Eötvös Loránd Tudományegyetem
Természettudományi Kar

Alkalmazott Modul III

6. gyakorlat

Objektumorientált programozás: öröklődés és polimorfizmus

© 2011.10.25. Giachetta Roberto
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

Feladat: Készítsünk el egy geometriai alakzatokat megvalósító alkalmazást, amelybe feltölthetünk különböző típusú alakzatokat (vízszintes vonal, függőleges vonal, kör, négyzet).

- az alakzatokra lehessen együttesen területet és kerültet lekérdezni, valamint meghatározni, mely alakzatok tartalmaznak pontot, illetve eltolni őket egy vektorral
- annak érdekében, hogy egységes kezelést tudjunk megvalósítani, az összes alakzatot helyezzük egy közös listába

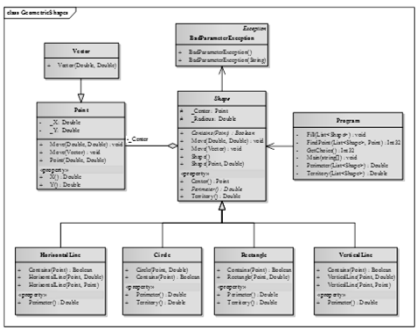
Elemzés:

- öröklődést és polimorfizmust kell használnunk egy közös ősosztállyal (`Shape`), amelyből leszármaznak a speciális osztályok (`HorisontalLine`, `VerticalLine`, `Circle`, `Rectangle`)
- mind a négy alakzattípus reprezentálható egy középponttal (`_Center`), valamint egy sugárral (`_Radious`), így a reprezentáció megvalósítható az ősosztályban, és az eltolás művelete (`Move`) is megfogalmazható annak szintjén
- az ősosztályban definiáljuk valamennyi további műveletet felüldefiniálhatóra, a területlekérdezés (`Territory`) virtuális, míg a kerületlekérdezés (`Perimeter`) és a pont tartalmazás (`Contains`) absztrakt műveletek lesznek

Elemzés:

- a megfelelő kezelésért ezen felül megvalósítjuk a pont (`Point`) és a vektor (`Vector`) osztályokat is, utóbbi az előbbi egy speciális esete
- a hibás paraméterek kezelésére definiáljunk egy új kivétel típust (`BadParameterException`), amelyet a általános kivétel típusból (`Exception`) származtatunk
- a főosztályban (`Program`) megvalósítjuk az alakzatok bekérését (`Fill`), valamint a programozási tételek végrehajtását (`Perimeter`, `Territory`, `FindPoint`), ezeket egy közös, alakzatokat tartalmazó listán (`shapeList`) futtatjuk, ebbe polimorfizmussal helyezzük a speciális osztályok példányait

Tervezés:



Feladat: Készítsünk el egy alkalmazást, amelyben vonatokat állíthatunk össze. A vonatok kocsikból állnak, melyek lehetnek mozdonyok, személykocsik, teherkocsik, és egy speciális a bicikliszállító.

- minden kocsiról ismert az azonosítószáma valamint a hossza
- mozdony esetén ismert a meghajtás típusa (gőz, elektromos, …), valamint a szállítható vagonok száma, személykocsik esetén ismert az ülések száma és az osztály, teherkocsik esetén ismert a kapacitás és az áru típusa
- a bicikliszállító személy- és teherkocsi egyben, mindig másodosztályú és mindig kerékpárt szállít

Objektumorientált programozás
Feladatok

- a vonatoknak lehessen lekérdezni a teljes hosszát, az összes ülésszámot és árukapacitást, lehessen vagonokat csatolni és leválasztani, valamint elindítani a vonatot egy megadott célállomásra (persze csak akkor, ha a mozdonyok elbírják a szerelvényt)
- mozdony csatolásánál ügyeljünk arra, hogy mindig a vonat elejére kapcsoljuk, és csak olyan mozdony engedélyezett, amelynek meghajtása megegyezik a már a vonathoz csatolt mozdonyokéval (ha van olyan)

Objektumorientált programozás
Feladatok

Elemzés:

- felvesszük az absztrakt vasúti kocsi (`RailCar`) ősosztályt, valamint annak három leszármazottját, a mozdonyt (`Locomotive`), a személykocsit (`Coach`) és az áruszállítót (`CargoCar`)
- mivel a bicikliszállítónak (`BikeCarrierCar`) mindkét osztály leszármazottjának kéne lenni, ezért valósítsunk meg két interfészt (`IPassengerCarrier`, `ICargoCarrier`), ami lehetővé teszi a közös kezelést, így a bicikliszállító is egy speciális kocsi lesz, amely megvalósítja mindkét interfészt, míg a személyszállító és az áruszállító csak a megfelelőt valósítják meg

Objektumorientált programozás
Feladatok

Elemzés:

- a vonatban egy `RailCar` lista segítségével tároljuk a kocsikat
- a vonatnál felvett lekérdező műveletek (`PassengerCapacity`, `CargoCapacity`) esetén a megfelelő interfészt fogjuk ellenőriz
- felüldefiniáljuk az `Object`-ből örökölt `ToString()` műveletet mindenhol, a megfelelő szöveges kiíráshoz
- két vasúti kocsi akkor egyenlő, ha a azonosítójuk megegyezik, ezért felüldefiniáljuk az egyelőség vizsgálatot (`Equals`) is, és ezt felhasználjuk, amikor egy új kocsit behelyezünk a vonatba

Objektumorientált programozás
Feladatok

Tervezés: