Prokkály Dávid

CMY8C0

Féléves beadandó feladat dokumentáció

Allami Beruhazasok

A program egy állami beruházásokat kezelő program, mely 2db txt fájlt olvas be:

A projectek.txt a cégek számára megpályázható projecteket tartalmaznak. A bemenet pontos követelményei a program főkönyvtárában található BemenetLeiras.txt néven.

A beszalltiok.txt a ”cégeket” tartalmazzák, melyek megfelelnek az IBeszalltio interface, és az interface-t megvalósító osztályok követelményeinek. A bemenet pontos követelményei a BemenetLeiras.txt nevű fájlban találhatók.

IBeszallito.cs:

A class-on kívül hoztam létre a szakterueletek felsorolás típust(enum), mely a következő szakterületeket tartalmazza: Tervezés, Alapozás, Útépítés, Hídépítés

Az alábbi változókat és függvényeket tartalmazza: egy string típusú Nev változót, egy Szakteruletek nevű enum típusú tömb, egy double típusú visszaosszeg, amely a cég által elvállalt projectek utáni visszafizetés(ek) összegét tárolja (ennek a 3-nak beállítottam a {get;set;}-et ahogy az interface megköveteli), egy logikai változó értékkel visszatérő ErtHozza metódus, melynek feladata az lesz, hogy az osztály (amely megvalósítja az interface-t) kap egy szakterületet, és ha megvalósító osztály tartalmazza az adott szakterületet, akkor true-t ad vissza, ha nem, akkor false-t, tartalmaz egy double típusú MegvesztegetesiAr metódust, mely majd a megvalósító osztályoknál (mindig más) vissza adja a project összegének egy bizonyos részét, vagy nem ad vissza semmit. Az OsszegLekerdezo metódus visszaadja a visszaosszeg által tárolt értéket. Az OsszegHozzaadasa metódus hozzáadja a MegvesztegetesiAr metódusból visszatérő értéket a visszaosszeg változónkhoz, az OsszegKivonasa pedig kivonja belőle a MegvesztegetesiAr metódusból visszatérő értéket (Erre a visszalépéses keresésnél van szükség).

AllamiCeg.cs:

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterülete: Tervezés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig 0-t ad vissza.

UjCeg:

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterületei: Alapozás, Útépítés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig a project ár 10%-át adja vissza.

BaratiCeg.cs:

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterületei: Tervezés, Alapozás, Útépítés, Hídépítés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig 1-t ad vissza (1M).

Maganszemely.cs:

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterületei: Tervezés, Hídépítés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig 0-t ad vissza.

KulfoldiCeg.cs

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterületei: Alapozás, Útépítés, Hídépítés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig a project ár 5%-át adja vissza.

IdegenCeg.cs:

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterülete: Tervezés, Útépítés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig 2-t ad vissza (2M).

IsmertCeg.cs:

Egy IBeszallito intercate-t megvalósító osztály, melynek szakterülete: Alapozás, Hídépítés. Ennek függvényében működik az ErtHozza metódusa, a MegvesztegetesiAr metódusa mindig 1.5-t ad vissza (1.5M).

Projectek.cs:

EgyProject class:

Tartalmaz egy string típusú Nev mezőt, mely a project nevét fogja tárolni, egy double típusú Koltseg, mely a project árát tartalmazza, illetve egy szakterulet típusú tömb Szekteruletek néven, mely a projecthez szükséges szakterületeket tartalamazza, valamint egy konstruktor, amely beállítja a korábban felsoroltak értékeit.

Projectek class:

Egy EgyProject elemeket tartalmazó listát tartalmaz, mely egyben tárolja a beolvasott projecteket, valamint egy ProjectHozzaad metódus. mely beszúrja egymás után a projecteket a listába.

BinarisFa.cs:

FaElem class:  
Generikus osztály, ahol a T-re van egy olyan megkötés, melyre az a megkötés igaz, hogy IBeszallito interface típusúnak kell lennie. Van egy T típusú tartalom, melyben az IBeszallito interface-nek megfelelő objektumokat tárolja, van egy int(szám) típusú kulcs változója, mely a T tartalomból kinyert Nev alapján tárol egy számot melyek kulcsként alkalmaz, mivel nem lehet 2 ugyan olyan nevű cég. Tartalmaz egy Kulcs metódust is, melyet a konstruktorban használunk/hívunk meg a kulcs változó beállítására. Működése során a T tartalomból kinyert Nev változóban tárolt érték minden betűjét int-é kasztoljuk, és hozzáadjuk a kulcs változóhoz.

BinarisFa class:

Generikus osztály, ahol a T-re van egy megkötésünk, hogy csak IBeszallito interface típusú lehet. Van egy publikus Beszuras metódusa, amely meghív egy private \_Beszuras metódust, mely a tanultak alapján felépíti a bináris fát, elindul a gyökértől melyet referenciaként adunk át. Ha az aktuális kulcs nagyobb, mint a beillesztendő, rekurzív módon meghívjuk a \_Beszuras metódust, csak az aktuális elem jobbra mutató elemét adjuk át referenciaként, ha az aktuális kulcs kisebb mint a beszúrandó, akkor referenciaként (ugyancsak rekurzív módon) az aktuális bal oldalát hívjuk meg. Ha az aktuális elem null, akkor beillesztjük a beillesztendő elemet. Van egy private IBeszallito típusú Kereses metódus, melyet a public Palyazo cegekbe hívunk meg. A Kereses a Felhasználótól bekért cég nevét keresi ki a fából. A PalyazoCegek metódus a Kereses metódus segítségével kigyüjti egy IBeszalltio típusokat tartalmazó listába a cégeket, akik pályázni fognak a projectekre. Itt is megtalálható egy Kulcs metódus, funkciója ugyan az mint korábban, a Kereses metódusnak segít. Az osztály végén található 7db lista, melyek az IBeszallito-t megvalósító osztályok típusait tartalmazzák, az alattuk lévő Csoportositas metódus ezekbe helyezi bele a fából kikeresett pályázó cégeket aszerint, hogy melyik cég mihez ért, és ezáltal milyen feladatokat tud megvalósítani.A PalyazoCegek metódus addig fut, míg nem kapja meg a ”mehet” szót, ez állítja le a while ciklust, mely folyamatosan kéri be a cégek neveit, akik pályázni fognak a projectekre, és ezeket a cégeket hozzáadja egy IBeszallito elemeket tároló palyazoCegek nevű listához.A PalyazoCegekUjra metódus visszaadja a palyazoCegek nevű listát. A public TorlendoCegek metódus addig fut, míg nem kapja meg a ”mehet” szót, ez állítja le a while ciklust, mely folyamatosan kéri be a cégek neveit, akiket megakarunk szüntetni. Kitörli a palyazoCegek listábó, és a referenciaként átvett osszesbeszallito listából is. A private torlendoCegKereses a törlendő cégeket keresi ki név szerint a bináris keresőfából, a TorlendoCegek metódusban van meghívva.

Kivetelek.cs:

Az eldobandó kivételeket tartalmazza, melyeket a szükséges helyeken dobunk el a throw paranccsal. Az egyes kivételeket leszármaztatjuk az Exception beépítettből, a kontruktorában megvan adva egy üzenet melyet kitudunk majd iratni.

ProjectKioszto.cs

Mintaosztályokat megvalósító tömböket tartalmaz, melyekben a pályázó cégeket fogjuk tárolni a ”típusaik” szerint, valamint egy EgyProjectTomb mely EgyProject elemet tartalmaz, az összes project megtalálható benne. A konstruktor a példányosításkor megkapott listákból az elemeket a megfelelő tömbbe teszi, illetve beállítjuk az r fűrészfogas tömbünket, mely IBeszallitokat tartalmaz (mivel ez tömb a tömbbent jelent, ezért átadjuk neki a korábban említett mintaosztály tömbjeinket), az e mátrixunkat, melynek annyi sora lesz ahány projectünk van, és 2 oszlopa, az első oszlopban a project nevét fogjuk tárolni, a 2. oszlopban a projectet megnyerő cég nevét. A public BTS metódus meghívja a private \_BTS metódust, mely elvégzi visszalépéses keresést a fűrészfogas tömb segítségével. Addig megy, amíg a van logikai változó hamis (ez csak akkor lesz igaz ha az utolsó projectre is találtunk céget), illetve amíg a notfound logikai változónk hamis(a visszalépés részéhez kell). Ezen belül a szint változó a minta osztályokat reprezentálják, 1 szinten csak 1 ”fajta” cégek vannak, és ezek között lékedünk az i változóval (ha több cég is van egy cégtípuson belül, akkor kell). Ezután megnézzük, hogy az r tömbben a szintedig féle cégen belül az 1. cég megfelelő szakterületekkel rendelkezik e, ha nem akkor megnöveljük a szintet (ha az első cég nem lesz jó, akkor a többi sem, mivel ugyan abba a ”cégcsoportba” tartoznak), ha megfelelő megnézzük a ”cégtípusokon” belül az egyes cégeket, hogy belefér e az adott cégnek a visszaosszeg változójába a MegvesztegetesiAr metódusa által visszaadott összeg, ha igen beljebb lépünk, ha nem, megnézzük hogy van e még hátra ilyen ”típusú” cég, ha van megnézzük a következőt, ha nem, megnöveljük a szintet. ha mindkettő feltétel tejesült és beljebb léptünk, az eredményeket tároló e mátrixunkba beszúrjuk a project nevét, és a cég nevét is, majd hozzáadjuk a cég visszaosszeg-ének a MegvesztegetesiAr metódusából visszatérő összeget. Ezután megnézzük hogy elértünk e már a j változónkkal a projectek végére, ha igen akkor a van változónkat igazra állítjuk, mely lefogja állítani a ciklust(a belső meghívásokból való kilépéseknél is leállítja mivel referenciaként vesszük át). Ha nem értünk a végére meghívjuk rekurzív módon a \_BTS metódust, előre léptetjük 1-el a j változónkat, hogy a következő projectre keressen munkát, és 0ról indítjuk a szintet, hogy miden lehetőséget megvizsgálhasson. Ezek után a kivonjuk a cég visszaosszeg-éből a MegvesztegetesiAr metódus visszatérési értékét, illetve a notfound-ot hamisra állítjuk, ezekre a visszalépés miatt van szükség, a kivonás azért, hogy ha visszakell lépnünk, ne maradjon benne véletlenül sem a visszaosszeg-ben annak a projectnek az értéke, amelyet (lehjet) mégsem neki osztunk ki, ha befejeződik a \_BTS teljesen, akkor pedig nem fog problémát okozni, mivel minden projectre lesz cég akkorra. a notfound-ot azért állítjuk false-ra, mert ha visszakell lépnünk, a van változónkat nem állíthatjuk igazra, mivel az másért felelős, ezért a notfound foja leállítani hogy kiljebb lépjen a \_BTS, és mikor ez megtörténik és hamisra állítjuk, a ciklus folytatódhat, HA a van továbbra is hamis, magyarul keresunk egy másik céget az adott projectre.  
A private JoASzakterulet metódus a \_BTS-ben segít, ez vizsgálja meg hogy az adott projecthez a cégnek van e megfelelő szakterülete. A private Fk metódus a \_BTS-ben segít, ez vizsgálja meg hogy az adott cégnek belefér e még a visszaosszeg változójába a MegvesztegetesiAr visszatérési értéke. a private KiosztottMunkakKiirasa metódus az e eredmény mátrixunkat írja ki a console-ra.

Program.cs

Itt található projectek és a beszállítók beolvasása(region-be szedtem őket). A projectek beolvasásánál létrehozok egy FileStream-et (fs), illetve egy StreamReader-t (sr). Ezek segítségével olvasom be a bin\Debug mappában lévő txt-t/txt-ket. A fájlból soronként olvasom ki a tartalmat, majd split-elem ’/’ jelek szerint. 1 szegmensben lesz a project neve, melyet eltárolok egy változóban, a 2. szegmensben lesz a költsége a projectnek, a végén ’M’ betűvel (ez jelzi a beolvasásnak hogy eddig tartanak a számjegyek, eddig kell eltárolni), a 3. szegmensben lesz a projecthez szükséges szakterületek felsorolása ’,’-vel elválasztva, e szerint spliteljük és tároljuk el, és alakítjuk át a szakterületek enum formájára. Ezután meghívjuk a Projectek.cs ProjectHozzaadaasa metódusát, mely eltárolja nekünk az adott projectet, ezután beolvassuk a fájl következő sorát, és az egész folyamat kezdődik előről, míg a fálj végére nem érünk. Ezt követően a beszállítókat olvassuk be hasonló módon, felül írjuk a FileStream-et és a StreamReadert, hogy alkamas legyen a beszallitok.txt fájl olvasására. Szintén soronként olvassuk be a fájlból, ’/’ jelek mentén split-eljük. 1. szegmens lesz a cég neve, 2. szegmens a cég szakterületeinek felsorolása ’,’-vel elválasztva. a nevet eltároljuk egy változóba, a szakterületeke egy tömbben, majd átalakítjuk az enum megfelelőire, illetve hozzáadjuk az osszesBeszallito nevű IBeszallito-kat tartalmazó listánkhoz (innen egyszerűbb kitörölni egy céget és úgy létrehozni újra a fát hogy az már nincs benne), majd beolvasunk egy újabb sort, és kezdődik előről egészen addig amíg elnem érünk a fájl végére. Ezután egy ciklussal végig megyünk az osszesBeszallito listán, magy egyenként megvizsgálva a szakterületek alapjána megfelelő ”típusú” cég formájában beszúrjuk a bináris keresőfába. Ezután, mivel megvannak a projectek, és a cégek is (bináris keresőfában is), meghívjuk a BinárisFa.cs PalyazoCegek metódusát mely a korábban említettek szerint működik. Ennek a metódusnak a visszatérési értékét átadjuk egy listának (mivel lista a visszatérési értéke), majd pedig ennek a listának az elemszámának alapján meghívjuk a BinarsFa.cs Csoportositas metódusát, mely ezeket a cégeket (amelyek pályázni fognak a projectekre) a mintaosztály szerinti listákba rendezi (mondhatni szakterületek szerint). Ezt követően lepéldányosítjuk a ProjectKioszto.cs-t, és meghívjuk a BTS metódusát, mely a korábban említettek szerint működik (meghívja a \_BTS-t). Majd ha kiirja a BTS által előállított e eredmény mátrix tartalmát, meghívjuk a BinarisFa.cs TorlendoCegek metódusát, mely a korábban leírtak szerint működik. A pályázó cégeket tároló listánkat felül írjuk egy új listával mely már nem tartalmazza azokat a cégeket, amelyek pályáztak, azonban megszűntek. Létrehozunk egy új BinarisFa osztályt, melyet feltöltünk a (esetleg a megszűnés miatt módosított, mivel referenciaként használtuk) osszesBeszalltio listából a korábbiakhoz hasonló módon (csoportosítjuk és beszúrjuk az ”új” fába). Végül újra lepéldányosítjuk a ProjectKioszto.cs-t, átadjuk neki az új BinárisFa-ban tárolt listákat, illetve a projectek listáját (korábban is, csak ott az 1. fa dolgait adtuk át), és ezekkel az ”újabb” adatokkal meghívjuk az újra példányosított ProjectKioszto.cs BTS metódusát, mely a megszűnt cégek nélkül végzi el a visszalépéses keresést a már korábban említett módszerrel. Egy Console.ReadLine() áll a végén mely azért kell, hogy szemrevételezhető legyen a BTS eredménye, hogy melyik projectet ki nyerte.