Zaawansowane Programowanie Obiektowe i Funkcyjne
Adnotacje wspierane reflekcjami
Zadanie oceniane nr 5
24-01-2022

Kod wstępny zadania znajduje się w repozytorium w katalogu zadanie5. Należy zsynchronizować się za pomocą polecenia git pull (lub git clone jeśli ktoś tego wcześniej nie zrobił) oraz dokonać importu źródeł znajdujących się w podkatalogu zadanie5b. Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (commit + push).

"Mafia 2"

Mafia lubi wracać. Dzisiaj zaimplementujemy niewielki "silnik mafijny" zawierający metody działające w oparciu o adnotacje użyte w stosunku do hierarchii członków mafii. Działanie aplikacji należy zaprezentować w jakimś demonstratorze. Nie używamy operatora "new".

Dostarczone zasoby:

1) Źródła znajdujące się w repozytorium w katalogu zadanie5 (należy sprowadzić je na komputer lokalny z repozytorium zdalnego)
   a) Klasa `pl.edu.pw.mini.zpoif.task5.people.MafiaWorker`
   Jest to rdzeń rodziny klas opisujących członków mafii, których nie można zmieniać chyba że zadanie będzie tego wymagało i zostanie to napisane wprost (np. dodanie adnotacji).
   b) Klasa `pl.edu.pw.mini.zpoif.task5.machine.MafiaMachine` klasa zawierająca metody które trzeba zaimplementować.

Prace do wykonania:

1. Adnotacja dla obiektu
   a) utwórz adnotację "ImportantWorker"dla klas widoczną również w czasie działania programu. Ma ona mieć parametr quantity z domyślną wartością 1.
   b) oznacz tą adnotacją klasy GodFather (quantity = 1) i Accountant (quantity = 3)
   c) zaimplementuje metodę `createImportantMafiaWorkers()` klasy `MafiaMachine`, która zwraca utworzone za pomocą refleksji obiekty zaadnotowanych klas w ilości określonej parametrem "quantity" adnotacji ImportantWorker. Tworzone są instancje tylko klasy z większą wartością quantity. W przypadku gdy będą równe, tworzymy obiekty obu klas. Adnotację zaimplementować w sensownym miejscu. Dla ułatwienia sprawdzane są klasy występujące w skład tej hierarchii (można założyć że struktura nigdy się nie zmieni i ustawić je na sztywno).
2. Adnotacja dla klasy
   a) utwórz adnotację PrimaryMafiaWorker dla klas widoczną w czasie działania programu.
   b) Oznacz tą adnotacją klasę GodFather, Accountant i Spy
   c) zaimplementuje metodę `createPrimaryMafiaWorker()` klasy `MafiaMachine`, która przeszukuje klasy opisujące przacowników. Dla ułatwienia sprawdzane są klasy występujące w skład tej hierarchii (można założyć że struktura nigdy się nie zmieni). Instancjonowany jest pracownik adnotowany tą adnotacją. Jeśli znajdziemy więcej niż jedną klasę zwracmy wylosowany obiekt. Zakładamy, że wywoływany jest konstruktor bezparametrowy. W przypadku braku adnotacji zwraca pustą listę.
3. Upgrade adnotacji dla klas
   a) Udoskonalić adnotację w poprzedniego podpunktu i dodać do niej parametr "priority" (domyślna wartość to najmniejszy Integer)
   b) Oznaczyć tą adnotację klasy GodFather (priorytet = 1) i Spy (priorytet = 3)
   c) zaimplementuje metodę `createPrioritizedPrimaryMafiaWorker()` klasy `MafiaMachine`, która w odróżnieniu od `createPrimaryMafiaWorker` gdy znajdzie więcej niż jedną klasę z adnotacją PrimaryMafiaWorker sprawdza parametr pririty i instancjonuje i zwraca obiekt klasy o wyższym priorytecie. Jeśli gdzieś priorytety będą takie same to wybiera tą klasę którą uważa. Zakładamy, że wywoływany jest konstruktor bezparametrowy. W przypadku braku adnotacji zwraca null.
4. Adnotacja dla pól
   a) Utwórz adnotację FillIfEmptyIt dla pól
   b) Zadnotuje nią "name", "surname" i "secondName" danego pracownika (jeśli je posiada)
   c) Zaimplementuj metodę `fillFields(Set<MafiaWorker> workers)` klasy `MafiaMachine`, która otrzymuje zbiór z kilkoma pracownikami z ustawionymi lub nie imionami i nazwiskami. Obiekt klasy Accountant również musi być w tym zbiorze. Metoda przeszukuje pola typu String w poszukiwaniu adnotacji FillIfEmptyIt i jeżeli jest ono puste to zamienia jego zawartość na losowy String (google: "java random string" :)).
5. Adnotacja dla parametrów konstruktora
   a) Utwórz adnotację dla parametrów konstruktora MafiaValidator z parametrami: notEmpty (domyślnie true) i maxLength (domyślnie 5).
   b) Ustaw na parametrach "name" w konstruktorach klasy cyngla (ButtonMan)
   c) zaimplementuje metodę `getKiller(String name, String surname)` klasy `MafiaMachine`, która instancjonuje zabójcę w ten sposób że przekazuje do jego konstruktora parametry name i surname przekazane do metody. Jeśli znajdzie na którymś z parametrów adnotację MafiaValidator to sprawdza czy wartość jest zgodna z parametrami adnotacji. Jeśli nie, to wyrzuca wyjątek MafiaException.
6. Adnotacja dla pól
   a) Stworzyć adnotację InitMe
   b) Oznaczyć nią pole pocket w klasie GodFather

c) zaimplementuje metodę `init(MafiaWorker mafiaWorker)` klasy `MafiaMachine`, która sprawdza czy któreś z pól nie ma przypadkiem ustawionej tej adnotacji. Jeśli ją znajdzie to szuka (na terenie klas z rodziny MafiaWorker) lokalnej klasy typu pola które jest zaadnotowane, a gdy się na nią natknie to instancjonuje ją i przyporządkowuje do tego właśnie pola.

7. Adnotacja dla metod
   a) Utwórz adnotację DoIt a parametremi times (domyślnie = 1)
   b) Zaadnotuje nią metody cyngla (ButtonMan) takie jak: `killHim()` i `killThemAll()` i ustaw na nich wartości 4 i 7
   c) Zaimplementuj metodę `goButtonMan(Set<MafiaWorker> buttonMan)` która wywołuje metody oznaczone tą adnotacją tyle razy na ile ustawione są ich parametry. Jako parametry przekazuje wartość z wylosowanego pola typu String z tej klasy, którego nazwa kończy się na "victim".
8. Stwórz nietrywialne zadanie w stylu pozostałych odnoszące się do adnotacji dla metod (stworzyć, zaadnotować i coś z tym zrobić w ramach nowej metody klasy `MafiaMachine`) i je rozwiąż (tylko dobrze!).