConstants Třída obsahující šablony několika přednastavených kategorií zbraní na blízko a také několik konkrétních zbraní z každé kategorie.

RangedWeaponConstants Třída obsahující šablony několika přednastavených kategorií zbraní na dálku a také několik konkrétních zbraní z každé kategorie.

FirearmsWeaponConstants Třída obsahující šablony několika přednastavených kategorií střelných zbraní a také několik konkrétních zbraní z každé kategorie.

2.4 Logická vrstva

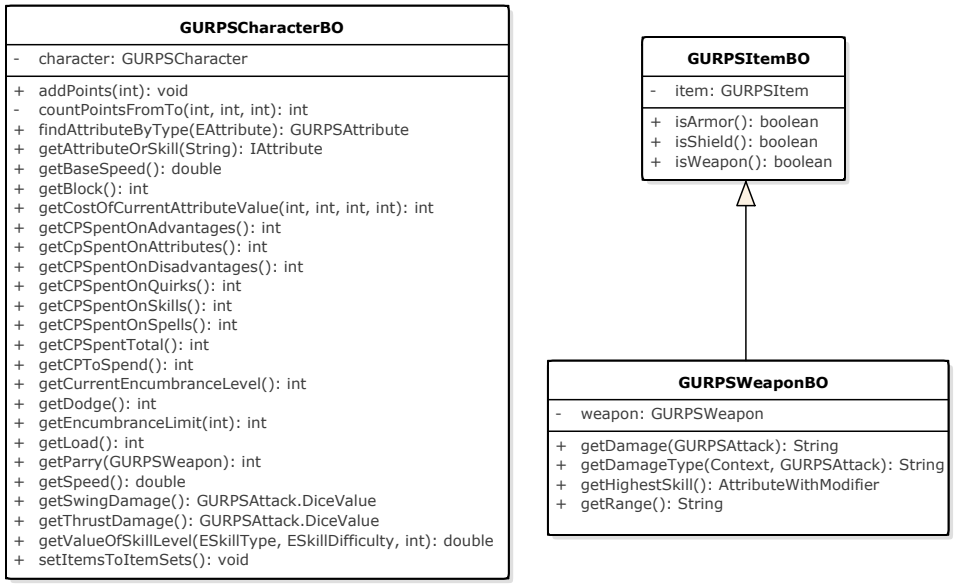
Logická vrstva slouží k transformaci a přístupu k datům, v čemž rozšiřuje datovou vrstvu. v balíčku se nacházejí dva typy objektů logické vrstvy - Business objekty (které transformují datový objekt na business úrovni) a manager třídy, které mají za úkol delegovat požadavky obdržené z prezentační vrstvy na vrstvu datovou.

2.4.1 Business objekty

Business objekty v sobě zapouzdřují objekty datové vrstvy a rozšiřují rozhraní pro práci s nimi o metody logické transformace. Pokud aplikace vyžaduje práci s objektem na datové i logické úrovni, v daném místě pak používá jak business objekt, tak i datový objekt. v případě potřeby pouze jedné vrstvy pak používá odpovídající objekt v dané vrstvě. Business objekty jsou znázorněny na obrázku 2.11.

GURPSCharacterBO Prvním business objektem v balíčku GURPS a zároveň nejrobustnějším je GURPSCharacterBO. Robustnost této třídy vyplývá z faktu, že o postavě evidujeme v aplikaci spoustu informací a zároveň v prezentační vrstvě je zobrazována spousta hodnot, které vznikají právě logickou transformací dat postavy.

Třída obsahuje metody pro počítání ceny zvyšování/snižování hodnoty atributu, metody pro zjišťování stupně zatížení a celkové hmotnosti nesené postavou, získání hodnoty poškození postavy, získání hodnot aktivní obrany, různé metody pro získání hodnoty bodů osobnosti utracených za jednotlivé kategorie, metodu pro vypočítání ceny aktuální hodnoty schopnosti postavy, metody pro získání konkrétního atributu/schopnosti postavy, získání rychlosti a přidání bodů osobnosti.



Obrázek 2.11: Business objekty

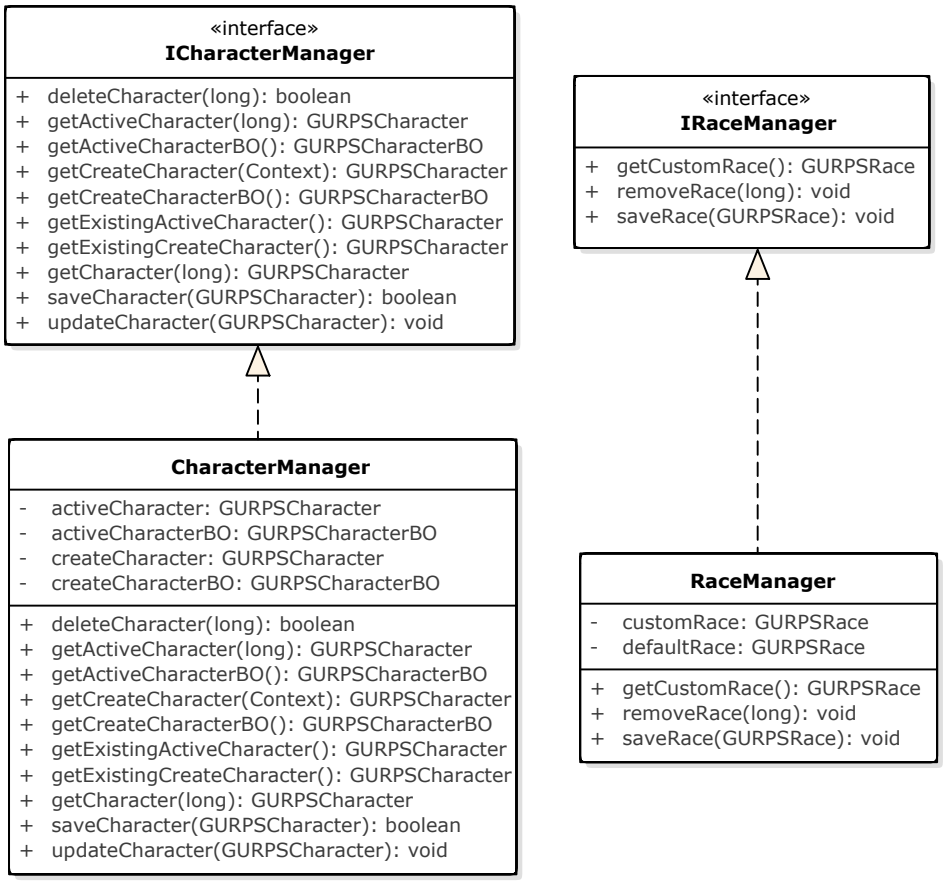
GURPSItemBO Dalším business objektem je GURPSItemBO. Tento objekt slouží k jednoduchému rozlišení typů předmětu - obsahuje metody pro zjištění, zda je objekt typu brnění, typu štít a typu zbraň.

GURPSWeaponBO Posledním business objektem je GURPSWeaponBO. Tato třída obsahuje metody vracející typ a hodnotu poškození zbraně, dosah/dostřel zbraně a metodu, která z postavy, které zbraň patří, vybere nejvyšší atribut/schopnost postavy, a vrátí jeho/její hodnotu.

2.4.2 Managery

Druhou částí business vrstvy balíčku jsou managery. Třídy manager slouží k rozšíření objektů datové vrstvy o logické úkony, například získání pouze předmětů určitého typu. Třídy manager jsou vytvářeny tovární třídou ManagerFactory a stejně jako DAO objekty využívají návrhový vzor Singleton. Obrázky 2.12 a 2.13 zachycují třídy manager vytvořené pro balíček GURPS, obrázek 2.14 znázorňuje rozšíření knihovní manager třídy.

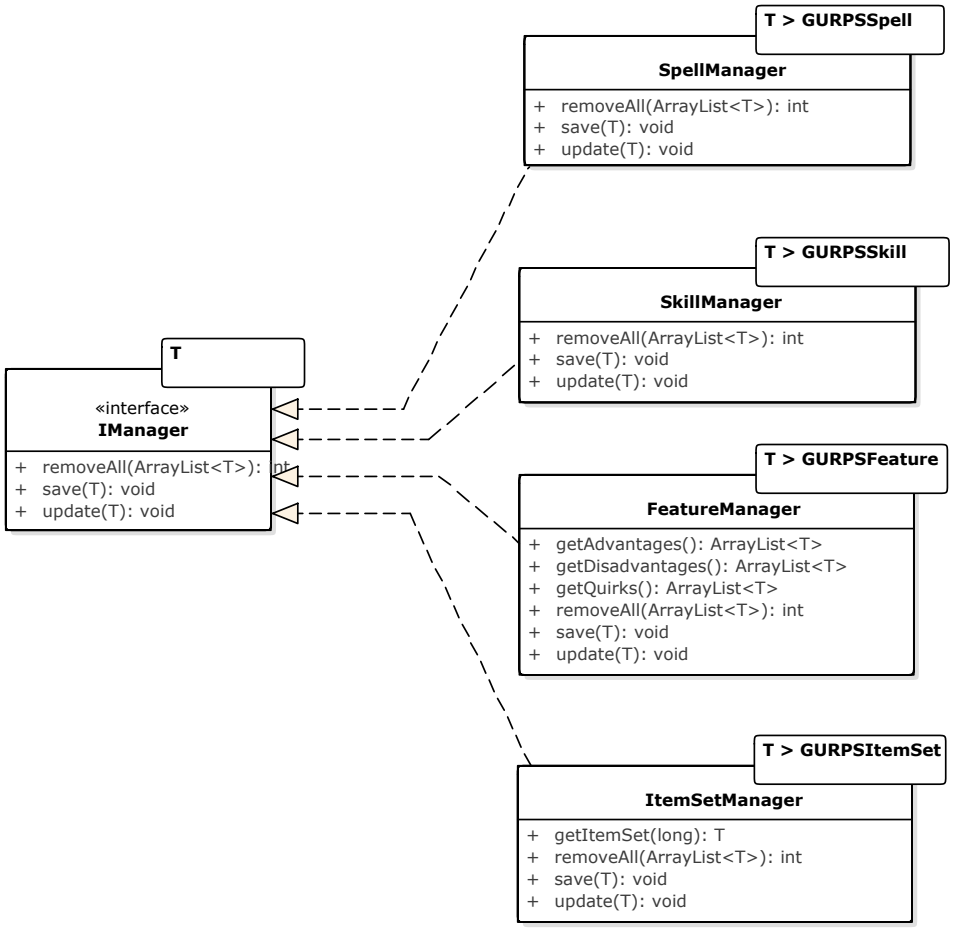
Každá entita balíčku GURPS má svůj manager, který poskytuje základní rozhraní pro komunikaci s datovou vrstvou; jedinou třídou, která manager nemá, je třída GURPSAttack. Tato třída je zpracovávána vždy vzhledem ke třídě GURPSWeapon, komunikace s datovou třídou tedy probíhá přes InventoryManager.



Obrázek 2.12: Třídy manager - 1. část

CharacterManager Třída CharacterManager slouží k úkonům logické vrstvy nad entitou postavy. Obsahuje instanci postavy, která je vytvářena, a instanci postavy, jejíž detail je zobrazován (pokud takové instance existují - při vytváření postavy například není instance postavy pro zobrazení detailu vůbec inicializována). Dále pak v CharacterManager najdeme metody pro aktualizaci existující postavy, uložení vytvářené postavy a smazání postavy.

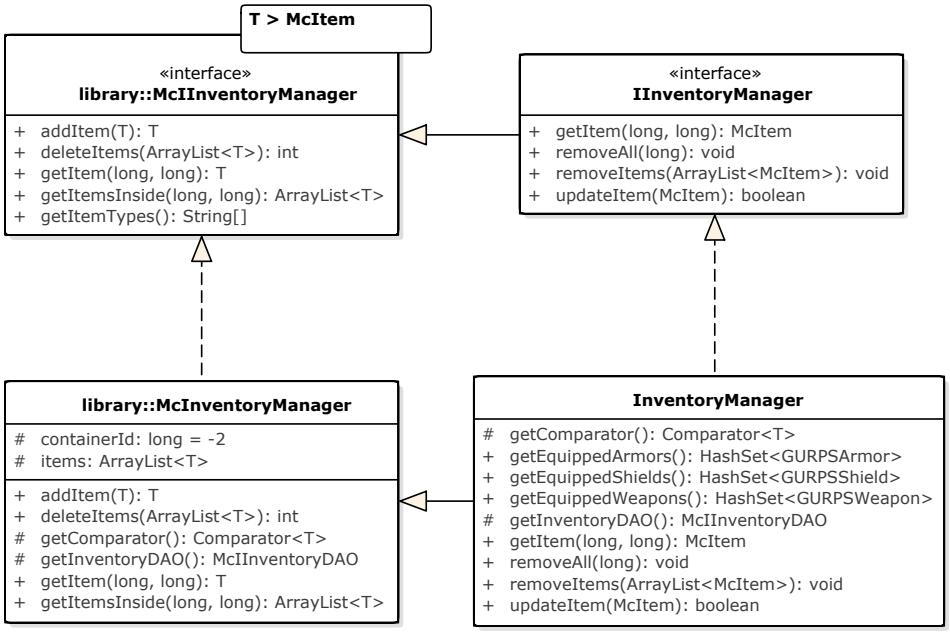
CharacterManager také obsahuje metodu, která všem setům dané postavy vyplní obsahované předměty (tato funkcionalita je opět vyžadována cachováním - není možné vytvořit pro sety nové instance předmětů přímo z DAO, protože by se vytvořené instance neshodovaly s již vytvořenými v seznamu předmětů přímo v postavě. To by vedlo k situaci, kdy by se úpravy předmětů v seznamu v postavě neprojevily v cachované verzi předmětů v setech a opačně).



Obrázek 2.13: Třídy manager - 2. část

RaceManager Poslední třídou typu manager je RaceManager. Tato třída zaobaluje logické metody pro práci s entitou rasy. Obsahuje v sobě dvě instance rasy - defaultRace a customRace. DefaultRace je použita při tvorbě postavy jako výchozí rasa, customRace tvoří základ pro vlastní rasu, kterou si hráč může upravit dle libosti. Třída pak poskytuje metody pro získání/upravení těchto instancí. Navíc třída obsahuje metody pro přidání/upravení/smazání záznamu rasy v databázi.

FeatureManager Třídy FeatureManager poskytuje metody pro úpravu entity GURPSFeature. Jsou zde metody pro uložení, upravení a vymazání rysu postavy. Dále jsou zde také metody pro získání rysů na základě typu.



Obrázek 2.14: Třídy manager - 3. část

SkillManager Třídy SkillManager poskytuje metody pro úpravu entity GURPSSkill. Jsou zde metody pro uložení, upravení a vymazání schopnosti.

SpellManager Třídy SpellManager poskytuje metody pro úpravu entity GURPSSpell. Jsou zde metody pro uložení, upravení a vymazání kouzla.

ItemSetManager ItemSetManager slouží k manipulaci se sety. Poskytuje metody pro přidání setu, aktualizaci a smazání setu.

InventoryManager InventoryManager je rozšířením knihovní předlohy McInventoryManager. Obohacuje tuto třídu o komparátor, pomocí kterého jsou v inventáři řazeny předměty podle jejich typu (a poté abecedně). Dále přetěžuje metodu getItemsInside() tak, že pokaždé obnovuje seznam věcí. v knihovně je totiž metoda naimplementována tak, že k obnovení seznamu věcí dochází pouze po zvolení jiného kontejneru, než který je zobrazován nyní, nebo po přidání/odebrání věci. v balíčku se však může změnit vlastnost předmětu carried, aniž by došlo k obnovení seznamu. Tato vlastnost se navíc projevuje na seznamu předmětů, proto jsem potřeboval danou funkcionalitu mírně upravit.

Dále jsou ve třídě přetíženy metody pro získání/přidání/úpravu/smazání předmětu z důvodu cachování informací v objektech za běhu programu (v kni-

hovní implementaci dochází k úpravám pouze v databázi). Třída pak také nabízí získání seznamu brnění/štítů/zbraní, které jsou aktuálně nošené.

2.4.3 Továrny

Tovární třídy slouží k vytváření instancí objektů podobného typu. Výhodou tohoto návrhového vzoru je fakt, že prostředí zapouzdřuje logiku vytváření objektů a poskytuje pouze rozhraní pro jejich tvorbu [6].

ManagerFactory Tovární třída ManagerFactory slouží k poskytování instancí jednotlivých manager tříd.

DAOFactory Stejně jako ManagerFactory funguje i DAOFactory, avšak poskytuje instance jednotlivých DAO objektů nacházejících se v balíčku.

RaceFactory Třídy RaceFactory slouží k vytváření instancí rasy ze šablony uchovávané v souboru RaceConstants. Továrna obsahuje jedinou metodu, která přijímá id požadované rasy a vrací novou instanci dané rasy, která navíc obsahuje vyplněné hodnoty podle dané šablony.

2.5 Prezentační vrstva

Prezentační vrstva tvoří nejvrchnější vrstvu architektury a slouží k prezentování dat získaných z logické, resp. datové vrstvy. Skládá se ze dvou hlavních jednotek: aktivit a fragmentů.

2.5.1 Aktivity

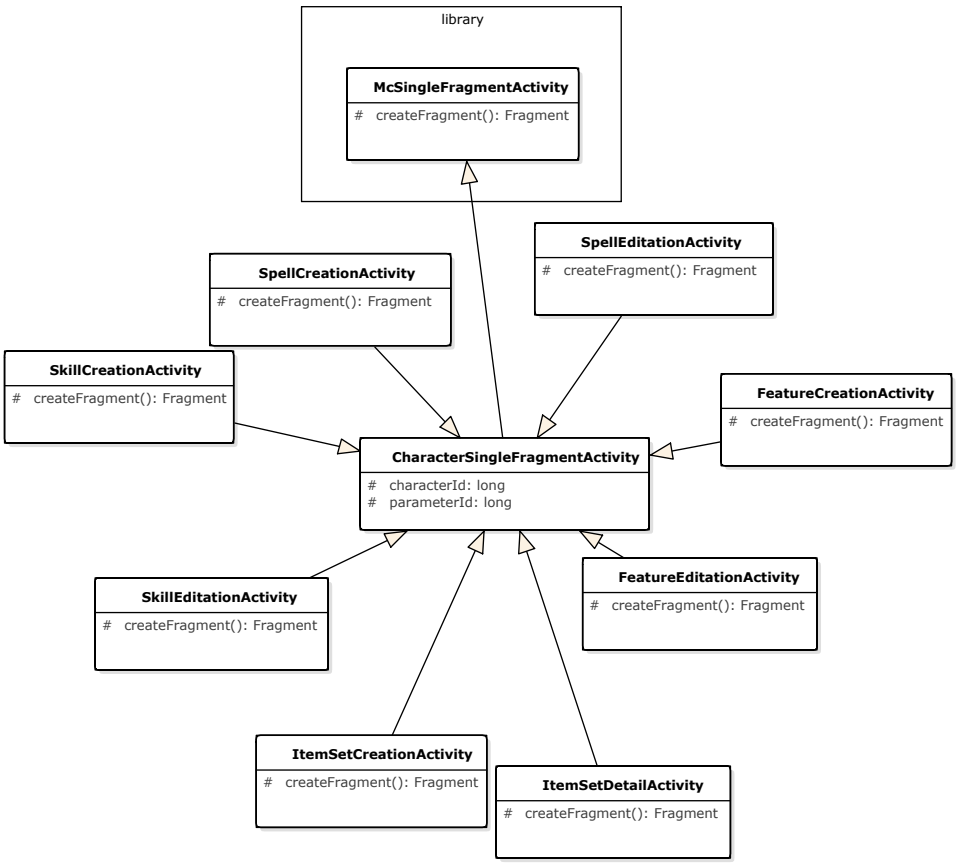
Aktivita je základní komponenta pro zobrazení části aplikace. Takových aktivit má aplikace obvykle více a je možné se mezi nimi přepínat za běhu aplikace (a tím měnit, co aplikace právě zobrazuje) [7].

2.5.1.1 Základ z knihovny

Základní aktivity jsou k dispozici v knihovně MobChar. Tyto aktivity zobrazue diagram 2.18. Aktivity balíčku GURPS jsem pro přehlednost rozdělil do čtyř částí, které postupně popíšu. Všechny aktivity, které jsem použil, vychází z aktivit knihovny. Většina aktivit přetěžuje funkci createFragment(), která vrací objekt typu Fragment, který daná aktivita vytváří a následně obsluhuje.

2.5.1.2 Aktivity detailu postavy

Největší skupinou aktivit vyskytující se v balíčku jsou aktivity, které se vztahují k zobrazování/editaci/vytváření detailu postavy. Všechny tyto aktivity

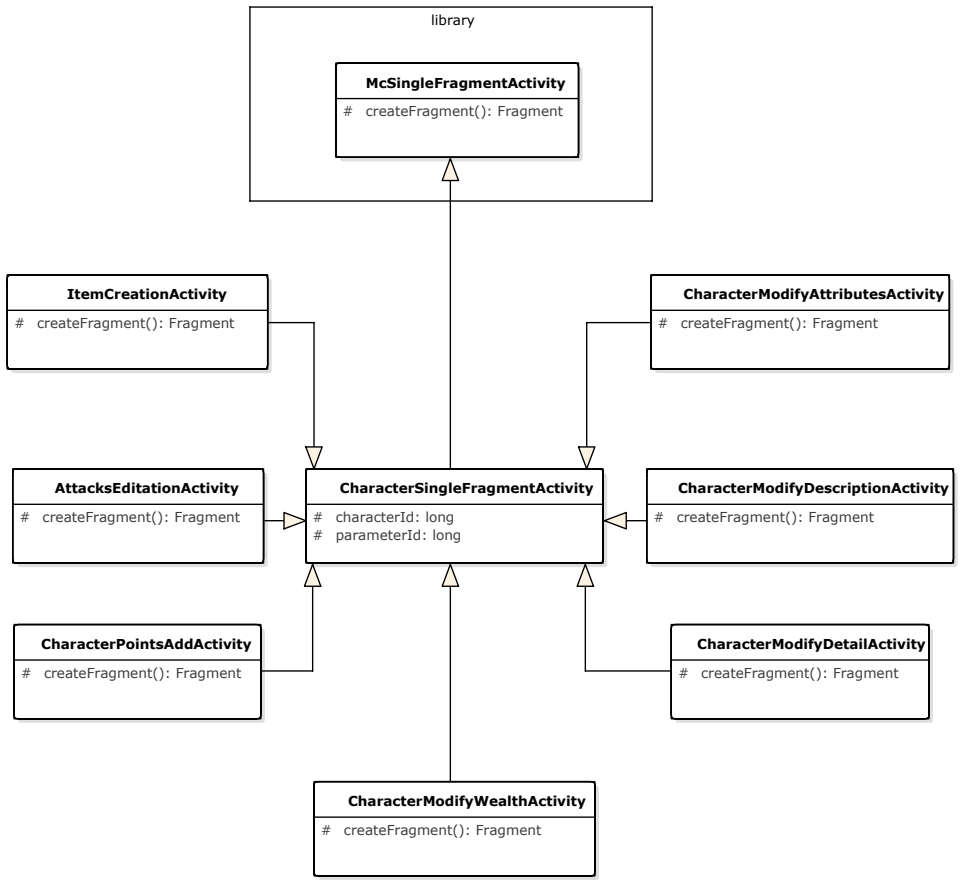


Obrázek 2.15: Aktivity - část 1

vycházejí z knihovní aktivity McSingleFragmentAktivity. Navíc mají téměř identickou strukturu, proto z knihovní aktivity vychází pouze abstraktní aktivita CharacterSingleFragmentActivity. Tuto pak rozšiřují všechny ostatní. Aktivity jsou znázorněny na dvou diagramech 2.15 a 2.16.

CharacterSingleFragmentActivity Abstraktní aktivita, která v sobě uchovává id postavy, jejíž detail zobrazuje.

Aktivity editace základních údajů postavy Skupinou aktivit vycházející z předchozí abstraktní aktivity jsou aktivity, které řídí editaci základních vlastností postavy. Jedná se o aktivity CharacterModifyAttributesActivity, CharacterModifyDescriptionActivity, CharacterModifyDetailActivity a CharacterModifyWealthActivity. Každá z těchto aktivit řídí jeden fragment, pomocí kterého může hráč upravit jednu z částí informací o postavě.

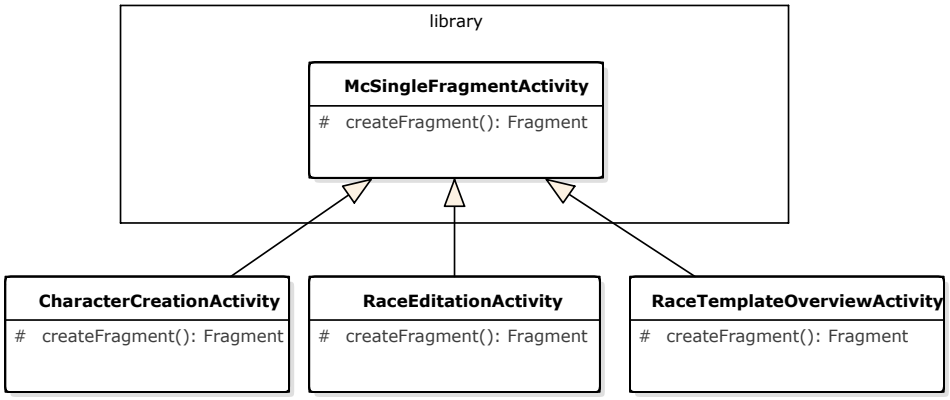


Obrázek 2.16: Aktivity - část 2

FeatureCreationActivity a FeatureEditationActivity Tato dvojice aktivit obsluhuje fragmenty pro vytváření a upravování rysů postavy. Obě aktivity volají stejný fragment FeatureFragment, rozlišení, zda se jedná o editaci či tvorbu, probíhá na základě poskytnutého id rysu. Pokud je id -1, pak se jedná o nově vytvořený rys, v opačném případě je poskytnutá hodnota hodnotou id, pod kterou je rys, který je editován, uložen v databázi.

SkillCreationActivity a SkillEditationActivity Podobná dvojice aktivit obsluhuje tvorbu a editaci schopností postavy.

SpellCreationActivity a SpellEditationActivity Další dvojicí aktivit pro tvorbu a editaci, nyní však kouzel, jsou SpellCreationActivity a SpellEditationActivity.



Obrázek 2.17: Aktivity - část 3

ItemSetCreationActivity a ItemSetDetailActivity Poslední dvojice podobných aktivit. Tyto aktivity slouží k tvorbě a editaci setů předmětů.

CharacterPointsAddActivity Aktivita obsluhující fragment, prostřednictvím kterého může hráč editovat počet bodů osobnosti.

AttacksEditationActivity Aktivita, která obsluhuje fragment upravující útoky zbraně.

ItemCreationActivity Poslední aktivitou detailu postavy je aktivita řídící vytvoření nového předmětu.

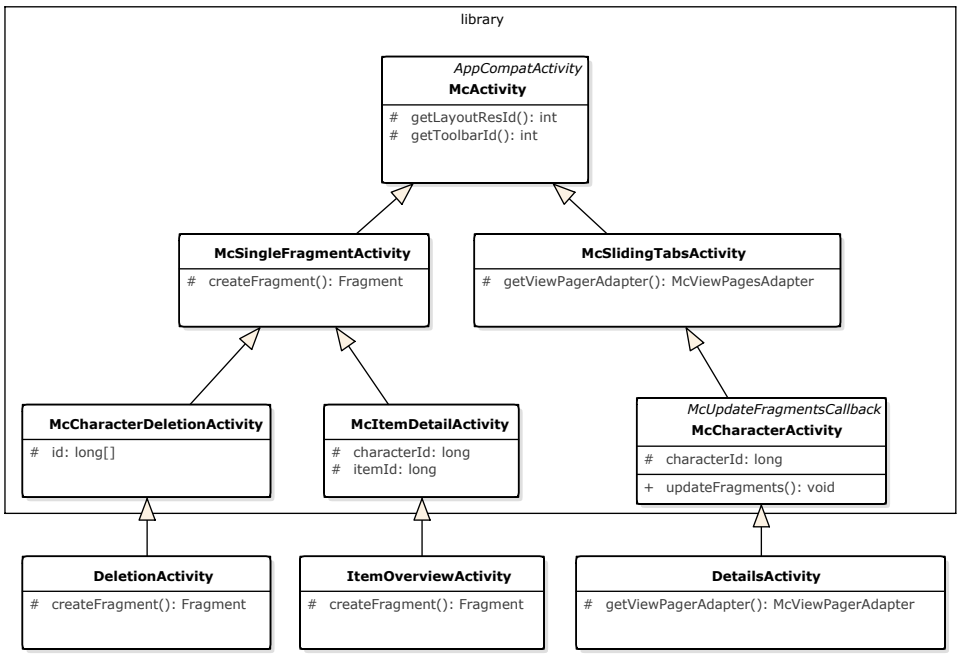
2.5.1.3 Ostatní aktivity

Další aktivity balíčku GURPS již nevychází z abstraktní aktivity CharacterSingleFragmentActivity, avšak rozšiřují přímo některou z aktivit v knihovně.

CharacterCreationActivity Aktivita sloužící k obsluhování fragmentu, který zprostředkovává tvorbu postavy.

RaceTemplateOverviewActivity a RaceEditationActivity Dvojice aktivit obsluhující vybírání, popř. vytvoření rasy. Zatímco RaceTemplateOverviewActivity slouží k zobrazování rasy uložené v šabloně, RaceEditationActivity umožňuje editovat vlastní rasu, kterou je poté možno uložit.

ItemOverviewActivity Tato aktivita vychází z knihovní aktivity McItemDetailActivity a slouží k obsluhování fragmentu zobrazujícího detail předmětu.



Obrázek 2.18: Aktivity - část 4

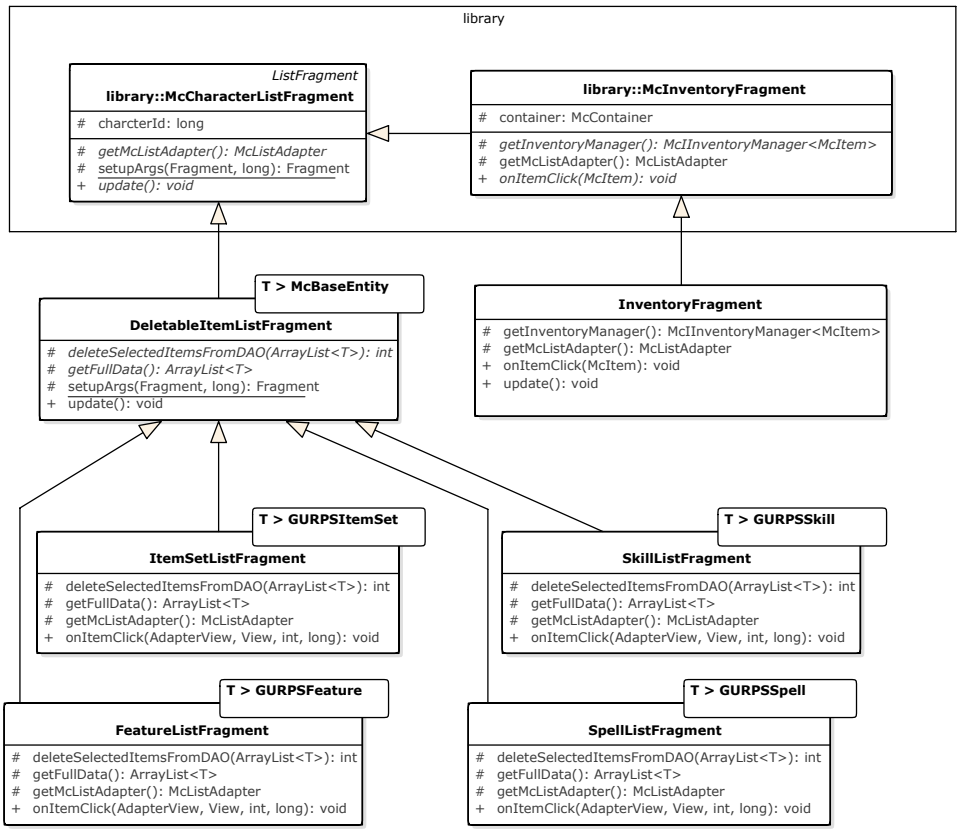
DetailsActivity Jediná aktivita v balíčku, která rozšiřuje knihovní aktivitu McCharacterActivity. Tato knihovní aktivita poskytuje obsluhu více různých fragmentů pomocí záložek na vrchní straně obrazovky (kde vybrání záložky v menu znamená přepnutí na fragment odpovídající vybrané záložce). Aktivita DetailsActivity za tímto účelem implementuje metodu getViewPagerAdapter(), ve které je deklarováno pořadí fragmentů v aktivitě.

DeletionActivity Poslední aktivitou v balíčku je aktivita obsluhující smazání postavy. Tato aktivita rozšiřuje knihovní McCharacterDeletionActivity.

2.5.2 Fragmenty

Fragment reprezentuje ucelenou část aplikace, neboli jednu grafickou obrazovku aplikace. Aktivita může obsahovat více fragmentů, ale zároveň může být ten samý fragment použit ve více aktivitách - představuje jakýsi modul aplikace. Fragment je vždy zaštiťován nějakou aktivitou a jeho životní cyklus s ní přímo koresponduje [8].

V balíčku je použito několik základních skupin fragmentů. První skupinou jsou fragmenty, které umožňují výpis seznamu prvků. Všechny fragmenty této skupiny rozšiřují ListFragment a jsou vyobrazeny na diagramu 2.19.



Obrázek 2.19: Fragmenty - část 1

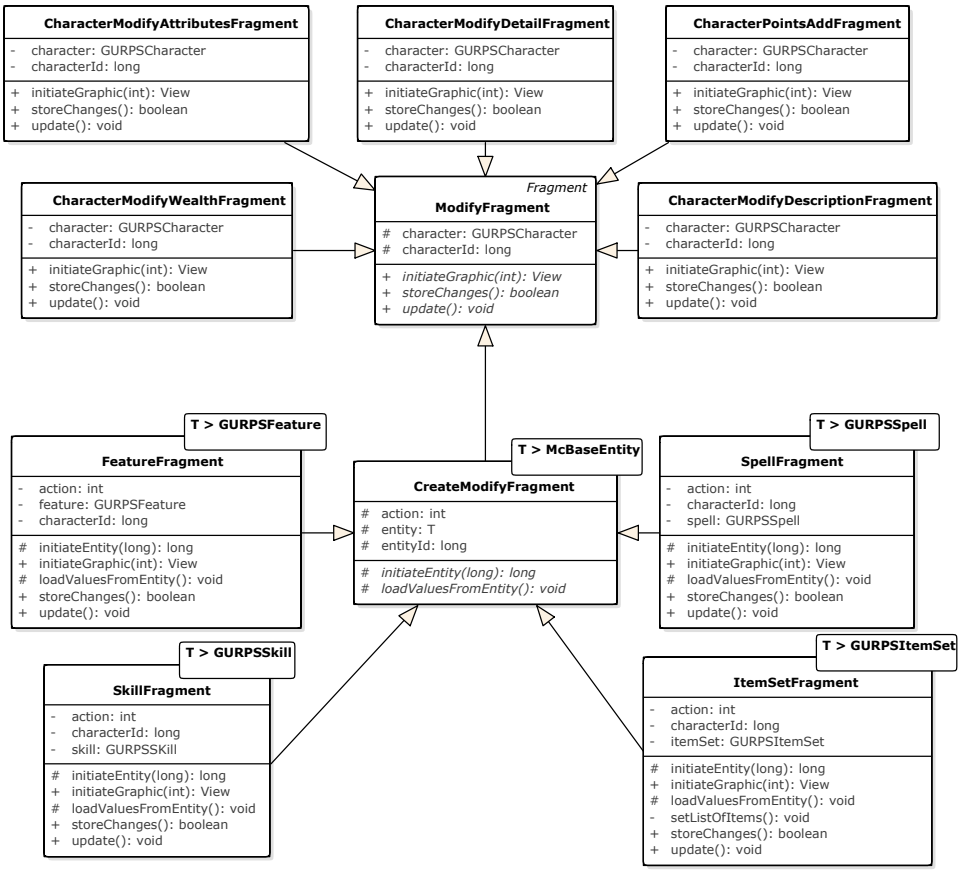
Druhou kategorií jsou fragmenty editace postavy. Tato skupina má ještě dvě podkategorie - fragmenty, které slouží pouze k editaci informací o postavě, a fragmenty, které slouží jak k editaci, tak k tvorbě entity patřící postavě. Všechny tyto fragmenty jsou vyobrazeny na diagramu 2.20.

Ostatní fragmenty jsou vyobrazeny na diagramu 2.21.

2.5.2.1 Fragmenty poskytující seznam

První zmiňovanou skupinou jsou fragmenty, které zobrazují seznam entit. Tyto fragmenty také prostřednictvím kontextového menu umožňují ze seznamu položky mazat. Kontextové menu se vyvolá dlouhým stisknutím položky v seznamu. InventoryFragment vychází z knihovního fragmentu McInventoryFragment a poskytuje seznam předmětů uvnitř daného kontejneru.

Zbylé fragmenty této kategorie jsou fragmenty rozšiřující pouze knihovní McCharacterListFragment. pro všechny čtyři fragmenty balíčku jsem vytvořil



Obrázek 2.20: Fragmenty - část 2

abstraktní fragment DeletableItemListFragment, který obsahuje parametr rozšiřující knihovní McBaseEntity. Tento fragment obsahuje funkcionalitu společnou pro všechny jeho potomky (například inicializace id postavy a inicializace postavy jako takové, deklarace menu v horní liště atd.).

Samotné fragmenty pak jsou ItemSetListFragment, FeatureListFragment, SkillListFragment a SpellListFragment. Jak název napovídá, jedná se o fragmenty zobrazující seznam setů, seznam rysů postavy, seznam schopností a kouzel postavy.

2.5.2.2 Fragmenty upravující část postavy

Druhou kategorií fragmentů použitých v balíčku GURPS jsou fragmenty upravující část postavy a fragmenty sloužící k editaci/tvorbě entit patřících dané postavě.

Pro všechny fragmenty spadající do této kategorie jsem vytvořil abstraktní trídu ModifyFragment rozšiřující Fragment. Tato třída uchovává informace o id postavy a instanci dané postavy. Dále deklaruje abstraktní metody pro vytvoření grafického layoutu fragmentu a pro uložení změn provedených uživatelem nad danými informacemi.

Pro druhou skupinu jsem navíc rozšířil ModifyFragment a výslednou abstraktní třídu pojmenoval navíc ještě abstraktní třídu CreateModifyFragment rozšiřující ModifyFragment. Tato třída pro potřeby evidování konkrétní entity uchovává instanci entity a její id. Navíc ještě uchovává hodnotu, zda je entita vytvářena nebo upravována. Poskytuje pak abstraktní metody initiateEntity(id), která na základě id vytvoří instanci (pokud je id -1, vytvoří se nová instance. Pokud je jiné, instance je vytvořena z databáze, popřípadě vyzvednuta z existující instance postavy).

Z diagramů je patrné, že fragmenty poskytující seznam entit a fragmenty upravující konkrétní entitu spolu úzce souvisí a pracují nad stejným seznamem entit. Jedná se konkrétně o entity rysu postavy, dále pak setu, schopnosti a nakonec kouzla.

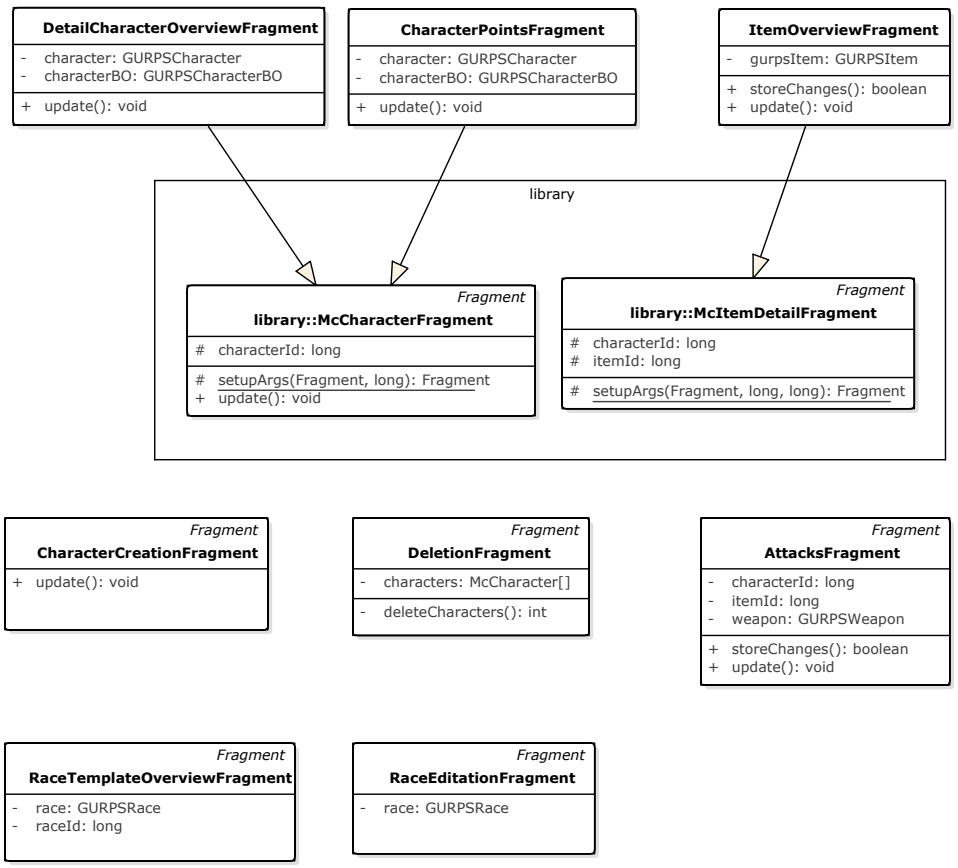
2.5.2.3 Ostatní fragmenty

Všechny fragmenty poskytující seznam entit spadají pod stěžejní aktivitu DetailsActivity, která operuje nad jednotlivými záložkami detailu postavy. Ne všechny fragmenty této aktivity však poskytují seznam, některé poskytují pouze náhled na informace o postavě.

Jedná se o fragmenty DetailCharacterOverviewFragment, který zobrazuje základní informace o postavě a její atributy, a CharacterPointsFragment, který zobrazuje statistiku použitých a dostupných bodů osobnosti. Tyto dva fragmenty rozšiřují McCharacterFragment z knihovny.

Posledním fragmentem vycházejícím z knihovny je ItemOverviewFragment zobrazující detail o předmětu. v knihovně definovaný McItemDetailFragment pouze pasivně zobrazuje informace, avšak v balíčku GURPS tento fragment slouží i k nastavování, jestli daný předmět postava nese či nikoli. Proto je knihovní fragment rozšířen o metody storeChanges() a update(), které se starají o zaznamenání této hodnoty.

Závěrem zbývá několik fragmentů, které ani nerozšiřují knihovnu, ani nemají společnou abstraktní třídu. do této kategorie se řadí fragmenty CharacterCreationFragment, RaceEditationFragment a RaceTemplateOverviewFragment. Tyto tři fragmenty slouží k zobrazování grafických elementů při tvorbě postavy. Dále se zde nachází DeletionFragment a AttacksFragment. AttacksFragment pracuje s entitou GURPSAttack, která nerozšiřuje knihovní třídu McBase-



Obrázek 2.21: Fragmenty - část 3

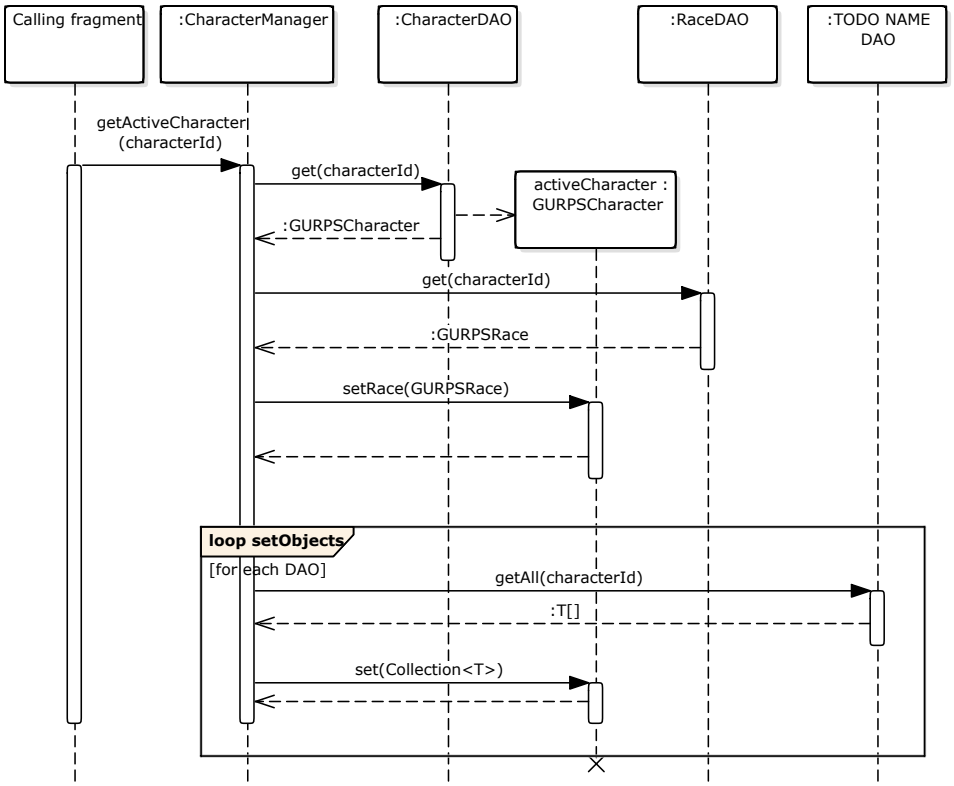
Entity, proto je tento fragment zařazen stranou.

2.5.3 Komunikace jednotlivých vrstev

Na následujících stránkách bych rád popsal, jako spolu jednotlivé vrstvy třívrstvé architektury komunikují. Princip, kterého jsem se snažil držet po celou dobu tvorby aplikace, byl pokud možno co nejméně využívat komunikaci s databází. Proto si po zvolení konkrétní postavy, jejíž detail je zobrazován, v aplikaci uchovávám instanci dané postavy spolu s instancemi všech dalších entit, které spadají pod danou postavu.

2.5.3.1 Načtení postavy

Jak bylo zmíněno výše, aplikace si drží aktuální stav postavy, jejíž detail je zobrazován, prostřednictvím entity GURPSCharacter. Tato entita v sobě obsahuje i všechny ostatní entity a seznamy entit, které spadají pod tuto postavu.

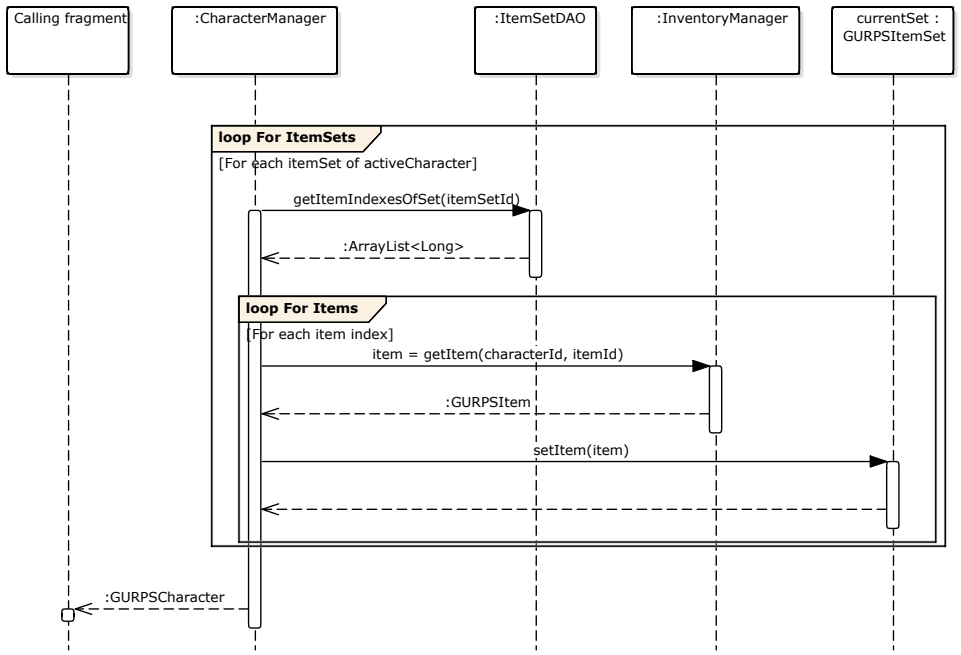


Obrázek 2.22: Komunikace mezi vrstvami - část 1

Instance této postavy je udržována ve třídě CharacterManager. Poprvé je inicializována v momentě, kdy o ni zažádá první fragment. po dobu zobrazování detailu stejné postavy je tato instance pouze upravována, ale není znovu inicializována.

Inicializace probíhá následujícím způsobem: fragment požádá ManagerFactory o instanci třídy CharacterManager prostřednictvím funkce getCharacterManager() třídy ManagerFactory. Vzhledem k tomu, že CharacterManager sleduje návrhový vzor Singleton, je jeho instance vytvořena (resp. vyzvednuta), pokud v minulosti nebyla (resp. byla) inicializována. po obdržení instance CharacterManager fragment v této třídě volá metodu getActiveCharacter(characterId) a jako parametr characterId dosazuje hodnotu obdrženou z aktivity.

Pokud již postava byla inicializována, pouze vrátí její instanci. Pokud však inicializována nebyla, CharacterManager za tímto účelem zavolá svoji metodu getCharacter(characterId), do které předá obdržený parametr. Tato metoda má za úkol kontaktovat datovou vrstvu a získat všechny části patřící postavě,



Obrázek 2.23: Komunikace mezi vrstvami - část 2

tyto části přiřadit do entity GURPSCharacter a výslednou instanci vrátit.

Metoda getCharacter slouží ke komunikaci s datovou vrstvou. Komunikace probíhá tak, že si z této metody aplikace vyžádá od třídy DAOFactory instanci CharacterDAO, ze kterého potřebuje obdržet konkrétní data, pomocí metody getCharacterDAO(). CharacterDAO opět sleduje návrhový vzor Singleton. po obdržení instance CharacterDAO je zavolána metoda get(characterId), jejíž návratovou hodnotou je instance GURPSCharacter vzniklá načtením hodnot v databázi.

CharacterManager poté tuto komunikaci provede i s ostatními DAO. Pokud jsou entity získané z DAO s postavou ve vztahu 1:1, probíhá komunikace stejně jako s CharacterDAO. Pokud je vztah 1:N, v DAO se volá metoda getAll(characterId).

ItemSetDAO vrací pouze prázdné sety bez předmětů. Důvodem je právě držení aktuální instance za běhu programu. Pokud by se vytvořily předměty v DAO, existovalo by v daném čase více instancí jednoho předmětu. Změny na daném předmětu by se tedy nepromítly do ostatních instancí uchovávaných v aplikaci (i když do databáze by se zapsaly).

Řešením je tedy dodatečné vyplnění předmětů do setů. CharacterManager

zavolá svoji metodu setItemsToItemSets(character). Tato metoda si pro každý set nalezený v instanci character vyžádá od ItemSetDAO seznam id všech předmětů obsaženém v setu pomocí funkce getItemIndexesOfSet(itemSetId). Poté si zažádá ManagerFactory o instanci třídy InventoryManager. nad touto instancí poté zavolá metodu getItem(characterId, itemId), která obdrží z aktivní postavy předmět s daným id, a instanci tohoto předmětu dosadí do setu.

Po dokončení všech výše vypsaných operací CharacterManager obsahuje již plně inicializovanou instanci postavy, kterou vrátí z metody getCharacter. Tuto instanci si nejprve uloží jako activeCharacter pro pozdější opětovné použití a poté ji také vrátí jako výsledek metody getActiveCharacter.

2.5.3.2 Získání části postavy

Druhým případem komunikace mezi různými vrstvami programu je situace, kdy fragment potřebuje zobrazit některé informace postavy. Díky natáhnutí celé postavy do paměti programu na začátku předpokládáme, že v tuto chvíli je postava inicializovaná. Pokud ne, proběhne inicializace podle sekce 2.5.3.1.

Jako příklad pro demonstraci této situace jsem vybral fragment SkillFragment, který obdržel od aktivity id postavy a id schopnosti, kterou má zobrazit. z fragmentu je zavolána metoda getSkillManager() nad třídou ManagerFactory, která vrátí instanci třídy SkillManager. Obdržení instance probíhá totožně jako obdržení instance ostatních manager tříd. nad instancí SkillManager je zavolána metoda get(skillId), která má za úkol prohledat seznam kouzel v aktuální postavě, najít kouzlo, jehož id odpovídá hodnotě proměnné skillId, a toto kouzlo vrátit. Vrácená instance je pak uložena do fragmentu SkillFragment a fragment pak potřebné informace extrahuje již z uložené entity. v celé komunikaci tedy díky udržování instance postavy za běhu aplikace nedojde ke komunikaci s datovou vrstvou.

2.5.3.3 Uložení změn ve fragmentu

Posledním významným případem komunikace mezi vrstvami je situace, kdy fragment (např. SkillFragment) zobrazil informace o schopnosti a uživatel některé z těchto informací změnil a přeje si změny uložit.

Aby vše fungovalo správně, je potřeba změny uložit jak do databáze (pro pozdější načtení), tak do aktuální instance postavy (pro korektní zobrazení za současného běhu aplikace). Tyto dvě funkcionality zprostředkovávají ve fragmentech, které tuto funkcionalitu poskytují, funkce storeChanges(), resp. update().

Metoda storeChanges() přečte postupně všechny informace vložené uživatelem a uloží je do instance entity, kterou fragment uchovává, v tomto případě SkillFragment uchovává instanci GURPSSkill. Protože byla instance obdržená

přímo z existující postavy, změny se automaticky projeví v seznamu schopností dané postavy, jelikož jde o jednu a tu samou instanci schopnosti.

Druhý případ implementovaný metodou update() deleguje požadavek uložení objektu do databáze na odpovídající manager třídu, v tomto případě SkillManager. Tuto instanci získá od třídy ManagerFactory (bylo již popsáno). v instanci třídy SkillManager je zavolána metoda update. v této metodě je od DAOFactory vyžádána instance odpovídajícího DAO (v tomto případě SkillDAO) a konečně nad touto instancí zavolána metoda update, která propíše změny do databáze.

Kapitola 3

Realizace

V následující kapitole práce budu popisovat detaily spojené s implementací balíčku GURPS. Popíšu nástroje, které jsem během implementace použil. Dále se budu věnovat zapojení balíčku do jádra aplikace MobChar - co vše je potřebné v balíčku deklarovat, aby mohl být úspěšně integrován. Nakonec se zmíním o testování - automatizovaném, ale i uživatelském.

3.1 Použité nástroje

V následujících odstavcích popíšu, jaké programy a jiné nástroje jsem pro tvorbu práce využil.

3.1.1 Vývojové prostředí

Android Studio je oficiálním IDE pro vývoj na platformě android. Tento program je zcela zdarma a je postaven nad Community verzí prostředí IntelliJ IDEA. To mu umožňuje převzít všechny výhody v práci s kódem (navigace, našeptávání, refaktoring, zvýrazňování syntaxe), ve kterých IDA vyniká [9].

K testování aplikace jsem používal svůj osobní mobilní telefon, na kterém jsem zprovoznil ladění pomocí USB. Tato funkcionalita mi velice usnadňovala práci v průběhu celého programování, jelikož mi umožňovala sledovat chybové výpisy z mobilu přímo v Android Studiu, navíc umožňovala i aplikaci na mobilu debuggovat přes samotné Android Studio.

3.1.2 Git

Pro správu kódu jsem použil nástroj Git. Kód balíčku jsem průběžně ukládal do lokálního úložiště a poté nahrál na společný repozitář aplikace MobChar. Zdrojové kódy balíčku GURPS jsou k dispozici na adrese `https://gitlab.fit.cvut.cz/rybolzde/MobChar`.

3.1.3 Obrázky

Ikony pro aplikace jsem převzal ze stránky `http://game-icons.net/`.

3.2 Začlenění balíčku do jádra

Balíček GURPS je modulem aplikace MobChar. Všechny balíčky jsou ale vyvíjeny jako samostatná aplikace. Ta sice nejde spustit, ale je spuštěna přímo z jádra MobCharu. k tomu, aby takové propojení fungovalo, vyžaduje jádro, aby balíček implementoval některé funkcionality, pomocí kterých může jádro správně vykonávat úkony, než je k obsluze postav zavolán balíček samotný.

Mezi požadované funkcionality patří deklarace základních aktivit balíčku, které jsou spouštěny jádrem, a poskytnutí seznamu postav, které jsou v balíčku k dispozici, nebo-li poskytnutí cesty, na které najde jádro aplikace onen seznam.

3.2.1 Vyžadované aktivity

Jádro aplikace potřebuje ke správné funkci deklarované aktivity pro zobrazení detailu postavy, pro vytvoření nové postavy a pro smazání postavy. Následující úryvek kódu ze souboru AndroidManifest.xml, který dané aktivity deklaruje. v tomto souboru jsou deklarovány i všechny ostatní aktivity, ale z úryvku zde jsem je vypustil.

```
1  <application android:allowBackup="true" android:icon="@mipmap/
       ic_launcher"
2          android:label="@string/app_name" android:supportsRtl="true"
               android:theme="@style/AppTheme">
3          <meta-data android:name="gameName" android:value="GURPS"
               />
4          <meta-data android:name="gameAbbreviation" android:value="
               GURPS" />
5          <meta-data android:name="characterProviderURI"
6              android:value="content://cz.cvut.mobchar.gurps.database.
                   provider.ContentProvider/characters" />
7
8          <activity
9              android:name=".presentation.detail.DetailsActivity"
10             android:label="@string/app_name" >
11             <intent-filter>
12                 <action android:name="cz.cvut.fit.mobchar.showcharacter"
                       />
13             </intent-filter>
14         </activity>
```

```
15
16          <activity
17              android:name=".presentation.creation.
                    CharacterCreationActivity"
18              android:label="@string/title_character_creating" >
19              <intent-filter>
20                  <action android:name="cz.cvut.fit.mobchar.createcharacter
                        " />
21              </intent-filter>
22          </activity>
23
24          <activity
25              android:name=".presentation.deletion.DeletionActivity"
26              android:label="@string/title_deletion" >
27              <intent-filter>
28                  <action android:name="cz.cvut.fit.mobchar.deletecharacter"
                        />
29              </intent-filter>
30          </activity>
31
32          <provider
33              android:name=".ContentProvider"
34              android:authorities="cz.cvut.mobchar.gurps.database.provider.
                    ContentProvider"
35              android:exported="true" />
36          <uses-library android:name="android.test.runner" />
37      </application>
```

3.2.2 Content provider

Druhou funkcionalitou, kterou jádro vyžaduje, je seznam postav, které jsou v balíčku vytvořeny. Tuto funkcionalitu poskytuje soubor s názvem ContentProvider. v tomto souboru jádro aplikace najde databázi, ve které se nachází seznam postav. v následujícím úryvku kódu se nachází ukázka souboru ContentProvider.

```
1   public class ContentProvider extends McCharacterContentProvider {
2
3       protected static final String AUTHORITY = "cz.cvut.mobchar.gurps.
            database.provider.ContentProvider";
4       protected static final String CHARACTERS_TABLE =
            CharacterTable.TABLE_CHARACTERS;
5
6       private ContentResolver cr;
```

```
 7
 8      public static final int CHARACTERS = 1;
 9      public static final int CHARACTERS_ID = 2;
10
11      protected static final UriMatcher URIMatcher = new UriMatcher(
            UriMatcher.NO_MATCH);
12
13      static {
14          URIMatcher.addURI(AUTHORITY, CHARACTERS_TABLE,
                CHARACTERS);
15          URIMatcher.addURI(AUTHORITY, CHARACTERS_TABLE + "
                /#", CHARACTERS_ID);
16      }
17
18      private DatabaseHandler db;
19
20      @Override
21      public boolean onCreate() {
22          db = DatabaseHandler.getInstance(getContext());
23          return false;
24      }
25
26      @Override
27      public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {
28          SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
29          queryBuilder.setTables(CharacterTable.TABLE_CHARACTERS);
30
31          int uriType = URIMatcher.match(uri);
32
33          switch (uriType) {
34              case CHARACTERS_ID:
35                  queryBuilder.appendWhere(CharacterTable.COLUMN_ID
                        + "="
36                              + uri.getLastPathSegment());
37                  break;
38              case CHARACTERS:
39                  break;
40              default:
41                  throw new IllegalArgumentException("Unknown URI");
42          }
43
44          Cursor cursor = queryBuilder.query(db.getReadableDatabase(),
45                  projection, selection, selectionArgs, null, null,
```

```
                sortOrder);
46          cursor.setNotificationUri(getContext().getContentResolver(), uri);
47          return cursor;
48      }
49  }
```

3.3 Testování

Důležitou součástí vývoje aplikace je také testování aplikace. Protože v balíčku je použita třívrstvá architektura, je potřeba otestovat každou z vrstev. Testování datové třídy a logické třídy je implementováno automatizovanými jednotkovými testy, zatímco testování prezentační vrstvy je realizováno pomocí uživatelských testů. Uživatelské testování probíhá pomocí manuálního testování aplikace uživatelem, který má k dispozici testovací scénář. Tento scénář obsahuje instrukce, které uživatel postupně plní a ověřuje, jestli aplikace pracuje správně a jestli zobrazí očekávaný výsledek či nikoli.

3.3.1 Testy DAO

První sekcí testů jsou testy datové třídy. v aplikaci jsem implementoval testy pro dva vybrané DAO - FeatureDAO a SkillDAO. Tyto soubory zprostředkovávají komunikaci s databází pro entity, které jsou s postavou ve vztahu 1:N. v testovacích metodách pro tyto objekty je testováno, zda jsou informace o entitách GURPSFeature (resp. GURPSSkill) správně ukládány do databáze, správně v databázi aktualizovány a také mazány.

```
1   public class SkillDAOTest extends AndroidTestCase{
2       private void setTestData();
3
4       Comparator<GURPSSkill> comparator = new Comparator<
            GURPSSkill>() {
5           @Override
6           public int compare(GURPSSkill lhs, GURPSSkill rhs) {
7
8               ...
9           }
10      };
11
12      public boolean isEquals(ArrayList<StringWithModifier> lhs,
            ArrayList<StringWithModifier> rhs);
13
14      public void setUp() throws Exception;
15
```

```
16      public void tearDown() throws Exception;
17
18      public void testGetSkills() throws Exception;
19
20      public void testDeleteSkills() throws Exception;
21
22      public void testUpdateSkills() throws Exception;
23  }
```

Nejprve je zavolána funkce setTestData(), která zajistí inicializaci objektů třídy GURPSCharacter a GURPSCharacterBO. Poté tato funkce vytvoří instance konkrétních objektů třídy GURPSSkill a těmto instancím jsou nastaveny počáteční údaje. Dále se tato funkce postará o nastavení vytvořených instancí GURPSSkill do dříve vytvořených postav. Tímto je testovací prostředí připraveno pro zavolání testovací metody. Funkce setTestData() je volána před každou testovací metodou, aby se zajistilo, že data nejsou pozměněna z předchozích testů a testovací funkce je vždy volána za stejných podmínek.

Ve třídě je dále deklarován komparátor, který porovnává dvě instance objektu GURPSSkill a vyhodnocuje, zda jsou všechny atributy obou instancí shodné. Metody isEquals je volána z komparační funkce, slouží pouze k porovnání kolekcí výchozích hodnot daných schopností.

Dvojice funkcí setUp() a tearDown() zajišťuje nezávaznost jednotlivých testů. Tyto funkce před každým testem zahodí a znovu vytvoří testovací prostředí.

Funkce testGetSkills() vytvoří dvě instance GURPSSkill (každou jiné postavě). Poté zavolají nad DAO funkce getAll() pro obě postavy a porovnají, zda kolekce schopností získaná z DAO odpovídá původní kolekci před zápisem do databáze.

Funkce testDeleteSkills() testuje, zda jsou z DAO správně vymazávány instance schopností. Funkce postupně odmazává záznamy z databáze a poté testuje, zda se záznam opravdu smazal.

Funkce testUpdateSkills() uloží do databáze instanci schopnosti a pak postupně změní všechny atributy dané schopnosti. Poté zavolá funkci update datového objektu. Nakonec voláním metody get() obdrží objekt z databáze a porovná ho se změněným objektem.

3.3.2 Testy logické vrstvy

Zatímco testy datové vrstvy testují správnost operací nad datovým úložištěm, testy business vrstvy mají za úkol ověřit, zda business funkce pracují správně a vracejí výsledky, jaké se od nich očekávají. ze tříd logické vrstvy, které jsou v balíčku k dispozici, jsem pro ukázku vybral testovací soubor třídy GURPSCharacterBOTest.

Třída obsahuje víc business funkcí, pro otestování jsem vybral funkci getCostOfCurrentAttributeValue(). Tato funkce vypočítává cenu bodů osobnosti, a to od jedné hodnoty k druhé. Výpočet je o něco složitější: u postavy jsou k dispozici tři hodnoty daného atributu - výchozí hodnota rasy, výchozí hodnota po vytváření postavy a současná hodnota. Cena je definována v souboru AttributeConstants. od výchozí hodnoty rasy k výchozí hodnotě po vytváření postavy je cena nezměněna, avšak druhá část výpočtu od výchozí hodnoty po vytváření postavy do současné hodnoty je násobena dvěma (podle pravidel GURPS-Lite [1]).

Testovací soubor obsahuje ještě testovací metody pro jiné funkce, avšak do ukázky kódu jsem zahrnul pouze ty metody, které jsou relevantní pro zde popisovanou metodu getCostOfCurrentAttributeValue.

V souboru GURPSCharacterBOTest testuji různé kombinace vstupních hodnot do této funkce abych tak ověřil, že všechny možné průběhy výpočtu funkce pracují správně.

Následující ukázka poskytuje náhled na strukturu testovacího souboru:

```
1  public class GURPSCharacterBOTest extends AndroidTestCase {
2      private void setInitialData();
3
4      public void setUp() throws Exception;
5
6      public void tearDown() throws Exception;
7
8      public void testGetCostOfCurrentAttributeValue() throws Exception
           {
9          setInitialData();
10         assertEquals(errorMessage1, 0, characterBO.
               getCostOfCurrentAttributeValue(0, 10, 10, 10));
11         assertEquals(errorMessage1, 20, characterBO.
               getCostOfCurrentAttributeValue(0, 10, 10, 11));
12         assertEquals(errorMessage1, -10, characterBO.
               getCostOfCurrentAttributeValue(0, 10, 11, 10));
```

```
13          assertEquals(errorMessage1, 10, characterBO.
                getCostOfCurrentAttributeValue(0, 10, 11, 11));
14
15          ...
16      }
17  }
```

Podobně jako v testech datové vrstvy, i zde jsou k dispozici funkce setUp() a tearDown(), které zajišťují stejné výchozí podmínky pro všechny testy.

Funkce setInitialData() má na starosti inicializaci objektů třídy GURPSCharacter a GURPSCharacterBO.

Pak už je volána opakovaně funkce getCostOfCurrentAttributeValue s různými parametry. Výsledky jsou testovány oproti předpočítaným hodnotám.

3.3.3 Testy prezentační vrstvy

Testování prezentační vrstvy probíhá formou uživatelských testů. Uživatel dostane scénář testování a poté spustí aplikaci. v aplikaci postupuje podle pokynů ve scénáři a testuje tak správný chod aplikace.

3.3.3.1 Testy tvorby postavy

První scénář uživatele provede tvorbou postavy. pro úspěšné provedení testu se předpokládá, že má uživatel nainstalován aplikaci MobChara a balíček GURPS. Tento scénář popíšu podrobněji, u ostatních scénářů spíše nastíním základní principy testování a jednotlivé části aplikace, na které se testování zaměří.

Scénář

1. Tvorba postavy

   a) v úvodní obrazovce klikneme na tři tečky v pravém horním rohu představující menu aplikace. Vybereme možnost Vytvořit postavu a v dialogovém okně vybereme GURPS. Aplikace spustí obrazovku, ve které se nacházejí textová pole pro vložení informací o postavě. do textových polí vyplníme libovolné informace.

   b) v selektoru rasy vybereme buď přednastavenou rasu Human, nebo položku s názvem Custom. Pak klikneme na ikonu oka vedle selektoru. v aplikaci se otevře nová obrazovka - pokud jsme si vybrali přednastavenou rasu, uvidíme pouze souhrn informací o ní, pokud jsme si vybrali Custom rasu, jsou informace editovatelné. v takovém

případě do textových polí nastavíme libovolné informace. Pokud jsme si vybrali Custom rasu, máme také k dispozici čtyři selektory hodnoty umožňující nastavit výchozí hodnoty základních atributů dané rasy. Nastavíme dle libosti.

c) na původní obrazovku tvorby postavy se navrátíme stisknutím šipky zpět (v případě přednastavené rasy) nebo symbolu diskety uložením informací o rase (v případě vlastní rasy). na obrazovce vidíme čtyři selektory hodnoty pro zvolení výchozí hodnoty primárních atributů naší postavy. Tyto selektory nastavíme dle libosti.

d) po dokončení editace informací vytvoříme postavu kliknutím na symbol diskety v pravém horním rohu obrazovku. Tato akce uloží postavu do databáze a aplikace se přepne zpět do obrazovky pro výběr postavy. v tomto seznamu již uvidíme naši nově vytvořenou postavu.

3.3.3.2 Detail postavy

Druhým scénářem sloužícím k otestování aplikace uživatelem je scénář zobrazení detailu postavy. Tento scénář provede uživatele zobrazením a editací základních informací o postavě. Předpokladem k tomuto testu je vytvořená postava v balíčku GURPS.

V tomto scénáři uživatel v hlavní obrazovce aplikace klikne na svoji vytvořenou postavu, načež uvidí detaily postavy. uživatel otestuje editovatelnost informací tak, že postupně pomocí dlouhého kliknutí vyvolá obrazovku pro editování konkrétních částí obrazovky. Uživatel poté libovolně informace zedituje a pomocí symbolu diskety uloží změny. Poté zkontroluje, zda se změny projevily na obrazovce s detaily o postavě.

3.3.3.3 Inventář

Dalším scénářem je scénář pro otestování inventáře. v tomto scénáři uživatel otestuje tvorbu jednotlivých předmětů - postupně vytvoří předměty od každého typu. Dalším krokem testování bude ověření, zda po kliknutí na předmět jsou v detailu předmětu zobrazeny ty samé informace, které při tvorbě zadal.

Uživatel ještě vyzkouší vytvořit batoh, do něhož se pak kliknutím přesune. Zde opět vyzkouší tvorbu předmětu, aby otestoval správnou hierarchii inventáře.

Uživatel dále vyzkouší změnit stav předmětu z nošeného na nenošený (a poté zase zpět). při každé změně by měl být předmět v seznamu předmětů zesvětlený (pokud nošen není) nebo v klasických barvách (pokud nošen je).

Nakonec ještě uživatel vyzkouší editovat jednotlivé informace o předmětech a změny uložit. Pak zkontroluje, zda se změny uložily.

3.3.3.4 Sety

Další scénář má za úkol otestovat tvorbu a editací setů předmětů. Doporučeným předpokladem pro tento test je existence alespoň jednoho předmětu v inventáři.

Uživatel nejprve vytvoří nový set a zkontroluje, jestli je vytvořen a jestli obsahuje všechny informace, které zadal. Dále zkusí do setu přiřadit věci z inventáře.

Druhou částí testu setů je editace informací existujícího setu a ověření, jestli se změněné informace uložily. Dalším úkonem je změna předmětů náležících danému setu.

3.3.3.5 Rysy, schopnosti a kouzla

Tyto tři scénáře sice testují každý jinou obrazovku, ale vzhledem k totožné struktuře testování jsem je zahrnul do jednoho celku.

Uživatel vyzkouší tvorbu rysu, nastavení libovolných vlastností rysu a po uložení opět zkontroluje, zda se informace správně uložily. Poté rys rozklikne a změní některé informace. Dále set uloží a zkontroluje korektní uložení změn.

Podobný scénář bude testován i pro schopnosti a kouzla. u schopností navíc ještě uživatel otestuje nastavování a rušení aktuální hodnoty schopnosti stejně jako nastavení žádné/jedné/více výchozích hodnot.

3.3.3.6 Kontrola výpočtů

Posledním scénářem uživatelského testování je kontrola správnosti výpočtů údajů, které nejsou uživatelský editovatelné. Scénář obsahuje takových kontrol hned několik.

První skupinou kontrol jsou testy v záložce Combat. Uživatel nasazuje a svléká sety zahrnující zbraně, brnění a štíty a kontroluje, jestli pouze vybavení nasazených setů je k dispozici v bojovém přehledu. Dále zkouší u předmětů měnit vlastnost, zda jsou neseny postavou, a znovu testuje, zda se nesené předměty u postavy zobrazí a odložené předměty zobrazeny nejsou.

Další skupinou testů jsou testy schopností. Kouzla typu Missile vyžadují k zobrazení schopnosti jimi vrhnout, aby postava měla naučenou schonpost „Spell throwing“. Uživatel vyzkouší nastavit/rušit tuto schopnost a měnit její hodnotu a zjišťuje, jestli se hodnota v záložce Combat správně aktualizuje. Tento samý test uživatel provede také u zbraně. Podle toho, jakou zbraň vytvořil, edituje atribut/schopnost, na kterých závisí schopnost boje s danou zbraní, a ověří aktualizaci údajů v záložce Combat.

Poslední skupinou testů jsou testy výpočty naložení postavy a stupně zatížení. Uživatel přidává a odebírá předměty z inventáře a nastavuje jim libovolné váhy, a přitom kontroluje, jestli součet vah v inventáři koresponduje s celkovou zátěží zobrazenou v záložce „Character overview“ v sekci „Movement & Encumbrance“.

Kapitola 4

Příručka pro uživatele

Součástí práce je také příručka usnadňující uživatelům orientaci a používání aplikace. Příručka je k dispozici na přiloženém CD.

Závěr

V práci Mobchar - balíček pro GURPS jsem analyzoval pravidla pro hraní her na hrdiny GURPS. Dále jsem navrhl rozšiřující balíček, který rozšiřuje stávající aplikaci MobChar. Balíček jsem poté podle návrhu implementoval, integroval jsem jej do stávající aplikace a otestoval. Nakonec jsem vytvořil uživatelskou příručku, která by měla pomoci budoucím uživatelům lépe se v aplikaci orientovat.

Budoucnost projektu

GURPS je velice rozsáhlý soubor pravidel, který obsahuje stovky různých knih. Záleží pouze na hráčích, jak pokročilí jsou ve hrách na hrdiny. Mohou používat ke hraní pouze základní pravidla, pro které je tento balíček určen, mohou se však rozhodnout používat některá další pravidla. Logickým rozšířením mého balíčku tedy je rozšíření pro některé hojně využívané knihy pravidel, například pro hru rozšiřující GURPS o detailní popis kouzel.

Další funkcionalitou, která by velmi pomohla stávající aplikaci, je import a export postavy, který jsem nestihl implementovat. Import a export postav by byl velice praktickou funkcionalitou, která by se dala hojně využít při praktickém používání aplikace.

Posledním rozšířením, které by balíček obohatilo, je implementace přesouvání předmětů v inventáři. Tuto funkcionalitu jsem také nestihl implementovat. Aplikace se dá používat i bez přesouvání věcí, avšak možnost přesunout věci do jiného batohu by obohatila herní zážitek a zjednodušila ovládání.

Dosažení cílů

Kromě výše zmíněných dvou funkcionalit, které jsem nestihl, se mi podařilo splnit všechny dílčí úkoly. Vytvořil jsem funkční balíček umožňující vytvořit postavu podle pravidel GURPS Lite.

Práce mě obohatila v mnoha směrech. Naučil jsem se programovat pro platformu Android, stejně tak jsem si vyzkoušel vytvořit aplikace od základního návrhu až po finální testování.

Nicméně jsem při implementaci často postupoval přístupem pokus-omyl, jelikož jsem nikdy dříve na mobilní platformu aplikaci nepsal. Věřím, že vývojový proces následující aplikace pro Android bude díky zkušenostem z tohoto projektu daleko snazší a plynulejší.

Tvorba aplikace mi zabrala spoustu čistého času, pravděpodobně o dost víc, než jsem původně odhadoval. Jsem proto rád, že se mi vše podařilo dokončit ve stanoveném termínu i přes všechny obtíže, které se objevily během vývoje aplikace.

Literatura

[1] Jackson, S.: GURPS Lite, An Introduction To Roleplaying. [online], [cit. 2017-03-24]. Dostupné z: http://www.sjgames.com/gurps/lite/3e/gurpslite.pdf

[2] Nižaradze, J.: *MobChar - jádro aplikace*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

[3] Pastorek, T.: *MobChar - balíček pro Dungeons and Dragons*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

[4] Core J2EE Patterns - Data Access Object. [online], [cit. 2017-05-09]. Dostupné z: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

[5] Gupta, L.: Singleton Design Pattern in Java. [online], [cit. 2017-05-09]. Dostupné z: http://howtodoinjava.com/design-patterns/creational/singleton-design-pattern-in-java/

[6] Factory Pattern. [online], [cit. 2017-05-10]. Dostupné z: http://www.oodesign.com/factory-pattern.html

[7] Hlavík, J.: 3. díl - Android programování - Životní cyklus a nový projekt. [online], [cit. 2017-05-11]. Dostupné z: https://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-zivotni-cyklus-a-novy-projekt

[8] Fragments. [online], [cit. 2017-05-11]. Dostupné z: https://developer.android.com/guide/components/fragments.html

[9] Semecký, V.: Android Studio – nové vývojové prostředí. [online], [cit. 2017-05-14]. Dostupné z: https://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/

Příloha A

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

DrD Dračí doupě

DnD Dungeons & Dragons

GM Game master

SR Success roll

RR Reaction roll

NPC Non-player character

ST Strength

DX Dexterity

IQ Intelligence

HT Health

RoF Rate of fire

DAO Data access object

BO Business object

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF