ZMWO – laboratorium z programowania aspektowego w AspectJ

Temat ćwiczenia: Repozytorium projektów

Opis aplikacji wykorzystywanej w ćwiczeniu

Ćwiczenie oparte jest na aplikacji będącej symulacją repozytorium projektów. Aplikacja dostarcza podstawowy interfejs umożliwiający zarządzanie projektami, zadaniami i dokumentami. Składa się ona z następujących elementów:

- *Repository*

Klasa reprezentująca repozytorium projektów. Umożliwia dostęp do przechowywanych projektów oraz zarządzanie projektami (dodawanie, aktualizację i usuwanie).

- *Project*

Klasa reprezentuje projekt, który może być przechowywany w repozytorium. Każdy z projektów może posiadać wiele zadań oraz dokumentów. Klasa *Project* udostępnia operacje umożliwiające dostęp oraz zarządzanie przypisanymi zadaniami i dokumentami (dodawanie, aktualizację i usuwanie).

- *ProjectExporter*

Klasa udostępnia publiczną metodę *export()*, która wypisuje szczegółowe dane dotyczące zadanego projektu na podany strumień.

Wypisywane są:

  - nazwa projektu
  - liczba i szczegóły dotyczące przypisanych zadań
  - liczba i szczegóły dotyczące przypisanych dokumentów
- *Task*

Klasa reprezentująca zadanie. Zadanie identyfikowane jest poprzez numer nadawany podczas jego tworzenia. Każde zadanie powiązane jest z dwiema osobami: zgłaszającym oraz przypisanym. Ponadto, zadanie posiada swój status oraz opis. Klasa *Task* udostępnia operacje pozwalające na dostęp oraz modyfikację parametrów zadania (powiązanych osób, opisu oraz statusu).

- *Document*

Klasa reprezentująca dokument tekstowy. Udostępnia operację odczytu i zapisu danych.

- *Employee*

Klasa reprezentująca osobę, która może zostać powiązana z zadaniem.

Przykładowe działanie aplikacji zostało umieszczone w klasie *Main*.

Treść ćwiczenia laboratoryjnego:

Rozwiązania zadań 1 - 4 należy wykonać **bez ingerencji w kod źródłowy aplikacji**.

Zadanie 1

Zbuduj aspekt śledzący, który dla każdego wywołania metody zapisuje do pliku dziennika:

- klasę obiektu, na rzecz którego została wykonana metoda
- nazwy, typy i wartości argumentów metody
- wynik zwracany przez metodę
- czas wykonania metody podany w milisekundach

Punkty złączeń mają wystąpić tylko w oryginalnym kodzie aplikacji, z pominięciem rozszerzeń aspektowych.

Zadanie 2

Zaimplementuj aspekt, który umożliwi automatyczne zapisywanie stanu repozytorium po każdej jego modyfikacji oraz wczytanie jego wcześniejszego stanu po utworzeniu nowego obiektu repozytorium.

Podpowiedź: można wykorzystać serializację obiektów oraz mechanizm *Inter-type declarations*

Zadanie 3

Zaimplementuj aspekt, który przyspieszy działanie metody *export()* klasy *ProjectExporter*. Przetwarzanie danych polegające na odczycie na danych projektu – nazwy, informacji o zadaniach i dokumentach – powinno odbywać się jedynie, gdy metoda wywoływana jest po raz pierwszy lub dokonane zostały zmiany w danych klasy *Project* (np. dodano nowe zadanie). W przeciwnym przypadku na podany jako parametr metody *export()* strumień powinna zostać wypisana wartość wyznaczona po ostatniej modyfikacji danych (bez jej ponownego wyznaczania).

Zadanie 4

Zaimplementuj aspekt, który:

- pobiera dane użytkownika (imię lub imię i nazwisko)
- hasło użytkownika
- weryfikuje hasło w momencie wykonywania przez użytkownika pierwszej operacji modyfikującej dane w repozytorium (z wyłączeniem modyfikacji wykonywanych w aspekcie zdefiniowanym w zadaniu 2)

Informacje o użytkownikach i hasłach mogą być przechowywane w zwykłym pliku tekstowym lub np. w serializowanym słowniku.

Zadanie 5

Zaimplementuj aspekt oraz dokonaj zmian w kodzie aplikacji, aby wykorzystać instrukcję *declare soft* języka AspectJ. Instrukcja ta pozwala na konwersję wyjątków weryfikowalnych do typu *SoftException*, niewymagających jawnej obsługi.