Zadanie:

Zadanie polega na stworzeniu bazy danych w pamięci zapewniającej efektywny dostęp do danych – baza osób.

Na kolejnych zajęciach projekt będzie rozwijana i uzupełniana o kolejne elementy omawiane na wykładzie.

Na obecnych zajęciach poznasz:

1) Podstawy programowania obiektowego

2) Wzorzec fabryki – oraz cel jego stosowania

3) Zagadnienia określania odpowiedniego operatora widoczności/zasięgu

4) Manipulacje ze strukturami danych oraz dobór struktury do problemu

Zadanie 1)

Stwórz pakiet

baza.element – pakiet będzie zawierał obiekty składowe przechowywane w bazie

w pakiecie *baza.element* stwórz klasę osoba. Klasa powinna mieć pola do przechowywania imienia, nazwiska oraz daty urodzenia (wykorzystaj klasę LocalDate). Do każdego z pól powinien być dostęp za pomocą odpowiednich getterów i setterów.

Klasa powinna mieć polę idOsoby –tj. numer identyfikujący osobę, do którego powinien być dostępny jedynie getter!!!

Klasa powinna przesłaniać metodę toString, tak aby wyświetlały się informacje o użytkowniku. Zwróć uwagę na wartości null.

Klasa powinna posiadać dwa konstruktory. Konstruktory powinny być dostępne jedynie z poziomu tego samego pakietu. Konstruktory powinny przyjmować id – uwaga pole id powinno być niezmienne!!! Oraz drugi konstruktor powinien inicjalizować obiekt imieniem i nazwiskiem oraz odpowiednim id osoby

Zadanie 2)

Wygeneruj lub stwórz interfejs IOsoba na podstawie klasy Osoba – (możesz wykorzystać w tym celu polecenie Refactor -> Extract Interface

Zadanie 3)

Stwórz test jednostkowy z wykorzystaniem polecenia Tools -> Create/Update Tests

Przetestuj program na ewentualność występowania wartości null w imieniu i nazwisku i dacie w szczególności dla metody toString, oraz dla getterów i setterów

Zadanie 4)

Stwórz fabrykę osób .

Fabryka to klasa zawierająca metody służące do generacji obiektów danej klasy. Dzięki temu użytkownik w programie nie stosuje słowa *new* jak i nie wywołuje jawnie konstruktora obiektu. Dzięki temu programista – twórca biblioteki ma możliwość podmiany tworzonej przez fabrykę klasy, czyli może łatwo dokonać jej reimplementacji/podmiany w zależności od potrzeb/sytuacji/w miarę rozwoju projektu. Pamiętaj że fabryka powinna każdorazowo nadawać unikatowe ID dla każdego nowego użytkownika. Fabrykę można utworzyć poprzez metody statyczne ale również w postaci zwykłych metod związanych z obiektem (nie z klasą). Zastanów się nad wadami i zaletami poszczególnych rozwiązań.

PYTANIE: W jakim pakiecie należy umieścić fabrykę osób?

Fabryka powinna generować obiekty klasy IOsoba. Powinna ona udostępniać dwie metody getOsoba oraz getOsoba(imie,nazwisko).

Zadanie 5)

Stwórz pakiety

baza – pakiet ten będzie zawierał elementy związane z bazą osób

Stwórz klasę *BazaOsob* -

Klasa *BazaOsob* powinna oferować funkcjonalność:

1) Dodawanie nowej osoby do bazy – metoda *addOsoba(osoba)*
2) Odczytywanie osoby po *id* –metoda *getOsoba(id)*
3) Odczytywanie osoby po *nazwisku* –metoda *getOsoba(nazwisko)*
4) Zmianę danych osoby w bazie – metoda *changeOsoba(osoba)*
5) Metodę *reset()*- służącą do resetu bazy – usuwającą wszystkie osoby z bazy i zerującą jej zawartość
6) *Metodę remove(id)* –służącą do usunięcia z bazy osoby o określonym *id*
7) Matodę zwracającą liczbę osób w bazie *size()*
8) Iterator po osobach z bazy

Uwaga dot. iteratorów – pozwalają one w łatwy sposób trawersować strukturę danych zawierającą określone elementy.

Iterator w Javie powinien implementować interfejs Iterator<T> oferując następujące metody:

1) boolean hasNext() – zwraca informację czy istnieją nowe elementy na liście

2) T next() – zwracająca referencję kolejnego elementu T przechowywanego na liście

3) void remove() – metoda pozwalająca usunąć obecnie analizowany obiekt T

Do implementacji iteratora albo stwórz nową klasę albo skorzystaj z klas anonimowych

UWAGA DO BAZA OSÓB:

Zastanów się jak efektywnie zaimplementować wyszukiwanie osób po *id* i po *nazwisku.*

W tym celu zrób test. Ze strony prowadzącego pobierz klasę StringGenerator oraz TestWydajności . Pierwsza klasa służy do generacji losowych napisów, druga służącą do testowania. W klasie TestWydajności zaimplementuj brakujące metody i sprawdź ile czasu zajmuje bezpośrednio wyszukiwanie napisu w liście a ile z wykorzystaniem odpowiednich struktur danych. Jaka to struktura?

Zadanie 6)

Stwórz test jednostkowy weryfikujący poprawność bazy – sprawdź czy poprawnie działają poszczególne metody

Zadanie 7)

Dodaj możliwość zapisu i odczytu bazy (utrwalenia bazy). Zastanów się jak to efektywnie zaimplementować. Czy lepiej zrobić obiekt służący do zapisu bazy, czy może metodę statyczną. Docelowo należy wziąć pod uwagę kilka faktów tj:

a) Możliwość wystąpienia różnych form zapisu – plik CSV, XML, Baza Danych, ….

b) Całość powinna być jak najbardziej ogólna – zastanów się nad API

UWAGA: bazę zapisz w postaci prostego pliku CSV, w którym nagłówki odpowiadają poszczególnym polom w obiekcie.

Zadanie 8)

Przerób klasę BazaOsób, na postać umożliwiającą przechowywanie w niej dowolnego typu obiekty – stwórz klasę generyczną. Zastanów się jak to zrobić tak aby całość poprawnie działała.

Zadanie 9)

Stwórz nową implementację pozwalającą na utrwalenie bazy w postaci pliku XML. Zastanów się nad API.

Zadanie 10)

Zrób zadanie dot. wielowątkowości

Zadanie 11)

Zrób zadanie dot. RMI

Zadanie 10)

Wykorzystaj obiekt BazaOsób do stworzenia komunikatora dodatkowo wykorzystując RMI i wielowątkowość w tym celu:

a) Stwórz nową klasę pochodną do Osoba np. OsobaKomunikator, która dodatkowo będzie posiadała dwie metody: jedną pozwalającą na dodanie wiadomości dla tej osoby, a drugą na odczytanie wiadomości wysłanej dla danej osoby.

Uwagi:

- Możesz się tutaj wspomóc np. kolejką i w tym celu wykorzystać np. LinkedList, tak iż nowa wiadomość będzie dodawana na koniec kolejki a pierwsza wiadomość z kolejki będzie odsyłana

- Pamiętaj aby obsługa kolejki była synchroniczna tj. aby można było dodawać do kolejki i odczytywać z kolejki w wielu wątkach

- Odczytywanie z kolejki powinno być blokujące tj. jeśli wątek odczytujący nie ma nic do przeczytania to powinien przejść w stan uśpienia na tak długo aż w kolejce pojawi się wiadomość

- Dodawanie do kolejki powinno iść automatycznie bez oczekiwania (ale całość kolejki musi być sychroniczna)

b) Dokonaj adaptacji fabyrki osób, tak aby fabryka obecnie zamiast obiektów klasy Osoba zwracała OsobaKomunikator (zwróć uwagę jak to łatwo zrobić w programie który korzysta z fabryk)

Uwaga:

- Fabryka też będzie wykorzystywana przez wiele wątków, więc zastanów się czy nie będzie problemów w systemie wielowątkowym, jeśli tak, to jakie i jak się zabezpieczyć

c) Stwórz interfejs RMI do obsługi bazy

```
public interface UserService extends Remote {
    int addUser(String imie, String nazwisko) throws RemoteException;
    String getMessage(int userId) throws RemoteException;
    void setMessage(int userId, String message) throws RemoteException;
    List<String> getUsers() throws RemoteException;
    List<String> getUsers(String name) throws RemoteException;
}
```

Następnie dokonaj jego implementacji

UWAGI

- Pamiętaj, że RMI dla scenariusza wielowątkowego wymaga odpowiedniej synchronizacji. Bez odpowiedniej adaptacji klasy BazaOsob RMI nie będzie działał: podaj kiedy to może nastąpić

d) Dokonaj implementacji klienta RMI

UWAGI:

- pamiętaj, że polecenie getMessage() jest blokujące, co więc trzeba zrobić aby klient mógł działać jak komunikator.

e) Popraw i zaadaptuj komunikator wg. własnych potrzeb: np. dodaj uwierzytelnianie w pierwszej kolejności a potem pozwól pisać jedynie uwierzytelnionym użytkownikom. ….

Zaimplementuj operacje JOIN dla klas BazaOsób – jak to efektywnie zaimplementować?

W tym celu stwórz klasę Tools zawierającą metodę statyczną JOIN przyjmującą jako argument dwa obiekty BazaOsob i zwracającą połączoną bazę.

W tym celu wczytaj podane przez prowadzącego dwie bazy i stwórz jedną, zawierającą jedynie elementy unikatowe.