Zaawansowane programowanie obiektowe
Lab. 8
(Refleksja, adnotacje, wątki)

1 (REFLEKSJA, 0.75 pkt.)
Proszę napisać klasę o nazwie MaxSearchAlgorithms, która będzie zawierała 3 algorytmy (jako osobne metody o domyślnej/pakietowej widoczności) szukania maksimum w tablicy liczb typu int: od lewej do prawej, od prawej do lewej oraz najpierw czytane elementy na indeksach parzystych, a następnie na nieparzystych (uwaga: nie należy zakładać, że tablica przekazana jako argument zawiera parzystą bądź nieparzystą liczbę elementów). Wszystkie te 3 funkcje mają zwracać tablicę/listę (np. ArrayList) elementów, które były „chwilowymi" elementami maksymalnymi przy odnośnej strategii skanowania.

Przykład: jeśli na wejściu funkcji skanującej od prawej do lewej będzie tablica {4, 10, 3, 7, 4, 1, 6, 2}, to funkcja ma zwrócić tablicę {2, 6, 7, 10}.

Opisane 3 metody mają być wywołane na rzecz obiektu klasy MaxSearchAlgorithms utworzonego w innej klasie, za pomocą mechanizmu refleksji. Należy mianowicie odczytać wszystkie metody zdefiniowane w tej klasie, a następnie „odfiltrować" te, które nie zawierają w nazwie ciągu „Scan" (opisane wyżej 3 metody będą taki ciąg znaków w nazwie posiadały). Metody zawierające ww. napis mają zostać uruchomione (metoda invoke(...) z klasy Method), a zwrócone przez nie tablice wypisane na ekran.

2 (REFLEKSJA I ADNOTACJE, 1 pkt)
Utwórz klasę posiadającą przynajmniej 4 pola (prywatne) różnych typów (w tym prymitywnych), np. int, String, boolean...
Wygeneruj, przy użyciu IDE (Eclipse lub IntelliJ), settery i gettery dla tych pól.
Następnie napisz na dwa sposoby metodę equals(...) porównującą obiekty Twojej klasy, uwzględniając wszystkie pola z wyjątkiem jednego (np. przy klasie Osoba można zignorować pole telefon):
rozwiązanie tradycyjne („ręczne" porównywanie pól),
z użyciem refleksji i adnotacji.

Ad a) Opatrz metodę equals(...) odpowiednią adnotacją oznaczającą metodę przeciążoną (z Object).

Ad b) Dodaj własny typ adnotacyjny:

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface IgnoreEquals {}
```

a następnie wykorzystaj go przy wybranym polu stosując odpowiednią metodę z klasy Method.

Rozwiązanie przetestuj.

3. (WĄTKI, 1.25 pkt)
Napisz aplikację wielowątkową wypisującą na konsoli napisy z tablicy po znaku, z zachowaniem kolejności wątków w każdej „rundzie”.
Dokładniej: na początku każdy wątek wypisuje po jednym znaku – to jest pierwsza runda. W następnych rundach wątki wypisują też po znaku, w kolejności wątków określonych przez pierwszą rundę. Gdy dany wątek zakończy swoją pracę (tj. skończą mu się znaki), wtedy oczywiście jest pomijany.

Przykład:
private String[] strings = {"aaaa", "bb", "ccccccccccc", "dddddd"};
Wynikiem (spacje pomiędzy rundami dodane tylko dla przejrzystości) może być:
abcd abcd acd acd cd cd c c c c c c c
ale też może być np.
dbac dbac dac dac dc dc c c c c c c c