KOLOKWIUM nr 1 część praktyczna, specjalizacja IO

studia zaoczne

dr inż. Łukasz Sosnowski
Akademia WIT
pod auspicjami Polskiej Akademii Nauk

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

Rozpakuj plik na PULPIT. Wyedytuj plik pom.xml projektu nadając nazwę dla projektu (tag „artifactId”) jako IMIE_NAZWISKO_NRIDX zamiast aktualnej wartości „`wit-zaoczne-kolokwium1łąęćź”`,

gdzie: IMIE to Twoje imię pisane bez znaków diakrytycznych, NAZWISKO to Twoje nazwisko pisane bez znaków diakrytycznych, NRIDX to Twój numer indeksu.

Po zapisaniu pom.xml zaimportuj projekt do ECLIPSE. Usuń plik ZIP.

Na wykonanie zadania jest łącznie 60 minut (test + część praktyczna). Po wykonaniu zadania praktycznego należy: spakować katalog projektu ze wszystkimi plikami i katalogami. Plik ZIP ma mieć nazwę: Nazwisko_Imię_NRIndeksu.zip. Spakowany projekt należy odesłać poprzez zadanie TEAMS w wyznaczonym czasie.

Godzina zakończenia kolokwium zostanie zdefiniowana w zadaniu TEAMS oraz INSPERA. Po upływie wyznaczonego czasu nie ma możliwości odesłania plików.

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 Kontekst zadania

Do wykonania jest program, który umożliwi przechowywanie i przetwarzanie danych o modelach pojazdów poszczególnych typów (np. ciężarowe, autobusy, osobowe, motocykle). Program będzie pewnego rodzaju bazą danych o pojazdach, umożliwiającą ewidencję pojazdów dla danego typu oraz wpisywanie wartości parametrów w ramach następujących kryteriów: typ_silnika, pojemność, kolor, moc, max moment obrotowy, typ

nadwozia. Do wykonania zadania został udostępniony projekt mavenowy który należy użyć. Nie można w niej zmieniać zadeklarowanych elementów, chyba że instrukcja dla danego elementu o tym mówi. Można dodawać nowe elementy do klas i metod jeśli jest to uzasadnione. Rezultat zadania trzeba pokazać za pomocą testów jednostkowych.

2 Dodaj komentarze (1 pkt)

We wszystkich klasach dodaj komentarze do:
a) klas i interfejsów 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 już istniejących w przekazanych klasach oraz nowych, utworzonych w ramach realizacji zadania kolokwium.

3 W pakiecie pl.wit.kolokwium1.types dotyczącym typów pojazdów: (3 pkt)

a) w klasie *AbstractVehicleType* zaimplementuj metodę „*toString*" w taki sposób aby zwracała obiekt klasy *String* zawierający *id* typu pojazdu i nazwę typu.

b) w klasie Motorcycle*Type* zaimplementuj metodę:

- „*getVehicleTypeId*" poprzez zastąpienie „return null" implementacją zwracającą identyfikator typu „motorcycle" z klasy *EnVehicleTypeCodes*

- „*getVehicleTypeName*" poprzez zastąpienie „return null" implementacją zwracającą nazwę typu „motorcycle" z klasy *EnVehicleTypeCodes* używając wskazanej stałej wyliczeniowej

c) stwórz klasę w Bus*Type* dziedziczącą po klasie *AbstractVehicleType* i zaimplementuj wymagane metody analogicznie jak w pkt b) jednakże w tym przypadku w odniesieniu do stałej wyliczeniowej „bus"

4 W pakiecie pl.wit.kolokwium1.enums: (2 pkt)

a) w klasie *EnBodyType* dodaj stałe wyliczeniowe reprezentujące typ nadwozia jako: chaser, chopper, articulated, onepiece. Stałe oznaczają odpowiednio: ścigacz, chopper, przegubowy, jednoczęściowy (natomiast nie należy w tym przypadku dodawać ich jako danych stowarzyszonych ze stałymi).

b) w klasie *EnEngineTypes* dodaj 3 stałe wyliczeniowe reprezentujące rodzaje silnika jako: diesel, benzynowy, elektryczny. Użyj nazw angielskich do zdefiniowania stałych, ale

dodatkowo przechowaj w zmiennej typu String powiązanej z daną stałą nazwy polskie podane powyżej. Zdefiniuj do tego wymagane zmienne i metody w klasie *EnEngineTypes*;

5 W pakiecie pl.wit.kolokwium1.exceptions: (1 pkt)

Utwórz klasę wyjątku o nazwie *BodyTypeException* dziedziczącą po klasie *Exception*. Zaimplementuj w niej 2 konstruktory. Jeden przyjmujący parametr typu String, drugi przyjmujący dwa parametry, pierwszy typu String, drugi typu *Exception*. Oba konstruktory w swej implementacji mają wywoływać odpowiednie konstruktory klasy bazowej zgodne z sygnaturami tychże konstruktorów.

6 W klasie AbstractVehicle: (7 pkt)

a) zaimplementuj interfejs *IVehicle*

b) zdefiniuj 7 składowych klasy (zmiennych) prywatnych odpowiadających za przechowywanie: typu pojazdu, typu silnika, pojemności silnika, koloru, mocy silnika, momentu obrotowego silnika, typu nadwozia. Nazwy zmiennych oraz ich typy mają być odpowiednie dla metod zdefiniowanych przez interfejs *IVehicle*

c) dodaj w konstruktorze inicjalizację każdej z tych siedmiu zmiennych w zaznaczonym w komentarzu miejscu

d) dodaj do konstruktora deklarację pozwalającą na generowanie wyjątków typu *BodyTypeException* przez ten konstruktora

e) jako ostatnie instrukcje konstruktora dodaj sprawdzenie warunku czy pojazd ma zdefiniowany zakazany typ nadwozia (metoda abstrakcyjna w tej klasie). Jeśli tak to należy wtedy wygenerować wyjątek z komunikatem: „Błędny typ nadwozia dla tego pojazdu!”

f) wygeneruj gettery dla siedmiu dodanych zmiennych składowych klasy

g) zaimplementuj metodę *forbiddenTypesToString* aby zwracała w postaci String’a zakazane typy. Wykonaj to za pomocą pętli „for”.

7 W pakiecie pl.wit.kolokwium1 (3 pkt)

a) Dodaj klasę Bus dziedziczącą po *AbstractVehicle* i zaimplementuj konstruktor wywołujący konstruktor z klasy bazowej lecz nie przekazując pierwszego argumentu konstruktora klasy bazowej z parametru, lecz tworząc nowy obiekt klasy BusType (analogicznie jak jest to zrobione w klasie Motorcycle). Ponadto zaimplementuj metodę *getForbiddenTypes* tak aby zabraniała ona definiowania typów nadwozia innych niż *articulated* i *onepiece* dla obiektów tej klasy.

b) W klasie Motorcycle zdefiniuj zmienną składową klasy (prywatną) przechowującą informację czy motocykl posiada boczne kufry, nazwa zmiennej: *sidePocketsExist*, typ: Boolean i przy deklaracji zainicjuj ją poprzez „null". W konstruktorze zdefiniowanym w w tej klasie dodaj kolejny parametr *sidePocketsExist* i dopisz kod obsługujący ustawianie zmiennej składowej klasy. Wygeneruj getter dla tej zmiennej

c) W klasie Motorcycle dodaj do konstruktora deklarację umożliwiającą generowanie wyjątków typu *BodyTypeException* oraz zaimplementuj metodę *getForbiddenTypes* tak aby zabronić dla motocykli definiowania typu: *articulated* i *onepiece*.

8 W klasie VehiclesDatabase: (1 pkt)

a) zaimplementuj metodę *addVehicleToDb* tak aby dodawała w bezpieczny sposób przekazany obiekt do zbioru setVehicles.

9 W klasie testu jednostkowego VehiclesDatabaseTest (2 pkt):

a) zaimplementuj test *dbSearchAllTest* zgodnie z wzorcem AAA, sprawdzający czy wyszukiwanie zwróci wszystkie elementy zbioru obiektów jeśli metoda *searchDb* z klasy *VehiclesDatabase* zostanie wywołana ze wszystkimi parametrami ustawionymi jako null

b) zaimplementuj test *dbSearchBodyTypeTest* zgodnie z wzorcem AAA, sprawdzający czy wyszukiwanie zwróci obiekty motocyklów dla zadanego w teście typu nadwozia (np. chopper oraz koloru „czarny").