6.

Zadanie 6.1 Policz wybrane słowo

Napisz program, który obliczy ile razy dane słowo występuje w pliku (np. Tadeusz w pliku pan-tadeusz.txt).

Podpowiedź:

Najlepiej użyć klasy Scanner bezpośrednio na pliku i użyć odpowiedniego wyrażenia regularnego opisującego separator pomiędzy słowami. Najlepszy sposób to:

```
Scanner sc = new Scanner(new File("pan-tadeusz.txt"));
sc.useDelimiter("[^\\p{L}]+");
while(sc.hasNext()) {
     String slowo = sc.next();
     // TODO co zrobić z każdym słowem
}
```

Wersja interaktywna

Użyj JOptionPane do pobrania szukanego słowa i wyświetlenia wyniku.

Użyj JFileChooser do wybrania pliku z dysku (showOpenDialog i inne kroki - poszukajcie "w internetach" jak się tego używa...).

Zadanie 6.2 Policz wszystkie słowa

Napisz w Javie program, który czyta plik tekstowy i wylicza oraz wypisuje bez powtórzeń wszystkie słowa występujące w pliku wraz z informacją ile razy dane słowo występuje. Na przykład w ten sposób:

```
Zosia  -> 34
Asesor -> 35
dwóch  -> 35
Tadeusz -> 107
```

Podpowiedzi:

Słownik (`Map<String, Integer>`) będzie najlepszy do zbierania informacji.

Dalsze rozszerzenia (opcjonalnie kto da radę, jako osobne programy):

- Posortuj wypisywane słowa alfabetycznie.
- Posortuj wypisywane dane według policzonej ilości tych słów w pliku (to trudniejsze).

Zadanie 6.3 Dane z pliku "sprzedaż"

W pliku sprzedaz.csv znajdują się dane opisujące sprzedaż różnych sklepów - w każdej linii jest jedna transakcja. Cena jest ceną jednostkową, a w osobnej kolumnie jest podana ilość sztuk. Dopiero ich iloczyn jest wartością transakcji – trzeba to liczyć w programie.

Napisz programy, które czytają dane z tego pliku i obliczają:

1. Sumę wartości wszystkich transakcji z całego pliku.
2. Ilość, sumę oraz minimalną, maksymalną i średnią wartość transakcji z wybranego miasta (niech program pyta o nazwę miasta na początku).
3. Dla każdego miasta występującego w pliku – sumę transakcji z tego miasta (wypisać bez powtórzeń), czyli zastosować schemat grupowania.

Zadanie 6.4 Klasa dla ułamków

Stwórz klasę, której obiekty reprezentują liczby wymierne w postaci ułamkowej. Klasa powinna implementować operacje na ułamkach.

Z kilku możliwych podejść najbardziej rekomenduję utworzenie klasy niemutowalnej, zgodnie z wzorcem „value object”: brak setterów, operacje matematyczne zawsze zwracają nowy obiekt w wyniku, a nie modyfikują bieżącego; metody porównujące biorą pod uwagę wartość, a nie tożsamość obiektu. Przykładami takich klas są BigDecimal oraz LocalDate.

Zgodnie z najnowszymi wzorcami ukryj konstruktor jako prywatny, a udostępnij zestaw metod statycznych, za pomocą których pobiera się obiekty (jak w LocalDate). Nie dopuszczaj do sytuacji, aby w mianowniku znalazło się zero – wyrzucaj wyjątek IllegalArgument albo IllegalState. Zalecam taką „normalizację” tworzonych ułamków, aby mianownik był zawsze dodatni, a tylko licznik mógł być ujemny. Operacje arytmetyczne powinny także zwracać wynik w postaci maksymalnie skróconej.

Proponowane publiczne API klasy (tym razem piszę po angielsku, możecie przetłumaczyć):

- static Fraction of(int nom, int denom) (mogą być longi jeśli ktoś preferuje...)
- static Fraction of(int i) // na podstawie liczby całkowitej, mianownik = 1
- static final Fraction ZERO, ONE, HALF, ... – kilka stałych, podobnie jak w BigInteger
- int getNominator(), int getDenominator()
- String toString()
- equals + hashCode – w oparciu o wartości,

  do decyzji autora czy 3/4 i 6/8 mają być równe czy różne...
- int compareTo(Fraction other) – klasa powinna implementować interfejs Comparable
- Fraction shortened() – zwraca ułamek o tej samej wartości, skrócony w miarę możliwości (o największy wspólny dzielnik licznika i mianownika)
- Fraction reciprocal() – zwraca odwrotność ułamka
- Fraction neg() – zwraca liczbę ze zmienionym znakiem (plus/minus)
- Fraction add(Fraction) – suma
- Fraction sub(Fraction) – różnica
- Fraction mul(Fraction) – iloczyn
- Fraction div(Fraction) – iloraz
- double getAsDouble() – zwraca wartość tego ułamka jako wartość typu double

(być może w przybliżeniu)