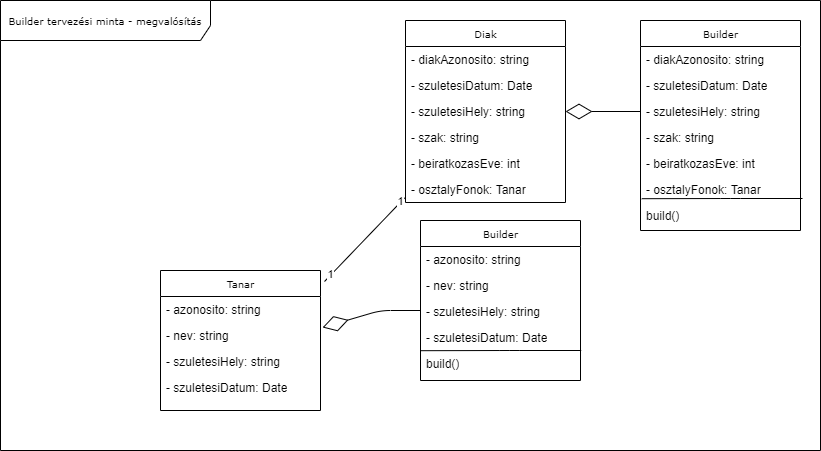
**A builder pattern**

Az én választásom a design patternek közül a Builder vagyis az építő tervezési mintára esett. Ez a tervezési minta az objektumorientáltságban az egyes objektumok előállítására szolgál, melyet akkor érdemes használni, amikor egy adott objektumnak sok attribútuma van, aminek az egyes értékeit a program futása során adunk meg, ez által felépítve az adott objektumunkat.

A builder pattern lényegében akkor adja vissza a kész objektumot, amikor már minden – az adott objektumhoz szükséges - paraméterünk rendelkezésre áll, és meghívjuk a kreálásra szolgáló függvényünket (ez általában a build()-nek nevezzük el)



A megvalósításban diákokat, valamint tanárokat is tudunk készíteni Builder segítségével. A Diák, valamint a tanár között egy asszociációs kapcsolat megfigyelhető, hiszen egy diáknak meg kell adni az osztályfőnökét, mely jelen esetben egy Tanár objektum lesz.

Mind a Tanár, mind a Diák osztálynak van egy Builder belső statikus osztálya, mely aggregált kapcsolatban vannak az előbb említett osztályokkal. Ezek az osztályok teszik lehetővé azt, hogy elkészítsék számunkra a Tanár, illetve a Diák objektumokat.

Az alap osztályokban (tehát a Tanár és Diák osztály) konstruktorai private metódusként vannak implementálva. Ezzel megkerüljük azt, hogy a felhasználó a Builder osztály megkerülésével hozza létre ezeket az objektumokat.

A Main osztály futtatásával jelenleg egy Géza nevű tanárt, és két diákot hozunk létre, akiknek osztályfőnökük Géza lesz. Létrehozunk egy harmadik diákot is, mely *UjDiak* változónevet kapta. Itt a program ennek az UjDiak-nak a nevét és szakját fogja bekérni konzolon keresztül, viszont nem minden adatot adtunk meg (pl. születési hely, és idő), mint az előző objektumoknál.

Fontos megjegyezni, hogy fennáll annak a lehetősége, hogy a felhasználó a Builder használata során nem minden paramétert tölt ki, ezért elég fontos, hogy a szükséges paraméterek esetében különböző validációkat helyezzünk el az objektum létrehozása során.

Jelenleg például az objektum toString() metódus felülírása során használok olyan validációt, mely esetében ha az objektum egy adott mezője nincs kitöltve, abban az esetben nem is fogjuk kiíratni.

Úgy gondolom, hogy ez a létrehozási minta jelentősen hatékonyabb létrehozást tesz lehetővé, amikor nem feltétlen áll rendelkezésünkre a létrehozás pillanatában minden információ, valamint nincs szükség az adott osztályhoz többfajta paraméteres konstruktor létrehozására, hiszen a Builder segítségével az adott objektumnak, csak azon paramétereit adjuk meg, amiket meg szeretnénk adni.