**Projekt**

**Technologie obiektowe  
dr inż. Mariusza Bedla**

**Kamil Jasiak 1ID21A**

**Temat: CSV-Object mapper**

# Wstęp

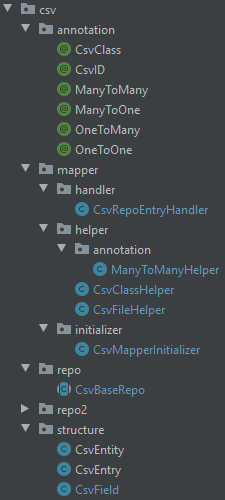
Tematem projektu jest aplikacja, bądź biblioteka, która umożliwi użytkownikowi mapowanie obiektów na pliki CSV i odwrotnie. Założenie jest podobne do JPA, czyli na podstawie adnotacji użytkownik tworzy klasy modelowe oraz odpowiadające im repozytoria.

# O aplikacji

Aplikacja jest napisana w Javie w wersji 11 w środowisku programistycznym Intellij IDEA Ultimate. Aplikacja jest osobnym modułem, który użytkownik dołącza do swojego projektu, aby móc z niego korzystać. Aplikacja posiada plik z logami (log.txt), do którego zapisują się dane o operacjach wykonywanych przez program jak również dane błędów.

# Struktura aplikacji

Zawartość mappera:

* Adnotacje
* Klasy mapujące obiekty i pliki csv
* Bazowe repozytorium
* Pomocnicze klasy modelowe

Adnotacja *CsvClass* jest przeznaczona dla klas, natomiast reszty używamy przy polach danej klasy. W klasie *CsvRepoEntryHandler* istnieją metody do zapisu oraz usunięcia obiektu do/z repozytorium. Klasa *ManyToManyHelper* pomaga w przetwarzaniu adnotacji *ManyToMany­* podczas inicjalizacji oraz modyfikacji danego repozytorium. *CsvClassHelper*  oraz *CsvFileHelper* pomagają w przetwarzaniu klas oraz plików. Zawierają podstawowe operacje na klasach/plikach. *CsvMapperInitializer* posiada metodę *init()*, która jest najważniejszą metodą. Bowiem to ona inicjuje całego mappera. *CsvBaseRepo* jest klasą abstrakcyjną. Użytkownik korzystający z mappera czasami będzie musiał rozszerzać tę klasę. *CsvEntity, CsvEntry, CsvField* są to klasy modelowe pomocnicze.

# Schematy mapowania

1. **Inicjalizacja – tworzenie repozytoriów**

Program przeszukuje folder z klasami modelowymi (w przyszłości rozwiązanie z wykrywaniem klas na podstawie adnotacji). Dla każdej klasy modelowej program tworzy plik z rozszerzeniem *csv* reprezentujący repozytorium danej klasy:

* Wyciąga nazwę klasy oraz nazwę pliku
* Sprawdza czy istnieje już repozytorium dla danej klasy w podanej lokalizacji
* Jeżeli nie istnieje, wyciąga pola klasy wraz z adnotacjami i generuje plik repozytorium uzupełniony o pierwszy wiersz (z odpowiednimi nazwami pól klasy) – więcej informacji o przetwarzaniu adnotacji w punkcie 4)

1. **Zapis obiektu do repozytorium**

Do zapisu obiektu potrzebny nam będzie obiekt klasy repozytorium danej klasy modelowej. Na ten moment program tworzy wpis do repozytorium na podstawie pomocniczej klasy modelowej *CsvEntry* oraz wartości string, która przechowuje nazwę danej klasy modelowej od użytkownika. Klasa *CsvEntry* reprezentuje listę obiektów klasy *CsvField*. Program:

* Tworzy obiekt *Class* na podstawie nazwy klasy modelowej użytkownika
* Wyciąga ścieżkę do repozytorium z adnotacji *CsvClass*
* Na podstawie ścieżki odczytuje z pliku *csv* wpisy danego repozytorium (bez wiersza z nazwami pól) i zapisuje go jako wartość string
* Na podstawie obiektu *CsvEntry* tworzy wiersz oraz dodaje do istniejącego stringa
* Zapisuje stringa jako repozytorium w pliku *csv*

1. **Usunięcie obiektu z repozytorium**

Usuwanie obiektu odbywa się podobnie jak dodawanie, ale tu nie potrzebujemy pomocniczych klas modelowych. Do usunięcia obiektu potrzebujemy właściwie tylko jego ID i nazwy klasy modelowej, której dotyczy ten właśnie obiekt. Program:

* Tworzy obiekt *Class* na podstawie nazwy klasy modelowej użytkownika
* Wyciąga ścieżkę do repozytorium z adnotacji *CsvClass*
* Na podstawie ścieżki odczytuje z pliku *csv* wpisy danego repozytorium (bez wiersza z nazwami pól) i zapisuje go jako wartość string
* Wartość string jest przekonwertowana na listę stringów i na podstawie ID obiektu odpowiedni element listy jest usuwany
* Zapisuję listę do pliku repozytorium

1. **Mapowanie adnotacji**

CsvClass:

Jeżeli na danej klasie występuje adnotacja @CsvClass to dana klasa jest brana pod uwagę jako klasa modelowa biorąca udział w mapowaniu csv.

ID:

Jeżeli na danym polu występuję adnotacja @CsvID to dane pole jest brane pod uwagę jako klucz podstawowy danej klasy.

Wiele do jednego:

Jeżeli na danym polu występuje relacja wiele do jednego do nazwy pola (w pierwszym wierszu repozytorium) dodawany jest znak specjalny „@” – co oznacza, że dane pole jest kluczem obcym do innej „tabeli”.

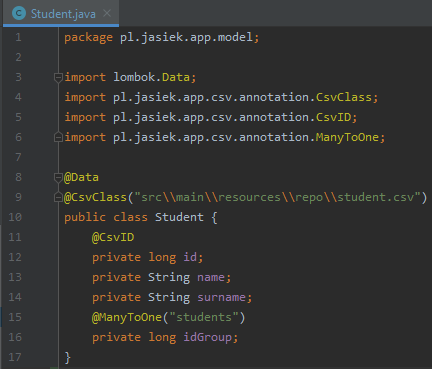
Wiele do wielu:

Jeżeli na danym polu występuje relacja wiele do wielu pole nie jest dodawane do pierwszego wiersza w repozytorium. Natomiast tworzona jest „tabela” pośrednia, która przechowuje klucze obce dwóch „tabel” połączonych ze sobą relacją wiele do wielu. Adnotacja *@ManyToMany* zawiera dwie wartości:

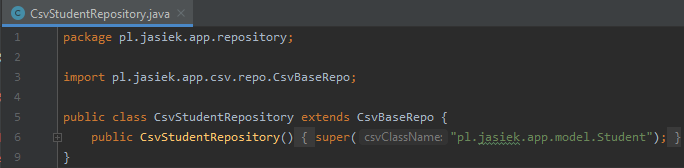
* relationshipCsvClass – przechowuje nazwe klasy (oznaczonej adnotacją *@CsvClass*)
* intermediateTable – przechowuje nazwę dla repozytorium pośredniego

# Jak korzystać z Mappera?

1. Na początku użytkownik musi dołączyć bibliotekę mappera do swojego projektu poprzez dodanie pliku jar mappera.
2. Tworzenie modelu – użytkownik powinien stworzyć co najmniej jedną klasę z adnotacją *@CsvClass*. Przykład:



1. Tworzenie repozytorium dla modelu – użytkownik powinien stworzyć co najmniej jedno repozytorium, które będzie rozszerzać klasę *CsvBaseAbstract*. Rozszerzając tą klasę musimy zaimplementować konstruktor zawierający wywołanie konstruktora z klasy bazowej poprzez słowo kluczowe *super* przekazując w parametrze nazwę klasy modelowej (wraz z nazwą pakietu), której odpowiada dane repozytorium. Oczywiście liczba repozytoriów jest równa liczbie klas modelowych.

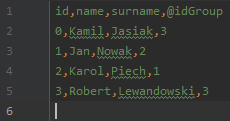


1. Ostatnim krokiem do użycia biblioteki jest odpowiednia inicjalizacja mappera – najlepiej w sekcji konfiguracyjnej lub inicjalizacyjnej projektu. Inicjalizacja następuje poprzez wywołanie metody *init()* z klasy *CsvMapperInitializer*:

*CsvMapperInitializer.init();*

Podczas wywołania tej metody są tworzone pliki repozytoriów w formacie *csv*. Przykładowe repozytorium dla klasy *Student* (bez danych i z danymi):





Wykonanie poprzednich kroków daję użytkownikowi możliwość korzystania z mappera, a tym samym ułatwienie w pisaniu funkcji do:

* Dodawania itemu do repozytorium (wraz z generowaniem ID),
* Usuwanie itemu z repozytorium,
* Modyfikacji itemu z repozytorium,
* Odczyt wszystkich itemów z repozytorium,
* Odczyt pojedynczego itemu z repozytorium.

Aby korzystać z powyższych funkcji wystarczy tylko utworzyć obiekt danego repozytorium. Przykład:



# Co nie zostało zrobione

* Przetwarzanie dziedziczenia
* Wygenerowanie pliku *JAR* dla mappera (na ten moment jest to oddzielny pakiet w projekcie)
* Przechwytywanie konkretnych wyjątków dla mapowania i ich zapis w pliku *log.txt*