Zadania z programowania w języku Java dla II roku Informatyki.

dr Agnieszka Zbrzezny

1 Kolekcje

1. Napisz program wykorzystujący mapę, w której klucze i wartości są łańcuchami znaków: klucz to nazwisko studenta, a wartość to ocena z egzaminu. Program ma umożliwiać dodawanie i usuwanie studentów, zmianę oceny wybranego studenta oraz wypisanie listy studentów wraz ze stopniami. Lista powinna być posortowana według nazwisk i sformatowana w następujący sposób.

   ```
   Carl: db+
   Joe: db
   Susan: bdb
   ```

2. Zmodyfikuj poprzednie zadanie tak, aby kluczami w mapie były obiekty klasy `Student` z polami `imię`, `nazwisko` oraz `id`. Pole `id` to unikalny identyfikator będący liczba całkowitą. W przypadku zmiany oceny oraz usuwania, wyszukiwanie powinno odbywać się poprzez identyfikator. Wypisana lista powinna być posortowana według nazwisk. Jeżeli dwaj studenci mają to samo nazwisko, to w dalszej kolejności należy uwzględnić imię. Jeżeli imiona są identyczne, to kolejnym kryterium sortowania ma być identyfikator. **Wskazówka**: Użyj dwóch map.

3. Postaraj się znaleźć dwa słowa w dużym pliku mające tę samą wartość funkcji mieszającej (`hashCode`). Użyj mapy `map<Integer, HashSet<String>>`. Po przeczytaniu kolejnego słowa oblicz wartość `h` jego funkcji mieszającej i umieść to słowo w zbiorze, którego kluczem jest `h`. Po zakończeniu wczytywania wszystkich słów przeprowadź iterację po kluczach mapy i wypisz wszystkie zbiory, których liczebność jest większa od 1.

4. Użyj stosu (obiektu klasy `Stack<String>`, aby odwrócić słowa w zdaniu. Czytaj kolejne słowa, aż do napotkania słowa, które kończy się kropką, umieszczając je na stosie. Następnie wypisz kolejne słowa zdejmując je ze stosu. Zakończ program, gdy na wejściu nie ma już żadnych słów. Przykładowo, dla wejścia postaci

   ```
   Ala ma kota. Jej kot lubi myszy.
   ```

   program powinien wypisać

   ```
   Kota ma ala. Myszy lubi kot jej.
   ```

   Zadbaj o dużą literę na początku i o kropkę na końcu każdego zmienionego zdania.

5. Wczytaj nieujemną liczbę całkowitą i podziel ją na poszczególne cyfry. Przykładowo, po wczytaniu liczby 2015 program powinien wypisać: 2 0 1 5. Ostatnią cyfrę liczby `n` daje wyrażenie `n % 10`. Ponieważ w ten sposób uzyskamy cyfry w odwrotnej kolejności, nalezy je najpierw umieścić na stosie, a następnie wypisać kolejne cyfry umieszczone na stosie zdejmując je ze stosu.

2 Ciekawe zadania z tematów nie na egzamin: Strumienie wejścia-wyjścia: pliki tekstowe

1. Dla każdego z poniższych punktów napisz funkcję, która pobiera jako argumenty łańcuch znaków będący nazwą pliku, próbuje otworzyć ten plik do odczytu, a następnie czyta z tego pliku kolejne linie. Po przeczytaniu wszystkich linii funkcja zamyka plik. Jako swój wynik funkcja zwraca:

   (a) długość najdłuższej linii z tego pliku.

   (b) najdłuższą linię z tego pliku (jeżeli jest więcej linii o tej samej długości, to funkcja zwraca pierwszą z tych linii.

W przypadku, gdy pliku nie udało się otworzyć, pierwsza funkcja powinna zwrócić liczbę `-1`, a druga powinna zwrócić wartość **null**.

2. Napisz program, który łączy zawartość kilku plików w jeden plik. Przykładowo,

   ```
   $ java ConcatenateFiles rozdz.txt rozdz2.txt rozdz3.txt książka.txt
   ```

   tworzy długi plik `książka.txt`, który zawiera zawartość plików `rozdz1.txt`, `rozdz2.txt` i `rozdz3`.txt. Plik docelowy jest zawsze ostatnim plikiem podanym w wierszu poleceń.

3. Napisz program, który wczytuje plik tekstowy linia po linii i zapisuje przeczytane linie w pliku wyjściowym poprzedzając je numerem linii. O nazwy plików należy zapytać użytkownika programu na początku programu.

4. Szyfr Cezara jest jedną z najprostszych technik szyfrowania. Jest to rodzaj szyfru podstawieniowego, w którym każda litera tekstu niezaszyfrowanego zastępowana jest inną, oddaloną od niej o stałą liczbę pozycji w alfabecie, literą, przy czym kierunek zamiany musi być zachowany. Nie rozróżnia się przy tym liter dużych i małych. Szyfr Cezara można bardzo łatwo złamać.

   Lepszym pomysłem jest użycie jako klucza *słowa* zamiast *liczby*. Załóżmy, że kluczem jest słowo MATEMATYKA. Z tego klucza usuwamy duplikaty otrzymując słowo MATEYK. Teraz dołączamy kolejne litery alfabetu w odwrotnej kolejności. A zatem poszczególne litery będą zaszyfrowane w następujący sposób:

A	Ą	B	C	Ć	D	E	Ę	F	G	H	I	J	K	L	Ł	M	N	Ń	O	Ó	P	R	S	Ś	T	U	W	Y	Z	Ź	Ż
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
M	A	T	E	Y	K	Ż	Ź	Z	W	U	Ś	S	R	P	Ó	O	Ń	N	Ł	L	J	I	H	G	F	Ę	D	Ć	C	B	Ą

   Napisz program, który szyfruje i odszyfrowuje plik za pomocą tego szyfru. Słowo będące kluczem określa się za pomocą opcji `-k` w wierszu poleceń. Opcja `-d` w wiersza poleceń określa operację deszyfrowania. Przykładowo:

   ```
   java Szyfrator -d -k MATEMATYKA tajny.txt jawny.txt
   ```

   odszyfrowuje plik `tajny.txt` za pomocą słowa kluczowego MATEMATYKA i zapisuje wynik w pliku `jawny.txt`. Nie podanie słowa kluczowego jest traktowane jako błąd.

   Uwaga: Przetwarzaj plik linia po linii korzystając z metody `readLine` z klasy `BufferedReader`. Przetworzone linie zapisuj korzystając z metody `println` z klasy `PrintWriter`. Szczegóły znajdziesz w podrozdziale **Strumienie tekstowe** z przykładowego rozdziału udostępnionego przez wydawnictwo Helion (link podany na slajdach).

5. Napisz program `Reverse.java`, który wczytuje wszystkie linie z pliku i zapisuje je w odwrotnej kolejności w innym pliku.

6. Napisz program `ReverseEachLine.java`, który zamienia każdą linię w pliku jej rewersem. Odwrócone linie zapisuj w innym pliku. Do odwracania wczytanych linii możesz użyć metody `reverse` z klasy `StringBuffer`.

3 Ciekawe zadania z tematów nie na egzamin: Strumienie wejścia-wyjścia: pliki binarne

1. Wiadomo, że pierwszych 8 bajtów pliku w formacie `PNG` ma następujące wartośći `137, 80, 78, 71, 13, 10, 26, 10`. Napisz program `IsPng.py`, który sprawdza czy plik podany jako jego argument jest obrazem w formacie `PNG`.

2. Uogólnij program z poprzedniego zadania, tak aby można mu było podać w linii wywołania programu więcej plików.

3. Napisz program, który otwiera plik binarny (o nazwie podanej w wierszu poleceń) i wypisuje kolejne występujące w tym pliku znaki ASCII, tj. bajty o wartościach od 32 do 126. Wypisuj znak nowej linii po każdych 64 znakach. Sprawdź jak działa Twój program dla plików typu `.doc` oraz `.class`

4. Napisz program, a w nim dwuargumentową statyczną metodę `szyfruj`, której pierwszy argument `buffer` jest obiektem typu `byte[]`, a drugi argment `mask` jest liczbą całkowitą z przedziału `[0..255]`. Funkcja ta modyfikuje tablicę `buffer`, poprzez zastosowanie operatora alternatywy rozłącznej do każdego jej elementu oraz liczby `mask`. W metodzie `main` przetestuj poprawność działania funkcji `szyfruj`.

5. Napisz program `SzyfrXor.py` składający się z funkcji z metody `main` oraz z metody `szyfruj` z poprzedniego zadania.

   Metoda `main` sprawdza czy program został wywołany z trzema argumentami. Jeżeli nie, to wypisuje odpowiedni komunikat i kończy wykonanie programu. Następnie próbuje otworzyć w trybie binarnym do odczytu plik o nazwie podanej jako pierwszy argument programu oraz próbuje otworzyć w trybie binarnym do zapisu plik o nazwie podanej jako drugi argument programu. W przypadku niepowodzenia wypisuje odpowiedni komunikat i kończy wykonanie programu. Ponadto, jeżeli trzeci argument nie jest liczbą z przedziału `[0..255]`, to wypisuje odpowiedni komunikat i kończy wykonanie programu.

   W przeciwnym przypadku metoda `main` w pętli czyta z pliku wejściowego kolejne porcje danych o wielkości 32 bajtów, wywołuje metodę `szyfruj` dla kolejnej porcji oraz liczby podanej jako trzeci argument programu, po czym zapisuje wynik zwrócony przez funkcję `szyfruj` do pliku wyjściowego.

6. Rozważmy następujące wywołania programu `SzyfrXor`:

   ```
   java SzyfrXor /etc/passwd zaszyfrowany.txt liczba
   java SzyfrXor zaszyfrowany.txt odszyfrowany.txt liczba
   ```

   w których `liczba` oznacza liczbę całkowitą z przedziału `[0..255]`.

   Pliki `/etc/passwd` oraz `odszyfrowany.txt` powinny mieć taką samą zawartość, o ile w obu wywołaniach użyto tej samej wartości argumentu `liczba`. Wyjaśnij dlaczego.