Szombathy Tamás

JKSVQF

TicketToGo vonatjegyrendszer

Programozói dokumentáció

A program vázlata

A program menüvezérelt, nem tartalmaz grafikus felhasználói felületet (GUI), a menüpontok közül a menüponthoz rendelt menüszám beírásával lehet választani. A TicketToGo osztályban lévő main() függvény indul el a program elindításakor, innen a megfelelő menüszám beírása után a program az egyes menüpontoknak dedikált osztályok action() függvényeivel végez feladatokat. A program elindulásakor elvégzi a szükséges inicializálásokat (ha kell) és betölti az előző futásból származó adatokat (ha lehet). Ugyanígy a kilépésnél/leállásnál menti az adatokat. A program a TicketToGo osztály köré épül, ha az eköré épülő osztályokat kategorizálni kéne, akkor a Train, a Ticket és a Wagon osztályok az adatszerkezetet megvalósító osztályok, a TrainNumberTaken és a Serializer a segédosztályok végül pedig a DeleteTrain, az Exit, az OverwriteTrain, a RevertDatabase, a TicketMaker, a TrainLister és a TrainMaker a menüpont osztályok, amik beletartoznak a segédosztályok halmazba, a menüpont osztályok közös őse, összekötő eleme a TicketToGo osztállyal a Menupoint osztály, amelyből minden menüpont osztály származik. A program osztályai egy csomagban, a jegyrendszerben találhatóak.

Adatszerkezetek

Programkönyvtár

A program working directory-jában kettő fontos mappát hoz létre magának a program. Az egyik az application_data, ahol a programban létrehozott adatokat fogja tárolni és a printed_tickets mappát, ahova a nyomtatott jegyeket helyezi el. Az application_data mappában két fájl, a Trains és a Tickets lesz tárolva. A Trains a program által futás alatt létrehozott Train objektumokat és azok adatait fogja tárolni, a Tickets pedig Ticket típusúakat. Ezzel megvalósul, hogy a program ne

„felejtse” el a benne létrehozott vonatokat és jegyeket. Ezt szerializálással éri el, aminek a pontos módját a Serializer osztály leírása tartalmazza. Ezeket a mappákat és fájlokat nem a felhasználónak kell létrehoznia, a program ezt automatikusan elvégzi.

Osztályok

TicketToGo

Ez az osztály tartalmazza a főmenüt ami egy main() függvény, az adattárolókként szolgáló trains és tickets ArrayList-eket, (illetve azok néhány műveleti függvényét) és a menupoints ArrayList-et, ami a főmenüben elérhető menüpontokat (menüpont objektumokat) tartalmazza. A trains ArrayList-ben Train típusú objektumok vannak, a tickets-ben Ticket típusúak és a menupoints-ban pedig a Menupoint osztályból leszármazó alosztályok.

main(): statikus, visszatérési értékkel nem rendelkező (void) függvény, ami a corruption_found függvény meghívásával kezdődik, amelyik ellenőrzi a program mappájában a működéshez szükséges mappák és adatok meglétét. Ez azért fontos, mert ezekből a mappákból szeretnénk egy sorral később betölteni az adatokat, ha nincs miből, a program kifagyna, a függvényhívás eredménye azonban ha true értéket ad, a program kihagyja az adatok betöltését. A főmenü rész egy while (true) blokk miatt mindaddig fut, amíg a felhasználó a 7-es menüpontot nem választja, vagy ameddig nem történik hiba/dobódik kivétel. Amíg ez nem történik meg, a felhasználótól kéri a kívánt menüponthoz rendelt szám beírását, amit egy Scanner objektummal old meg. A menupoints ArrayList-ben található menüpontok közül annak az menüpont objektumnak az action() metódusát hívja meg, amelyik menuid attribútuma megegyezik a beírt számmal. A menüpontok a következők:

1. Vonat létrehozása
2. Jegy kiadása
3. Vonatok kilistázása
4. Vonat adatainak felülírása
5. Vonat törlése
6. Változtatások visszavonása
7. Mentés és kilépés

Osztály függvényei:

getTrains(): statikus függvény, a trains ArrayList tartalmával tér vissza ami Train típusú objektumokat tartalmaz.

listTrains(): statikus, visszatérési értékkel nem rendelkező(void) függvény, a kimenetre kiírja az összes trains ArrayList-ben lévő Train típusú objektumot a Train objektum toString() metódusát használva. Ha üres a trains lista, akkor a következőt írja ki a kimenetre: „`Jelenleg nincs vonat a nyilvántartásban.”`

addTrain(): statikus, visszatérési értékkel nem rendelkező(void) függvény, paraméterként egy Train típusú objektumot kap, amit a trains ArrayList-be helyez.

addTicket(): statikus, visszatérési értékkel nem rendelkező(void) függvény, paraméterként egy Ticket típusú objektumot kap, amit a tickets ArrayList-be helyez.

Az osztály változói:

- trains: statikus ArrayList, ami Train típusú objektumokat tárol.
- tickets: statikus ArrayList, ami Ticket típusú objektumokat tárol.
- menupoints: statikus ArrayList, ami a Menupoint osztály alosztályai típusú objektumokat tárol.
- fomatter: statikus DateTimeFormatter, ami az összes LocalDateTime típusú objektumot formázza a megadott mintának megfelelően.

Train

Ez az osztály reprezentál egy vonatot a programban. Az osztály paraméterei a number, ami int típusú és a vonat számát tárolja, a wagons ,ami egy Wagon objektumokat tároló ArrayList és a vonathoz csatlakoztatott kocsikat tárolja, a startPoint, ami string típusú és a kiindulóállomás nevét tárolja , a startTime, ami LocalDateTime típusú és az indulás dátumát tárolja, az endPoint, ami string típusú és a célállomás nevét tárolja, és az endTime, ami LocalDateTime típusú és az érkezés dátumát tárolja. Mindegyik attribútum privát ebben az osztályban. Az osztály továbbá implementálja a Serializable interfészt, aminek köszönhetően az adatok elmentése (Szerializálása) lehetséges. Az osztály szülőosztálya a Ticket osztálynak, mivel attribútumaik 75% megegyezik.

Osztály függvényei:

Train(): Az osztálynak két konstruktora van, a kettő között a különbség, hogy az első dátumként már LocalDateTime objektumot kér, még a másik ugyanezt az adatot string típusban kéri és konstruktoron belül alakítja át az attribútum által megkívánt típusra. Mindkét konstruktor a Train osztály mindegyik attribútumát beállítja a kapott értékekre. A LocalDatTime-os konstruktor paraméterei a vonatszám, a kiindulási állomás neve, a LocalDatTime típusú indulási dátum, a célállomás neve és a LocalDateTime típusú érkezési dátum. A string-es konstruktor a LocalDateTime objektumok helyett string-eket vár. A string-es konstruktorra azért volt szükség, mert a vonat adatokat úgyis string-ként kapja a program a beolvaáskor, így kényelmesebb, ha ennek átalakítását a konstruktor végzi. Publikus konstruktor.

getNumber(): A vonatszám getter metódusa, az int típusú number attribútummal tér vissza. Publikus függvény.

getStartPoint(): A kiinduló állomás nevének getter metódusa, a string típusú startPoint attribútummal tér vissza. Publikus függvény.

getStartTime(): Az indulás dátumának getter metódusa, a LocalDateTime típusú startTime attribútummal tér vissza. Publikus függvény.

getEndPoint(): A célállomás nevének getter metódusa, a string típusú endPoint attribútummal tér vissza. Publikus függvény.

getEndTime(): Az érkezés dátumának getter metódusa, a LocalDateTime típusú endTime attribútummal tér vissza. Publikus függvény.

getNumberOfWagons(): Megszámolja a wagons ArrayList (.size()) hosszát és ezt az int típusú értéket adja vissza. Publikus függvény.

toString(): A vonat attribútumainak értékével tér vissza. De nem csak az értékek vannak ebben a string-ben, hanem minden adat elé oda van írva mit reprezentál. Publikus függvény.

getFreeSeats(): Megszámolja a vonat wagons ArrayList-jét végignézve , hogy összesen a kocsikban hány szabad hely van, ez adja meg, hogy hány szabad hely van a vonaton összesen. A visszatérési érték int típusú, publikus függvény.

connectWagons(amount: int): Létrehoz és hozzáad annyi wagon objektumot a wagons ArrayList-hez amennyi az amount int típusú változóban meg van adva. Publikus függvény.

reserveSeat(): Végigmegy a wagons ArrayList-en és ha egy kocsiban van szabad hely, a Wagon osztály reserveSeat() függvényével lefoglal egyet, visszatérési értéke pedig egy int-eket tároló tömb aminek az első eleme a kocsiszám, ahol a helyet sikerült lefoglalni, a második az ülésszáma az ülésnek, amit sikerült lefoglalni.Ha nem sikerül lefoglalni helyet akkor null értékkel tér vissza. Publikus függvény.

Wagon

Ez az osztály reprezentál egy kocsit a programban. Az osztály paraméterei a number, ami int típusú és a kocsi számát reprezentálja, a reserved, ami int értékeket tárol, ArrayList típusú és a foglalt ülésszámokat tárolja, és a free, ami int értékeket tárol, ArrayList típusú és a szabad ülésszámokat tárolja. A felsorolt attribútumok mind privátak. A két ArrayList-nek az elemszáma maximum 30 lehet (tehát pl a reserved maximum 30 elemet tárolhat). Az osztály implementálja a Serializable interfészt, hogy a vonattal együtt szerializálni lehessen.

Osztály függvényei:

Wagon(): Az osztály egyetlen konstruktora, a kocsiszámot kapja csak paraméterként, ezt be is állítja a number attribútum értékének. A free attribútumot egy for ciklussal feltölti 1-30-ig számokkal, a reserved attribútumot pedig nem állítja. Publikus konstruktor.

getAmountOfFreeSeats(): A free attribútum (ArrayList) elemszámát adja vissza értéknek ami int típusú. Publikus függvény.

getNumber(): A number attribútum értékével tér vissza (getter metódusa) ami int típusú. Publikus függvény.

reserveSeat(): Feladata a kocsiban lefoglalni egy helyet. Ezt úgy teszi meg, hogy a free ArrayList-ből (attribútum), ha van benne elem, akkor az első elemet áthelyezi a reserved ArrayList-be (attribútum), illetve ezt az ülésszámot egy int

típusú értékben visszaadja. Ha nem talál üres helyet a kocsiban, 0-a értéket ad vissza. Publikus metódus.

printFreeSeats(): Egy for-each ciklussal a free ArrayList-ben lévő ülésszámokat írja ki a kimenetre. Nincs visszatérési értéke, publikus metódus.

Ticket

Ez az osztály egy jegyet reprezentál a programban. A Train osztály gyereke, az ősosztályt csak két attribútummal egészíti ki, a seatnumber-el, ami int típusú és a jegyen szereplő ülésszámot adja, illetve a wagonnumber-el ami szintén int típusú és a jegyen szereplő kocsiszámot adja. Mindkét attribútum privát. Az osztály alapvetően az ősosztály konstruktorát használja, azonban az „új" értékeket is beállítja. Az osztály, mivel a Train osztály gyereke örökli a Szerializálhatóság tulajdonságát.

Osztály függvényei:

Ticket(): Az osztály egyetlen konstruktora, paraméterként kapja az ülésszámot, ami int típusú, a kocsiszámot, ami int típusú, a vonatszámot, ami megint int típusú, a kiinduló állomás nevét, ami string típusú, az indulás dátumát, ami LocalDateTime típusú, a célállomás nevét, ami string típusú és végezetül az érkezés dátumát, ami LocalDateTime típusú. Belül a Train-el közös attribútumok beállítását rábízza az ős konstruktorra (ami ezesetben nem a string-es megoldás), illetve beállítja az „új" (az ősosztályban nem létező) attribútumokat. Publikus konstruktor.

toString(): Az attribútumok értékeit adja vissza, de az értékek előtt mindig ott van, hogy mit reprezentál (pl: Vonatszám: 1). Ezt egy egységes string típusú változóban adja vissza. Azért nem a Train (ősosztálya) toString() metódusát használja, mert kicsit módosított sorrendben írja ki az adatokat ahhoz képest. Publikus metódus.

printTicket(): Feladata egy .txt fájlba „kinyomtatni" egy jegyet, tehát olvasható/rendezett módon beleírni egy text fájlba a jegy adatait. A fájlok (hogy csökkentsük az esélyét szinte 0-ra annak, hogy file already exists error-t kapjunk) nevei az aktuális dátum nano értékéből, a vonatszámból, a kocsiszámból és az

ülésszámból állnak. Ezek az adatok a névben kötőjellel vannak elválasztva. Az így kiexportálandó jegyeket a program working directory-ján belül a printed_tickets mappába helyezi. Sikeres kiexportáláskor a kimeneten jelzi ezt. Publikus metódus.

Menupoint

Feladata mint egy családban a családfőnek összetartani és egy közös pontra visszavezetni a belőle leszármazó menüpont osztályokat. Egyszóval ősosztálya minden menüpont osztálynak. Az osztály maga absztrakt, azaz nem példányosítható. Két attribútuma a menuid, ami int típusú és a menupointname, ami string típusú. Mindkét attribútum privát. Egy konstruktora van az osztálynak.

Osztály függvényei:

Menupoint(): Két paramétere a menüpontszáma = menuid, ami int típusú, illetve a menüpont neve = menupointname, ami string típusú. Publikus konstruktor.

getMenuid(): A menuid attribútum getter metódusa, visszaadja a menüpontszámot. Publikus metódus.

getMenupointName(): A menupointname getter metódusa, visszaadja a menüpont nevét. Publikus metódus.

action(): Absztrakt metódus, ezt fogja minden egyes, az ebből asz osztályból származó osztály saját action() metódusa felülírni.

DeleteTrain

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): Mint azt az osztály neve is sugallja, az osztály feladata törölni egy vonatot a nyilvántartásból, ezt pedig az action() függvényével végzi el. A függvény először minden, a TicketToGo trains ArrayList-ben lévő vonatot a Train osztály toString() metódusával kiír a kimenetre, bekéri annak a vonatnak a vonatszámát, amelyiket törölni szeretnénk. Hogy ne pluszba tároljon vonatokat ideiglenesen, egy indexcache nevű ArrayList-be, már amikor kilistázta a vonatokat, a

vonatszámukat elhelyezte, így nem a vonatokat tárolja, hanem csak a vonatok vonatszámait az indexcache-ben. Ezután a kért vonatszámot az indexcache-ben megkeresi majd ahányadik indexen megtalálja, azt az indexű vonatot törli az eredeti trains ArrayList-ből. Ha nemlétező vonatszámot ír be a felhasználó, a program jelzi, hogy nincs ilyen vonat a nyilvántartásban.

Exit

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): Meghívja a Serializer osztály saveData() metódusát, ami elmenti a trains és a tickets ArrayList-ek objektumait, adataikkal együtt. Kilépni a program egyébként a főmenü while ciklusának megszakításával tud, ezt a TicketToGo main() függvényében lehet megtalálni.

OverwriteTrain

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): A függvény először minden, a TicketToGo trains ArrayList-ben lévő vonatot a Train osztály toString() metódusával kiír a kimenetre, bekéri annak a vonatnak a vonatszámát, amelyik adatait felül szeretnénk írni, majd bekéri a vonat új adatait. Minden vonat létrehozásánál a program megkérdezi, hány kocsi lesz csatlakoztatva a vonathoz, ezért ezt itt is kéri. Az indexcache-es technika itt is működik. Eltávolítja a kért vonatot a TicketToGo trains ArrayList-ből és az új adatokkal létrehozott Train típusú objektumot hozzáadja a trains ArrayList-hez. Ha olyan számot ír be a felhasználó, ami egyik vonatnak sem vonatszáma, „A keresett vonat nem található” szöveget írja ki a kimenetre.

RevertDatabase

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): Meghívja a Serializer osztály loadData() metódusát, ami az application_data mappában található Trains és Tickets szerializált fájlokból visszaolvassa a nyilvántartásba az adatokat. Ebből az következik, hogy csak a programban futás alatt változtatott adatokat felejti el a program, illetve a más „kinyomtatott” jegyeket nem törli, de erre figyelmezteti is a felhasználót.

Serializer

A szerializálási (adatmentési) folyamatokat megvalósító osztály.

Osztály függvényei:

loadData(): Feladata az application_data mappában tárolt Trains és Tickets szerializált adatok visszatöltése a TicketToGo trains és tickets ArrayList-ekbe. Ehhez először egy FileInputStream típusú objektumot hoz létre amit átad egy ObjectInputStream típusú osztálynak ami paraméterként kapja az előbb létrehozott FileInputStream típusú objektumot, majd az ObjectInputStream osztály readObject() metódusával visszaolvassa az ArrayList-et a benne lévő Train, vagy Ticket (attól függ melyik fájlt olvassuk) objektumokat értékeikkel együtt.

filesystemCheck(): Feladata ellenőrizni, hogy a működéshez (adatmentés/adat visszahozása, jegy nyomtatása) a program working directory-jában megvannak-e a szükséges mappák, fájlok. Ha nincs, feladata ezeket létrehozni(mappákat a File osztály mkdir() metódusával, a fájlokat pedig a saveData() metódussal) és jelezni a „corruptiont” ami egy boolean érték, ezzel tér majd vissza a metódus. Ez jelzi azt, hogy nem kell beolvasni az adatokat, mert úgyis üresek lennének.

saveData(): Feladata az adatok mentése, vagyis a TicketToGo trains és tickets ArrayList-ekben tárolt Train illetve Ticket típusú objektumok és adataik kiszerializálása a program working directory-ján belül az application_data mappába. Az adatokat két fájlba rakja, az egyik a Trains a másik a Tickets, értelemszerű, hogy melyiket melyikbe. Ehhez FileOutputStream illetve ObjectOutputStream objektumokat és osztályaik metódusait használja.

TicketMaker

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): Feladata bekérni, hogy a felhasználó honnan, hova és mikor kíván menni, ezeknek a feltételeknek megfelelő vonatokat kilistázni és bekérni melyik vonatot választja a felhasználó. Itt is működik az indexcache megoldás. Egy vonat akkor felel meg a felhasználó által beírt feltételeknek, ha az indulóállomás neve és az érkezési állomás neve pontosan megegyezik a beírtakkal, illetve, ha a beírt indulási dátumnál később indul a vonat és van még rajta szabad hely. Ez azért fontos, mert saját bőrömön tapasztaltam, milyen rossz dolog, ha arra a vonatra adnak jegyet, ami abban a percben indul.:) Így az egyenlőséget a két dátum objektum között nem engedjük meg, esetleg még egy kis deltát hozzá lehet adni, hogy még csak olyan vonatra se adjon jegyet, ami 1 perc múlva indul és futva sem érhető el, mert a pályaudvar másik felén van, de ez minden pályaudvaron (vagy egyéb jegyrendszerrel működő egységnél így van, ahol az esemény nem vár) más és más. Ha sikerült a feltételeknek megfelelő vonatot találni és kiválasztani, akkor a kiválasztott vonaton a reserveSeat() metódussal lefoglal egy helyet a függvény és a vonat, illetve a reserveSeat() adataival létrejön egy Ticket típusú objektum, amit a TicketToGo tickets ArrayList-hez hozzáad, illetve a printTicket() metódussal „kinyomtatja” a jegyet.

TrainLister

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): Feladata a TicketToGo osztály listTrains() metódus segítségével az összes Train típusú vonat objektum kiírása a kimenetre a nyilvántartásból (trains ArrayList).

TrainLister

A Menupoint osztály egyik gyereke, az ős konstruktorát használja konstruktornak.

Osztály függvényei:

action(): Feladata, hogy bekérje, egy vonat létrehozásához szükséges adatokat a felhasználótól, ellenőrizze, hogy a beírt vonatszám nem foglalt-e (hiszen a vonatszámnak egyedinek kell lennie). Ha foglalt, akkor a TrainNumberTakenException-t dob, amit el is kap (try-catch). Ha az előbb említett eset nem következik be, a vonat létre jön, ekkor már csak a kocsik hozzákapcsolása a vonathoz szükséges. A mennyiséget ismét a felhasználótól kéri a program. Ezután a vonatot hozzáadja a TicketToGo trains ArrayList-hez a TicketToGo osztály addTrain() metódusának segítségével.

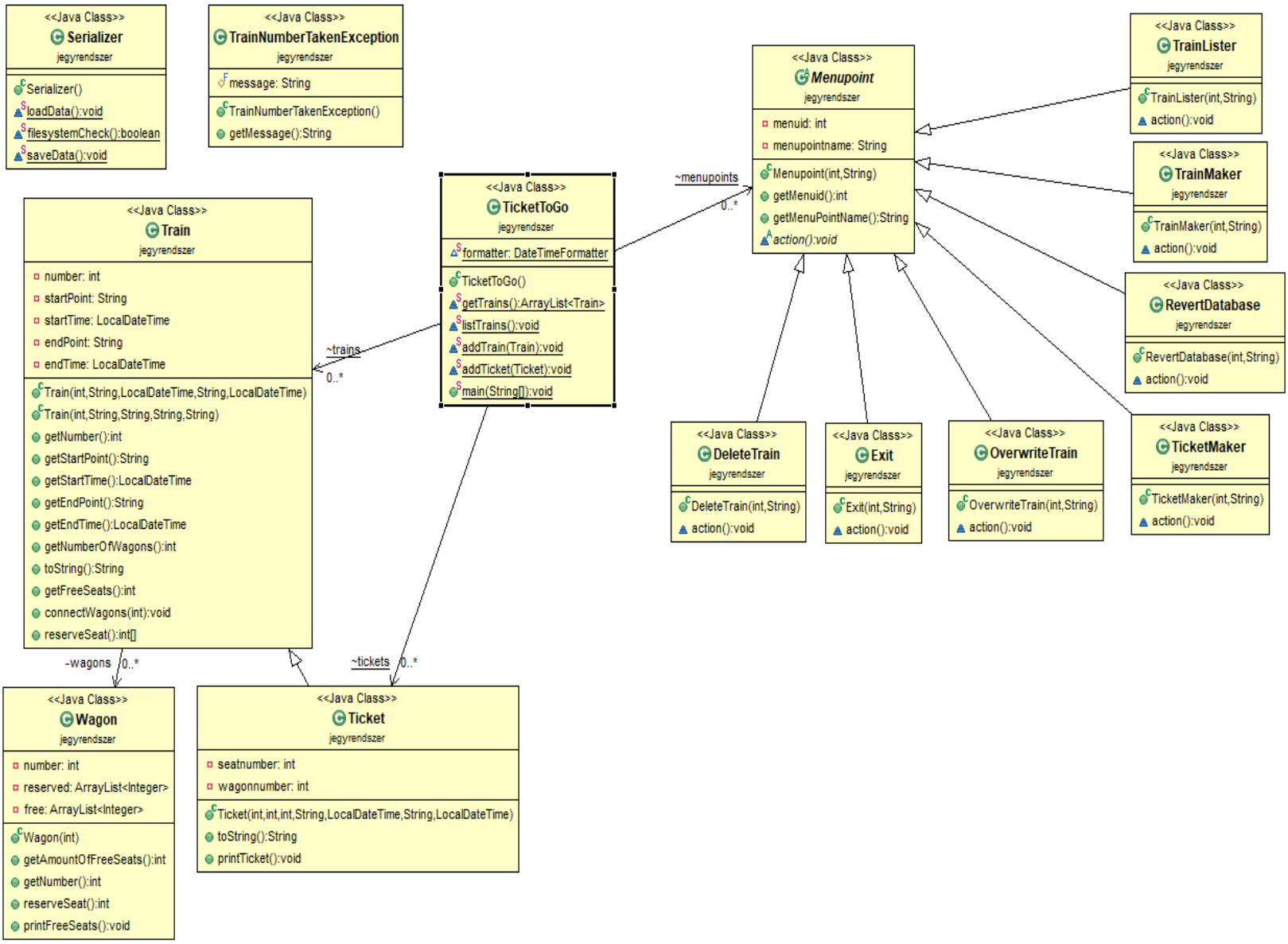
TrainNumberTakenException

Az Exception osztály gyereke, egy attribútuma van a message, ami string típusú és kezdetben megkapja a hibaüzenetet értéknek.

Osztály függvényei:

getMessage(): A message attribútum getter metódusa, a hibaüzenetet adja vissza, ami string típusú adat.

Alkalmazás UML diagramja


<<Java Class>>
Serializer
jegyrendszer
Serializer()
loadData():void
filesystemCheck():boolean
saveData():void
<<Java Class>>
TrainNumberTakenException
jegyrendszer
message: String
TrainNumberTakenException()
getMessage():String
<<Java Class>>
Menupoint
jegyrendszer
menuid: int
menupointname: String
Menupoint(int,String)
getMenuid():int
getMenuPointName():String
action():void
<<Java Class>>
TrainLister
jegyrendszer
TrainLister(int,String)
action():void
<<Java Class>>
TrainMaker
jegyrendszer
TrainMaker(int,String)
action():void
<<Java Class>>
RevertDatabase
jegyrendszer
RevertDatabase(int,String)
action():void
<<Java Class>>
TicketMaker
jegyrendszer
TicketMaker(int,String)
action():void
~menupoints
0..*
<<Java Class>>
TicketToGo
jegyrendszer
formatter: DateTimeFormatter
TicketToGo()
getTrains():ArrayList<Train>
listTrains():void
addTrain(Train):void
addTicket(Ticket):void
main(String[]):void
<<Java Class>>
Train
jegyrendszer
number: int
startPoint: String
startTime: LocalDateTime
endPoint: String
endTime: LocalDateTime
Train(int,String,LocalDateTime,String,LocalDateTime)
Train(int,String,String,String,String)
getNumber():int
getStartPoint():String
getStartTime():LocalDateTime
getEndPoint():String
getEndTime():LocalDateTime
getNumberOfWagons():int
toString():String
getFreeSeats():int
connectWagons(int):void
reserveSeat():int[]
~trains
0..*
-wagons
0..*
~tickets
0..*
<<Java Class>>
DeleteTrain
jegyrendszer
DeleteTrain(int,String)
action():void
<<Java Class>>
Exit
jegyrendszer
Exit(int,String)
action():void
<<Java Class>>
OverwriteTrain
jegyrendszer
OverwriteTrain(int,String)
action():void
<<Java Class>>
Wagon
jegyrendszer
number: int
reserved: ArrayList<Integer>
free: ArrayList<Integer>
Wagon(int)
getAmountOfFreeSeats():int
getNumber():int
reserveSeat():int
printFreeSeats():void
<<Java Class>>
Ticket
jegyrendszer
seatnumber: int
wagonnumber: int
Ticket(int,int,int,String,LocalDateTime,String,LocalDateTime)
toString():String
printTicket():void