**1. FELADAT**

személykocsi, teherkocsi modell

Hozz létre agy Vehicle osztályt. Az általános jármű osztálynak van önsúlya, és legalább egy vezető. Ezek adják a teljes súlyát.

* Vezess be egy vehicleWeight privát attribútumot, a jármű súlya.
* PERSON\_AVERAGE\_WEIGHT konstans érték: egy személy átlagos súlyát tartalmazza. Az értéke legyen 75.
* Hozz létre egy konstruktort, amely megkap egy értéket az attribútum számára, és beállítja azt.
* Szükséges metódus a getGrossLoad visszaadja a mozgó jármű súlyát. (A jármű súlyához adjuk hozzá a sofőr súlyát.)

Hozz létre egy Car osztályt, amely az általánosabb jarmű osztályból származik (Vehicle) a következők alapján

* tartalmaz egy attribútumot
* numberPassenger: egész szám, amely az utasok számát jelenti (sofőr nélkül)
* konstruktor, amely megkap két értéket az attribútumok számára, és beállítja azokat (az ős attribútumát a superhívással).
* getGrossLoad visszaadja a mozgó gépkocsi súlyát. Hívd az ős azonos nevű metódusát, és add hozzá az utasok súlyát (Ez lesz a visszatérési érték).
* Definiáld felül a toString metódust, a következő formára:
* Car{numberOfPassenger=4, vehicleWeight=1700}

Hozz létre egy Van osztályt, amely egy kisteherautót modellez. Ez az osztály az általános autóból származik, kiterjeszti a Car osztályt.

* Tartalmaz egy cargoWeight egész attribútumot a rakomány súlyának tárolására.
* Definiálj egy konstruktort, amely megkapja a rakománysúlyt, és az ős osztály két attribútumához szükséges értékeket is (összesen három egész érték). Hívd a super-t az ős attribútumok inicilizálására, és állítsa be az új attribútumot is.
* Definiáld felül a getGrossLoad metódust. Hívd az ős azonos nevű metódusát, és add hozzá a rakomány súlyát (Ez lesz a visszatérési érték).
* Definiáld felül a toString metódust, a következő formára:
* Van{cargoWeight=1222, numberOfPassenger=4, vehicleWeight=1200}

Virtuális metódusok használata (Mindig az hívódik, amelyikre szükség van.)

A komp FerryBoat képes bármilyen autót tárolni, aminek kisebb a súlya a megengedettnél. Viszont a komp is egy speciális jármű.

* Definiálj egy állandót MAX\_CARRY\_WEIGHT néven, amely tárolja a maximálisan szállítható autó súlyát.
* Ha szállít autót, akkor azt egy Car típusú car attribútumba tárold. (Ha nem szállít, akkor ez null.)
* Hozz létre egy konstruktort, amely megkap egy egész értéket, amivel meghívod super-t.
* Definiáld felül a getGrossLoad metódust. Hívd az ős azonos nevű metódusát, és add hozzá a szállított autó súlyát (Ez lesz a visszatérési érték).
* Definiálj egy canCarry metódust, ami igazat ad vissza, ha a paraméterben kapott autót szállíthatja, azaz a szállítandó autó súlya kisebb, mint a MAX\_CARRY\_WEIGHT (egyébként hamis). (Akár autó, akár kisbusz a referencia célja mindig a megfelelő metódus hívódik meg.)
* Definiálj egy load metódust, ami igazat ad vissza, ha a paraméterben kapott autót berakodta, azaz a súlya kisebb, mint a MAX\_CARRY\_WEIGHT (ekkor tárolja el az autót a referenciába).
* Definiáld felül a toString metódust, a következő formára FerryBoat= + a tárolt autó toString eredménye:
* FerryBoat{car=Van{cargoWeight=200, numberOfPassenger=1, vehicleWeight=1200}}
* FerryBoat{car=Car{cargoWeight=200, numberOfPassenger=1, vehicleWeight=1200}}

**2. FELADAT**

## Feladat - Flotta

A flottában vegyesen vannak teherszállító hajók (cargo ship), személyszállító hajók (liner) és kompok (ferry boat). A kompok személyeket és terhet is szállíthatnak. Ha a flotta behajóz, akkor folyamatosan töltik fel hajókat, mindaddig, amíg meg nem telnek, el nem fogy az utas, vagy teher.

Hozz létre egy Ship interfészt, mely a hajót jelöli, egy CanCarryGoods és CanCarryPassengers interfészt, mely azt jelöli, hogy egy hajó tud-e személyeket, vagy terhet szállítani.

A CanCarryGoodsBehaviour implementálja a CanCarryGoods interfészt. Ennek az osztálynak két attribútuma van: int cargoWeight és int maxCargoWeight. Hozz létre egy loadCargo metódust, melynek paramétere a betöldendó rakomány súlya, és a weight attribútumba eltárolja a letárolt rakományt, és visszatér le nem tárolt rakomány súlyával.

Vezessünk be egy CanCarryPassengersBehaviour osztályt, ami implementálja a CanCarryPassengers interfészt. Ennek az osztálynak két attribútuma van: int passangers és int maxPassengers. Hozz létre egy loadPassanger metódust, melynek paramétere a beszálló utasok száma, és a passangers attribútumba letárolja a beszállt utasok számát, és visszatér kintmaradó utasok számával.

A Liner osztály implementálja a Ship és a CanCarryPassengers interfészt, és legyen egy CanCarryPassengers típusú attribútuma.

A CargoShip osztály implementálja a Ship és a CanCarryGoods interfészt, és legyen egy CanCarryGoods típusú attribútuma.

A FerryBoat osztály implementálja a Ship, CanCarryGoods, CanCarryPassengers interfészt, és legyen mindkét típusú attribútuma.

Mindhárom osztálynak legyen olyan konstruktora, mely elvárja a CanXxxBehaviour osztályok konstruktorában elvárt adatokat.

Minden szükséges (interfész által kikényszerített) metódust úgy implementálj, hogy továbbítsd a kérést a megfelelő attribútumnak.

Azaz pl. a FerryBoat osztály loadCargo() metódusa hívja a CanCarryGoods loadCargo() metódusát.

Készíts egy Fleet osztályt, melynek van egy List<Ship> attribútuma, mely a hajókat tartalmazza. A lift() metódusa végigmegy a listán, és sorban feltölti a hajókat a személlyekkel és terhekkel. A maradék személyeket és terheket (melyek nem fértek el) a waitingPersons és waitingCargo attribútumokban tárolja el. Ez utóbbiakhoz generálj getter metódusokat is.