B-OOP: Úloha č. 6

Pozorne si najprv prečítajte celé zadanie!

Máte za úlohu implementovať informačný systém pre dopravný inšpektorát, ktorý má na starosti registráciu nových vozidiel do databázy. Zamestnanci dopravného inšpektorátu musia pri každej registrácii rozhodnúť, či je dané vozidlo ťažké (nad 3.5 tony) alebo ľahké (pod 3.5 tony). Musia tiež rozhodnúť, či ide o osobný automobil, motocykel alebo kamión. Každé vozidlo má vlastníka, ktorý má meno a adresu trvalého bydliska. Niekedy sa stane, že zamestnanci inšpektorátu musia do databázy zaradiť ťažko kategorizovateľné vozidlo, napríklad vznášadlo či tank. Takéto vozidlo registrujú do databázy s použitím všeobecnejšej kategórie, ktorá ho najviac vystihuje. Napríklad, tank by registrovali ako ťažké vozidlo.

Vytvorte triedu **Task6** v balíčku **sk.stuba.fei.uim.oop**. V tejto triede vytvorte metódu **main** s korektnou signatúrou. V tejto metóde:

- vytvorte inštanciu triedy **Database** s kapacitou 10 vozidiel;
- vytvorte niekoľko inštancií vozidiel rôznych typov a pridajte ich do databázy;
- vypíšte údaje o všetkých registrovaných vozidlách. Využite metódu **toString** triedy **Database**.

Vytvorte triedu **Database** v balíčku **sk.stuba.fei.uim.oop.database**. V tejto triede vytvorte:

- privátnu premennú **registeredVehicles** typu **Vehicle[]**;
- privátnu premennú **registeredVehiclesCount** typu **int**, ktorá reprezentuje aktuálny počet registrovaných vozidiel. K tejto premennej implementujte aj prislúchajúcu getter metódu;
- parametrický konštruktor s parametrom **capacity** typu **int**, ktorý inicializuje pole **registeredVehicles** tak, aby malo kapacitu rovnú hodnote **capacity**, a premennú **registeredVehiclesCount** inicializuje na 0;
- metódu **register** s návratovou hodnotou **boolean** s parametrom **vehicle** typu **Vehicle**. Táto metóda pridá vozidlo do zoznamu registrovaných vozidiel (ak je na to ešte voľná kapacita), inkrementuje počítadlo a vráti **true**. Ak nie je dostačujúca kapacita, nevykoná žiadnu operáciu a vráti **false**;

- metódu **toString**, ktorá vráti formátované informácie o všetkých registrovaných vozidlách. Využite metódy **toString** jednotlivých vozidiel.

Implementujte triedu **Vehicle**. Od tejto triedy budú dediť triedy **LightVehicle** a **HeavyVehicle**, ktoré reprezentujú vozidlá s váhou pod 3.5 tony a nad 3.5 tony. Od triedy **LightVehicle** budú dediť triedy **Car** a **Motorcycle**. Od triedy **HeavyVehicle** bude dediť trieda **Truck**. Všetky tieto triedy vytvorte v balíčku **sk.stuba.fei.uim.oop.entity**.

V triede **Vehicle** vytvorte:

- privátnu premennú **owner** typu **Person** a zodpovedajúce getter a setter metódy. Zamyslite sa, v akej situácii by zamestnanci dopravného inšpektorátu mali použiť setter metódu na tejto premennej;
- privátnu konštantu **make** typu **String**, ktorá bude obsahovať názov výrobcu vozidla, a zodpovedajúcu getter metódu;
- parametrický konštruktor s parametrami **owner** typu **Person** a **make** typu **String**, pomocou ktorých sa nastaví zodpovedajúca premenná a konštanta;
- preťaženú metódu **toString**, ktorá vytlačí všetky informácie o tomto vozidle.

V triede **HeavyVehicle** vytvorte:

- privátnu premennú **height** typu **int**, ktorá bude reprezentovať výšku vozidla. Vytvorte príslušný getter;
- parametrický konštruktor s parametrom **height** typu **int**, ktorý nastaví zodpovedajúcu premennú, a s ostatnými parametrami potrebnými pre volanie konštruktora rodiča;
- preťaženú metódu **toString**, ktorá vytlačí všetky informácie o tomto vozidle. Využite metódu **toString** rodiča a pridajte dodatočné informácie špecifické pre túto triedu.

V triede **Truck** vytvorte:

- privátnu konštantu **maxLoadWeight** typu **int**, ktorá bude reprezentovať maximálnu hmotnosť nákladu kamiónu. Vytvorte príslušný getter;
- parametrický konštruktor s parametrom **maxLoadWeight** typu **int**, ktorý nastaví zodpovedajúcu konštantu, a s ostatnými parametrami potrebnými pre volanie konštruktora rodiča;
- preťaženú metódu **toString**, ktorá vytlačí všetky informácie o tomto vozidle. Využite metódu **toString** rodiča a pridajte dodatočné informácie špecifické pre túto triedu.

V triede **LightVehicle** vytvorte:

- privátnu konštantu **maxPassengerCapacity** typu **int**, ktorá bude reprezentovať maximálny počet prepravovaných osôb (vrátane vodiča). Vytvorte príslušný getter;
- parametrický konštruktor s parametrom **maxPassengerCapacity** typu **int**, ktorý nastaví zodpovedajúcu konštantu, a s ostatnými parametrami potrebnými pre volanie konštruktora rodiča;
- preťaženú metódu **toString**, ktorá vytlačí všetky informácie o tomto vozidle. Využite metódu **toString** rodiča a pridajte dodatočné informácie špecifické pre túto triedu.

V triede **Car** vytvorte:

- privátnu premennú **color** typu **Color** (môžete využiť triedu z knižnice AWT, ktorá bola demonštrovaná na seminári). Vytvorte príslušné getter a setter metódy;
- parametrický konštruktor s parametrom **color** typu **Color**, ktorý nastaví zodpovedajúcu premennú,a s ostatnými parametrami potrebnými pre volanie konštruktora rodiča;
- preťaženú metódu **toString**, ktorá vytlačí všetky informácie o tomto vozidle. Využite metódu **toString** rodiča a pridajte dodatočné informácie špecifické pre túto triedu.

V triede **Motorcycle** vytvorte:

- privátnu konštantu **type** typu **MotorcycleType**, ktorá bude reprezentovať typ motocykla. Vytvorte príslušný getter;
- parametrický konštruktor s parametrom **type** typu **MotorcycleType**, ktorý nastaví zodpovedajúcu konštantu, a s ostatnými parametrami potrebnými pre volanie konštruktora rodiča;
- preťaženú metódu **toString**, ktorá vytlačí všetky informácie o tomto vozidle. Využite metódu **toString** rodiča a pridajte dodatočné informácie špecifické pre túto triedu.

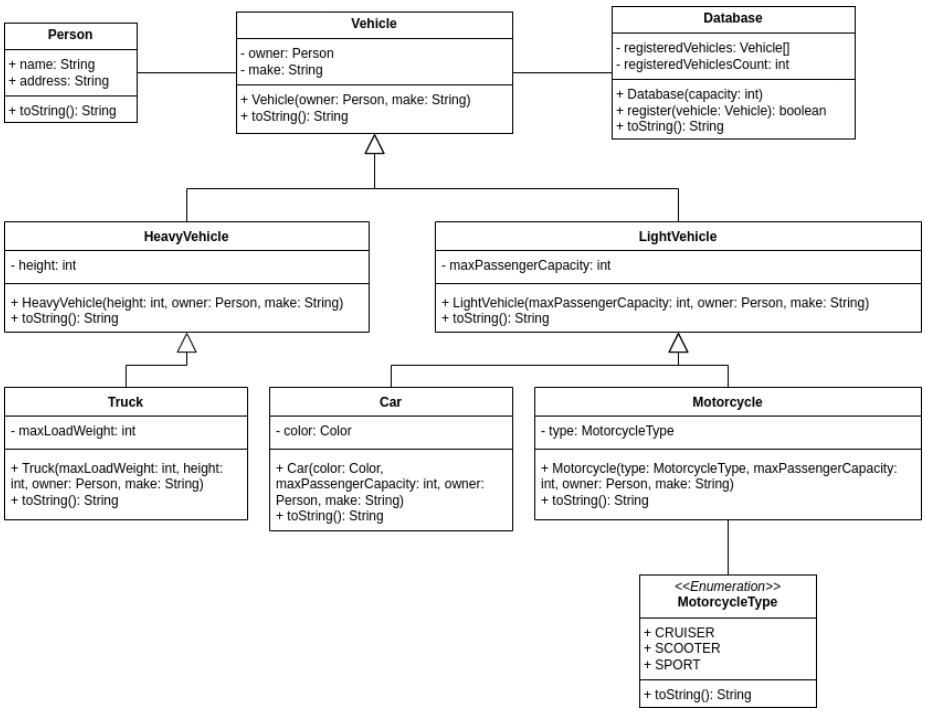
V balíčku **sk.stuba.fei.uim.oop.entity** vytvorte enumeráciu **MotorcycleType** s hodnotami **CRUISER**, **SCOOTER** a **SPORT**. Implementujte aj metódu **toString**.

Ďalej v tomto balíčku vytvorte triedu **Person**, ktorá bude obsahovať meno a adresu osoby. Implementujte aj metódu **toString**.

Zamyslite sa, či môžu niektoré triedy z hierarchie vozidiel byť abstraktné. Berte do úvahy všetky prípady použitia vášho softvéru, ktoré boli popísané v požiadavkách z úvodu tohto zadania.

Určite ste si všimli, že popis tohto zadania je veľmi dlhý. Avšak množstvo kódu, ktoré musíte napísať je relatívne malé. Tento nesúlad je spôsobený faktom, že prirodzený jazyk je na popis tried, metód, atribútov a vzťahov či hierarchie

ťažkopádny. V praxi sa používa na zachytenie týchto súvislostí UML diagram tried (UML class diagram). Zatiaľ nie je potrebné, aby ste do detailov rozumeli tomuto typu diagramu, no mali by ste mať aspoň znalosť, že existuje, a schopnosť čítať ho. Triedy, ktoré sme popísali v tomto zadaní, sú zachytené na obrázku 1.



Obr. 1: UML diagram popisujúci štruktúru a vzťahy tried zadania

V UML diagrame tried sú obdĺžniky, ktoré reprezentujú triedy, rozhrania (interface) alebo enumerácie. Obdĺžniky bývajú segmentované na 3 časti. Vrchná časť obsahuje názov objektu a prípadne aj jeho typ (ak ide o rozhranie alebo enumeráciu). Stredná časť popisuje atribúty objektu (premenné a konštanty). Spodná časť popisuje metódy objektu. Pred názvom atribútov a metód sa objavuje symbol označujúci ich modifikátor viditeľnosti:

- +, ak sú public
- −, ak sú private
- #, ak sú protected

Všimnite si, že dátový typ atribútu, resp. návratová hodnota metódy, sa nachádza za jej názvom. Napríklad, atribút uvedený v diagrame ako "- name: String" je privátna premenná typu **String** s názvom **name**. Metóda označená ako "+ register(vehicle: Vehicle): boolean" je verejná metóda, ktorá vracia **boolean** a berie jeden parameter typu **Vehicle** s názvom **vehicle**.

Getter a setter metódy sa často z UML diagramu tried vynechávajú, pretože ich pri veľkých triedach môže byť veľmi veľa. Nie je však chybou uviesť ich, ak to architekt softvéru považuje za nevyhnutné. Ide teda najmä o preferenciu toho, kto vytvára diagram. V uvedenom príklade diagramu sme ich pre účely kompaktnosti vynechali.

Vzťahy medzi objektami zachytávajú šípky a čiary. Šípka s trojuholníkovou hlavou reprezentuje dedenie. Čiara medzi objektami reprezentuje asociáciu. Ide o vzťah typu "jeden(viacero) objekt(ov) využíva jeden(viacero) iných objekt(ov)". Napríklad, trieda **Vehicle** používa triedu **Person**, pričom jedna osoba môže vlastniť viacero vozidiel. V UML diagrame tried existujú aj iné typy šípok, ktoré reprezentujú odlišné typy asociácií podľa toho, ako dlho žijú (aký je lifetime) zapojené objekty. Pre jednoduchosť ich však zatiaľ neuvedieme.

Do AIS odovzdajte zdrojové súbory (s príponou .java): ZIP súbor priečinku src.