# NHF - Vonatjegy
# Programozás alapjai 2.
# DORAU5

Toldi Lőrinc László

2024.05.19.

# 1. fejezet

# Feladat

Tervezze meg egy vonatjegy eladó rendszer egyszerűsített objektummodelljét, majd valósítsa azt meg! A vonatjegy a feladatban mindig jegyet és helyjegyet jelent együtt. Így egy jegyen minimum a következőket kell feltüntetni:

- vonatszám, kocsiszám, hely
- indulási állomás, indulási idő
- érkezési állomás, érkezési idő

A rendszerrel minimum a következő műveleteket kívánjuk elvégezni:

- vonatok felvétele
- jegy kiadása

A rendszer később lehet bővebb funkcionalitású (pl. késések kezelése, vonat törlése, menetrend, stb.), ezért nagyon fontos, hogy jól határozza meg az objektumokat és azok felelősségét.
Valósítsa meg a jeggyel végezhető összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni az összes operátor átdefiniálásához!

# 2. fejezet

# Terv

A program célja

A feladat egy vonatjegy eladó rendszer megvalósítása, amely képes kezelni a vonatok rendszerbe történő felvételét és törlését, illetve a már mentett vonatokra szóló jegyek kiadását. A rendszerben regisztrált vonatok listájából menetrend lekérdezhető. A program képes a vonatokat és kiadott jegyeket fájl-ba kiírni és fájl-ból beolvasni. A program menüvezérelt, grafikus megjelenítést nem alkalmaz.

menü megjelenése

A menü a konzolban jelenik meg szövegesen. A menüpontok egymást követő sorokba vannak írva, minden sor a megfelelő sorszámmal kezdődik. A választható menüpontok a következőek; (1) jegykiadás, (2) vonat felvétel, (3) vonat törlés, (4) adatok mentése, (5) adatok betöltése, (6) menetrend lekérdezés, (7) kilépés.

menüpont választás

A menüpont választás az ahhoz tartozó szám beírásával majd enter-el történik. (pl. (1) - Jegykiadás esetében az '1' konzolban való bevitelével.) Hibás bevitelével esetén a program a felhasználót tájékoztatja és újbóli választásra kéri. A menüpont választás után a választott menüpont jelenik meg.

jegykiadás

A jegykiadás menüpontban egy jegy kiadására van lehetőség. A felhasználónak elsőnek az indulási és az érkezési állomás nevét kell megadnia, majd az indulási időpontot. A program arra a vonatra ad jegyet, mely az adott állomásra ahhoz leghamarabb érkezik. Amennyiben nem találtunk vonatot a kettő állomás között a felhasználót tájékoztatjuk és felajánljuk a folyamat újrakezdését. Ha találtunk vonatot az indulás időpontját kiírjuk. A felhasználónak a jegy típusát is kikell választania, amelynek ára a menetidőből ered (x Ft/perc alapon). A jegyek típusai a következők: első osztály - 48Ft/perc, másod osztály - 30Ft/perc, diák 12Ft/perc, nyugdíjas 12Ft/perc. Az adatok bekérése egyesével és egymást

követően történik. A készített objektumot a rendszerben tároljuk. A ‘kiadott’ jegy a konzolban jelenik meg.

### vonat felvétel

A vonat felvétel menüpontban új vonat regisztrálására van lehetőség. A felhasználótól a következő adatokat kérjük be; vonatszám [egész], kocsi szám [egész], kocsikban elérhető helyek száma [egész], állomások száma [egész, minimum 2]. A megadott állomások számának megfelelően annyiszor jelenik meg az állomás bekérése, melyek sorrendje egymást követő kell, hogy legyen. Az állomás bekért attribútumai: állomás neve [szöveg], érkezési idő [óra: egész, perc: egész]. A készített objektumot a rendszerben tároljuk. Hiba esetén a felhasználót tájékoztatjuk és újbóli bevitelre kérjük. Sikeres esetben visszakerülünk a főmenübe.

### vonat törlés

A vonat törlése menüpontban létező vonat törlése elérhető. A felhasználótól a vonat számát kérjük be, majd az ahhoz tartozó vonatot töröljük a rendszerből. Hiba esetén a felhasználót tájékoztatjuk és újbóli bevitelre kérjük. Sikeres esetben visszakerülünk a főmenübe.

### adatok mentése

Az adatok mentése menüpontban fájl-ba tudjuk kiírni a vonatokat és a hozzájuk tartozó állomásokat, kocsikat és jegyeket. A felhasználótól a fájl nevét kérjük be. Hiba esetén a felhasználót tájékoztatjuk és újbóli bevitelre kérjük. Sikeres esetben visszakerülünk a főmenübe.

### adatok betöltése

Az adatok betöltése menüpontban fájl-ból olvassuk be a vonatokat és a hozzájuk tartozó állomásokat, kocsikat és jegyeket. A beolvasott adatokat a rendszerben tároljuk. A felhasználótól a fájl nevét kérjük be. Hiba esetén a felhasználót tájékoztatjuk és újbóli bevitelre kérjük. Sikeres esetben visszakerülünk a főmenübe.

### menetrend lekérdezés

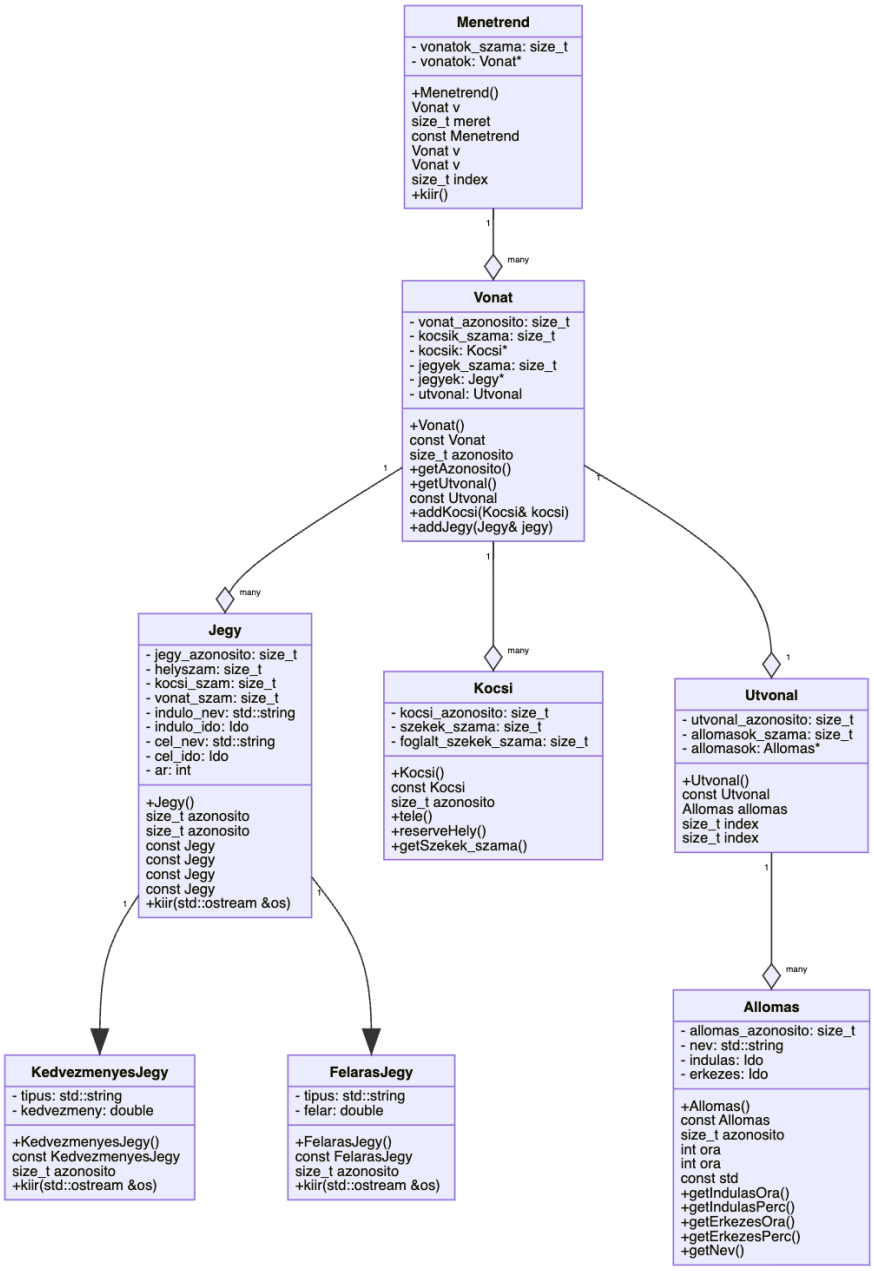
A menetrend lekérdezésnél a következő paramétereket kérjük be és azokat szűrőként alkalmazzuk a menetrendre; Indulási állomás ["szöveg"], érkezési állomás ["szöveg"]. Az adatok bekérése egyesével és egymást követően történik. Az üresen hagyott paramétereket nem alkalmazzuk. Az

# 2. fejezet

adatok bevitelét követően kiírjuk a megfelelő vonatok számát és listázzuk őket. kilépés A kilépés menüpont leállítja a programot.

# 3. fejezet

# Objektum terv



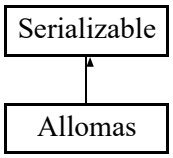
# 4. fejezet

# Osztályok dokumentációja

## 4.1. Allomas osztályreferencia

`#include <allomas.h>`

Az Allomas osztály származási diagramja:




### Publikus tagfüggvények

- Allomas ()

  *Az érkezési időpont tárolója.*
- Allomas (const Allomas &other)
- Allomas & operator= (const Allomas &other)

  *Értékadó operátor.*
- Allomas (size_t azonosito, const std::string &nev, int erkezes_ora, int erkezes_perc, int indulas_ora, int indulas_perc)
- void changeErkezes (int ora, int perc)
- void changeIndulas (int ora, int perc)
- void changeNev (const std::string &other)
- int getIndulasOra ()
- int getIndulasPerc ()
- int getErkezesOra ()
- int getErkezesPerc ()
- Ido & getIndulas ()
- Ido & getErkezes ()
- std::string & getNev ()
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~Allomas ()

  *Az osztály destruktora.*

**Privát attribútumok**

- size_t allomas_azonosito
- std::string nev

  *Az állomás azonosítója.*
- Ido indulas

  *Az állomás neve.*
- Ido erkezes

  *Az indulási időpont tárolója.*

### 4.1.1. Részletes leírás

Az Állomás osztály reprezentálja az állomásokat. Tartalmazza az állomás azonosítóját, az állomás nevét és a vonat érkezésének is indulásának időpontját.

### 4.1.2. Konstruktorok és destruktorok dokumentációja

#### 4.1.2.1. Allomas() [1/3]

```
Allomas::Allomas ( )
```

Az érkezési időpont tárolója.

Az osztály alapértelmezett konstruktora.

#### 4.1.2.2. Allomas() [2/3]

```
Allomas::Allomas (
            const Allomas & other )
```

Másoló konstruktor.

**Paraméterek**

| | |
|---|---|
| *other* | - Másolni kívánt Allomas objektum. |

#### 4.1.2.3. Allomas() [3/3]

```
Allomas::Allomas (
            size_t azonosito,
            const std::string & nev,
            int erkezes_ora,
            int erkezes_perc,
            int indulas_ora,
            int indulas_perc )
```

Egyedi konstruktor.

Paraméterek

| | |
|---|---|
| *azonosito* | - Az állomás azonosítója. |
| *nev* | - Az állomás neve. |
| *erkezes_ora* | - Az érkezés órája. |
| *erkezes_perc* | - Az érkezés perce. |
| *indulas_ora* | - Az indulás órája. |
| *indulas_perc* | - Az indulás perce. |

#### 4.1.2.4. ~Allomas()

```
Allomas::~Allomas ( )
```

Az osztály destruktora.

### 4.1.3. Tagfüggvények dokumentációja

#### 4.1.3.1. changeErkezes()

```
void Allomas::changeErkezes (
            int ora,
            int perc )
```

Az érkezés idejének módosítása.

Paraméterek

| | |
|---|---|
| *ora* | - Az új érkezés órája. |
| *perc* | - Az új érkezés perce. |

#### 4.1.3.2. changeIndulas()

```
void Allomas::changeIndulas (
            int ora,
            int perc )
```

Az indulás idejének módosítása.

Paraméterek

| | |
|---|---|
| *ora* | - Az új indulás órája. |
| *perc* | - Az új indulás perce. |

#### 4.1.3.3. changeNev()

```
void Allomas::changeNev (
            const std::string & other )
```

Az állomás nevének módosítása.

**Paraméterek**

| | |
|---|---|
| *other* | - Az új név. |

#### 4.1.3.4. getErkezes()

```
Ido & Allomas::getErkezes ( )
```

Az érkezési idő referenciájának lekérdezése.

**Visszatérési érték**

Az érkezés idejének referenciája.

#### 4.1.3.5. getErkezesOra()

```
int Allomas::getErkezesOra ( )
```

Az érkezés órájának lekérdezése.

**Visszatérési érték**

Az érkezés órája.

#### 4.1.3.6. getErkezesPerc()

```
int Allomas::getErkezesPerc ( )
```

Az érkezés percének lekérdezése.

**Visszatérési érték**

Az érkezés perce.

#### 4.1.3.7. getIndulas()

```
Ido & Allomas::getIndulas ( )
```

Az indulási idő referenciájának lekérdezése.

**Visszatérési érték**

Az indulás idejének referenciája.

#### 4.1.3.8. getIndulasOra()

```
int Allomas::getIndulasOra ( )
```

Az indulás órájának lekérdezése.

**Visszatérési érték**

Az indulás órája.

#### 4.1.3.9. getIndulasPerc()

```
int Allomas::getIndulasPerc ( )
```

Az indulás percének lekérdezése.

**Visszatérési érték**

Az indulás perce.

#### 4.1.3.10. getNev()

```
std::string & Allomas::getNev ( )
```

Az állomás névének lekérdezése.

**Visszatérési érték**

Az állomás neve.

#### 4.1.3.11. operator=()

```
Allomas & Allomas::operator= (
            const Allomas & other )
```

Értékadó operátor.

#### 4.1.3.12. read()

```
void Allomas::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

**Paraméterek**

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.1.3.13. write()

```
void Allomas::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

**Paraméterek**

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.1.4. Adattagok dokumentációja

#### 4.1.4.1. allomas_azonosito

```
size_t Allomas::allomas_azonosito  [private]
```

#### 4.1.4.2. erkezes

```
Ido Allomas::erkezes  [private]
```

Az indulási időpont tárolója.

#### 4.1.4.3. indulas

```
Ido Allomas::indulas  [private]
```

Az állomás neve.

#### 4.1.4.4. nev

```
std::string Allomas::nev  [private]
```

Az állomás azonosítója.

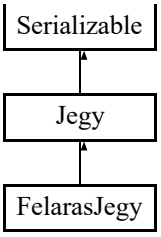
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- allomas.h
- allomas.cpp

## 4.2. FelarasJegy osztályreferencia

`#include <jegy.h>`

A FelarasJegy osztály származási diagramja:




### Publikus tagfüggvények

- FelarasJegy ()

  *A felár értéket.*
- FelarasJegy (const FelarasJegy &other)
- FelarasJegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_↔ idopont, const std::string &cel, Ido cel_idopont, double felar, std::string tipus)
- FelarasJegy & operator= (const FelarasJegy &other)
- void kiir (std::ostream &os) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~FelarasJegy ()

  *Az osztály destruktora.*

### Publikus tagfüggvények a(z) Jegy osztályból származnak

- Jegy ()

  *A jegy árát számító függvény.*
- Jegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont)
- Jegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont, double pred)
- Jegy (const Jegy &other)
- Jegy & operator= (const Jegy &other)
- bool operator== (const Jegy &other) const
- bool operator!= (const Jegy &other) const
- void kiir (std::ostream &os) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- virtual ~Jegy ()

  *Az osztály virtuális destruktora.*

### Privát attribútumok

- std::string tipus
- double felar

  *A féláras típus neve.*

**További örökölt tagok**

### Védett tagfüggvények a(z) Jegy osztályból származnak

- virtual int getAr (double pred)

  *A jegy ára.*

### Védett attribútumok a(z) Jegy osztályból származnak

- size_t jegy_azonosito
- size_t helyszam

  *A jegy azonosítója.*
- size_t kocsi_szam

  *A jegyhez tartozó helyszám.*
- size_t vonat_szam

  *A kocsi azonositója, ahol a hely található.*
- std::string indulo_nev

  *A vonat azonosítója, amelyre a jegy szól.*
- Ido indulo_ido

  *Az indulási állomas neve.*
- std::string cel_nev

  *Az indulási időpont.*
- Ido cel_ido

  *A célállomás neve.*
- int ar

  *A célállomás időpontja.*

## 4.2.1. Részletes leírás

A FélárasJegy osztály a Jegy osztály leszármazottja, és reprezentálja a féláras jegyeket. Tartalmazza a típust és a felárat.

## 4.2.2. Konstruktorok és destruktorok dokumentációja

### 4.2.2.1. FelarasJegy() [1/3]

```
FelarasJegy::FelarasJegy ( )
```

A felár értéket.

Az osztály alapértelmezett konstruktora.

### 4.2.2.2. FelarasJegy() [2/3]

```
FelarasJegy::FelarasJegy (
            const FelarasJegy & other )
```

Másoló konstruktor.

Paraméterek

| | |
|---|---|
| *other* | - másolni kívánt FélárasJegy objektum |

#### 4.2.2.3. FelarasJegy() [3/3]

```
FelarasJegy::FelarasJegy (
            size_t azonosito,
            size_t hely,
            size_t kocsi,
            size_t vonat,
            const std::string & indulo,
            Ido indulo_idopont,
            const std::string & cel,
            Ido cel_idopont,
            double felar = 0.6,
            std::string tipus = "felaras jegy" )
```

Paraméteres konstruktor.

Paraméterek

| | |
|---|---|
| *azonosito* | - a jegy azonosítója |
| *hely* | - a helyszám |
| *kocsi* | - a kocsi azonosítója |
| *vonat* | - a vonat azonosítója |
| *indulo* | - az indulási állomás |
| *cel* | - a célállomás |
| *felar* | - a felár értéket |
| *tipus_nev* | - a féláras típus neve |

#### 4.2.2.4. ~FelarasJegy()

```
FelarasJegy::~FelarasJegy ( )
```

Az osztály destruktora.

### 4.2.3. Tagfüggvények dokumentációja

#### 4.2.3.1. kiir()

```
void FelarasJegy::kiir (
            std::ostream & os ) const
```

Objektum kiírása adatfolyamba esztétikus formában.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

#### 4.2.3.2. operator=()

```
FelarasJegy & FelarasJegy::operator= (
            const FelarasJegy & other )
```

Értékadó operátor.

Paraméterek

| | |
|---|---|
| *other* | - másolni kívánt FélárasJegy objektum |

Visszatérési érték

A másolt FélárasJegy objektum referenciája.

#### 4.2.3.3. read()

```
void FelarasJegy::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

Paraméterek

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.2.3.4. write()

```
void FelarasJegy::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.2.4. Adattagok dokumentációja

#### 4.2.4.1. felar

```
double FelarasJegy::felar  [private]
```

A féláras típus neve.

#### 4.2.4.2. tipus

```
std::string FelarasJegy::tipus  [private]
```

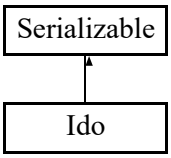
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- jegy.h
- jegy.cpp

## 4.3. Ido osztályreferencia

```
#include <ido.h>
```

Az Ido osztály származási diagramja:




**Publikus tagfüggvények**

- Ido ()
  *Az idő objektumban tárolt perc. (0-59)*
- Ido (int o, int p)
- Ido (const Ido &other)
- int getOra () const
- int getPerc () const
- void setOra (int o)
- void setPerc (int p)
- void setIdo (int o, int p)
- void addPerc (int p)
- void addOra (int o)
- bool operator== (const Ido &other) const
- bool operator!= (const Ido &other) const
- void kiir (std::ostream &os) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~Ido ()
  *Az osztály destruktora.*

**Privát attribútumok**

- int ora
- int perc

  *Az idő objektumban tárolt óra. (0-23)*

### 4.3.1. Részletes leírás

Az Idő osztály reprezentál egy időpontot. Tartalmazza az időpontot alkotó órát és percet.

### 4.3.2. Konstruktorok és destruktorok dokumentációja

#### 4.3.2.1. Ido() [1/3]

```
Ido::Ido ( )
```

Az idő objektumban tárolt perc. (0-59)

Az osztály alapértelmezett konstruktora.

#### 4.3.2.2. Ido() [2/3]

```
Ido::Ido (
            int o,
            int p )
```

Paraméteres konstruktor.

**Paraméterek**

| | |
|---|---|
| *o* | - A tárolni kívánt óra. |
| *p* | - A tárolni kívánt perc. |

#### 4.3.2.3. Ido() [3/3]

```
Ido::Ido (
            const Ido & other )
```

Az osztály másoló konstruktora.

**Paraméterek**

| | |
|---|---|
| *other* | - Másolni kívánt Ido objektum. |

#### 4.3.2.4. ~Ido()

```
Ido::~Ido ( )
```

Az osztály destruktora.

### 4.3.3. Tagfüggvények dokumentációja

#### 4.3.3.1. addOra()

```
void Ido::addOra (
            int o )
```

Óra hozzáadása az aktuális időhöz.

**Paraméterek**

| | |
|---|---|
| *o* | - A hozzáadandó óra értéke. |

#### 4.3.3.2. addPerc()

```
void Ido::addPerc (
            int p )
```

Perc hozzáadása az aktuális időhöz.

**Paraméterek**

| | |
|---|---|
| *p* | - A hozzáadandó perc értéke. |

#### 4.3.3.3. getOra()

```
int Ido::getOra ( ) const
```

Getter függvény az óra lekérdezéséhez.

**Visszatérési érték**

Az aktuális óra értéke.

#### 4.3.3.4. getPerc()

```
int Ido::getPerc ( ) const
```

Getter függvény a perc lekérdezéséhez.

**Visszatérési érték**

Az aktuális perc értéke.

#### 4.3.3.5. kiir()

```
void Ido::kiir (
            std::ostream & os ) const
```

Objektum kiírása adatfolyamba esztétikus formában.

**Paraméterek**

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

#### 4.3.3.6. operator"!=()

```
bool Ido::operator!= (
            const Ido & other ) const
```

Nem egyenlőség operátor.

**Paraméterek**

| | |
|---|---|
| *other* | - A másik idő, amivel a nem egyenlőség vizsgálat történik. |

**Visszatérési érték**

igaz, ha az idők nem egyeznek meg, különben hamis.

#### 4.3.3.7. operator==()

```
bool Ido::operator== (
            const Ido & other ) const
```

Egyenlőség operátor.

**Paraméterek**

| | |
|---|---|
| *other* | - A másik idő, amivel az egyenlőség vizsgálat történik. |

**Visszatérési érték**

igaz, ha az idők megegyeznek, különben hamis.

#### 4.3.3.8. read()

```
void Ido::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

Paraméterek

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.3.3.9. setIdo()

```
void Ido::setIdo (
            int o,
            int p )
```

Idő beállítása adott óra és perc értékekkel.

Paraméterek

| | |
|---|---|
| *o* | - A beállítandó óra értéke. |
| *p* | - A beállítandó perc értéke. |

#### 4.3.3.10. setOra()

```
void Ido::setOra (
            int o )
```

Setter függvény az óra beállításához.

Paraméterek

| | |
|---|---|
| *o* | - A beállítandó óra értéke. |

#### 4.3.3.11. setPerc()

```
void Ido::setPerc (
            int p )
```

Setter függvény a perc beállításához.

Paraméterek

| | |
|---|---|
| *p* | - A beállítandó perc értéke. |

#### 4.3.3.12. write()

```
void Ido::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.3.4. Adattagok dokumentációja

#### 4.3.4.1. ora

```
int Ido::ora  [private]
```

#### 4.3.4.2. perc

```
int Ido::perc  [private]
```

Az idő objektumban tárolt óra. (0-23)

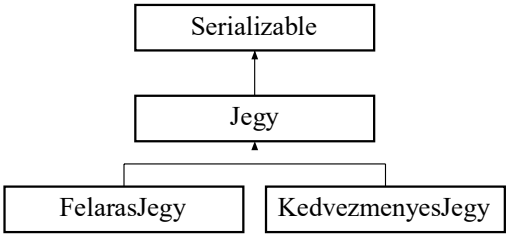
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- ido.h
- ido.cpp

## 4.4. Jegy osztályreferencia

```
#include <jegy.h>
```

A Jegy osztály származási diagramja:




**Publikus tagfüggvények**

- Jegy ()

  *A jegy árát számító függvény.*
- Jegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont)
- Jegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont, double pred)
- Jegy (const Jegy &other)
- Jegy & operator= (const Jegy &other)
- bool operator== (const Jegy &other) const
- bool operator!= (const Jegy &other) const
- void kiir (std::ostream &os) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- virtual ~Jegy ()

  *Az osztály virtuális destruktora.*

**Védett tagfüggvények**

- virtual int getAr (double pred)

  *A jegy ára.*

**Védett attribútumok**

- size_t jegy_azonosito
- size_t helyszam

  *A jegy azonosítója.*
- size_t kocsi_szam

  *A jegyhez tartozó helyszám.*
- size_t vonat_szam

  *A kocsi azonositója, ahol a hely található.*
- std::string indulo_nev

  *A vonat azonosítója, amelyre a jegy szól.*
- Ido indulo_ido

  *Az indulási állomas neve.*
- std::string cel_nev

  *Az indulási időpont.*
- Ido cel_ido

  *A célállomás neve.*
- int ar

  *A célállomás időpontja.*

### 4.4.1. Részletes leírás

A Jegy osztaly reprezentalja a vonatjegyeket. Tartalmazza a jegy azonositojat, a helyszamot, a kocsi szamat, a vonat szamat, az indulo es a cel allomast, valamint az arat.

### 4.4.2. Konstruktorok és destruktorok dokumentációja

#### 4.4.2.1. Jegy() [1/4]

```
Jegy::Jegy ( )
```

A jegy árát számíto függvény.

Az osztaly alapertelmezett konstruktora.

#### 4.4.2.2. Jegy() [2/4]

```
Jegy::Jegy (
            size_t azonosito,
            size_t hely,
            size_t kocsi,
            size_t vonat,
            const std::string & indulo,
            Ido indulo_idopont,
            const std::string & cel,
            Ido cel_idopont )
```

Paraméteres konstruktor.

Paraméterek

| | |
|---|---|
| *azonosito* | - a jegy azonosítója |
| *hely* | - a helyszám |
| *kocsi* | - a kocsi azonosítója |
| *vonat* | - a vonat azonosítója |
| *indulo* | - az indulási állomás |
| *cel* | - a célállomás |

#### 4.4.2.3. Jegy() [3/4]

```
Jegy::Jegy (
            size_t azonosito,
            size_t hely,
            size_t kocsi,
            size_t vonat,
            const std::string & indulo,
            Ido indulo_idopont,
            const std::string & cel,
            Ido cel_idopont,
            double pred )
```

Paraméteres konstruktor (alosztályhoz).

Paraméterek

| | |
|---|---|
| *azonosito* | - a jegy azonosítója |
| *hely* | - a helyszám |
| *kocsi* | - a kocsi azonosítója |
| *vonat* | - a vonat azonosítója |
| *indulo* | - az indulási állomás |
| *cel* | - a célállomás |
| *pred* | - az árat meghatározó kedvezmény vagy felár értéke |

#### 4.4.2.4. Jegy() [4/4]

```
Jegy::Jegy (
            const Jegy & other )
```

Az osztály másoló konstruktora.

Paraméterek

| | |
|---|---|
| *other* | - Másolni kívánt Jegy objektum. |

#### 4.4.2.5. ~Jegy()

```
virtual Jegy::~Jegy ( )  [inline], [virtual]
```

Az osztály virtuális destruktora.

### 4.4.3. Tagfüggvények dokumentációja

#### 4.4.3.1. getAr()

```
int Jegy::getAr (
            double pred = 1 )  [protected], [virtual]
```

A jegy ára.

#### 4.4.3.2. kiir()

```
void Jegy::kiir (
            std::ostream & os ) const
```

Objektum kiírása adatfolyamba esztétikus formában.

**Paraméterek**

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

#### 4.4.3.3. operator"!=()

```
bool Jegy::operator!= (
            const Jegy & other ) const
```

Nem egyenlőség operátor.

**Paraméterek**

| | |
|---|---|
| *other* | - A másik jegy, amivel a nem egyenlőség vizsgálat történik. |

**Visszatérési érték**

igaz, ha a jegyek nem egyeznek meg, különben hamis.

#### 4.4.3.4. operator=()

```
Jegy & Jegy::operator= (
            const Jegy & other )
```

Értékadó operátor.

Paraméterek

| | |
|---|---|
| *other* | - Másolni kívánt Jegy objektum. |

Visszatérési érték

A másolt Jegy objektum referenciája.

#### 4.4.3.5. operator==()

```
bool Jegy::operator== (
            const Jegy & other ) const
```

Egyenlőség operátor.

Paraméterek

| | |
|---|---|
| *other* | - A másik jegy, amivel az egyenlőség vizsgálat történik. |

Visszatérési érték

igaz, ha a jegyek megegyeznek, különben hamis.

#### 4.4.3.6. read()

```
void Jegy::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

Paraméterek

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

Újraimplementáló leszármazottak: KedvezmenyesJegy.

#### 4.4.3.7. write()

```
void Jegy::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

Újraimplementáló leszármazottak: KedvezmenyesJegy.

## 4.4.4. Adattagok dokumentációja

### 4.4.4.1. ar

```
int Jegy::ar  [protected]
```

A célállomás időpontja.

### 4.4.4.2. cel_ido

```
Ido Jegy::cel_ido  [protected]
```

A célállomás neve.

### 4.4.4.3. cel_nev

```
std::string Jegy::cel_nev  [protected]
```

Az indulási időpont.

### 4.4.4.4. helyszam

```
size_t Jegy::helyszam  [protected]
```

A jegy azonosítója.

### 4.4.4.5. indulo_ido

```
Ido Jegy::indulo_ido  [protected]
```

Az indulási állomas neve.

### 4.4.4.6. indulo_nev

```
std::string Jegy::indulo_nev  [protected]
```

A vonat azonosítója, amelyre a jegy szól.

#### 4.4.4.7. jegy_azonosito

```
size_t Jegy::jegy_azonosito  [protected]
```

#### 4.4.4.8. kocsi_szam

```
size_t Jegy::kocsi_szam  [protected]
```

A jegyhez tartozó helyszám.

#### 4.4.4.9. vonat_szam

```
size_t Jegy::vonat_szam  [protected]
```

A kocsi azonositója, ahol a hely található.

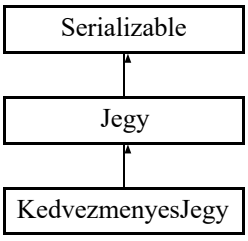
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- jegy.h
- jegy.cpp

## 4.5. KedvezmenyesJegy osztályreferencia

```
#include <jegy.h>
```

A KedvezmenyesJegy osztály származási diagramja:




### Publikus tagfüggvények

- KedvezmenyesJegy ()

  *A kedvezmény mértékét százalékban kifejező érték.*
- KedvezmenyesJegy (const KedvezmenyesJegy &other)
- KedvezmenyesJegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont, double kedvezmeny, std::string tipus)
- KedvezmenyesJegy & operator= (const KedvezmenyesJegy &other)
- void kiir (std::ostream &os) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~KedvezmenyesJegy ()

  *Az osztály destruktora.*

## Publikus tagfüggvények a(z) Jegy osztályból származnak

- Jegy ()

  *A jegy árát számító függvény.*
- Jegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont)
- Jegy (size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido indulo_idopont, const std::string &cel, Ido cel_idopont, double pred)
- Jegy (const Jegy &other)
- Jegy & operator= (const Jegy &other)
- bool operator== (const Jegy &other) const
- bool operator!= (const Jegy &other) const
- void kiir (std::ostream &os) const
- virtual ~Jegy ()

  *Az osztály virtuális destruktora.*

### Privát attribútumok

- std::string tipus
- double kedvezmeny

  *A kedvezményes típus neve.*

### További örökölt tagok

## Védett tagfüggvények a(z) Jegy osztályból származnak

- virtual int getAr (double pred)

  *A jegy ára.*

## Védett attribútumok a(z) Jegy osztályból származnak

- size_t jegy_azonosito
- size_t helyszam

  *A jegy azonosítója.*
- size_t kocsi_szam

  *A jegyhez tartozó helyszám.*
- size_t vonat_szam

  *A kocsi azonositója, ahol a hely található.*
- std::string indulo_nev

  *A vonat azonosítója, amelyre a jegy szól.*
- Ido indulo_ido

  *Az indulási állomas neve.*
- std::string cel_nev

  *Az indulási időpont.*
- Ido cel_ido

  *A célállomás neve.*
- int ar

  *A célállomás időpontja.*

### 4.5.1. Részletes leírás

A KedvezményesJegy osztály a Jegy osztály leszármazottja, és reprezentálja a kedvezményes jegyeket. Tartalmazza a típust (pl. diák, nyugdíjas) és a kedvezmény mértékét.

### 4.5.2. Konstruktorok és destruktorok dokumentációja

#### 4.5.2.1. KedvezmenyesJegy() [1/3]

```
KedvezmenyesJegy::KedvezmenyesJegy ( )
```

A kedvezmény mértékét százalékban kifejező érték.

Az osztály alapértelmezett konstruktora.

#### 4.5.2.2. KedvezmenyesJegy() [2/3]

```
KedvezmenyesJegy::KedvezmenyesJegy (
            const KedvezmenyesJegy & other )
```

Másoló konstruktor.

**Paraméterek**

| | |
|---|---|
| *other* | - másolni kívánt KedvezmenyesJegy objektum |

#### 4.5.2.3. KedvezmenyesJegy() [3/3]

```
KedvezmenyesJegy::KedvezmenyesJegy (
            size_t azonosito,
            size_t hely,
            size_t kocsi,
            size_t vonat,
            const std::string & indulo,
            Ido indulo_idopont,
            const std::string & cel,
            Ido cel_idopont,
            double kedvezmeny = -0.6,
            std::string tipus = "kedvezmenyes jegy" )
```

Paraméteres konstruktor.

**Paraméterek**

| | |
|---|---|
| *azonosito* | - a jegy azonosítója |
| *hely* | - a helyszám |
| *kocsi* | - a kocsi azonosítója |
| *vonat* | - a vonat azonosítója |
| *indulo* | - az indulási állomás |

Paraméterek

| | |
|---|---|
| *cel* | - a célállomás |
| *kedvezmeny* | - a kedvezmény mértékét kifejező érték |
| *tipus_nev* | - a kedvezményes típus neve |

#### 4.5.2.4. ~KedvezmenyesJegy()

```
KedvezmenyesJegy::~KedvezmenyesJegy ( )
```

Az osztály destruktora.

### 4.5.3. Tagfüggvények dokumentációja

#### 4.5.3.1. kiir()

```
void KedvezmenyesJegy::kiir (
            std::ostream & os ) const
```

Objektum kiírása adatfolyamba esztétikus formában.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

#### 4.5.3.2. operator=()

```
KedvezmenyesJegy & KedvezmenyesJegy::operator= (
            const KedvezmenyesJegy & other )
```

Értékadó operátor.

Paraméterek

| | |
|---|---|
| *other* | - másolni kívánt KedvezmenyesJegy objektum |

Visszatérési érték

A másolt KedvezmenyesJegy objektum referenciája.

#### 4.5.3.3. read()

```
void KedvezmenyesJegy::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

Paraméterek

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Újraimplementált ősök: Jegy.

#### 4.5.3.4. write()

```
void KedvezmenyesJegy::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Újraimplementált ősök: Jegy.

### 4.5.4. Adattagok dokumentációja

#### 4.5.4.1. kedvezmeny

```
double KedvezmenyesJegy::kedvezmeny  [private]
```

A kedvezményes típus neve.

#### 4.5.4.2. tipus

```
std::string KedvezmenyesJegy::tipus  [private]
```

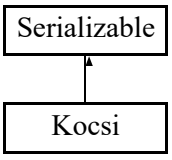
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- jegy.h
- jegy.cpp

## 4.6. Kocsi osztályreferencia

```
#include <kocsi.h>
```

A Kocsi osztály származási diagramja:



**Publikus tagfüggvények**

- Kocsi ()

  *A kocsihoz tartozo foglalt szekek szama.*
- Kocsi (const Kocsi &other)
- Kocsi (size_t azonosito, size_t szekek)
- bool tele () const
- size_t reserveHely ()
- size_t getSzekek_szama () const
- size_t getAzonosito () const
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~Kocsi ()

  *Az osztály destruktora.*

**Privát attribútumok**

- size_t kocsi_azonosito
- size_t szekek_szama

  *A kocsi azonositoja.*
- size_t foglalt_szekek_szama

  *A kocsihoz tartozo szekek szama.*

### 4.6.1. Részletes leírás

A Kocsi osztály reprezentálja a vonatkocsikat. Tartalmazza a kocsi azonosítóját, a székek számát, valamint a foglalt székek számát.

### 4.6.2. Konstruktorok és destruktorok dokumentációja

#### 4.6.2.1. Kocsi() [1/3]

```
Kocsi::Kocsi ( )
```

A kocsihoz tartozo foglalt szekek szama.

Az osztály alapértelmezett konstruktora.

#### 4.6.2.2. Kocsi() [2/3]

```
Kocsi::Kocsi (
            const Kocsi & other )
```

Az osztály másoló konstruktora.

**Paraméterek**

| | |
|---|---|
| *other* | - Másolni kívánt Kocsi objektum. |

#### 4.6.2.3. Kocsi() [3/3]

```
Kocsi::Kocsi (
            size_t azonosito,
            size_t szekek )
```

Egyedi konstruktor.

**Paraméterek**

| | |
|---|---|
| *azonosito* | - A kocsi azonosítója. |
| *szekek* | - A kocsiban található üres székek száma. |

#### 4.6.2.4. ~Kocsi()

```
Kocsi::~Kocsi ( )
```

Az osztály destruktora.

### 4.6.3. Tagfüggvények dokumentációja

#### 4.6.3.1. getAzonosito()

```
size_t Kocsi::getAzonosito ( ) const
```

Az osztály azonosítójának lekérdezése.

**Visszatérési érték**

A kocsi azonosítója.

#### 4.6.3.2. getSzekek_szama()

```
size_t Kocsi::getSzekek_szama ( ) const
```

Székek számának lekérdezése.

**Visszatérési érték**

A kocsihoz tartozó székek száma.

#### 4.6.3.3. read()

```
void Kocsi::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

Paraméterek

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.6.3.4. reserveHely()

```
size_t Kocsi::reserveHely ( )
```

Foglal egy helyet a kocsiban, és visszaadja a hely azonosítóját.

Visszatérési érték

A foglalt hely azonosítója.

#### 4.6.3.5. tele()

```
bool Kocsi::tele ( ) const
```

Megvizsgálja, hogy a kocsiban van-e üres szék.

Visszatérési érték

igaz, ha a kocsi tele van, különben hamis.

#### 4.6.3.6. write()

```
void Kocsi::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.6.4. Adattagok dokumentációja

#### 4.6.4.1. foglalt_szekek_szama

```
size_t Kocsi::foglalt_szekek_szama  [private]
```

A kocsihoz tartozo szekek szama.

#### 4.6.4.2. kocsi_azonosito

```
size_t Kocsi::kocsi_azonosito  [private]
```

#### 4.6.4.3. szekek_szama

```
size_t Kocsi::szekek_szama  [private]
```

A kocsi azonositoja.

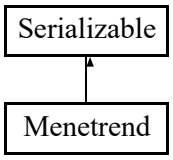
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- kocsi.h
- kocsi.cpp

## 4.7. Menetrend osztályreferencia

```
#include <menetrend.h>
```

A Menetrend osztály származási diagramja:




### Publikus tagfüggvények

- Menetrend ()

  *A menetrendben szereplő vonatok tömbje.*
- Menetrend (Vonat v)
- Menetrend (Vonat *v, size_t meret)
- Menetrend (const Menetrend &m)
- void addVonat (Vonat v)
- void changeVonat (Vonat v, size_t index)
- Vonat & getVonat (size_t index) const
- size_t getVonatokSzama () const
- void removeVonat (size_t index)
- void createJegy (std::string indulo, std::string erkezo, int indulo_ora, int indulo_perc, double discountOrFee=0, const std::string &tipus="")
- void clear ()

  *Menetrendben tárolt adatok törlése.*
- void kiir (std::ostream &os, std::string indulo, std::string erkezo) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~Menetrend ()

  *Az osztály destruktora.*

**Privát attribútumok**

- size_t vonatok_szama
- Vonat * vonatok

*A menetrendben szereplő vonatok száma.*

### 4.7.1. Részletes leírás

A Menetrend osztály reprezentálja a vonatok menetrendjét. Tartalmazza a vonatok számát és a vonatokat.

### 4.7.2. Konstruktorok és destruktorok dokumentációja

#### 4.7.2.1. Menetrend() [1/4]

```
Menetrend::Menetrend ( )  [inline]
```

A menetrendben szereplő vonatok tömbje.

Az osztály alapértelmezett konstruktora.

#### 4.7.2.2. Menetrend() [2/4]

```
Menetrend::Menetrend (
            Vonat v )
```

Paraméteres konstruktor.

**Paraméterek**

| | |
|---|---|
| *v* | - hozzáadni kívánt vonat |

#### 4.7.2.3. Menetrend() [3/4]

```
Menetrend::Menetrend (
            Vonat * v,
            size_t meret )  [inline]
```

Paraméteres konstruktor.

**Paraméterek**

| | |
|---|---|
| *v* | - a vonatok tömbje |
| *meret* | - a vonatok tömbjének mérete |

#### 4.7.2.4. Menetrend() [4/4]

```
Menetrend::Menetrend (
            const Menetrend & m )
```

Masoló konstruktor.

**Paraméterek**

| | |
|---|---|
| *m* | - másolni kívánt Menetrend objektum |

#### 4.7.2.5. ~Menetrend()

```
Menetrend::~Menetrend ( )
```

Az osztály destruktora.

### 4.7.3. Tagfüggvények dokumentációja

#### 4.7.3.1. addVonat()

```
void Menetrend::addVonat (
            Vonat v )
```

Vonat hozzáadása a menetrendhez.

**Paraméterek**

| | |
|---|---|
| *v* | - hozzáadni kívánt vonat |

#### 4.7.3.2. changeVonat()

```
void Menetrend::changeVonat (
            Vonat v,
            size_t index )
```

Vonat módosítása a menetrendben.

**Paraméterek**

| | |
|---|---|
| *v* | - a módosítandó vonat |
| *index* | - a vonat indexe a menetrendben |

#### 4.7.3.3. clear()

```
void Menetrend::clear ( )
```

Menetrendben tárolt adatok törlése.

#### 4.7.3.4. createJegy()

```
void Menetrend::createJegy (
            std::string indulo,
            std::string erkezo,
            int indulo_ora,
            int indulo_perc,
            double discountOrFee = 0,
            const std::string & tipus = "" )
```

Jegy létrehozása a megadott állomások között.

**Paraméterek**

| | |
|---|---|
| *indulo* | - Az induló állomás neve. |
| *erkezo* | - Az érkező állomás neve. |
| *indulo_ora* | - Az indulási idő órája. |
| *indulo_perc* | - Az indulási idő perce. |
| *discountOrFee* | - Kedvezmény vagy felár értéke. |
| *tipus* | - A jegy típusa. |

#### 4.7.3.5. getVonat()

```
Vonat & Menetrend::getVonat (
            size_t index ) const
```

Vonat lekérdezése a megadott indexen.

**Paraméterek**

| | |
|---|---|
| *index* | - a kívánt vonat indexe |

**Visszatérési érték**

Az adott indexen található vonat

#### 4.7.3.6. getVonatokSzama()

```
size_t Menetrend::getVonatokSzama ( ) const
```

Vonatok számának lekérdezése.

**Visszatérési érték**

A vonatok száma a menetrendben

#### 4.7.3.7. kiir()

```
void Menetrend::kiir (
            std::ostream & os,
            std::string indulo = "",
            std::string erkezo = "" ) const
```

Kiírja a menetrendben található vonatok útvonalát a megadott állomások között. Ha indulo és erkezo üres, akkor mindet. Ha csak indulo üres, akkor kiír minden útvonalat, amely az adott állomásba tart. Ha csak erkezo üres, akkor kiír minden útvonalat, amely az adott állomásból indul. Ha erkezo és indulo üres, akkor kiír minden útvonalat, amely a két állomás között van.

**Paraméterek**

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |
| *indulo* | - Az induló állomás neve. |
| *erkezo* | - Az érkező állomás neve. |

#### 4.7.3.8. read()

```
void Menetrend::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

**Paraméterek**

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.7.3.9. removeVonat()

```
void Menetrend::removeVonat (
            size_t index )
```

Vonat törlése a megadott indexen.

**Paraméterek**

| | |
|---|---|
| *index* | - a törlendő vonat indexe |

#### 4.7.3.10. write()

```
void Menetrend::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.7.4. Adattagok dokumentációja

#### 4.7.4.1. vonatok

```
Vonat* Menetrend::vonatok  [private]
```

A menetrendben szereplő vonatok száma.

#### 4.7.4.2. vonatok_szama

```
size_t Menetrend::vonatok_szama  [private]
```

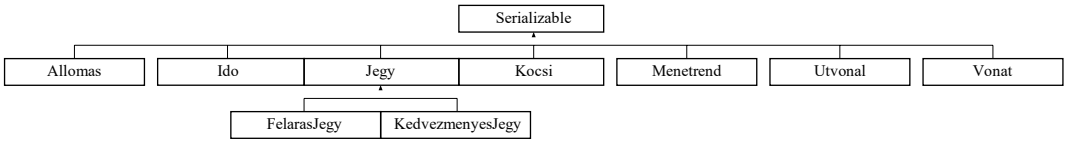
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- menetrend.h
- menetrend.cpp

## 4.8. Serializable osztályreferencia

```
#include <serializable.h>
```

A Serializable osztály származási diagramja:




**Publikus tagfüggvények**

- virtual void write (std::ostream &os) const =0
- virtual void read (std::istream &is)=0

### 4.8.1. Tagfüggvények dokumentációja

#### 4.8.1.1. read()

```
virtual void Serializable::read (
            std::istream & is )  [pure virtual]
```

Megvalósítják a következők: Allomas, Ido, Jegy, KedvezmenyesJegy, FelarasJegy, Kocsi, Menetrend, Utvonal és Vonat.

#### 4.8.1.2. write()

```
virtual void Serializable::write (
            std::ostream & os ) const  [pure virtual]
```

Megvalósítják a következők: Allomas, Ido, Jegy, KedvezmenyesJegy, FelarasJegy, Kocsi, Menetrend, Utvonal és Vonat.

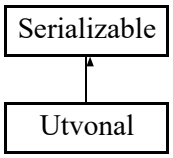
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- serializable.h

## 4.9. Utvonal osztályreferencia

```
#include <utvonal.h>
```

Az Utvonal osztály származási diagramja:




### Publikus tagfüggvények

- Utvonal ()

  *Az állomások tömbje az útvonalon.*
- Utvonal (const Utvonal &other)
- void addAllomas (Allomas allomas)
- void createAllomas (std::string nev, int erkezes_ora, int erkezes_perc, int indulas_ora, int indulas_perc)
- Utvonal & operator= (const Utvonal &other)
- void changeAllomas (size_t index, Allomas allomas)
- void removeAllomas (size_t index)
- size_t getAllomasokSzama () const

  *Az útvonalon található állomások számának lekérdezése.*
- Allomas & getAllomas (size_t i) const
- void kiir (std::ostream &os) const
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~Utvonal ()

  *Az osztály destruktora.*

### Privát attribútumok

- size_t utvonal_azonosito
- size_t allomasok_szama

  *Az útvonal azonosítója.*
- Allomas * allomasok

  *Az állomások száma az útvonalon.*

### 4.9.1. Részletes leírás

Az Útvonal osztály az egyes vonatútvonalakat reprezentálja. Tartalmazza az útvonal azonosítóját, az állomások számát, valamint az állomásokat.

### 4.9.2. Konstruktorok és destruktorok dokumentációja

#### 4.9.2.1. Utvonal() [1/2]

```
Utvonal::Utvonal ( )
```

Az állomások tömbje az útvonalon.

Az osztály alapértelmezett konstruktora.

#### 4.9.2.2. Utvonal() [2/2]

```
Utvonal::Utvonal (
            const Utvonal & other )
```

Az osztály másoló konstruktora.

**Paraméterek**

| | |
|---|---|
| *other* | - másolni kívánt Utvonal objektum |

#### 4.9.2.3. ~Utvonal()

```
Utvonal::~Utvonal ( )
```

Az osztály destruktora.

### 4.9.3. Tagfüggvények dokumentációja

#### 4.9.3.1. addAllomas()

```
void Utvonal::addAllomas (
            Allomas allomas )
```

Állomás hozzáadása az útvonalhoz.

**Paraméterek**

| | |
|---|---|
| *allomas* | - hozzáadni kívánt állomás |

#### 4.9.3.2. changeAllomas()

```
void Utvonal::changeAllomas (
            size_t index,
            Allomas allomas )
```

Állomás cseréje az útvonalon.

**Paraméterek**

| | |
|---|---|
| *index* | - az állomás indexe, amelyet cserélni kívánunk |
| *allomas* | - beszúrni kívánt állomás |

#### 4.9.3.3. createAllomas()

```
void Utvonal::createAllomas (
            std::string nev,
            int erkezes_ora,
            int erkezes_perc,
            int indulas_ora,
            int indulas_perc )
```

Új állomás létrehozása az útvonalon.

**Paraméterek**

| | |
|---|---|
| *nev* | - az állomás neve |
| *erkezes_ora* | - az állomásra érkező vonat óra |
| *erkezes_perc* | - az állomásra érkező vonat perc |
| *indulas_óra* | - az állomásról induló vonat óra |
| *indulas_perc* | - az állomásról induló vonat perc |

#### 4.9.3.4. getAllomas()

```
Allomas & Utvonal::getAllomas (
            size_t i ) const
```

Visszaad egy kívánt számú állomást a listából az index alapján.

**Paraméterek**

| | |
|---|---|
| *i* | - az állomás indexe |

**Visszatérési érték**

Az állomás referenciája az adott indexen

#### 4.9.3.5. getAllomasokSzama()

```
size_t Utvonal::getAllomasokSzama ( ) const
```

Az útvonalon található állomások számának lekérdezése.

#### 4.9.3.6. kiir()

```
void Utvonal::kiir (
            std::ostream & os ) const
```

Objektum kiírása adatfolyamba esztétikus formában.

**Paraméterek**

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

#### 4.9.3.7. operator=()

```
Utvonal & Utvonal::operator= (
            const Utvonal & other )
```

Az útvonal értékadás operátora.

**Paraméterek**

| | |
|---|---|
| *other* | - másolni kívánt Utvonal objektum |

#### 4.9.3.8. read()

```
void Utvonal::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

**Paraméterek**

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.9.3.9. removeAllomas()

```
void Utvonal::removeAllomas (
            size_t index )
```

Állomás eltávolítása az útvonalról.

Paraméterek

| | |
|---|---|
| *index* | - az eltávolítani kívánt állomás indexe |

#### 4.9.3.10. write()

```
void Utvonal::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.9.4. Adattagok dokumentációja

#### 4.9.4.1. allomasok

```
Allomas* Utvonal::allomasok  [private]
```

Az állomások száma az útvonalon.

#### 4.9.4.2. allomasok_szama

```
size_t Utvonal::allomasok_szama  [private]
```

Az útvonal azonosítója.

#### 4.9.4.3. utvonal_azonosito

```
size_t Utvonal::utvonal_azonosito  [private]
```

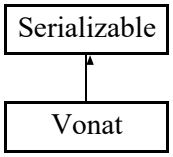
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- utvonal.h
- utvonal.cpp

## 4.10. Vonat osztályreferencia

`#include <vonat.h>`

A Vonat osztály származási diagramja:




### Publikus tagfüggvények

- Vonat ()

  *A vonathoz tartozó útvonal.*
- Vonat (const Vonat &other)
- Vonat & operator= (const Vonat &other)
- Vonat (size_t azonosito, size_t kocsik_sz, Kocsi kocsik_tomb[], Utvonal utv, size_t jegyek_sz, Jegy **jegyek_ptr)
- size_t getAzonosito () const
- Utvonal getUtvonal () const
- void setAzonosito (size_t azonosito)
- void setUtvonal (const Utvonal &utvonal)
- void addKocsi (Kocsi &kocsi)
- void createKocsi (size_t szekek)
- void addJegy (Jegy &jegy)
- Jegy * getJegy (size_t index) const
- bool routeExists (std::string indulo, std::string erkezo) const
- size_t findAllomas (std::string nev) const
- int indulasiIdoKulonbseg (std::string nev, int ora, int perc)
- size_t createJegy (std::string indulo, std::string erkezo, int indulo_ora=0, int indulo_perc=0, double discountOrFee=0, const std::string &tipus="")
- void write (std::ostream &os) const
- void read (std::istream &is)
- ~Vonat ()

  *Az osztály destruktora.*

### Privát attribútumok

- size_t vonat_azonosito
- size_t kocsik_szama

  *A vonat azonosítója.*
- Kocsi * kocsik

  *A vonathoz tartozó kocsik száma.*
- Jegy ** jegyek

  *A vonathoz tartozó kocsik tömbje.*
- size_t jegyek_szama

  *A vonathoz tartozó jegyek tömbje.*
- Utvonal utvonal

  *A vonathoz tartozó jegyek száma.*

### 4.10.1. Részletes leírás

A Vonat osztály a reprezentálja a vonatokat. Tartalmazza a vonat azonosítóját, a vonathoz tartozó kocsik számát, a kocsik tömbjét, a vonathoz tartozó jegyeket, azok számát, valamint az útvonalát.

### 4.10.2. Konstruktorok és destruktorok dokumentációja

#### 4.10.2.1. Vonat() [1/3]

```
Vonat::Vonat ( )
```

A vonathoz tartozó útvonal.

Az osztály alapértelmezett konstruktora.

#### 4.10.2.2. Vonat() [2/3]

```
Vonat::Vonat (
            const Vonat & other )
```

Az osztály másoló konstruktora.

**Paraméterek**

| | |
|---|---|
| *other* | - Másolni kívánt Vonat objektum. |

#### 4.10.2.3. Vonat() [3/3]

```
Vonat::Vonat (
            size_t azonosito,
            size_t kocsik_sz,
            Kocsi kocsik_tomb[ ],
            Utvonal utv,
            size_t jegyek_sz,
            Jegy ** jegyek_ptr )
```

Paraméteres konstruktor.

**Paraméterek**

| | |
|---|---|
| *azonosito* | - A vonat azonosítója. |
| *kocsik_sz* | - A vonathoz tartozó kocsik száma. |
| *kocsik_tomb* | - A kocsik tömbje. |
| *utv* | - Az útvonal, amelyhez a vonat tartozik. |
| *jegyek_sz* | - A vonathoz tartozó jegyek száma. |
| *jegyek_ptr* | - A jegyek tömbjére mutató pointer. |

#### 4.10.2.4. ~Vonat()

```
Vonat::~Vonat ( )
```

Az osztály destruktora.

### 4.10.3. Tagfüggvények dokumentációja

#### 4.10.3.1. addJegy()

```
void Vonat::addJegy (
            Jegy & jegy )
```

Jegy hozzáadása a vonathoz.

**Paraméterek**

| | |
|---|---|
| *jegy* | - hozzáadni kívánt jegy |

#### 4.10.3.2. addKocsi()

```
void Vonat::addKocsi (
            Kocsi & kocsi )
```

Kocsi hozzáadása a vonathoz.

**Paraméterek**

| | |
|---|---|
| *kocsi* | - hozzáadni kívánt kocsi |

#### 4.10.3.3. createJegy()

```
size_t Vonat::createJegy (
            std::string indulo,
            std::string erkezo,
            int indulo_ora = 0,
            int indulo_perc = 0,
            double discountOrFee = 0,
            const std::string & tipus = "" )
```

Új jegy létrehozása az adott indulási és érkezési állomások között.

**Paraméterek**

| | |
|---|---|
| *indulo* | - Az indulási állomás neve. |
| *erkezo* | - Az érkezési állomás neve. |
| *indulo_ora* | - Az indulási idő órája (alapértelmezett érték: 0). |
| *indulo_perc* | - Az indulási idő perce (alapértelmezett érték: 0). |
| *discountOrFee* | - Kedvezmény vagy felár mértéke (alapértelmezett érték: 0). |
| *tipus* | - Jegy típusa (alapértelmezett érték: üres string). |

Visszatérési érték

Az újonnan létrehozott jegy azonosítója.

#### 4.10.3.4. createKocsi()

```
void Vonat::createKocsi (
            size_t szekek )
```

Új kocsi létrehozása a vonathoz.

Paraméterek

| | |
|---|---|
| *szekek* | - Az új kocsin elérhető székek száma. |

#### 4.10.3.5. findAllomas()

```
size_t Vonat::findAllomas (
            std::string nev ) const
```

Megkeresi az állomás indexét az állomás neve alapján.

Paraméterek

| | |
|---|---|
| *nev* | - Az állomás neve, aminek az indexét keresi. |

Visszatérési érték

Az állomás indexe, vagy -1, ha nem található az állomás.

#### 4.10.3.6. getAzonosito()

```
size_t Vonat::getAzonosito ( ) const
```

Azonosító lekérdezése.

Visszatérési érték

A vonat azonosítója.

#### 4.10.3.7. getJegy()

```
Jegy * Vonat::getJegy (
            size_t index ) const
```

Jegy lekérdezése adott indexen.

Paraméterek

| | |
|---|---|
| *index* | - A jegy indexe a tömbben. |

Visszatérési érték

Az adott indexen található jegy pointer.

#### 4.10.3.8. getUtvonal()

```
Utvonal Vonat::getUtvonal ( ) const
```

Útvonal lekérdezése.

Visszatérési érték

Az útvonal, amelyhez a vonat tartozik.

#### 4.10.3.9. indulasiIdoKulonbseg()

```
int Vonat::indulasiIdoKulonbseg (
            std::string nev,
            int ora,
            int perc )
```

Meghatározza, hogy hány perc különbség van a megadott időpont és a vonat indulásának időpontja között.

Paraméterek

| | |
|---|---|
| *nev* | - Az állomás neve, ahonnan az indulási időt számítja. |
| *ora* | - Az indulási idő órája. |
| *perc* | - Az indulási idő perce. |

Visszatérési érték

Perc különbség van a megadott időpont és a vonat indulásának időpontja között vagy -1, ha az állomás nem található vagy a vonat hamarabb indul mint a megadott időpont.

#### 4.10.3.10. operator=()

```
Vonat & Vonat::operator= (
            const Vonat & other )
```

Értékadó operátor.

Paraméterek

| | |
|---|---|
| *other* | - Másolni kívánt Vonat objektum. |

Visszatérési érték

Az értékadás eredménye, az új Vonat objektum referenciája.

#### 4.10.3.11. read()

```
void Vonat::read (
            std::istream & is )  [virtual]
```

Serializable interfész implementációja: objektum olvasása adatfolyamból.

Paraméterek

| | |
|---|---|
| *is* | - A beolvasásra használt bemeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

#### 4.10.3.12. routeExists()

```
bool Vonat::routeExists (
            std::string indulo,
            std::string erkezo ) const
```

Ellenőrzi, hogy létezik-e útvonal az adott indulási és érkezési állomások között.

Paraméterek

| | |
|---|---|
| *indulo* | - Az indulási állomás neve. |
| *erkezo* | - Az érkezési állomás neve. |

Visszatérési érték

igaz, ha létezik útvonal, különben hamis.

#### 4.10.3.13. setAzonosito()

```
void Vonat::setAzonosito (
            size_t azonosito )
```

Azonosító beállítása.

Paraméterek

| | |
|---|---|
| *azonosito* | - Az új vonat azonosítója. |

#### 4.10.3.14. setUtvonal()

```
void Vonat::setUtvonal (
            const Utvonal & utvonal )
```

Útvonal beállítása.

Paraméterek

| | |
|---|---|
| *utvonal* | - az útvonal, amelyhez a vonat tartozik |

#### 4.10.3.15. write()

```
void Vonat::write (
            std::ostream & os ) const  [virtual]
```

Serializable interfész implementációja: objektum írása adatfolyamba.

Paraméterek

| | |
|---|---|
| *os* | - Az írásra használt kimeneti adatfolyam. |

Megvalósítja a következőket: Serializable.

### 4.10.4. Adattagok dokumentációja

#### 4.10.4.1. jegyek

```
Jegy** Vonat::jegyek  [private]
```

A vonathoz tartozó kocsik tömbje.

#### 4.10.4.2. jegyek_szama

```
size_t Vonat::jegyek_szama  [private]
```

A vonathoz tartozó jegyek tömbje.

#### 4.10.4.3. kocsik

```
Kocsi* Vonat::kocsik  [private]
```

A vonathoz tartozó kocsik száma.

#### 4.10.4.4. kocsik_szama

```
size_t Vonat::kocsik_szama  [private]
```

A vonat azonosítója.

#### 4.10.4.5. utvonal

```
Utvonal Vonat::utvonal  [private]
```

A vonathoz tartozó jegyek száma.

#### 4.10.4.6. vonat_azonosito

```
size_t Vonat::vonat_azonosito  [private]
```

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- vonat.h
- vonat.cpp

# 5. fejezet

# Fájlok dokumentációja

## 5.1. allomas.cpp fájlreferencia

```
#include "allomas.h"
#include <cstring>
#include <cstddef>
#include <iostream>
```

## 5.2. allomas.h fájlreferencia

```
#include "ido.h"
#include <iostream>
#include "memtrace.h"
#include "serializable.h"
```

### Osztályok

- class Allomas

## 5.3. allomas.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef ALLOMAS_H
00002 #define ALLOMAS_H
00003
00004 // Idő tárolásához.
00005 #include "ido.h"
00006
00007 // Szabványos bemenet és kimenet kezeléséhez.
00008 #include <iostream>
00009
00010 // Memória helyes használatának ellőrzéséhez.
00011 #include "memtrace.h"
00012
00013 // Interfész az objektumok sorosításához és deszerializálásához.
00014 #include "serializable.h"
00015
```

```
00018 class Allomas : public Serializable
00019 {
00020   size_t allomas_azonosito;
00021   std::string nev;
00022   Ido indulas;
00023   Ido erkezes;
00024 public:
00026   Allomas();
00027
00030   Allomas(const Allomas &other);
00031
00033   Allomas &operator=(const Allomas &other);
00034
00042   Allomas(size_t azonosito, const std::string &nev, int erkezes_ora, int erkezes_perc, int
      indulas_ora, int indulas_perc);
00043
00047   void changeErkezes(int ora, int perc);
00048
00052   void changeIndulas(int ora, int perc);
00053
00056   void changeNev(const std::string &other);
00057
00060   int getIndulasOra();
00061
00064   int getIndulasPerc();
00065
00068   int getErkezesOra();
00069
00072   int getErkezesPerc();
00073
00076   Ido &getIndulas();
00077
00080   Ido &getErkezes();
00081
00084   std::string &getNev();
00085
00088   void write(std::ostream &os) const;
00089
00092   void read(std::istream &is);
00093
00095   ~Allomas();
00096 };
00097
00098 #endif // ALLOMAS_H
```

## 5.4. ido.cpp fájlreferencia

```
#include "ido.h"
```

## 5.5. ido.h fájlreferencia

```
#include <iostream>
#include "serializable.h"
#include "memtrace.h"
```

### Osztályok

- class Ido

## 5.6. ido.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef IDO_H
00002 #define IDO_H
00003
00004 // Szabványos bemenet és kimenet kezeléséhez.
00005 #include <iostream>
00006
00007 // Interfész az objektumok sorosításához és deszerializálásához.
00008 #include "serializable.h"
00009
00010 // Memória helyes használatának ellőrzéséhez.
00011 #include "memtrace.h"
00012
00015 class Ido : public Serializable
00016 {
00017 private:
00018     int ora;
00019     int perc;
00020 public:
00022     Ido();
00023
00027     Ido(int o, int p);
00028
00031     Ido(const Ido &other);
00032
00035     int getOra() const;
00036
00039     int getPerc() const;
00040
00043     void setOra(int o);
00044
00047     void setPerc(int p);
00048
00052     void setIdo(int o, int p);
00053
00056     void addPerc(int p);
00057
00060     void addOra(int o);
00061
00065     bool operator==(const Ido &other) const;
00066
00070     bool operator!=(const Ido &other) const;
00071
00074     void kiir(std::ostream &os) const;
00075
00078     void write(std::ostream &os) const;
00079
00082     void read(std::istream &is);
00083
00085     ~Ido();
00086 };
00087
00088 #endif
```

## 5.7. jegy.cpp fájlreferencia

```
#include "jegy.h"
#include <iostream>
```

## 5.8. jegy.h fájlreferencia

```
#include <iostream>
#include "serializable.h"
#include "memtrace.h"
#include "allomas.h"
#include "ido.h"
```

### Osztályok

- class Jegy
- class KedvezmenyesJegy
- class FelarasJegy

## 5.9. jegy.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef JEGY_H
00002 #define JEGY_H
00003
00004 // Szabványos bemenet és kimenet kezeléséhez.
00005 #include <iostream>
00006
00007 // Interfész az objektumok sorosításához és deszerializálásához.
00008 #include "serializable.h"
00009
00010 // Memória helyes használatának ellőrzéséhez.
00011 #include "memtrace.h"
00012
00013 #include "allomas.h"
00014
00015 // Idő tárolásához.
00016 #include "ido.h"
00017
00021 class Jegy : public Serializable
00022 {
00023 protected:
00024   size_t jegy_azonosito;
00025   size_t helyszam;
00026   size_t kocsi_szam;
00027   size_t vonat_szam;
00028   std::string indulo_nev;
00029   Ido indulo_ido;
00030   std::string cel_nev;
00031   Ido cel_ido;
00032   int ar;
00033   virtual int getAr(double pred);
00034 public:
00036   Jegy();
00037
00045   Jegy(size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido
      indulo_idopont,
00046        const std::string &cel, Ido cel_idopont);
00047
00056   Jegy(size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo, Ido
      indulo_idopont,
00057        const std::string &cel, Ido cel_idopont, double pred);
00058
00061   Jegy(const Jegy &other);
00062
00066   Jegy &operator=(const Jegy &other);
00067
00071   bool operator==(const Jegy &other) const;
00072
00076   bool operator!=(const Jegy &other) const;
00077
00080   void kiir(std::ostream &os) const;
00081
00084   void write(std::ostream &os) const;
00085
00088   void read(std::istream &is);
00089
00091   virtual ~Jegy() {}
00092 };
00093
00096 class KedvezmenyesJegy : public Jegy
00097 {
00098   std::string tipus;
00099   double kedvezmeny;
00100 public:
00102   KedvezmenyesJegy();
00103
00106   KedvezmenyesJegy(const KedvezmenyesJegy &other);
00107
00117   KedvezmenyesJegy(size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string
      &indulo, Ido indulo_idopont,
00118                    const std::string &cel, Ido cel_idopont, double kedvezmeny, std::string tipus);
```

```
00119
00123   KedvezmenyesJegy &operator=(const KedvezmenyesJegy &other);
00124
00127   void kiir(std::ostream &os) const;
00128
00131   void write(std::ostream &os) const;
00132
00135   void read(std::istream &is);
00136
00138   ~KedvezmenyesJegy();
00139 };
00140
00143 class FelarasJegy : public Jegy
00144 {
00145   std::string tipus;
00146   double felar;
00147 public:
00149   FelarasJegy();
00150
00153   FelarasJegy(const FelarasJegy &other);
00154
00164   FelarasJegy(size_t azonosito, size_t hely, size_t kocsi, size_t vonat, const std::string &indulo,
      Ido indulo_idopont,
00165               const std::string &cel, Ido cel_idopont, double felar, std::string tipus);
00166
00170   FelarasJegy &operator=(const FelarasJegy &other);
00171
00174   void kiir(std::ostream &os) const;
00175
00178   void write(std::ostream &os) const;
00179
00182   void read(std::istream &is);
00183
00185   ~FelarasJegy();
00186 };
00187
00188 #endif // JEGY_H
```

## 5.10. kocsi.cpp fájlreferencia

```
#include "kocsi.h"
```

## 5.11. kocsi.h fájlreferencia

```
#include <iostream>
#include "serializable.h"
#include "memtrace.h"
```

### Osztályok

- class Kocsi

## 5.12. kocsi.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef KOCSI_H
00002 #define KOCSI_H
00003
00004 // Szabványos bemenet és kimenet kezeléséhez.
00005 #include <iostream>
00006
```

```
00007 // Interfész az objektumok sorosításához és deszerializálásához.
00008 #include "serializable.h"
00009
00010 // Memória helyes használatának ellőrzéséhez.
00011 #include "memtrace.h"
00012
00015 class Kocsi : public Serializable
00016 {
00017   size_t kocsi_azonosito;
00018   size_t szekek_szama;
00019   size_t foglalt_szekek_szama;
00020 public:
00022   Kocsi();
00023
00026   Kocsi(const Kocsi &other);
00027
00031   Kocsi(size_t azonosito, size_t szekek);
00032
00035   bool tele() const;
00036
00039   size_t reserveHely();
00040
00043   size_t getSzekek_szama() const;
00044
00047   size_t getAzonosito() const;
00048
00051   void write(std::ostream &os) const;
00052
00055   void read(std::istream &is);
00056
00058   ~Kocsi();
00059 };
00060
00061 #endif // KOCSI_H
```

## 5.13. main.cpp fájlreferencia

```
#include "menetrend.h"
#include "vonat.h"
#include "jegy.h"
#include "utvonal.h"
#include "ido.h"
#include "allomas.h"
#include "kocsi.h"
#include "test.h"
#include <iostream>
#include <sstream>
#include <fstream>
#include "memtrace.h"
```

### Függvények

- int main ()

### 5.13.1. Függvények dokumentációja

#### 5.13.1.1. main()

```
int main ( )
```

## 5.14. menetrend.cpp fájlreferencia

```
#include "menetrend.h"
```

## 5.15. menetrend.h fájlreferencia

```
#include "vonat.h"
#include <iostream>
#include "serializable.h"
#include "memtrace.h"
```

### Osztályok

- class Menetrend

## 5.16. menetrend.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef MENETREND_H
00002 #define MENETREND_H
00003
00004 #include "vonat.h"
00005
00006 // Szabványos bemenet és kimenet kezeléséhez.
00007 #include <iostream>
00008
00009 // Interfész az objektumok sorosításához és deszerializálásához.
00010 #include "serializable.h"
00011
00012 // Memória helyes használatának ellőrzéséhez.
00013 #include "memtrace.h"
00014
00017 class Menetrend : public Serializable
00018 {
00019   size_t vonatok_szama;
00020   Vonat *vonatok;
00021 public:
00023   Menetrend() : vonatok_szama(0), vonatok(NULL) {}
00024
00027   Menetrend(Vonat v);
00028
00032   Menetrend(Vonat *v, size_t meret) : vonatok_szama(meret), vonatok(v) {}
00033
00036   Menetrend(const Menetrend &m);
00037
00040   void addVonat(Vonat v);
00041
00045   void changeVonat(Vonat v, size_t index);
00046
00050   Vonat &getVonat(size_t index) const;
00051
00054   size_t getVonatokSzama() const;
00055
00058   void removeVonat(size_t index);
00059
00067   void createJegy(std::string indulo, std::string erkezo, int indulo_ora, int indulo_perc, double
      discountOrFee = 0, const std::string &tipus = "");
00068
00070   void clear();
00071
00080   void kiir(std::ostream &os, std::string indulo, std::string erkezo) const;
00081
00084   void write(std::ostream &os) const;
00085
00088   void read(std::istream &is);
00089
00091   ~Menetrend();
00092 };
00093
00094 #endif // MENETREND_H
```

## 5.17. serializable.h fájlreferencia

```
#include <iostream>
#include "memtrace.h"
```

### Osztályok

- class Serializable

## 5.18. serializable.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef SERIALIZABLE_H
00002 #define SERIALIZABLE_H
00003
00004 // Ostream és istream használatához.
00005 #include <iostream>
00006
00007 // Memória helyes használatának ellőrzéséhez.
00008 #include "memtrace.h"
00009
00010 // Absztrakt osztály a perzisztencia megvalósításához.
00011 // Tartalmazza a write() és read() virtuális függvényeket, amelyeket a leszármazott osztályoknak meg
      kell valósítaniuk.
00012 class Serializable
00013 {
00014 public:
00015   // Adatfolyamra írásért felelős függvény.
00016   // @param os - Az objektum írása során használt kimeneti adatfolyam, például egy fájl vagy a konzol
      kimenete.
00017   virtual void write(std::ostream &os) const = 0;
00018
00019   // Adatfolyamról olvasásért felelős függvény.
00020   // @param is - Az objektum olvasása során használt bemeneti adatfolyam, például egy fájl vagy a
      konzol bemenete.
00021   virtual void read(std::istream &is) = 0;
00022 };
00023
00024 #endif
```

## 5.19. test.cpp fájlreferencia

```
#include "test.h"
```

### Függvények

- void test ()

  *Programhoz tartozó teszteket futtató függvény.*

### 5.19.1. Függvények dokumentációja

#### 5.19.1.1. test()

```
void test ( )
```

Programhoz tartozó teszteket futtató függvény.

# 5.20. test.h fájlreferencia

```
#include "menetrend.h"
#include "vonat.h"
#include "jegy.h"
#include "utvonal.h"
#include "ido.h"
#include "allomas.h"
#include "kocsi.h"
#include <sstream>
#include "memtrace.h"
#include "gtest_lite.h"
```

### Függvények

- void test ()

  *Programhoz tartozó teszteket futtató függvény.*

## 5.20.1. Függvények dokumentációja

#### 5.20.1.1. test()

```
void test ( )
```

Programhoz tartozó teszteket futtató függvény.

# 5.21. test.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef TEST_H_
00002 #define TEST_H_
00003
00004 // Teszteléskor vizsgált osztályok.
00005 #include "menetrend.h"
00006 #include "vonat.h"
00007 #include "jegy.h"
00008 #include "utvonal.h"
00009 #include "ido.h"
00010 #include "allomas.h"
00011 #include "kocsi.h"
00012
00013 // Perzisztencia vizsgálatához.
00014 #include <sstream>
00015
00016 // Memória helyes használatának ellőrzéséhez.
00017 #include "memtrace.h"
00018 // Teszteléshez használt makrók.
00019 #include "gtest_lite.h"
00020
00022 void test();
00023
00024 #endif /* TEST_H_ */
```

# 5.22. utvonal.cpp fájlreferencia

```
#include "utvonal.h"
```

## 5.23. utvonal.h fájlreferencia

```
#include "allomas.h"
#include "serializable.h"
#include "memtrace.h"
```

**Osztályok**

- class Utvonal

## 5.24. utvonal.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef UTVONAL_H
00002 #define UTVONAL_H
00003
00004 #include "allomas.h"
00005
00006 // Interfész az objektumok sorosításához és deszerializálásához.
00007 #include "serializable.h"
00008
00009 // Memória helyes használatának ellőrzéséhez.
00010 #include "memtrace.h"
00011
00014 class Utvonal : public Serializable
00015 {
00016   size_t utvonal_azonosito;
00017   size_t allomasok_szama;
00018   Allomas *allomasok;
00019 public:
00021   Utvonal();
00022
00025   Utvonal(const Utvonal &other);
00026
00029   void addAllomas(Allomas allomas);
00030
00037   void createAllomas(std::string nev, int erkezes_ora, int erkezes_perc, int indulas_ora, int
      indulas_perc);
00038
00041   Utvonal &operator=(const Utvonal &other);
00042
00046   void changeAllomas(size_t index, Allomas allomas);
00047
00050   void removeAllomas(size_t index);
00051
00053   size_t getAllomasokSzama() const;
00054
00058   Allomas &getAllomas(size_t i) const;
00059
00062   void kiir(std::ostream &os) const;
00063
00066   void write(std::ostream &os) const;
00067
00070   void read(std::istream &is);
00071
00073   ~Utvonal();
00074 };
00075
00076 #endif // UTVONAL_H
```

## 5.25. vonat.cpp fájlreferencia

```
#include "vonat.h"
```

# 6. fejezet

## 5.26. vonat.h fájlreferencia

```
#include <iostream>
#include "memtrace.h"
#include "serializable.h"
#include "kocsi.h"
#include "jegy.h"
#include "utvonal.h"
```

### Osztályok

- class Vonat

## 5.27. vonat.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef VONAT_H
00002 #define VONAT_H
00003
00004 #include <iostream>
00005
00006 #include "memtrace.h"
00007
00008 #include "serializable.h"
00009 #include "kocsi.h"
00010 #include "jegy.h"
00011 #include "utvonal.h"
00012
00016 class Vonat : public Serializable
00017 {
00018 private:
00019     size_t vonat_azonosito;
00020     size_t kocsik_szama;
00021     Kocsi *kocsik;
00022     Jegy **jegyek;
00023     size_t jegyek_szama;
00024     Utvonal utvonal;
00025 public:
00027     Vonat();
00028
00031     Vonat(const Vonat &other);
00032
00036     Vonat &operator=(const Vonat &other);
00037
00045     Vonat(size_t azonosito, size_t kocsik_sz, Kocsi kocsik_tomb[], Utvonal utv, size_t jegyek_sz, Jegy
      **jegyek_ptr);
00046
00049     size_t getAzonosito() const;
00050
00053     Utvonal getUtvonal() const;
00054
00057     void setAzonosito(size_t azonosito);
00058
00061     void setUtvonal(const Utvonal &utvonal);
00062
00065     void addKocsi(Kocsi &kocsi);
00066
00069     void createKocsi(size_t szekek);
00070
00073     void addJegy(Jegy &jegy);
00074
00078     Jegy *getJegy(size_t index) const;
00079
00084     bool routeExists(std::string indulo, std::string erkezo) const;
00085
00089     size_t findAllomas(std::string nev) const;
00090
00097     int indulasiIdoKulonbseg(std::string nev, int ora, int perc);
00098
00107     size_t createJegy(std::string indulo, std::string erkezo, int indulo_ora = 0, int indulo_perc = 0,
      double discountOrFee = 0, const std::string &tipus = "");
```

```
00108
00111     void write(std::ostream &os) const;
00112
00115     void read(std::istream &is);
00116
00118     ~Vonat();
00119 };
00120
00121 #endif // VONAT_H
```

# 6. fejezet

# Tesztek

1. Idő osztály kiírás funkciójának vizsgálata (IdoKiiras, Test1):

   - Cél: Az idő objektumok helyes kiírásának ellenőrzése.
   - Módszer: Két idő objektum létrehozása és kiírása egy stringstream-be, majd a kiírt értékek ellenőrzése.
   - Eredmény: Az idő objektumok kiírásának megfelelő formátumban kell megjelennie ("10:12\n02:01\n").

2. Idő osztály perzisztenciájának vizsgálata (IdoPerzisztencia, Test2):

   - Cél: Az idő objektumok helyes mentésének és visszaolvasásának ellenőrzése.
   - Módszer: Egy idő objektumot kiírunk egy stringstreambe, majd egy másik idő objektumba beolvassuk, és az egyenlőséget ellenőrizzük.
   - Eredmény: A két idő objektumnak egyenlőnek kell lennie.

3. Idő osztály alapvető műveleteinek vizsgálata (IdoMuveletek, Test3):

   - Cél: Az idő objektumok alapvető műveleteinek (értékbeállítás, aritmetikai műveletek) helyes működésének ellenőrzése.
   - Módszer: Egy idő objektum létrehozása és különböző értékek beállítása, majd a műveletek ellenőrzése.
   - Eredmény: Az idő objektumok értékei és műveletei helyesen kell, hogy működjenek, beleértve az egyenlőség és különbözőség operátorokat is.

4. **Állomás konstruktorának vizsgálata (AllomasKonstruktor, Test4):**

   - **Cél:** Az állomás objektum konstruktora és paraméteres konstruktora által beállított értékek helyességének ellenőrzése.
   - **Módszer:** Két állomás objektum létrehozása: egy alapértelmezett és egy paraméteres konstruktorral. Az alapértelmezett konstruktor esetében az összes paraméter 0-ra vagy üres stringre van állítva. A paraméteres konstruktorral létrehozott objektumnál a megadott értékek ellenőrzése.
   - **Eredmény:** Az alapértelmezett konstruktor által létrehozott állomás paraméterei mind 0 vagy üres string. A paraméteres konstruktor által létrehozott állomás paraméterei helyesen vannak beállítva (10:30 érkezés, 12:45 indulás, név: "Teszt Allomas").

5. **Állomás osztály attribútum változtatásának vizsgálata (AllomasModositas, Test5):**

    - **Cél:** Az állomás objektum attribútumainak helyes módosíthatóságának ellenőrzése.
    - **Módszer:** Egy állomás objektum létrehozása paraméteres konstruktorral, majd az indulás és érkezés időpontjainak, valamint a név módosítása. Az új értékek helyességének ellenőrzése.
    - **Eredmény:** Az indulás és érkezés időpontjainak, valamint a név módosítása helyesen történik (indulás: 11:00, érkezés: 10:45, név: "Uj Teszt Allomas").

6. **Állomás osztály perzisztenciájának vizsgálata (AllomasPerzisztencia, Test6):**

    - **Cél:** Az állomás objektumok helyes mentésének és visszaolvasásának ellenőrzése.
    - **Módszer:** Egy állomás objektum kiírása egy `stringstream` objektumba, majd egy másik állomás objektumba beolvasás. Az eredeti és beolvasott objektumok közötti egyezés ellenőrzése.
    - **Eredmény:** Az eredeti és a beolvasott állomás objektumok paraméterei (név, indulás, érkezés) megegyeznek.

7. **Útvonal osztály állomás hozzáadás függvényének vizsgálata (UtvonalAllomasHozzaadas, Test7):**

    - **Cél:** Az útvonal objektum állomás hozzáadásának helyes működésének ellenőrzése.
    - **Módszer:** Létrehozunk egy útvonal objektumot, majd két állomás objektumot. Az állomásokat hozzáadjuk az útvonalhoz, és ellenőrizzük, hogy az útvonal megfelelően tárolja az állomásokat.
    - **Eredmény:** Az útvonal objektum helyesen tartalmazza a hozzáadott állomásokat (2 állomás, első neve "Allomas1", második neve "Allomas2").

8. **Útvonal állomás cseréjének vizsgálata (UtvonalAllomasCsere, Test8):**

    - **Cél:** Az útvonal objektum állomás cseréjének helyes működésének ellenőrzése.

- **Módszer:** Létrehozunk egy útvonal objektumot, majd három állomás objektumot. Két állomást hozzáadunk az útvonalhoz, majd a második állomást egy harmadik állomásra cseréljük. Ellenőrizzük, hogy az állomások cseréje helyesen történt-e.
- **Eredmény:** Az útvonal objektum helyesen tartalmazza az állomásokat a csere után (2 állomás, első neve "Allomas1", második neve "Allomas3").

9. **Útvonal osztály perzisztenciájának vizsgálata (UtvonalPerzisztencia, Test9):**

- **Cél:** Az útvonal objektumok helyes mentésének és visszaolvasásának ellenőrzése.
- **Módszer:** Létrehozunk egy útvonal objektumot, majd két állomás objektumot, amelyeket hozzáadunk az útvonalhoz. Az útvonal objektumot kiírjuk egy `stringstream` objektumba, majd egy másik útvonal objektumba beolvassuk. Az eredeti és a beolvasott útvonal objektumok közötti egyezést ellenőrizzük.
- **Eredmény:** Az eredeti és a beolvasott útvonal objektumok azonos számú állomást tartalmaznak, és az állomások nevei is megegyeznek.

10. **Kocsi osztály szék foglalás funkciójának vizsgálata (KocsiHelyFoglalas, Test10):**
    - **Cél:** Ellenőrizni a kocsi osztály szék foglalás funkciójának helyes működését.
    - **Módszer:** Létrehozunk egy kocsi objektumot 5 ülőhellyel. Minden üléshelyet lefoglalunk, majd megpróbálunk újra lefoglalni egy helyet, amikor már nincs több szabad hely.
    - **Eredmény:** Az ülőhelyek foglalása helyesen működik, és hibát dob, amikor nincs több szabad hely.
11. **Kocsi osztály foglaltság ellenőrző függvényének vizsgálata (KocsiTele, Test11):**
    - **Cél:** Ellenőrizni a kocsi osztály foglaltság ellenőrző függvényének helyes működését.
    - **Módszer:** Létrehozunk egy kocsi objektumot 2 ülőhellyel, majd lefoglalunk minden üléshelyet, és ellenőrizzük, hogy az osztály helyesen jelez-e, ha nincs több szabad hely.
    - **Eredmény:** A foglaltság ellenőrzése helyesen működik, és igaz értéket ad, amikor nincs több szabad hely.
12. **Kocsi osztály perzisztenciájának vizsgálata (KocsiPerzisztencia, Test12):**
    - **Cél:** Ellenőrizni a kocsi objektumok helyes mentését és visszaolvasását.
    - **Módszer:** Létrehozunk egy kocsi objektumot 3 ülőhellyel, lefoglalunk egy helyet, majd kiírjuk egy `stringstream`-be. A kocsi objektumot beolvassuk

egy másik kocsi objektumba, és ellenőrizzük, hogy az azonosító, a székek száma és a lefoglalt hely egyezik-e.
- **Eredmény:** Az eredeti és a beolvasott kocsi objektumok azonosak, és a lefoglalt hely is megegyezik.

13. **Vonat osztály útvonal beállító és lekérdező függvényeinek vizsgálata (VonatUtvonalBeallitas, Test13):**
    - **Cél:** Ellenőrizni a vonat osztály útvonal beállító és lekérdező függvényeinek helyes működését.
    - **Módszer:** Létrehozunk egy útvonal objektumot két állomással, majd egy vonat objektumot. Beállítjuk a vonat útvonalát az előzőleg létrehozott útvonalra, majd lekérdezzük és ellenőrizzük az útvonal megfelelőségét.
    - **Eredmény:** Az útvonal beállítása és lekérdezése helyesen működik.
14. **Vonat osztály jegy hozzáadó függvényének vizsgálata (VonatJegyHozzaadas, Test14):**
    - **Cél:** Ellenőrizni a vonat osztály jegy hozzáadó függvényének helyes működését.
    - **Módszer:** Létrehozunk egy vonat objektumot, egy idő objektumot az indulási és érkezési időponttal, majd egy jegy objektumot az induló és érkező állomással. Hozzáadjuk a jegyet a vonathoz, majd ellenőrizzük, hogy a hozzáadás helyesen történt-e.
    - **Eredmény:** A jegy hozzáadása a vonathoz helyesen működik, és az újonnan hozzáadott jegy megfelel az elvárásoknak.
15. **Vonat osztály indulási idő különbség számító függvényének vizsgálata (VonatIndulasiIdoKulonbseg, Test15):**
    - **Cél:** Ellenőrizni a vonat osztály indulási idő különbség számító függvényének helyes működését.
    - **Módszer:** Létrehozunk egy útvonal objektumot két állomással, majd egy vonat objektumot, beállítjuk az útvonalát. Kiszámítjuk az indulási idő különbséget a vonat indulása és egy megadott időpont között egy adott állomásról.
    - **Eredmény:** Az indulási idő különbség helyesen számolódik ki és megfelel az elvárásoknak.

16. **Vonat osztály perzisztenciájának vizsgálata (Test16):**
    - **Cél:** Ellenőrizni, hogy a vonat objektumok helyesen kerülnek-e kiírásra és beolvasásra.
    - **Módszer:** Különböző objektumok létrehozása (kocsik, idők, jegyek, útvonal), majd ezek összefűzése egy vonat objektumba. Az írás és olvasás folyamatának végén összehasonlítjuk az eredeti és beolvasott vonat objektumokat.
    - **Eredmény:** Az írás és olvasás helyesen működik, a beolvasott vonat objektum megfelel az eredetinek.

*Jegy osztály == operátorának vizsgálata (Test17):*
- **Cél:** Ellenőrizni, hogy a Jegy osztály == operátora helyesen működik-e.
- **Módszer:** Két jegy létrehozása azonos adatokkal, majd az == operátor használata a két jegy összehasonlítására.

- **Eredmény:** Az == operátor helyesen működik, a két jegy egyezik.

***Jegy osztály másoló konstruktorának vizsgálata (Test18):***

- **Cél:** Ellenőrizni, hogy a Jegy osztály másoló konstruktora helyesen másolja-e az objektumokat.
- **Módszer:** Egy jegy létrehozása, majd annak másolása másoló konstruktorral.
- **Eredmény:** A másoló konstruktor helyesen működik, az eredeti és másolt jegy egyezik.

*Jegy osztály értékadó operátorának vizsgálata (Test19):*

- **Cél:** Ellenőrizni, hogy a Jegy osztály értékadó operátora helyesen működik-e.
- **Módszer:** Egy jegy létrehozása, majd az értékadó operátor használata egy másik, üres jegyre. Ezután összehasonlítjuk a két jegyet.
- **Eredmény:** Az értékadó operátor helyesen működik, a két jegy egyezik.

*Jegy osztály perzisztenciájának vizsgálata (Test20):*

- **Cél:** Ellenőrizni, hogy a Jegy osztály helyesen kerül-e kiírásra és beolvasásra.
- **Módszer:** Egy jegy létrehozása adott adatokkal, majd ennek kiírása és beolvasása StringStream segítségével.
- **Eredmény:** A kiírás és beolvasás helyesen működik, a beolvasott jegy megegyezik az eredetivel.

*Felarasjegy osztály perzisztenciájának vizsgálata (Test21):*

- **Cél:** Ellenőrizni, hogy a Felarasjegy osztály helyesen kerül-e kiírásra és beolvasásra.
- **Módszer:** Egy feláras jegy létrehozása adott adatokkal, majd ennek kiírása és beolvasása StringStream segítségével.
- **Eredmény:** A kiírás és beolvasás helyesen működik, a beolvasott feláras jegy megegyezik az eredetivel.

*Kedvezmenyesjegy osztály perzisztenciájának vizsgálata (Test22):*

- **Cél:** Ellenőrizni, hogy a KedvezmenyesJegy osztály helyesen kerül-e kiírásra és beolvasásra.
- **Módszer:** Egy kedvezményes jegy létrehozása adott adatokkal, majd ennek kiírása és beolvasása StringStream segítségével.
- **Eredmény:** A kiírás és beolvasás helyesen működik, a beolvasott kedvezményes jegy megegyezik az eredetivel.

*Állomás konstruktorának vizsgálata (Test23):*

- **Cél:** Ellenőrizni, hogy a Menetrend osztály konstruktora helyesen inicializálja-e az objektumot.
- **Módszer:** Menetrend objektum létrehozása és a vonatok számának ellenőrzése.
- **Eredmény:** A konstruktor helyesen inicializálja az objektumot.

*Menetrend vonat hozzáadó függvényének vizsgálata (Test24):*

- **Cél:** Ellenőrizni, hogy a Menetrend osztály helyesen kezeli-e a vonatok hozzáadását.
- **Módszer:** Menetrend objektum létrehozása, majd egy vonat létrehozása útvonallal és hozzáadása a menetrendhez.
- **Eredmény:** A vonat sikeresen hozzáadásra került a menetrendhez, és a vonatok száma növekedett 1-gyel.

*Menetrend vonat módosításának vizsgálata (Test25):*

- **Cél:** Ellenőrizni, hogy a Menetrend osztály helyesen kezeli-e a vonatok cseréjét.

- **Módszer:** Menetrend objektum létrehozása, majd két útvonallal rendelkező vonat létrehozása és hozzáadása a menetrendhez. Az egyik vonat cseréje a másikra.
- **Eredmény:** A vonat sikeresen lecserélődött a menetrendben.

*Menetrend osztály perzisztenciájának vizsgálata (Test26):*

- **Cél:** Ellenőrizni, hogy a Menetrend osztály helyesen kerül-e kiírásra és beolvasásra.
- **Módszer:** Menetrend objektum létrehozása, majd egy vonat létrehozása és hozzáadása a menetrendhez. A menetrend kiírása StringStream-be, majd beolvasása.
- **Eredmény:** A kiírás és beolvasás helyesen működik, a beolvasott menetrend megegyezik az eredetivel.