Kertész János (AM2VZ8) OEP 22.csoport 2023.05.14.

Feladat

Modellezzük egy **kamionos** cég működését!

A cég telephelyekkel (ezeket a címük azonosítja), kamionokkal és sofőrökkel rendelkezik. Egy kamionoknak ismert a rendszáma, a tengelyeire vetített terhelhetősége, fogyasztása (100km-en), tengelyeinek száma (nyergesvontató: 3, fülkésvázas: 2). A kamionok vagy úton vannak, vagy az egyik telephelyen állnak.

a. Melyik telephelyhez tartozik éppen most a legkevesebb kamion?
b. Mekkora terhet képesek egyszerre elfuvarozni egy adott telephely kamionjai?

Amikor egy kamionok egy fuvarra kap megbízást, akkor meg kell adni a fuvar távolságát, súlyát, díját (amit a cég kap a szállításért), az indulási dátumát, tervezet szállítási időt (órában), és a sofőrt. A fuvar teljesítése után ismert lesz annak érkezési ideje (addig ez egy extremális érték). Új megbízást nemcsak egy telephelyen állomásozó kamion kaphat, hanem egy már úton levő is, ha annak nincs teljesítendő fuvarja.

c. Volt-e olyan kamion, amely túl volt terhelve valamelyik fuvarban?
d. Mekkora a nyeresége a cégnek? (fuvar nyeresége = szállítási díj – üzemanyag költség – bér)?

A sofőrök között vannak kezdők, több éves gyakorlattal rendelkezők, illetve a cég törzsgárdájához tartozók. A sofőrök a teljesített fuvarjaik után kapnak bért, amelynek összege a fuvarral megtett távolságtól, a kamion típusától (nyerges, fülkés), illetve a sofőr besorolásától (kezdő, gyakorlott, törzstag) függ az alábbi módon: bér = távolság · együttható

együttható	kezdő	gyakorlott	törzstag
nyerges	25	35	40
fülkés	20	30	40

Készítsen használati eset diagramot! Ebben jelenjenek meg használati esetként a később bevezetett fontosabb metódusok. Adjon meg a fenti feladathoz egy olyan objektum diagramot, amely mutat két telephelyet, két sofőrt, három kamiont, és néhány fuvart.

Rajzolja fel a feladat osztály diagramját (először csak a konstruktorokkal)! Azoknak a privát/védett adattagoknak a láthatóságát, amelyekhez getter-t is, és setter-t is kell készíteni, jelölheti publikusnak. (A triviális getter/setter-eket később sem kell beírni a modellbe.)

Készítse el egy kamion objektum állapotgépét! Különböztesse meg a „telephelyen áll”, „fuvart teljesít”, „üresen közlekedik” (el a telephelyről, vagy vissza a telephelyre) állapotokat. Az állapot-átmeneteket megvalósító tevékenységeket majd az vontató osztály metódusaiként definiálhatja.

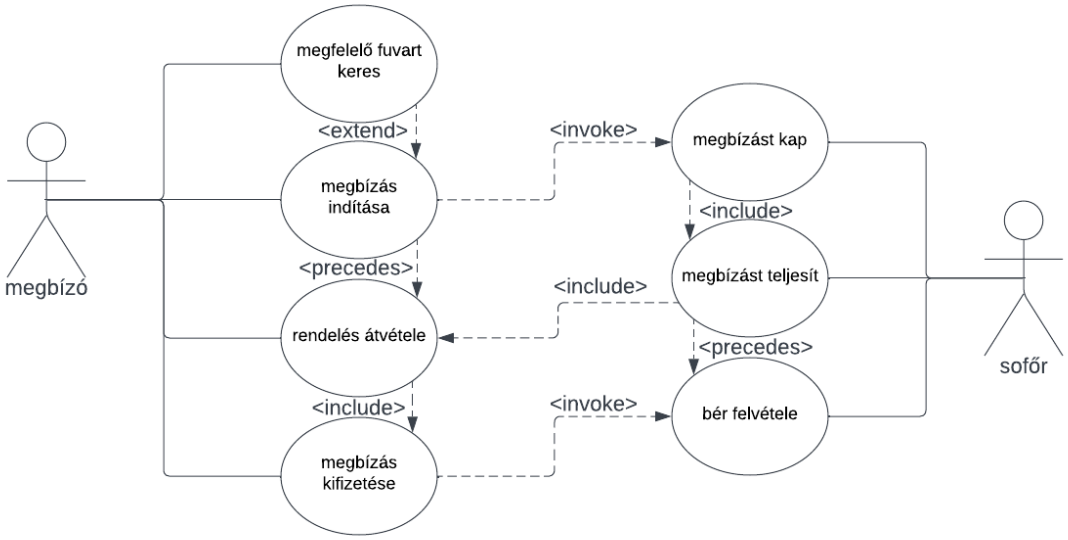
Egészítse ki az osztálydiagramot az objektum-kapcsolatokat létrehozó metódusokkal, valamint a feladat kérdéseit megválaszoló metódusokkal. A metódusok leírásában a félév első felében bevezetett végrehajtható specifikációs jelöléseket használja. Azoknak a konstruktoroknak a törzsét, amelyek kizárólag az adattagok inicializálását végzik, nem kell feltüntetni. Ilyenkor a konstruktor paraméterlistája helyén elég felsorolni az inicializálandó adattagok neveit. Az összes közvetlen (tehát nem szerepnév) adattag felsorolása helyett elég ”…”-ot írni.

Használjon tervezési mintákat, és mutasson rá, hogy hol melyiket alkalmazta.

Implementálja a modellt! Szerkesszen olyan szöveges állományt, amelyből fel lehet populálni egy cég telephelyeit, vontatóit, sofőrjeit, fuvarjait. Készítsen teszteseteket, néhánynak rajzolja fel a szekvencia diagramját, és hozzon létre ezek kipróbálására automatikusan tesztkörnyezetet!

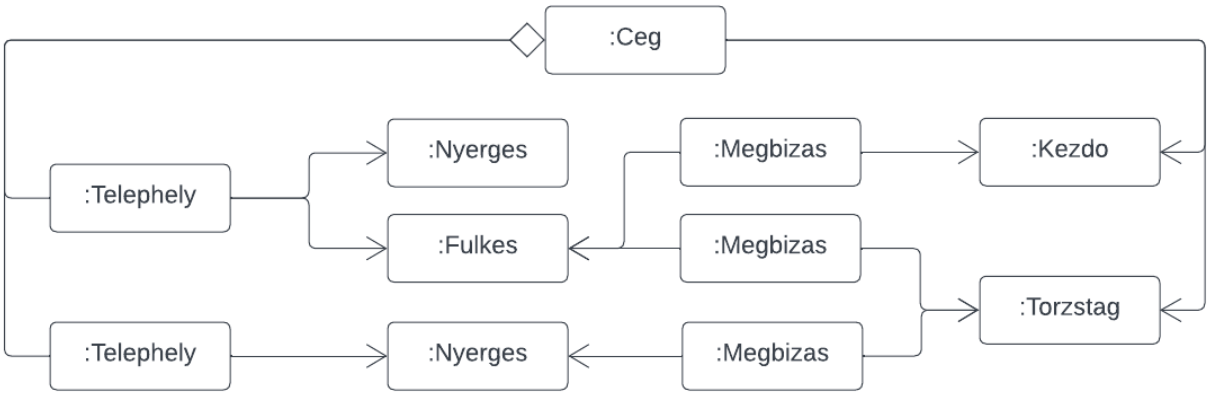
Terv

A feladat során a fő mozzanatokért a megbízó fél és a sofőr a felelős. „”Amikor egy kamionok egy fuvarra kap megbízást, akkor meg kell adni a fuvar távolságát, súlyát, díját (amit a cég kap a szállításért), az indulási dátumát, tervezet szállítási időt (órában), és a sofőrt.” Ezek kiválasztása mind a megbízó fél feladatköre. Elvégzése pedig a sofőré.



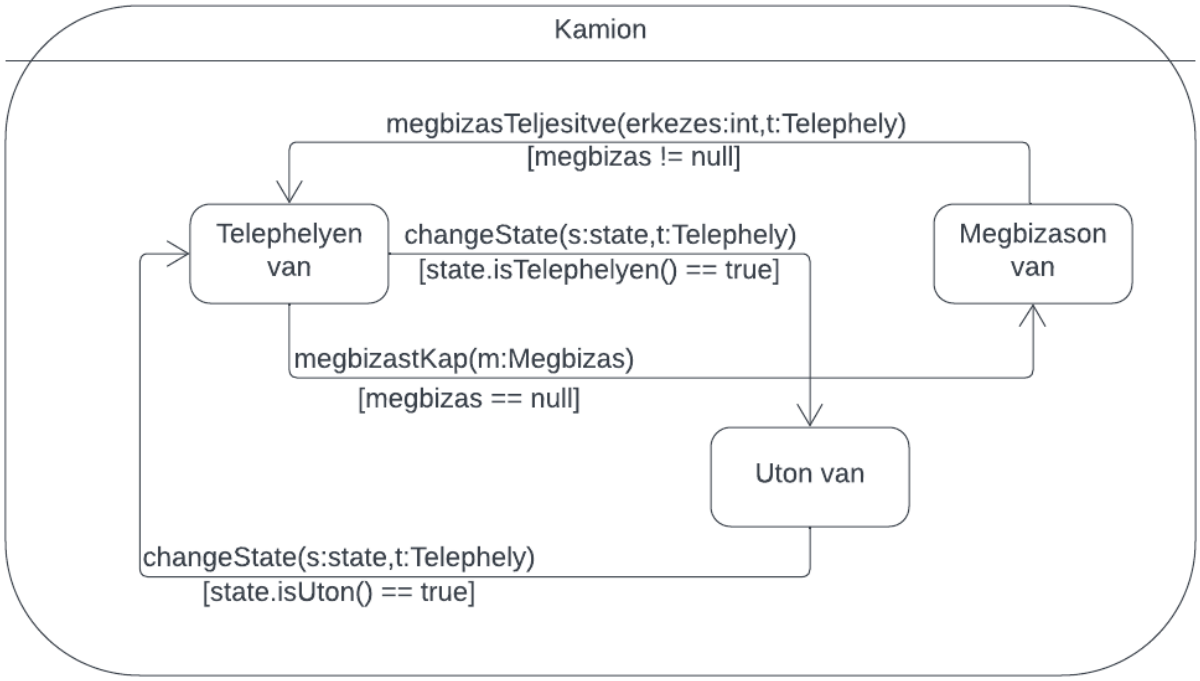
(használati eset diagram)

Az alábbi objektum diagrammon látható maga a cég, amely 2 telephellyel rendelkezik, az elsőn 2 kamion, a másodikon 1 állomásozik. A cég 3 megbízást kapott.



(objektum diagramm)

Egy kamionnak három állapota van: „telephelyen van”, „megbizason van”, „uton van” (ebben az utóbbi esetben nem teljesít megbízást, és úgy közlekedik pl.: tankolni/szervízbe megy). Amikor megkapja a megbízást a „telephelyen van” vagy az „uton van” állapotok egyikében van, az utóbbi esetben először vissza kell menjen a telephelyre hogy a megfelelő sofőr vezethesse majd ebből vált „megbizason van” állapotba. Amint a fuvart leadta, majd visszaért a telephelyre, a fuvart sikeresen teljesítette. Ezek után a „telephelyen van” állapotot veszi fel, ahonnan ez az egész indul elölről.

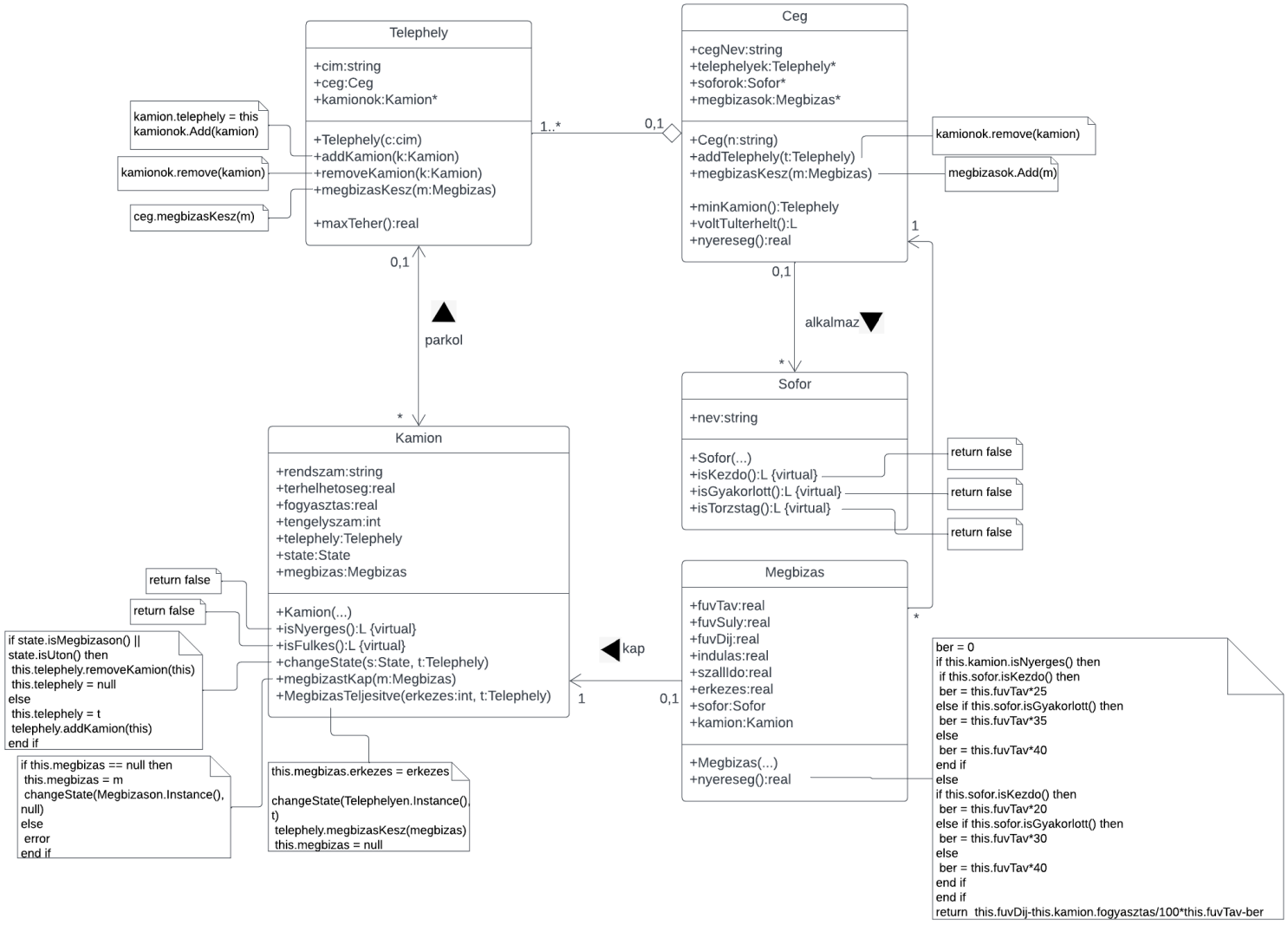


(kamion objektum állapotgépe)

A feladatban 5 fő objektum jelenik meg, maga a „Ceg” valamint a „Sofor” aki vezeti a „Kamion” amely kiszállítja a „Megbizas”-okat, valamint a „Telephely”, ahol a kamionok állomásoznak.

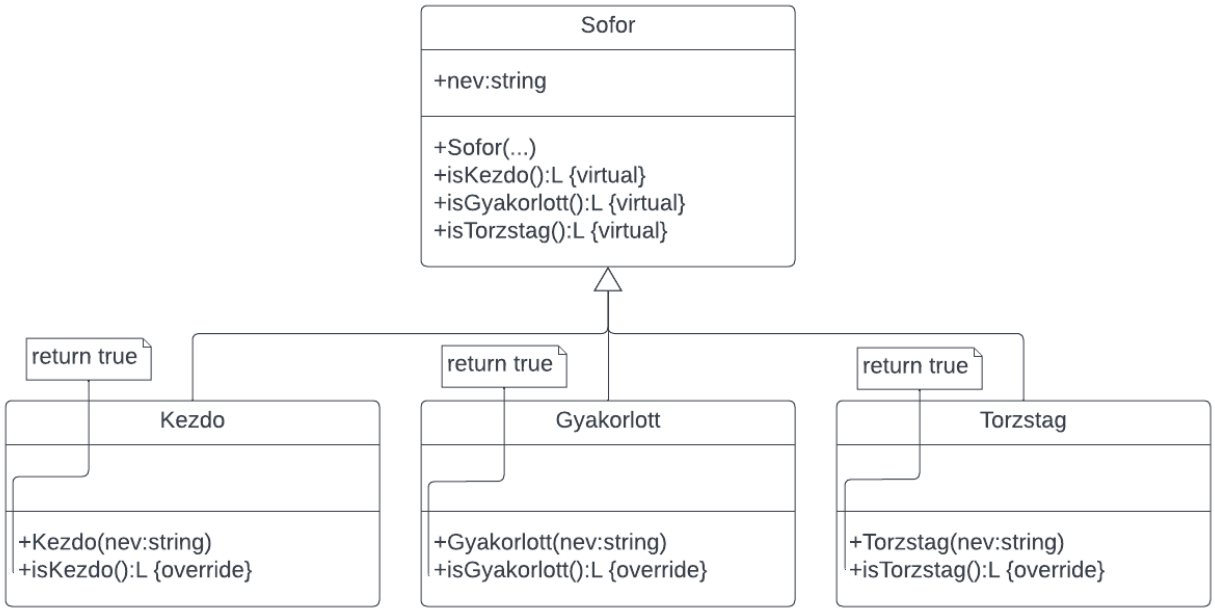
A megbízásokat a kamion illetve a sofőr kapja, azonban ez nem tárolja véglegesen. Amint sikeresen teljesítette a fuvart a megbízás a kiegészített érkezési idővel a cégben tárolódik el. Ezen folyamat kivitelezésére szolgál a „Telephely” és a „Ceg” megbizasKesz(m:Megbizas) metódus.

Az objektumok és a kommunikációt megvalósító metódusok kidolgozása az alábbi diagrammon látható:



Ezen objektumok közötti kapcsolat a következő féle képpen néz ki:

A sofőrök között lehet „Kezdo”, „Gyakorlott” és „Torzstag”. Ezek a „Sofor” szülőosztályból való leszármaztatással, az alábbi módon jelennek meg:

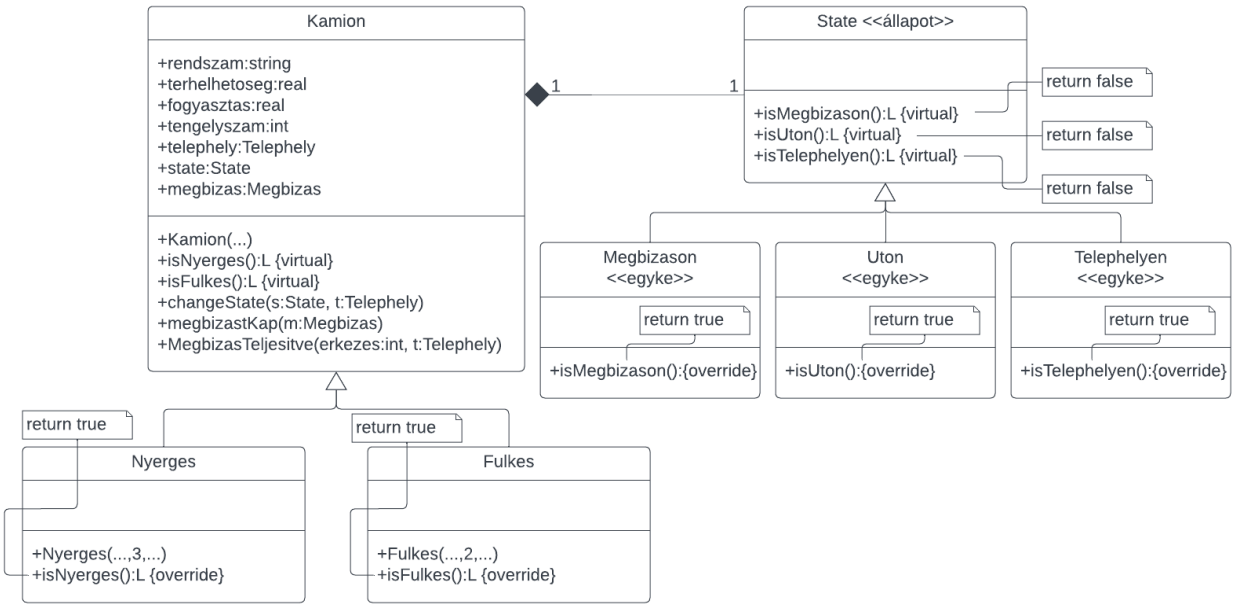


A kamionok lehetnek „Nyerges” illetve „Fulkes” típusúak. Ezek szintén öröklődés formájában valósulnak meg.

Valamint a kaionok allapotára a <<state>> (<<állapot>>) tervezési minta van alkalmazva, ez által a kamionok állapota jól követhető, lekérése egyszerű, nem utolsó sorban utólag könnyedén bővíthető pl. egy „tankol” vagy „szervízben van” állapottal, ha az annyira speciális hogy külön igényli ezt.

A state leszármazottjaira pedig a <<singleton>>. (<<egyke>>) tervezési minta lép életbe, mivel két azonos állapot között nincs különbség, így fölösleges lenne minden példányosításkor új egyedet létrehozni.

A „Nyerges” és a „Fulkes” alosztályokban látható hogy a konstruktor beégetve egy 3 illetve 2 értéket kap. A példányosítás során ezt az adatot küldi tovább a szülő konstruktorának, a többi adattag ugyanúgy tetszés szerint kitölthető.



A 4 (a,b,c,d) feladat megvalósítása.

a) Melyik telephelyhez tartozik éppen most a legkevesebb kamion?

„Ceg”-ben a „minKamion():Telephely”

Specifikáció: *(minker)*

A = (telephelyek:Telephely*, minT:Telephely)

Ef = (telephelyek=telephelyek', |telephelyek| > 0)

Uf = ((minT, minKamion) = $MIN_{e \in telephelyek'}$ |e.kamionok|)

Pszeudokód:

```
Telephely minT = telephelyek[0]
minKamion = |telephelyek[0].kamionok|
for i=0 to |telephelyek|
    if minKamion > |telephelyek[i].kamionok| then
            minKamion = |telephelyek[i].kamionok|
            minT = telephelyek[i]
    endif
endfor
return minT
```

b) Mekkora terhet képesek egyszerre elfuvarozni egy adott telephely kamionjai?

„Telephely”-ben a „maxTeher():real”

Specifikáció: *(összegzes)*

A = (kamionok:Kamion*,teherSum:real)

Ef = (kamionok = kamionok')

Uf = ((teherSum) = $\sum_{e \in kamionok'} e.terhelhetoseg$)

Pszeudokód:

```
teherSum = 0
for i=0 to |kamionok|
    teherSum = teherSum + kamionok[i].terhelhetoseg
endfor
return teherSum
```

c) Volt-e olyan kamion, amely túl volt terhelve valamelyik fuvarban?

„Ceg"-ben a „voltTulterhelt():L"

Specifikáció: *(linker)*

A = (megbizasok:Megbizasok*, volt:L)

Ef = (megbizasok= megbizasok')

Uf = ((volt)=SEARCH $_{e \in megbizasok'}$ e.kamion.terhelhetoseg < e.fuvSuly)

Pszeudokód:

```
for i=0 to |megbizasok|
    if megbizasok[i].kamion.terhelhetoseg < megbizasok[i].fuvSuly then
            return true
    endif
endfor
return false
```

d) Mekkora a nyeresége a cégnek? (fuvar nyeresége = szállítási díj – üzemanyag költség – bér)?

„Ceg”-ben a „nyereseg():real”

Specifikáció: *(összegzes)*

A = (megbizasok:Megbizasok*, nyereseg:real)

Ef = (megbizasok= megbizasok')

Uf = ((nyereseg) = $\sum_{e \in megbizasok'} e.nyereseg()$)

Pszeudokód:

```
nyereseg = 0
for i=0 to |megbizasok|
    nyereseg = nyereseg megbizasok[i].nyereseg()
endfor
return nyereseg
```

Tesztelési terv

1) Objektumok kapcsolatának tesztelése
    a. „Ceg” – „Telephely” (az addTelephely() jól működik)
    b. „Ceg” – „Sofor” (probléma nélkül addolhatók sofőrök a listához)
    c. „Telephely” – „Kamion” (az addKamion(), removeKamion(): jól működnek)
    d. „Kamion” – „Telephely” – „Ceg” (a megbizasKesz(): jól működik)
    e. „Kamion” – „Megbizas” (megbizastKap(), megbizasTeljesitve(): jól működnek)

2) A 4 feladatot megvalósító függvények tesztelése (ezzel az objektum kapcsolatok ellenőrzése)
    a. minKamion() tesztelése
        i. 1 telephely esetén
        ii. több telephely esetén
    b. maxTeher() tesztelése
        i. 0 kamion esetén
        ii. több kamion esetén
    c. voltTulterhelt() tesztelése
        i. 0 teljesített megbízás esetén
        ii. több teljesített megbízás esetén
    d. nyereseg() tesztelése

i. 0 teljesített megbízás esetén

ii. több teljesített megbízás esetén