NHF2 – Terv

Hubai Rajmund Szilveszter CQFKI2

Egy nyilvántartásjellegű programot, azon belül a „Vonatjegy” eladó rendszer megvalósítását választottam. A program képes lesz vonat összes CRUD funkcióját ellátni (Create, Read, Update, Delete), menetrend elkészítésére, jegy kiadására.

# Az osztályok közötti kapcsolat

A program az alábbi módon fog működni, egyelőre osztályok közötti hivatkozások bemutatását az alábbi UML diagram mutatja:

A program „.txt” kiterjesztésű fájlokban fogja tárolni az adatokat, ezeket elindulásakor a program beolvassa, majd tömbökben tárolja, hogy ne kelljen újra kiolvasni a fájlból az adatokat, amikor csak használva van. A fájlokból „**;**”-tal elválasztva, az alábbi látható módon a jegyek lesznek eltárolva:

ID;vonatID;kocsiSzam;hely;allInd;allErk;ind;erk;utasNev;szulIdo;kedvez;ar

A vonatjegyek a való élettel ellentétben kizárólag csak egyetlen vonatra érvényesek, mivel minden esetben fel kell tüntetni a helyjegyet. A „kedvez” paraméterrel a program kezelni fogja az egyes kedvezményeket.

Paraméterek – zárójelben jelölve, milyen változóval lesz eltárolva a programban

* ID: a jegy egyedi azonosítója, pl.: 1530113 (int)
* vonatID: a vonat egyedi azonosítója, pl.: IC994 (string)
* kocsiSzam: a vonat kocsiszáma, ahova a jegy szól, pl.: 9 (int)
* hely az ülés száma, ahova a jegy szól, pl.: 32 (int)
* allInd: állomás, ahonnan érvényes a jegy, pl.: Kelenföld (string)
* allErk: az állomás, ameddig érvényes a jegy, pl.: Győr (string)
* ind: várható időpont, amikor a vonat indul, pl.: 15:32 (int[])
* erk: várható időpont, amikor a vonat érkezik, pl.: 16:56 (int[])
* utasNev: utas neve, pl.: Gipsz Jakab (string)
* szulIdo: utas születési ideje, pl.: 1970.01.01 (Ido())
* kedvez: kedvezmények, pl.: „50% diák (nappali)” (string[])
* ar: a jegy ára ft-ban (int)

A fenti paraméterek egy jegyhez szólnak, természetesen a vonatokhoz is ehhez hasonló módon lesz eltárolva, szintúgy az alább látható módon:

vonatNeve;vonatID;teljesMenetido;vonatTipus

Paraméterek – zárójelben jelölve, milyen változóval lesz eltárolva a programban

* vonatID: vonat egyedi azonosítója, pl.: IC994, S70, Z70 (string)
* vonatNev: az adott vonat neve, pl.: 994 Scarbantia Intercity (string)
* teljesMenetido: az egész vonat útjának ideje percben, pl.: 182 (int)
* vonatTipus: vonat típusa, pl.: Intercity, Railjet, Személyvonat (string)
* allomasok vonat állomásai egy tömbben (string[])
* kocsik vonat kocsijai, sorszáma és helykapacitása (int[][])

Az alábbi paraméterek a menetrendhez szólnak, az alábbi módon lesz eltárolva hasonló módon „.txt” fájlban:

vonatID;allomasIDmerre;erkIdo

Paraméterek – zárójelben jelölve, milyen változóval lesz eltárolva a programban

* vonatID: vonat egyedi azonosítója, pl.: IC994 (string)
* allomasID: állomás egyedi azonosítója, pl.: GY101 (string)
* merre: vonat úticélja, pl.: Győr-Gyártelep (string)
* erkIdo: vonat állomásra való beérkezésének az időpontja (Ido())

Az alábbi paraméterek az Ido osztályhoz szólnak, kivételesen ezek nem lesznek eltárolva „.txt” fájlban.

Paraméterek – zárójelben jelölve, milyen típusúak az attribútumok:

* ev (int)
* honap (int)
* nap (int)
* ora (int)
* perc (int)

# Egyéb és fontosabb algoritmusok

A menü switch-case-zel lesz megoldva a main() függvényben. Lépkedni az opciók között billentyűkkel lehet, a switch egyes case-eiben fogják megjeleníteni az új menüpontokat. Ez a menü egy while ciklusban lesz megvalósítva, és addig fog futni, ameddig a felhasználó le nem nyomja az adott billentyűt, ahogy a lenti példa mutatja:

1. Új jegy felvétele
2. Menetrendek
3. Vonat testreszabása
4. Kilépés

A menüpont kiválasztásához, nyomja meg a hozzátársított számot!

A program kezdetekor beolvassa a kívánt fájlokat, ezekhez lesz külön-külön, ilyen célra lementett „.txt” kiterjesztésű fájlok, ahonnan be lesz olvasva. Ezeket tömbökben tárolja, ezen osztályok összekötésére szolgál majd a vonatID, hiszen mind a jegyhez, mind a vonathoz és mind a menetrend osztályhoz szükséges.

## Vonat függvényei

A konstruktor függvényekkel természetesen az osztály példányosítását lehet megvalósítani. Van egy paraméter nélküli konstruktor, és van egy, amiben az összes paraméter van.

void kiir() kiírja az objektum adatait a konzolra tördelten

bool operator==() ez a függvény overload-olja a már meglévő „operator==” függvényt, ez a függvény teszi lehetővé, hogy két Vonat objektum azonos-e vagy sem

void incrementIdo() ez a függvény lehetővé teszi, hogy a vonat teljes menetidejét növelje

void decrementido() ez a függvény lehetővé teszi, hogy a vonat teljes menetidejét csökkentse

bool cmpID() ezzel a függvénnyel ellenőrizni lehet

void update() ezzel a függvénnyel lehet módosítani egy objektum attibútumai közül valamelyiket. Ez sablonosan fog működni, tehát nem lesz külön minden egyes attribútumra egy update függvény, hanem a megadott paraméter alapján fogja frissíteni az adott tulajdonságát.

bool allomasKeres() ez a függvény ellenőrzi, hogy paraméterként kapott állomáson megáll-e a vonat

void allomasAdd() ezzel a függvénnyel lehet beszúrni egy új állomást

## Jegy függvényei

A konstruktor függvényekkel természetesen az osztály példányosítását lehet megvalósítani. Van egy paraméter nélküli konstruktor, és van egy, amiben az összes paraméter van.

void kiir() kiírja az objektum adatait a konzolra tördelten

bool jegyErvenyesseg() ez a függvény ellenőrzi, hogy az adott jegy érvényes-e a vonaton a megadott helyen és időpontban

void arKiszamitasa() ez a függvény kiszámítja a jegy árát figyelembe véve a kedvezmények és a vonat típusát

void update() ezzel a függvénnyel lehet módosítani egy objektum attibútumai közül valamelyiket. Ez sablonosan fog működni, tehát nem lesz külön minden egyes attribútumra egy update függvény, hanem a megadott paraméter alapján fogja frissíteni az adott tulajdonságát.

## Menetrend függvényei

A konstruktor függvényekkel természetesen az osztály példányosítását lehet megvalósítani. Van egy paraméter nélküli konstruktor, és van egy, amiben az összes paraméter van.

void kiir() kiírja az objektum adatait a konzolra tördelten

bool vonatMegall\_e() ezzel a függvénnyel lehet ellenőrizni, hogy a paraméterként adott vonat objektum megáll-e az állomáson, vagy nem

void update() ezzel a függvénnyel lehet módosítani egy objektum attibútumai közül valamelyiket. Ez sablonosan fog működni, tehát nem lesz külön minden egyes attribútumra egy update függvény, hanem a megadott paraméter alapján fogja frissíteni az adott tulajdonságát.

## Ido függvényei

A konstruktor függvényekkel természetesen az osztály példányosítását lehet megvalósítani. Van egy paraméter nélküli konstruktor, van egy két paraméteres konstruktor, amivel az óra;perc szerint lehet példányosítani, illetve van egy három paraméteres, amivel év.hónap.nap formátumban lehet példányosítani.

int getNapokAHonapban()

ez a függvény visszaadja az adott hónap napjainak számát, ügyelve a szökőévekre

Ido operator-() ez a függvény lehetővé teszi, hogy két időt ki lehessen vonni egymásból