Úlohy k tématům z předmětu ALG1

Pole znaků, třída `String`. Regulární výrazy. Použití dynamického pole `ArrayList`.

Úlohy

Pole znaků

1. Vyzkoušejte třídu `Character` pro práci se znaky (testování znaků apod.). Pole znaků a `String` – převody.
2. Zapište program, který postupně vypíše všechny znaky, které lze uchovat v proměnné typu `char`. Vypisujte po blocích 256 znaků, 64 znaků na jeden řádek výstupu, mezi jednotlivými bloky vynechte jeden řádek. Před každým blokem vypište index/číslo bloku.

Třída (typ) `String`, textové řetězce

3. Otestujte metody `next()` a `nextLine()` třídy `Scanner`. Testujte i případy, kdy se v jednom programu kombinuje (střídá) použití obou metod.
4. Zapište program, který bude načítat a testovat textové řetězce do zadání prázdné řádky. Program má zjistit, zda jednotlivé zadané textové řetězce obsahují palindrom (symetrický textový řetězec). Pro realizaci vlastního testu jednoho textového řetězce zapište vlastní metodu, využijte `charAt()`. Program modifikujte tak, aby při testování byly ignorované všechny (b) počáteční a koncové mezery (c) všechny mezery, které se v řetězci nachází. Využijte vhodné prostředky třídy `String`.
5. Zapište program, který bude opakovaně načítat textové řetězce do zadání prázdné řádky (prázdného textového řetězce). Pro každý textový řetězec má program určit nejdelší podřetězec s neopakujícími se znaky.
6. Zapište program, který bude opakovaně načítat dvojici textových řetězců do zadání prázdné řádky namísto prvního z nich. Program má pro každou dvojici textových řetězců určit výskyt prvního řetězce v druhém řetězci (popřípadě všechny výskyty). Pozicí výskytu je myšlen index prvního znaku výskytu podřetězce v řetězci. Implementujte vlastní prostředky (metodu/metody) pro vyhledání podřetězce v řetězci, procházejte po znacích. Na internetu lze vyhledat i různé algoritmy pro řešení této úlohy.
7. Zapište program, který bude opakovaně načítat textové řetězce do zadání prázdné řádky (prázdného textového řetězce). Pro každý zadaný textový řetězec má program určit, zda se jedná o platnou cestu k souboru nebo adresáři. Pro tyto účely definujme platnou cestu jako textový řetězec, který může začínat názvem disku a dvojtečkou, textový řetězec pak může obsahovat pouze oddělovače vnořené úrovně (lomítka), písmena, číslice, mezery, tečky, podtržítka, pomlčky.
8. Zapište program, který bude načítat a analyzovat textové řetězce představující cestu k souboru (bez ohledu, zda soubor na disku existuje, či nikoli) do zadání prázdné řádky. Program má ze zadané cesty zjistit umístění (adresář, nebo disk a adresář), název souboru a příponu souboru. Textové řetězce načítat metodou `nextLine()`.
9. Vytvořte vhodné prostředky (metody v samostatné knihovní třídě) a následně zapište program, který bude realizovat převod čísel mezi jednotlivými soustavami o základu 2–16 . Jaké prostředky má pro tyto účely Java API – viz obalové třídy nebo třída `Formatter` a formátovaní celých čísel (použití v metodě `format()`).
10. Zapište program, který bude testovat, zda zadaný textový řetězec je platný časový/ datumový údaj apod. Pro testování použijte regulární výraz/výrazy v kombinaci s vhodnými prostředky třídy `String`. Program má umožnit opakované testování uživatelem zadaných textových řetězců do zadání prázdného textového řetězce.
11. Zapište vlastní metody pro separaci jednotlivých slov z textového řetězce – respektive rozdělení textového řetězce na jednotlivá slova. Metoda/y mohou vracet slova v datové struktuře typu `ArrayList<String>` nebo `String[]`. Různé metody mají respektovat následující definice slova:
    - Slovo je sekvence znaků oddělená od dalšího slova jedním nebo více mezerovými znaky (mezery, tabulátory, znaky sekvence konce řádky – znaky `Character.isWhitespace()`).

- Slovo je sekvence písmen oddělená od jiného slova libovolným nepísmenným znakem.
- Slovo je sekvence znaků oddělená od dalšího slova předem zadaným oddělovačem, kde oddělovač je textový řetězec minimální délky 1.

Použijte různé implementace: vlastní algoritmus, který bude procházet daný textový řetězec po znacích a vytvářet „jednotlivá“ dílčí slova; metodu `split()` ze třídy `String` v kombinaci s vhodným regulárním výrazem; vytvoření instance `Scanner` pro daný textový řetězec, nastavení separátoru `useSeparator()` a následná separace slov metodou `next()`; využití prostředků třídy `StringTokenizer`.

12. Vytvořte knihovní třídu, tj. třídu se statickými metodami, které ze zadaného textového řetězce od zadané pozice načtou další slovo. Dále vytvořte metody, které pro zadaný textový řetězec poskytnou pole slov (textových řetězců) obdoba metody `split()`. Porovnejte vlastní metody s možnostmi třídy `StringTokenizer`.
13. Zapište program, který umožní šifrování a dešifrování textové zprávy zadané uživatelem ze standardního vstupu. Realizujte šifrování, které jednotlivá slova v textové zprávě převrátí. Na výstupu vypište celou zprávu bez jakéhokoli členění do slov, jako jednolitý textový řetězec bez mezer. Slovem chápejme libovolnou sekvenci znaků oddělenou od jiného slova mezerou. Variantně lze použít různé definice slova a různé požadavky na podobu výsledného řetězce – např. slovem můžeme chápat libovolnou sekvenci pouze písmenných znaků oddělenou od jiného slova libovolným nepísmenným znakem.
14. Zapište program, který načte textová řetězce do pole a následně zajistí uspořádání tohoto pole. Realizujte vlastní algoritmus třídění/uspořádání pole. V první variantě realizujte alfanumerické uspořádání (pro porovnání dvou řetězců použijete metodu `compareTo()`) ze třídy `String`. V další variantě implementujte vlastní metodu pro porovnání řetězců, která bude porovnávat dva textové řetězce primárně dle délky sekundárně (při shodě délky) pak alfanumericky.
15. Zapište program, který načte textovou zprávu od uživatele (předpokládá se, že se bude jednat o text bez diakritiky) a umožní její zašifrování pomocí jednoduché transformace zadané jedním písmenem anglické abecedy. Pro šifrování vytvořte samostatnou metodu. Při šifrování: šifrujeme pouze písmena anglické abecedy, všechna písmena jsou převedena na velká a podrobena příslušné transformaci (transformace je zadaná jediným písmenem: ‘A‘ představuje nulový posun, ‚B‘ posun o jedno písmeno atd., je třeba realizovat cyklický posun v rámci uspořádané řady písmen anglické abecedy). Zašifrovanou zprávu vypište pouze jako sled velkých písmen anglické abecedy. Alternativně program může nabízet formou menu jak šifrování, tak dešifrování zadané zprávy.
16. Zapište program, který bude načítat standardní vstup do zadání prázdné řádky. Program má provést reverzi celého zadaného vstupu a převrácený textový řetězec vypsat až po zadání celého vstupu v původním členění do řádků, variantně v členění do řádků s pevně daným počtem znaků (zavedení vhodné konstanty v programu).
17. Použijte prostředky třídy `StringBuilder` pro postupné vytváření delšího textového řetězce.