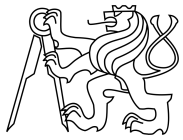
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Tvorba modulu pro sériovou komunikaci s RFID čtečkou na platformě Netbeans
Student:	Marcel Morávek
Vedoucí:	Ing. Radek Dobiáš, Ph.D., MBA
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Na platformě NetBeans vytvořte moduly pro komunikaci s RFID čtečkou. Komunikace se čtečkou bude probíhat přes sériové rozhraní RS-232. Funkčnost jednotlivých modulů bude odpovídat příslušným vrstvám komunikace.

Dílčí úkoly v rámci práce:

1) Proveďte analýzu současného řešení.
2) Proveďte analýzu knihoven podporujících sériovou komunikaci. Identifikujte jejich odlišnosti přístupu v rámci míry jejich abstrakce. Zohledněte licenční podmínky jednotlivých knihoven.
3) Proveďte návrh implementace funkčního modulu. Prozkoumejte využití návrhových vzorů použitelných pro sériovou komunikaci a pro ilustraci funkčnosti modulů využijte vhodné UML diagramy.
4) Implementujte moduly podle vámi provedeného návrhu a patřičně jej otestujte.
5) Vytvořte dokumentaci pro vytvořené moduly.

Práce bude vedena formou projektu s tím, že student bude vykonávat funkci manažera, analyzátora, výkonného pracovníka. Zákazníkem je Vývojové pracoviště VPR12, AŽD Praha. s.r.o. Vedoucí práce vykonává funkci sponzora.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 6. ledna 2016

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

Bakalářská práce

Tvorba modulů pro sériovou komunikaci s RFID čtečkou na platformě Netbeans

Marcel Morávek

Vedoucí práce: Ing. Radek Dobiáš, Ph.D., MBA

16. května 2016

Poděkování

Rád bych tímto poděkoval svému vedoucímu p. Ing. Radkovi Dobiášovi za zprostředkování a umožnění realizace této práce pro společnost AŽD a jeho připomínky k organizaci projektu. Stejně tak bych rád poděkoval p. Ing. Michaelu Charvátovi za věcné připomínky týkající se analýzy a vývoje na platformě NetBeans a zároveň za připomínky k textové části 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 a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 16. května 2016 .....

Odkaz na tuto práci

Morávek, Marcel. *Tvorba modulů pro sériovou komunikaci s RFID čtečkou na platformě Netbeans*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá návrhem a tvorbou Netbeans modulů určených pro sériovou komunikaci s RFID čtečkou přes rozhraní RS-232. Součástí návrhu je průzkum použitelných knihoven umožňujících přístup k sériovým portům a jejich případné využití v implementaci.

Klíčová slova Seriová komunikace, Java, Netbeans, RFID, RS-232

Abstract

This bachelor's thesis describes the design and implementation of Netbeans modules for serial communication with RFID reader via RS-232 interface. The design includes a survey of useful libraries providing access to serial ports and their possible use in implementation.

Keywords Serial communication, Java, Netbeans, RFID, RS-232

Obsah

Seznam obrázků

Úvod

Moderní veřejná doprava se dá hodnotit z mnoha pohledů, ale mezi ty důležitější zcela nepochybně patří pohodlí, rychlost a bezpečnost. O všechny tři zmíněné se na vybraných železničních tratích nově začíná starat evropský vlakový zabezpečovací systém ETCS (*European Train Control System*) [1].

Plošným zavedením tohoto systému bude docíleno sjednocení odlišných národních zabezpečovacích systémů v Evropě a odstraní se tak problémy s přeshraničním provozem. Celý systém lze rozdělit do dvou hlavních částí na traťovou (*infrastrukturní*) a palubní (*mobilní*) část.

Jedním ze základních prvků infrastrukturní části je eurobalíza. Jedná se o pevné traťové zařízení umístěné mezi kolejnicemi, které slouží ke komunikaci s mobilní částí systému ETCS upevněném na projíždějícím vozidle. Z eurobalíz musí být zajištěn velmi rychlý a spolehlivý přenos dat. Je totiž nutné zajistit přesun informací až do rychlosti 500 km/hod projíždějícího vlaku [2].

Systém ETCS je již na řadě tratí zprovozněn, přesto je doposud ve stádiu vývoje. Jednou z firem, které se podílí na vývoji a testování eurobalízy u nás je i firma AŽD Praha, s.r.o. K tomu využívá interní aplikaci realizovanou na platformě NetBeans, plnící různé potřebné funkcionality. K identifikaci eurobalízy se používají uvnitř umístěné RFID tagy.

Používaná interní aplikace je tvořena devíti moduly a jedním z nich je i modul zajišťující komunikaci s RFID čtečkou, který pro sériovou komunikaci využívá knihovnu RXTX. Tato knihovna již v současné době není vyvíjena a některými operačními systémy podporována a proto se zadavatel práce AŽD rozhodl vyhledat nové řešení pro tento modul. A právě návrh spolu s realizací nového modulu je předmětem této práce.

Práce zahrnuje popis kompletního vývoje od zadání prvního požadavku zadavatelem přes analyzování použitelných knihoven spolu s technologií RFID, návrhu tříd, implementaci až k závěrečnému testování funkčnosti vytvořeného modulu.

Kapitola 1

Cíl práce

Cílem této práce je navrhnout a implementovat modul, který bude umožňovat sériovou komunikaci s RFID čtečkou přes sériový port RS-232. Jelikož je v současném řešení používána knihovna RXTX, která již není vyvíjena a podporována, je zapotřebí vyhledat alternativní knihovnu a v novém řešení ji využít. V rámci potřebného otestování funkčnosti modulu je žádoucí vytvořit další modul, jenž umožní odesílat data do RFID čtečky a zároveň zachytit případnou odpověď. Aby činnost byla viditelná, je nutné implementovat jednu z funkcí požadované od RFID čtečky.

1.1 Rešeršní část

Jedním z cílů rešeršní části je vyhledání a prozkoumání použitelných knihoven umožňujících přístup k sériovým portům. Nalezené knihovny následně porovnat a vybrat knihovnu, která bude použita při implementaci požadovaného modulu.

Nadále je zapotřebí se seznámit s vývojem Rich Client aplikací a to především na platformě NetBeans.

Jelikož pro znázornění funkčnosti RFID čtečky bude zapotřebí základní znalost RFID technologie, je žádoucí, aby i ta byla v rámci této práce popsána.

1.2 Praktická část

Cílem praktické části je analýza aktuálně používaného řešení, návrh a implementace nových modulů. Součástí bude i zpracování UML diagramů pro zobrazení požadavků a znázornění navržených tříd s popisem funkčnosti stěžejních metod. Výsledná aplikace musí být otestována s RFID čtečkou používanou společností AŽD. K vytvořeným třídám je nutné vytvořit dokumentaci.

1.3 Co není cílem práce

Ačkoli cílem práce je vytvoření nového modulu pro sériovou komunikaci přes rozhraní RS-232 spolu s jeho otestováním vůči RFID čtečce, tak není požadována realizace veškeré funkčnosti očekávané od RFID čtečky. Není tedy nutné implementovat velké množství akcí, které konkrétní čtečka používána společností AŽD umožňuje, ale pouze provést pár úkonů, jenž funkčnost modulu dostatečně znázorní.

Kapitola 2

Platforma Netbeans

Jelikož zadavatelem požadovaný modul musí být realizován jako modul na platformě NetBeans, je zde uvedena tato kapitola, ve které je popsána obecná hierarchie Rich Client, ze které následně vychází i platforma NetBeans.

Rich Client jakožto programová architektura, je aplikace, v níž dochází ke zpracování dat na straně klienta. Jelikož velké množství desktopových aplikací má celou řadu společných vlastností a prvků, tak je žádoucí, aby byl vývoj takových aplikací co možná nejvíce usnadněn. A právě k tomu slouží dále popsána platforma Rich Client [3].

2.1 Platforma Rich Client

Rich Client platforma je program, který poskytuje prostředí pro životní cyklus aplikace a je základem desktopové aplikace. Nabízí velké množství komponent jako jsou například úvodní obrazovky, nástrojové lišty, nástroje pro zobrazení nebo zápis dat a mnoho dalších.

Nejdůležitější vlastností Rich Client platformy je její architektura. Aplikace se vytvářejí ve formě modulů, jenž obsahují jednotlivé logicky ucelené části aplikace.

Rich Client platforma sebou přináší následující výhody:

- zkrácení doby vývoje
- konzistence uživatelského rozhraní
- znovupoužitelnosti funkcí a modulů
- nezávislost na platformě

2.2 Platforma NetBeans

Kromě základních předností Rich Client platformy nabízí platforma NetBeans velmi příznivou podporu uživatelského prostředí a také vlastní editor. Dále lze například využívat již vytvořené moduly a použít je jako pluginy, které lze jednoduše přidat do aplikace. To se hodí, pokud jsou v rámci vývojového týmu vyvíjeny některé funkcionality opakovaně a jejich implementace je poté realizována pouhým vytvořením vazby na potřebný modul.

2.2.1 Modularita

Stejně tak jako Rich Client platforma, tak i platforma NetBeans je založena na modulové architektuře. Jednotlivé moduly by měli obsahovat své vlastní API (*Application Programming Interface*) a zpřístupnit tak modulovou funkcionalitu ostatním. To umožňuje upravovat výslednou aplikaci přidáváním a odebíráním nezávislých modulů. Některé moduly mohou mít definovanou závislost na jiné a potom je nutné mít v aplikací přítomné veškeré závislé moduly.

Modul lze použít i jako vlastní knihovnu a to jako speciální modul *Library wrapper*, který může obsahovat jednu ale i více potřebných externích knihoven. Poté lze jednoduše nastavit z ostatních modulů na obalující knihovní modul potřebnou závislost.

Základem modulární architektury je běhový kontejner (*NetBeans Runtime Container*), jenž představuje minimální skupinu modulů, které aplikace platformy NetBeans vyžadují pro svůj běh. Tento kontejner se skládá z následujících modulů [3]:

- **Bootstrap** - Tento modul je spuštěn jako první s tím, že zavolá všechna registrované handlery pro příkazovou řádku a připraví počáteční zavaděč tříd.
- **Startup** - Nastartuje aplikaci, inicializuje modulový systém a modul souborového systému.
- **Module System** - Odpovídá za správu modulů, jejich nastavení a závislosti.
- **File System** - Zpřístupní virtuální datový systém s přístupem nezávislým na platformě. Použije se k načtení zdrojů modulů do aplikace.
- **Utilities** - Poskytuje základní komponenty například pro komunikaci mezi moduly

2.2.1.1 Struktura modulu

Modul je realizován JAR souborem, které obsahuje následující části:

- **Manifest.mf** - Je textový soubor obsahující informace o modulu a jeho rozhraní, závislostech a prostředí. Jedná se například o jediný povinný atribut s názvem modulu, jenž se využívá při aktualizaci nebo deklarování závislosti.
- **Layer.xml** - V tomto souboru jsou definovány veškeré informace o tom, co modul přidává do platformy NetBeans. Existence a pojmenování konfiguračního souboru je definována v manifestu.
- **Class soubory** - Obsahují veškerou implementaci funkcionalit modulu.
- **Zdroje** - Zahrnují ikony, properities, soubory lokalizace, nápovědu atd.

2.2.1.2 Typy modulů

Moduly podle času, kdy jsou zařazeny do aplikace, lze rozdělit na následující typy:

- **Regular** - Načítá se již při startu aplikace, který je prodloužem o dobu inicializace modulu.
- **Autoload** - Start tohoto modulu musí být vynucen jiným modulem. Využívá se zde principu odloženého načtení.
- **Eager** - Tyto moduly jsou načteny pouze jsou-li splněny všechny závislosti.

Kapitola 3

Knihovny pro přístup k sériovým portům

Jedním ze stěžejních bodů vývoje modulu pro sériovou komunikaci je výběr knihovny, jenž bude schopna nahradit aktuálně používanou knihovnu RXTX. A právě touto problematikou se zabývá tato kapitola.

V následujících podkapitolách jsou uvedeny tři zástupci knihoven nebo rozhraní, umožňující přístup k sériovému portu. Poté jsou jednotlivý zástupci navzájem porovnány a celá kapitola je pak zakončena mnou vybranou knihovnou pro další použití ve vývoji vytvářeného modulu.

3.1 Java Communications API

Java Communications API je rozšíření jazyka Java, jež umožňuje aplikacím přistupovat ke komunikujícím periferiím, jako jsou čipové karty, vestavěné systémy, pokladny, robotická zařízení a mnohé další. Java Communications API poskytuje aplikacím přístup k sériovému rozhraní RS-232 a s omezeným přístupem v kompatibilním režimu k paralelnímu portu IEEE-1284. [4]

Jádro rozhraní tvoří abstraktní třída CommPort a její podtřídy SerialPort a ParallelPort. Třída CommPort zastává pozici univerzální komunikační třídy a její potomci implementují konkrétní druh přístupu k portu. Pro vytvoření objektu představujícího port je nutné využít statickou metodu open třídy CommPortIdentifier, jenž poskytuje seznam dostupných portů, z nichž si programátor, respektive uživatel, jeden vybere. [15]

Třída SerialPort má metody poskytují InputStream a OutputStream, přes které probíhá zápis a čtení dat na sériovém portě [5].

Implementace Java Communications API je v současné době dostupná pro následující operační systémy:

- Solaris SPARC

- Solaris x86
- Linux x86

Pro použití rozhraní Java Communications je zapotřebí souhlasit s Oracle Binary Code License Agreement for Java SE and JavaFX Technologies [6].

3.2 RXTX

RXTX je knihovna, která poskytuje sériovou a paralelní komunikaci pro Java Development Toolkit (*JDK*).

Knihovna RXTX umožňující práci se sériovými porty umí vyhledat všechny dostupné porty a poskytnout jejich seznam. Dále umožňuje nastavit důležité vlastnosti spojení (rychlost přenosu, počet davových bitů, paritu a počet stop bitů). Zápis a čtení je umožněno pomocí proudů InputStream a OutputStream, které lze získat příslušnými metodami od třídy SerialPort [7].

Knihovna je schopna pracovat na těchto operačních systémech:

- MacOS X
- Windows
- Linux

RXTX je poskytován pod GNU LGPL v2.1 licencí a poskytuje stejnou podporu jako javax.comm library [12].

3.3 JSSC

JSSC (*Java Simple Serial Connector*) je knihovna, která umožňuje realizovat běžnou sériovou komunikaci. JSSC nabízí snadno uchopitelné API, které je stále vyvíjeno.

Knihovna je tvořena několika třídami, z nichž nejpodstatnější jsou třídy SerialPort a SerialPortEvent. Třída SerialPort obsahuje metody pro otevření a zápis dat na sériový port. Třídě je pak nadále možno přiřadit posluchače, jež je instancí třídy SerialPortEvent. Pro správnou realizaci naslouchání na otevřen portu je třeba implementovat rozhraní SerialPortEventListener, kde po předefinování metody serialEvent s jediným vstupním parametrem typu SerialPortEvent obsahující informace o typu události spolu s příslušnými daty, která jsou nutná dále zpracovat [13].

Knihovna je schopna pracovat na těchto operačních systémech:

- Windows
- Linux

- Solaris
- OS X

A to jak na 32, tak na 64 bitovém systému.

Knihovna jSSC je poskytována pod licencí GNU LGPL [14].

3.4 Porovnání knihoven

Pro prostudování a vyzkoušení některých knihoven umožňujících přístup k periferiím zde uvádím rozdíly, které jsem vysledoval. Jednotlivé odlišnosti jsem rozdělil do tří kategorií, které jsou popsány v podsekcích níže. Původně jsem měl v plánu porovnávat knihovny pouze na základě technických parametrů a vlastností, ale v průběhu zkoumání jsem vysledoval rozdílnou aktivitu komunity vyvíjející nebo používající konkrétní knihovny a přišlo mi zajímavé zde poznamenat pár postřehů.

3.4.1 Dle platformní závislosti

Všechny knihovny se snaží obsloužit běžnými uživateli používané platformy, jako jsou MS Windows a Linuxové distribuce. Nicméně ne všechny to umí. Většina výše uvedených rozhraní a knihoven se navzájem mezi sebou zpravidla liší o jednu zásadní vlastnost. Například jako jediné rozhraní Java Communications API není funkční na platformě MS Windows a naopak jediná knihovna RXTX funguje pod operačním systém MacOS X.

3.4.2 Dle míry abstrakce

Ač jsem očekával opačný výsledek, tak je přístup jednotlivých knihoven k periferiím velmi podobný. Jsou zpravidla tvořeny jednou či dvěma třídami, jenž uchovávají informace o příslušném portě a řídí zbytek tříd realizující zápis dat na nejnižší vrstvě spolu s prostředkováním odpovědi. Ale právě způsob naslouchání odpovědi se napříč knihovnami nejvíce liší.

Rozhraní Java Communications API spolu s knihovnou RXTX používají pro předání odpovědi InputStream, z něhož probíhá čtení cyklickým dotazováním o případných příchozích datech na vstupu. Knihovna jSSC však předávání přijatých dat řeší pomocí událostí obsahující informace o typu události spolu s odpovídajícími daty, které je třeba nadále zpracovávat.

3.4.3 Dle typu použitelných periferií

Jako zásadní rozdíl v možnosti použití je ten, že rozhraní Java Communications API spolu s knihovnou RXTX umožňují oproti knihovně jSSC navíc komunikaci s paralelním portem.

3.4.4 Dle činnosti komunity

V této podkapitole uvádím poznatky, na které jsem narazil v průběhu vyhledávání knihoven a posléze jejich zkoumání z pohledu dostupnosti informací, činnosti tvořící komunity a možnosti případné zpětné vazby od uživatele.

Byl jsem trochu zklamán, když například při pokusu nahlášení chyby u knihovny RXTX pomocí odkazu vedoucího do příslušné části webové stránky jsem byl přesměrován na neexistující stránky. Takových odkazů jsem poté na stránkách objevil více.

Rozhraní Java Communications API má na webu společnosti Oracle sice veškeré odkazy funkční, ale žádný z nich mě nenavedl nikam, kde bych mohl nalézt zkušenosti ostatních vývojářů využívajících toto rozhraní nebo nějaké vize do budoucna.

Zato stále vyvíjená knihovna jSSC a její jednotlivě vydávané verze řešící nově nahlášené chyby jsou systematicky popisovány a lehce přístupny. Poslední verze knihovny 2.8.0 byla vydána dne 24. 1. 2014, ale vzhledem k nahlášeným chybám u nejnovějšího operačního systému Windows 10 se dá očekávat nová verze řešící nahlášené chyby.

3.5 Výběr knihovny

Na základě výše popsaných vlastností a parametrů jsem se nakonec rozhodl pro nahrazení knihovny RXTX knihovnou jSSC, u níž se líbí odlišné pojetí naslouchání portu oproti druhému kandidátovi Java Communications API. Se zpracováváním událostí jsem se totiž doposud nesetkal a je to tak pro mě něco nového, co bych si rád vyzkoušel.

Kapitola 4

RFID

V této kapitole je popsána technologie RFID (*Radio Frequency Identification*) sloužící primárně k bezkontaktní identifikaci objektů.

4.1 RFID

Jedná se o radiofrekvenční systém identifikace používající se k identifikaci objektů pomocí radiových vln. Informace identifikující jednotlivé objekty jsou ukládány v čipu, jenž může mít libovolnou, zpravidla však malou, velikost. Ke čtení a zápisu dat do čipu se používají RFID čtečky různých provedení [21].

4.2 EPC

EPC (*Electronic Product Code*) je navržen jako univerzální identifikátor poskytující jedinečnou identitu pro jednotlivé fyzické objekty. Jeho struktura je definovaná neziskovou organizací EPCglobal.

4.2.1 Struktura EPC

Struktura EPC se vyskytuje ve dvou provedeních lišících se v délce identifikačního čísla. Prozatím se používají ve velikosti 64 a 96 identifikačních číslic. Celá struktura EPC se skládá ze čtyř následujících bloků [20]:

- **Hlavička** - Je přiřazena organizací EPCGlobal. Obsahuje informace o délce, typu, verzi kódu.
- **Číslo vlastníka EPC** - Je přiřazeno organizací EPCGlobal. Slouží jako identifikátor společnosti využívající kód EPC.
- **Číslo produktu** - Přiřazeno vlastníkem EPC. Používá se pro identifikaci druhu výrobku.

- **Seriové číslo** - Přiřazeno vlastníkem EPC. Jednoznačně identifikuje konkrétní kus označeného objektu.

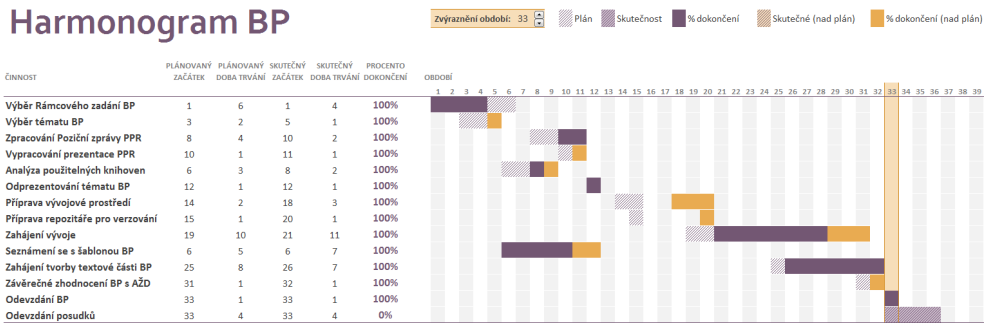
Kapitola 5

Projektové řízení

V této kapitole jsou uvedeny organizační informace, potřebné pro vývoj nových NetBeans modulů, dohodnuté se zadavatelem.

5.1 Časový harmonogram

Pro plánování a znázornění celkového procesu tvoření této práce byl používán Ganttův diagram [8]. Celkový proces vytvoření výsledné aplikace včetně textové části byl naplánován na 33 týdnů. Obecně lze říci, že veškeré analyzační a rešeršní práce spadají do první poloviny časové osy a v druhé je převážná část implementační. Ukázku z harmonogramu zobrazujícího průběh celkové tvorby můžete vidět na obrázku 5.1.



Obrázek 5.1: Časový harmonogram

Harmonogram znázorňuje stav prací v době odevzdávání bakalářské práce.

5.2 Osobní schůzky

Osobní schůzky se zadavatelem AŽD probíhali na základě předem domluvených termínů ve Vědecko-technickém parku ve Mstěticích. V průběhu první,

analyzační, části projektu proběhly 3 schůzky, během niž jsem byl seznámen s požadavky na nový modul a následně byly prodiskutovány mnou vyhledané použitelné knihovny. V průběhu implementační části proběhlo schůzek více a to v mnohem kratších intervalech. Ty probíhaly zpravidla tak, že jsem představil mnou nově vypracované části s případnými problémy, které jsme poté společně řešili.

5.3 Sdílení aktuálního stavu projektu

V počátcích vývoje byly informace mezi mnou a zadavatelem předávány prostřednictvím aplikace sdílených dokumentů od společnosti Google [9].

Ovšem pro sdílení aktuální verze programu a posléze i textové části práce bylo zapotřebí použít lepší způsob sdělování aktuálního stavu a proto jsem pro sdílení použil verzovací systém Git společně s repozitářem uloženém na serveru Bitbucket.

5.3.1 Git

Git je distribuovaný open-source verzovací systém vyvíjený od roku 2005, jenž umožňuje vývojářům mít lokální přístup ke všem verzovaným souborům. Vývojářem provedené změny se nejprve zapisují do lokálního repozitáře a až poté v závislosti na vývojářově rozhodnutí do sdíleného repozitáře, do které již mají přístup všichni ostatní, kteří se nějakým způsobem podílejí na vývoji [10].

5.3.2 Bitbucket

Bitbucket je open-source webová služba, jenž slouží jako podpůrný software při používání verzovacích nástrojů Git a Mercurial. Obsahuje velmi snadno použitelné a uživatelsky příjemné funkcionality issue tracker a wiki [11].

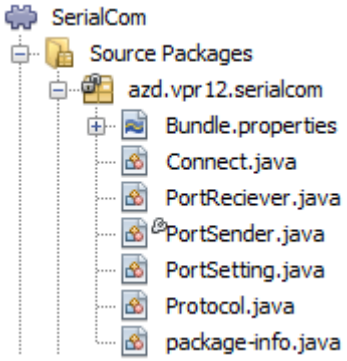
Kapitola 6

Analýza a návrh

Tato kapitola se věnuje analýze současného a mnou navrženého řešení. Obsahuje jednotlivé podkapitoly zaměřující se na zpracování požadavků kladených na výslednou aplikaci, funkčnost RFID čtečky i návrh kompletního třídního modelu.

6.1 Současné řešení

Současné provedení s přístupem na sériový port je realizováno pomocí NetBeans modulu SerialCom, jehož struktura je zobrazena na obrázku 6.1.



Obrázek 6.1: Struktura modulu SerialCom

Stěžejními třídami tohoto modulu jsou třídy Protocol, PortSender a PortReciever. Protokol je řídící třída obstarávající chod celého modulu. Vytvořené instanci SerialPort z knihovny RXTX je přiřazen InputStream a OutputStream, které jsou ovládány z tříd PortSender a PortReciever. Zápis dat probíhá pomocí volání RXTX knihovních metod a čtení neustálým dotazováním portu zda-li jsou k dispozici nová příchozí data.

6.2 Proces práce s RFID čtečkou

Tato kapitola obsahuje popis kompletního procesu při práci s RFID čtečkou, jehož znalost je nezbytná pro správnou realizaci výsledné aplikace. Detailně je zde uveden výčet úkonů, jenž je nutný pro správné provedení čtení a zápisu dat. Čtečka, pro niž je primárně výsledná aplikace určena je **Ha-VIS RFID Reader RF-R200** [18] a té také budou odpovídat veškeré konkrétní údaje a parametry uvedené v této práci.

Základní komunikační údaje budou však ve výsledné aplikace parametrizovatelné a nebude tak funkčnost omezena pouze pro práci s konkrétním typem čtečky. Nicméně konfigurační a jiné speciální zprávy, jež nelze obecně parametrizovat, budou použity pro tento konkrétní typ. Kompletní parametrizace by bylo docíleno pouze sběrem velkého množství příkazů pro všechny případné čtečky a to není předmětem této práce.

6.2.1 Odeslání dat

Pro odeslání dat na RFID čtečku je nutné nejdříve vybrat a posléze otevřít sériový port. Dále je nutné pro správné provedení odeslání dat nastavit následující parametry.

- **Modulační rychlost** - Je jednotka udávající počet změn stavu přenosového média za jednu sekundu.
- **Počet bitů** - Představuje počet datových bitů přenášených v jednom odesílaném bitovém bloku.
- **Koncové bity** - Počet bitů následujících po sekvenci datové části.
- **Paritní bit** - Je jeden bit označující počet jedniček odesílaných v datové části bloku.

6.2.2 Konfigurace

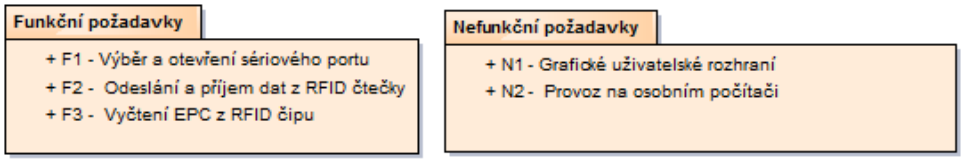
Konfigurace RFID čtečky probíhá odesíláním speciálních pokynů realizující dílčí kroky nastavení čtečky. Jednotlivé příkazy jsou realizovány pomocí sekvence bajtů dané konkrétním typem čtečky.

6.2.3 Příjem dat

Příjem dat od RFID čtečky se z protokolu vynucuje odesláním specifického příkazu. Eventuálně, pokud to konkrétní typ čtečky umožňuje, je možné odeslat data pomocí manuálního úkonu přímo na RFID čtečce například použitím různých spínačů a jiných ovládacích prvků.

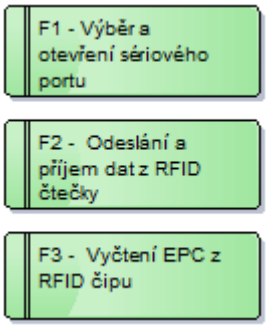
6.3 Model požadavků

V této kapitole jsou uvedeny veškeré požadavky kladené na výsledné moduly, které byly probírány se zadavatelem. Většina z nich byla stanovena ihned po určení rámcového zadání, ale některé byly přidány nebo lehce upraveny v rámci návrhu a implementace tak, jak se upravovalo zadání práce. Následující výčet požadavků je rozdělen do dvou kategorií a to do funkčních a nefunkčních požadavků.



Obrázek 6.2: Seznam požadavků

6.3.1 Funkční požadavky



Obrázek 6.3: Seznam funkčních požadavků

6.3.1.1 F1: Výběr a otevření sériového portu

Proces výběru sériového portu musí být umožněn za běhu aplikace. Aby bylo zabráněno trvalému blokování sériového portu kupříkladu ostatním aplikacím, musí být implementována funkčnost připojení a odpojení vybraného portu ovládaná uživatelem.

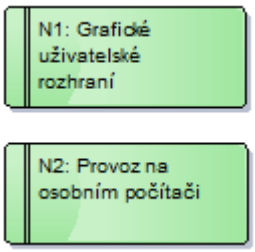
6.3.1.2 F2: Odeslání a příjem dat z RFID čtečky

Operace odeslání dat bude na sériový port posílat data zadaná uživatelem. Kromě zadaných dat musí být umožněno odesílat specifické zprávy představující povely pro RFID čtečku.

6.3.1.3 F3: Vyčtení EPC z RFID čipu

Po odeslání dat na RFID čtečku je zapotřebí odposlouchávat odpověď. Z návrhu protokolu RFID čteček vychází vlastnost zaručené, byť chybové, odpovědi.

6.3.2 Nefunkční požadavky



Obrázek 6.4: Seznam nefunkčních požadavků

6.3.2.1 N1: Grafické uživatelské rozhraní

Aplikace bude umožňovat uživateli provádět jednotlivé potřebné úkony prostředním grafického uživatelského rozhraní.

6.3.2.2 N2: Provoz na osobním počítači

Aplikaci musí být umožněna funkčnost minimálně na operačních systémech Windows a Linux. Dále je pro umožnění veškeré funkcionality přítomnost sériového portu RS-232.

6.4 Návrh uživatelského rozhraní

Dalším z důležitých faktorů aplikace je její uživatelské rozhraní. Pro tuto aplikaci by měla vystačit jedna obrazovka obsahující skupiny komponent pro následující akce.

- **Výběr sériového portu** - Pro potřebu této akce je zapotřebí poskytnout uživateli výběrovou lištu s dostupnými porty, jimž bude umožněna operace aktualizace dostupnosti. V závislosti na požadavku s možností dočasného odpojení vybraného portu je nutné vhodně umístit tlačítka pro připojení a odpojení aplikace k sériovému portu.
- **Odeslání uživatelem zadaných dat** - Aby bylo umožněno uživateli odeslat vlastnoručně zadaná data, je zapotřebí umístit do obrazovky komponentu pro zadávání libovolného textu a tlačítko pro odeslání dat na sériový port.

- **Zpracování specifických požadavků** - Pro potřebu opakování určitého typu požadavku je vhodné do obrazovky umístit komponentu pro výběr ze seznamu úkonů a tlačítko pro provedení požadované akce.

- **Zobrazení přijatých dat** - Poslední komponentou potřebnou pro splnění všech funkcionalit je textové pole pro zobrazování odpovědi. Vzhledem k opakovanému provádění úkonů je zapotřebí uživateli umožnit vyčištění dané komponenty a to ideálně obnovovacím tlačítkem.

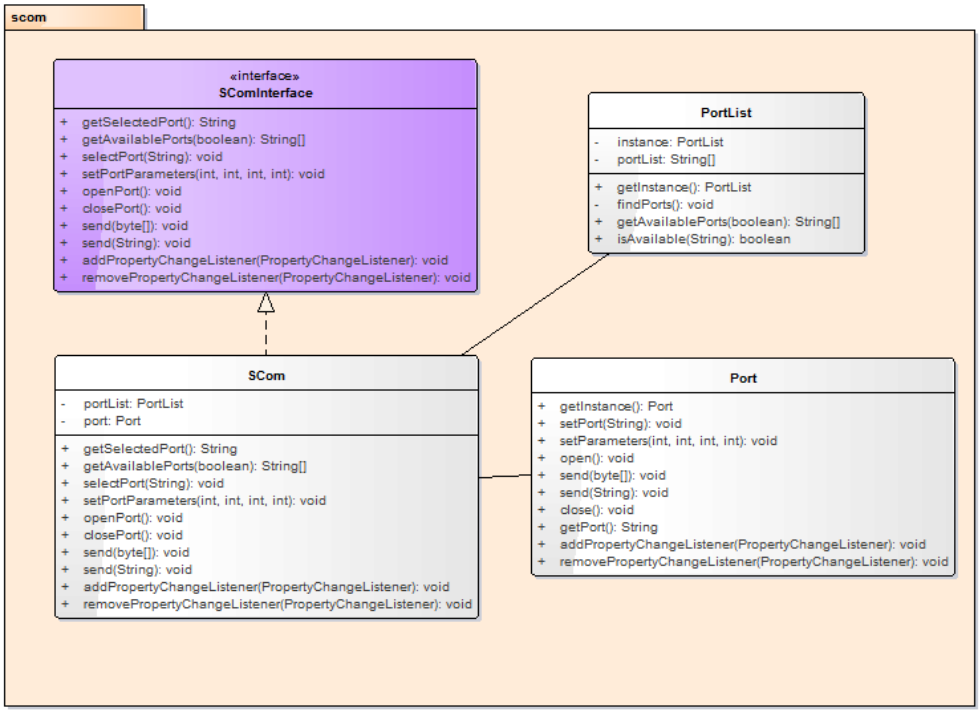
6.5 Výběr používané knihovny pro práci se sériovým portem

Na základě získaných znalostí o použitelných knihovnách s přístupem k periferiím z kapitoly 3, jsem se rozhodl využít knihovnu jSSC. Líbí se mi u ní stále aktivní vývoj s pravidelným zveřejňováním nových verzí a zároveň knihovní struktura včetně přístupu k jednotlivým metodám byla nejshodnější s mojí původní představou o hierarchii takové knihovny.

6.6 Třídní model

V této sekci popisuji navržené třídní struktury pro jednotlivé NetBeans moduly. Z analýzy vyšly dva moduly SCom a RFIDCom, z nichž každý ma svojí přidělenou funkcionalitu. Modul SCom realizuje ovládání sériového portu, které zahrnuje otevírání portu, zapisování a čtení dat. Modul RFIDCom využívá funkcionalit modulu SCom pro realizaci komunikace s RFID čtečkou. Navíc obsahuje uživatelské rozhraní a slouží také pro testování modulu SCom.

6.6.1 Modul SCom



Obrázek 6.5: Třídní návrh modulu SCom.

6.6.1.1 SComInterface

Navržené rozhraní SComInterface definuje základní rozhraní modulu SCom. Níže popisuji výčet navržených metod.

- **getSelectedPort** - Vrací název právě vybraného sériového portu.
- **getAvailablePorts** - Vrací pole názvů všech dostupných portů. Vstupní parametr refresh určuje, zdali má dojít k obnovení seznamu uchovávaného v rámci balíček SCom nebo využití knihovního volání knihovny jSSC.
- **selectPort** - Má jednoduchou funkcionalitu výběru aktuálního portu pro budoucí použití. Jméno portu je jediným vstupním parametrem této metody.
- **setPortParameters** - Nastavuje parametry právě používaného sériového portu. Má tyto čtyři následující vstupní parametry: modulovou rychlost, počet datových bitů, počet koncových bitů a bit paritní.

- **openPort** - Otevře aktuálně vybraný sériový port.
- **closePort** - Uzavře aktuálně vybraný sériový port.
- **send** - Přetížená metoda pro typ vstupního parametru. Vstupním parametrem může být libovolný řetězec nebo pole bajtů.
- **addPropertyChangeListener** - Zaregistruje sériovému portu posluchače předávaného jediným parametrem, jenž bude objekty implementující rozhraní PropertyChangeListener informovat o příchozích datech na sériovém portu.
- **removePropertyChangeListener** - Odregistruje parametrem zadaného posluchače.

6.6.1.2 SCom

Třída implementující rozhraní SComInterface. Z důvodu přetížení některých metod zde bude vhodné kromě metod z rozhraní implementovat i privátní metody pro sdílené části kódu.

6.6.1.3 Port

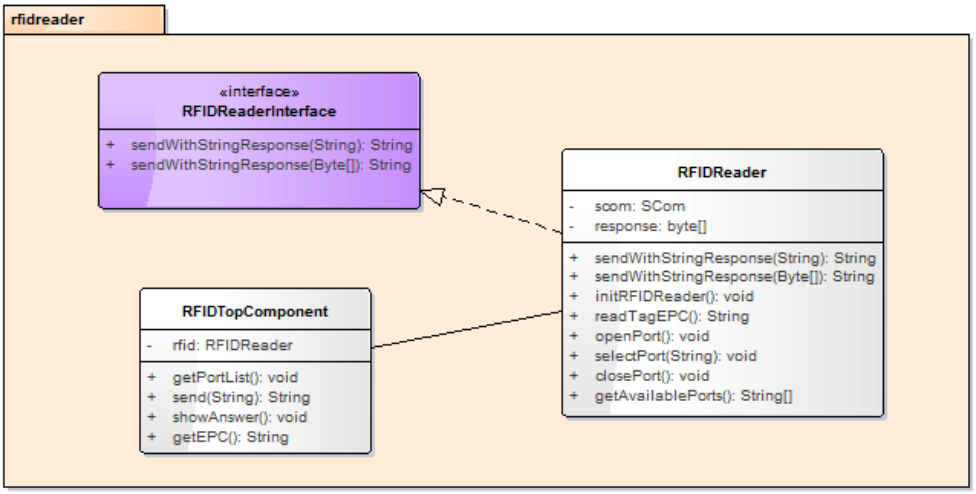
Třída udržuje vlastnosti o vybraném sériovém portu. Implementuje nejdůležitější metody pro nastavení komunikace skrze knihovnu jSSC. Třída se také bude registrovat jako posluchač u třídy SerialPort knihovny jSSC a tudíž zde bude nutné implementovat knihovní metodu serialEvent.

Ta bude mít na vstupu jediný parametr a to SerialPortEvent obsahující informaci o typu události a v případě události odpovídající příchodu dat i konkrétní přijatá data ve formě pole bajtů. V této metodě bude nutné informovat všechny zaregistrované posluchače o proběhlé události a to metodou firePropertyChange.

6.6.1.4 PortList

Poslední třída balíčku SCom obsahuje informace o dostupných portech. Obsahuje metody pro zjištění seznamu aktuálně dostupných portů pomocí třídy SerialList knihovny jSSC, ale také zobrazení seznamu portů uchovávaných přímo v této třídě. Musí také umožnit ověření dostupnosti zadaného portu pomocí metody isAvailable se vstupním parametrem obsahující právě kontrolovaný port.

6.6.2 Modul RFIDCom



Obrázek 6.6: Třídní návrh modulu RFIDCom.

6.6.2.1 RFIDReaderInterface

Navržené rozhraní definuje základní rozhraní modulu SCom. Níže popisuji výčet navržených metod.

- **sendWithStringResponse** - Je jedinou metodou, kterou jsem navrhl do RFIDReaderInterface. V podstatě plné veškerou funkčnost, která se od tohoto balíčku očekává a to odeslání dat na RFID čtečku s odpovědí ve formě řetězce. Jediné co je žádoucí, aby bylo volání odesílacích metod SComu jak pro zadaný řetězec, tak i pro zadané pole bajtů. Proto je tato metoda přetížena dle vstupního parametru.

6.6.2.2 RFIDReader

Třída implementující rozhraní RFIDReaderInterface a PropertyChangeListener.

Z rozhraní RFIDReaderInterface přebírá pouze přetíženou metodu sendWithStringResponse, u které bude vhodné sdílenou část kódu provádět v nové privátní metodě pro budoucí údržbu.

Z rozhraní PropertyChangeListener bude zapotřebí implementovat metodu propertyChange, jenž bude mít na vstupu parametr PropertyChangeEvent obsahující informaci o události. V případě identifikování události od třídy Port z SCom budou obsahovat i přijatá data ze Sériového portu v podobě pole bajtů.

Dále bude tato třída obsahovat následující výčet metod.

- **initRFIDReader** - Metoda realizující řadu speciálních pokynů odesílaných na RFID čtečku. Mezi odesílané zprávy patří například sekvence bajtů „0x02 0x00 0x08 0x00 0x52 0x00 0xB9 0x05“ ověřující správné komunikace, jenž je ověřena totožnou příchozí zprávou. Dále jsou odesílány zprávy například pro nastavení antény, čtení diagnostiky, kontroly verze firmwaru a další.

- **readTagEPC** - Realizuje vyčtení EPC Tagu odesláním příkazu v podobě následující sekvence bajtů „0x02 0x00 0x0A 0x00 0xB0 0x01 0x10 0x08 0x74 0x7C“. Relevantní jsou pouze dvě odpovědi a to za prvé „0x02 0x00 0x08 0x00 0xB0 0x01 0x19 0xCE“ informující o nepřečtení EPC tagu. S největší pravděpodobností proto, že čtený tag není dostatečně blízko RFID čtečce. Za druhé odpověď splňující předepsánou strukturu dat odesílaných z RFID čtečky ať už obsahující požadované data nebo ne.

- **setPort** - Metoda volající metodu balíčku SCom pro nastavení uživatelem vybraného sériového portu.

- **openPort** - Metoda volající metodu balíčku SCom pro otevření sériového portu již vybraného uživatele. Je zapotřebí ošetřit případ, kdy v době volání ještě není port vybrán.

- **closePort** - Metoda volající metodu balíčku SCom pro uzavření sériového portu. Je zapotřebí ošetřit případ, kdy není vybraný port otevřen.

- **getAvailablePorts** - Metoda volající metodu balíčku SCom pro zajištění seznamu dostupných sériových portů.

6.6.2.3 RFIDReaderTopComponent

Jedná se o třídu definující uživatelské rozhraní mé aplikace. Bude využívat NetBeans okenního systému realizovaným kontejnerem spravující nabídky, panely a další nástroje potřebné pro vytvoření grafického uživatelského rozhraní. Třída musí splňovat veškeré požadavky, které jsem analyzoval dříve v kapitole 6.4.

RFIDReaderTopComponent bude obsahovat instanci třídy RFIDReader, jejíž metody bude volat v závislosti na tom, jaké úkony bude uživatel provádět.

Kapitola 7

Implementace

Tato kapitola se zabývá samostatným vývojem navržených NetBeans modulů popsaných v kapitole 6. V rámci implementace jsou zdůrazněny stěžejní funkčnosti jednotlivých modulů spolu s názornými ukázkami. Struktura modulů vychází z proběhlé analýzy a návrhu popsaného v kapitole 6.

7.1 Použité nástroje

V podkapitolách níže popisuji veškeré aplikace, které jsem použil v rámci implementace modulů.

7.1.1 Aplikace Free Virtual Serial Ports

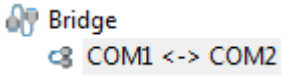
Jelikož můj notebook, který primárně využívám k vývoji, nedisponuje rozhraním RS-232 musel jsem nalézt vhodné řešení, jenž mi umožní plnohodnotně vyvíjet a testovat aktuální stav aplikace.

To jsem nalezl v podobě aplikace Free Virtual Serial Ports [17] určené pro platformu Windows (operační systém Windows Vista a novější), jenž uživateli umožňuje vytvoření softwarového virtuálního sériového portu a simulovat jeho veškerou funkčnost. A to vše dokonce pouze v uživatelském režimu bez nutných administrátorských práv.

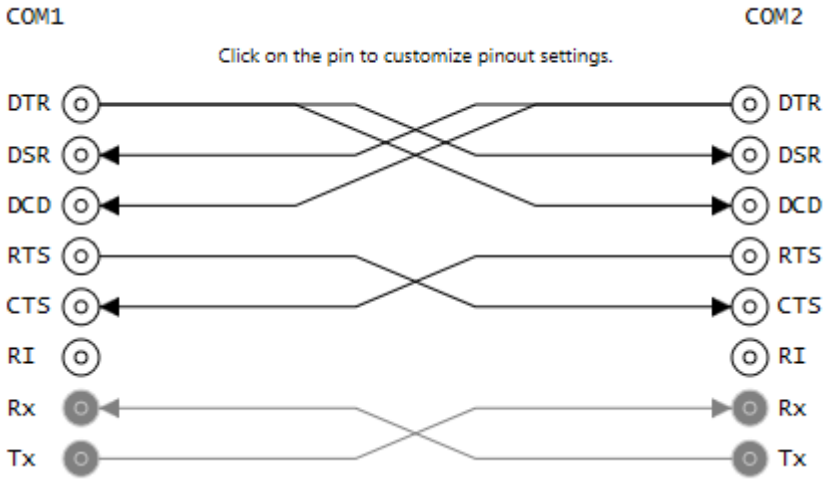
Aplikace poskytuje dva základní druhu vytvoření sériového portu, jenž jsem ve vývoji několikrát využil. Podrobněji jsou popsány níže.

- **Lokální sériový port** - Základní funkčnost aplikace. Dojde k vytvoření pouze naslouchajícího sériového portu. Tuto funkcionalitu jsem využil pouze v počátcích implementace, když docházelo k vývoji modulu SCom pro sériovou komunikaci.

- **Lokální sériový most** - Vytvoření mostu je realizováno vytvořením dvou sériových portů, jejichž konektory jsou vodně navzájem propojeny.



Obrázek 7.1: Vytvořené propojení virtuálních portů [17].



Obrázek 7.2: Propojení jednotlivých pinů sériových portů [17].

7.1.2 Správa verzí

Verzování zdrojových kódů spolu s jejich sdílením se společností ADŽ probíhalo dle dohodnutého režimu popsaného v kapitole 5.3.

7.1.3 NetBeans IDE

Prostředí, ve kterém bude realizován vývoj, je ze zadání práce v podstatě daný. Z důvodu tvorby modulů na platformě NetBeans se přímo nabízí využít prostředí NetBeans IDE, jakožto grafické prostředí s velkou škálou nástrojů pro podporu velkého množství programovacích jazyků jako například Java, PHP, C/C++ plně podporujícího vývoj NetBeans modulů.

Jedná se asi o nejznámější Rich Client aplikaci založenou na platformě NetBeans. Veškerá funkčnost a vlastnosti této aplikace jsou vytvořeny ve formě modulů platformy NetBeans. To umožňuje rozšířit funkčnost prostředí dalšími moduly a přizpůsobit ho tak potřebám konkrétního uživatele.

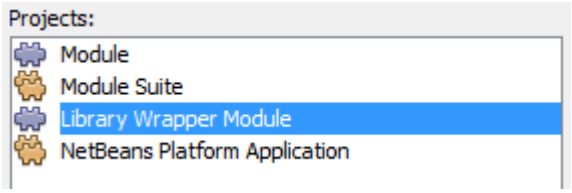
7.2 Moduly

V následující sekci popisuji implementaci jednotlivých modulů.

7.2.1 JSSC

Původně jsem používal pouze moduly SCom a RFIDReader, do nichž jsem importoval externí knihovní soubory typu JAR. Po prostudování knižního průvodce pro vývojáře na platformě NetBeans jsem se seznámil s jiným způsobem používání externích knihoven pro modulární vývoj a ten aplikoval i v mé aplikaci.

Vytvořil jsem tedy úplně nový speciální NetBeans modul pro zabalování knihoven. Tento obalovací modul je ve standardní nabídce průvodců NetBeans IDE jak je možno vidět na obrázku 7.3.



Obrázek 7.3: Seznam typů projektů pro vytváření NetBeans modulů.

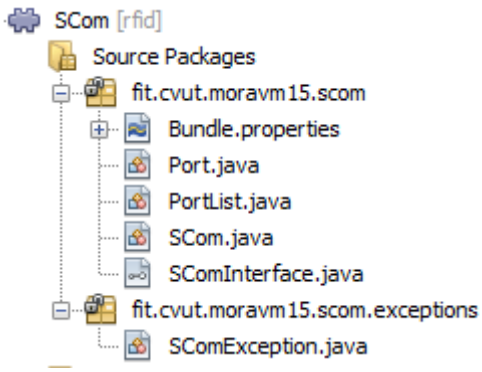
Jedná se o autoload modul, tudíž je jeho spuštění vyvoláno až v případě potřeby obalených knihoven.

7.2.2 SCom

SCom je vůbec první modul, který jsem vytvářel a prvně v něm něco programoval. Ale NetBeans IDE svými průvodci ledacos ulehčí a tak i přes mé velmi omezené znalosti s jazykem Java to nebyl velký problém.

V modulu jsem dle provedeného návrhu vytvořil balíček scom, jehož jméno jsem dle konvencí doporučených společností Oracle pro eliminaci duplicitních názvů pojmenoval fit.cvut.moravm15.scom.

Výsledná hierarchie modulu SCom po všech svých proměnách, které v průběhu vývoje zaznamenala, vypadá tak, jak je vidět na obrázku 7.4.



Obrázek 7.4: Hierarchie modulu SCom.

Rozhraní SComInterface balíčku scom obsahuje všechny, v analýze navržené, metody. Ty jsou pak nadále implementovány ve třídě SCom.

7.2.2.1 Použité návrhové vzory

- **Singleton** - Neboli jedináček je návrhový vzor umožňující globální přístup k jedné instanci tak, aniž by ji bylo nutné nějakým způsobem propagovat napříč objekty pomocí metod a konstruktorů. Navíc při správné implementaci může vzniknout právě jedna instance, což může být ve spoustě případu žádoucí [22].

  Já tento návrhový vzor využívám u instance třídy PortList, jak je vidět v ukázce kódu 1.

Algorithm 1 Vytvoření instance třídy PortList

```
private static PortList instance = null;

public static PortList getInstance() {
  if (instance == null) {
    instance = new PortList();
  }
  instance.findPorts();
  return instance;
}
```

V této ukázce je vidět statická metoda getInstance, která vrací jedinou instanci třídy PortList s použitím návrhové vzoru Singleton.

- **Observer** - Neboli pozorovatel je druh návrhového vzoru, jenž umožňuje reagovat na změny konkrétního objektu případně parametru ve všech registrovaných posluchačích. To se hodí, když je potřeba reagovat na změnu uživateli zobrazovaných dat, u nichž dojde ke změně. Poté je totiž

nutné donutit program nebo dialogové okno zaktualizovat zobrazovaná data [23].

Jelikož data, která SCom přijímá ze sériového portu, jsou přenášena pomocí události, kterou odchytává přepsaná metoda serialEvent z rozhraní SerialPortEventListener, rozhodl jsem se zde použít právě tento návrhový vzor.

V jazyku Java se k implementování posluchače hodí rozhraní PropertyChangeListener, které obsahuje veškeré potřebné metody pro realizovaní procesu informování posluchačů o provedení sledované změny. Ukázku kódu s informováním posluchačů můžete vidět v ukázce kódu 2.

Algorithm 2 Zpracování události na sériovém portu

```
@Override
public void serialEvent(SerialPortEvent evt) {
  if (event.isRXCHAR()) {
    int byteCount = evt.getEventValue();
    if (byteCount > 0) {
      try {
        byte buffer[] = serialPort.readBytes();
        this.pchs.firePropertyChange(SCom.DATA_RECEIVED,
                null, buffer);
      } catch (SerialPortException ex) {
        this.pchs.firePropertyChange(SCom.DATA_ERROR,
                null, ex.getExceptionType().toString());
    }
  }
}
}
```

V algoritmu výše můžete vidět ukázku zpracování přijatých dat. Vyvolaná událost na sériovém portu zavolá metodu serialEvent, z níž jsou vyčtena příchozí data, která jsou všem zaregistrovaným posluchačům předána vyvoláním metody firePropertyChange. Informaci o správném či chybném přečtení dat jsou posluchači informování pomocí speciálních konstant definovaných na úrovni třídy SCom. Jedná se o konstanty DATA_RECEIVED označující správné přečtení a DATA_ERROR odpovídající chybnému stavu. Spolu s těmito konstantami jsou přenášena příslušná data. V pozitivním případě je to pole bajtů přečtených dat a v opačném případě je to text s typem výjimky.

7.2.2.2 Zpracování výjímek

Vzhledem k zpracovávání různých typů výjimek v rámci celého balíčku SCom jsem se rozhodl pro vytvoření výjimkové třídy SComException. Tu uchovávám v novém balíčku cvut.fit.moravm15.scom.exceptions. Veškeré chyby způsobené používáním metod knihovny jSSC nebo chybových stavů v rámci balíčku způsobí vyhození výjimky odpovídající nové třídě SComException. Ukázku můžete vidět v kódu 3.

Algorithm 3 Zjištění vybraného portu s ukázkou zpracování výjimky

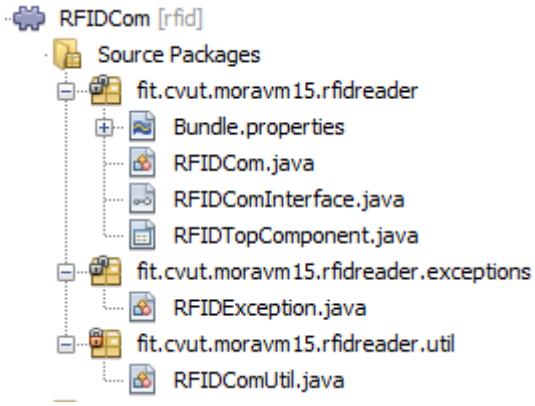
```
@Override
public String getSelectedPort() throws SComException {
  try {
    return port.getPort();
  } catch (SerialPortException ex) {
    throw new SComException(ex.getPortName(),
            ex.getMethodName(), ex.getExceptionType());
  }
}
```

7.2.3 RFIDCom

V modulu jsem dle provedeného návrhu vytvořil balíček rfidreader, jehož jméno jsem dle konvencí doporučených společností Oracle pro eliminaci duplicitních názvů pojmenoval fit.cvut.moravm15.rfidreader. Nadále ale budu používat pouze označení rfidreader.

Tento modul má na starost nejen řízení komunikace na úrovni RFID, ale také zobrazení uživatelského okna pro práci se čtečkou.

K jednotlivým částem výsledné hierarchie modulu RFIDCom, která je na obrázku 7.5, se budu věnovat dále.



Obrázek 7.5: Hierarchie modulu RFIDCom

7.2.3.1 Použité návrhové vzory

- **Observer** - Ve spolupráci s používáním toho návrhové vzoru v balíčku scom je tento vzor použit i v tomto modulu. Ovšem zde plní funkci registrovaného posluchače čekajícího na vyvolání události ze sériového portu [23].

  Toho je dosaženo implementováním rozhraní PropertyChangeListener ve třídě RFIDCom. Implementace spočívá v předefinování metody propertyChange, která je volána v případě změny sledovaného parametru nebo objektu, tak jako tomu dochází v metodě serialEvent uvnitř ve třídě Port v balíčku scom.

  Metoda propertyChange realizující zpracování příchozích dat je znázorněna v ukázce kódu 4.

Algorithm 4 Zpracování události se zápisem do sdíleného bufferu

```
@Override
public void propertyChange(PropertyChangeEvent evt) {
  if (evt.getPropertyName().equals(SCom.DATA_RECEIVED)) {
    byte[] byteArray = (byte[]) evt.getNewValue();
        synchronized (this) {
          for (byte b : byteArray) {
            response.add(b);
          }
        }
  }
}
```

Jelikož není zaručeno, že bude odpověď od čtečky zpracována v rámci

jedné vyvolané události, jsou jednotlivé bajty ukládány do třídní privátního atributu response představujícího buffer. O výlučný přístup k atributu response se stará konstrukce synchronized, která je v této metodě použita. Tímto je zaručeno ukládání veškerých dat do bufferu.

Jeho vyčtení je realizováno vytvořením instance rozhraní ExecutorService, jenž umožňuje správu vláken. Jedním z řízených vláken je pak implementace rozhraní Future, jež je v konstruktoru přiřazena třída implementující rozhraní Callable. Předefinová metoda call je znázorněna v ukázce kódu 5.

Algorithm 5 Třída detekující změny na sdíleném bufferu

```
@Override
public byte[] call() throws Exception {
  byte[] array;
  while (true) {

    TimeUnit.MILLISECONDS.sleep(10);
    synchronized (this) {
      if (response.size() >= 3) {

        int stx = Byte.toUnsignedInt(response.get(0));
        if (stx != 2) {
          throw new RFIDException("stx", "");
        }

        int msb = Byte.toUnsignedInt(response.get(1));
        int lsb = Byte.toUnsignedInt(response.get(2));
        int responseExpectedSize = msb * 256 + lsb;

        if (response.size() == responseExpectedSize) {
          array = new byte[responseExpectedSize];
          for (int i = 0; i < responseExpectedSize; i++) {
            array[i] = response.get(i);
          }
          response.clear();
          return array;
        }
      }
    }
  }
}
```

Ukázka obsahuje i zpracování formátu dat definované v oficiálním [18] manuálu.

Ve výše zobrazené metodě je vidět opětovné použití konstrukce synchronized, čímž je zaručen výlučný přístup k bufferu. V případě detekování odpovědi delší nežli tři bajty, dojde k přečtení prvního bajtu a ověření odpovědi RFID čtečky. Další dva bajty obsahují velikost celkové odpovědi. V případě schody velikosti přijatých dat s vypočtenou předpokládanou velikostí dojde k vrácení pole bajtů a resetování bufferu. V případě nevyhovující nebo žádné odpovědi dojde z nastavení implementace k TimeOutException, která vlákno ukončí.

7.2.3.2 Zpracování vstupních a výstupních dat

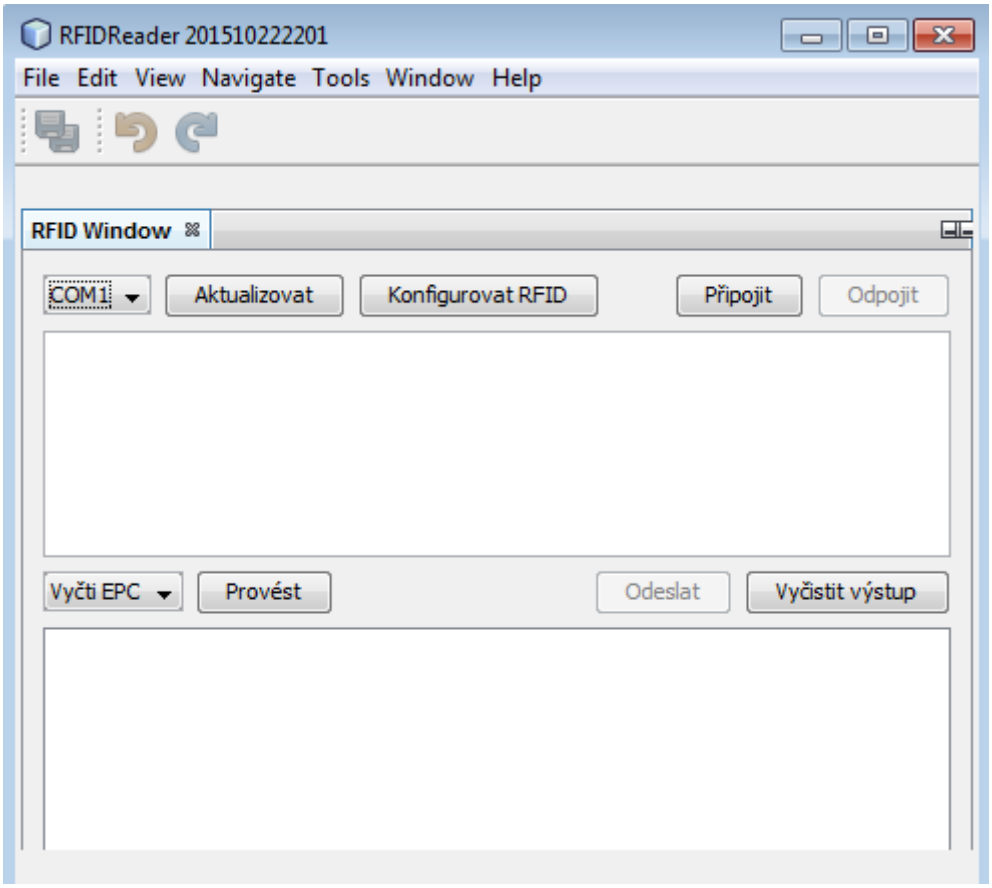
Jelikož je nutné pravidelně zpracovávat vstupní a výstupní data tím, že jsou navzájem převáděna z pole bajtů na řetězec a opačně vytvořil jsem statickou Util třídu RFIDComUtil umístěnou v balíčku cvut.fit.moravm15.util. Tím je umožněno volat jednotlivé metody pro převod bez vytváření instance a zároveň nezávisle na tom, z jaké části kódu se volají, protože je jejich viditelnost nastavena na public.

7.2.3.3 Zpracování výjimek

Zpracování výjimek jsem vyřešil obdobně jako v balíčku scom. Vytvořil jsem nový balíček rfidcom.exceptions, ve kterém je třída RFIDComException odvozená od Třídy Exception. Třída předefinovává dvě základní zděděné metody pro přenos chybové zprávy a to konkrétně metody getMessage a toString.

7.2.3.4 Grafické rozhraní

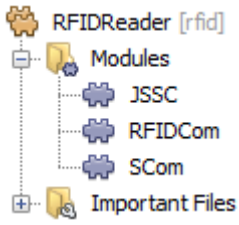
Grafické rozhraní jsem realizoval tak, jak je vidět na obrázku níže.



Obrázek 7.6: Ukázka běžící NetBeans aplikace RFIDReader

7.2.4 RFIDReader

Je NetBeans aplikací obalující již dříve zmíněné a popsané moduly. Struktura celé aplikace je k vidění na obrázku níže.



Obrázek 7.7: Struktura modulu RFIDReader

7.3 Testování

V následující sekci se věnuji testování konkrétních tříd v rámci jednotlivých modulů i celkové aplikace při komunikaci s RFID čtečkou.

Při implementaci jednotlivých modulů byly použity jednotkové testy sloužící pro otestování malých části zdrojového kódů (metod). Takové testy umožní udržovat vývojáři program neustále v požadovaném stavu. Tyto testy byly použity při implementaci Modulů SCom a RFIDCom, jak je popsáno níže.

7.3.1 Modul SCom

Jako první z testů uvedu Unit testy používané v modulu SCom. Unit testy používám pro ověření správnosti implementace tříd Port a PortList. V testovací třídě jsou testovány všechny metody uvnitř příslušné třídy.

7.3.1.1 Struktura testovací třídy

Pro ukázku testovací třídy jsem zvolil test třídy Port. V následující ukázce kódu 6 jsou odstraněny těla metod pro lepší čitelnost.

Algorithm 6 Rozhraní testovací třídy pro Unit testování

```
public class PortTest {
  static Port instance;

  public PortTest();
  public static void setUpClass();
  public static void tearDownClass() ;
  public void setUp();
  public void tearDown();

  public void testSetPort();
  public void testOpen() throws Exception;
  public void testSend() throws Exception;
  public void testClose() throws Exception;
  public void testGetPort();
  public void testAddPropertyChangeListener();
  public void testRemovePropertyChangeListener();
  public void testSerialEvent();

}
```

Při spuštění testu dojde k vytvoření instance třídy Port z metody setUpClass, která je vyvolána před provedením prvního testu. Dojde k nastavení

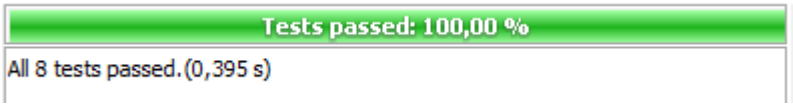
portu na COM1, který mám přiřazen z virtuálního portu vytvořeného pomocí aplikace Free Virtual Serial Ports. Zároveň je kontrolováno, zda-li došlo k správnému nastavení portu a v případě neshody označení portu dojde k vyvolání chyby.

Metody setUp a tearDown zajištují nezávislost testů, jsou totiž volány mezi jednotlivými voláními testovaných metod. Pro otestování metod, u nichž je očekávána odpověď je nutné vytvoření nové instance SCom. V mém případě se jedná o port COM2 a simulace probíhá odesláním dat na COM1 s tím, že po odeslání je poslána příslušná odpověď na port COM2.

Na následujících dvou obrázcích 7.8 a 7.9 jsou ukázky výstupu provedených jednotkových testů v prostředí NetBeans IDE.

Obrázek 7.8: Výstup jednotkové testu

Ukázka testu s chybovou hláškou vzniklou při snaze otevření již otevřeného portu ve třídě Port.



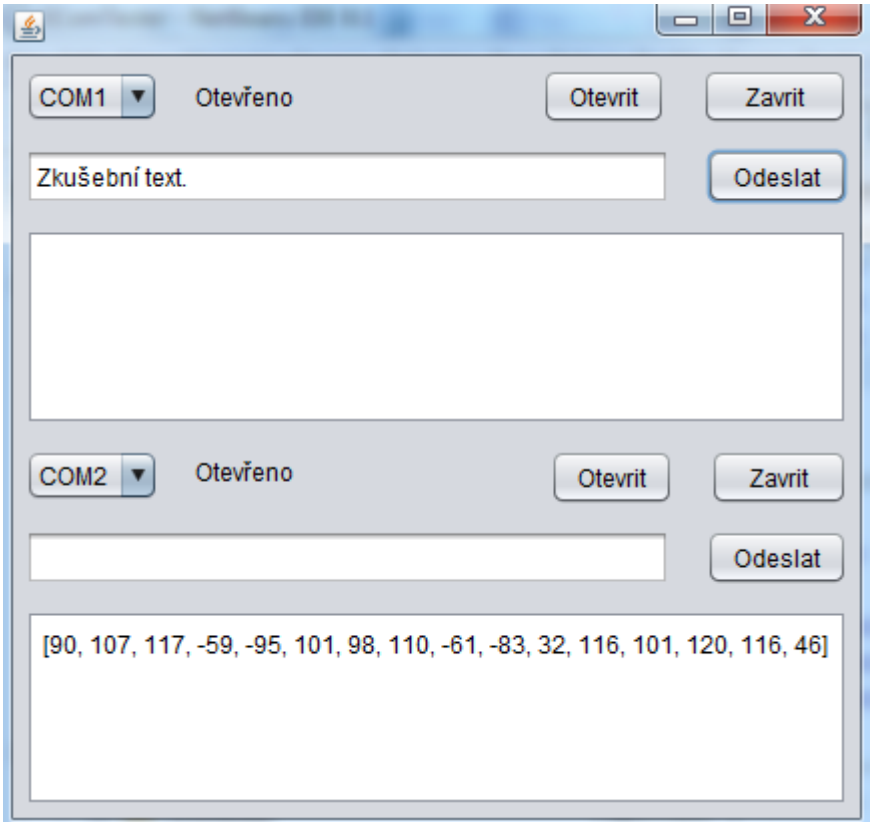
Obrázek 7.9: Výstup jednotkové testu

Ukázka úplně správného testu třídy Port.

7.3.1.2 SComTester

V rámci testování modulu SCom jsem vytvořil jednoduchou aplikací využívající právě tento modul. Ta mi spolu s vytvořenými virtuálními porty **??** umožňovala vzájemné přeposílání dat mezi jednotlivými porty jako je vidět na obrázku 7.10.

Na obrázku 7.10 je ukázka funkčnosti aplikace.



Obrázek 7.10: Ukázka běžící NetBeans aplikace RFIDReader

7.3.2 Modul RFIDCom

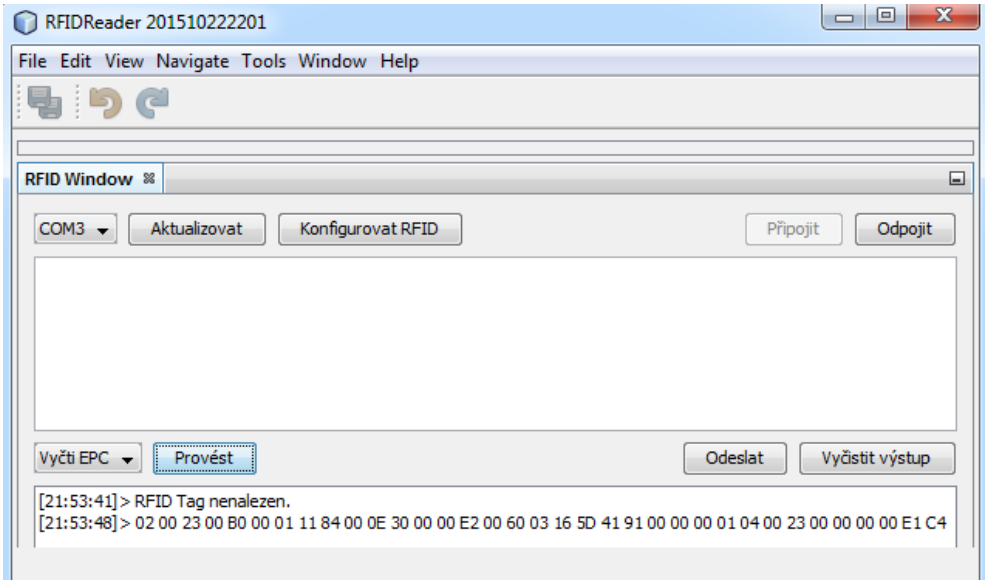
Testy v modulu RFIDCom jsou trochu více integrační než v předchozím případě. A to především díky tomu, že je zde zapotřebí využívání funkčnosti jiných modulů.

Testování je realizováno sice jednotkovými testy, ale z důvodu nutnosti využití funkcionalit modulu SCom dochází zároveň k otestování vazeb mezi jednotlivými moduly.

7.3.3 Komunikace s RFID čtečkou

Pro vyzkoušení komunikace s RFID čtečkou mi byla jedna vypůjčena společností AŽD. Jelikož si čtečka vyžaduje externí stejnosměrné napájení v rozmezí od 12V do 24V, jenž jsem neměl k dispozici. Vypomohl jsem si starým počítačovým zdrojem a redukcí z molexu na 4-pin používaný pro disketovou mechaniku.

Poté jsem mohl vyzkoušet kompletní funkčnost celého aplikačního modulu RFIDReader. Ukázka vyčtení EPC je k vidění na obrázku 7.11.



Obrázek 7.11: Vyčtení EPC

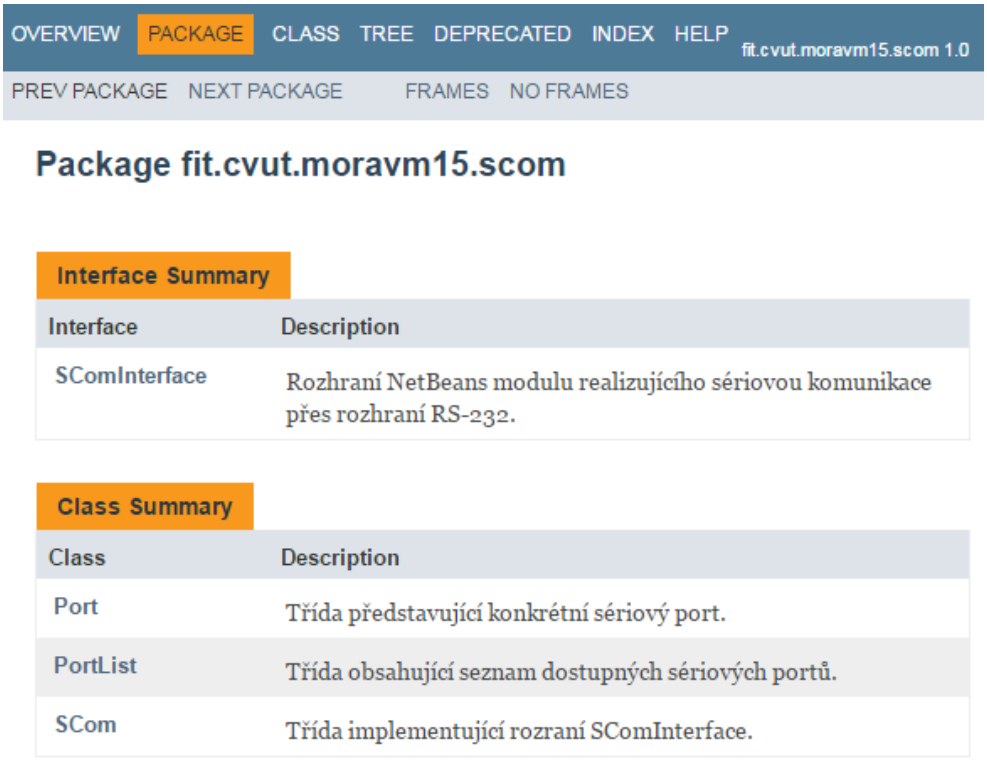
7.4 Generování Dokumentace

Pro vytvoření dokumentace jsem využil NetBeans utilitu pro generování Javadoc. Ta si pro správné interpretování vyžaduje řádné komentování jednotlivých tříd a metod. Ukázku okomentované metody můžete vidět v ukázce kódu 7 níže.

Algorithm 7 Správně okomentovaná metoda

```
/**
* This method returns name of currently selected port.
*
* @return String: Name of currently selected port
* @throws fit.cvut.moravm15.scom.exceptions.SComException
*/
String getSelectedPort() throws SComException;
```

Výstup generované dokumentace je součástí jednotlivých adresářů příslušných modulů a html stránka zobrazující dokumentaci je zahrnuta v obsahu CD. Ukázku výstupu pro balíček scom je zobrazena na obrázku 7.12 níže.



Obrázek 7.12: Javadoc

Ukázka vygenerované Javadoc dokumentace pro balíček scom.

Závěr

Hlavním cílem této práce bylo nalezení a implementování alternativního řešení pro současně používaný NetBeans modul, který využívá již nepodporovanou knihovnu RXTX. Dále pak provést průzkum použitelných knihoven umožňující přístup k sériovým portům. Posledním z cílů byla demonstrace funkčnosti modulu pomocí vybrané akce proveditelné RFID čtečkou.

Všechny tyto cíle byly postupně naplněny. V počátcích práce jsem si se zadavatelem stanovil režim, kterým bude probíhat vzájemná výměna informací o aktuálním stavu spolu s definováním používaných technologií. V první části práce jsem analyzoval současné řešení používané společností AŽD a poté vyhledal knihovny, kterými lze aktuální knihovnu nahradit. Na základě provedeného porovnání jsem vybral knihovnu jSSC, kterou jsem poté použil při implementaci. Třídní model jsem navrhl na základě poznatků, které jsem posbíral v průběhu zkoumání hierarchie použitelných knihoven. Veškeré diagramy, které byly v návrhu použity, byly vytvořeny notací UML.

V průběhu implementace jsem použil pro mě doposud nevyzkoušené verzovací technologie a programové konstrukce jako například návrhové vzory, za což jsem zpětně velmi rád. Mezi další nové zkušenosti určitě patří i realizace projektu pro reálného zadavatele, v průběhu níž probíhali osobní schůzky s řešením aktuálního stavu projektu. Je to také poprvé, co jsem v takovém množství používal programový jazyk Java.

Všechny vytvářené moduly byly v průběhu vývoje programově testovány pomocí jednotkových testů. A ihned poté, co jsem si byl s funkčností aplikace jistý, byly provedeny testy s RFID čtečkou ve výzkumném pracovišti ve Mstěticích. Po prvních úspěšných pokusech vyčtení EPC jsem se zaměřil na řádné komentování kódu a generování Javadoc, které je součástí CD přiloženého k této práci.

Do budoucna by bylo velmi příjemné vzhledem k možnosti komunikace s RFID čtečkou pomocí rozhraní USB umožnit nově vytvořenému modulu komunikovat právě i přes toto rozhraní.

Literatura

[1] Pinkas, P.: *Řízení a zabezpečení železniční dopravy [online]*. 2014. Dostupné z: http://automa.cz/index.php?id_document=52955

[2] Dvořák, P.: *Software stimulačního systému pro funkční MR zobrazování [online]*. Diplomová práce, České vysoké učení technické v Praze, Katedra dopravních systémů, 2014. Dostupné z: https://www.fel.cvut.cz/cz/education/prace/00067.pdf

[3] Böck, H.: *Platforma NetBeans: podrobný průvodce programátora*. Brno: Computer Press, 2010, ISBN 978-80-251-3116-9.

[4] Oracle: *Java Communications API [online]*. [cit. 2016-05-07]. Dostupné z: http://www.oracle.com/technetwork/java/index-jsp-141752.html

[5] Oracle: *Java Communications API [online]*. [cit. 2016-05-07]. Dostupné z: http://docs.oracle.com/cd/E17802_01/products/products/javacomm/reference/api/javax/comm/package-summary.html

[6] Oracle: *BCL For Java SE [online]*. Dostupné z: http://www.oracle.com/technetwork/java/javase/terms/license/index.html

[7] RXTX: *RXTX: The Prescription for Transmission [online]*. [cit. 2016-04-09]. Dostupné z: http://users.frii.com/jarvi/rxtx/

[8] Miroslav Lorenc: *Ganttův diagram [online]*. [cit. 2016-04-24]. Dostupné z: http://lorenc.info/3MA381/graf-ganttuv-diagram.htm

[9] Google: *Google dokumenty [online]*. Dostupné z: https://www.google.cz/intl/cs/docs/about

[10] Git: *Git [online]*. Dostupné z: https://git-scm.com/

[11] AtlassianBitbucket: *Bitbucket - the Git solution for professional teams [online]*. Dostupné z: https://www.atlassian.com/software/bitbucket

[12] RXTX: *Rxtx [online]*. [cit. 2016-05-10]. Dostupné z: http://rxtx.qbang.org/wiki/index.php/Main_Page

[13] jSSC: *Official jSSC (Java Simple Serial Connector) repository [online]*. [cit. 2016-04-23]. Dostupné z: https://github.com/scream3r/java-simple-serial-connector

[14] jSSC: *java-simple-serial-connector [online]*. [cit. 2016-04-22]. Dostupné z: https://code.google.com/archive/p/java-simple-serial-connector/

[15] Darwin, I.: *Java: kuchařka programátora*. Brno: Computer Press, 2006, ISBN 80-251-0944-5.

[16] Oracle: *Code Conventions for the Java Programming Language: 9. Naming Conventions [online]*. [cit. 2016-05-21]. Dostupné z: http://www.oracle.com/technetwork/java/codeconventions-135099.html

[17] HHD Software: *FREE Virtual Serial Ports driver, Rs-232 null modem emulator [online]*. [cit. 2016-03-02]. Dostupné z: http://freevirtualserialports.com/

[18] HARTING: *Ha-VIS RFID RF-R200 Reader [online]*. [cit. 2016-05-01]. Dostupné z: http://www.harting-rfid.com/fileadmin/harting/documents/rfid/produkte/reader/Ha-VIS_RFID_RF-R200_Reader_HARTING_GB_01.pdf

[19] HARTING: *Ha-VIS RFID RF-R200 Reader [online]*. [cit. 2016-05-01]. Dostupné z: http://www.harting-rfid.com/fileadmin/harting/documents/rfid/produkte/reader/Ha-VIS_RFID_RF-R200_Reader_HARTING_GB_01.pdf

[20] RFID-EPC: *Portál o technologii RFID–EPC a jejich aplikacích [online]*. [cit. 2016-04-26]. Dostupné z: http://www.rfid-epc.cz/

[21] RFID portál: *Co je RFID [online]*. [cit. 2016-04-13]. Dostupné z: http://www.rfidportal.cz/index.php?page=rfid_obecne

[22] ITnetwork: *Singleton (jedináček) [online]*. [cit. 2016-05-02]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/singleton-navrhovy-vzor/

[23] ITnetwork: *Observer (pozorovatel) [online]*. [cit. 2016-05-02]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/observer-pozorovatel-navrhovy-vzor/

Příloha **A**

Seznam použitých zkratek

ETCS - European Train Control System. Evropský vlakový zabezpečovací systém.

API - Application Programming Interface. Rozhraní pro programování aplikací.

jSSC - Java Simple Serial Connector.

RFID - Radio Frequency Identification. Identifikace na rádiové frekvenci.

EPC - Electronic Product Code. Jednoznačný identifikátor produktu.

Příloha **B**

Obsah přiloženého CD

- `thesis` ..... textová část bakalářské práce spolu s zadáním
  - `BP_Morávek_Marcel_2016.pdf` ..... bakalářská práce ve formátu PDF
  - `assignment.pdf` ..... zadání bakalářské práce ve formátu PDF
- `src`
  - `java` ..... zdrojové soubory jednotlivých NetBeans modulů a Javadoc
    - `scom` ..... zdrojové kódy modulu SCom
      - `src` ..... NetBeans projekt
      - `javadoc` ..... struktura vygenerovaného Javadoc
        - `index.html` ..... Javadoc HTML dokument
    - `rfidcom` ..... zdrojové kódy modulu RFIDCom
      - `src` ..... NetBeans projekt
      - `javadoc` ..... struktura vygenerovaného Javadoc
        - `index.html` ..... Javadoc HTML dokument
    - `jssc` ..... zdrojové kódy modulu JSSC
      - `src` ..... NetBeans projekt
    - `rfidreader` ..... zdrojové kódy modulu RFIDReader
      - `src` ..... NetBeans projekt
  - `latex` ..... zdrojové soubory textové části bakalářské práce