TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

OBJEKTOVĚ ORIENTOVANÝ FRAMEWORK PRO PROVOZNÍ DATA PODZEMNÍCH ZÁSOBNÍKŮ PLYNU

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jaroslav Hrabal**
Vedoucí práce: doc. Ing. Otto Severýn, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

OBJECT ORIENTED FRAMEWORK FOR UNDERGROUND GAS STORAGE OPERATIONAL DATA

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Jaroslav Hrabal**
Supervisor: doc. Ing. Otto Severýn, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jaroslav Hrabal**
Osobní číslo:	**M12000134**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Objektově orientovaný framework pro provozní data podzemních zásobníků plynu**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s problematikou podzemního skladování plynu obecně a dále s metodikou sběru a archivace provozních dat PZP na pracovišti Geo-services firmy RWE Gas Storage s.r.o.
2. Navrhněte a v jazyce Java implementujte softwarové rozhraní, které umožní přístup k takto uloženým datům.
3. Rozhraní bude respektovat hierarchii zásobník - skupina sond - sonda - veličina. Dále umožní základní manipulace s daty - agregace v čase a ve skupině, základní čištění dat, interpolace, výpočet odvozených veličin (kumulativních objemů, statických tlaků atp.) a export dat do souborů v obecně použitelných formátech.
4. Pomocí rozhraní vytvořeného v bodě 2 navrhněte a implementujte aplikaci pro zobrazování okamžitého stavu PZP.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Eckel, B.: Thinking in Java, 4th edition, Prentice Hall, 2006.**

[2] **Dake, L.: Fundamentals of reservoir engineering. Elsevier, 1978.**

Vedoucí bakalářské práce: **doc. Ing. Otto Severýn, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci
L.S.
- 14 -

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16. 5. 2016

Podpis: Jirabal

Poděkování:

Rád bych na tomto místě chtěl poděkovat všem, s jejichž pomocí tato práce vznikla. Zvláště chci poděkovat svému vedoucímu práce, jímž je doc. Ing. Otto Severýn, Ph.D. za jeho odborné vedení bakalářské práce a pomoc, kterou mi při práci poskytl.

Anotace

Bakalářská práce se zabývá objektově orientovaným programováním. Zaměřuje se na jeho užívání a na návrh frameworku schopného ukládání a manipulace s daty získanými při měření provozních hodnot na podzemním zásobníku plynu. Framework byl realizován použitím jazyku Java. v práci jsou popsány metody pro načítání datových souborů, ukládání dat do paměti, matematické operace s daty, výběr dat na základě stanovených parametrů a metody pro zobrazení dat. Pomocí navrženého frameworku je sestaveno grafické rozhraní umožňující uživatelsky jednodušší práci s daty.

Klíčová slova

Java, podzemní zásobník plynu, objektově orientované programování, framework, grafické rozhraní.

Annotation

This bachelor’s thesis deals with object-oriented programming. It focuses on using object-oriented programming to develop a framework capable of storing and manipulating data obtained by measuring underground gas storage. The framework was built using Java. The work describes methods to read data files, store data, work data using mathematical functions, select data and to view data. The framework is used to make a graphical user interface for easy data manipulation.

Key Words

Java, underground gas storage, object-oriented programming, framework, graphical user interface.

Obsah

1 Úvod ... 12
2 Základní principy ukládání plynů do podzemních zásobníků ... 14
2.1 Vlastnosti plynů ... 14
2.2 Zdroje zemního plynu ... 14
2.3 Oxid uhličitý ... 15
2.4 Základní možnosti ukládání plynů ... 16
2.5 Základní parametry zásobníků plynu v přírodních strukturách ... 18
2.6 Základní parametry úložišť plynů v umělých strukturách ... 19
2.7 Technické parametry úložišť plynů ... 19
2.8 Příklady zásobníků provozovaných v České Republice ... 20
2.8.1 Zásobník Dolní Dunajovice ... 23
3 Měření dat na zásobníku a datové operace ... 24
3.1 Měření tlaku a teploty ... 24
3.2 Měření průtoku ... 24
3.3 Datové operace ... 24
3.3.1 Agregační funkce ... 24
3.3.2 Čištění dat ... 24
3.3.3 Interpolace ... 25
4 Obecné datové formáty ... 26
4.1 JSON 26
5 Technologie použité při vývoji ... 28
5.1 Java Collection Framework ... 28
5.2 Rozhraní List<E> ... 28
5.3 Rozhraní SortedMap<K,V> ... 28
5.4 Aplikační programovací rozhraní Reflection ... 29
5.5 The Apache Common Mathematics ... 29
5.6 JSON-simple ... 29
5.7 JUnit 29
6 Struktura a popis provozních dat ... 30
6.1 Struktura TBD souborů ... 30
6.1.1 Skupinový soubor ... 30
6.1.2 Hlavičkový a datový soubor ... 30

7 Specifikace požadavků ... 32
7.1 Uložení dat ... 32
7.2 Umožnění přístupu k uloženým datům ... 32
7.3 Dodržení hierarchie zásobník – skupina sond – sonda – veličina ... 32
7.4 Umožnění základních manipulací s daty ... 33
8 Konfigurační soubory ... 34
8.1.1 Hlavní konfigurační soubor ... 34
8.1.2 Konfigurační soubor veličiny ... 34
9 Design Frameworku ... 36
9.1 MeasurePoints ... 36
9.2 Readers ... 38
9.3 Functions ... 38
9.4 Writers ... 40
9.5 Tests ... 40
10 Podrobný popis metod ... 42
10.1 Třídy a metody balíčku meassurePoints ... 42
10.1.1 AbstractGroup ... 42
10.1.2 Group ... 43
10.1.3 Well ... 43
10.1.4 AbstractPoint ... 44
10.2 Balíček readers ... 45
10.2.1 Metody tříd ReaderTBD a ReaderJSON ... 46
10.3 Balíček functions ... 47
10.3.1 Balíček aggregate ... 47
10.3.2 Balíček cumAggregate ... 47
10.3.3 Balíček dataFilter ... 47
10.3.4 Balíček interpolate ... 48
11 Použití frameworku ... 49
11.1 Instalace frameworku ... 49
11.2 Vytvoření objektové struktury ze souboru ... 49
11.3 Načtení dat ze souboru ... 50
11.4 Využití JSON souborů ... 50
11.5 Operace s daty a zobrazení výsledků ... 50
11.5.1 Operace s instancemi balíčku measurePoint ... 51

12 Rozšířitelnost frameworku 53
12.1 Přidání nových operací s daty 53
12.2 Přidání nových metod pro čtení odlišných souborů 53
12.3 Přidání nových tříd pro objekty představující oblasti měření 53
13 Grafické rozhraní 55
13.1 Vytvoření objektové struktury ze souboru 55
13.2 Načtení dat ze souboru 56
13.3 Využití JSON souborů 56
13.4 Operace s daty a zobrazení výsledků 57
13.5 Čištění dat 59
14 Závěr 60
15 Seznam použité literatury 61

Seznam obrázků, grafů a tabulek:

Obrázek 1: Schéma ložiska plynu strukturního typu 16

Obrázek 2: Schéma zásobníku plynu aquiferového typu. 17

Obrázek 3: Umístění podzemních zásobníků v ČR. [2] 22

Obrázek 4: Letecká fotografie zásobníku Dolní Dunajovice [4] 23

Obrázek 5: Struktura jazyka `JSON` [5] 27

Obrázek 6: Skupinový TBD soubor. 30

Obrázek 7: Hlavičkový a datový TBD soubor. 31

Obrázek 8: Hierarchie objektů a jejich třídy. 33

Obrázek 9: Hlavní konfigurační JSON soubor. 34

Obrázek 10: Konfigurační JSON soubor veličiny obsahující pouze povinné atributy ... 35

Obrázek 11: Konfigurační JSON soubor veličiny obsahující všechny atributy 35

Obrázek 12: Část balíčku představující skupinové objekty. 37

Obrázek 13: Rozdělení balíčku functions. 39

Obrázek 14: Schéma balíčku functions. 40

Obrázek 15: Balíčky frameworku. 41

Obrázek 16: Hlavní menu. 55

Obrázek 17: Vytvoření zásobníku z TBD souboru. 55

Obrázek 18: Přidání dat do zásobníku. 56

Obrázek 19: Vytvoření zásobníku a načtení dat z JSON souboru. 57

Obrázek 19: Výpočet kumulativních funkcí. 58

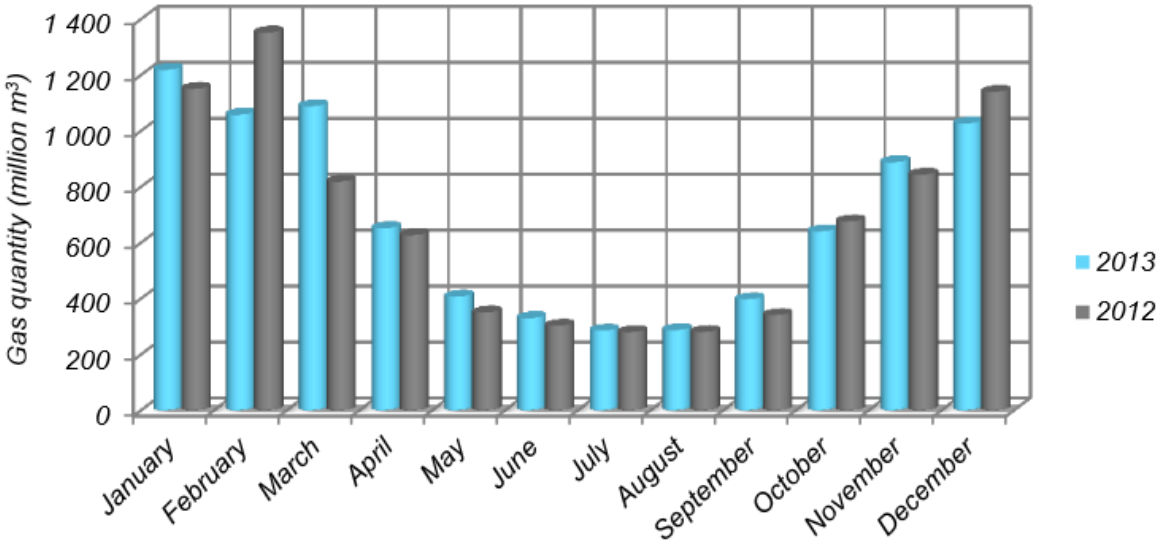
Obrázek 20: Filtrování dat. 59

Graf 1: Spotřeba plynu v letech 2012-2013 [1]……...……12

Tabulka 1: Základní parametry podzemních zásobníků v ČR [2]..... 22

1 Úvod

Technologie skladování plynu se rozvíjí v souvislosti s využíváním plynů v průmyslu i komunální sféře a s požadavky na zvětšování skladovací kapacity pro vykrývání sezónních i denních špiček. Zásobování plynem nelze realizovat metodou „just in time“, jak je to běžné u jiných komodit. Rozvoj využití plynných paliv jako ekologicky nezávadného zdroje energie má v posledním desetiletí značně stoupající tendenci ve všech průmyslově vyspělých státech.



Graf 1: Spotřeba plynu v letech 2012-2013 v ČR[1]

Distribuce plynných paliv s nepravidelnou roční odběrovou křivkou a s výraznou kulminací spotřeby v zimních měsících, graf 1, se stala jedním z nejzávažnějších problémů, na jehož vyřešení závisí provoz podniků, služeb a domácností. Zemní plyn se tak stává sezónní surovinou a tím i možným nástrojem politicko-ekonomického nátlaku. Důležitou skutečností je i lokalizace světových zásob plynu, přičemž nejvýznamnější současné zdroje zemního plynu se nacházejí v politicky nestabilních oblastech. v České republice se nacházejí pouze malá ložiska zemního plynu, která zdaleka nemohou pokrýt požadovanou potřebu (pouze 1%). Nezbytou nutností je tak doprava plynu a jeho skladování, respektive vytváření sezónních zásob, ze kterých je možno krýt špičkové odběry, popřípadě výpadky v dodávkách od producentů, ať již jsou způsobeny jakýmikoliv důvody. k zajištění dostatečného

množství plynu v období sezónních nebo denních špičkových odběrů se jeví jako nejvhodnější metoda jeho podzemního skladování.

Provoz podzemních zásobníků předpokládá vysoce sofistikovaný způsob budování technologického celku a manipulace s médiem. Při těchto činnostech je zcela nezbytné využívat metody numerického modelování ať již v rámci budování zásobníku, ale především při jeho provozu.

2 Základní principy ukládání plynů do podzemních zásobníků

2.1 Vlastnosti plynů

Základní vlastností využívanou pro skladování plynu je jejich stlačitelnost. Ideální plyn je dokonale stlačitelný bez vnitřního tření a je využíván pro popis mechanických a termodynamických vlastností plynů. Reálné plyny se svými vlastnostmi mírně liší, přičemž ideálním plynům se přibližují při dostatečně vysoké teplotě a nízkých tlacích. Pro termodynamické děje v plynech platí stavová rovnice ideálního plynu:

$$pV = NkT \qquad (1)$$

kde p je tlak plynu, v je objem, N celkový počet částic plynu, T termodynamická teplota a k Boltzmannova konstanta (vyjadřuje množství energie potřebné k zahřátí jedné částice ideálního plynu o jeden kelvin). Při změnách objemu plynu tak dochází k změnám jeho teploty, které lze charakterizovat jako polytropický děj (krajními případy jsou adiabaticky a izotermický děj), jelikož u reálných podmínek nelze systém izolovat od okolí. Obecně při expanzi plynu teplota klesá a při jeho stlačování stoupá. Tomu musí být přizpůsobena technologie skladování a musí být prováděno chlazení a dohřev plynu.

Zemní plyn těžený na ložiscích je směs plynů s převažujícím metanem (CH_4), příměsí dalších plynných uhlovodíků (ethan – C_2H_6, propan – C_3H_8, butan – C_4H_{10}), oxidu uhličitého, dusíku, sirovodíku a stopami vzácných plynů a kyslíku. Důležitou vlastností zemního plynu vedle jeho hořlavosti je i výbušnost při smísení s kyslíkem. Na některých ložiscích může podíl nehořlavých plynů dosahovat i větších hodnot a plyn je tak nutno před jeho použitím nebo transportem upravovat.

2.2 Zdroje zemního plynu

Zemní plyn je v přírodě produkován několika hlavními způsoby. Na povrchu nebo mělce pod povrchem vzniká bakteriálním rozkladem organické hmoty (bioplyn, skládkový plyn a podobně), přičemž obsah metanu závisí na vlhkosti a přístupu kyslíku.

Při vyšší vlhkosti a omezeném kontaktu s atmosférou dochází k metanogenezi (produkci metanu), za opačných podmínek k acidogenezi, kdy je organická hmota rozkládána na oxid uhličitý a dusík. v hlubších zónách zemské kůry dochází k termogenické přeměně pohřbené organické hmoty. v závislosti na teplotě prostředí vzniká ropa (při nižších teplotách) nebo zemní plyn (při vyšších teplotách), který pak migruje horninovým prostředím. Metan je také poměrně běžnou součásti ložisek uhlí, kde vzniká při karbonizaci rostlinných zbytků a je adsorbován na uhelné sloje. Má velmi nízkou příměs dalších plynu a komplikuje těžbu z důvodu jeho výbušné směsi s kyslíkem. Posledním typem vzniku metanu je jeho anorganická syntéza během tuhnutí magmatu, tento typ metanu nemá průmyslový význam.

2.3 Oxid uhličitý

Ukládání oxidu uhličitého do podzemních zásobníků je jednou z možných cest snižování jeho emisí do ovzduší a tím i předcházení dopadu činností člověka na globální oteplování planety Země. Má však také řadu odpůrců poukazující na agresivitu tohoto plynu a dlouhodobou neudržitelnost těsnosti zásobníků. v současné době již k ukládání oxidu uhličitého do podzemních struktur dochází, avšak nikoliv z důvodu zabránění jeho emisí do ovzduší, ale z důvodů snížení nákladů na dopravu zemního plynu a zlepšení výtěžnosti ložisek. Jedná se o ložiska těžená z ropných plošin v Norsku, s vyšším obsahem oxidu uhličitého. Zemní plyn je dopravován ve stlačeném stavu speciálními tankery. Vyčištěním zemního plynu od nespalitelných příměsí se snižuje objem přepravované suroviny a tak je ekonomicky výhodnější provádět čištění již na místě těžby. Vhodně uložené odpadní plyny pak také umožňují zvýšit efektivitu těžby tím, že vytlačují surovinu z okraje ložiska do centrální těžené části.

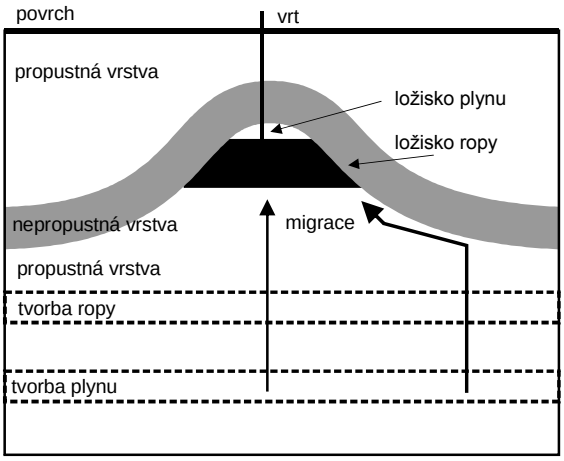
Využití ukládání oxidu uhličitého v podmínkách České republiky je však vysoce nepravděpodobné. z hlediska nakládání s tímto plynem by se však jednalo o obdobný technologický postup jako v případě zemního plynu (bez nutnosti instalovat technologii zpětné těžby), přičemž oxid uhličitý je nevýbušný.

2.4 Základní možnosti ukládání plynů

Plyny mohou být ukládány do vybudovaných technologických zařízení na povrchu - plynojemů. Jedná se většinou o tlakovou nádrž a technologii na stlačování plynu. Toto ukládání má své výhody a nevýhody. Především výstavba těchto úložných kapacit je drahá a neopomenutelné je i bezpečnostní riziko havárií, popřípadě i cíle teroristických útoků. Tyto kapacity nejsou reálně využitelné pro ukládání sezónních rezerv zemního plynu, a jsou používány jako pohotovostní zásoba plynu.

Pro uložení velkých objemů zemního plynu, popřípadě i oxidu uhličitého, jsou využívány vhodné struktury povrchové části zemské kůry. Jako technicky nejjednodušší lze takto upravit vytěžená ložiska zemního plynu nebo ropy. Tato ložiska vznikla tak, že zemní plyn (ropa) migrovala z místa svého vzniku horninovým prostředím a byla zachycena ve vhodné struktuře. Takováto struktura se nazývá ropná nebo plynová past. Při tomto procesu jsou využity základní fyzikální vlastnosti horninového prostředí i migrujícího média.

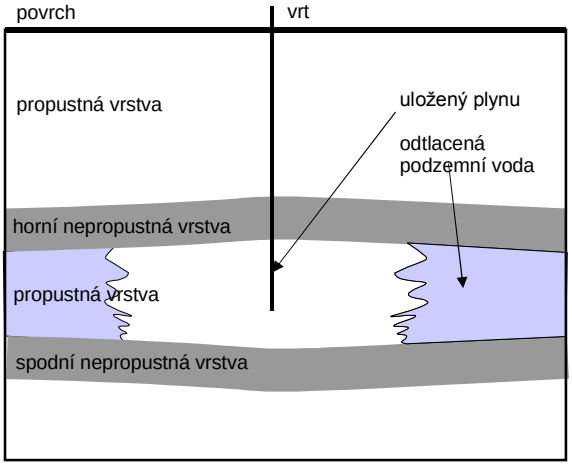
Plyn nebo kapalina mohou migrovat pouze propustným horninovým prostředím. Propustnost je určována porozitou (existencí propojených pórů mezi zrny horninové matrice) nebo puklinatostí (systémem propojených puklin horniny). Vyšší propustnost vykazují nezpevněné sedimenty (písky štěrky) a pískovce. Sedimenty s vysokým obsahem jílu neobsahují spojité póry ani pukliny a jsou pro plyn a kapaliny (voda, ropa) nepropustné. Plyn je litostatickým tlakem (tlakem horninových vrstev) vytlačován k zemskému povrchu. Ropa má nižší hustotu než voda a tak stoupá k zemskému povrchu. Pokud narazí na nepropustnou vrstvu je postup zastaven a dochází k její akumulaci. Vrstvy hornin jsou zvrásněny a vytváří antiklinály (místa vyklenutí) a synklinály (místa poklesnutí). Uhlovodíky tak migruji směrem k povrchu a jsou koncentrovány pod vrcholem antiklinály. Schéma tohoto procesu je uvedeno na obrázku 1. Vytěžením ložiska, tedy odčerpáním ropy a zemního



Obrázek 1: Schéma ložiska plynu strukturního typu

plynu se na uvolněné místo dostane podzemní voda. Ta je většinou nevhodná pro využití, protože obsahuje soli a zbytky ropných uhlovodíků. Takové místo pak může být využito pro znovuukládání zemního plynu, pokud není těžbou porušena nepropustná vrstva. Případně netěsnosti způsobené vrty lze technicky zajistit.

Podobné geologické struktury lze využít i v případech, že neobsahovaly ropu nebo zemní plyn. Nezbytná je však přítomnost nepropustné vrstvy a struktury antiklinály nebo propustné vrstvy uzavřené dvěma nepropustným vrstvami, kde se vtláčený plyn zachytí a nemůže unikat mimo toto kontrolované pásmo. Takovéto struktury bývají obvykle vyplněny podzemní vodou. Pokud je struktura hlouběji uložena, pak je podzemní voda stlačena litostatickým a hydrostatickým tlakem, tj. tlakem nadložních hornin a sloupce vody. Po provrtání izolační vrstvy může být voda vytlačena až na zemský povrch. Tento parametr je důležitý z důvodu možného stlačení plynu. Voda ve struktuře způsobuje protitlak. Plyn tak musí svým tlakem vytěsnit vodu z pórů horniny a tím je uvolnit pro ukládání.



Obrázek 2: Schéma zásobníku plynu aquiferového typu.

Pro ukládání plynu lze také využít přírodní struktury, které neobsahují podzemní vodu. Jedná se o masivní horniny nebo prostory vytěžené z jiného důvodu. Typickým příkladem jsou uhelné, solné nebo rudné doly. Sůl a uhlí tvoří v zemské kůře sloje a sůl také tlakem hornin může být vytlačena a vytvářet deformační pně. Po těžbě soli zůstávají uvolněné rozsáhlé prostory, přitom okolní horniny jsou většinou nepropustné a bez přístupu vody. Pokud se takto vytěžené ložisko utěsní, tedy se odstraní veškeré netěsnosti vytvořené člověkem, vznikne dutina ve skalním masivu (kaverna), kterou lze vyžít pro ukládání plynu. Obdobná situace může nastat u uhelných a u rudných dolů, kde se těžily kovy pro průmyslové využití.

Poslední možností je vybudovat podzemní zásobník v kompaktním skalním masívu, třeba v žule, který neobsahuje netěsnosti (pukliny). Je však nutno vytěžit

horninu a vytvořit tak umělou dutinu ve skalním masívu. Je zřejmé, že těženou horninu nelze ekonomicky výhodně uplatnit na trhu a náklady na tento typ zásobníku budou vysoké. Výhodou je však zpřístupnění prostoru a důkladní utěsnění zásobníku, což u přírodních struktur je vyloučeno a u prostor po odtěžených surovinách bývá poněkud složitější. Manipulace s plynem v takovýchto úložištích je také technicky jednodušší.

2.5 Základní parametry zásobníků plynu v přírodních strukturách

U přírodních struktur v sedimentárních horninách se jako úložný prostor využívá pórovitost popřípadě puklinatost hornin, tedy drobné volné prostory mezi zrny sedimentu nebo puklinami pevnějších hornin. Tyto prostory jsou běžně vyplněny vodou. Stanovovány jsou tak základní parametry zásobníku:

Kolektorská hornina – je horninová struktura schopná pojmout vtláčený plyn. Jedná se o propustné horniny, které plyn akumulují ve svých pórech. Její vlastnosti jsou charakterizovány především propustností a pórovitostí.

Izolátor – je horninová struktura tvořená nepropustnou horninou, která tvoří přírodní fyzikální barieru zabraňující úniku vtláčeného plynu. Její vlastnosti jsou charakterizovány propustností.

Porozita – je objem póru v hornině. Udávána je v % a představuje teoretický prostor přístupný pro vtláčení plynu. Porozita se udává jako celková (objem všech póru) nebo efektivní (objem pórů vzájemně propojených). Některé póry totiž mohou být uzavřeny mezi zrny minerálů a nemusí komunikovat s ostatními póry. Velikost této veličiny závisí na typu horniny. Nejvyšší efektivní porozitu vykazují štěrkopísky a nejnižší jíly. u hornin hlouběji uložených pak jistou roli hraje i litostatický tlak, který způsobuje kompakci sedimentů, tedy snižování porozity.

Propustnost – je schopnost horniny propouštět kapalinu nebo plyn. Je funkcí rychlosti šíření kapaliny vlivem hydraulického gradientu (spádu hladiny) nebo hydrostatickho tlaku či viskozity kapaliny. u plynů je funkcí rychlosti šíření vlivem tlaku. Udává se v jednotkách $m.s^{-1}$.

Retenční schopnost – je schopnost horniny pojmout určité množství kapaliny nebo plynu. Udává je v jednotkách kg/m^3. u kapalin je v podstatě funkcí efektivní pórovitosti, jelikož kapaliny jsou nestlačitelné. u plynů je situace poněkud

složitější a retenční schopnost je vedle pórovitosti horniny určována tlakem plynu. v přírodních strukturách nelze tlak plynu zvyšovat nad určitou mez stanovenou podmínkami lokality.

Celková kapacita zásobníku – je množství plynu, jež je možnou do zásobníku uložit. Udává se ve standardních kubických metrech Sm^3 (m^3 za teploty 15 °C, a absolutního tlaku: 1.01325 barA), u přírodních zásobníků proti sobě působí dva základní parametry a to hydrostatický tlak podzemní vody přítomné v horninové struktuře a tlak vtláčeného plynu. Se zvyšováním hloubky zásobníku roste i hydrostatický tlak vody a tím roste i možný tlak uskladněného plynu. Zároveň je vytlačována podzemní voda a rozšiřován úložný prostor zásobníku. Úložný prostor nelze zvyšovat nad hraniční geometrii struktury, jelikož by docházelo k úniku plynu mimo tuto strukturu.

2.6 Základní parametry úložišť plynů v umělých strukturách

V umělých strukturách je plyn ukládán do kaveren v horninových masívech, ať už byly vyhloubeny účelově pro těžbu surovin nebo jako speciální zásobník plynu. Jediným parametrem vztaženým na přírodní podmínky je celková kapacita zásobníku. Ta je definována dostupným úložným objemem a maximálním pracovním tlakem. v zásobnících se také musí dodržovat rozsah daný úřady, kde každý zásobník má danou maximální a minimální velikost pracovních tlaků.

2.7 Technické parametry úložišť plynů

Těžební výkon - je nejčastěji vyjádřen jako míra množství plynu, které může být denně vytěženo ze zásobníku. Většinou se měří v Sm^3/den. Těžební výkon podzemního zásobníku je proměnná závislá na mnoha faktorech (množství plynu v zásobníku v určitou dobu, tlak uvnitř zásobníku, schopnost komprese zásobníku, vlastnosti povrchních zařízení spojených se zásobníkem a dalších faktorech). Těžební výkon je závislý na celkovém objemu plynu v zásobníku. Je nejvyšší v okamžiku, kdy je naplněna kapacita zásobníku a postupně

s vyčerpáním zásob klesá, jelikož klesá i tlak v zásobníku. Platí tedy přímá úměra mezi těžebním výkonem a zásobou plynu.

Vtláčený výkon - je podobně jako těžební výkon míra množství plynu, které může být denně vtlačeno do zásobníku. Obvykle se udává v jednotkách Sm^3/den. Na rozdíl od těžebního výkonu je vtláčený výkon nejnižší, když je zásobník plný. Platí nepřímá úměra mezi vtláčením výkonem a zásobou plynu.

Poduška – je minimální zásoba plynu v zásobníku, která je nezbytně nutná pro provoz technologie. Jedná se o zásobu plynu, kterou nelze využít pro zásobování.

Provozní kapacita zásobníku – je množství plynu, se kterým lze manipulovat, tedy využít pro ukládání a těžbu.

Aktivní skladovací kapacita - charakterizuje objem plynu, který může být odtěžen, aniž by zásobník byl ohrožen z hlediska vtláčení a zabezpečení požadovaných nároků na příští těžební sezónu

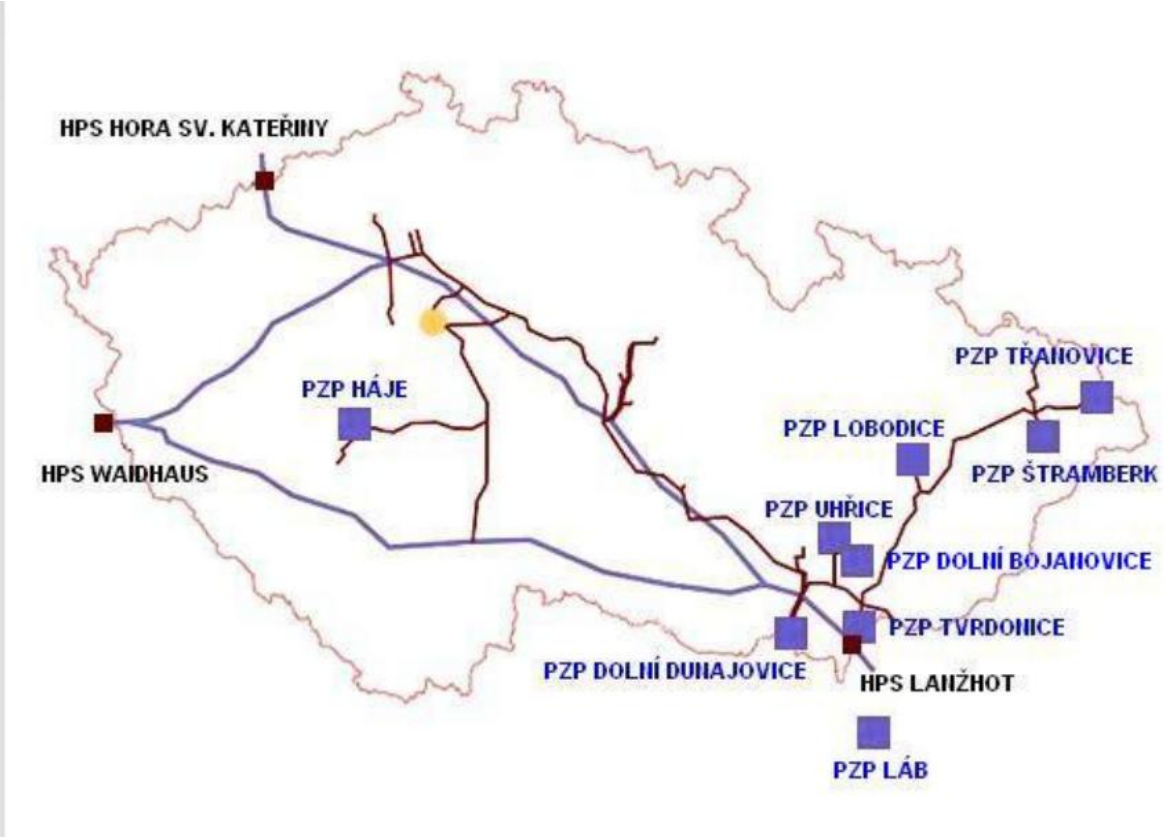
Špičkový výkon - je maximálním výkonem podzemního zásobníku, který může být realizován na základě daných provozních podmínek. Vyrovnávání odběrových špiček je technicky výhodnější z kavernových zásobníků.

2.8 Příklady zásobníků provozovaných v České Republice

Podle informací převzatých ze serveru www.mojeenergie.cz byl první tuzemský podzemní zásobník plynu uveden do provozu v Lobodicích v roce 1965. v té době byl využíván pro skladování svítiplynu. Ve zmodernizované podobě funguje dodnes. v současnosti celková tuzemská kapacita podzemních zásobníků plynu dosahuje hodnoty 3,072 mld.m^3. Celkový maximální denní teoretický těžební výkon všech tuzemských zásobníků tedy činí 58,7 mil. Sm^3 zemního plynu**.** Uvedené hodnoty zajišťují České republice v poměru k celkové tuzemské spotřebě zemního plynu (uskladňovací kapacita se pohybuje na úrovni 35,43 % celkové roční spotřeby) jedno z čelných míst mezi státy EU.

Nejvíce podzemních zásobníků plynu provozuje společnost RWE Gas Storage. Šest jejích zásobníků disponuje úhrnnou kapacitou 2,321 mld. Sm^3 při maximálním denním těžebním výkonu 43,7 mil. Sm^3 . Jedná se o podzemní zásobníky plynu

Lobodice, Tvrdonice, Štramberk, Dunajovice, Háje a Třanovice. Další podzemní zásobník plynu v Uhřicích vlastní a provozuje společnost Moravské naftové doly a posledním zásobníkem provozovaným v tuzemsku je zásobník v katastru obce Dolní Bojanovice, vlastněný společností SPP Bohemia. Tento zásobník je však zatím na základě smluv využíván pouze pro Slovensko, s jehož plynárenskou soustavou je přímo propojen. Rozmístění podzemních zásobníků plynu v České republice a jejich parametry je uvedeno na obrázku 3 a v tabulce 1.



Obrázek 3: Umístění podzemních zásobníků v ČR. [2]

	Lobodice	Tvrdonice	Štramberk	Dunajovice	Háje	Třanovice	Uhřice	Dolní Bojanovice
vlastník	RWE Gas Storage	RWE Gas Storage	RWE Gas Storage	RWE Gas Storage	RWE Gas Storage	RWE Gas Storage	MND	SPP Bohemia
typ	aquifer	plynové ložisko	plynové ložisko	plynové ložisko	kaverna	plynové ložisko	plynové ložisko	plynové ložisko
rok uvedení do provozu	1965/91	1975	1983	1989	1998	2001	2001	1999
provozní zásoba určená pro obchod s plynem (mil.m^3)	177	460	480	900	59	240	180	576
maximální denní těžební výkon (mil.m^3)	3,6	7,0	7,0	16,0	6,0	4,1	6,0	9,0
hloubka obzoru (m)	440	1260	575	1048	950	440	1750	1660

Tabulka 1: Základní parametry podzemních zásobníků v ČR [2]

2.8.1 Zásobník Dolní Dunajovice

Pro zpracování bakalářské práce byla poskytnuta provozní data zásobníku Dolní Dunajovice. Jedná se o zásobník, který je vybudován na plynonosné struktuře na Jižní Moravě. Strop kolektorských hornin byl naražen v hloubce 1050 m pod povrchem. Po odtěžení části zásob zemního plynu bylo rozhodnuto o využití geologické struktury pro ukládání zásob plynu a byl tak vybudován největší zásobník plynu v České republice, který byl uveden do provozu již v roce 1989. Maximální denní těžební výkon činí 16 mil m^3, což je cca 1,77% provozní zásoby. Těžební výkon je právě určován hloubkou uložení plynonosné struktury.

Obrázek 4: Letecká fotografie zásobníku Dolní Dunajovice [4]

Bylo řečeno [3], že drenážní horninu plynového ložiska tvoří glaukonitické pískovce až prachovce, uložené místy až v sedmi hydrodynamický spojených vrstvách částečně oddělených nepravidelnými proplástky jílovců, takže tvoří jeden hydrodynamický celek. Vrchol struktury tvoří zlom o výšce skoku až 150 m v úrovni druhohorních vápenců. Nadloží plynonosného horizontu tvoří nepropustné vápnité jílovce.

Zásobník je vybaven technologií na vtláčení plynu, která obsahuje vstupní filtry pro čištění plynu, měření množství plynu před vtláčením do zásobníku, čtyři kompresory, chladiče plynu, odlučovače oleje pro odloučení oleje z plynu, sběrné středisko s měřícími a regulačními tratěmi sond, provozní sondy, plynovody s propojovacími kolektory. Technologie pro těžbu se skládá z provozních sond a zařízení pro nástřik metanolu, sběrného střediska s měřícími a regulačními tratěmi sond, zařízení pro separaci (odvodnění ložiskové vody) a ohřev plynu (zvýšení teploty plynu před snížením jeho tlaku), sušení plynu (odstraněním vodních par z plynu) a regenerace glykolu, výstupních filtrů pro čištění plynu, měření množství plynu před jeho expedicí do přepravní soustavy a plynovodů s propojovacími kolektory.

3 Měření dat na zásobníku a datové operace

Na zásobníku je prováděno provozní měření tlaku, teploty a průtoku plynu. Bakalářská práce se zabývá zpracováním dat získaných těmito měřeními. Měření probíhá na určitých částech plynovodu, kterým se říká kontrolní měřící body. Pro ovládání kontrolních měřících bodů se používají telemetrické stanice propojené do dispečerského centra na počítače SCADA systému. Zde je následně sledován stav přepravy plynu a dispečer je schopen uzavřít nebo otevřít trasový uzávěr, který se nachází u konkrétního kontrolního měřícího bodu.

3.1 Měření tlaku a teploty

Měření tlaku probíhá v rozmezí 10-200 Bar. Naměřený tlak je následně ukládán v těchto jednotkách, stejně jako teplota, která je měřena ve stupních Celsia. Hodnoty tlaku a teploty jsou pak použity s daty získanými při měření průtoku.

3.2 Měření průtoku

Množství plynu při průtoku se mění v závislosti na tlaku a na teplotě. Pro vyjádření množství plynu používáme jednotku látkového množství Sm^3. Pro měření množství plynu existují přepočítávače množství, které měří objem, tlak i teplotu. Průtok je měřen na jednotlivých uzávěrech. Plynovody se pak sdružují dohromady a tlak je následně znovu měřen. Data se měsíčně přepočítají a opravují, aby sumy průtoku na uzávěrech odpovídaly celkovému průtoku.

3.3 Datové operace

S naměřenými daty je potřeba provádět různé operace. Jedná se o agregace, čištění dat, interpolace a výpočet odvozených veličin.

3.3.1 Agregační funkce

Agregační funkce jsou statistické funkce, pomocí kterých je možné seskupit výsledky určité aritmetické nebo statistické funkce. Mezi agregační funkce potřebné pro práci s veličinami patří součet a aritmetický průměr. Průměr je nutný počítat zejména u teplot, zatímco součty se používají u průtoků.

3.3.2 Čištění dat

Při měření a zaznamenávání dat může dojít k chybě v měření. Je potřebné tyto chyby najít a patřičně s nimi naložit. Chyby se hledají na základě následujících principů:

- Čištění podle zadaného rozsahu – všechny hodnoty, které nespadají do zadaného rozsahu hodnot jsou chybné.
- Čištění `peak` – při měření se stává, že hodnota určitého měření se výrazně vychyluje od hodnot jí sousedících. Takováto hodnota je pak označena jako chybná.
- Čištění konstantního tlaku – pokud tlak na určitém uzávěru zůstává stejný, ale průtok není nulový, jedná se o chybu.

Po nalezení chybných dat můžeme měření odhadnout pomocí interpolačních metod.

3.3.3 Interpolace

Interpolační funkce slouží k přibližnému nalezení hodnoty funkce v intervalu, pokud je hodnota známa v odlišných bodech tohoto intervalu.

První a nejjednodušší metodou je Lineární interpolace. Jedná se o proložení sousedících bodů přímkou. Následující vzorec popisuje lineární interpolaci, kde souřadnice x_1,x_3,y_1,y_3 patří k sousedícím bodům a hledaný bod má souřadnice x_2,y_2.

$$y_2 = \frac{(x_2 - x_1)(y_3 - y_1)}{(x_3 - x_1)} + y_1 \tag{2}$$

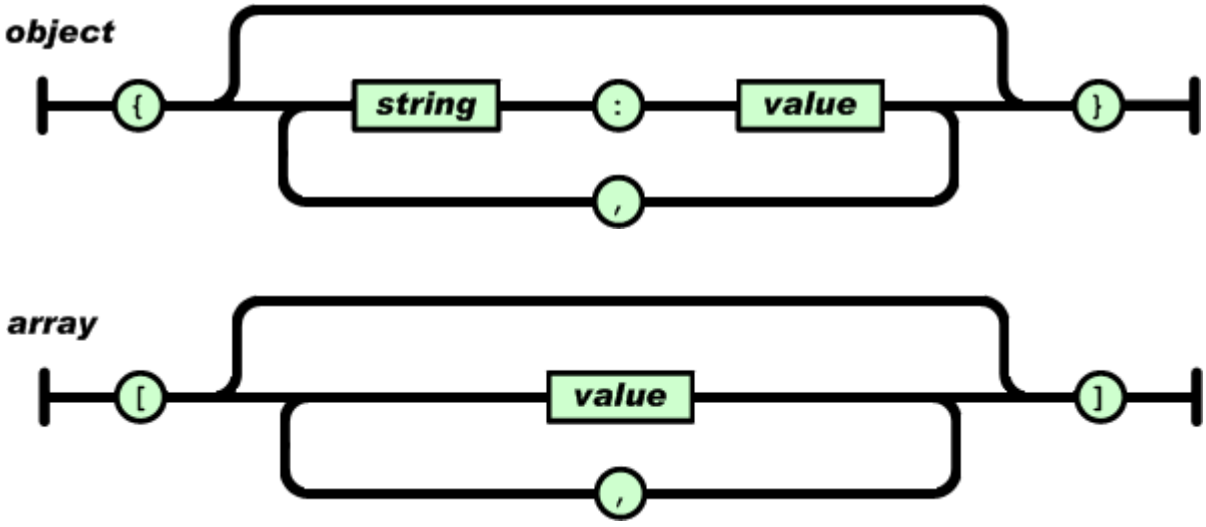
Další metodou použitou v této práci je interpolace `spline`. Jedná se o přirozenou kubickou křivku. Bodům, které známe, říkáme uzly. k definici křivky musíme mít alespoň tři uzly, ale větší počet uzlů zlepší přesnost interpolace. Hodnoty funkce a tedy i chybějícího měření je pak možné dopočítat po dosazení do této funkce. Bližší informace a výpočet této funkce je možné najít na adrese https://en.wikipedia.org/wiki/Spline_interpolation [6].

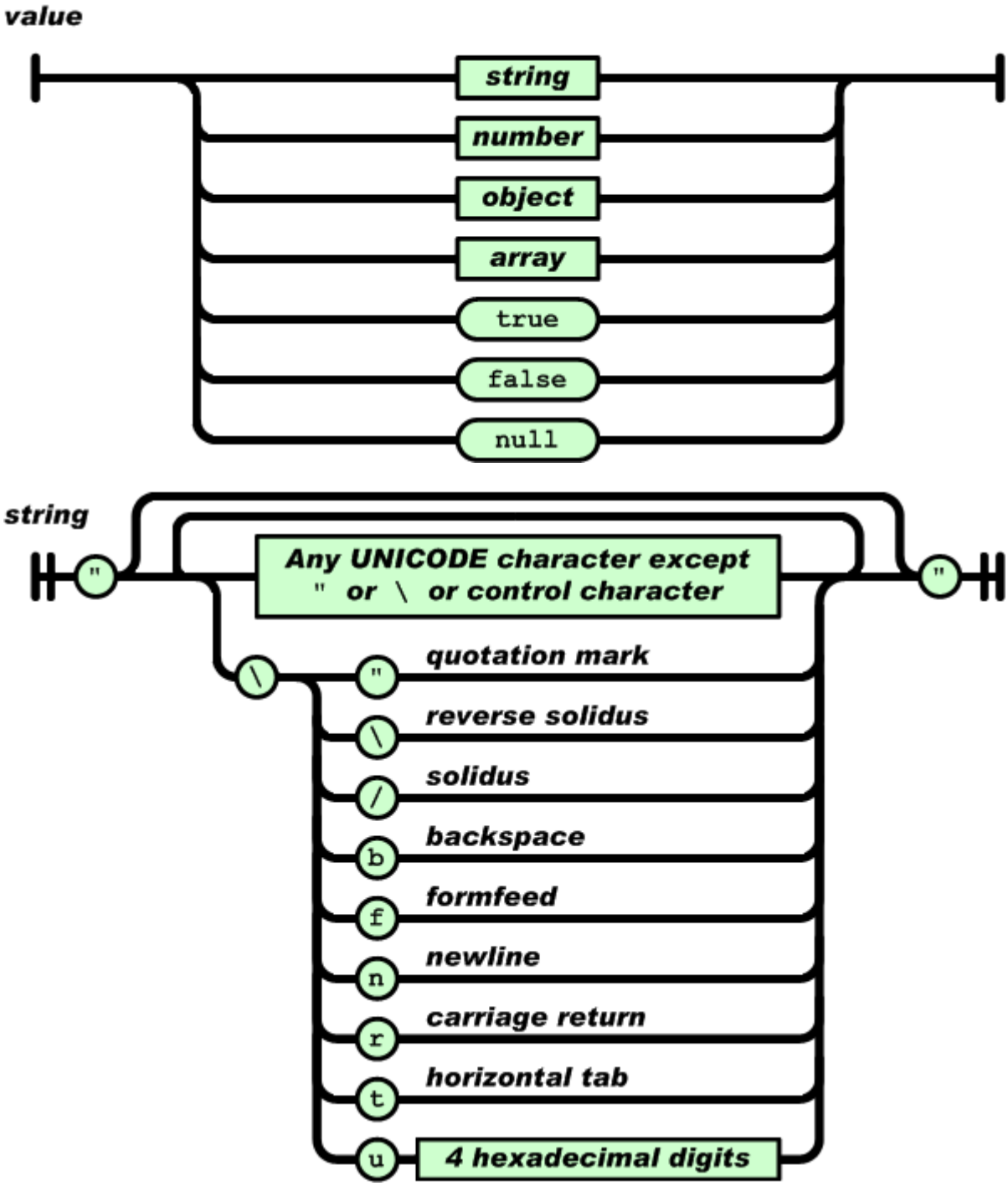
4 Obecné datové formáty

XML, YAML a JSON jsou textové soubory použité k uložení strukturovaných dat. Nejrozšířenější je XML (Extensible Markup Language) pro jeho flexibilní užívání. YAML užívá mezer a řádků pro oddělení příkazů, na rozdíl od ostatních, kde jsou použity specifické symboly. JSON (JavaScript Object Language) je používán v jazyce JavaScript a je užíván především při vývoji webových stránek a client/server komunikací. Při tvorbě frameworku bylo použito souborů JSON pro řízení chodu aplikace. JSON byl zvolen kvůli jeho jisté a snadno přehledné struktuře. Naměřená data jsou ukládána se špatnou srozumitelností pro člověka a bez popisků. Přidáním řídících souborů se práce zpřehlední a je pak uživatelsky přijatelnější.

4.1 JSON

JSON je jazyk, který slouží pro vyjádření objektů, polí, čísel, textů, hodnot boolean a null. Je založen na syntaxi JavaScript, ale je od ní odlišný. JSON je založen na dvou odlišných strukturách. Jedná se o kolekce názvů a hodnot, jako jsou objekty, slovníky nebo asociativní pole a seznamy hodnot, mezi které patří pole, vektory, seznamy nebo sekvence. Struktura jazyka použitá v rámci bakalářské práce je zobrazena na obrázku 5.





Obrázek 5: Struktura jazyka JSON [5]

5 Technologie použité při vývoji

K vývoji frameworku byl použit objektově orientovaný programovací jazyk Java. Kromě základních využití objektově orientovaného programování jako je abstrakce, třída, objekt, dědičnost, polymorfismus a rozhraní, bylo také využito následujících nástrojů.

5.1 Java Collection Framework

`Java Collection Framework` je jednotná struktura, pro reprezentování a manipulování kolekcí. Snižuje programovací náročnost a zároveň zvyšuje výkonnost programu. Framework je založen na několika rozhraních kolekcí a zahrnuje implementaci těchto rozhraní a algoritmy k jejich manipulaci.

5.2 Rozhraní List<E>

Jedná se o seřazenou kolekci, tedy seznam. Uživatel má přesnou kontrolu nad tím, kam každý element vloží. Na rozdíl od setů, seznamy většinou umožňují duplicitní elementy a výskyt nulových elementů. Rozhraní `List` přidává další možnosti nad rámec rozhraní `Collection` jako jsou metody `add`, `remove`, `equals`, `hashcode` a `iterator`. Dále `List` poskytuje čtyři metody pro přístup pomocí indexu, dvě metody hledání konkrétních prvků, dvě metody pro efektivní vložení a odstranění prvků a speciální `ListIterator` umožňující vložení a nahrazení prvků v kolekci.

5.3 Rozhraní SortedMap<K,V>

Rozhraní `Map` je veřejné rozhraní, které umožňuje tvorbu objektů mapujících klíče a hodnoty. Rozhraní `SortedMap` dále rozšiřuje rozhraní `Map`. Konkrétně jde o mapy, které jsou řazeny podle klíčů přirozeným pořadím, nebo rozhraním `Comparator` definovaným při tvorbě mapy. Také obsahuje několik dalších operací, které jsou takovýmto seřazením klíčů umožněny (jedná se o podobné metody jako u rozhraní `SortedSet`). Všechny klíče vložené do uspořádaných map musejí implementovat rozhraní `Comparable`, nebo musejí pracovat s rozhraním `Comparator`, který byl použit při vytvoření mapy. Dále musejí být klíče navzájem porovnatelné. Několik metod rozhraní `SortedMap` vrací zmenšenou mapu, takzvanou `subMap`. Takto zmenšené mapy jsou polootevřené. To znamená, že obsahují nejnižší hodnotu, na rozdíl od nejvyšší hodnoty, která zde chybí.

5.4 Aplikační programovací rozhraní Reflection

Aplikační programovací rozhraní `Reflection` lze použít na průzkum tříd, rozhraní, polí a metod za běhu programu bez znalosti jejich názvu. Avšak průzkum není jediné použití tohoto rozhraní. Také je možné vytvářet nové instance objektů, vyvolávat metody, nebo zjišťovat či měnit hodnoty polí. Běžné použití rozhraní zahrnuje funkce pro rozšiřitelnost, průzkumníky tříd, grafické vývojové prostředí, nebo testovací nástroje.

5.5 The Apache Common Mathematics

Tato knihovna obsahuje matematické a statistické funkce. Konkrétně bylo ve Frameworku využito balíčku `org.apache.commons.math3.analysis`, který slouží k hledání kořenů, integraci, interpolaci a práci s polynomy.

5.6 JSON-simple

`JSON-simple` je jednoduchý nástroj v jazyce `Java` pro `JSON` soubory. Tato knihovna byla testována pro `JSON` specifikaci `RFC4627`. Poskytuje mnoho funkcí pro kódování, dekódování a parsování `JSON` textových souborů. Další výhodou je flexibilní a jednoduchý k použití přístup pro práci s rozhraními map a seznamů.

5.7 JUnit

`JUnit` je Framework sloužící k psaní opakujících se testů. Je součástí `xUnit` architektury, která vznikla z Frameworku `SUnit`.

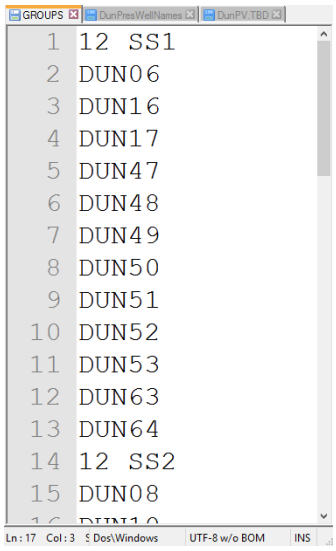
6 Struktura a popis provozních dat

Pro potřeby bakalářské práce byla využita reálná provozní data. Jedná se o data popisující vývoj teploty, tlaku a provozních objemů plynu v období od 1.8.2008 do 15.7.2014. Data jsou zaznamenávána automaticky, čímž jsou eliminovány chyby při odečtech a archivaci dat. Data jsou předávána ve formátu `TBD`.

6.1 Struktura TBD souborů

6.1.1 Skupinový soubor

Rozmístění sond na zásobníku je definováno ve skupinovém souboru, kde je na každém řádku název nové sondy nebo název nové skupiny sond, do které následující sondy patří. Nová skupina se rozezná tak, že začíná číslicí, na rozdíl od sondy, která začíná písmenem. Příklad skupinového souboru je zobrazen na obrázku 6.

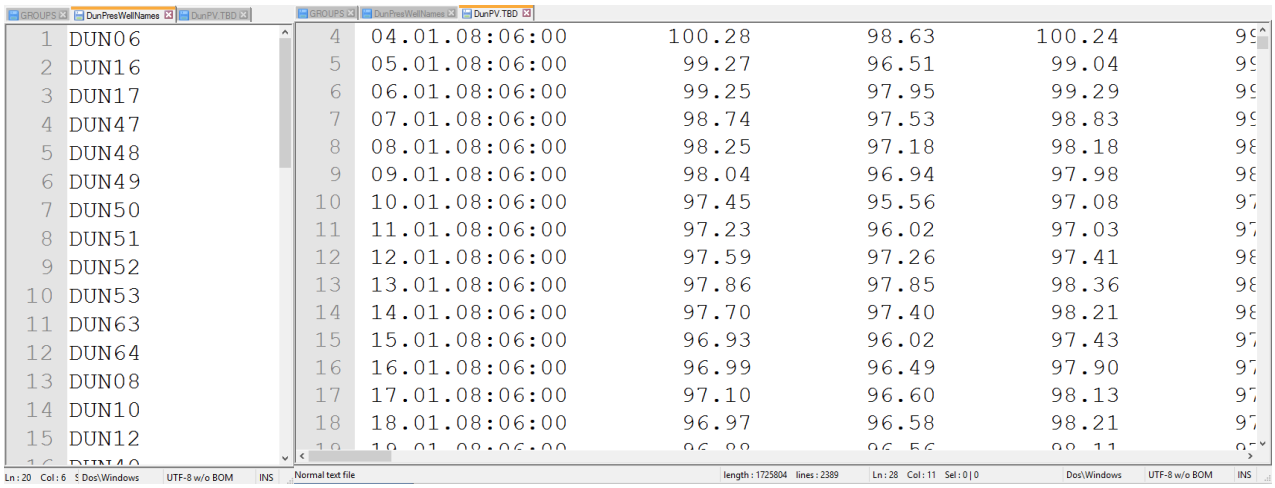


Obrázek 6: Skupinový TBD soubor.

6.1.2 Hlavičkový a datový soubor

Vlastní data měření jsou zaznamenány v datovém souboru. v prvním sloupci je čas měření a každý další sloupec obsahuje hodnoty měření pro jednotlivé sondy. Jejich pořadí je uvedeno v hlavičkovém souboru. Každý řádek hlavičkového souboru představuje sondu a podle pořadí v hlavičkovém souboru můžeme přiřadit hodnoty z datového souboru. Jednotlivé sloupce měření datového souboru se pak dají přiřadit

k sondám z hlavičkového souboru ve stejném pořadí, ve kterém byly zaznamenány. Ukázka datového a hlavičkového souboru se nachází na obrázku 7.



Obrázek 7: Hlavičkový a datový TBD soubor.

7 Specifikace požadavků

Hlavním úkolem práce bylo vytvoření frameworku. Na framework jsou kladeny následující požadavky.

7.1 Uložení dat

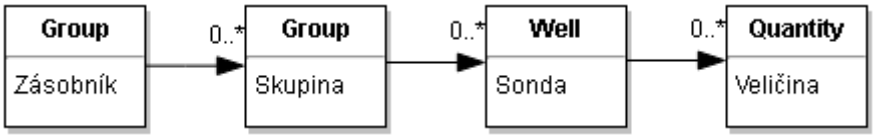
Framework musí být schopen přečíst data ze skupinových, hlavičkových a datových souborů. Tyto data následně uloží do uspořádané struktury objektů. Data reprezentují hodnoty měření a je k nim také potřeba uchovávat jejich čas, kdy byli naměřena, název a jednotky měřené veličiny a název sondy, u které měření proběhlo. Pro dodržení tohoto požadavku se měření ukládají do objektu, který obsahuje název veličiny, název jednotek a asociativní pole se záznamem měření.

7.2 Umožnění přístupu k uloženým datům

Výběr dat probíhá na základě uživatelem zadaných parametrů. Konkrétně musí framework umožnit přístup podle časů měření, které určí hledaný časový úsek. Za tímto účelem bylo navrženo uchovávání dat v datové struktuře `TreeMap`, kde jsou data automaticky uspořádány podle časů měření. Dále musí být framework schopen třídit data podle toho na jaké sondě a v jaké skupině sond měření proběhlo.

7.3 Dodržení hierarchie zásobník – skupina sond – sonda – veličina

Objektově orientované programování bylo zvoleno, protože umožňuje snadnější orientaci a pochopení člověkem. Pro jeho nejlepší využití musí framework použít objekty, které představují praktické věci. Konkrétně se v tomto případě jedná o sondu, skupinu sond a zásobník. k vyjádření veličin slouží objekty typu `Quantity`. Sondy jsou představeny jako objekty třídy `Well`. Skupina sond je sdružením sond, zásobník je skupinou skupin sond a případně i virtuál je skupinou zásobníků. Tyto 3 typy se od sebe neliší a mají stejné vlastnosti. Proto byla pro ně navržena jedna třída `Group`. Objektová hierarchie a její zastoupení tříd je zobrazena na obrázku 8.



Obrázek 8: Hierarchie objektů a jejich třídy.

7.4 Umožnění základních manipulací s daty

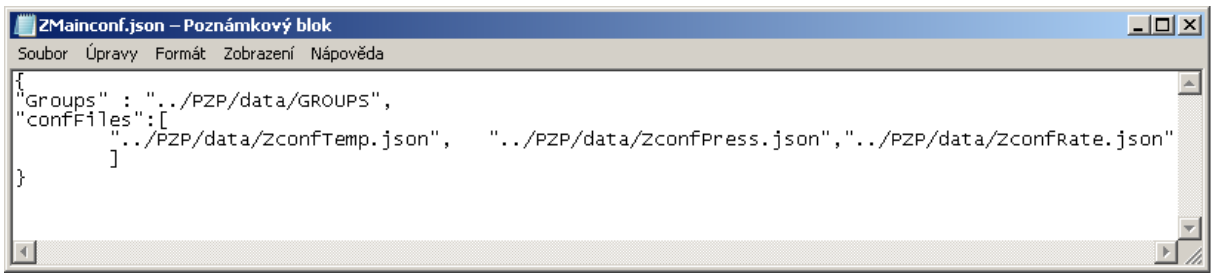
Framework by také měl obsahovat metody pro agregace v čase a ve skupině, základní čištění dat, interpolace a výpočet odvozených veličin (kumulativních objemů, statických tlaků atp.). Operace s daty v určitém čase je už splněna při výběru dat a není ji zde nutné řešit. Pro manipulaci s daty byl zvolen přístup na dvou úrovních. První možností je výběr konkrétních měření do mapové struktury a s ní zvolenou funkci provést. Druhá volba pracuje s objekty tříd `Group` a `Well`. Umožňuje tak přímo počítat s virtuály, zásobníky, skupinami a sondami.

8 Konfigurační soubory

Pro snadnější načítaní dat a pro jejich následnou manipulaci ve frameworku byly navrženy dva typy konfiguračních souborů. Data je možné načíst i bez použití těchto souborů, jen s metodami třídy `LoadTBD`.

8.1.1 Hlavní konfigurační soubor

Prvním typem je hlavní konfigurační soubor. Ten obsahuje adresu skupinového souboru, podle kterého se vytvoří struktura zásobníku. Adresa skupinového souboru musí být pod názvem `"Groups"`. Dále jsou zde adresy k dalším konfiguračním souborům, které umožňují načtení konkrétní veličiny, jejich dat a následnému vytvoření objektové struktury. Adresy konfiguračních souborů jsou zapsány v jednorozměrném poli pod názvem `"confFiles"`. Konkrétní provedení hlavního konfiguračního souboru je možné vidět na obrázku 9.



Obrázek 9: Hlavní konfigurační JSON soubor.

8.1.2 Konfigurační soubor veličiny

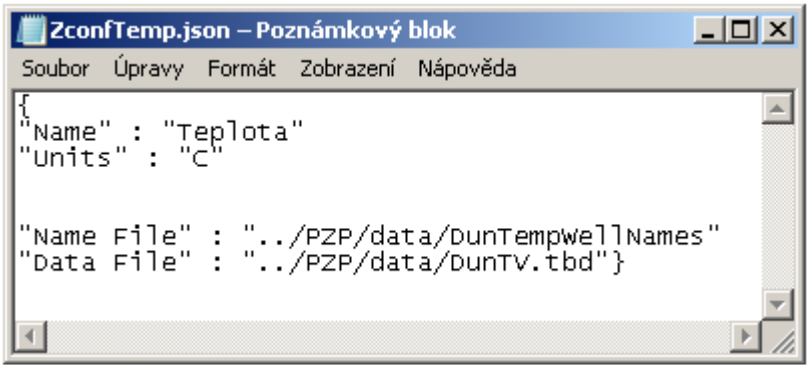
Konfigurační soubor veličiny musí obsahovat povinné atributy:

- `"Name"` - Představující název veličiny.
- `"Units"` - Představující jednotky veličiny.
- `"Name File"` – Adresa hlavičkového souboru.
- `"Data File"` – Adresa datového souboru.

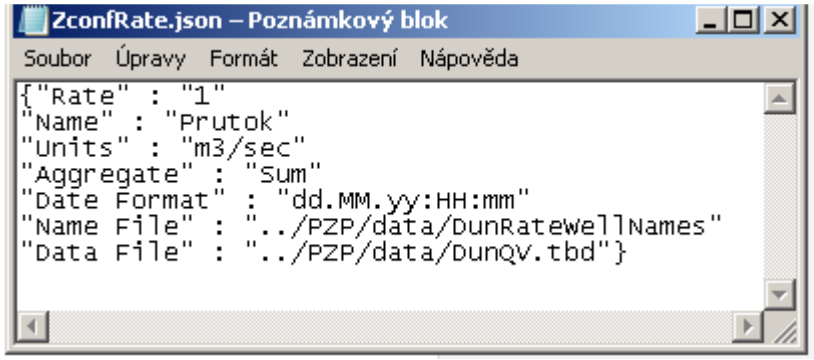
Dále mohou být specifikovány nepovinné atributy:

- "Rate" – Tato hodnota je nutná pro rozeznání veličiny, která se bude načítat. v případě průtoku je v prvním sloupci součet všech následujících hodnot, a je proto nutné s tímto souborem jednat odlišně, na rozdíl od ostatních veličin. Pokud se jedná o průtok, hodnota by měla být 1. v opačném případě ji není nutné uvádět, nebo je možné ji napsat s hodnotou 0.
- "Aggregate" – Název třídy, která bude použita pro výpočet agregace. Podle tohoto atributu bude použita operace sumy nebo průměru. Při nezadání bude zvolen atribut podle třídy objektu představující veličinu.
- "Date Format" – Styl zápisu časových hodnot měření v datovém souboru. Tento řetězec typu String bude použit pro funkci SimpleDateFormat. Při nezadání bude nastavena hodnota "dd.MM.yy:HH:mm".

  Příklady konfiguračních souborů veličin jsou zobrazeny na obrázcích 10 a 11.



Obrázek 10: Konfigurační JSON soubor veličiny obsahující pouze povinné atributy



Obrázek 11: Konfigurační JSON soubor veličiny obsahující všechny atributy

9 Design Frameworku

Následující text je věnován rozvržení Frameworku. Jsou zde vypsány třídy a jejich veřejné metody. Framework se skládá z pěti balíčků: `readers`, `functions`, `measurePoints`, `writers` a `tests`.

9.1 MeasurePoints

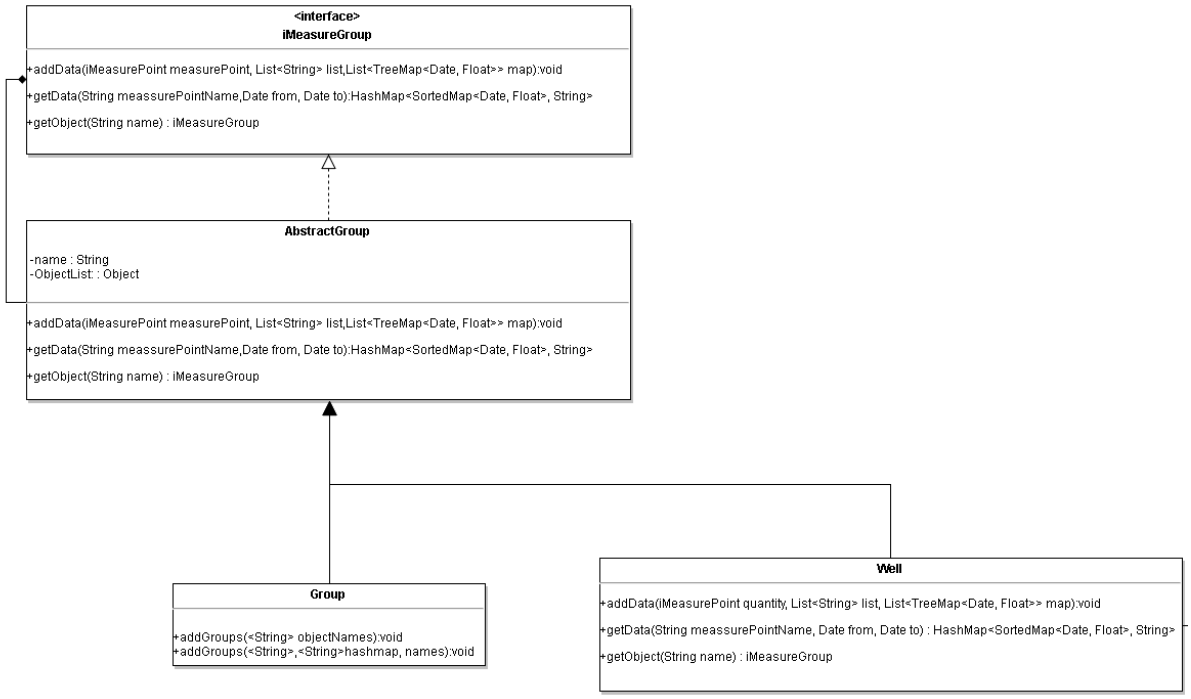
Balíček `measurePoints` vlastní třídy na vytváření objektů reprezentujících reálné virtuály, zásobníky, sondy a veličiny. Pro vyjádření virtuálů, zásobníků, skupin a sond bylo navrženo rozhraní `iMeasureGroup`. Při vývoji této části balíčku bylo využito strukturového vzoru `composite`. Pod tímto rozhraním se nachází abstraktní třída `AbstractGroup`. Třída má pouze dva atributy. Prvním je atribut `Name`, který slouží k identifikaci. Druhým atributem je `ListObject`, ve kterém jsou uloženy objektu podřazené objekty. Pro konkrétní vyjádření objektů slouží třídy `Group` a `Well`. Dále se zde nacházejí metody pro vytváření, přidávání a výběr objektů. Metody frameworku umožňující vytváření struktury budou vždy dodržovat hierarchii virtuál-zásobník-skupina-sonda (viz kapitola 7.3), tedy objekt třídy `Group` bude obsahovat ve svém atributu `ListObject`, pouze objekty typu `Group` nebo `Well`, představené rozhraním `iMeasureGroup`. `Well` se odlišuje tím, že je posledním článkem vzoru `composite` a v atributu `ListObject` obsahuje objekty z další částí balíčku. Za použití metod frameworku budou objekty tříděny do stromové struktury podle hierarchie zobrazené na obrázku 10. Avšak je možné vytvořit vlastní metody s jinou hierarchií. Metody frameworku jsou navržené rekurzivně, tedy framework bude s takto vytvořenou hierarchií umět operovat. Jedinou výjimkou jsou objekty třídy `Well`, které mají ve svém atributu `ListObject` objekty typu `iMeasurePoint`.

Druhá část balíčku `measurePoints` obsahuje rozhraní `iMeasurePoint`, které je určen pro třídy objektů představujících veličiny. Konkrétně se jedná o třídy `Quantity`. Tato třída je podobně jako u první části balíčku ukryta za abstraktní třídou `AbstractPoint`. Mezi její atributy patří:

- `Name` – Název veličiny, slouží k identifikaci.

- `Units` – Název jednotek veličiny.
- `Aggregate` – Název třídy použité k výpočtu agregace.
- `Data` – Mapa, kde klíč je datum měření a hodnota je číselný údaj.

Třída `Quantity` dále obsahuje konkrétní metodu `cloneQuant`() určenou pro vytváření nových instancí podle vzoru. Na obrázku 12 je znázorněna část balíčku představující skupinové objekty.



Obrázek 12: Část balíčku představující skupinové objekty.

9.2 Readers

Zde se nacházejí třídy s metodami pro načítání dat a jejich uložení do objektové struktury. Metody těchto tříd jsou typu `void` a komunikují s třídami balíčku `measurePoints`. Konkrétně metody vyžadují na vstup adresu v podobě proměnné `Address` typu `String`. Následně využijí vhodnou pomocnou třídu `(ParserTBD, ParserJSON, ParserData)` a uloží výstup do objektů vytvořených pomocí metod tříd v balíčku `measurePoints`. Balíček obsahuje dvě různé třídy pro dva různé typy souborů.

Pro vytvoření dat za pomoci konfiguračních souborů typu `JSON` (viz kapitola 8) je možné použít třídu `LoadJSON`. Jedinými vstupy jsou konfigurační soubory a objekt, do kterého se budou data ukládat. Není možné přidat další parametry jako je například název zásobníku nebo jednotky veličin. Vše je již obsaženo v konfiguračních souborech.

Na rozdíl od `JSON` souborů, soubory typu `TBD` tyto informace neobsahují. Po zavolání metod z třídy `LoadTBD` je tedy tyto informace třeba doplnit s využitím metod tříd v balíčku `objects`. Třída `LoadTBD` pracuje podobně jako třída `LoadJSON`. Vstupními parametry jsou opět adresy souborů a objekt, do kterého budou data uloženy.

Vlastní načítání dat ze souborů zajišťují pomocné třídy `ParserTBD, ParserJSON a ParserData`. Vstupem metod těchto tříd jsou adresy souborů. Jejich výstupy se však liší. Třídy `ParserTBD` a `ParserJSON` slouží k vytvoření seznamu hodnot typu `String`, pomocí kterého lze vytvořit hierarchii objektů a správně uložit data do příslušných objektů, kterým data patří. Data lze získat díky metodám ve třídě `ParserData`, která je vrací v podobě mapových struktur.

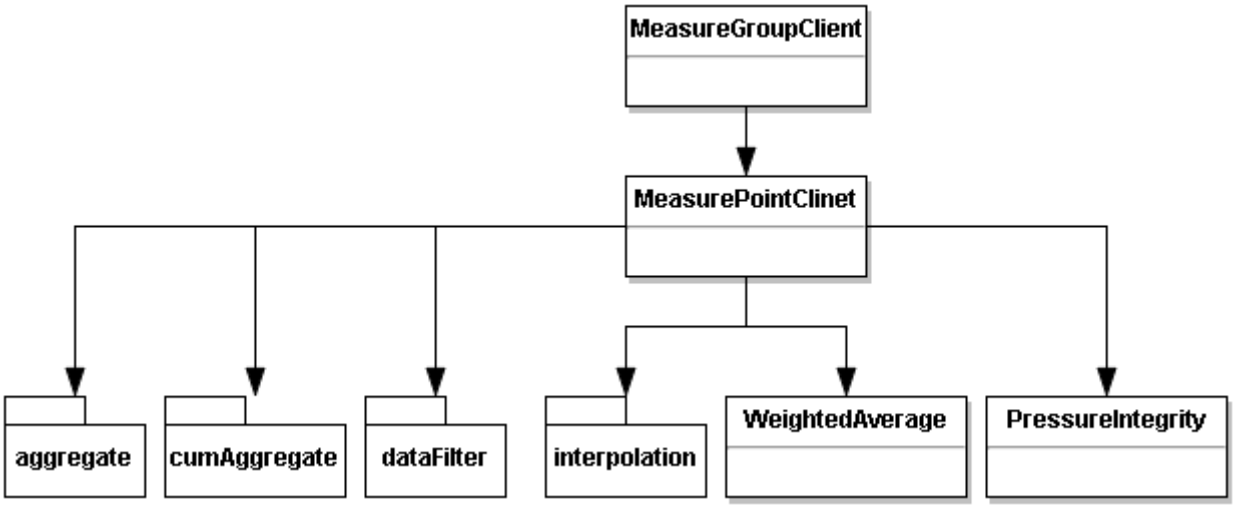
9.3 Functions

Metody pro práci s daty jsou obsaženy ve třídách tohoto balíčku. Jedná se o různé matematické operace, nalezení chybných dat a jejich doplnění. Třídy umožňující tyto operace jsou rozděleny do několika dalších balíčků, podle jejich vstupních parametrů a výstupních hodnot. Prvním vstupním parametrem je vždy mapa obsahující hodnoty dat, další parametry se ale liší podle toho, do kterého balíčku třída

patří. Každý tento balíček obsahuje také rozhraní, která umožňují vybrat požadovanou metodu na základě hodnoty typu `String`, jenž odpovídá třídě, kterou bylo třeba použít. Konkrétně byl balíček rozdělen na následující balíčky:

- `Aggregate` – obsahuje třídy pro počítání sum a průměrů.
- `CumAggregate` – obsahuje třídy pro počítání kumulativních součtů.
- `DataFilter` – obsahuje třídy pro filtrování dat na základě vstupních parametrů.
- `Interpolation` – obsahuje třídy pro dopočítávání chybějících hodnot.

Dále jsou zde třídy, které mají odlišné vstupní parametry a výstupní hodnoty od ostatních tříd, proto je nebylo možné zařadit do předchozích balíčků. Tyto třídy také pracují s větším počtem map zároveň. Jedná se o třídu `WeightedAvaragePressure`, která dokáže vypočítat vážený průměr. Dále je zde třída `RatePressureIntegrity`. Ta dokáže zjistit, zda se mění tlak při průtoku, který není nulový. Rozdělení balíčků je znázorněno na obrázku 13.



Obrázek 13: Rozdělení balíčku functions.

Pro uspořádání a jednodušší výběr metod bylo využito strukturového vzoru `factory`. Jsou zde obsaženy dva různé klienty `ClientMeasurePoint` a `ClientMeasureGroup` pracující s rozhraními balíčku `measurePoints`. Podle vstupních parametrů typu `String` se následně využije konkrétní třídy `Factory`, která

pak použije požadované třídy. Každý balíček (`Aggregate, CumAggregate, DataFilter a Interpolation`) má vlastní třídu `Factory`. Na obrázku 14 je schéma zobrazeno.

Obrázek 14: Schéma balíčku functions.

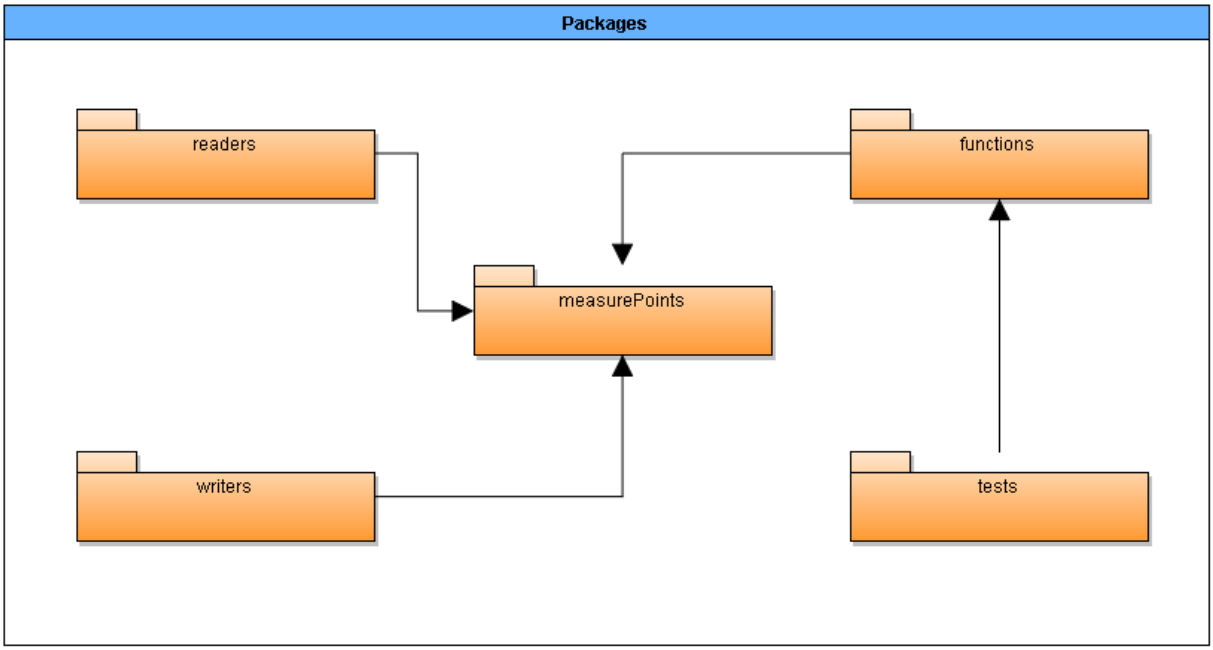
9.4 Writers

Balíček `writers` obsahuje třídy pro zápis konfiguračních a datových souborů. Dále jsou zde třídy pro výpis dat z objektů na konzolu. Pro zobrazení dat na konzolu slouží třída `ConsoleOutput`, Díky metodám v této třídě je možné vypsat data obsažené v objektové struktuře. Tato data jsou uložena v posledním členu objektové hierarchie, tedy se jedna o objekty tříd představených rozhraním `iMeasurePoint`. Ty dále obsahují mapovou strukturu. Metody třídy `ConsoleOutput` umožňují tuto strukturu nebo rozhraní zobrazit na konzolu v podobě textové tabulky.

Třída `SaveData` pracuje na stejném principu jako třída `ConsoleOutput`. Rozdíl je v tom, že vybraná data nevypisuje na konzolu, ale do předem definovaného, nebo nově vytvořeného souboru.

9.5 Tests

Tento balíček slouží k testování složitějších funkcí z balíčku `functions`. Bylo požito frameworku `JUnit`. Každý balíček v balíčku `functions` má vlastní testovací třídu. Testy mohou být spuštěny dohromady pomocí hlavní testovací třídy `RunTest`. Na obrázku 15 je schéma balíčku frameworku a jejich vzájemná komunikace.



Obrázek 15: Balíčky frameworku.

10 Podrobný popis metod

Tato kapitola se věnuje všem důležitým třídám a jejich metodám. Je zde uvedeno jejich použití a princip, na kterém pracují a jejich vzájemné komunikace.

10.1 Třídy a metody balíčku meassurePoints

První třídou tohoto balíčku je třída `AbstractGroup` využívající rozhraní `iMeasureGroup` a slouží jako abstraktní třída pro třídy `Well a Group`.

10.1.1 AbstractGroup

Třída `AbstractGroup` obsahuje těla metod z rozhraní `iMeasureGroup`. Jsou zde definovány proměnné `Name` typu `String` a `ObjectList` typu `List`. Dále se zde nalézá konstruktor, kterého využívají třídy ji implementující. Vstupním parametrem je pouze název objektu představen proměnnou `Name`. Konstruktor také inicializuje seznam podřadných objektů jako proměnnou `ObjectList`.

Metoda `getCount()` slouží k zjištění počtu objektů podřazených objektu, ze kterého byla metoda zavolána. Toho je dosaženo zjištěním velikosti proměnné `ObjectList`. Metoda nemá vstupní parametr. Na výstupu vrátí metoda hodnotu představující určitý počet objektů.

Metoda `getMeasureGroup(String)` projde rekurzivně celý seznam `ObjectList` a porovná název všech objektů se vstupním parametrem. Na výstup pak vrátí objekt v podobě rozhraní `iMeasureGroup`. Metoda je určena pro hledání konkrétního objektu představujícího virtuál, zásobník, skupinu nebo sondu.

Metoda `getMeasurePoint(String)` pracuje na stejném principu jako metoda `getMeasureGroup(String)`. Na rozdíl od předchozí metody je ale určena pro hledání konkrétního objektu představujícího veličinu. Vrací tedy objekty pod rozhraním `iMeasurePoint`.

Metoda `getData(String)` a metoda `getData(String, Date, Date)` slouží k získání dat z volaného objektu. Prvním parametrem je vždy proměnná typu `String`, která představuje název veličiny, jejíž data je potřeba získat. Druhá

metoda se od první liší tím, že je možné data časově omezit. Kvůli charakteristice rozhraní `SortedMap` je výběr dat včetně prvního datumového údaje, ale druhý datum představující konec hledání již ve výsledku obsažen není. Metoda projde obsah seznamu `ObjectList`. Pokud právě zkoumaný objekt není třídy `Well`, metoda se rekurzivně zavolá. Výsledkem metody je seznam map, ve kterých jsou klíče proměnné typu `Date` a hodnoty jsou proměnné typu `Float`. Takto získaná datová struktura je popsána pomocí textové proměnné, která je uložena ke každé datové mapě ve struktuře `HashMap`.

Metodu `addData(iMeasurePoint,List<String>,List<TreeMap<Date,Float>>)` lze využít pro vložení datových hodnot do objektů. Metoda se orientuje pomocí vstupního seznamu textových řetězců a objektu představujícího potřebnou veličinu. Metoda prochází seznamy podřazených objektů `ObjectList` rekurzivně, a v případě objektu třídy `Well` se zavolá metoda této třídy a data vloží do potřebné veličiny ze seznamu map `List<TreeMap<Date,Float>>`. v případě nenalezení požadovaného objektu se vytvoří objekt nový.

10.1.2 Group

Objekty představující virtuály, zásobníky a skupiny lze vyjádřit pomocí této třídy. Na rozdíl od své abstraktní třídy `AbstractGroup` obsahuje tři metody navíc. Jedná se o metody `addGroups(List<String>)`, `addWells(List<String>)` a `addGroups(HashMap<String><String>)` určené pro vytváření stromové skupiny objektů podle vstupního parametru obsahujícího jména konkrétních objektů. Tyto metody jsou používány balíčkem `readers`.

10.1.3 Well

Objekty třídy `Well` jsou posledním článkem vzoru `composite`, podle kterého byl balíček `measurePoints` navrhnut. Neobsahují tedy ve svém atributu `ObjectList` žádné další objekty typu `iMeasureGroup`. Sondy, které takto definované objekty představují, nemají pod sebou další skupiny, ale veličiny jako tlak, teplota nebo průtok. `ObjectList` tedy obsahuje objekty z rozhraní `iMeasurePoint`. Všechny její metody pro výběr nebo ukládání dat a objektů zděděné z abstraktní třídy `AbstractGroup` jsou tedy přizpůsobeny pro práci

s rozhraním iMeasurePoint. Konkrétně metody getMeasurePoint(String), getData(String), getData(String, Date, Date) a addData(iMeasurePoint,List<String>,List<TreeMap<Date,Float>>), zde ukončí svůj rekurzivní cyklus, udělají vyžadovanou operaci s hledanou veličinou (objektem typu iMeasurePoint) a vrátí požadovaný výsledek.

10.1.4 AbstractPoint

Druhá část balíčku measurePoints se skládá z části pod rozhraním iMeasurePoint. Podobně jako v první části se zde nalézá abstraktní třída AbstractPoint. Není už však potřeba vytvářet stromovou strukturu objektů, proto je zde pouze jedna třída Quantity představující veličiny. Rozhraní a abstraktní třída při aktuálním návrhu význam sice moc nemají, ale byly implementovány pro jednodušší možnost rozšiřitelnosti frameworku. Všechny metody a proměnné jsou definovány ve třídě AbstractPoint. Mezi její atributy patří:

- Name – Textová proměnná představující název veličiny.
- Units – Textová proměnná představující jednotky veličiny.
- Aggregate – Textová proměnná podle které se řídí funkce aggregate.
- Data – Mapová struktura typu TreeMap<Date,Float> kde jsou uloženy data měření.

Dále jsou zde dva konstruktory. Oba obsahují vstupní parametry Name a Units. Jeden ale navíc vlastní i proměnnou Aggregate. v druhém případě je tato hodnota nastavena na "Sum". Následně jsou zde metody typu get, set a metody pro vytváření objektů, načítání a hledání dat.

Metoda getData() a metoda getData(Date, Date) slouží k získání dat z volaného objektu. První metoda se od druhé liší tím, že je možné data časově omezit. Výběr dat probíhá včetně prvního datumového údaje, ale druhý datum představující konec hledání již ve výsledku obsažen není. Metoda vrací na výstup mapu

SortedMap<Date,Float>, kde se nacházejí požadovaná data. Tyto metody jsou využívané svými protějšky metod z první části balíčku measurePoints (iMeasureGroup).

Obdobně jako v předchozím případě, iMeasureGroup využívá i metody addData(SortedMap<Date,Float>). Metoda slouží k naplnění dat do atributu Data a je typu void.

K odstranění dat se dá využít metoda removeData(Date), kde se odstraní záznam atributu Data podle klíče. Metoda nemá výstupní parametr.

Metoda cloneQuant() vytvoří novou instanci pomocí objektu, ze kterého byla metoda zavolána. Metody cloneQuant() a addData(SortedMap<Date,Float>) jsou využity balíčkem readers.

10.2 Balíček readers

Balíček readers se stará o čtení souborů a následnému vytváření objektové hierarchie a uložení dat do takto vytvořených objektů. Balíček obsahuje zvláštní třídy pro čtení TBD a JSON souborů. Data se ukládají do konkrétního objektu typu Group. Rozhraní iMeasureGroup nebylo použito z toho důvodu, protože formáty a způsoby uložení datových souborů pro různé zásobníky se může velice odlišovat. s naprosto odlišnou strukturou je tedy při rozšiřování frameworku nutné počítat s odlišnými načítacími metodami. Metody pro vytváření objektové struktury tedy nejsou součástí rozhraní iMeasureGroup, ale jsou konkrétní záležitostí tříd, ke kterým objekty patří.

Čtení TBD souborů probíhá v metodách třídy ReaderTBD. Tyto metody využívají dalších metod z pomocných tříd ParserTBD a ParserData. ParserTBD byla vytvořena za účelem přečtení hlavičkových a skupinových souborů, zatímco ParserData dokáže přečíst datové soubory. Výstupem metod těchto dvou pomocných tříd jsou seznamy, hashmapy nebo seznamy datových map, které jsou pak využity třídou ReaderTBD. ReaderTBD následně s těmito údaji pracuje a vytváří nebo mění již vytvořenou strukturu objektů.

JSON soubory jsou použity jako konfigurační soubory pro načítání objektové struktury. o jejich využití se stará třída ReaderJSON a její pomocná třída ParserJSON. Využívá podobných metod jako třída ReaderTBD. ReaderJSON je

schopná načíst více souborů najednou. Je zde také možnost podrobnějšího popisu objektů. v případě nevyužití konfiguračních souborů `JSON` je možné tyto vlastnosti nastavit později u objektů manuálně.

10.2.1 Metody tříd ReaderTBD a ReaderJSON

Metoda `read(Group, String)` se nalézá u tříd `ReaderTBD` a `ReaderJSON`. v případě `ReaderTBD` vytvoří pouze stromovou strukturu objektů z parametru typu `String`, který představuje adresu hlavičkového souboru. Strukturu objektů pak uloží do objektu předaného v druhém vstupním parametru typu `String`. u třídy `ReaderJSON`, vstupní parametr typu `String`, představuje hlavní konfigurační `JSON` soubor. Podobně jako v předchozím případě podle souboru vytvoří objektovou strukturu, ale rozdíl je v tom, že metoda také načte data z adres definovaných v konfiguračním souboru a uloží je do předaného objektu s vlastnostmi definovanými v konfiguračních souborech.

Metoda `readData(Group, iMeasurePoint, String headFile, String dataFile, String dateFormat)` patří k třídě `ReaderTBD`. Slouží k uložení dat do objektové struktury. Vstupní parametr `iMeasurePoint` je vzorový objekt představující veličinu, jejíž data budou ukládána. Pomocí tohoto vzorového objektu se následně vytvářejí další objekty v objektové struktuře. První vstupní parametr `headFile` typu `String` je adresa k hlavičkovému souboru, podle kterého se následně metoda orientuje a ukládá data do příslušných objektů. Druhý `String` parametr `dataFile` je pak adresa k datovému souboru, ke kterému patřil předchozí hlavičkový soubor. Poslední textový parametr `dateFormat` slouží k identifikaci datumového formátu, který byl použit v datovém souboru. Tento textový řetězec musí být kompatibilní s rozhraním `SimpleDateFormat`. Parametr typu `Group` je objekt, do kterého jsou objektová struktura a data ukládána. Metoda je typu `void`, takže nemá žádný výstup, ale výsledek metody je uložen právě v tomto objektu typu `Group`.

Metoda `readData(Group, String)` je odlišnou verzí předchozí metody. Slouží pro načítání dat za pomoci `JSON` konfiguračních souborů. Všechny parametry jsou obsaženy v konfiguračním souboru na adrese druhého vstupního parametru a metoda se podle nich řídí. Výsledek metody je stejný jako v předchozím případě. Tedy je vše uloženo do objektu ve vstupním parametru typu `Group`.

10.3 Balíček functions

Zde se nacházejí třídy pro různé operace s daty. Konkrétně se jedná o agregační funkce, hledání chyb, interpolace nebo počítání vážených průměrů. Balíček obsahuje dvě řídící třídy, které jsou schopny pracovat s objekty tříd balíčku `measurePoints`. `MeasurePointClient` komunikuje s objekty typu `MeasurePoint`, které zastupují veličiny. `MeasureGroupClient` využívá předchozí třídy a její metody jsou schopny operovat s objekty představující virtuály, zásobníky, skupiny nebo sondy. Jednotlivé funkce pro operace s daty jsou rozděleny do podřadných balíčků podle vstupních parametrů a výstupních typů. Každý takovýto balíček má vlastní třídu `Faktory`, která je schopna na základě parametrů typu `String` vybrat požadovanou třídu a její metodu. Toho je dosaženo pomocí rozhraní `Reflection`. `MeasureGroupClient` a `MeasureGroupPoint` pak obsahují metodu pro všechny podřadné balíčky. Vstupní parametry konkrétních metod pro operace se mohou lišit v typu i četnosti, ale jeden parametr zůstává vždy stejný. Jedná se o data, která je potřeba zpracovat. Data jsou předávány jako mapa `SortedMap<Date,Float>`.

10.3.1 Balíček aggregate

Pro výpočet sum a průměrů lze použít třídy z balíčku `aggregate`. Vstupní parametr je pouze jeden, datová mapa. Výstupem je pak jedna hodnota typu `Float`.

10.3.2 Balíček cumAggregate

Od předchozího balíčku se liší funkce tím, že jsou prováděny kumulativně. Na výstupu je tedy seznam hodnot typu `Float`. Dále je balíček obohacen o třídy `Injected`, `Produced` a `Total`, které sčítají hodnoty podle jejich znamének. `Injected` sčítá pouze kladné hodnoty a `Produced` záporné. `Total` sčítá všechny hodnoty, ale v absolutní hodnotě.

10.3.3 Balíček dataFilter

Zde se nacházejí třídy pro hledání chyb v datech. Na rozdíl od předchozích balíčků obsahují metody tříd balíčku `dataFIlter` další dva vstupní parametry typu `Float`. Ty jsou pak použity pro hledání chyb v datech. Pokud funkce nalezne chybu, změní hodnotu chybného záznamu na `NaN (Not a Number)`. Výstupem je pak datová mapa `SortedMap<Date,Float>`. Třída `Peak` hledá chyby, kde se liší

hodnota předchozího a následujícího záznamu vice než je zadaný parametr. Druhým parametrem je šířka vrcholu, tedy kolik záznamů může být ve vrcholu. Třída `Range` má metodu pro hledání dat nesplňujících podmínku, kdy nepatří do zadaného rozmezí dvou čísel. Třída `Constant` kontroluje, zda data zůstávají stejné. První parametr je počet hledaných záznamů a druhým parametrem je odchylka.

10.3.4 Balíček interpolate

Balíček `interpolate` byl navržen pro vzájemnou spolupráci s předchozím balíčkem. Po nalezení chyb se záznamy s hodnotami `NaN(Not a Number)` dají dopočítat pomocí požadované metody. Balíček obsahuje třídy `Average, Linear a Spline. Average` počítá s průměrem sousedících záznamů. `Linear` pracuje na základě lineární interpolace a pro `Spline` bylo využito externí knihovny `org.apache.commons.math3.analysis.` Výstupem funkcí jsou opět datové mapy `SortedMap<Date,Float>.`

11 Použití frameworku

11.1 Instalace frameworku

Framework byl vytvořen pod verzí `java` 1.8. k jeho chodu je tedy potřeba verze 1.8 nebo vyšší. Framework je možné použít jako externí knihovnu `jar`. v případě přímého použití zdrojového kódu je potřeba přidat externí knihovny `json.simple`, `org.apache.commons.math3.analysis a Junit`.

11.2 Vytvoření objektové struktury ze souboru

Nejdříve je třeba vytvořit instance třídy `Group`, do které se bude objektová struktura ukládat. Tento objekt je pak předán jako vstupní parametr do metody pro čtení souboru a vytvoření struktury. Dále je nutná instance třídy `ReaderTBD`. Následně můžeme vyvolat metodu `read` s instancí `Group` a adresou skupinového souboru, podle kterého chceme strukturu vytvořit.

```
Group zásobník = new Group("jméno zásobníku");
ReaderTBD reader = new ReaderTBD();
reader.read(zásobník, "cesta k skupinovému souboru");
```

Třídy v balíčku `readers` obsahují metody, které vytvářejí strukturu podle konkrétních souborů. Jsou navrženy tak aby vytvořily skupiny, sondy a veličiny. Ty pak uloží do zásobníku. v případě sloučení několika vytvořených zásobníků do jednoho virtuálu, je možné použít následující postup.

```
Group zásobník1 = new Group("jméno zásobníku");
Group zásobník2 = new Group("jméno zásobníku");

ReaderJSON reader = new ReaderJSON();

reader.read(zásobník1, "cesta k JSON souboru");
reader.read(zásobník2, "cesta k JSON souboru");

Group virtual = new Group("jméno zásobníku");
virtual.addObject(zásobník1);
virtual.addObject(zásobník2);
```

Takto je možné zacházet i s jinými třídami a mezi sebou je i míchat. Například se dá takto vytvořit struktura, kde zásobník bude obsahovat skupiny a zároveň sondy. Framework s touto strukturou bude schopen pracovat, protože jsou všechny metody

volány rekurzivně. Ovšem při samotném použití metod balíčku `readers` tento případ nikdy nenastane.

11.3 Načtení dat ze souboru

Pokud už je vytvořena objektová struktura ze skupinového souboru, můžeme tento objekt použít pro doplnění dat. Na vyvolání funkce `readData` je třeba mít objektovou strukturu, instanci objektu představujícího veličinu, instanci `ReaderTBD`, hlavičkový a datový soubor a formát datumů použitých v datovém souboru. Objekt představující veličinu slouží jako vzor pro vytváření dalších instancí v objektové struktuře. Konkrétně je možné vytvořit instanci rozhraní `iMeasurePoint`. Formát datumů musí být kompatibilní s rozhraním `SimpleDateFormat`.

```
iMeasurePoint veličina = new Quantity("název veličiny", "jednotky", "aggregate");
reader.readData(zásobník, veličina, "cesta k hlavičkovému souboru",
        "cesta k datovému souboru", "datumový formát");
```

11.4 Využití JSON souborů

Pro přečtení více souborů najednou lze využít konfiguračních souborů `JSON` a třídy `ReaderJSON`. Všechny potřebné údaje jsou poskytnuty již v souborech. Na zavolání funkce `read` je potřeba pouze instance tříd `Group` a `ReaderTBD`. v předchozím případě je nutné kroky pro čtení dat opakovat pro každou veličinu zvlášť. Při využití konfiguračního souboru lze poustup udělat automatickým.

```
Group zásobník = new Group("jméno zásobníku");
ReaderJSON reader = new ReaderJSON();
reader.read(zásobník, "cesta k JSON souboru");
```

11.5 Operace s daty a zobrazení výsledků

Konkrétní objekt ze struktury je možné zobrazit pomocí metody `getObject`() s parametrem odpovídajícím názvu objektu. Nebo lze získat `objectList` dané instance a procházet v něm manuálně.

```
zásobník.getObject("název hledaného objektu");

List<Object> list = zásobník.getObjectList();
iMeasureGroup skupina = (iMeasureGroup) list.get(0);
```

Data je možné vybrat z jakékoliv instance balíčku `measurePoints` pomocí metody `getData`(). Je možné definovat datumový úsek nebo vyplnit datum hodnotou `null`. v případě instancí rozhraní `iMeasureGroup` je nutné definovat jaká veličina je hledána. To probíhá na základě textového řetězce, který je následovně porovnáván s názvy instancí rozhraní `iMeasurePoint`. Výsledkem je pak skupina map, kde je každá mapa popsaná textovým řetězcem, popisujícím pozici mapy v objektové struktuře (tedy názvem sondy, skupiny a zásobníku).

```
DateFormat format = new SimpleDateFormat("dd.MM.yy:HH:mm");
HashMap<SortedMap<Date, Float>,String> data = zásobník.getData("teplota",
        format.parse("03.01.08:06:00"), format.parse("08.01.08:06:00"));
```

Pokud se jedná o instanci `iMeasurePoint`, metoda `getData`() vrátí pouze jednu mapu dat. Instanci `iMeasurePoint` z objektové struktury je možné dostat pomocí metody `getMeasurePoint`(), podle jejího názvu.

```
iMeasureGroup sonda = zásobník.getObject("název hledaného objektu");
iMeasurePoint veličina = sonda.getMeasurePoint("název hledané veli5iny");
SortedMap<Date, Float> data = veličina.getData(format.parse("03.01.08:06:00"),
        format.parse("08.01.08:06:00"));
```

Předchozím způsobem získanou mapu lze použít k funkcím z balíčku `functions`. Je nutné vytvořit instanci třídy pro danou funkci.

```
Average průměr = new Average();
float výsledek = průměr.calculate(data);
```

Mapu nebo strukturu map lze zobrazit na konzoli za pomoci třídy `ConsoleOutput` balíčku `writers`.

```
ConsoleOutput out = new ConsoleOutput();
out.viewMap(data,format.parse("03.01.08:06:00"),
        format.parse("08.01.08:06:00"));
```

11.5.1 Operace s instancemi balíčku measurePoint

Operace nemusejí být prováděny pouze s datovými mapami. Lze využít i tříd `MeasureGroupClient` a `MeasurePointClient` z balíčku `functions`. Metody obsažené v těchto třídách jsou schopné práce s instancemi rozhraní `iMeasureGroup` a `iMeasurePoint`. Lze provádět i více metod najednou. Je potřeba využít konkrétní metody, která využije příslušné třídy `Factory` z příslušného balíčku. Orientace je prováděna na základě textového řetězce ve vstupu. Na příkladu je

znázorněna metoda pro hledání chyb pomocí datového filtru Range v rozmezí 10 až 20 a následnému dopočítání chybných hodnot pomocí lineární interpolace.

```
MeasureGroupClient client =  new MeasureGroupClient();
client.filterData(zásobník, "veličina", format.parse("03.01.08:06:00"),
        format.parse("08.01.08:06:00"), "Replace", "Linear", 10, 20);
```

12 Rozšířitelnost frameworku

Framework byl navržen tak, aby byl lehce rozšířitelný. Je možnost definovaní nových tříd do balíčků `measurePoints` i `functions`. Pokud třídy využijí příslušného rozhraní, problém nenastane.

12.1 Přidání nových operací s daty

Operace s daty se nacházejí v balíčku `functions`. Pro přidání nové operace je nutné vytvořit novou třídu. Pokud se vstupní parametry a typ výstupu operace shoduje už s nějakou předchozí, je možné pouze vytvořit novou třídu v příslušném balíčku a problém je vyřešen. v opačném případě je nutné definovat novou třídu `Factory` a metodu ve třídě `MeasurePointClinet`, případně i `MeasureGroupClient`. Pokud ale stačí pouze operace s mapou dat a není potřeba ji provádět nad instancemi tříd balíčku `measurePoint`, tato procedura není nutná. V tomto případě stačí získat mapu dat pomocí metod `iMeasureGroup`, nebo `iMeasurePoint`, kterou následně využijeme pro příslušnou operaci a usnadní se tak práce s `Factory` a `Client` třídami.

12.2 Přidání nových metod pro čtení odlišných souborů

Balíček readers byl navržen na čtení konkrétních souborů. Pokud bude struktura souborů odlišná, je potřeba vytvořit nové metody pro jejich čtení a následné vytvoření objektové struktury. Pro vytváření objektové struktury je ve třídě `Group` balíčku `measurePoints` obsaženo několik metod. Jejich vstupy mohou být seznamy nebo hashmapy textových řetězců představujících názvy oblastí měření jako jsou sondy, skupiny, zásobníky nebo virtuály. Je proto potřeba nejdříve vytvořit třídu `Parser`, která tyto seznamy a hashmapy dostane ze souboru. Následně je nutné vytvořit třídu `Reader`, která podle nich vytvoří objektovou strukturu.

12.3 Přidání nových tříd pro objekty představující oblasti měření

Za potřeby nových oblastí, které vyžadují odlišné atributy nebo metody tříd je možné přidat třídu do balíčku `measurePoints`. Je nutné využít rozhraní `iMeasurePoint` nebo `iMeasureGroup`, jinak nebude možné pracovat se zbytkem frameworku. Aby se tyto třídy daly účinně využít, bude vhodné vytvořit i nové třídy pro

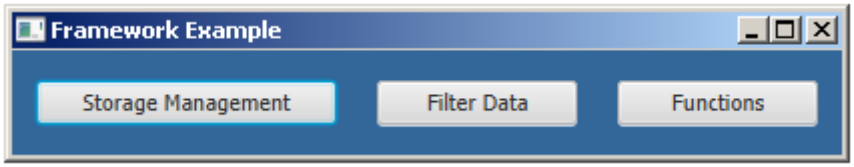
čtení ze souborů a vytvoření nové objektové struktury. Pro jednodušší definici tříd je možnost využít abstraktních tříd, které již obsahují potřebné atributy a metody pro usnadnění práce.

13 Grafické rozhraní

Grafické rozhraní bylo vytvořeno nad rámec frameworku. Důvodem byla možnost prezentace funkcí frameworku při práci s konkrétními daty a předvedení použití frameworku. v následujícím textu je popsáno použití a výsledky grafického rozhraní.

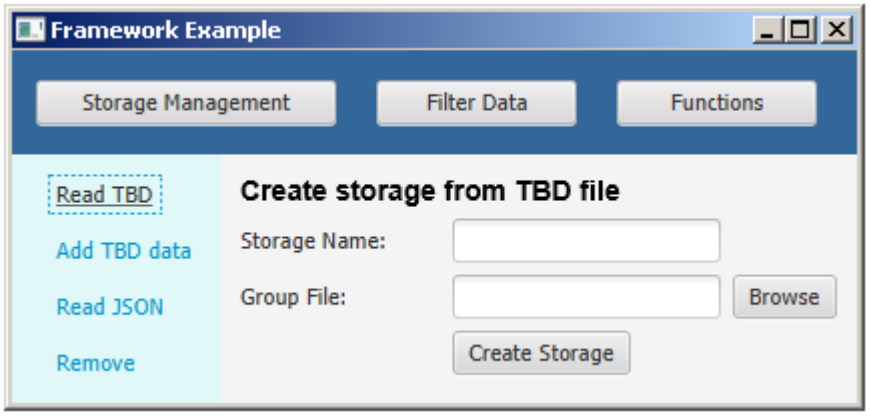
13.1 Vytvoření objektové struktury ze souboru

Po spuštění programu se objeví menu se třemi tlačítky. Menu je znázorněno na obrázku 16.



Obrázek 16: Hlavní menu.

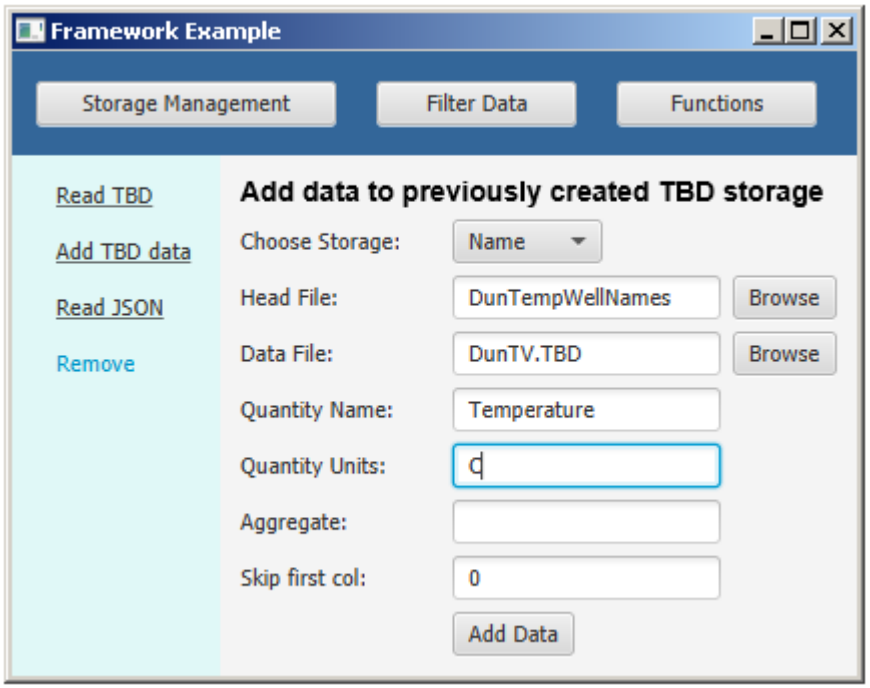
Prvním krokem po spuštění programu je vytvoření nového zásobníku. Po zvolení `Storage Management` se objeví další menu na levé straně okna (viz Obrázek 17). Nový zásobník je možné vytvořit v okně, které se objeví po stisku tlačítka `Read TBD`. Je nutné zvolit adresu skupinového souboru a název zásobníku (viz kapitola 6.1). Stiskem klávesy `Create Storage` se soubor přečte a vytvoří se podle něj zásobník, který je možné dále používat.



Obrázek 17: Vytvoření zásobníku z TBD souboru.

13.2 Načtení dat ze souboru

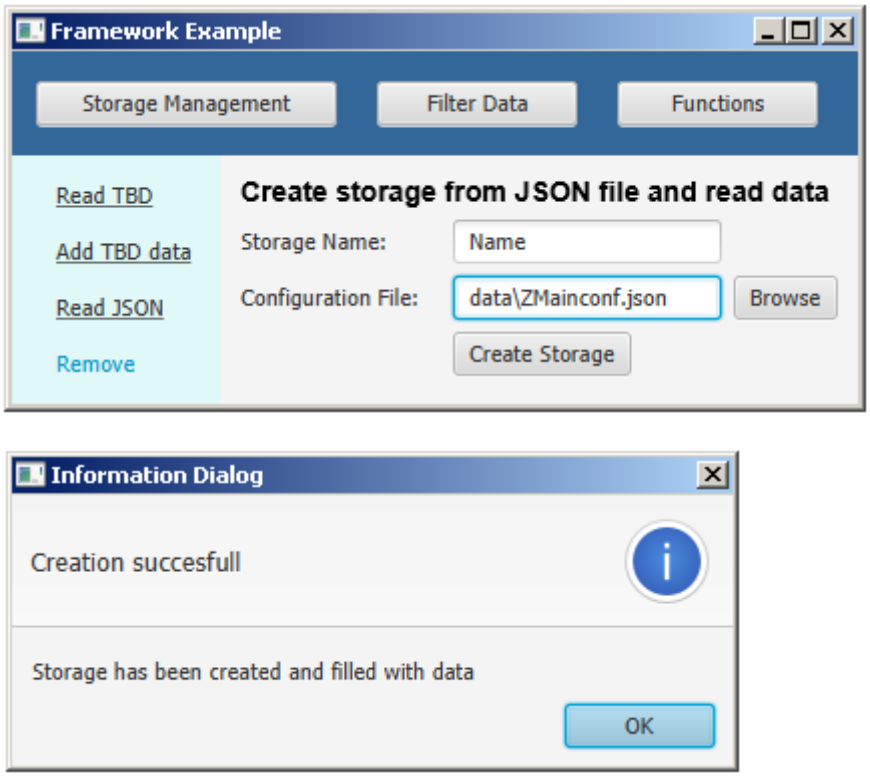
Pokud už je zásobník vytvořen, je možné zvolit `Add TBD data`. v tomto okně je třeba nejdříve zvolit zásobník, do kterého budou data přidána. Dále se musí zadat dva soubory. První soubor je hlavičkový, kde je seznam názvů skupin a sond. Další adresou je datový soubor příslušného hlavičkového souboru, kde se nacházejí konkrétní časy a hodnoty měření. Po vyplnění obou adres, názvu, jednotek a zvolení zásobníku, je možné stisknout tlačítko `ok`. Parametr `Skip first col` se využívá u průtoků, kde je nutné přeskakovat první sloupec souboru. v tomto případě je potřeba zvolit hodnotu `1`. Po stisku se data ze souboru uloží do zásobníku.



Obrázek 18: Přidání dat do zásobníku.

13.3 Využití JSON souborů

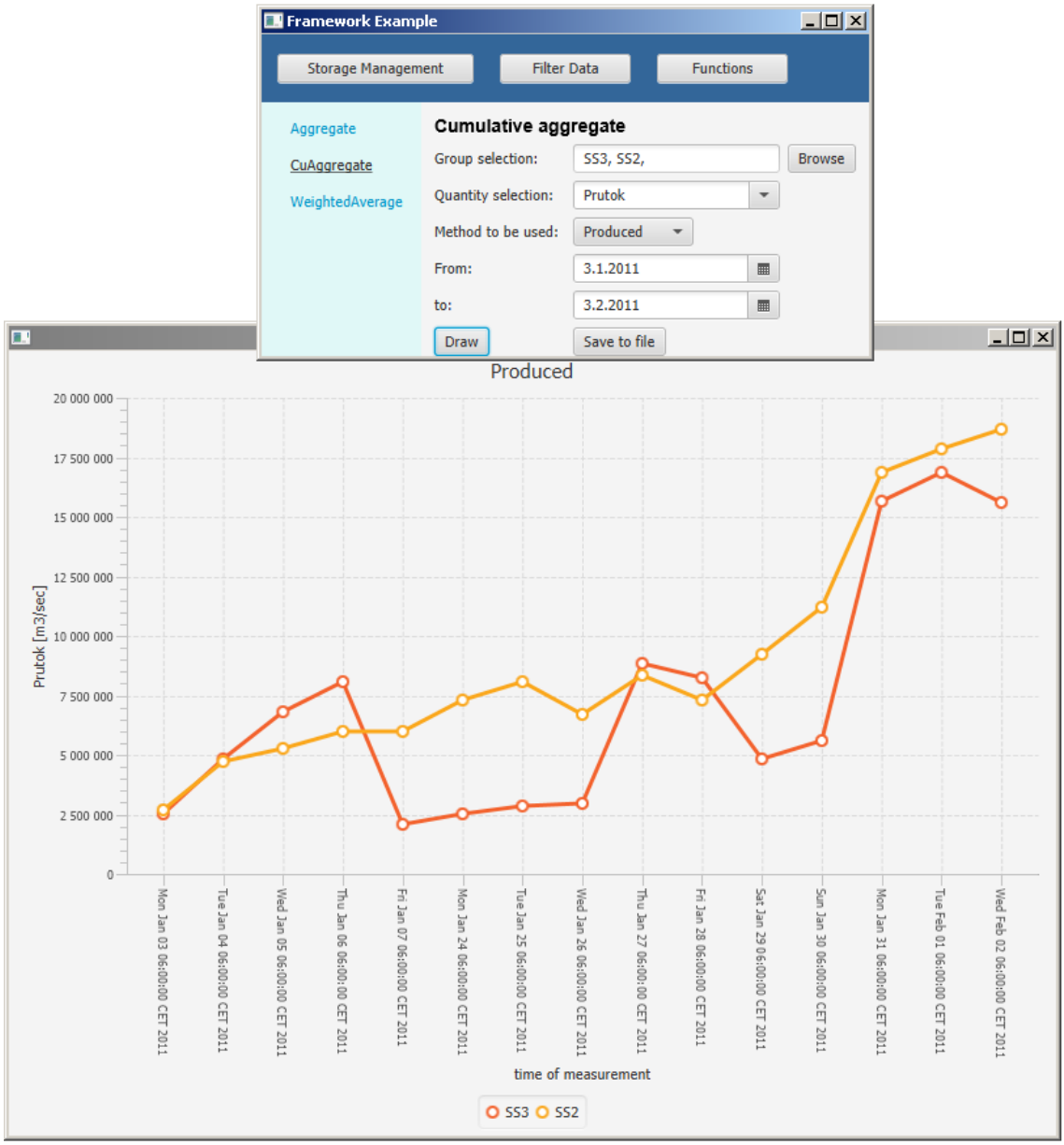
Předchozí vytváření zásobníků a přidávání dat je možně zautomatizovat využitím `JSON` konfiguračních souborů. v okně, které se zobrazí po stisknutí tlačítka `read JSON` se zobrazí pouze jedno pole pro adresu a jedno pole pro název zásobníku. Zde je třeba zadat adresu k hlavnímu konfiguračnímu souboru (viz kapitola 8.1.1). Následně se stiskne tlačítko `ok`. Pokud je obsah souborů podle návrhu frameworku, program provede kroky pro vytvoření zásobníku a načítání dat zároveň.



Obrázek 19: Vytvoření zásobníku a načtení dat z JSON souboru.

13.4 Operace s daty a zobrazení výsledků

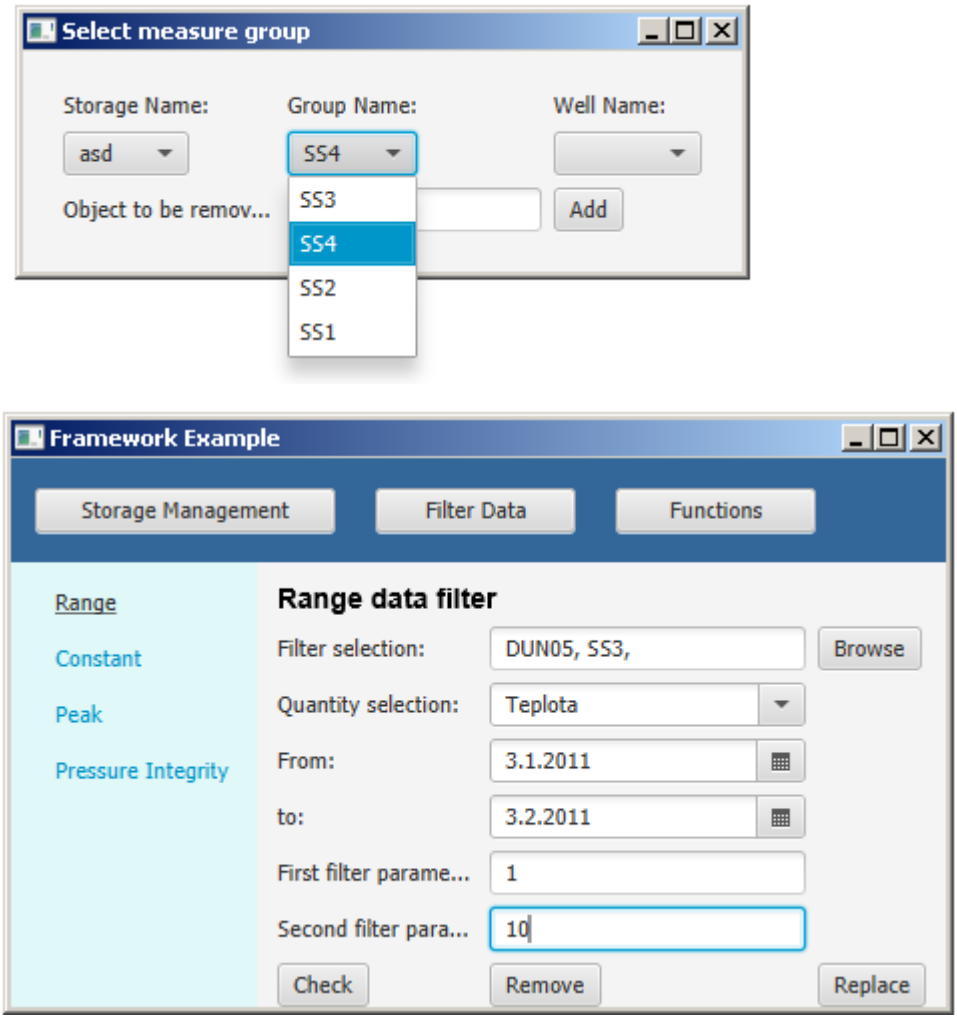
Po stisknutí tlačítka `functions` se zobrazí okno, kde je možné vybrat data a provádět různé operace. Nejdříve je nutno zvolit data, na kterých se budou operace provádět. v kolonce `Group Selection` je nutné zvolit požadovanou oblast měření. To znamená, že je nutné vybrat konkrétní sondu, skupinu nebo zásobník. Následně se výběr přidá a bude zobrazen v textovém poli. Výběry je možné libovolně odebírat. Následně je nutné zvolit potřebnou operaci a časové rozmezí (viz kapitola 6). Po stisknutí tlačítka `Draw` se operace provede nad zvolenými výběry a zobrazí se v novém okně. Je také možné stisknout tlačítko `Save to file` a uložit operaci do textového souboru.



Obrázek 19: Výpočet kumulativních funkcí.

13.5 Čištění dat

Výběr v okně, které se objeví po stisku `Filter Data` a vybrání požadovaného filtru, se objeví podobný výběr jako v předchozím případě. Po vyplnění výběru je možné zobrazit chybná data pomocí tlačítka `Check`, smazat tlačítkem `Remove`, nebo zaměnit za data vypočítaná interpolací.



Obrázek 20: Filtrování dat.

14 Závěr

Zadáním bakalářské práce bylo vytvořit framework, který umožní přečíst soubory obsahující provozní data podzemních zásobníků plynu. Data bylo potřeba uložit do objektové struktury s hierarchií virtuál-zásobník-skupina sond-sonda-veličina a s tou následně pracovat. Pro jednodušší orientaci a načítání dat byly navrženy konfigurační soubory `JSON`, pomocí kterých se framework může při čtení souborů provozních dat řídit. Pro čtení souborů byly vytvořeny třídy balíčku `readers`. Metody těchto tříd umožňují data přečíst, vytvořit stromovou strukturu objektů v podobě zadané hierarchie a přečtená data uložit do příslušných objektů.

Dále měl framework umožnit výběry dat podle času, zásobníků, skupin a sond. Toho bylo dosaženo vytvořením společných rozhraní a abstraktních tříd pro objekty ve stromové struktuře objektů. Třídy obsahují rekurzivní metody, které umožňují strukturu projít a najít hledaný výběr. Posledním objektem hierarchie jsou instance představující veličiny, ve kterých jsou data uloženy ve struktuře `TreeMap<Date,Float>`. Díky tomuto návrhu se usnadnil výběr dat podle času.

Poslední část, kterou měl framework splňovat, bylo umožnění různých matematických operací s daty. Mezi ně patří agregační metody, vážený průměr, hledání chyb a dopočítávání chybějících záznamů pomocí interpolací. Tento problém byl řešen na dvou úrovních. Framework je schopný provádět operace se samotnými daty ve formě `TreeMap<Date,Float>`, nebo je možné operace provádět přímo s objekty stromové struktury. Balíček `functions`, který umožňuje tyto operace, byl rozdělen podle druhu operací na několik podbalíčků. v každém balíčku se nachází rozhraní pro třídy operací a třída `Factory` pro usnadnění přístupu k metodám. v balíčku `functions` byla navržena třída `Client`, která umožňuje operovat s objekty.

Zdrojový kód byl zdokumentován pomocí techniky `javadoc`. Testování funkcí bylo provedeno rozhraním `JUnit`, zbytek frameworku byl testován spouštěním metod při vývoji a porovnáváním dat a struktur se vstupními soubory. Při možnosti rozšiřitelnosti byl kladen důraz na jednoduché přidávání nových operací.

Bylo také vytvořeno grafické rozhraní za účelem předvedení použití frameworku. Grafické rozhraní umožňuje provést všechny operace zadání a bylo zdokumentováno v podobě komentářů ve zdrojovém kódu.

Seznam použité literatury

[1] Yearly Report on Natural Gas Supply and Consumption in the Czech Gas System 2013, kapitola 3, s.5, www.eru.cz [online] © 2014 Energy Regulatory Office, Dostupné z: http://www.eru.cz/documents/10540/462888/ Annual_report_gas_2013.pdf/08d4cbdf-e991-40db-8888-91bf0f11461c

[2] Rozmístění a parametry podzemních zásobníků v České republice, mojeenergie.cz [online]. © 2009-2015 [vid. 2010]. Dostupné z: www.mojeenergie.cz

[3] Rwe-gasstorage.cz Dolní Dunajovice [online] © 2015 Dostupné z: http://www.rwe-gasstorage.cz/cs/dolni-dunajovice/

[4] Rosťa Jančar, Podívejte se, jak se v Česku skladuje plyn, který se nám teď hodí, technet.idnes.cz [online] © Copyright 1999–2015 [vid. 12. 1. 2009]. Dostupné z: http://technet.idnes.cz/podivejte-se-jak-se-v-cesku-skladuje-plyn-ktery-se-nam-ted-hodi-p6c-/tec_technika.aspx?c=A090108_200359_tec_technika_rja

[5] ECMA-404 The JSON Data Interchange Standard. Dostupné z: http://json.org/

[6] wikipedia.org [online] [vid. 2001] . Dostupné z : https://en.wikipedia.org/wiki/Spline_interpolation