Zaawansowane programowanie obiektowe i funkcyjne
Wyrażenia Lambda

Zestaw zadań ocenianych 2a

05-11-2018

Zadania znajdują się w repozytorium (każdy ma swoje konto w podkatalogu Lab2a). Należy zsynchronizować się za pomocą polecenia git pull. Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (commit + push). Proszę pamiętać o poleceniu add.

Ważnym jest, aby:

- używać wyrażeń lambda gdzie się tylko da (o ile to możliwe). W przeciwnym wypadku używać klas anonimowych.
- używać referencji na metody, w przypadku gdy wyrażenie lambda jest na tyle proste iż użycie jest zasadne.

Zadanie 1 – Świat zwierząt (kod w pakiecie: pl.edu.pw.mini.de.zpoif.assessed.lab2.task1)

1. Utworzyć rodzinę klas zawierającą elementy takie jak: Pies(wiek, imię, ilość zjedzonych jednostek jedzenia(1-5-), liczbaPogryzionych osób(0-10)), Kot(wiek, imię, ilość zjedzonych jednostek jedzenia(1-5)), liczba podrapanych osób, Mysz (ilość zjedzonych jednostek jedzenia(1-5), ilość zniszczeń(1-100)) i zainicjalizować nimi kolekcję zawierającą 30 elementów. Wszystkie dane są generowane automatycznie – losowo. Instancje zwierząt występują w liście z identycznym prawdopodobieństwem. Metoda toString() powinna zwracań wszystkie informacje zawarte w obiekcie. Elementy te zostają stworzone w konstruktorze.

2. Każde z tych zwierząt posiada metody: nakarm oraz pogłaszcz. Wywołanie tych metod powoduje odpowiednio zwiększanie ilości zjedzonego jedzenia oraz wypisywanie informacji na konsoli: "Ktoś mnie pogłaskał".

3. Stworzyć klasę z metodą (metodami) demonstrującymi inicjalizację oraz wykonanie powyższych i poniższych punktów zadania.

4. Za pomocą wyrażenia lambda stworzyć implementację obiektu java.util.function.Consumer, która konsumuje ciągi znaków (Stringi) w sposób następujący:

-jeżeli ciąg znaków jest liczbą, to na konsoli pojawia się informacja o tym, czy jest parzysta, czy też nie.
- jeżeli ciąg znaków nie jest liczbą i jest nie dłuższy niż 20, to zostaje wypisany na konsoli (w przeciwnym wypadku pojawia się napis: ciąg za długi).
- wypisanie informacji, jeżeli ciąg znaków zaczyna się na 'A'

5. Używając metody należącej do listy (**forEach**) z pomocą wyrażeń lambda dostarczyć sposobu wykonania następujących operacji na wszystkich elementach listy (implementacja interfejsu java.util.function.Consumer):

- wywołanie metody toString na każdym elemencie i wypisanie jej na konsolę
- wywołanie metody nakarm na każdym obiekcie
- wywoływanie metody pogłaszcz tylko* na kotach
- wypisanie na konsolę ilości pogryzionych osób o ile obiektem jest pies*
- zliczanie zwierząt młodszych niż 2 lata
- wypisanie długości imienia danego zwierzęcia.

Użycie wzorca projektowego "Odwiedzający" (Visitor) będzie premiowane. Dopuszczalne jest posłużenie się operatorem instanceof.

6. Posortować listę ze zwierzętami używając następujących implementracji interfejsu java.util.Comparator (stworzonych za pomocą wyrażeń lambda):

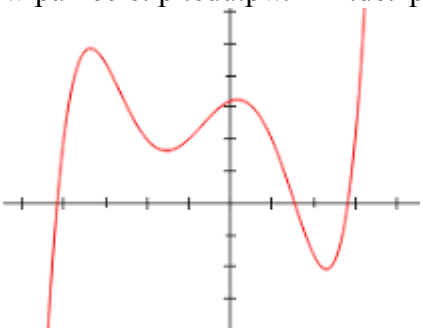
- sortowanie po wieku
- sortowanie po imieniu (jeżeli jest dłuższe niż 5 znaków)
- sortowanie względem stopnia najedzenia zaczynając od najbardziej głodnych.

7. Stworzyć interfejs funkcyjny (będzie on używany tylko w tej klasie), którego charakter będzie nadzorowany przez kompilator i zaimplementowanie go za pomocą wyrażeń lambda. Zawiera on metodę: najedzony(Zwierzę, minimalny stopień najedzenia), który w przypadku gdy metoda zostanie wywołana na osobniku o mniejszym stopniu najedzenia niż argument wyppisuje informaję na konsoli: Zwierzę głodne.

8. Stworzyć interfejs funkcyjny (którego natura nie jest pilnowana przez kompilator) używany jedynie przez klasę demonstracyjną, do losowego generowania jednego z trzech rodzajów komparatora, o którym była mowa w punkcie 6-tym. Oczywiście z użyciem wyrażeń lambda.

9. Stworzyć interfejs funkcyjny generujący wątki z których każdy wypisuje swój identyfikator.

Zadanie 2 Wielomiany (kod w pakiecie: pl.edu.pw.mini.de.zpoif.assessed.lab1.task2)



1. Użyć interfejsu funkkcyjnego BiFunction, do zdefiniowania implementacji funkcji wyszczególnionych poniżej:
   - wielomian 2-go stopnia
   - wielomian 3-go stopnia
   - wielomian 4-go stopnia
   - sin (x)
   - moduł z x + 1
   - log x

2. Zdefiniować interfejs FunctionGenerator, którego implementacja dostarczona przez wyrażenie lambda zwraca nam kolekcję funkcji o losowej ilości (1-5).

3. Zdefiniować interfejs MyFunction, którego implementacja dostarczona przez wyrażenie Lambda, zwraca wartość będącą sumą wartości funkcji składowych znajdujących się w kolekcji z punktu drugiego.

4. Zdefiniować interfejs SophisticatedFunction (implementowany za pomocą wyrażeń lambda), który w odpowiedzi na parametr zwraca wartość wynikającą ze złożenia funkcji składowych.