Programowanie w JAVA
Lab. 3 – Algorytmy

Każde zadanie zaimplementuj w osobnej paczce. Zadbaj o przejrzystość kodu (w metodzie main powinno być tylko uruchomienie zadania, zadanie powinno być zaimplementowane w dedykowanej klasie, testy powinny być w osobnej klasie).

1. Stwórz generyczną, iterowalną klasę Matrix wraz z metodą pozwalającą dodawać macierze (zgodnie z zasadami dodawania macierzy). Pokaż że dodawanie działa. Wykorzystaj iterator do wypisania macierzy na ekran.
2. Napisz metodę `int solution(List<Integer> a)`, która otrzyma listę N liczb całkowitych i zwróci najmniejszą dodatnią wartość (większą niż 0) która NIE występuje na liście.
   Na przykład:
   - a = [1, 3, 6, 4, 1, 2], solution = 5
   - a = [1, 2, 3], solution = 4
   - a = [-1, -3], solution =1

   Założenie są następujące:
   - Lista może mieć dowolną wielkość, ale załóżmy, że algorytm ma obsługiwać poprawnie tylko zainicjalizowane niepuste listy do rozmiaru 1E5 elementów. W innym przypadku odpowiedni wyjątek powinien zostać rzucony.
   - Każdy element listy jest liczbą całkowitą z przedziału [−1000000..1000000] i może obejmować dowolną ilość powtórzeń.

Zaimplementuj własne klasy wyjątków na wypadek nie spełnienia któregoś z wymogów algorytmu.

3. Napisz metodę `int substring(String a, String b)` zwracającą liczbę powtórzeń łańcucha *a*, aby *b* stał się podłańcuchem *a* (zawierał się w nim). Jeżeli *b* nie może być podłańcuchem *a*, to zwróć -1. Np. a = "abcd", b="cdabcdab" => 3 ponieważ dopiero trzy powtórzenia a dadzą "ab**cdabcdab**cd", który zawiera łańcuch b.
4. Napisz metodą `int[] solution (float[] arr, float target)`zwracającą indeksy dwóch elementów tablicy arr które sumują się do target. Np.
   ```
   float[] arr = {2, 7, 11, 15};
   float target = 9;
   int[] res = solution(arr, target);      // 0,1
   ```
   Jeśli zadanie nie ma rozwiązania należy rzucić wyjątek.
   Maksymalna ilość punktów będzie przyznana jeśli rozwiązanie będzie posiadało złożoność obliczeniową mniejszą niż $O(n^2)$.
5. Wybierz pięć dowolnych algorytmów sortowania i zmierz prędkość ich działania dla optymistycznego, pesymistycznego i realistycznego przypadku.
   https://www.geeksforgeeks.org/sorting-algorithms/

Przykład:
```
long tStart = System.currentTimeMillis();
// uruchom sortowanie.
long tEnd = System.currentTimeMillis();
long tDelta = tEnd - tStart;
double elapsedSeconds = tDelta / 1000.0;
```

Aby wyniki były porównywalne wielkość tablicy powinna być relatywnie duża. Tablicę w przypadku realistycznym należy wypełnić losowymi liczbami (zastosuj klasę *Random*)

Teoria:

1. Własne typy wyjątków, hierarchia dziedziczenia wyjątków
2. Do czego służy klasa Object i jakie ma zastosowanie ?
3. Jak budować własne kolekcje i struktury danych w Java? Generyczność i hierarchia dziedziczenia kolekcji https://www.javatpoint.com/collections-in-java

Po uzyskaniu zaliczenia na zajęciach, prześlij źródła w archiwum **zgodnie z konwencją nazewniczą** (patrz Lab0.pdf) do chmury na adres:
https://cloud.kisim.eu.org/s/k6TzoEJjrArMigH
najpóźniej do następnych zajęć