Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Sada úloh pro PilsProg a softwarová aplikace pro shromažď'ování vyřešených úloh

Plzeň, 2014 Martin Bláha

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7. května 2014

Martin Bláha

Poděkování

Mé poděkování patří Ing. Arnoštce Netrvalové, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnovala.

Abstract

The objectives and outcomes of this work are a set of problems for PilsProg programming contest and a software application for collecting of solved problems.

The first part deals with the study of the PilsProg programming contest website and subsequently with the creation of the problem set using the available ACM validators. The purpose of the problem set is to provide a basis for a problem definition for the PilsProg programming contest.

The second part of this work describes the application for storing and sharing of solved problems. In the future, the application should also provide an additional teaching material for students involved in programming.

Obsah

1 Úvod

V první kapitole mé bakalářské práce bych vás chtěl seznámit s programovací soutěží PislProg. Pro kterou budu vytvářet sadu úloh z jednoho z neoblíbenějšího veřejného validačního serveru pro trénování programátorských úloh UVa Online Judge, se kterým vás také blíže seznámím.

Dále vám ukážu vybrané úlohy z této vytvořené sady, která bude sloužit jako podklad pro zadávání úloh pro programátorskou soutěž PilsProg.

V druhé kapitole mé bakalářské práce vám ukážu softwarovou aplikaci, která bude sloužit pro uchovávání a sdílení vyřešených úloh s popisem, s jakým algoritmem daná úloha byla vyřešena. Dále si popíšeme platformu Java s programovacím jazykem Java, která bude použita pro naprogramování aplikace. Dále se seznámíme s relační databází MySQL, která bude zvolena pro uchovávání dat a Apache Ant, který bude použit pro překlad aplikace a také si ukážeme architektury, které budou použity v aplikaci.

2 Vytvoření sady úloh pro PilsProg

Cílem této části bylo vybrat si jeden z veřejných validačních serverů a z něho vytvořit sadu úloh, která by měla sloužit jako podklad pro vytvoření zadání pro programátorskou soutěž PilsProg.

Pro vytvoření sady budeme používat validační server UVa Online Judge. Tento validační server jsem si vybral proto, že sním pracuji již od prvního ročníku na vysoké škole a mám s ním nejvíce dobrých zkušeností.

Každá úloha obsahuje zadání přeložené do češtiny a zdrojový kód, který danou úlohu řeší. Ke každé úloze je také přidán vstupní a výstupní soubor, který slouží pro reprezentaci testovaných případů.

2.1 Programovací soutěž PilsProg

2.1.1 Základní údaje

PilsProg [1] je programovací soutěž pro studenty středních škol, jejímž garantem je Katedra informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni. Tato soutěž slouží pro porovnání programátorských schopností studentů různých středních škol. Soutěž může také pomoci úspěšným soutěžícím k přijetí na Fakultu aplikovaných věd.

2.1.2 Registrace

Pro účast v soutěži je nutná bezplatná registrace na webových stránkách PilsProgu [1] v době, která je určená v harmonogramu soutěže, který se nachází na stránkách. Soutěže se můžou zúčastnit jednotlivci, ale podmínkou pro registraci je, aby byl studentem střední školy v roce, ve kterém se soutěž pořádá. Po registraci přijde soutěžícímu na e-mail registrační e-mail, který uvedl při registraci. Soutěžící je následně schopen se přihlásit. Po přihlášení si soutěžící může prohlédnout ilustrační úlohy a vyzkoušet si jejich odevzdání.

2.1.3 Pravidla soutěže

Každý soutěžící pracuje samostatně a má za úkol v určité době vyřešit a naprogramovat správné řešení pro zadané problémy.

Soutěž probíhá ve dvou kolech - kvalifikační a finálové. Kvalifikačního kola se zúčastní všichni registrovaní soutěžící. Do finálového kola už pak postupují pouze nejlepší soutěžící.

Kvalifikační soutěž probíhá z domova pomocí webových stránek [1]. Při spuštění kvalifikačního kola jsou na webových stránkách vyvěšeny programovací úlohy, které soutěžící řeší. K počtu správně odevzdaných úloh se také udává doba od začátku kvalifikačního kola do doby, kdy soutěžící odevzdal správně vyřešenou úlohu. Časy, za které soutěžící jednotlivé úlohy odevzdal, se sčítají. Tento čas poté slouží pro vyhodnocení pořadí, pakliže soutěžící mají stejný počet správně odevzdaných úloh (viz Obrázek 2.1). Soutěžící proto musí zvolit strategii takovou, že začne vypracovávat úlohy od té, která mu zabere nejméně času.

VÝSLEDKY KVALIFIKACE

Příjmení	Jméno	Splněných úloh	Čas	
Raszyk	Martin	6	14,401	postupuje do finále
Fabík	Jan-Sebastian	6	30,006	postupuje do finále
Hübsch	Ondřej	6	279,271	postupuje do finále
Davídek	Hynek	5	219,096	postupuje do finále
Hájek	Dalimil	5	1 803,039	postupuje do finále
Převrátil	Michal	3	205,653	postupuje do finále
Soukup	Jan	3	1 227,248	postupuje do finále
Rozhoň	Václav	2	144,215	postupuje do finále
Mayerhöfer	Záboj	2	145,960	postupuje do finále
Rada	Václav	2	252,667	postupuje do finále
Zima	Tomáš	2	254,788	postupuje do finále
Lejnar	Jan	2	486,171	postupuje do finále
Ngo	Thien Long	2	734,674	postupuje do finále
Valeček	Dominik	1	126,164	
Jánská	Eva	1	171,119	

Obrázek 2.1: Výsledek kvalifikace

Finálové kolo probíhá v prostorách Katedry informatiky a výpočetní techniky a je časově omezen podle počtu řešených úloh. Každý soutěžící má přiřazený počítač, na kterém řeší zadané úlohy. V průběhu finálového kola soutěžící může používat pouze knihy, které si sám donesl. Veškeré elektro-

technické pomůcky jsou zcela zakázány. Hodnocení finálového kola se provádí stejně jako v kvalifikačním kole s tím rozdílem, že za každou špatně odevzdanou úlohou se přičítá deset minut do celkového času, která byla potřeba pro vyřešené této úlohy.

2.1.4 Podporované programovací jazyky

V PilsProgu soutěžící mohou používat čtyři programovací jazyky. Podporované jazyky jsou C, C++, Java a Pascal. Každý soutěžící může využívat prostředků, které mu jazyk dovoluje mimo níže uvedených možností:

- program nesmí otevírat soubory mimo standardního vstupu a výstupu,
- program nesmí využívat síťově prostředky,
- program musí využívat pouze jeden proces.

Jazyk C

Je použit nejnovější standart C11 a úlohy jsou překládány příkazem ***gcc -std=c11 -lm -O2 vstup.c***.

Jazyk C++

Nejnovější standart je použit i u jazyka C++, kterým je C++11. Úlohy se překládají pomocí příkazu ***g++ -std=c++11 -lm -O2 vstup.cpp***.

Jazyk Java

U jazyka Java je použita verze 1.7.0. Z důvodu bezpečnosti nesmí soutěžící vytvářet vlastní balíčky.

Jazyk Pascal

U jazyka Pascal je použita verze 2.6.0.

2.1.5 Softwarové vybavení

Při programování v kvalifikačním kole je možné použít libovolný editor, který soutěžícímu vyhovuje. Ve finálovém kole budou na počítači k dispozici níže uvedené vývojové prostředí, které uživatel může využít:

- **DevC++** (verze 5.4.2),
- **Eclipse** (Kepler - verze 4.3.1),
- **FreePascal IDE** (verze 1.0.12, verze překladače 2.6.0).

2.1.6 Odevzdání úloh a odpovědi validátoru

Odevzdávání vyřešených úloh se provádí pomocí webových stránek [1]. Každý soubor, který soutěžící odevzdává, musí mít příponu podle jazyka, který soutěžící použil. C musí mít příponu .c, C++ .cpp, Java .java a Pascal .pas.

Při odevzdání programu soutěžící může dostat jednu z níže uvedených odpovědí:

- **Chyba při kompilaci** - nelze přeložit zdrojový soubor,
- **Správné řešení** - program byl uznán jako správné řešení zadané úlohy,
- **Chybné řešení** - program vrátil špatné výsledky pro zadanou úlohu,
- **Překročení časového limitu** - program neskončil v časovém limitu,

které poskytují informaci o tom, jestli daný program splňuje danou úlohu nebo ne.

2.2 Validátor UVa Online Judge

2.2.1 Základní údaje

UVa Online Judge [2] je veřejný validátor pro nácvik úloh, které se využívají v ACM soutěžích. Tento validátor patří pod španělskou univerzitu Universidad

de Valladolid a je přístupný pro všechny uživatele po bezplatné registraci. Validátor má přes 100 000 registrovaných uživatelů, kteří můžou vypracovávat přes 4 300 problémů z různých oblastí. Celé stránky jsou kompletně napsané v anglickém jazyce včetně zadání úloh. Mezi podporované programovací jazyky patří C/C++, Java a Pascal.

2.2.2 Registrace

Registrace je bezplatná a probíhá na webových stránkách [2]. Při registraci je nutné zadat jméno, e-mail, uživatelské jméno a heslo. Také máme možnost při registraci zaškrtnout, jestli chceme, aby se výsledky úloh zasílaly na e-mail. Po zaregistrování je ihned možné se přihlásit a začít řešit úlohy.

2.2.3 Úlohy

Všechny úlohy, které můžeme řešit, lze prohlížet pomocí webových stránek a jsou seřazeny do několika různých sekcí, které mimo jiné obsahují také všechny úlohy z ACM-ICPC světového finále od roku 1990 do roku 2013. Každou úlohu je také možné stáhnou ve formátu PDF. U každé úlohy je zadáno, v jaké časové době musí program skončit a kolik daný algoritmus může využít paměti, pro vyřešení dané úlohy.

2.2.4 Způsob odevzdání a odpovědi validátoru

Úlohy se odevzdávají pomocí webové stránky [2], kde je uživatel přihlášen pomocí svého uživatelského účtu. Při odevzdávání je nutné vyplnit číslo řešené úlohy, zaškrtnout jazyk, v kterém je úloha naprogramována a poté je možné dvěma způsoby vložit zdrojový kód řešení. První způsob je vybrání souboru se zdrojovým kódem. Druhý způsob je vložení zdrojového kódu do textového pole (viz Obrázek 2.2).

Obrázek 2.2: Způsob odevzdání úlohy

Po odeslání úlohy si můžeme pomocí webových stránek [2] zjistit, jakou odpověď jsme dostali od validátoru, jak můžeme vidět na obrázku (viz Obrázek 2.3). Na tomto obrázku vidíme, že každé odevzdání má svoje identifikační číslo, které je ve sloupci #. Následuje sloupec, ve kterém je společně identifikační číslo úlohy s jejím názvem. Ve sloupci *Verdict*, nalezneme odpověď od validátoru. Sloupec *Language* obsahuje jazyk, ve kterém jsme řešení napsali a ve sloupcích *Run Time* a *Submission Date*, jsou informace o času běhu a datu, kdy úloha byla odevzdána.

My Submissions

#		Problem	Verdict	Language	Run Time	Submission Date
12834805	542	France '98	Accepted	JAVA	0.259	2013-12-13 01:50:17
12833505	1449	Dominating Patterns	Accepted	JAVA	1.426	2013-12-12 18:03:40
12833500	1449	Dominating Patterns	Accepted	JAVA	1.686	2013-12-12 18:02:01
12833494	1449	Dominating Patterns	Runtime error	JAVA	0.000	2013-12-12 18:00:18
12833452	1449	Dominating Patterns	Time limit exceeded	JAVA	3.000	2013-12-12 17:45:56
12799474	12105	Bigger is Better	Time limit exceeded	JAVA	3.000	2013-12-06 22:01:23
12799420	12105	Bigger is Better	Time limit exceeded	JAVA	3.000	2013-12-06 21:40:15
12768569	147	Dollars	Accepted	JAVA	2.342	2013-12-01 20:47:29
12585282	160	Factors and Factorials	Accepted	JAVA	0.218	2013-10-29 16:43:02
12585263	160	Factors and Factorials	Wrong answer	JAVA	0.202	2013-10-29 16:39:52
12585248	160	Factors and Factorials	Wrong answer	JAVA	0.199	2013-10-29 16:33:16
12585205	160	Factors and Factorials	Wrong answer	JAVA	0.202	2013-10-29 16:25:13
12585103	160	Factors and Factorials	Wrong answer	JAVA	0.266	2013-10-29 16:08:42
12578951	834	Continued Fractions	Accepted	JAVA	0.175	2013-10-28 16:19:26
12578830	834	Continued Fractions	Runtime error	JAVA	0.000	2013-10-28 16:00:17
12578095	444	Encoder and Decoder	Accepted	JAVA	0.216	2013-10-28 14:37:12
12578005	444	Encoder and Decoder	Runtime error	JAVA	0.000	2013-10-28 14:26:41
12577544	10013	Super long sums	Time limit exceeded	JAVA	3.000	2013-10-28 13:20:54
12577496	10013	Super long sums	Time limit exceeded	JAVA	3.000	2013-10-28 13:12:23
12577457	10013	Super long sums	Time limit exceeded	JAVA	3.000	2013-10-28 13:07:14

Obrázek 2.3: Informace o odevzdaných úlohách

Výsledek se také zasílá na náš e-mail, pakliže jsme při registraci zaškrtli odesílání výsledků na e-mail. Možné odpovědi od validátoru jsou:

- **In Queue (QU)** - Validátor je zaneprázdněn a nemůže hned poskytnout informace o správnosti. Výsledky budou hned co bude moc.
- **Accepted (AC)** - Řešení bylo akceptované pro dané zadání.
- **Presentation Error (PE)** - Výstup řešení je správný, ale výstup je špatně prezentován.
- **Wrong Answer (WA)** - Výstup vašeho řešení je chybný.
- **Compile Error (CE)** - Při kompilaci došlo k chybě.
- **Runtime Error (RE)** - V průběhu běhu programu se vyskytla chyba.
- **Time Limit Exceeded (TL)** - Vaše řešení překročilo časový limit pro danou úlohu.
- **Memory Limit Exceeded (ML)** - Program překročil paměťové nároky, které bylo možné využít pro danou úlohu.

- **Output Limit Exceeded (OL)** - Program vypisuje příliš mnoho informací.
- **Submission Error (SE)** - Nepodařilo se odevzdat úlohu.
- **Restricted Function (RF)** - Program využívá funkce, které jsou zakázané.
- **Can't Be Judged (CJ)** - Program nemůže být posouzen z důvodu, že validátor nemá vstupní a výstupní soubory.

2.3 Vybrané úlohy ze sady úloh

Níže si můžete prohlédnout ukázky úloh z vypracované sady. Všechny úlohy byly naprogramovány pomocí programovacího jazyka Java a všechny byly úspěšně validovány na validačním serveru UVa Online Judge [2]. U každé úlohy je originální zadání, přeložené originální zadání a je popsána základní myšlenka úlohy s možným způsobem řešení. U každé úlohy je napsáno, jaké problémy se u dané úlohy při řešení mohou vyskytnout.

Na přiloženém CD poté můžete najít všech třiadvacet vyřešených úloh (viz Obrázek 2.4) s PDF souborem, ve kterém je originální zadání úlohy. Naleznete tam také přeložené zadání do češtiny s popisem možného způsobu řešení pro danou úlohu. Ke každé úloze jsou vytvořeny vstupy a výstupy, které taktéž naleznete na přiloženém CD. Všechny úlohy jsou na přiloženém CD rozděleny do tří složek *Lehké*, *Středně obtížné* a *Obtížné*, podle obtížnosti řešení.

13038516	465	Overflow	Accepted	JAVA	0.235	2014-01-27 07:29:34
13038511	10077	The Stern-Brocot Number System	Accepted	JAVA	1.332	2014-01-27 07:28:42
13038509	10034	Freckles	Accepted	JAVA	0.395	2014-01-27 07:28:13
13038503	10336	Rank the Languages	Accepted	JAVA	0.198	2014-01-27 07:27:37
13038498	10405	Longest Common Subsequence	Accepted	JAVA	0.239	2014-01-27 07:26:55
13038492	10800	Not That Kind of Graph	Accepted	JAVA	0.326	2014-01-27 07:26:15
13038485	10815	Andy's First Dictionary	Accepted	JAVA	1.602	2014-01-27 07:25:09
13038482	10878	Decode the tape	Accepted	JAVA	0.532	2014-01-27 07:23:57
13038479	11371	Number Theory for Newbies	Accepted	JAVA	0.192	2014-01-27 07:23:16
13038478	11827	Maximum GCD	Accepted	JAVA	0.199	2014-01-27 07:22:35
13038475	1235	Anti Brute Force Lock	Accepted	JAVA	2.029	2014-01-27 07:21:53
13038471	1449	Dominating Patterns	Accepted	JAVA	1.459	2014-01-27 07:21:10
13038466	147	Dollars	Accepted	JAVA	2.639	2014-01-27 07:19:58
13038463	160	Factors and Factorials	Accepted	JAVA	0.192	2014-01-27 07:19:07
13038460	290	Palindroms <---> smordnilaP	Accepted	JAVA	2.018	2014-01-27 07:18:32
13038453	401	Palindromes	Accepted	JAVA	1.122	2014-01-27 07:17:43
13038452	441	Lotto	Accepted	JAVA	0.359	2014-01-27 07:17:12
13038447	444	Encoder and Decoder	Accepted	JAVA	0.242	2014-01-27 07:16:34
13038421	536	Tree Recovery	Accepted	JAVA	0.189	2014-01-27 07:12:54
13038416	542	France '98	Accepted	JAVA	0.259	2014-01-27 07:11:52
13038403	543	Goldbach's Conjecture	Accepted	JAVA	1.825	2014-01-27 07:09:52
13038397	834	Continued Fractions	Accepted	JAVA	0.202	2014-01-27 07:09:13
13038357	10013	Super long sums	Accepted	JAVA	1.099	2014-01-27 07:01:08

Obrázek 2.4: Seznam vyřešených úloh

Při řešení úloh, jsem využíval knížku Učebnice jazyka Java [3], která mi pomohla se syntaxí Javy. Pro algoritmy, jsem využíval knížku Programming Challenges [4], která je dostupná ke stažení na Internetu a knížku Algorithms (4th Edition) od autorů Roberta Sedgewicka a Kevina Wayna. Nejvíce mi, ale pomohla při řešení webová stránka [5], kde se dají najít diskuze o dané úloze.

2.3.1 Seřad'te jazyky

Zadání

Pravděpodobně jste si všimli, že angličtinou a španělštinou se mluví v mnoha oblastech po celém světě. Teď by bylo zajímavé, seřadit všechny jazyky podle počtů států, kde se daným jazykem mluví.

Dostanete mapu, na které jsou zobrazeny státy s jazykem, se kterým daný stát mluví. Podívejte se na následující mapu jako příklad.

ttuuttdd

ttuuttdd
uuttuudd
uuttuudd

Každé písmeno je zkratka pro jazyk a státy. Stát je definován jako spojená plocha se stejným písmenem. Dvě písmena jsou spojena, pokud druhé písmeno je vlevo, vpravo, nad nebo pod. To znamená, že na mapě jsou tři státy, kde se mluví jazykem ‚t‘ , tři státy, kde se mluví jazykem ‚u‘ a jeden stát, kde lidé mluví jazykem ‚d‘ .

Vaším úkolem je určit počet států pro každý jazyk a vytisknout výsledky v zadaném pořadí.

Vstup

První řádek obsahuje počet testovaných případů N. Každý testovaný případ se skládá z řádku s dvěma čísly H a W, která udávají výšku a šířku mapy. Poté následuje H řádků s řetězcem W písmen. Tyto písmena budou vždy malá od ‚a‘ do ‚z‘ .

Výstup

Pro každý testovaný případ vypište „World #n“, kde n je číslo testovaného případu. Poté vypište řádek pro každý jazyk, který se objeví v testovaném případě. Řádek obsahuje jazyk, dvojtečku, mezeru a počet států, kde se tímto jazykem mluví. Tyto řádky jsou vypsány sestupně podle počtu států. Pokud dva jazyky mají stejný počet států, seřadí se podle abecedy, což znamená, že například jazyk ‚i‘ bude před jazykem ‚q‘ .

Příklad vstupu

2
4 8
ttuuttdd
ttuuttdd
uuttuudd
uuttuudd

```
9 9
bbbbbbbbb
aaaaaaaab
bbbbbbbab
baaaaacab
bacccccab
bacbbbcab
baccccab
baaaaaaab
bbbbbbbbb
```

Příklad výstupu

```
World #1
t: 3
u: 3
d: 1
World #2
b: 2
a: 1
c: 1
```

Shrnutí zadání

Cílem tohoto úkolu je zjistit z mapy počet států, ve kterých se mluví stejným jazykem. Vstupní mapa je dána jako pole o velikosti m x n. Jazyk je definován jako písmeno na mapě. Pokud se stejná písmena dotýkají hranou, jde o stejný stát.

Způsob řešení

Tento úkol lze vyřešit pomocí rekurzivního programování, kde budeme postupovat od jednoho písmena k dalšímu a od každého písmena půjdeme směrem doleva, nahoru, doprava a dolů. Každý znak, který objevíme si označíme určitým znakem, který bude označovat, že dané písmeno bylo zpracováno. Tento znak bude sloužit s hranicemi mapy jako ukončení rekurze. Při zpracovávání písmen si děláme statistiky o počtu jazyků.

Možné problémy

Z důvodu, že jsme použili pro řešení úlohy rekurzivní algoritmus se může vyskytnout problém ohledně nedostatku paměti. Z tohoto důvodu je potřeba zařídit, aby každé písmeno, které už bylo zpracované rekurzí, zastavilo postup další rekurze, které by dané písmeno chtělo zpracovávat.

2.3.2 Dekódování pásky

Zadání

Váš šéf právě objevil v archívu staré počítačové pásky. Na těchto páskách s otvory, můžou být užitečné informace. Je na vás, abyste zjistili, co je na nich napsáno.

Vstup

Vstup bude obsahovat jednu pásku.

Výstup

Výstupní zpráva, která je napsaná na pásce.

Příklad vstupu

```
-----------
| ooo .   o|
| ooo .o o|
| oo o.   o|
| oo   . oo|
| oo o. oo|
-----------
```

Příklad výstupu

quick

Shrnutí zadání

Šéf dal programátorovi starou děrnou pásku a chce zjistit, jestli na ní nejsou uloženy nějaké důležité informace.

Způsob řešení

Tato úloha je velmi jednoduchá, pokud řešitel ví, jak jsou data uložena na pásce. Každá páska má na každé řádce 6 otvorů. Pokud tento otvor je prázdný, znamená to, že jeho hodnota je jedna, pokud plný tak 0. Ze šesti otvorů dostaneme 6-ti místné dvojkové číslo, které po převodu do desítkové soustavy udává pozici znaku v ASCII tabulce (viz Obrázek 2.5).

```
ooo .   o -> 1110001 -> 113 -> q
ooo .o o -> 1110101 -> 117 -> u
oo o.   o -> 1101001 -> 105 -> i
oo   . oo -> 1100011 -> 99  -> c
oo o. oo -> 1101011 -> 107 -> k
```

Obrázek 2.5: Způsob převodu

Možné problémy

Jediný problém, který může nastat při řešení této úlohy je, že programátor nebude znát, jak jsou data uložena na pásce. Pokud by to nastalo, úlohu by nebyl schopen vyřešit.

2.3.3 Přetečení

Zadání

Napište program, který přečte výraz skládající se ze dvou kladných čísel a operátoru. Zjistěte, zda celá čísla nebo výsledek výrazu je příliš velký pro prezentaci jako „normální“ neznaménkový *integer*. Typ *integer*, pokud pracujete v Pascalu, typ *int* pokud pracujete v C.

Vstup

Neznámý počet řádků. Každý řádek obsahuje celé číslo, jeden ze dvou operátorů ‚+‘ nebo ‚*‘ a druhé celé číslo.

Výstup

Pro každý řádek vstupu, vytisknete vstup nasledovaný 0 až 3 řádky obsahující jednu z těchto zpráv podle vhodnosti: „first number too big“ (první číslo je příliš velké), „second number too big“ (druhé číslo je příliš velké), „result too big“ (výsledek je příliš velký).

Příklad vstupu

300 + 3
99999999999999999999 + 11

Příklad výstupu

300 + 3
99999999999999999999 + 11
first number too big
result too big

Shrnutí zadání

Cíl této úlohy je načíst ze řádky dvě čísla a jeden operátor a zjistit, jestli zadaná čísla nebo spočítaný výsledek nepřetekl datový typ *integer*.

Způsob řešení

Načteme řádek, z něhož získáme dvě čísla a operand. Musíme počítat s tím, že mezi čísly a operandem může být víc jak jeden prázdný znak. Pro uchování všech čísel použijeme *BigInteger*. Následně budeme testovat, jestli první číslo nepřeteklo. Pokud číslo má méně než 10 znaků, nepřeteče. Pokud bude mít více jak 10 znaků, přeteče. Pakliže číslo má právě 10 znaků, musíme nejdříve otestovat, jestli dané číslo je menší nebo větší než největší hodnota typu *integer* a podle toho rozhodnou, jestli dané číslo přeteče nebo ne. Tento postup provedeme i pro druhé číslo. Poté obě čísla sečteme či vynásobíme podle vstupního operandu. Výsledek budeme vyhodnocovat jako předešlá čísla.

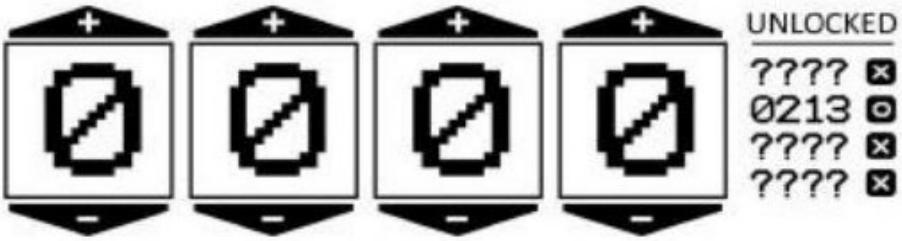
Možné problémy

Může se stát, že validátor bude stále dokola dávat odpověď, že dané řešení je nesprávné. Jedním z důvodů může být, že jsme si neuvědomili, že mezi zadanými čísly a operátorem se může vyskytnout libovolný počet bílých znaků, které jsme zapomněli odstranit.

2.3.4 Zámek

Zadání

V poslední době je jeden závažný problém s Panda Land Safe Box. Několik trezorů bylo vykradeno. Trezory používají 4 číselnou kombinaci válcového zámku (viz Obrázek 2.6). Můžete pouze otáčet číslicemi směrem nahoru nebo dolů, dokud všechna čtyři čísla neodpovídají klíči. Každá číslice je navržena tak, aby se otáčela od 0 do 9. Otočením nahoru z 9 se dostaneme na číslo 0 a otočení dolů z 0 se dostaneme na číslici 9. Vzhledem k tomu existuje 10 000 klíčů, 0000 až 9999. Každý může vyzkoušet všechny kombinace, dokud se sejf neodemkne.



Obrázek 2.6: Válcový zámek

Co se stalo, stalo se. Aby bylo možné zpomalit útok budoucích lupičů, Panda Security Agency (PSA) vymyslela nový bezpečnostní zámek s několika tlačítky. Namísto použití jedné kombinace klíče, zámek nyní může mít až N klíčů, které musí být všechny odemknuty, aby šel trezor odemknout. Tento zámek funguje takto:

- Zpočátku jsou čísla na 0000.
- Klíče můžou být uvolněny v libovolném pořadí.
- Magické tlačítko JUMP může nastavit číslice do některého z odemčených klíčů, aniž by bylo nějaké otočení.
- Sejf se odemkne jenom tehdy, pokud jsou všechny klíče odemknuty s minimálním počtem otočení s výjimkou JUMP.
- Pokud dojde k překročení počtu otáčení, čísla se vrátí do 0000 a všechny klíče budou opět zamčeny. Tedy bude stav zámku vynulován a prolomení bylo neúspěšné.

PSA je zcela přesvědčené, že tento nový systém zpomalí prolamování a dá jim dostatek času na identifikaci a chycení zloděje. Aby bylo možné určit minimální potřebný počet otočení, chce PSA napsat program. Dostanete všechny klíče, spočítejte minimální počet otočení, které je potřeba k odemčení sejfu.

Vstup

První řádek obsahuje celé číslo T, které udává počet testovaných případů. Každý případ začíná celým číslem $N(1<=N<=500)$ následuje počet klíčů. Každá z N dalších řádek obsahuje přesně 4 číselná čísla (přední nuly jsou povoleny) představující klíče pro odemknutí.

Výstup

Pro každý testovaný případ, vytisknete na jednom řádku minimální počet otáčení, potřebný k odemknutí všech klíčů.

Vysvětlení pro 2. případ:

- Dejte 0000 do 1111, otočení: 4
- Dejte 1111 do 1155, otočení: 8
- Skok z 1155 do 1111, můžeme to udělat, protože 1111 bylo už odemknuto.
- Dejte 1111 do 5511, otočení 8

Celkem otočení = 4 + 8 + 8 = 20

Příklad vstupu

```
4
2 1155 2211
3 1111 1155 5511
3 1234 5678 9090
4 2145 0213 9113 8113
```

Příklad výstupu

16
20
26
17

Shrnutí zadání

Dostaneme množinu klíčů, které mají odemknout čtyřciferný rolovací zámek. Zámek se odemkne, pokud odemkneme všechny klíče s nejméně otočením. Zámek má také funkci, která dokáže nastavit cifry na hodnotu z dříve odemčeného klíče. Při této funkci se nezapočítává žádné otočení.

Způsob řešení

Načteme všechny klíče, které budou sloužit jako vrcholy grafu. Poté vytvoříme hrany, mezi všemi vrcholy a vypočítáme hodnotu hrany tak, že zjistíme, kolik je potřeba otočení na zámku, abychom jsme se dostali z jednoho klíče do druhého. Následně budeme hledat minimální kostru grafu. Nejdříve si seřadíme hrany od nejmenší hodnoty, kterou jsme si vypočítali. Poté budeme procházet seřazené hrany a ukládat si výsledné hrany, které budou spojovat vrcholy. Hranu dáme do výsledných, pokud hrana, se kterou spojíme dva vrcholy nevytvoří v grafu cyklus. Pokud ho vytvoří, hranu zahodíme. Po zjištění hran, která tvoří kostru, projedeme všechny tyto hrany a budeme sčítat jejich hodnoty. Poté musíme k výsledku přičíst nejmenší hodnotu, z které se dostaneme od počáteční hodnoty do některého z klíčů. Výsledná hodnota je řešení pro daný případ.

Možné problémy

Tato úloha nejde řešit pomocí brutální síly z důvodu, že se nevejdeme do časového limitu. Z tohoto důvodu si musíme uvědomit, že daný problém lze řešit pomocí minimální kostry grafu.

2.3.5 Nejdelší společná posloupnost

Zadání

Máte dvě sekvence znaků. Vypište délku nejdelší společné posloupnosti obou sekvencí. Například nejdelší společná posloupnost z těchto dvou sekvencí

abcdgh
aedfhr

je *adh* délky 3.

Vstup se skládá ze dvou řádek. První řádek obsahuje první řetězec, druhý řádek obsahuje druhý řetězec. Každý řetězec je na samostatné řádce a obsahuje nejvýše 1000 znaků.

Pro každé dva řádky vypište řádek obsahující jedno celé číslo, které splňuje výše uvedená kritéria.

Příklad vstupu

a1b2c3d4e
zz1yy2xx3ww4vv
abcdgh
aedfhr
abcdefghijklmnopqrstuvwxyz
a0b0c0d0e0f0g0h0i0j0k0l0m0n0o0p0q0r0s0t0u0v0w0x0y0z0
abcdefghijklmnzyxwvutsrqpo
opqrstuvwxyzabcdefghijklmn

Příklad výstupu

4
3
26
14

Shrnutí zadání

Na vstupu dostaneme dva řetězce a musíme zjistit délku nejdelší společné posloupnosti.

Způsob řešení

Tento problém lze vyřešit pomocí dynamického programovaní. Vytvoří se celočíselné dvourozměrné pole o délce prvního slova a šířce druhého slova. K poli přidáme ještě jeden řádek a sloupek. První sloupek a první řádek se vyplní nulami. Index řádku ukazuje na znak v prvním slově a index sloupce ukazuje na znak ve druhém slově. Následující buňky se vyplňují podle následujícího postupu. Buňky se budou vyplňovat po řádcích. Pokud jsou si znaky rovny, vloží se do buňky číslo o jednu hodnotu větší, než je na pozici číslo o jeden sloupek a jeden řádek zpět. Pokud si znaky nejsou rovny, vloží se do buňky číslo, které je o sloupek zpět nebo o řádek níž podle toho, které je větší. Tímto postupem se vyplní celé pole (viz Tabulka 2.1). Výslednou hodnotu najdeme v poslední buňce.

		z	**z**	**1**	**y**	**y**	**2**	**x**	**x**	**3**	**w**	**w**	**4**	**v**	**v**
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
b	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	0	1	1	1	2	2	2	2	2	2	2	2	2
c	0	0	0	1	1	1	2	2	2	2	2	2	2	2	2
3	0	0	0	1	1	1	2	2	2	3	3	3	3	3	3
d	0	0	0	1	1	1	2	2	2	3	3	3	3	3	3
4	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4
e	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4

Tabulka 2.1: Vyplněné pole

Možné problémy

Pokud programátor zná algoritmus, který nejdelší společnou posloupnost vypočítá, není problém úlohu vyřešit.

2.4 Problémy při řešení úloh

Největší problém při řešení úloh se týká anglického jazyka. V zadání bývá spousta detailů, které je potřeba splnit. Pokud ale programátor neovládá anglický jazyk na výborné úrovni, tyto detaily nemusí pochopit a při odevzdávání složitě hledat důvod, proč dané řešení validátor neakceptuje.

Dalším velkým problémem při řešení je časový limit na vyřešení daného problému. Tento limit je nastaven stejně pro všechny programovací jazyky. Z tohoto důvodu je znevýhodněna Java oproti jazyku C/C++, které jsou výrazně rychlejší. Často se proto stává, že algoritmus, který řeší daný problém a je naprogramován v Javě, neprojde na časový limit, ale pokud se naprogramuje v C/C++, tento algoritmus projde.

3 Softwarové aplikace

3.1 Cíle aplikace

Mnoho studentů a zaměstnanců Západočeské univerzity se účastní programátorských soutěží. Tyto soutěže mají společný základ v tom, že je zadaná úloha a je potřeba vymyslet optimální algoritmus, který ji řeší a realizovat ho pomocí zdrojového kódu.

Pro řešení zadaného problému se často využívají základní algoritmy, jako je například prohledávání grafu do hloubky, prohledávání grafu do šířky, dynamické programování atd.. Rozdíl je pouze v zadání úlohy, z které musí soutěžící poznat, jaký algoritmus pro daný problém má použít.

Z tohoto důvodu vznikla tato softwarová aplikace, ve které by měli uživatelé shromažďovat a sdílet vyřešené úlohy s popisem, s jakým algoritmem danou úlohu vyřešili. Uživatelé by díky této aplikaci měli mít možnost z vyřešených úloh poznat, jaký algoritmus mají použít pro daný problém.

Hotové příklady s doprovodným textem by v budoucnu měly sloužit jako pomocný učební materiál pro uživatele zabývající se programováním.

Aplikace by měla fungovat ve dvou režimech. V prvním režimu, by aplikace měla být schopna prohlížet a upravovat všechny úlohy, které jsou v aplikaci uloženy, pokud je aplikace připojena na Internet. V druhém režimu by uživatel měl mít možnost procházek úlohy, které si v prvním režimu uložil bez toho, že by aplikace musela být připojena na Internet. K úlohám by měla být také diskuze, kde by uživatelé mohli diskutovat o dané úloze.

3.2 Použité technologie

Aplikace byla naprogramována pomocí platformy Java SE s programovacím jazykem Java. Pro sestavení aplikace byla použita aplikace Apache Ant.
K uložení dat aplikace využívá relační databázi MySQL.

3.2.1 Platforma Java

Java platforma je sada softwarového vybavení, která slouží pro vývoj a spuštění aplikací, které byly napsány v programovacím jazyce Java. Platforma byla vyrobena a specifikována firmou Sun Microsystems, která byla v roce 2009 koupena společností Oracle.

Základem platformy Javy je Java Virutal Machine(JVM) sloužící pro spuštění Java bytecodu a Java Application Programming Interface(API) obsahující základní knihovny pro vývoj aplikací. Java bytecode dostaneme přeložením zdrojového kódu napsaného v programovacím jazyce Java. Existují i překladače, které jsou schopny vytvořit Java bytecode ze zdrojového kódu, který byl napsaný v jiném programovacím jazyce např. Python. Protože náš program se nejdříve překládá do Java byte kódu a ten se spouští v JVM, může naše aplikace běžet na různých zařízení, na kterých běží JVM.

Platforma je vydávaná ve více verzí, které obsahují potřebné softwarové vybavení pro vývoj v daném prostředí. A to jsou:

- **JavaCard** - vývoj aplikací pro platební a kreditní karty,
- **Java SE** - vývoj aplikací pro domácí počítače,
- **Java EE** - vývoj rozsáhlých podnikových aplikací,
- **Java ME** - vývoj pro mobilní aplikace.

3.2.2 Programovací jazyk Java

Programovací jazyk Java patří mezi objektové orientované programovací jazyky, který byl vyroben firmou Sun Microsystems v roce 1995. Je to hlavní programovací jazyk pro vývoj aplikací na platformě Java.

Java patří mezi nejpoužívanější programovací jazyky na světe. Největší její předností je její multiplatformní nezávislost. Pomocí programovacího jazyka Java můžeme vytvářet aplikace pro různá zařízení od počítačů až po čipové karty.

Výhody jazyka Javy

- Multiplatformní nezávislost
- Jednoduchá syntaxe
- Objektově orientovaný
- Velké množství knihoven

Nevýhody jazyka Javy

- **Pomalý start aplikace** - Tento problém vzniká z důvodu, že zdrojový kód je nejdříve přeložen do Java bytecodu, který je následně spuštěný v JVM.
- **Velká paměťová náročnost** - Následkem použití Garbage collectoru, který se stará v Javě o přidělení a uvolňování paměti.

3.2.3 Apache Ant

Apache Ant je open source softwarová aplikace sloužící pro sestavování softwarových aplikací napsaný převážně v jazyce Java. Protože je Ant kompletně napsán v jazyce Java, jde o aplikaci multiplatformně nezávislou. Pro napsaní scriptu, který Ant zpracovává se používá jazyk XML.

3.2.4 Databáze MySQL

MySQL je relační databázový systém vytvořený firmou MySQL AB a nyní vlastněno firmou Oracle. Jde o multiplatformní databázi, která slouží pro uchovávání dat. Pro komunikaci s databází se využívá dotazovací jazyk SQL.

Jazyk SQL

Jazyk SQL je standardizovaný dotazovací jazyk pro získávání dat z relačních databází. Přestože jde o dotazovací jazyk, můžeme také pomocí SQL příkazu

vytvářet, mazat a upravovat data v databázi. SQL příkazy lze rozdělit do tří skupin.

- **Příkazy pro manipulaci s daty** - Tyto příkazy slouží pro získání a manipulaci dat z databáze. Označují se zkratkou **DML**. Mezi tyto příkazy patří *SELECT*, *UPDATE*, *INSERT*, *DELETE* atd.
- **Příkazy pro definici dat** - Příkazy sloužící pro vytvoření struktury v databázi. Označují se zktratkou **DDL**. Mezi tyto příkazy patří *CREATE*, *DROP* a *ALTER*.
- **Příkazy pro řízení dat** - Příkazy slouží pro nastavení přístupových práv a řízení transakcí. Označují se zkratkou **DCL**. Mezi tyto příkazy patří *REVOKE*, *COMMIT*, *ROLLBACK* atd.

3.3 Použité architektury

Pro vytvoření aplikace byla použita síťová architektura klient-server. Obě aplikace jsou napsány pomocí vícevrstvé architektury.

3.3.1 Síťová architektura klient-server

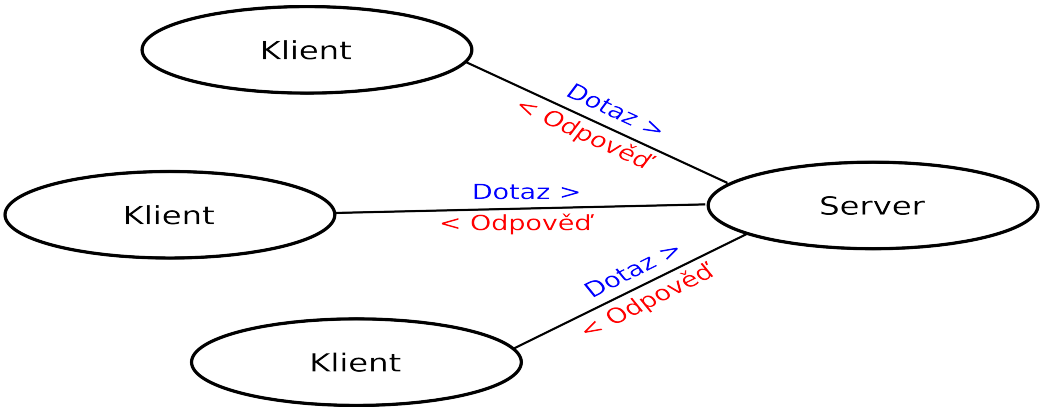
Klient-server je síťová architektura, která rozděluje aplikaci na klienta a server. Tyto dvě aplikace mezi sebou komunikují pomocí počítačové sítě.

Server

Aplikace, která slouží jako server, pracuje pasivně, kdy čeká na dotazy od připojeného klienta. Tento dotaz server zpracuje a odesílá klientovi odpověď (viz Obrázek 3.1).

Klient

Tato aplikace se připojuje k serveru a pracuje aktivně. Po odeslání dotazu serveru klient čeká, až server zašle odpověď (viz Obrázek 3.1).

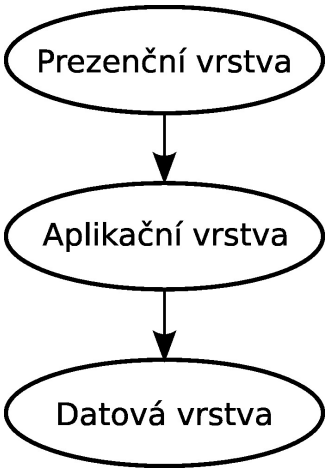


Obrázek 3.1: Klient-server

3.3.2 Vícevrstvá architektura

Vícevrstvá architektura je způsob vývoje aplikace, která rozdělí aplikaci do několika vzájemně nezávislých vrstev, které mezi sebou komunikují podle definovaného rozhraní. Každá z vrstev se může starat o různou část aplikace. Mezi nejznámější vícevrstvou architekturu patří třívrstvá (viz Obrázek 3.2), která rozděluje vrstvy do těchto tří vrstev:

- **Prezenční vrstva** - Prezenční vrstva se stará o komunikaci s uživatelem. Pro komunikaci s uživatelem se nejčastěji používá grafické uživatelské rozhraní.
- **Aplikační vrstva** - Aplikační vrstva se stará o logickou část aplikace.
- **Datová vrstva** - Datová vrstva se stará o načtení a uložení dat. Tato vrstva může využívat např. databázi, soubory atd.



Obrázek 3.2: Třívrstvá architektura

3.4 Programátorská dokumentace

Celá softwarová aplikace je pojmenovaná ACM úložiště. Je rozdělena na dvě aplikace a to ACM úložiště - server a ACM úložiště - klient.

3.4.1 Databáze

Pro uložení dat jsou v databázi vytvořeny čtyři tabulky:

- diskuze,
- uzivatele,
- slozky,
- ulohy.

Tabulka diskuze

V této tabulce se ukládají informace o diskuzích v úlohách. Každý záznam v tabulce musí obsahovat id záznamu, id rodiče, kdo záznam vložil, datum vložení a k jaké úloze záznam patří. Každá úloha, která byla vytvořena, musí

vytvořit pro danou úlohu záznam v diskuzi s rodičem null, který znamená, že daný záznam, nemá rodiče.

Tabulka uzivatele

V tabulce *uzivatele* se ukládají informace o uživateli. Tabulka obsahuje id uživatele, nick, e-mail, heslo, které se při vložení šifruje, datum registrace, potvrzovací číslo a údaj, jestli uživatel potvrdil registraci.

Tabulka slozky

Tabulka *slozky* obsahuje názvy složek, do kterých můžeme přidávat úlohy. Každá složka obsahuje id složky, číslo rodiče, název složky, číslo, které udává, jestli složka byla vymazána, datum vymazání a číslo uživatele, který složku vymazal. Tabulka musí obsahovat jednu složku, která je hlavní. Tato složka má hodnotu rodiče null, která znamená, že žádného rodiče nemá.

Tabulka ulohy

Všechny úlohy se ukládají do tabulky *ulohy*. Tato tabulka obsahuje všechny informace a data o úloze. Každá úloha má id, číslo rodiče, který ukazuje, do které složky daná úloha patří a název úlohy, číslo, které udává, jestli úloha byla vymazána, datum vymazání a číslo uživatele, který úlohu vymazal. Dále jsou uloženy tři soubory s jejich názvy. V těchto třech souborech je uložené zadání, řešení a zdrojový kód, které dané zadání řeší. Jako poslední informace v tabulce je uvedena informace o tom, v jakém programovacím jazyce byl problém vyřešen.

3.4.2 Server

Server využívá třívrstvou architekturu. Prezenční vrstva je dělaná pomocí textového uživatelského rozhraní. Aplikační vrstva se stará o připojení klienta a zpracování požadavků od klienta. Datová vrstva obsahuje třídy pro odeslání e-mailu a třídy pro spolupráci s databází. Datová vrstva také obsahuje třídy, které se odesílají mezi serverem a klientem.

Použité externí knihovny

Aplikace využívá dvě externí knihovny:

- **JavaMail** - Slouží pro připojení k smtp serveru a odeslání e-mailu,
- **MySQL Connectors** - Používá se pro připojení k MySQL databázi a práci sní.

Tyto knihovny se starají o připojení databáze a práci sní a pro připojeni k smtp serveru a odeslání e-mailu.

Konfigurační soubor

Konfigurační soubor se musí jmenovat *config.properties*. Tento soubor musí obsahovat údaje o připojení k smtp serveru, které boudou použity pro odeslání potvrzovacího e-mailu, a přihlašovací údaje do databáze, která bude sloužit pro ukládání dat. Dále musí byt uveden port, na kterém aplikace bude naslouchat.

Konfigurační soubor obsahuje na každé řádce jednu informaci. Informace je uložena ve formátu *proměnná = hodnota*. Všechny proměnné až na proměnnou *domena* jsou povinné. Pokud u proměnné *domena*, je potřeba nastavit více e-mailů, jsou oddělené čárkou. Proměnné, které se musí nastavit až na proměnnou *domena* jsou:

- **smtp_server** - adrese smtp serveru,
- **smtp_ucet** - účet na smtp serveru,
- **smtp_heslo** - heslo k účtu na smtp serveru,
- **databaze_url** - url adresa databáze,
- **databaze_uzivatel** - jméno uživatele v databázi,
- **databaze_heslo** - heslo uživatele v databázi,
- **port** - číslo portu na kterém aplikace bude naslouchat,
- **domena** - doména, kterou musí e-mail obsahovat (nepovinné).

Třída Hlavni

Hlavní třída obsahuje statickou metodu *main*, která se volá při spuštění aplikace. Tato třída nepatří do žádné vrstvy. Třída se stará o načtení konfiguračního souboru, kde se nacházejí údaje o smtp serveru, databázi a portu, na kterém má aplikace naslouchat. Pokud jsou údaje z konfiguračního souboru správně načtené, vytvářejí se dvě instance *PotvrzovaciMail* a *PripojeniMySQL*. Následně se vytváří *ServerSocet*, který v nekonečné smyčce bude čekat na připojení klienta. Pokud je klient připojen, vytvoří se nové vlákno pomocí instance *VlaknoUzivatel*.

Třída VlaknoUzivatel (Aplikační vrstva)

Je to jediná třída, která patří do aplikační vrstvy. Třída se stará o zpracování dotazů od klienta a odeslání odpovědí. Třída obsahuje metody pro zpracování příkazu od nepřihlášeného tak i přihlášeného uživatele. Dále zde také najdeme metodu, která se stará o vygenerování potvrzovacího čísla.

Třída PrispevekDiskuze (Datová vrstva)

Třída obsahuje informace o jednom příspěvku v diskuzi, kde jsou veřejné atributy o id příspěvku, id rodiče, kdo příspěvek napsal, text, datum vložení a id úlohy, kterému příspěvek patří.

Třída Slozka (Datová vrstva)

Tato třída obsahuje informace o jedné složce, kde jsou veřejné atributy, které obsahují id složky, id rodiče a název složky.

Třída UlohyObsah (Datová vrstva)

Třída slouží pro uchovávání základních údajů o úloze v atributech, které jsou veřejné. Údaje o úloze jsou id úlohy, id rodiče a název úlohy. Tato třída neobsahuje soubory k úloze.

Třída Uloha (Datová vrstva)

Datová třída obsahuje všechny údaje o jedné úloze. Uchovává svoje id a id rodiče. Dále třída uchovává název úlohy s názvy souborů. Třída obsahuje soubory se zadáním, řešením a se zdrojovým kódem.

Třída Obsah (Datová vrstva)

Třída slouží pro uchování všech složek a úloh, které obsahuje aplikace. Třída obsahuje *ArrayList* instancí třídy *Slozek* a *ArrayList* instancí třídy *Uloh*.

Třída PotvrzovacíMail (Datová vrstva)

Třída slouží k odeslání potvrzovacího e-mailu uživateli při registraci. Při vytvoření instance je potřeba zadat údaje pro vytvoření spojení se smtp serverem. Pro odeslání potvrzovacího e-mailu má třída metodu *odesliPotvrzovaciMail*, kde je nutné zadat jako parametr e-mail příjemce a potvrzovací číslo.

Třída Prikaz (Datová vrstva)

Třída *Prikaz* patří do datové vrstvy a slouží pro komunikaci mezi serverem a klientem, obsahuje statické číselné konstanty všech příkazů, které se mohou v aplikaci použít. Při odesílání třídy je potřeba nastavit číslo příkazu. Pokud příkaz obsahuje data, jsou uložena ve třídě jako atribut instance *Object*.

Třída PripojeniMySQL (Datová vrstva)

Třída se stará o připojení a práci s databází MySQL. Při vytvoření instance je vytvořené spojení mezi aplikací a databází MySQL. Třída obsahuje spoustu metod pro získávaní a úpravu dat v databázi. Poslední metodou ve třídě je ukončení spojení s databází.

Třída PripojeniTCP (Datová vrstva)

Třída VlaknoSpojeniTCP vytváří TCP spojení mezi klientem a servem. Obsahuje metody pro příjem příkazu od klienta a odeslání odpovědi.

Třída Konzole (Prezenční vrstva)

Třída vytváří textové uživatelské rozhraní, které slouží pro komunikaci s uživatelem pomocí příkazů.

3.4.3 Klient

Klientská část má třívrstvou architekturu. Pro komunikaci s uživatelem je využito grafické uživatelské rozhraní.

Konfigurační soubor

Konfigurační soubor se musí jmenovat *config.properties* a musí obsahovat údaje o adrese serveru, kde běží serverová část aplikace, a portu na kterém tento server naslouchá.

Konfigurační soubor obsahuje na každé řádce jednu informaci. Informace je uložena ve formátu *proměnná = hodnota*. Proměnné, které musíme nastavit jsou:

- **ip** - adresa serveru,
- **port** - číslo portu na kterém naslouchá server.

Použité externí knihovny

Aplikace využívá čtyři externí knihovny:

- **Apache PDFBox** - tisknutí PDF souboru,

- **Java Syntax Highlighter** - zobrazení syntaxe u zdrojového kódu,
- **Pdf-renderer** - zobrazení a manipulaci s PDF souborem,
- **datove-tridy** - knihovna obsahující datové třídy pro komunikaci se serverem.

Tyto knihovny se starají o zobrazení PDF souboru, zvýraznění syntaxe ve zdrojovém kódu a tisk PDF souboru.

Třída Hlavni

Hlavní třída aplikace obsahuje statickou metodu *main* volající při spuštění aplikace. Po spuštění aplikace se načte z konfiguračního souboru adresa serveru a port, na kterém server naslouchá. Po úspěšném načtení se vytváří instance *PripojeniTCP*, která se poté dává jako parametr při vytváření instance *Uzivatel*.

Třída Uzivatel (Aplikační vrstva)

Tato třída se stará o zpracování dotazů, které se odesílají na server a které přišly ze serveru. Třída také obsahuje mnoho metod, které se volají při události v GUI.

Třída Filter (Aplikační vrstva)

Třída *Filter* se stará o to, aby instance třídy *JFileChooser*, která tento filtr používá, mohla vybrat pouze z PDF souborů.

Třída PripojeniTCP (Datová vrstva)

Třída z datové vrstvy vytváří spojení mezi klientem a servem. Třída funguje jako vydavatel. Pro příjem dat je použita třída *ObjectInputStream*, pokud data přijdou, je na to upozorněn předplatitel, kterému se data předají. Pro odeslání dat je vytvořena metoda *odesliData*, která využívá třídu *ObjectOutputStream*.

Třída UlozeniDat (Datová vrstva)

Třída se stará o uložení dat do počítače a jejich načtení zpět. Třída obsahuje statické metody, které se proto využívají.

Třída SouboryKUloham (Datová vrstva)

Třída obsahuje tři *File* proměnné:

- zadani,
- zdrojak,
- reseni.

Ty odkazují na soubory, které slouží pro zobrazení jedné úlohy.

Třída HlavniOkno (Prezenční vrstva)

Hlavní třída prezenční vrstvy se stará o zobrazení hlavního okna aplikace. Okno aplikace obsahuje strom s obsahem, panel se záložkami, které zobrazují vybranou úlohu a diskuzi k ní. Úlohu lze vybrat pomocí stromu. Pro zobrazení v syntaxe u zdrojového kódu třída využívá knihovnu Java Syntax Highlighter.

Třída DiskuzePanel (Prezenční vrstva)

Třída vytváří panel, který zobrazuje diskuzi k vybrané úloze. Tento panel využívá třída *HlavniOkno*.

Třída MujRendererStrom (Prezenční vrstva)

Třída dědí od třídy *DefaultTreeCellRenderer* a upravuje zobrazení ikon instance *JTree*, která danou třídu využívá.

Třída ZobrazeniPdfPanel (Prezenční vrstva)

Prezenční třída starající se o zobrazení PDF souboru v *JPanelu*. Třída využívá knihovnu Pdf-renderer. Tato třída také implementuje metody pro práci s PDF jako je například přesun na další stránku nebo nastavení PDF souboru.

Třída PotvrzeniOkno (Prezenční vrstva)

Třída zobrazuje okno, které slouží pro zadání potvrzovacího čísla od uživatele.

Třída PresunouDialog (Prezenční vrstva)

Třída vytváří dialog, kde uživatel může přesunout úlohu nebo složku ve stromě na jinou pozici.

Třída PrihlaseniOkno (Prezenční vrstva)

Tato třída vytváří okno, kde je možné zadat údaje od uživatele pro přihlášení do aplikace. Okno má také odkazy na registraci a spuštění aplikace offline.

Třída RegistraceOkno (Prezenční vrstva)

Třída vytváří okna, které slouží pro zadání údajů od uživatele pro registraci do aplikace.

Třída StromPanel (Prezenční vrstva)

Třída Strom dědí od třídy JTree a slouží pro vytvoření stromu s obsahem aplikace. Třída obsahuje mnoho metod, které nastavují chování stromu.

Třída TiskDialog (Prezenční vrstva)

Třída zobrazuje dialog, kde si uživatel může vybrat, zda si chce vytisknout zadání úlohy, popis řešení úlohy nebo zdrojový kód, který řeší danou úlohu.

Třída VlozeniUlohyDialog (Prezenční vrstva)

Třída vytváří dialog, který se stará o vložení údajů, které jsou potřeba pro vytvořené nové úlohy.

Třída UpravaUlohyDialog (Prezenční vrstva)

Třída vytváří dialog, který se stará o úpravu údajů, které jsou uloženy u úlohy.

Třída VlozitPrispevekDialog (Prezenční vrstva)

Třída vytváří dialog pro vložení příspěvku do diskuze k úlohám.

Třída VlozitSlozkuDialog (Prezenční vrstva)

Třída vytváří dialog, pro vytvoření nové složky v aplikaci.

Třída ZdrojovyKodPanel (Prezenční vrstva)

Třída se stará o zobrazení zdrojového kódu a zobrazení syntaxe. Tato třída také obsahuje metody pro výměnu zdrojového kódu. Třídu využívá třída *HlavniOkno*.

4 Závěr

V první kapitole mé bakalářské práce jsem se zabýval programovací soutěží PilsProg a veřejným validačním serverem UVa Online Judge, který jsem následně použil jako zdroj pro výběr a vytvoření sady úloh.

Při vytváření sady úloh jsem si procvičil své programátorské schopnosti z hlediska vyřešení problému v co nejlepším čase. Při těchto úlohách jsem použil programátorské strategie, které jsem se naučil ve škole, jako je například rekurzivní programovaní nebo dynamické programování. Při úlohách jsem si také vyzkoušel, že úlohy jsou obvykle nastavené tak, aby se nedaly vyřešit pomocí hrubé síly. Všech třiadvacet úloh, které jsem vyřešil, můžete najít na přiloženém CD. Na tomto CD naleznete originální a přeložené zadání do češtiny. U každé úlohy je také popis možného způsobu řešení se zdrojovým kódem. Všechny úlohy jsou na přiloženém CD rozděleny do tří složek *Lehké*, *Středně obtížné* a *Obtížné*, podle obtížnosti řešení. Ke každé úloze byly také vytvořeny vstupy a výstupy, které taktéž naleznete na přiloženém CD.

V druhé kapitole jsem se zaměřil na vytvoření aplikace, kam by uživatelé mohli shromažďovat a sdílet vyřešené úlohy s popisem, s jakým algoritmem danou úlohu vyřešili. Ke každé úloze je také diskuze, kde by uživatelé mohli diskutovat o dané úloze.

Aplikace dokáže fungovat ve dvou režimech. Online, kdy aplikace je připojena na server a offline, kdy aplikace dokáže fungovat bez toho, že by byla připojena na server. Online režimu aplikace může provádět všechny operace jako je například vložení nové úlohy, vložení příspěvku, uložení úlohy atd.. Offline režimu můžeme pouze procházet a prohlížet si úlohy, které jsme si v online režimu uložili a tisknout jejich části.

Pro naprogramování aplikace jsem použil platformu Java s programovacím jazykem Java. Při vytváření aplikace jsem získal zkušenosti, jak naprogramovat aplikace, které mezi sebou komunikují pomocí počítačoví sítě. Protože všechna data jsem si ukládal do databáze MySQL, procvičil jsem si také práci s ní a s tím, jak k datům přistupovat z Javy. Pro naprogramování této aplikace jsem také využil externí knihovny, které jsem vyhledával na internetu a s jejichž funkcí a použití jsem se poté seznamoval. Jako poslední jsem použil aplikaci Apache Ant pro sestavení aplikace.

Při vytváření bakalářské práce jsem se naučil mnoho nových věcí, které,

věřím, využiji v navazujícím studiu a následně ve své budoucí programátorské práci.

Seznam zkratek

UVa - University of Valladolid
ACM - Association for Computing Machinery
ACM-ICPC - ACM International Collegiate Programming Contest
PDF - Portable Document Format
JDK - Java Development Kit
JVM - Java virtual machine
API - Application Programming Interface
Java SE - Java Platform, Standard Edition
Java ME - Java Platform, Enterprise Edition
Java EE - Java Platform, Micro Edition
DML - Data Manipulation Language
DDL - Data Definition Language
DCL - Data Control Language
SMTP - Simple Mail Transfer Protocol
GUI - Graphical User Interface

Literatura

[1] Programátorská soutěž PilsProg
URL:<https://pilsprog.fav.zcu.cz/index.php/aktuality>
[citováno: 4.5.2014]

[2] Validátor UVa Online Judge
URL:<http://uva.onlinejudge.org/>
[citováno: 4.5.2014]

[3] Steven S. Skiena, Miguel A. Revilla. *Programming Challenges.*
URL:<http://acm.cs.buap.mx/downloads/Programming_Challenges.pdf>
[citováno: 4.5.2014]

[4] Doc. Ing. Pavel Herout, Ph.D. *Učebnice jazyka Java.* 2007.
ISBN 978-80-7232-323-4

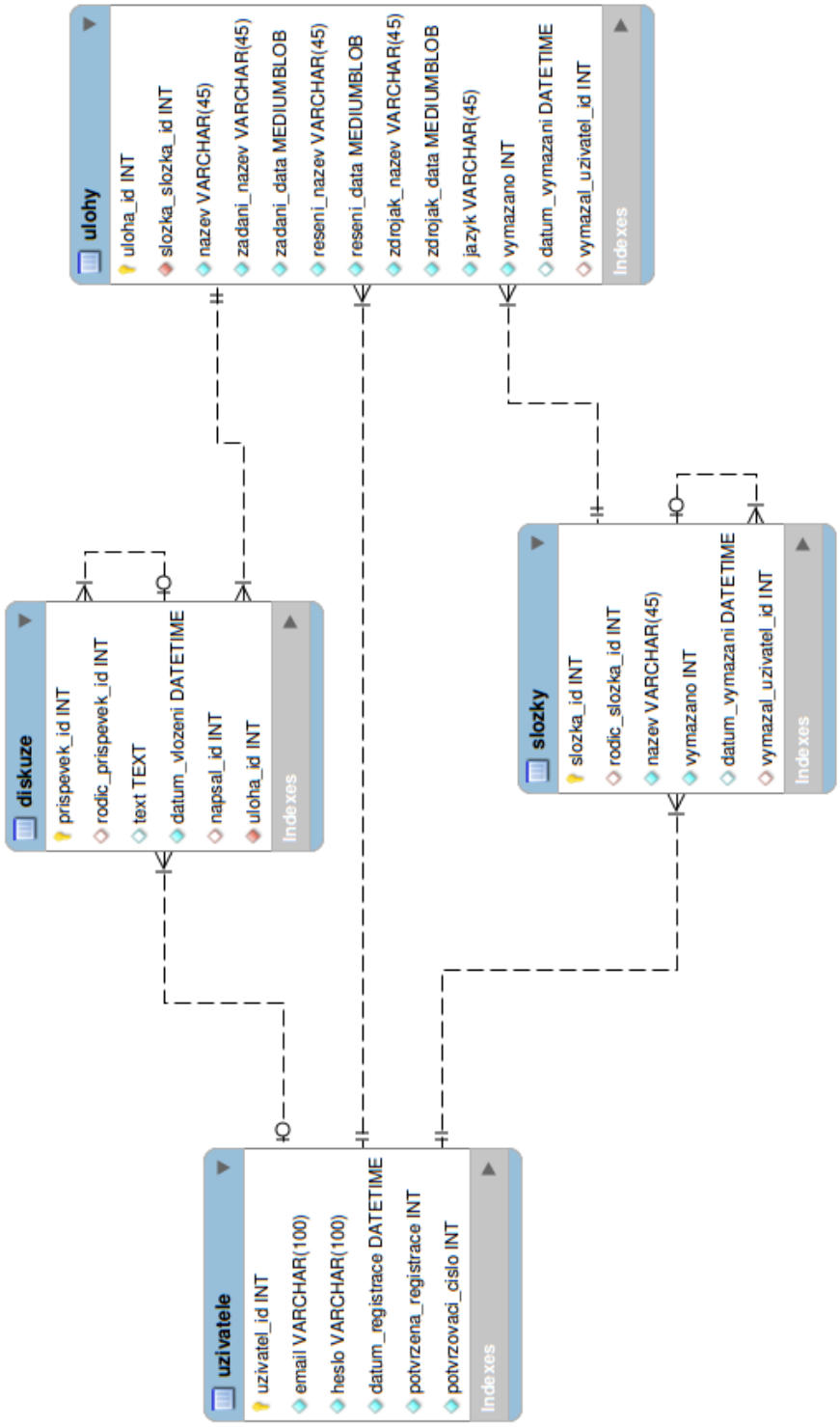
[5] Diskuze k UVa Online Judge validátoru
URL:<http://online-judge.uva.es/board/>
[citováno: 4.5.2014]

[6] Robert Sedgewick a Kevin Wayne *Algorithms (4th Edition).* 2011.
ISBN 978-0-321-57351-3

Přílohy

Seznam příloh

Schéma 1 ERA model databáze



Uživatelská příručka

Pro přeložení a běh obou aplikací je nutné mít nainstalovaný JDK 1.7 nebo vyšší.

Server

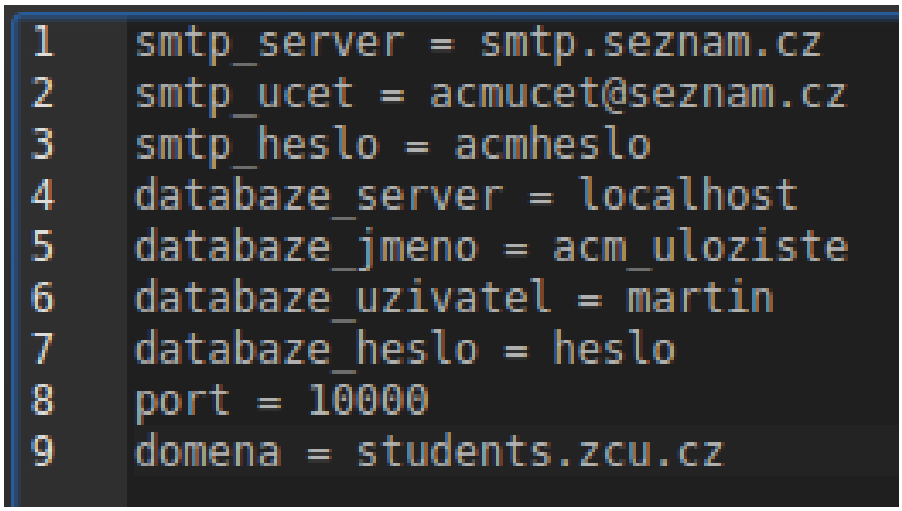
Přeložení

Aplikace se přeloží pomocí příkazu *ant*, ve složce server. Tímto příkazem vznikne spustitelný soubor *ACM úložiště - server.jar*.

Před spuštěním

Před spuštěním aplikace je nutné nastavit konfigurační soubor, který se musí jmenovat *config.properties* a musí být ve stejné složce jako *ACM úložiště - server.jar*. Konfigurační soubor obsahuje na každé řádce jednu informaci. Informace je uložena ve formátu *proměnná = hodnota* (viz Obrázek 1). Všechny proměnné až na proměnnou *domena* je nutné nastavit. Pokud se proměnná *domena* nenastaví, jsou povoleny pro registraci všechny e-maily. Pokud je potřeba u proměnné *domena* nastavit více e-mailů, jsou oddělené čárkou.

- **smtp_server** - adrese smtp serveru,
- **smtp_ucet** - účet na smtp serveru,
- **smtp_heslo** - heslo k účtu na smtp serveru,
- **databaze_url** - url adresa databáze,
- **databaze_uzivatel** - jméno uživatele v databázi,
- **databaze_heslo** - heslo uživatele v databázi,
- **port** - číslo portu, na kterém aplikace bude naslouchat,
- **domena** - doména, kterou musí e-mail obsahovat při registraci.



Obrázek 1: Konfigurační soubor

Spuštění

Spuštění probíhá pomocí příkazu *java -jar 'ACM úložiště - server.jar'*. Poté je možno zadat dva příkazy. Příkaz *vytvor databazi*, vytvoří novou databázi a příkaz *konec*, slouží pro ukončení aplikace.

Klient

Aplikace se přeloží pomocí příkazu *ant*, ve složce klient. Tímto příkazem vznikne spustitelný soubor *ACM úložiště - klient.jar*.

Před spuštěním

Před spuštěním aplikace je nutné nastavit konfigurační soubor, který se musí jmenovat *config.properties* a musí být ve stejné složce jako *ACM úložiště - klient.jar*. Konfigurační soubor obsahuje na každé řádce jednu informaci. Informace je uložena ve formátu *proměnná = hodnota*. Proměnné, které musíme nastavit jsou:

- **ip** - adresa serveru,

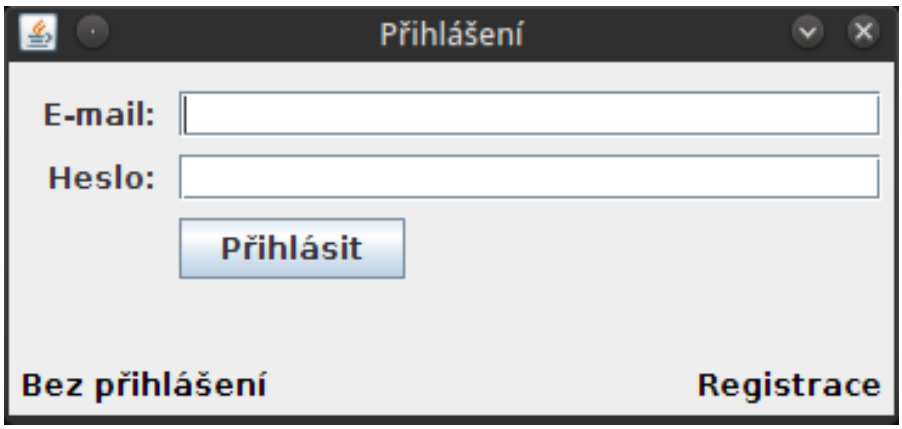
- **port** - číslo portu na kterém naslouchá server.

Spuštění

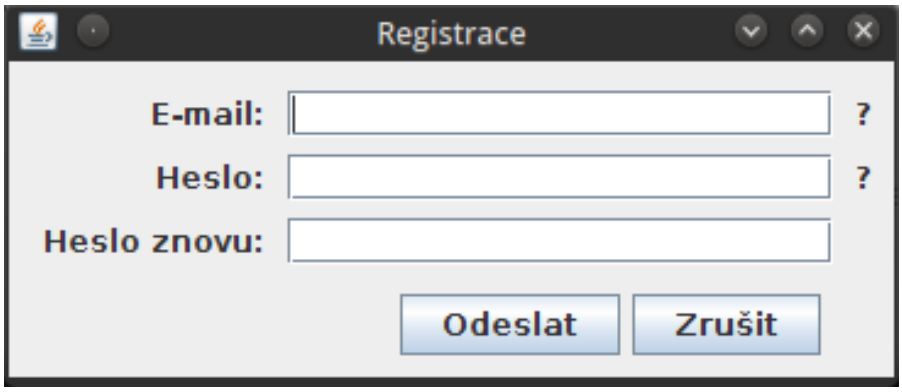
Spuštění aplikace probíhá pomocí příkazu *java -jar 'ACM úložiště - klient.jar'* nebo spuštění souboru *ACM úložiště - klient.jar*.

Přihlašovací okno a registrace

Po spuštění aplikace se nám spustí okno pro přihlášení (viz Obrázek 2). V okně můžeme vyplnit e-mail s heslem a tím se přihlásit do aplikace, která funguje online. To znamená, že je připojena na server. Pokud uživatel nemá přihlašovací údaje, je nejdříve potřeba se zaregistrovat. Do registrace se dostaneme kliknutím na odkaz *registrovat*. Po kliknutí na odkaz *registrovat* se nám spustí okno (viz Obrázek 3), kde je nutné vyplnit e-mail, který musí mít doménu, která je určena serverem, a heslo, které musí být delší jak 5 znaků. Tyto informace můžeme také zjistit, když najedeme myší na otazník. Po registraci přijde uživateli e-mail s číslem, které je potřeba zadat při vyžádání. Zadání čísla se musí provést do 24 hodin, nebo se bude muset znova provést registrace. Poté je možné se do aplikace přihlásit. Na okně pro přihlášení můžeme také kliknout na odkaz *Bez přihlášení*, která spustí aplikaci v offline režimu. To znamená, bez připojení na server.



Obrázek 2: Okno pro přihlášení



Obrázek 3: Okno pro registraci

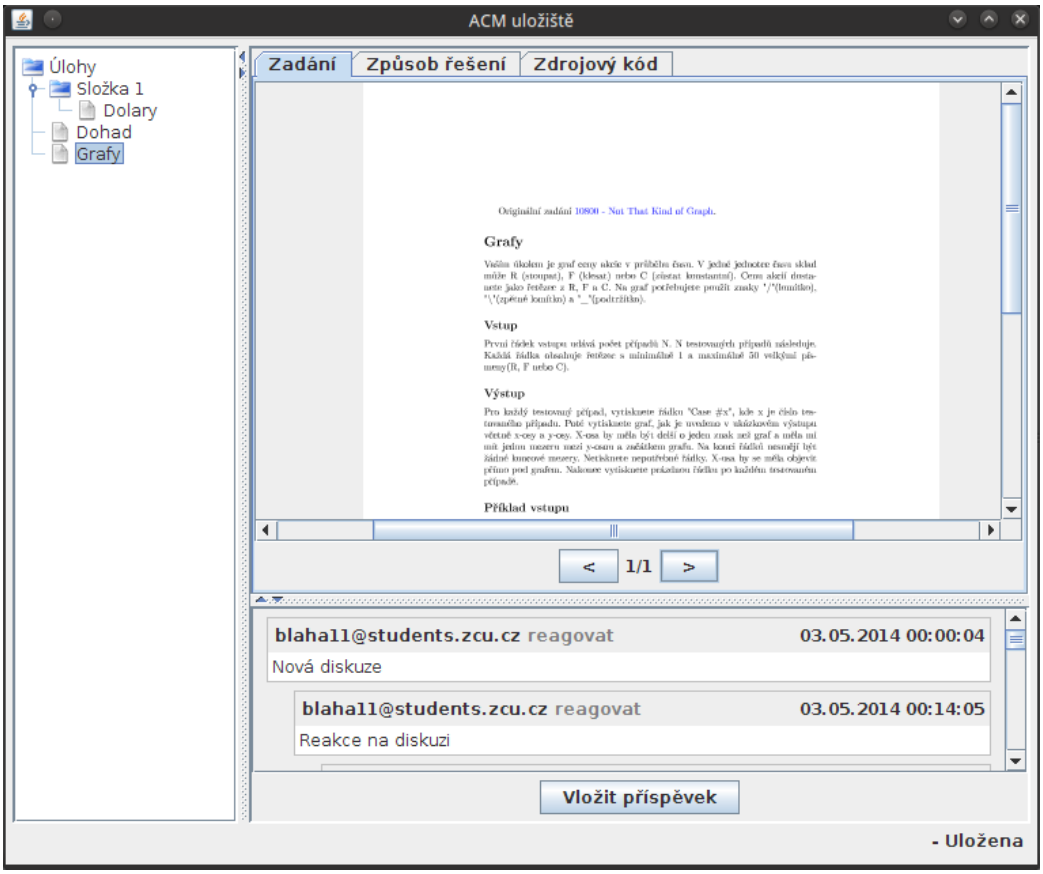
Hlavní okno s přihlášením

Hlavní okno aplikace je rozděleno na tři části (viz Obrázek 4). V pravé části aplikace je strom, který obsahuje složky a názvy úloh. Pravá strana je rozdělena vodorovně na dvě části. Horní část má tři záložky a to *Zadání*, *Řešení* a *Zdrojový kód*, kde se zobrazují části úlohy. V dolní části se zobrazuje diskuze k této úloze.

Ve stromě lze vybrat úlohu pomocí toho, že na ní klikneme levým tlačítkem myši. Po kliknutí pravým tlačítkem myši na složku nebo úlohu se zobrazí menu. U složky lze vybrat, jestli danou složku chceme přejmenovat, odstranit nebo do ní vložit novou složku či novou úlohu. U úlohy můžeme vybrat, jestli chceme danou úlohu přejmenovat, upravit nebo odstranit. U úlohy můžeme také vybrat, že danou úlohu chceme vytisknou. Po kliknutí na tisk je poté potřeba vybrat, jakou část úlohy se má vytisknout.

U každé úlohy, můžeme přidávat nový příspěvek pomocí tlačítka *Vlož příspěvek* nebo reagovat na příspěvek jiného uživatele tak, že klikneme u jeho e-mailu na odkaz *reagovat*. Při vložení nového příspěvku nebo reakci na příspěvek se zobrazí dialog, kam daný příspěvek napíšeme.

Hlavní okno má také v dolním pravém rohu napsáno, jestli daná úloha je uložena nebo ne. Před tímto popisem je znak + nebo -, při jehož kliknutí uložíme nebo odstraníme danou úlohu.



Obrázek 4: Hlavní okno aplikace s vybranou úlohou

Hlavní okno bez přihlášení

Hlavní okno v režimu offline vypadá stejně jako v režimu online. Je omezena pouze jeho funkčnost, kdy lze pouze prohlížet úlohy, které jsme si v online režimu uložili. V tomto režimu nelze upravovat žádná data, pouze prohlížet a tisknout části úloh.