Todolister - dokumentace

Václav Balcar, 2. ročník, 31. studijní skupina

ZS 2016/2017

Obsah

1 Uživatelská dokumentace

1.1 O programu

Todolister je program na zpracovávání dat obsahujících objekty mající informaci o čase, ve kterém se vyskytují (konají, začínají atp.).
Takovým objektům říkejme individua z časových organizérů nebo zkráceně "TOI" z angl. "time organizer individual".
Množině obsahující libovolné množství TOI říkejme "organizéry času" nebo zkráceně "TO" z angl. "time organizer".
Zkratky TOI a TO mohou reprezentovat i množné číslo zkráceného tvaru.
Program nemá GUI a ovládá se pomocí argumentů.

Zpracovávaná data lze charakterizovat pomocí následujících veličin:

Struktura TOI je řetězec, ve kterém každý znak reprezentuje jednu položku v TOI a pořadí znaků odpovídá pořadí těchto položek v TOI.

Znaky, jenž lze použít a jejich význam

1. U = unikátní identifikátor
2. C = datum a čas vytvoření události/položky todolistu
3. S = datum a čas začátku události
4. E = datum a čas konce události
5. D = datum a čas deadliny položky todolistu
6. N = pojmenování události/položky todolistu

Typ TOI je typ TOI, jaký TO obsahuje. Implemetované typy TOI jsou "event" a "todo". Event může obsahovat položky reprezentované znaky U,C,S,E,N a todo může obsahovat položky reprezentované znaky U,C,D,N.

Oddělovač argumentů TOI je retězec obsahující všechny možné jednoznakové oddělovače argumentů TOI. V případě zápisu výstupu jsou všechny argumenty TOI oddělena celým daným řetězcem.

Rozložení dat v souboru reprezentuje způsob, jakým jsou data uvnitř souboru uspořádána. Některá rozložení potřebují ještě dodatečné informace o oddělovači TOI a o pořadí a typech informací, jaké jsou v souboru s TO uloženy.
Implementovaná rozložení dat v souboru jsou "ical" a "table", kde table reprezentuje tabulkové rozložení dat v souboru, kdy jeden řádek představuje jedno TOI, kdy jednotlivé položky TOI jsou oddělené podle informací v oddělovači a "ical" reprezentuje rozložení dat dle formátu iCalendar.

Formát dat v souboru určuje formát časových dat na vstupu/výstupu, tudíž způsob, jakým se mají taková data parsovat a způsob, jakým se mají tisknout.
Implementované formáty jsou "cs-cz" (český long date-time formát), "en-us" (americký long date-time formát) a "ical" (date-time formát používaný ve formátu iCalendar).

Argumenty programu jsou posloupnost slov reprezentovaná právě tím způsobem, jakým jsou reprezentovány argumenty předávané programům přes příkazovou řádku při spouštění. Tato slova lze rozdělit na řídící a datová, kdy datová slova reprezentují argumenty posledního předcházejícího řídícího slova. Každé řídící slovo má svojí reprezentaci - Unixovou a HR (= human-readable). Řídící slova jsou dále uvedena ve své HR reprezentaci, unixová reprezentace k jednotlivým HR reprezentacím je uvedena dále v samostatné sekci.
Každá reprezentace každého řídícího slova je case-insenzitivní, datová slova jsou case-senzitivní.
Řídící slovo bez argumentů nazvěme klíčové slovo.
Nazvěme řídící slovo následované svými argumenty souslovím. Pak sousloví je uspořádaná dvojice (řídící slovo, argumenty řídícího slova).
Nazvěme posloupnost sousloví větou a posloupnost vět atomickým souvětím. Pak každé atomické souvětí je souvětí a libovolná posloupnost souvětí je opět souvětí.
Nechť seznam je souvětí obsahující prvky stejné třídy.
Nechť s-dvojice (= sémanticky jednoznačná dvojice) je uspořádaná dvojice (řídící slovo, souvětí), sousloví nebo uspořádaná dvojice (sousloví, souvětí).
Pak validní argumenty programu jsou buď posloupnost s-dvojic, nebo klíčové slovo.
Implementované s-dvojice a k nim potřebná souvětí jsou uvedené dále.

Soubor je s-dvojice ("file", jméno souboru)

Typ je s-dvojice ("containing", typ TOI)

Oddělovače jsou s-dvojice ("delimiters", oddělovač argumentů v TOI)

Obsah je s-dvojice ("select", struktura TOI)

Layout je s-dvojice ("layout", rozložení dat v souboru)

Formát je s-dvojice ("format", formát dat v souboru)

PopisIO je souvětí typ, oddělovače, obsah, layout, formát

IO je souvětí soubor, popisIO reprezentující vstup/výstup ze/do souboru nebo souvětí popisIO reprezentující vstup/výstup na standardní vstup/výstup

Registrace TO je s-dvojice ("register", (jméno, IO)). Registrace vrací zaregistrované IO vložitelné do jakéhokoliv seznamu IO.

Načtení zaregistrovaného TO je s-dvojice ("load", jméno zaregistrovaného TO). Načtení zaregistrovaného TO vrací zaregistrované IO vložitelné do jakéhokoliv seznamu IO.

Vstup je s-dvojice ("from", seznam IO)

Výstup je s-dvojice ("to", IO)

Operace je s-dvojice (název operace, argumenty operace)

Provedení operace je s-dvojice ("do", operace)

1.2 Implementované operace

Implementované jsou následující operace:

1. insert = s-dvojice (”insert”, seznam IO). Postupně vloží všechna TO ze seznamu IO na konec aktuálního TO.
2. filter = s-dvojice (”filter”, filtr). Z aktuálního TO vytvoří TO obsahující TOI splňující filtr.
3. remove = s-dvojice (”remove”, filtr). Z aktuálního TO vytvoří TO obsahující TOI nesplňující filtr.
4. sort = s-dvojice (”sort”, ASC/DESC). Setřídí aktuální TO vzestupně (¡= ASC) nebo sestupně (¡= DESC).
5. merge = s-dvojice (”merge”, seznam IO). Provede nejprve (”merge”, seznam IO) a následně (”sort”, DESC).

1.2.1 Implementované filtry

Implementované jsou následující filtry:

1. current = posloupnost argumentů (”current”, ”year”/”month”/”day”/”hour”/”minute”). Tento filtr splňují TOI s časem odpovídající aktuálnímu roku/měsíci/dni/hodině/minutě.
2. datetime = posloupnost argumentů (”datetime”, ”since”/”matches”/”until”, datetime). Tento filtr splňují TOI s časem od/v/do času datetime, což je řetězec obsahující čas/datum/časvhodný oddělovačdatum v libovolném implementovaném formátu nebo řetězec now/today reprezentující aktuální čas/dnešní datum.
3. top = posloupnost argumentů (”top”, n). Tento filtr splňuje prvních n TOI z aktuálního TO.

1.3 Zpracování argumentů

Ze vstupu se načte TO, provede se na něm posloupnost operací přečtených z argumentů a výsledek je vypsán do výstupu.
Všechna IO ze seznamu IO ze vstupu zůstávají nezměněna. Obsahuje-li seznam IO ze vstupu více položek, jsou TO z těchto položek parsovány ve stejném pořadí, v jakém jsou uvedeny na vstupu a výsledně načtené TO bude sestávat ze za sebou poskládaných TO z jednotlivých IO právě v tom pořadí, v jakém byly parsovány. V tomto seznamu stačí uvést popisIO pouze u jednoho IO, přičemž první výskyt těchto informací je považován za výchozí a pokud u některého z IO explicitně chybí, je použita právě výchozí hodnota.
Operace jsou provedeny právě v tom pořadí, v jakém jsou přečteny z argumentů.
Vstupní IO může výt zároveň i výstupní.

1.3.1 Jak předat programu argumenty

Jsou tři možnosti:

1. předat je programu standardním způsobem při spuštění přes příkazovou řádku
2. spustit program v interaktivním režimu tím, že je spuštěn bez argumentů. Argumenty jsou pak čteny ze standardního vstupu, kdy každá řádka je vyhodnocena jako samostatná posloupnost argumentů. Je-li zpracovávaná posloupnost pouze klíčové slovo ”end” nebo ”q”, je interaktivní režim a s ním i celý program ukončen.
3. spustit skript. To lze provést spuštěním programu pouze s argumenty ”skript ’cesta k souboru obsahujícímu skript’”. Každý řádek skriptu je pak vyhodnocen jako samostatná posloupnost argumentů. Je-li řádek prázdný, je na výstup zapsán také prázdný řádek. Začíná-li řádek skriptu dvojicí znaků ”**”, je zbytek řádky vypsán na výstup jako nadpis (=¿ barevně a zdobně). Začíná-li řádek skriptu dvojicí znaků ”//”, je zbytek řádky vypsán na výstup jako komentář (=¿ barevně).

1.4 Funkce Todolisteru

Todolister má například následující (i mnohé další) funkce:

1. správa vícera souborů obsahujících TO
2. načítaní TO dle IO
3. přidávání/odebírání/filtrace TOI do/z TO
4. filtrování TOI z TO (horizontální filtrování)
5. filtrování informací z TOI v nějakém TO (vertikální filtrování)
6. třídění TO
7. slučování TO
8. slévání TO
9. ukládání výsledného TO dle IO
10. konverze vstupu dle IO výstupu

1.5 Unixová reprezentace odpovídající jednotlivým HR reprezentacím

1. ”from” = ”-i”
2. ”file” = ”-f”
3. ”layout” = ”-l”
4. ”format” = ”-m”
5. ”containing” = ”-t”
6. ”delimiters” = ”-c”
7. ”select” = ”-d”
8. ”do” = ”-p”
9. ”to” = ”-o”
10. ”register” = ”-r”
11. ”load” = ”-l”
12. ”end” = ”-q”
13. ”q” = ”-q”
14. ”script” = ”-s”

1.6 Jak funguje registrace souboru

Informace o zaregistrovaném souboru se uloží do souboru ’nastavený název pro zaregistrovaný souboru’.conf. Tyto informace přesně odpovídají zaregistrovanému IO. Při načtení zaregistrovaného souboru se tyto informace z tohoto souboru opět přečtou, proto není nutné je již zadat.

2 Programátorská dokumentace

2.1 O Todolisteru

Celý Todolister je postaven na factory patternu. Chování programu je vysvětleno v uživatelské dokumentaci, zde pouze nastíním programátorské řešení některých problémů.

2.2 Význam jednotlivých složek tříd - filtrů

2.2.1 DateTime

Datetime obsahuje všechny třídy týkající se času obecně.

2.2.2 Exception

Exception obsahuje všechny výjimky, které Todolister hází. Každá taková výjimka je buď TodolisterException nebo její potomek.

2.2.3 TimeOrganizer

TimeOrganizer obsahuje implementaci obecného TO, implementace operací nad prvky z TO, implementace typů TOI a implementace filtrů.
TO je vnitřně implementovaný jako vector unikátních ukazatelů na TOI.

2.2.4 DataDescription

DataDescription obsahuje implentace layoutů a formátů.

2.3 Význam ostatních tříd

2.3.1 ArgTranslator

Třída řešící předzpracování argumentů - převedení velkých písmen na malá a následný překlad z HR reprezentace do unixové reprezentace, je-li tomu třeba.

2.3.2 ArgumentsProcessor

Třída přerozdělující argumenty mezi InputParser, OutputPrinter a TOOGeneralParser. Také poskytuje přístup k výsledkům parsování argumentů.

2.3.3 InputParser

Třída parsující vstup dle instrukcí z argumentů.

2.3.4 OutputPrinter

Třída tiskající výstup dle instrukcí z argumentů.

2.3.5 TOOGeneralParser

Třída přerozdělující argumenty mezi parsery konkrétních implementací operací nad prvky z TO.

2.4 Registry

Pilíře programu jsou třídy registrů, které obsahují konkrétní implementace všeho potřebného.
Registry jsou implementovány jako hašovací tabulky, konkrétně jako příslušný typ "unorded_map".

1. TOIIORegister = registr implementací printerů a parserů typu TOI

2. LayoutFactoryRegister = registr implementací továren layoutů
3. FormatRegister = registr implementací formátů
4. FilterParserRegister = registr implementací parserů filtrů
5. TOOParserRegister = registr parserů implementací operací nad prvky z TO

2.5 Existující implementace abstraktních tříd

Existující implementace abstraktních tříd lze vyčíst z uživatelské dokumentace a každé implementaci odpovídá nějaká třída, kterou lze nalézt v příslušné složce (filtru) v projektu.

2.6 Jak rozšířit program o další implementce

Stačí přidat instanci třídy implementace do příslušného registru. K tomu je třeba vytvořit implementaci vhodných abstraktních (virtuálních) tříd. Co představuje implementace (= vytvoření potomka) jaké abstraktní třídy, je uvedeno v následujícím seznamu:

1. TOI $\Longrightarrow$ implementace nového typu TOI
2. TOIParser $\Longrightarrow$ parser implementace nového typu TOI
3. TOIPrinter $\Longrightarrow$ printer implementace nového typu TOI
4. Layout $\Longrightarrow$ implementace nového layoutu
5. LayoutFactory $\Longrightarrow$ implementace nové továrny pro konkrétní layout
6. Format $\Longrightarrow$ implementace formátu
7. FilterParser $\Longrightarrow$ implementace parseru filtru (filtr je libovolný funktor typu function<bool(const TimeOrganizerIndividual&)>)
8. TOOParser $\Longrightarrow$ implementace parseru operace nad prvky z TO
9. TimeOrganizerOperation $\Longrightarrow$ implementace operace nad prvky z TO

2.7 Časová a paměťová složitost

Na závěr bych zmínil, že časová i paměťová složitost celého programu jsou O(velikost vstupu * počet operací).