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

1. (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, słowo angielskie. Na końcu testu należy podać wynik (tj. ile pytań poprawnych) oraz zużyty czas, z dokładnością do 0,01s.

   Jednemu słowu polskiemu może odpowiadać kilka (dozwolonych) tłumaczeń, np. krzyczeć -> shout / cry / scream. Program powinien być niewrażliwy na małe / wielkie litery (czyli cry / CRY / Cry etc. są równoważne).

   Co więcej, dopuszczamy 1 błąd Levenshteina (użyj bilbioteki Apache Commons Text). Jeśli odpowiedź jest z jednym błędem, to dajemy za nią 0,5 pkt. Jeśli jest bezbłędna, to 1 pkt.

   Przykłady:
   krzyczeć --> cry (1 pkt.)
   krzyczeć --> SoUT (0,5 pkt) (gdyż jedna z prawidłowych odpowiedzi to „shout"; małe/duże litery nie mają znaczenia)
   krzyczeć --> cri (0,5 pkt)
   krzyczeć --> cray (0,5 pkt)
   krzyczeć --> crie (0 pkt) (gdyż tu są już 2 błędy w sensie miary Levenshteina)

   „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 (lub innej użytecznej biblioteki).
   Dokumentacja Gson:
   http://www.javadoc.io/doc/com.google.code.gson/gson/2.8.5
   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.

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

2. (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", 4 --> "Ala ", "ma k", "ota"
"abcd", 2 --> "ab", "cd"
"", 3 --> ""

Rzuć IllegalArgumentException, jeśli length < 0 lub s == null.

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

3. (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).

4. (1 pkt) W pliku dane.txt są dane osobowe:
   imię nazwisko kraj zarobki
   Należy wypisać sumę zarobków 2 najlepiej i 2 najgorzej zarabiających Polaków (PL) – prawidłowy wynik: 101999 i 38900.
   Ponadto dla każdego kraju występującego w pliku należy wypisać liczbę osób, w kolejności pojawiania się krajów w kolejnych wierszach.
   Czyli: PL - 6, DE - 2, CZ - 3, US - 2.

   Wykorzystaj konstrukcje Javy 8 (strumienie, wyrażenia lambda).
   Do odczytania pliku tekstowego poszukaj odpowiedniej metody klasy Files (leniwie wstawiającej linie tekstu do strumienia).
   Możesz 3 razy tworzyć strumień (z wejściowego pliku), nie więcej. W wybranych miejscach użyj peek() do podejrzenia zawartości strumienia (cel: debugowanie kodu... a może jeszcze do czegoś?).