Órai feladat – Gráfok I.

Készítse el az alábbi feladat megoldásához szolgáló, gráf adatszerkezetet implementációját szomszédsági lista reprezentáció alapon (további gyakorlásként javasoljuk csúcsmátrix alapon is elkészíteni).

Mivel a leadott feladatokat lehetséges, hogy automatizált módon fogjuk ellenőrizni, ezért kérünk mindenkit, hogy a lenti (ékezetek nélküli) elnevezéseket tartsa meg. Szükség esetén további mezőket fel lehet venni, bár ezekre általában nincs szükség.
Ugyanígy kérünk mindenkit, hogy próbálja meg önállóan megoldani a feladatot, mivel csak így fog bármit tanulni belőle. Szükség esetén persze a laborvezetőket nyugodtan meg lehet keresni, akik segíteni fognak.

Osztályleírások

Az egyes osztályokból a `Console` osztály metódusait meghívni **tilos**, azt csak a `Program` osztályon belül tegye!

class Graph

Az adatszerkezethez hozza létre a szükséges osztályt a tanult módon és biztosítsa a következőket:

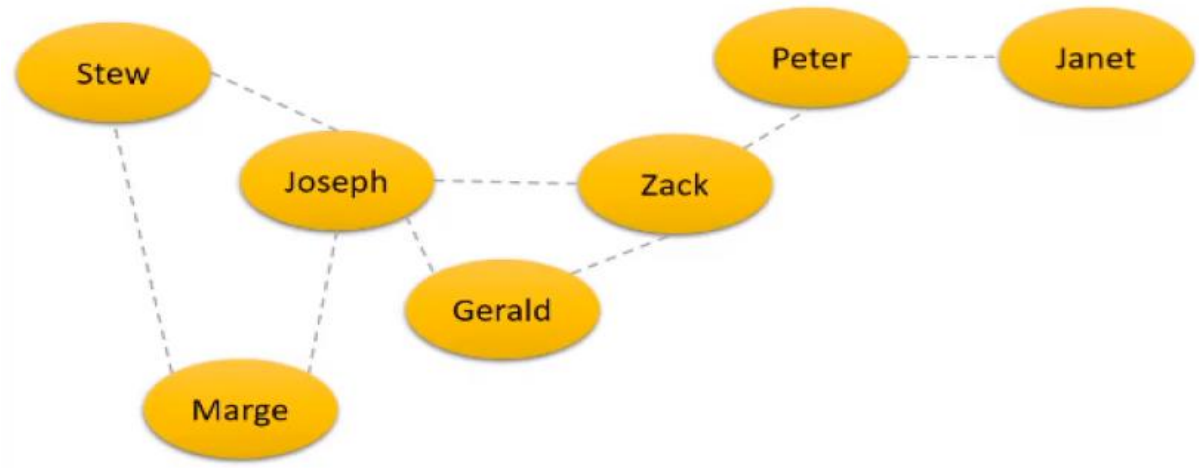
1. Legyen generikus, bármely `T` típusú elem tárolására legyen lehetőség!
2. Legyen alkalmas irányítatlan, súlyozatlan gráfok reprezentálására!
3. Rendelkezzen az alábbi publikus metódusokkal:
   - void AddNode(T node)
     `T` típusú `node` tartalmú csúcs hozzáadása az adatszerkezethez
   - void AddEdge(T from, T to)
     `from` és `to` csúcspontok között vezető él hozzáadása az adatszerkezethez
   - bool HasEdge(T from, T to)
     Annak vizsgálata, hogy vezet-e él `from` és `to` csúcsok között
   - List<T> Neighbors(T node)
     `node` elemet tartalmazó csúcs szomszédai
4. Legyen egy külső metódus megadására lehetőség, amely bejáráskor adott elem feldolgozásának módját fogja megadni!
   - ezt a funkciót a tanultak alapján valósítsa meg **metódusreferencia segítségével** mint
     void ExternalProcessor(string item)
     (Megjegyzés: az egyszerűség kedvéért az elemeket stringként fogjuk feldolgozni)
   - Tesztelje a működést:
     - egyrészt a kívülről átadott `Console.WriteLine()` metódussal;
     - másrészt pedig hozzon létre egy saját metódust, amely szignatúrája megfelel a metódusreferenciában adottnak: e metódus segítségével az adott feldolgozandó elemet fűzze hozzá egy fájl tartalmához!
5. írja meg a szélességi (Breadth First Search, BFS) valamint mélységi (Depth First Search, DFS) bejárásokat mint
   - void BFS(…)
   - void DFS(…)

   A metódusok bemeneti paramétereinek megválasztása a feladat része!

class Person

A személyeket reprezentáló objektumokhoz hozza létre az entitás osztályt `Person` néven, amelyben egy `Name` írható/olvasható tulajdonságot helyezzen el. Az osztályban írja felül a `ToString()` metódust, hogy a metódusreferenciát megfelelő módon a név megjelenítésére tudja használni.

Miután az adatszerkezet elkészült, az alábbi ábrán látható személyek közötti kapcsolati hálót kell elkészíteni, a megfelelő csúcsok és élek beállításával:



Tesztelésként dolgozza fel a gráf elemeit szélességi és mélységi bejárás szerint egy tetszőleges csúcsból kiindulva!

1. Egészítse ki a `Graph` osztályt egy eseménnyel (mely alapját szolgáltassa a void GraphEventHandler<T>(object source, GraphEventArgs<T> geargs) delegált), amely legyen elsütve egy él felvitelét követően.
   Az eseményben legyen benne, hogy melyik *A* és *B* csúcs között sikerült az élt felvenni, ennek megfelelően definiálja a GraphEventArgs osztályt!
   Mind az esemény argumentum osztály, mind a delegált és a hozzá tartozó esemény **generikus legyen** a T típusra nézve!
   A `Program` osztályban hozzon létre egy metódust, melyet iratkoztasson fel erre az eseményre!

2. Egészítse ki a szélességi bejárás algoritmusát: készítsen egy olyan algoritmust amely segítségével meg tudjuk mondani, hogy adott *X* személy hányad fokú ismeretségben áll adott *Y* személlyel.
   Az ismeretség fokát a két személy közötti ismerősök száma jelenti, egészen pontosan a rajtuk keresztül megtett legrövidebb út hossza (élek száma). Például az ábrán bemutatott minta esetén: Janet harmadfokú ismeretségben áll Gerald-dal, mert Peter-en majd Zack-en keresztül van kapcsolat közöttük (1. Janet → Peter; 2. Peter → Zack; 3. Zack → Gerald).
   Tesztelésképpen kiírásra kerülhet az ismeretség fokának alapjául számolt X és Y közötti út is.

A feladat megoldása során elfogadott és használható a beépített `List` valamit `Queue` osztály!