Házi Feladat/Kész
Csontos Levente Péter
Előző

1. Feladat

Mobil szolgáltató

Egy mobilszolgáltatónál egy egyedi nyilvántartó programmal szeretnék kezelni az ügyfeleket. Az ügyfeleknek van neve és címe, valamint telefonszáma, ami egyben az egyedi azonosítjuk is. A szolgáltató jelenleg három csomagot biztosít ügyfeleinek: Alap, MobiNet és SMSMax, de később több csomag is lehet. Minden csomaghoz más percdíj és SMS díj tartozik, valamint a számítás módszere is eltérő lehet. A MobiNet csomag esetén pl. az is megadható, hogy hány SMS-t küldhet az ügyfél ingyen. A program egy fájlból olvassa be az ügyfelek adatait és választott díjcsomagot. Egy másik fájlból pedig az adott hónapban küldött SMS darabszámot és a lebeszélt perceket. A program írja ki, hogy az egyes ügyfelek mennyit fizetnek a forgalom alapján.

A megoldáshoz ne használjon STL tárolót

2. Pontosított feladat specifikáció

Egyértelműsítés

-A feladat nem specifikálja hogy az adott csomagok mennyibe kerülnek, illetve ez a végső összegbe bele tartozik-e, így a program úgy veszi hogy a csomag ára elhanyagolható (0). Az egyszerűség kedvéért a valuta neve a „pénz”. A feladat 1 hónap adatait veszi figyelembe, ezért az egyszerűség kedvéért az ügyfelek/ vevők csak 1 havi adatát kezeli.

Ügyfél adatai

-A feladat nem specifikálja, hogy mennyi ügyfél lehet, ezért annyi ügyféllel fog dolgozni a program, amennyi szükséges, fájlból való beolvasás után manuálisan is lehetőség új ügyfél felvételére, további ügyfélt csak a program újra nyitásával lehet hozzáadni, az ugyfel.txt fájl megfelelő szerkesztésével. A program az ügyfeleket az ugyfel.txt fájlból olvassa be úgy, hogy 1 sor 1 ügyfél adatai, a különböző adatok 1 db space-el legyenek elválasztva úgy: **Név Telefonszám Város_név Utca_név Házszám Csomag** . Az utca névnél elég az utca nevét megadni, típusát (út, utca stb.) nem kell, a város nevénél irányító számot nem kell megadni. Telefonszámnál nincs kezdeti speciális karakter (+) csak számokból áll, pl.: 204567891, ennek 9 karakterből kell állnia, az első 2 szám körzet számot szimbolizálja, az egyszerűség kedvéért nem 11 számjegyű, mert feltételezzük, hogy a telefonszámok Kitaláltországhoz tartoznak mind, így az országkódja (első kettő szám, ha 11 karakter lenne a telefonszám) mindegyiknek a „14”. Hiányos ügyfél adatok esetén a program azonnal leáll és const char-t dob: „hiba”. A program működésének egyik feltétele ennek a fájlnak a hiba nélküli megléte.

Elérhető mobil csomag adatai

-A program a különböző csomagok és tulajdonságaikat a csomag.txt fájlból olvassa, melyben 1 sorban vannak a csomag tulajdonságai space-el elválasztva úgy, hogy a csomag neve csak 1 szó lehet: **CsomagTípus Csomagnév Percdíj SMSdíj Ingyenespercek IngyenesSMSdb** . Végtelen ingyenes perc vagy SMS esetén az ingyenes percek vagy db sms-ek száma 0, de hozzátartozó díjazása is nulla, pl.: a MaxiSMS végtelen sms küldést tesz lehetővé melynek sms díja nulla, de perc díja 30 pénz, ekkor sms díja nulla, míg az ingyen elküldhető sms-ek db száma is nulla.

Havi adatok egyeztetése

-A program az adott ügyfelek havi sms és perc használatát a ho.txt fájlból olvassa be, melynek adatai space-kel legyen elválasztva úgy: **Telefonszám Percdb SMSdb** . Az egyszerűség kedvéért egy telefonszámhoz csak egy csomag tartozhat, ha adott telefonszám

kétszer is szerepel a fájlban, akkor az utóbbi felülírja az előző adatokat. Ha az ügyfél adata fel lett véve, de nincs adat a havi felhasználásáról akkor az egyszerűség kedvéért úgy van véve, hogy abban a hónapban 0 db sms-t, percet és mobil netet használt .

Kimenet egyeztetése

-A program a standard kimenetre kiírja hogy az egyes ügyfeleknek mennyit kell fizetniük, úgy hogy 1 sorba 1 ügyfél kerül, telefonszáma és végösszege, ezek space-vel elválasztva ilyen módon: **Telefonszám Végösszeg** . Pl.: egy esetben a kimenet: 204567891 200 pénz.

-a standard bemenetről tud manuálisan új adatokat is megadni, akár új ügyfél adatait, meglévő ügyfél adatainak változtatását, új csomag hozzáadását/változtatását, új havi adat hozzáadása/változtatása.

3.Terv

A feladat 7 objektum megtervezését igényli.

3.1.Objektum terv

Szolgáltató osztály

A feladatot 7 osztállyal oldom meg úgy, hogy a fő osztály a Szolgáltató, ő tárolja a különböző vevőket is egy dinamikus tömbben, illetve a tömb darab számát is. Azért tömb, mert így dinamikusan tudok annyi evőt kezelni, amennyit beolvasok.

Vevő, Csomagok osztály

-A vevő osztály mobil csomag tagjához van hozzá rendelve a csomagok osztály heterogén kollekciós tömbben lévő adott elem.

Csomag, Csomagok osztály

-A csomagok osztály heterogén kollekciójának tömbjét a csomag struktúra határozza meg. Azért tömb, mert így akármennyi csomagot fogok tudni létrehozni, ezzel szabadon tudok a különböző vásárlói igényekre bővíthető megoldást adni.

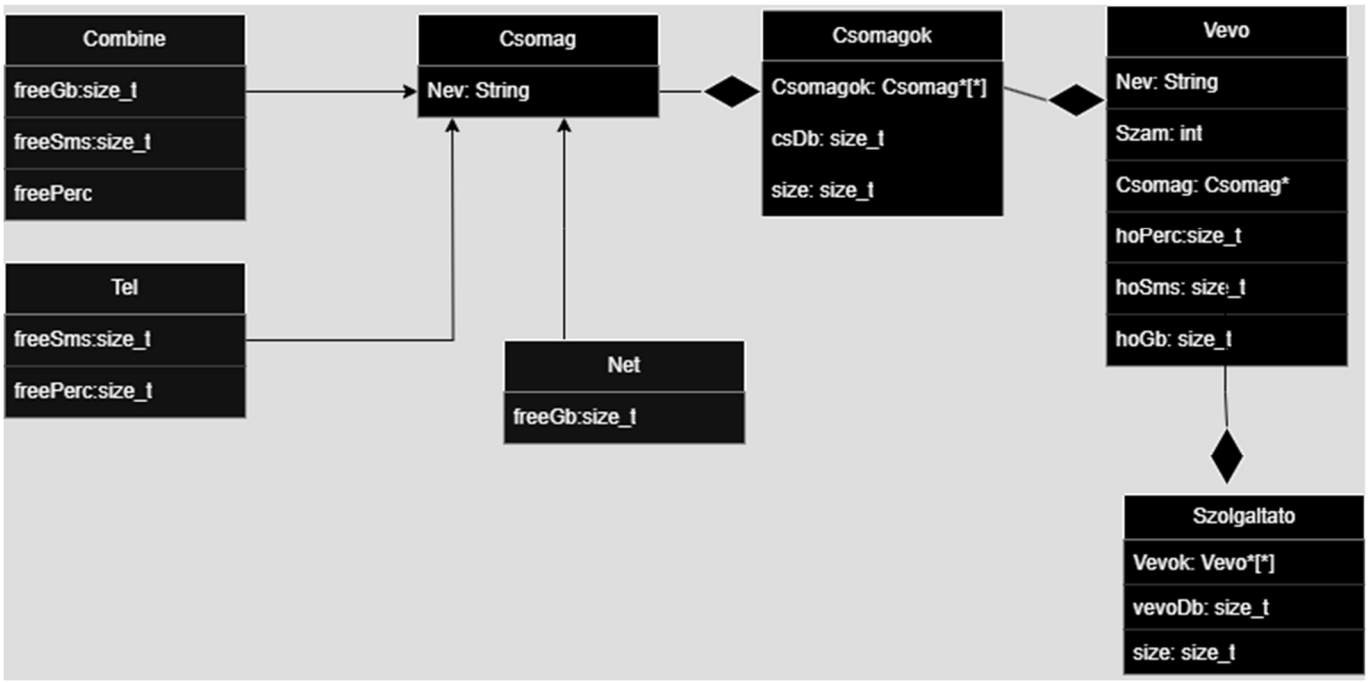
Vevő osztály

- A vevő osztály a költségei kiszámolásáért felelős.

Egyéb osztályok

-Az osztály egy hibakezelő osztály, mely a program futtatásakor ellenőrzi a bemeneti fájlokat, ha kell és hibákat dob, aszerint, hogy hol történt az. A program használja a memtrace-t, illetve az 5 laborban előállított string.h osztályt. A Csomag osztálynak van 3 leszármazott osztálya, amely a különböző csomagok szükségességét elégíti ki.

3.2.Feladat UML egyszerűsített diagrammja



Javított

1.Feladat

Mobil szolgáltató

Egy mobilszolgáltatónál egy egyedi nyilvántartó programmal szeretnék kezelni az ügyfeleket. Az ügyfeleknek van neve és címe, valamint telefonszáma, ami egyben az egyedi azonosítjuk is. A szolgáltató jelenleg három csomagot biztosít ügyfeleinek: Alap, MobiNet és SMSMax, de később több csomag is lehet. Minden csomaghoz más percdíj és SMS díj tartozik, valamint a számítás módszere is eltérő lehet. A MobiNet csomag esetén pl. az is megadható, hogy hány SMS-t küldhet az ügyfél ingyen. A program egy fájlból olvassa be az ügyfelek adatait és választott díjcsomagot. Egy másik fájlból pedig az adott hónapban küldött SMS darabszámot és a lebeszélt perceket. A program írja ki, hogy az egyes ügyfelek mennyit fizetnek a forgalom alapján.

A megoldáshoz ne használjon STL tárolót

2.Pontosított feladat specifikáció

Egyértelműsítés

-A feladat nem specifikálja hogy az adott csomagok mennyibe kerülnek, illetve ez a végső összegbe bele tartozik-e, így a program úgy veszi hogy a csomag ára elhanyagolható (0). Az egyszerűség kedvéért a valuta neve a „pénz”. A feladat 1 hónap adatait veszi figyelembe, ezért az egyszerűség kedvéért az ügyfelek/ vevők csak 1 havi adatát kezeli.

Ügyfél adatai

-A feladat nem specifikálja, hogy mennyi ügyfél lehet, ezért annyi ügyféllel fog dolgozni a program, amennyi szükséges. Fájlból való beolvasás után manuálisan is lehetőség van új ügyfél felvételére, további ügyfélt hozzáadni, illetve törölni bármikor lehet a program használata közben, a vevo.txt fájl megfelelő szerkesztésével fájlból is lehet hozzáadni, törölni csak manuálisan lehet, illetve a program bezárásával az összes eltárolt ügyfelet. A program az ügyfeleket az vevo.txt fájlból olvassa be úgy, hogy 1 sor 1 ügyfél adatai, a különböző adatok 1 db space-el legyenek elválasztva úgy: **Vezetéknév_Keresztnév Telefonszám Csomag** . Telefonszámnál nincs kezdeti speciális karakter (+) csak számokból áll, pl.: 204567891, ennek 9 karakterből kell állnia, az első 2 szám körzet számot szimbolizálja, az egyszerűség kedvéért nem 11 számjegyű, mert feltételezzük, hogy a telefonszámok Kitaláltországhoz tartoznak mind, így az országkódja (első kettő szám, ha 11 karakter lenne a telefonszám) mindegyiknek a „14”, feltételezzük továbbá hogy minden telefonszám első számjegye minimum az 1, az az a 1000000000 lesz az első valid telefonszám, míg a 999999999 lesz az utolsó jó. Hiányos ügyfél adatok esetén a program const char-t dob: „hiba”. A program működésének nem feltétele ennek a fájlnak a hiba nélküli megléte.

Elérhető mobil csomag adatai

-A program a különböző csomagok és tulajdonságaikat a csomag.txt fájlból olvassa, melyben 1 sorban vannak a csomag tulajdonságai space-el elválasztva úgy, hogy a csomag neve csak 1 szó lehet: **CsomagTípus Csomagnév IngyenesSmsDb IngyenesPercDb IngyenesGbDb SMSdíj Percdíj Gbdíj**. Végtelen ingyenes perc vagy SMS esetén az ingyenes percek vagy db sms-ek száma 0, de hozzátartozó díjazása is nulla, pl.: a MaxiSMS végtelen sms küldést tesz lehetővé melynek sms díja nulla, de perc díja 30 pénz, ekkor sms díja nulla, míg az ingyen elküldhető sms-ek db száma is nulla. A csomagok adatait nem csak fájlból tudja beolvasni, így a fálj hibátlan megléte nem feltétele a program működésének.

Havi adatok egyeztetése

-A program az adott ügyfelek havi sms és perc használatát a ho.txt fájlból olvassa be, melynek adatai space-kel legyen elválasztva úgy: **Telefonszám Percdb Smsdb Gbdb**. Az egyszerűség kedvéért egy telefonszámhoz csak egy csomag tartozhat, ha adott telefonszám kétszer is szerepel a fájlban, akkor az utóbbi felülírja az előző adatokat. Ha az ügyfél adata fel lett véve, de nincs adat a havi felhasználásáról akkor az egyszerűség kedvéért úgy van véve,

hogy abban a hónapban 0 db sms-t, percet és mobil netet használt .

Kimenet egyeztetése

-A program a standard kimenetre kiírja hogy az egyes ügyfeleknek mennyit kell fizetniük, úgy hogy 1 sorba 1 ügyfél kerül, telefonszáma, pár adata és végösszege, ezek space-vel elválasztva ilyen módon: **Név Csomagnév Telefonszám Végösszeg** . Pl.: egy esetben a kimenet: nev:Gyula csomag:MaxPerc telefon szam:204567891 befizetendo:200.

-a standard bemenetről tud manuálisan új adatokat is megadni, akár új ügyfél adatait, meglévő ügyfél adatainak változtatását, új csomag hozzáadását/változtatását, új havi adat hozzáadása/változtatása, csomag és vevő eltávolítása.

-a ho.txt, vevo.txt és csomag.txt fileokból tud adatokat beolvasni, ha azok helyes formátumban szerepelnek a fileokban.

3.Terv

A feladat 7 objektum megtervezését igényli.

3.1.Objektum terv

Szolgáltató osztály

A feladatot 8 osztállyal oldom meg úgy, hogy a fő osztály a Szolgáltató, ő tárolja a különböző vevőket is egy dinamikus tömbben, illetve a tömb darab számát is. Azért tömb, mert így dinamikusan tudok annyi evőt kezelni, amennyit beolvasok. Ez az osztály felelős a havi költség kiszámításáért is.

Vevő, Csomagok osztály

-A vevő osztály csomag adattagjában a hozzá tartozó csomag neve van elmentve. A csomagok osztály egy heterogén kollekció tárolója, amely eltárolja ezen csomagokat, felügyeli adott névből csak 1 legyen, mert ez az egyedi azonosító.

Csomag, Csomagok osztály

-A csomagok osztály heterogén kollekciójának tömbjét a csomag struktúra határozza meg. Azért tömb, mert így akármennyi csomagot fogok tudni létrehozni, ezzel szabadon tudok a különböző vásárlói igényekre bővíthető megoldást adni.

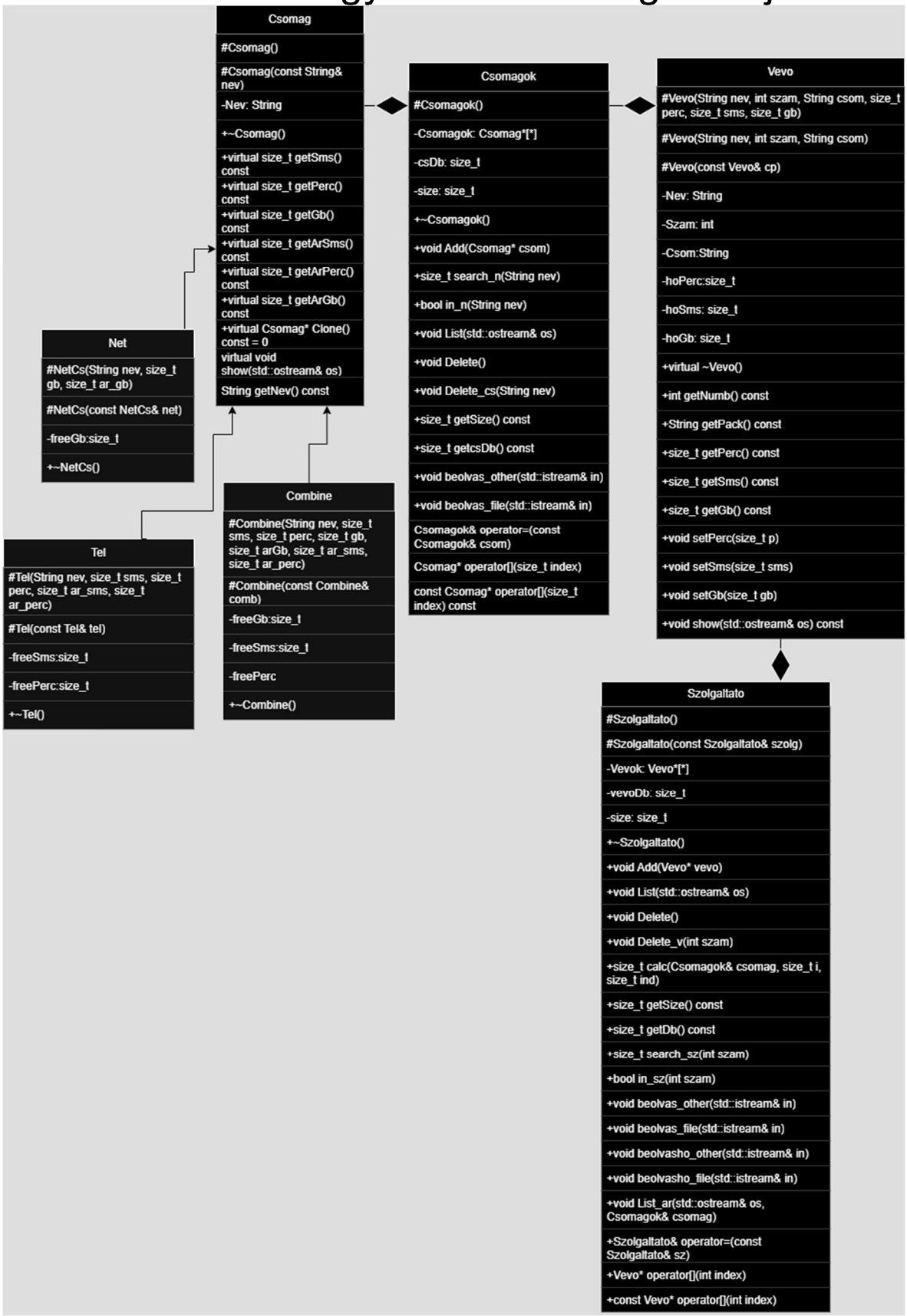
Vevő osztály

- A vevő osztály a vevő adatait tárolja, illetve hozzáférést enged a benne tárolt havi adatokhoz, hogy azt megváltoztatni lehessen.

Egyéb osztályok

-A program használja a memtrace-t, illetve az 5 laborban előállított string.h osztályt. A Csomag osztálynak van 3 leszármazott osztálya, amely a különböző csomagok szükségességét elégíti ki.

3.2.Feladat UML egyszerűsített diagrammja



Dokumentáció

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 1

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Chapter 4

Class Documentation

4.1 Combine Class Reference

Public Member Functions

- **Combine** (String nev, size_t sms, size_t perc, size_t gb, size_t arGb, size_t ar_sms, size_t ar_perc)

  alap konstruktor
- **Combine** (const Combine &comb)

  másoló konstruktor
- size_t getSms () const

  visszaadja a csomaghoz tartozó ingyenes sms-ek darabszámát
- size_t getPerc () const

  visszaadja a csomaghoz tartozó ingyenes perc-ek darabszámát
- size_t getGb () const

  visszaadja a csomaghoz tartozó ingyenes gb-ok darabszámát
- Csomag * Clone () const override

  clone függvény a deep copyhoz
- void show (std::ostream &os) const override

  kiíró függvény, mely kiírja a csomaghoz tartozó adatokat a megadott outputra
- size_t getArGb () const

  visszaadja a csomaghoz tartozó gb-ok árát
- size_t getArSms () const

  visszaadja a csomaghoz tartozó sms-ek árát
- size_t getArPerc () const

  visszaadja a csomaghoz tartozó percek árát

Public Member Functions inherited from Csomag

- **Csomag** (const String &nev)
- String **getNev** () const

4.1.1 Member Function Documentation

4.1.1.1 Clone()

```
Csomag * Combine::Clone () const  [override], [virtual]
```

clone függvény a deep copyhoz

Implements Csomag.

4.1.1.2 getArGb()

```
size_t Combine::getArGb () const  [virtual]
```

visszaadja a csomaghoz tartozó gb-ok árát

Reimplemented from Csomag.

4.1.1.3 getArPerc()

```
size_t Combine::getArPerc () const  [virtual]
```

visszaadja a csomaghoz tartozó percek árát

Reimplemented from Csomag.

4.1.1.4 getArSms()

```
size_t Combine::getArSms () const  [virtual]
```

visszaadja a csomaghoz tartozó sms-ek árát

Reimplemented from Csomag.

4.1.1.5 getGb()

```
size_t Combine::getGb () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes gb-ok darabszámát

Reimplemented from Csomag.

4.1.1.6 getPerc()

```
size_t Combine::getPerc () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes perc-ek darabszámát

Reimplemented from Csomag.

4.1.1.7 getSms()

```
size_t Combine::getSms () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes sms-ek darabszámát

Reimplemented from Csomag.

4.1.1.8 show()

```
void Combine::show (
            std::ostream & os) const  [override], [virtual]
```

kiíró függvény, mely kiírja a csomaghoz tartozó adatokat a megadott outputra

Implements Csomag.

The documentation for this class was generated from the following files:

- Combine.hpp
- Combine.cpp

4.2 Csomag Class Reference

Public Member Functions

- **Csomag** (const String &nev)
- virtual size_t getSms () const
- virtual size_t getPerc () const
- virtual size_t getGb () const
- virtual size_t getArGb () const
- virtual size_t getArSms () const
- virtual size_t getArPerc () const
- virtual Csomag * Clone () const =0
- virtual void show (std::ostream &os) const =0
- String **getNev** () const

4.2.1 Member Function Documentation

4.2.1.1 Clone()

```
virtual Csomag * Csomag::Clone () const  [pure virtual]
```

Implemented in Combine, NetCs, and Tel.

4.2.1.2 getArGb()

```
virtual size_t Csomag::getArGb () const  [inline], [virtual]
```

Reimplemented in Combine, NetCs, and Tel.

4.2.1.3 getArPerc()

```
virtual size_t Csomag::getArPerc () const  [inline], [virtual]
```

Reimplemented in Combine, NetCs, and Tel.

4.2.1.4 getArSms()

```
virtual size_t Csomag::getArSms () const  [inline], [virtual]
```

Reimplemented in Combine, NetCs, and Tel.

4.2.1.5 getGb()

```
virtual size_t Csomag::getGb () const  [inline], [virtual]
```

Reimplemented in Combine, NetCs, and Tel.

4.2.1.6 getPerc()

```
virtual size_t Csomag::getPerc () const  [inline], [virtual]
```

Reimplemented in Combine, NetCs, and Tel.

4.2.1.7 getSms()

```
virtual size_t Csomag::getSms () const  [inline], [virtual]
```

Reimplemented in Combine, NetCs, and Tel.

4.2.1.8 show()

```
virtual void Csomag::show (
            std::ostream & os) const  [pure virtual]
```

Implemented in Combine, NetCs, and Tel.

The documentation for this class was generated from the following file:

- Csomag.hpp

4.3 Csomagok Class Reference

Public Member Functions

- **Csomagok** ()

  Csomagok konstruktor, alapból 10 elemre elegendo˝ helyet állít be.
- ~**Csomagok** ()

  desktruktor, használja a Delete függvényt
- void **Add** (Csomag ∗Csom)

  Add fv. egy csomagot add hozzá a tárolóhoz, ha nincs elég hely akkor növeli a tároló kapacitását.
- size_t **search_n** (String csNev)

  visszaadja az adott nevu˝ objektum indexét a tárolóban ha benne van, egyébként a size_t -1 megfelelo˝jét adja vissza
- bool **in_n** (String csNev)

  visszadja hogy benne van-e az adott nevu˝ objektum a tárolt objektumok között
- void **List** (std::ostream &os)

  kiírja a megadott outputra az eltárolt objektumok adatait
- void **Delete** ()

  törli a tárolóban lévo˝ összes elemet
- void **Delete_cs** (String csNev)

  adott nevu˝ objektum törlése a tárolt objektumok közül ha benne van a tárolóban
- size_t **getSize** () const

  visszaadja a tároló méretét
- size_t **getcsDb** () const

  visszaadja jelenleg legnagyobb indexu˝ elemnél 1-el nagyobb indexet
- void **beolvas_file** (std::istream &in)

  adatok beolvasása fo˝ként fileból, de a megadott inputról megadott formátum szerint
- void **beolvas_other** (std::istream &in)

  adott inputról olvassa be az adatokat a megadott formátum szerint
- Csomagok & **operator=** (const Csomagok &csom)

  értékadó operátor
- Csomag ∗ **operator[ ]** (size_t index)

  indexelo˝ operátor
- const Csomag ∗ **operator[ ]** (size_t index) const

  const verziója az indexelés operátorának

The documentation for this class was generated from the following files:

- Csomagok.hpp
- Csomagok.cpp

4.4 NetCs Class Reference

Public Member Functions

- **NetCs** (String nev, size_t gb, size_t ar_gb)

  alap konstruktor
- **NetCs** (const NetCs &net)

  másoló konstruktor
- size_t getGb () const

  visszaadja a csomaghoz tartozó ingyenes gb-ok darabszámát
- size_t getSms () const

  visszaadja a csomaghoz tartozó ingyenes sms-ek darabszámát
- size_t getPerc () const

  visszaadja a csomaghoz tartozó ingyenes percek darabszámát
- Csomag * Clone () const override

  clone függvény a másoláskor deep copy-hoz
- void show (std::ostream &os) const override

  kiírja a csomag adatait a megadott outputra
- size_t getArSms () const

  visszaadja a csomaghoz tartozó sms-ek árát
- size_t getArPerc () const

  visszaadja a csomaghoz tartozó percek árát
- size_t getArGb () const

  visszaadja a csomaghoz tartozó gb-ok árát

Public Member Functions inherited from Csomag

- **Csomag** (const String &nev)
- String **getNev** () const

4.4.1 Member Function Documentation

4.4.1.1 Clone()

```
Csomag * NetCs::Clone () const  [override], [virtual]
```

clone függvény a másoláskor deep copy-hoz

Implements Csomag.

4.4.1.2 getArGb()

```
size_t NetCs::getArGb () const  [virtual]
```

visszaadja a csomaghoz tartozó gb-ok árát

Reimplemented from Csomag.

4.4.1.3 getArPerc()

```
size_t NetCs::getArPerc () const  [virtual]
```

visszaadja a csomaghoz tartozó percek árát

Reimplemented from Csomag.

4.4.1.4 getArSms()

```
size_t NetCs::getArSms () const  [virtual]
```

visszaadja a csomaghoz tartozó sms-ek árát

Reimplemented from Csomag.

4.4.1.5 getGb()

```
size_t NetCs::getGb () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes gb-ok darabszámát

Reimplemented from Csomag.

4.4.1.6 getPerc()

```
size_t NetCs::getPerc () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes percek darabszámát

Reimplemented from Csomag.

4.4.1.7 getSms()

```
size_t NetCs::getSms () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes sms-ek darabszámát

Reimplemented from Csomag.

4.4.1.8 show()

```
void NetCs::show (
            std::ostream & os) const  [override], [virtual]
```

kiírja a csomag adatait a megadott outputra

Implements Csomag.

The documentation for this class was generated from the following files:

- NetCs.hpp
- NetCs.cpp

4.5 String Class Reference

```
#include <string.h>
```

Public Member Functions

- size_t size () const
- const char * c_str () const
- String (char ch)

  Konstruktor: egy char karakterbo˝l (createStrFromChar)
- String (const char *p="")
- String (const String &s1)
- virtual ~**String** ()

  Destruktor.
- void printDbg (const char *txt="") const
- bool **operator==** (const String &rhs_s)

  összehasonlító operátor
- String & operator= (const String &rhs_s)
- String operator+ (const String &rhs_s) const
- String operator+ (char rhs_c) const
- char & operator[ ] (unsigned int idx)
- const char & operator[ ] (unsigned int idx) const

4.5.1 Detailed Description

String osztály. A pData-ban vannak a karakterek (a lezáró nullával együtt), len a hossz.A hosszba nem számít bele a lezáró nulla.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 String() [1/3]

```
String::String (
            char ch)
```

Konstruktor: egy char karakterbo˝l (createStrFromChar)

Konstruktor egy char karakterbo˝l

Parameters

ch	- karakter

4.5.2.2 String() [2/3]

```
String::String (
            const char * p = "")
```

Konstruktor egy nullával lezárt char sorozatból Ez a deafault is!

Parameters

p	- pointer egy C sztringre

4.5.2.3 String() [3/3]

```
String::String (
            const String & s1)
```

Másoló konstruktor

Parameters

s1	- String, amibo˝l létrehozzuk az új String-et

4.5.3 Member Function Documentation

4.5.3.1 c_str()

```
const char * String::c_str () const  [inline]
```

Default konstruktor String() :pData(0), len(0) {} helyett ""-val inicializáljuk a const char∗-osban C-sztringet ad vissza

Returns

pinter egy '\0'-val lezárt (C) sztringre

4.5.3.2 operator+() [1/2]

```
String String::operator+ (
            char rhs_c) const  [inline]
```

Sztrinhez karaktert összefu˝z

Parameters

rhs_c ⟷	- jobboldali karakter

Returns

új String, ami tartalmazza a sztringet és a karaktert egymás után

4.5.3.3 operator+() [2/2]

```
String String::operator+ (
            const String & rhs_s) const
```

Két Stringet összefu˝z

Parameters

rhs_s ⟷	- jobboldali String

Returns

új String, ami tartalmazza a két stringet egmás után

4.5.3.4 operator=()

```
String & String::operator= (
            const String & rhs_s)
```

Értékadó operátor.

Parameters

rhs_s ⟷	- jobboldali String

Returns

baoldali (módosított) string (referenciája)

4.5.3.5 operator[]() [1/2]

```
char & String::operator[] (
            unsigned int idx)
```

A string egy megadott indexu˝ elemének REFERENCIÁJÁVAL tér vissza.

Parameters

idx	- charakter indexe

Returns

karakter (referencia) Indexelési hiba esetén const char∗ kivételt dob.

4.5.3.6 operator[]() [2/2]

```
const char & String::operator[] (
            unsigned int idx) const
```

A string egy megadott indexu˝ elemének REFERENCIÁJÁVAL tér vissza.

Parameters

idx	- karakter indexe

Returns

karakter (referencia) Indexelési hiba esetén const char∗ kivételt dob (assert helyett).

4.5.3.7 printDbg()

```
void String::printDbg (
            const char * txt = "") const  [inline]
```

Kiírunk egy Stringet (debug célokra) Elo˝tte kiírunk egy tetszo˝leges szöveget.

Parameters

txt	- nullával lezárt szövegre mutató pointer

4.5.3.8 size()

```
size_t String::size () const  [inline]
```

Hossz lekérdezése.

Returns

Sztring hossza

The documentation for this class was generated from the following files:

- string.h
- string.cpp

4.6 Szolgaltato Class Reference

Public Member Functions

- **Szolgaltato** ()

  alap konstruktor, mely létrehozza a tárolót és felkészíti egy kezdeti 50 objektumra
- **Szolgaltato** (const Szolgaltato &szolg)

  másoló konstruktor
- ~**Szolgaltato** ()

  destruktor
- void **Add** (Vevo *vevo)

  tárolóhoz új objektum hozzáadása
- void **List** (std::ostream &os)

  kiírja a tárolóban eltárolt objektumok adatait
- void **Delete** ()

  tároló adatainak törlése
- void **Delete_v** (int szam)

  adott telefonszámmal rendelkezo˝ vevo˝ törlése a tárolóbol
- size_t **calc** (Csomagok &csomag, size_t i, size_t ind)

  segédfüggvény a havi költség kiszámításra
- size_t **getSize** () const

  visszaadja a tároló jelenlegi maximális méretét
- size_t **getDb** () const

  visszaadja a tárolóban eltárolt objektumok számát
- size_t **search_sz** (int szam)

  visszaadja adott telefonszámmal rendelkezo˝ vevo˝ tárolóban lévo˝ indexét ha van, egyébként a size_t -1 megfelelo˝jét
- bool **in_sz** (int szam)

  visszaadja adott telefonszámmal rendelkezo˝ vevo˝ benne van-e a tárolóban
- void **beolvas_file** (std::istream &in)

  fileból fo˝ként de adott inputról olvassa be a vevo˝k adatait
- void **beolvas_other** (std::istream &in)

  beolvassa az adott inputról a vevo˝ objektum adatait
- void **beolvasho_other** (std::istream &in)

  beolvassa egy adott vevo˝höz tartozó havi adatokat adott inputról
- void **beolvasho_file** (std::istream &in)

  fo˝ként fileból, de adott inputról olvassa be adott vevo˝k havi adatait
- void **List_ar** (std::ostream &os, Csomagok &csomag)

  kiírja a kiszamítható havi költséggel rendelkezo˝ vevo˝k havi költségét
- Szolgaltato & **operator=** (const Szolgaltato &szolg)

  értékadó operátor
- Vevo * **operator[ ]** (int index)

  tároló indexel való hozzáférése
- const Vevo * **operator[ ]** (int index) const

  tároló indexel való hozzáférése constans visszatérési értékkel

The documentation for this class was generated from the following files:

- Szolgaltato.hpp
- Szolgaltato.cpp

4.7 Tel Class Reference

Public Member Functions

- **Tel** (String nev, size_t sms, size_t perc, size_t ar_sms, size_t ar_perc)

  alap konstruktor
- **Tel** (const Tel &tel)

  másoló konstruktor
- size_t getSms () const

  visszaadja a csomaghoz tartozó ingyenes sms-ek darabszámát
- size_t getPerc () const

  visszaadja a csomaghoz tartozó ingyenes percek darabszámát
- size_t getGb () const

  visszaadja a csomaghoz tartozó ingyenes gb-ok darabszámát
- Csomag * Clone () const override

  clone függvény a másoláskor deep copy-hoz
- void show (std::ostream &os) const override

  kiíró függvény amely a megadott outputra írja ki a csomag adatait
- size_t getArGb () const

  visszaadja a csomaghoz tartozó gb-ok árát
- size_t getArSms () const

  visszaadja a csomaghoz tartozó sms-ek árát
- size_t getArPerc () const

  visszaadja a csomaghoz tartozó percek árát

Public Member Functions inherited from Csomag

- **Csomag** (const String &nev)
- String **getNev** () const

4.7.1 Member Function Documentation

4.7.1.1 Clone()

```
Csomag * Tel::Clone () const  [override], [virtual]
```

clone függvény a másoláskor deep copy-hoz

Implements Csomag.

4.7.1.2 getArGb()

```
size_t Tel::getArGb () const  [virtual]
```

visszaadja a csomaghoz tartozó gb-ok árát

Reimplemented from Csomag.

4.7.1.3 getArPerc()

```
size_t Tel::getArPerc () const  [virtual]
```

visszaadja a csomaghoz tartozó percek árát

Reimplemented from Csomag.

4.7.1.4 getArSms()

```
size_t Tel::getArSms () const  [virtual]
```

visszaadja a csomaghoz tartozó sms-ek árát

Reimplemented from Csomag.

4.7.1.5 getGb()

```
size_t Tel::getGb () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes gb-ok darabszámát

Reimplemented from Csomag.

4.7.1.6 getPerc()

```
size_t Tel::getPerc () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes percek darabszámát

Reimplemented from Csomag.

4.7.1.7 getSms()

```
size_t Tel::getSms () const  [virtual]
```

visszaadja a csomaghoz tartozó ingyenes sms-ek darabszámát

Reimplemented from Csomag.

4.7.1.8 show()

```
void Tel::show (
            std::ostream & os) const  [override], [virtual]
```

kiíró függvény amely a megadott outputra írja ki a csomag adatait

Implements Csomag.

The documentation for this class was generated from the following files:

- Tel.hpp
- Tel.cpp

4.8 Vevo Class Reference

Public Member Functions

- **Vevo** (String nev, int szam, String csom, size_t perc, size_t sms, size_t gb)
  konsturktor ha minden adat megvan
- **Vevo** (String nev, int szam, String csom)
  konstruktor hiányos adatokhoz, objektum létrehozása minimális adatokkal
- **Vevo** (const Vevo &cp)
  másoló konstruktor
- int **getNumb** () const
  visszaadja a telefonszámot
- String **getPack** () const
  visszaadja az objektumhoz tartozó csomag nevét
- size_t **getPerc** () const
  visszaadja a havi perc mennyiséget
- size_t **getSms** () const
  visszaadja a havi sms mennyiséget
- size_t **getGb** () const
  visszaadja a havi gb mennyiséget
- void **setPerc** (size_t p)
  beállítja a havi perc mennyiséget
- void **setSms** (size_t sms)
  beállítja a havi sms mennyiséget
- void **setGb** (size_t gb)
  beállítja a havi gb mennyiséget
- void **show** (std::ostream &os) const
  kiírja az objektum adataiból a nevet, csomag nevet és a telefonszámot

The documentation for this class was generated from the following files:

- Vevo.hpp
- Vevo.cpp

Chapter 5

File Documentation

5.1 Combine.hpp

```
00001 #ifndef COMBINE_HPP
00002 #define COMBINE_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006 #include "Csomag.hpp"
00007
00008 class Combine: public Csomag{
00009     size_t freeGb;
00010     size_t freeSms;
00011     size_t freePerc;
00012     size_t arGb;
00013     size_t arSms;
00014     size_t arPerc;
00015 public:
00016     Combine(String nev, size_t sms, size_t perc, size_t gb, size_t arGb, size_t ar_sms, size_t
    ar_perc);
00017     Combine(const Combine& comb);
00018     ~Combine(){}
00019     size_t getSms() const;
00020     size_t getPerc() const;
00021     size_t getGb() const;
00022     Csomag* Clone() const override;
00023     void show(std::ostream& os) const override;
00024     size_t getArGb() const;
00025     size_t getArSms() const;
00026     size_t getArPerc() const;
00027 };
00028
00029
00030 #endif
```

5.2 Csomag.hpp

```
00001 #ifndef CSOMAG_HPP
00002 #define CSOMAG_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006
00007 class Csomag{
00008     String Nev;
00009 public:
00010     Csomag(){}
00011     Csomag(const String& nev):Nev(nev){}
00012     virtual ~Csomag(){}
00013     virtual size_t getSms() const{ return 0;}
00014     virtual size_t getPerc() const{ return 0;}
00015     virtual size_t getGb() const{ return 0;}
00016     virtual size_t getArGb() const{ return 0;}
00017     virtual size_t getArSms() const{ return 0;}
00018     virtual size_t getArPerc() const{ return 0;}
00019     virtual Csomag* Clone() const = 0;
00020     virtual void show(std::ostream& os) const = 0;
```

```
00021     String getNev() const{
00022         return Nev;
00023     }
00024 };
00025
00026
00027 #endif
```

5.3 Csomagok.hpp

```
00001 #ifndef CSOMAGOK_HPP
00002 #define CSOMAGOK_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006 #include "Csomag.hpp"
00007
00008 class Csomagok{
00009     size_t csDb;
00010     size_t size;
00011     Csomag** csomagok;
00012 public:
00013     Csomagok();
00014     ~Csomagok();
00015     void Add(Csomag* Csom);
00016     size_t search_n(String csNev);
00017     bool in_n(String csNev);
00018     void List(std::ostream& os);
00019     void Delete();
00020     void Delete_cs(String csNev);
00021     size_t getSize() const;
00022     size_t getcsDb() const;
00023     void beolvas_file(std::istream& in);
00024     void beolvas_other(std::istream& in);
00025     Csomagok& operator=(const Csomagok& csom);
00026     Csomag* operator[](size_t index);
00027     const Csomag* operator[](size_t index)const;
00028 };
00029
00030
00031 #endif
```

5.4 NetCs.hpp

```
00001 #ifndef NETCS_HPP
00002 #define NETCS_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006 #include "Csomag.hpp"
00007
00008 class NetCs : public Csomag{
00009     size_t freeGb;
00010     size_t arGb;
00011 public:
00012     NetCs(String nev, size_t gb, size_t ar_gb);
00013     NetCs(const NetCs& net);
00014     ~NetCs(){}
00015     size_t getGb() const;
00016     size_t getSms() const;
00017     size_t getPerc() const;
00018     Csomag* Clone() const override;
00019     void show(std::ostream& os) const override;
00020     size_t getArSms() const;
00021     size_t getArPerc() const;
00022     size_t getArGb() const;
00023 };
00024
00025 #endif
```

5.5 string.h

```
00001 #ifndef STRING_H
00002 #define STRING_H
00009 #include <iostream>
```

```
00010
00016 class String {
00017     char *pData;
00018     size_t len;
00019 public:
00020
00021
00024     size_t size() const { return len; }
00025
00026
00030
00033     const char* c_str() const { return pData;}
00034
00037     String(char ch);
00038
00042     String(const char *p = "");
00043
00046     String(const String& s1);
00047
00049     virtual ~String() { delete[] pData; }
00050
00054     void printDbg(const char *txt = "") const {
00055         std::cout « txt « "[" « len « "], "
00056                   « (pData ? pData : "(NULL)") « std::endl;
00057     }
00058
00060      bool operator==(const String& rhs_s);
00061
00065     String& operator=(const String& rhs_s);
00066
00070     String operator+(const String& rhs_s) const ;
00071
00075     String operator+(char rhs_c) const { return *this + String(rhs_c);}
00076
00081     char& operator[](unsigned int idx);
00082
00087     const char& operator[](unsigned int idx) const;
00088 };
00089
00095 std::ostream& operator«(std::ostream& os, const String& s0);
00096
00101 std::istream& operator»(std::istream& is, String& s0);
00102
00107 inline String operator+(char ch, const String& str) { return String(ch) + str; }
00108
00109 #endif
```

5.6 Szolgaltato.hpp

```
00001 #ifndef SZOLGALTATO_HPP
00002 #define SZOLGALTATO_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006 #include "Vevo.hpp"
00007 #include "Csomagok.hpp"
00008
00009 class Szolgaltato{
00010     Vevo** vevok;
00011     size_t vevoDb;
00012     size_t size;
00013 public:
00014     Szolgaltato();
00015     Szolgaltato(const Szolgaltato& szolg);
00016     ~Szolgaltato();
00017     void Add(Vevo* vevo);
00018     void List(std::ostream& os);
00019     void Delete();
00020     void Delete_v(int szam);
00021     size_t calc(Csomagok& csomag, size_t i, size_t ind);
00022     size_t getSize() const;
00023     size_t getDb() const;
00024     size_t search_sz(int szam);
00025     bool in_sz(int szam);
00026     void beolvas_file(std::istream& in);
00027     void beolvas_other(std::istream& in);
00028     void beolvasho_other(std::istream& in);
00029     void beolvasho_file(std::istream& in);
00030   void List_ar(std::ostream& os, Csomagok& csomag);
00031   Szolgaltato& operator=(const Szolgaltato& szolg);
00032    Vevo* operator[](int index);
00033   const Vevo* operator[](int index)const;
00034 };
```

```
00035
00036 #endif
```

5.7 Tel.hpp

```
00001 #ifndef TEL_HPP
00002 #define TEL_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006 #include "Csomag.hpp"
00007
00008 class Tel: public Csomag{
00009     size_t freeSms;
00010     size_t freePerc;
00011     size_t arSms;
00012     size_t arPerc;
00013 public:
00014     Tel(String nev, size_t sms, size_t perc, size_t ar_sms, size_t ar_perc);
00015     Tel(const Tel& tel);
00016     ~Tel(){}
00017     size_t getSms() const;
00018     size_t getPerc() const;
00019     size_t getGb() const;
00020     Csomag* Clone() const override;
00021     void show(std::ostream& os) const override;
00022     size_t getArGb() const;
00023     size_t getArSms() const;
00024     size_t getArPerc() const;
00025 };
00026
00027
00028 #endif
```

5.8 test.hpp

```
00001 #pragma once
00002
00003 void test_run();
```

5.9 Vevo.hpp

```
00001 #ifndef VEVO_HPP
00002 #define VEVO_HPP
00003
00004 #include "memtrace.h"
00005 #include "string.h"
00006 #include "Csomag.hpp"
00007 #include "Csomagok.hpp"
00008
00009 class Vevo{
00010     String Nev;
00011     int Szam;
00012     String Csom;
00013     size_t hoPerc;
00014     size_t hoSms;
00015     size_t hoGb;
00016 public:
00017     Vevo(String nev, int szam, String csom, size_t perc, size_t sms, size_t gb);
00018     Vevo(String nev, int szam, String csom);
00019     Vevo(const Vevo& cp);
00020     virtual ~Vevo(){}
00021
00022     int getNumb() const;
00023     String getPack()const;
00024     size_t getPerc() const;
00025     size_t getSms() const;
00026     size_t getGb() const;
00027     void setPerc(size_t p);
00028     void setSms(size_t sms);
00029     void setGb(size_t gb);
00030
00031     void show(std::ostream& os)const;
00032 };
00033
00034 #endif
```