Zaawansowane programowanie obiektowe
Lab. 7
(Guava, programowanie sieciowe, strumienie IO/pliki)

1. (BISEKCJA, LISTA LENIWA) (1 pkt)
Algorytm bisekcji znajduje np. miejsce zerowe funkcji w danym przedziale. W takim przypadku wymagamy, aby funkcja była monotoniczna w tym przedziale i w jednym z krańców przedziału osiągała wartość ujemną, a w drugim wartość dodatnią.

Proszę napisać program realizujący algorytm bisekcji dla dowolnej prostej funkcji (np. kwadratowej), z wykorzystaniem Collections.binarySearch(...). Uwaga: nie chcemy tablicować wszystkich wartości funkcji (z zadanym krokiem, np. 0.001) w danym przedziale, gdyż mogłoby to być zbyt kosztowne. Niech funkcja będzie na bieżąco obliczana w danym punkcie.

Wskazówki:
- użyj listy leniwej (poszukaj „guava lazy list" etc, np. na stackoverflow) z biblioteki Guava,
- do doboru funkcji matematycznej i przedziału wejściowego pomocna może być strona:
  https://www.desmos.com/calculator

2. (KLIENT-SERWER) (1 pkt)
Proszę napisać konsolową aplikację klient-serwer z połączeniem TCP. Klient przekazuje serwerowi parę liczb naturalnych s i e (jak start, end), które określają przedział [s; e]. Zakładamy poprawność danych. Serwer ma policzyć i zwrócić ilość liczb pierwszych w tym przedziale, a także czas obliczeń. Serwer ma wykorzystać do obliczeń 2 wątki (jeśli więc na serwerze są co najmniej 2 nieobciążone rdzenie, to możemy oczekiwać przyspieszenia względem wersji 1-wątkowej).

Algorytm sprawdzenia, czy liczba naturalna *m* jest pierwsza:
- jeśli *m* = 1, to NIE jest pierwsza (choć akurat ta liczba nie jest też złożona),
- jeśli *m* = 2 lub *m* = 3, to jest pierwsza,
- jeśli *m* > 3, to sprawdź podzielność przez 2 oraz przez wszystkie liczby nieparzyste >= 3 do sqrt(*m*) włącznie; jeśli *m* nie dzieli się przez żadną z nich, to jest pierwsza, w przeciwnym razie jest złożona.

Algorytm ten uruchamiamy osobno dla każdej liczby z przedziału [s; e].

Uwaga: podział przedziału na równe części jest nieoptymalny; warto eksperymentalnie poszukać lepszego podziału. Przykładowo, dla przedziału [20000; 39999] być może warto wziąć podprzedziały: [20000; 30999] i [31000; 39999].

Proszę zaobserwować w Menedżerze zadań Windows (albo np. z użyciem narzędzia *htop* pod linuksem) obciążenie procesora za sprawą obliczeń na serwerze.

Prosty przykład aplikacji TCP klient-serwer:
http://systembash.com/content/a-simple-java-tcp-server-and-tcp-client/

3. (0.5 pkt)
Napisz program generujący *N* liczb losowych o rozkładzie normalnym (gaussowskim) o zadanych parametrach *m* (średnia) i σ (odchylenie standardowe) i zapisujący je do pliku binarnego. Następnie plik ten jest odczytywany, liczba po liczbie, a dane zapisywane do nowego pliku (tekstowego), każda liczba w osobnym wierszu. Dodatkowo należy zwizualizować (przy pomocy Excela lub w Pythonie/matplotlib) histogram tych liczb, aby sprawdzić, czy rzeczywiście rozkład przypomina krzywą dzwonową Gaussa. Zapis do pliku tekstowego ma wykorzystywać konwencję formatu przyjętą w Polsce (tj. przecinki zamiast kropek dziesiętnych).
Parametry *N* (liczba całkowita), *m* i σ (liczby zmiennoprzecinkowe) mają być argumentami wiersza poleceń. Przy pomocy mechanizmu asercji dopilnuj, aby odchylenie standardowe nie było liczbą ujemną.

Wskazówki.
a) Generacja liczb o rozkładzie normalnym – poszukaj odpowiedniej metody w klasie Random.
b) Zapis/odczyt liczb w postaci binarnej: klasy DataOutputStream / DataInputStream.
c) Utwórz odpowiedni obiekt klasy NumberFormat.