KOLOKWIUM nr 1 część praktyczna

studia zaoczne

dr inż. Łukasz Sosnowski
WIT Wyższa Szkoła Informatyki Stosowanej i Zarządzania
pod auspicjami Polskiej Akademii Nauk

Instrukcja: Pobierz udostępniony plik wit-zaoczne-kolokwium1.zip z systemu UBI z podkatalogu dzisiejszych zajęć (LabX) oraz podkatalogu kolokwium.

Rozpakuj plik do swojego ustalonego katalogu na projekty z przedmiotu Język Java.

Po rozpakowaniu zaimportuj projekt do ECLIPSE. Usuń plik ZIP.

Na wykonanie zadania jest łącznie 60 minut (test + część praktyczna). Po wykonaniu zadania praktycznego należy zapisać wszystkie pliki, zamknąć IDE oraz spakować katalog projektu ze wszystkimi plikami i katalogami. Plik ZIP ma mieć nazwę: Nazwisko_Imię_NRIndeksu.zip. Spakowany projekt należy odesłać w wyznaczonym czasie na mailem z tematem: Kolokwium na adres e-mail: lukasz.sosnowski@wit.edu.pl

Godzina zakończenia kolokwium zostanie podana przez prowadzącego przed rozpoczęciem kolokwium. Odesłanie pliku po czasie dyskwalifikuje tę część kolokwium.

Łączna punktacja za kolokwium 1: ½ * (liczba punktów z testu) + 1/2*(liczba punktów z części praktycznej). Maksymalna suma punktów z całego kolokwium: 20 pkt. Sumy z poszczególnych części również po 20 pkt (z wagą ½)

POWODZENIA!

1 Główny cel zadania

Do wykonania jest program, który umożliwi przechowywanie i przetwarzanie danych studentów w poszczególnych grupach studenckich. Program będzie pewnego rodzaju elektronicznym dziennikiem, umożliwiającym ewidencję studentów w danej grupie oraz wpisywanie danych o liczbie zdobytych punktów w ramach następujących kryteriów: aktywność, praca domowa, projekt, kolokwium 1, kolokwium 2, egzamin. Do wykonania zadania została udostępniona hierarchia dziedziczenia klas którą należy użyć. Nie można w niej zmieniać zadeklarowanych elementów (chyba że jest to jawnie podane w poleceniu), ale można dodawać nowe. Rezultat zadania trzeba pokazać za pomocą testów jednostkowych.

2 Dodaj komentarze (1 pkt)

We wszystkich klasach dodaj komentarze do:
a) klas podając po zmiennej @author swoje imię i nazwisko,
b) zmiennych (komentarze 1 liniowe)
c) metod komentarze generowane automatycznie z wyszczególnieniem parametrów i zwracanych wartości (poza getterami i setterami).

Komentarze należy dodać do elementów które są tam zdefiniowane oraz które będą dodane w ramach realizacji zadania.

3 W klasie AbstractPerson: (1 pkt)

a) Wygeneruj gettery do wszystkich zmiennych składowych.

b) Zaimplementuj konstruktor.

4 W klasie AbstractGroup: (1 pkt)

a) zdefiniuj wymagane zmienne składowe do przechowywania kodu grupy oraz nazwy specjalizacji w taki sposób aby można było zaimplementować metody abstrakcyjne w klasie pochodnej. Rozważ zmianę poziomu dostępu zmiennych na „chronione” jeśli to będzie potrzebne.

b) rozbuduj konstruktor o wymagane parametry oraz zaimplementuj go.

5 W klasie Criteria: (1 pkt)

Zdefiniuj wartości tablicy reprezentujące następujące kryteria oceny:

aktywność,
praca domowa,
projekt,
kolokwium 1,
kolokwium 2,
egzamin.

6 Dodaj klasę WITGroup (1 pkt)

a) Dodaj klasę WITGroup dziedziczącą po AbstractGroup.

b) Zaimplementuj wymagane metody oraz konstruktor.

7 Dodaj klasę StudentException (1 pkt)

a) Dodaj StudentException dziedziczącą po klasie Exception.

b) Zaimplementuj w niej 2 konstruktory. Jeden przyjmujący parametr typu String, drugi przyjmujący dwa parametry, pierwszy typu String, drugi typu Exception i wywołaj w ich implementacji konstruktory klasy bazowej.

8 W klasie Student: (7 pkt)

a) Zaimplementuj konstruktor 2 argumentowy dodając w jego implementacji inicjalizację mapy `mapPoints`

b) Wykonaj przeciążenie konstruktora, tak aby można było przy tworzeniu obiektu przekazać wszystkie dane ewidencyjne (imię, nazwisko, grupa, numer indeksu). Zaimplementuj ten konstruktor.

c) Zaimplementuj metodę getStudentData() w taki sposób, aby metoda zwracała 1 linię tekstu zawierającą: Nazwisko, imię, nrIndeksu, kodGrupy, specjalizacja, kryterium1:punkty, …, kryterium2:punkty.

d) Dodaj i zaimplementuj metodę publiczną nie zwracającą wartości, o nazwie *addPoints*. Metoda ma przyjmować dwa parametry. Pierwszy typu String, drugi typu byte. Pierwszy parametr stanowi kryterium oceny, a drugi przedstawia wartość punktową kryterium. Zapewnij aby przekazany parametr był jednym ze zdefiniowanych w klasie Criteria i tablic criteriaArr;

Uwzględnij w implementacji następujące warunki walidacyjne: Kryteria aktywność oraz praca domowa przyjmują wartości punktowe z zakresu <0,5>. Kryteria kolokwium 1 i kolokwium 2 przyjmują wartości punktowe z zakresu <0,20>. Kryterium projekt przyjmuje wartości z zakresu <0,10> oraz kryterium egzamin przyjmuje wartości z zakresu <0,40>.

W przypadku danych punktowych spoza zakresu metoda ma generować wyjątek klasy StudentException. Jeśli warunki walidacyjne są ok, to parametr pierwszy metody ma być dodany do mapy jako klucz, a parametr drugi jako wartość.

e) Wygeneruj gettery dla tej klasy

f) Zdefiniuj metodę createStudent, zwracającą nowy obiekt typu Student. Metoda ma przyjmować parametry niezbędne do utworzenia obiektu studenta (dane ewidencyjne bez mapy kryteriów i punktów)

9 W klasie StudentDatabase (4 pkt):

a) Zaimplementuj metodę addStudent w taki sposób aby dodać do zbioru studentów tylko wtedy gdy parametr jest różny od null. W takiej sytuacji powinien być generowany wyjątek StudentException. Zmodyfikuj stosownie do tego deklarację metody.

b) Zaimplementuj metodę searchStudents zwracającą zbiór zmiennych do obiektów studentów spełniających zadane kryteria w parametrach. Dla każdego parametru jeśli przekazano wartość null, to nie bierze on udziału w selekcji obiektów. W szczególności jeśli wszystkie parametry zostały przekazane jako null to mają być zwrócone wszystkie obiekty. Wartości przekazane w parametrach typu String mają wykazywać częściową zgodność z wartościami obiektów (pokrywać się z początkiem łańcucha znaków). Parametr atLeastPoint dotyczy co najmniej zadanej sumy punktów ze wszystkich kryteriów dla danego studenta.

c) Dodaj getter do zmiennej setStudents.

10 Dodaj klasę testu jednostkowego StudentTest (3 pkt):

a) Wykonaj testy jednostkowe dla klasy Student dla metod: *addPoints, createStudent, getStudentData*