Zaawansowane Programowanie Obiektowe i Funkcyjne
Wyrażenia lambda
Zadanie oceniane nr 3
13-11-2019

Dzisiaj będzie bez kodu wstępnego. Źródła powinny się znaleźć w Państwa katalogu roboczym (git) w podkatalogu Lab6-zadanie3. Po zakończeniu pracy konieczne jest wgranie zmian.

"Czekoladki"

"Życie jest jak pudełko czekoladek, nigdy nie wiadomo co si się trafi..."

W dniu dzisiejszym zasymulujemy sobie proces kontroli jakości wyrobów czekoladowych. Oczywiście będziemy używać do tych celów kodu zdefiniowanego za pomocą wyrażeń lambda jak i klas anonimowych w zależności od tego co będzie nam niezbędne. Będziemy tworzyć kolekcję czekoladek, a potem ją procesować poprzez choćby zaburzanie wartości, żeby w końcu wykryć nieprawidłowości w składzie i odrzucić czekoladkę jako niepełnowartościową. Miejmy gdzieś z tyłu głowy fakt, iż w porównaniu z klasą anonimową obiekt zdefioniowany za pomocą wyrażenia Lambda, można traktować jako metodę anonimową, czyli byt jeszcze bardziej "okrojony" z niepotrzebnego ciała niż klasa "bez nazwy".

Prace do wykonania:

1. Stworzyć hierarchię klas reprezentującą dobrze niektórym znane słodycze:
   - Kasztanki (zawartość wafli (6,1%), energia(100 kcal), zawartość czekolady(41,6%), waga(19 g))
   - Tiki Taki (zawartość wiórków kokosowych (9,1%), energia(102 kcal), zawartość czekolady(44%),śladowe ilości orzechów (tak), waga(18 g))
   - Malaga (waga(22 g), energia (105 kcal), czy zawiera spirytus (tak), śladowe ilości orzechów (tak),zawartość czekolady(50%))

Wartości podane w nawiasach są przyporządkowywane polom domyślnie.

Proszę pamiętać o:
- konstruktorach
- typach
- widoczności pól i metod z zewnątrz (tylko to co potrzebne)
- umożliwieniu dalszego dziedziczenia pól i metod
- poprawnemu skonstruowaniu hierarchii klas

2. Stworzyć klasę **ChockolateProcessor** która będzie jednocześnie klasą demonstracyjną (wywołującą wszystkie swoje metody) posiadającą referencję do obiektu klasy **ChockolateHolder**. Klasa zawiera metody:
   - *generateAll*, która wywołuje metodę generateChockolates (opisaną w następnym podpunkcie) i przekazuje jej implementację interfejsu za pomocą wyr. Lambda.
   - *disturb*, która wywołuje metody disturbKasztanki, disturbMalaga, disturbTikitaki (opisaną w następnym podpunkcie) i przekazuje im odpowiednie implementacje interfejsu za pomocą wyr. Lambda.
   - *validate*, która uruchamia każdą metodę klasy **ChockolateValidator** (pkt. 4) dla kolekcji czekoladek.
3. Stworzyć klasę **ChockolateHolder** posiadającą kolekcję czekoladek wszystkich typów zawierającą metody:
   - *generateChockolates* – metoda jako parametr przyjmuje interfejs funkcyjny (wykorzystywany tylko przez tą klasę), który trzeba stworzyć. Oczywiście jego implementacja (obiekt) za pomocą wyrażenia Lambda będzie dostarczona przez metodę *generateAll* klasy **ChockolateProcessor** podczas demonstracji działania (metoda *generateAll* wywołuje metodę *generateChockolates* i przekazuje do niej implementację interfejsu funkcyjnego). Ma zostać wygenerowanych po 20 obiektów każdego ze wspomnianych typów)
   - *disturbKasztanki* - metoda jako parametr przyjmuje interfejs funkcyjny (wykorzystywany tylko przez tą klasę) którego implementację dostarcza wywołująca ją metoda *disturb* klasy **ChockolateProcessor.** Interfejs ten pobiera obiekt klasy Kasztanki i modyfikuje zawartość wafli o losową wartość od -1 do 1. Jego metoda jest wywoływana dla każdego obiektu klasy Kasztanki.
   - *disturbMalaga* - metoda jako parametr przyjmuje interfejs funkcyjny (wykorzystywany tylko przez tą klasę) którego implementację dostarcza wywołująca ją metoda *disturb* klasy **ChockolateProcessor.** Interfejs ten pobiera obiekt klasy Malaga i raz na 10 przypadków zmienia fakt zawartości spirytusu na false. W p.p. z prawdpodobieństwem 0.5 zmienia fakt istnienia śladowej zawartości orzechów na false. Metoda tego interfejsu jest wywoływana dla każdego obiektu klasy Malaga.
   - *disturbTikitaki* - metoda jako parametr przyjmuje interfejs funkcyjny (mogący być wykorzystanym w sprawach związanych nie tylko z tą klasą oraz przez inne klasy) którego implementację dostarcza wywołująca ją metoda *disturb* klasy **ChockolateProcessor**. Interfejs ten pobiera obiekt klasy TikiTaki i zmniejsza zawartość wiórków kokosowych o 20%. Ponadto z prawdpodobieństwem 0.3 zwiększa masę dwukrotnie, a prawdpodobieństwem 0.7 zmniejsza zawartość czekolady 0 60%. Metoda tego interfejsu jest wywoływana dla każdego obiektu klasy TikiTaki.

4. Stworzyć klasę **ChockolateValidator** której metody przyjmują kolekcję czekoladek i walidują je w sposób opisany w podpunktach. Za pomocą wyrażenia lambda lub klasy anonimowej (tam gdzie to niezbędne) stworzyć w danej metodzie implementację któregoś z predefiniowanych pasujących jak ulał interfejsów funkcyjnych (java.function.*) i użyć go dla każdego elementu kolekcji. Dla ułatwienia: jest to interfejs służący do konsumowania.

Strumienie mile widziane ale nie obowiązkowe.

- sumuje zawartość czekolady w kolejno procesowanych czekoladkach, w taki sposób żeby wynik był dostępny również po przeprocesowaniu ostatniej czekoladki.
- skanuje czekoladkę Malaga pozytywną pod względem występowania w niej spirytusu. Jeśli będzie, wypisuje na konsoli: "Tylko od 18 lat!."
- jeśli napotka czekoladkę negatywną pod względem wystepowania śladowych ilości orzechów, to wypisuje na konsoli: "Alergik – ok!".
- jeżeli zawartość wiórek kokosowych jest większa niż 9.1, ustawia wartość boolowską jakiejś zmiennej poza ciałem wyrażenia ale w ciele metody.

5. Utworzyć klasę **ChockolateSorter** która nie musi być instancjonowana, dostarczającą następujących funkcjonalności (kod porównujący obiekty dostarczony oczywiście za pomocą wyrażenia Lambda):
    - zwraca tylko posortowane obiekty mające możliwość posiadania śladowych ilości orzechów (najpierw te które mają).
    - kolekcję obiektów posortowanych po masie malejąco.