Zaawansowane programowanie obiektowe
Lab. 4
(Collator, Java 8; JSON; Google Guava)

1. (1 pkt)
   Napisz klasę zawierającą metody sortujące napisy z uwzględnieniem alfabetu polskiego (np. „Łukasz" ma być między „Lucyna" a „Marek").
   Wskazówka: wykorzystaj klasę java.text.Collator.

   Konkretnie napisz 3 metody sortujące:

   public static void sortStrings(Collator collator, String[] words)
   – sortującą napisy ręcznie i naiwnie, z użyciem sortowania bąbelkowego,

   public static void fastSortStrings(Collator collator, String[] words)
   i
   public static void fastSortStrings2(Collator collator, String[] words)
   – sortujące napisy z użyciem Arrays.sort(…).
   Różnica między tymi dwiema metodami jest taka, że fastSortStrings ma używać anonimowego obiektu komparatora, zaś fastSortStrings2 ma wykorzystać funkcję lambda (Java 8).

   W testach (z użyciem JUnit4) porównaj zgodność wyników zwracanych przez wszystkie te 3 funkcje, a także wyświetl wyniki na konsoli dla następującej tablicy:

   ```
   String[] names = {"Łukasz", "Ścibor", "Stefania", "Bolek", "Agnieszka",
                     "Zyta", "Órszula"};
   ```

   Wykonaj również test wydajnościowy tych 3 metod, sortując powyższą tablicę imion w pętli 100 tys. razy (oczywiście na starcie ma być za każdym razem nieposortowana). Tym razem nie wypisuj tablicy na ekranie. Wykorzystaj metodę System.nanoTime().

2. (1.5 pkt)
   Zaimplementuj test ze znajomości słówek angielskich. Baza ma liczyć 10 pytań, z których do testu losujemy bez powtórzeń 5. „Pytaniem" ma być słowo polskie, odpowiedzią – wpisywane z tzw. palca, ale w okienku dialogowym (poszukaj odpowiedniej metody klasy JOptionPane), słowo angielskie. Na końcu testu należy podać wynik (tj. ile pytań poprawnych) oraz zużyty czas, z dokładnością do 0.01s.

   Uwaga: jednemu słowu polskiemu może odpowiadać kilka (dozwolonych) tłumaczeń, np. krzyczeć -> shout / cry / scream. Program powinien być niewrażliwy na małe / wielkie litery (czyli shout / SHOUT / Shout etc. są równoważne).

   „Baza" pytań i odpowiedzi, w formacie JSON, zawarta jest w pliku PolEngTest.json (należy go najpierw „ręcznie" utworzyć!), który należy odczytać (zdeserializować) przy użyciu biblioteki google-gson: https://github.com/google/gson
   Jej dokumentacja:
   http://www.javadoc.io/doc/com.google.code.gson/gson/2.8.2

Format JSON jest przedstawiony np. pod
http://www.tutorialspoint.com/json/json_tutorial.pdf.

Przebieg egzaminu (tj. zbiór par napisów: pytanie-odpowiedź) ma być również zapisany w pliku JSON o nazwie imie_nazwisko.json, w analogicznym formacie jak plik wejściowy.

Wskazówka:
obejrzyj dokumentację klasy com.google.gson.Gson, a w szczególności pierwszy zamieszczony przykład. Zamiast MyType użyj swojego typu (tablicowego lub kolekcji) zawierający obiekty z pytaniami i odpowiedziami.

Biblioteka Guava (https://github.com/google/guava)

3. (0.25 pkt) Napisz funkcję o dwóch parametrach: String s, int length, rozcinającą i zwracającą jako listę parametr s na kawałki o długości length (bez nakładek). Ostatni kawałek może być krótszy.
   Przykłady:
   "Ala ma kota", 3 --> "Ala", " ma", " ko", "ta"
   "abcd", 2 --> "ab", "cd"
   Rzuć IllegalArgumentException, jeśli length <= 0 lub s == null.

   Następnie dodaj adnotację @NonNull (org.eclipse.jdt.annotation.NonNull) do parametru s tej funkcji i zademonstruj jej działanie.
   Wskazówka: http://szgrabowski.kis.p.lodz.pl/zpo17/Java_2017_04.ppt (S7)

   Przetestuj swoją funkcję, porównując wyniki jej działania z odpowiednią konstrukcją z klasy Splitter biblioteki Guava. Porównanie wyników ma być zrealizowane przy pomocy jUnit.

4. (0.25 pkt) Przygotuj kolekcję listową 6 obiektów klasy Student, o polach: imię, nazwisko, data urodzenia, wzrost (dobierz odpowiednie typy danych).
   Posortuj studentów wg kryterium:
   decyduje ROK urodzenia,
   a w obrębie tego samego roku alfabetycznie względem PIERWSZEJ litery nazwiska,
   a w obrębie tej samej pierwszej litery nazwiska MALEJĄCO wg wzrostu.

   Komparator do tego sortowania napisz na dwa sposoby:
   - korzystając z JDK,
   - korzystając z biblioteki Guava (użyj ComparisonChain).

   Przetestuj rozwiązania (dobierz odpowiednie dane).