ListNotSorted osztály (*A ListSorted lista rendezett beszúrásra szolgál, de ezt végül nem használtam sehol*).

Elkészítettem benne egy egyszerű beszúrás metódust (Insert), ami mindig a lista elejére szúr be elemet.  
A törlés metódusomat (Delete) úgy módosítottam, hogy a törölni kívánt elemet a kiláncolás közben vissza is adja, hogy a vásárlás során azt egyből be tudjam rakni a vásárló Terméklistájába.

A BejarasString metódust arra használom, hogy egy *Elado* *Termeklistáját* bejárja és összegyüjtse egy stringben a termékeket, amik a *Product* osztályban levő *ToString* metódusban leírtak szerint fognak megjelenni majd a konzolon.

A ToArray metódust, ahogy a neve is jelzi, arra használom, hogy azt a listát amire meghívóm átalakítsa tömbbé. Először bejárja a listát, összeszámolja mennyi elemből áll, létrehozza a tömböt, majd beszúrja a listaelemeket a tömbbe.

Az *AruhazKezelo* osztályban hasznosítom, az *AjandekozasARendszerben* metódusnál. Így oldottam meg, hogy egyszerűen lehessen rendezni a vevők listáját, az alapján, hogy kinek mennyi megvásárolt terméke van.

A Print metódus a listaelemek kiírására szolgál.

A Pop metódus visszaadja a fej elemet a listából és törli is azt. Azért törli, mert ezt a metódust az ajándékozásnál használom, ahol az a cél, hogy azt a terméket, amit már egyszer kiosztott a program, ne tudja mégegyszer.

A Foreachmetódus lényegében azt csinálja, mint egy foreach ciklus, végigiterál a listán. Az enyémben az a különbség, hogy ez egy metódust vár el paraméternek, amit meg is hív minden egyes elemre iterálás során. A paraméter metódusnak T tipusúnak kell lenni, visszatérés nélkül, tehát ez egy Action<T1,T2> típusú delegált lesz.

AFindmetódus azt csinálja, hogy végigmegy a listán és mindegyik elemre meghívja a paraméternek beadott metódust, aminek bemenete T, visszatérési értéke bool típusu, tehát ez egy Func<T,TResult> típusú delegált lesz. Ha a metódus igazzal tér vissza, akkor visszaadja annak a listaelemnek az értékét, tehát megkeres egy értéket a listában.

Itt szeretném leírni azt, hogy mi az a lambda függvény. A lambda egy névtelen metódus lényegében, ami vagy egy kifejezést, vagy állítások sorozatát fogadja. Egyszerűbb használni, minthogy egy teljesen új metódust kelljen írni. Használatkor be kell adni neki, hogy milyen paraméterekkel működjön, ami később meghatározza, hogy milyen delegált típussá lehet átkonvertálni.

A *BejarasInt*, *Bejaras* és a *First* metódusok egy kezdetleges megoldáshoz voltak használva, a végleges programban nem csinálnak semmit. A *BejarasInt* bejárta a listát és összeszámolta hány eleme van a listának. A *Bejaras* egy szimpla lista bejárás. A *First* visszaadta az első listaelemet.

Szintén egy korábbi próbálkozáshoz használtam az *IEnumerable* és *IEnumerator* interfészeket, amiket megvalósítva lehetőség lett volna *foreach* használatára a lista bejárásához. Végül saját bejárást írtam, mert *NullReferenceException*t dobott a *foreach* használata közben.

# Product osztály

Megvalósítha az *IComparable* interfészt. Tartalmazza a szükséges property-ket és a konstruktort. A *CompareTo* metódussal két terméket lehet összehasonlítani a nevük alapján, így rendezni lehet a termékeket. Ha a visszaadott érték 0, akkor a két név megegyezik, ha kisebb mint 0 akkor az *obj* megelőzi az adott terméket a sorban, ha nagyobb mint 0, akkor pedig követi a sorban.

# Elado osztály

Megvalósítja az *ITranzakciobanResztvevo* interfészt. Tartalmazza a szükséges property-ket és a konstruktort, valamint itt található az elvárt osztályhierarchia. Az *Elado* osztály az ősosztály.

# Vevo osztály

Megvalósítja az *ITranzakciobanResztvevo* interfészt. Tartalmazza a szükséges property-ket és a konstruktort, valamint itt is megtalálható az osztályhierarchia. A *Vevo* osztály az ősosztály. Ez az osztály ki van egészítve egy *TermekekDB* property-vel, ami arra szolgál, hogy számolja, hogy egy példánynak mennyi terméke van.

# AruhazKezelo osztály

A VevoInsert és az EladoInsert vevők, illetve eladók regisztrálására szolgál konzolos felületről. A fentiekben leírt *Find* metódust hasznosítja, amivel leellenőrzi, hogy van-e már a listában megegyező adószámmal rendelkező elem. Ha van, akkor egy *VanIlyenElado* vagy *VanIlyenVevo* kivételt dob, ha nincs, akkor beszúrja az uj elemet a listába. Ugyan ezeket a metódusokat használja a program a fájlból történő feltöltésnél is.

A TermekRegisztraciometódus a termékek regisztrálását teszi lehetővé, fájlból történő feltöltés során. Felhasználja a *Find* metódust a duplikációk kiszűrésére.

Az EladoKereseseAdoSzamAlapjan*,* VevoKereseseAdoSzamAlapjan és TermekKereseseTermeknevAlapjanmetódusok mind ugyan azt a célt szolgálják. A *Find* segítségével visszaadják a keresett eladót, vevőt vagy terméket. Ezeket a vásárlás lebonyolításánál használja a program.

A Vasarlas metódus úgy működik, hogy amikor a felhasználó vevő és eladó esetében az adószámot, termék esetében a nevét beírja a konzolra, a program megkeresi azt és elmenti egy változóban. Ezek azért szükségesek, hogy tudjuk, hogy melyik vevővel akarunk vásárolni, melyik eladótól akarunk vásárolni és ennek az eladónak milyen termékei vannak. Ezek lesznek a *Vasarlas* paraméterei. Törlés közben visszaadja a megvásárolni kívánt terméket, majd azt beilleszti a vásárló Terméklistájába. Növeli a vásárló *TermekekDB* és *Ertekeles* tulajdonságait, amik az ajándékozáshoz kellenek majd.

Az EladoTermekek metódus arra szolgál, hogy létrehozzon egy új listát, amiben termék/a termék eladója szerepel. Minden eladóra lefut egy olyan függvény, ami végigmegy az eladó terméklistáján és a termék, eladó párost kigyűji egy másik listába. Ezzel elérve azt, hogy egy listában szerepeljen az összes eladó termék és, hogy ezeknek ki az eladója.

Az AdunkNekiEnnyiTermeket metódus végzi az ajándékozást. Utolsó paramétere egy olyan lista amiben az összes termék benne van és mellettük, hogy ki az eladója. Azért kell az *eladoTermekek* lista, mert így az összes eladó termék egy helyen van és csak ki kell vennem a *Pop* metódussal az első terméket, amit a mar meglévő *Vasarlas* metódusommal ajándékozok el. Ezt annyiszor teszi meg, ahány ajándékot kell kiosztani, azaz amekkora az értéke az *ajandekMennyisege* változónak.

Az AjandekozasARendszerben metódus használja a fenti két metódust az ajándékozási rendszer létrehozásához. Itt hasznosítom a *ToArray*-t arra, hogy tömbbé alakítsam a vevők listát és utána rendezni tudjam azt. A Sort második paramétere egy lambda függvény, amivel megadom, hogy mi alapján hasonlítsa össze a tömb elemeit, jelen esetben a vevők *TermekekDB* tulajdonságát. Ezután a cilus arra szolgál, hogy egy vevő értékelése alapján adjon neki ajándékot. Abban az esetben, ha nem sikerült kiosztani minden ajándékot, a program eldobja az *ElemException* kivételt.

# Program osztály

A menürendszert (MainMenu) valósítja meg elsősorban, amit egy switch szerkezettel hoztam létre. A *Main*-ben legelőször a fájlból történő adatbetöltés fut le, ha léteznek a fájlok.

Az EladokFeltolteseFajlbol*,* VevokFeltolteseFajlbol és a TermekekFeltolteseFajlbolmetódusok végzik a fájlból olvasást. Olvasás közben mindegyik meghívja a megfelelő konstruktort az adat eltárolására.