A feladat egy iskolai nyilvántartás egy részének a megvalósítása. A program architektúrája a Model-Repository-Service-Controller-View modellt követi. A model rétegben találhatók az adatleíró osztályok. A repository rétegben találja a tárolt adatokat. A service réteg feladata az üzleti logika megvalósítása. A controller réteg feladata a view rétegből érkező adatok ellenőrzése, illetve a megjelenítendő adatok megfelelő adattípusba való konvertálása. A feladat megoldása C# programozási nyelven készül, és Visual Studio IDE keretrendszer használatával fejleszthető. A feladat bizonyos részei már részben vagy teljesen ki vannak fejlesztve, más részek pedig hiányoznak. A már elkészült forrásállományok mappában vannak elhelyezve, amelyek elnevezései követik a program architektúrális felépítését.
A feladat megoldása során objektum orientált programozási módszer használjon. Az osztályok, azok tulajdonságainak és metódusainak megírása során használja a tiszta kód programozási elvet. A forráskód több helyen tartalmaz kódban írt dokumentációt. Ezek vagy segítenek megérteni a programkódot, kijelöli a feladatot, és útmutatást nyújtanak megírandó programkódhoz. Lehetséges olyan feladat, hogy a metódusok dokumentációját kell megírni.
Talál a projektben egy olyan mappát, amely UML diagramokat tartalmaz. Az itt található diagramok segítenek jobban megérteni az osztályokat, és azok közötti összefüggéseket. Lehetséges olyan feladat is, hogy el kell készíteni egy UML diagramot.
A repository réteg megértése fontos a feladat megoldása szempontjából. A program egy iskola adatait tartja nyilván. Az iskolában lehet több osztály. Egy osztályban meg több diák. Ezek az osztályok megtalálhatók a modell és repository rétegben a tesztadatokkal együtt. Tanulmányozza őket és az UML diagramokat a megoldás előtt!
A következő feladatok kijelölik azt az utat, amely a szoftver kifejlesztése érdekében szükséges. Haladjon lehetőleg sorrendben a következő feladatok szerint. Ha valamelyik feladatot nem tudja megoldani, áttérhet a következő feladatra. Értékelve a megírt kódok lesznek, nem a program működő vagy nem működő funkciói. Természesen, ha a kód jó, a program funkciója tesztelhetővé válik.

1. feladat

A munkát a modell rétegben található osztállyal kezdjük. Az első feladat az osztály mezőit frissítő metódus megírása (a metódus fejléce már megtalálható a forráskódban). A második feladat a ToString osztály felülírása (a teljes metódust el kell készíteni), hogy az minden mező megjelenítését biztosítsa. A metódust a hibaüzenetek megjelenítésére használjuk majd.

2. feladat

A feladatot a repository réteg osztályainak további fejlesztéseivel folytassa! Az első feladat az iskolaosztályt kezelő osztály megírása. A forráskódot az Osztaly.cs állományban találja.

- Először a konstruktort készítse el. Ebben az osztály tulajdonságait (mezőit) kell megadni a paraméter lapján, majd a diákok listáját példányosítani.
- Ezután az IOsztaly interfaceben lévő metódusok megírása a feladat. Módosítsa vagy írja meg a hiányzó metódusok kódját!
- Ezután kódolja le, hogy az osztály, az IOsztalyMuveletek interfacetől is öröklődik, és írja meg az interface által meghatározott metódusokat. A metódusok megírásában segít a kódban lévő dokumentáció!

3. feladat

Továbbra is a repository rétegben dolgozzon! A feladat az osztályban járó diákokat kezelő metódusok megírása! A kódok egy parciális osztályban találhatók. Rendelje az IOsztalyDiakokKezelese interfact az Osztaly osztályhoz, majd írja meg vagy módosítsa a szükséges metódusok kódjait! Kivételdobásra készítse el és használja az OsztalyException kivétel osztályt!

- A torolDiak és frissitDiak metódusokban kivételdobást készítsen, ha az adott adatokkal jelölt diák nem található!
- A hozzadDiak metódusban kivételdobást készítsen, ha az adott diák már jár az osztályba (minden adata megegyező egy már osztályba járó diákkal).

4. feladat

A szoftver fejlesztését folytassa a repository rétegben, de már az iskolát kezelő metódusokkal az Iskola.cs állományban.

- Keresse meg a „torol” metódust. Ez a metódus törli az osztályt. Az osztályba még járhatnak diákok. Őket az iskola törlése előtt törölje. Erre már írt metódust a diákok kezelése során. Ennek a metódusnak a megfelelő helyen történő hívásával egészítse ki a „torol” metódust.
- Keresse meg a „modosit” metódust. A metódus „bug”-os, nem végzi el feladatát. Javítsa a helyes működés érdekében!
- Keresse meg a „vanEOsztaly” metódust. Ez a metódus nincs megírva. Írja meg! A metódusok megírásában segít a kódban lévő dokumentáció!

5. feladat

Folytassuk a feladatot az iskola kezelő metódusok írásával, de már azokkal, amelyek az iskolába járó diákokat kezelik. A kódok egy parciális osztályban találhatók az IskolaDiakKezeles.cs fájlban.

- Írja meg a következő metódusokat:
    - hozzadDiakotOsztalyhoz
    - modositDiakotOsztalyban
    - torolDiakotOsztalybol

  A metódusok megírásánál dobjon a dokumentációban található kivételeket!
- A forráskódban több metódusnak nem lett megírva az elektronikus dokumentációja. Írja ezeket meg, az előző mintáknak megfelelően. Térjen ki a leírásra, a paraméterek leírására, a visszatérő értékre és a dobott kivételekre! Ügyeljen az algoritmus pontos leírására, mert pontozva lesz!

6. feladat

A validation rétegben megtalálható a „NevEllenorzo” osztály mellett az „AzonositoEllenorzo” osztály is. Feladata az ebben az osztályban lévő három még nem megírt metódus kifejlesztése a Test-driven development (TDD) módszerrel. A tesztet egy projektben találja meg!

7. feladat

A modell rétegben található Diak osztály konstruktorába kösse be a NevEllenorzo osztályt. Egy diák, csak akkor jöjjön létre, ha a neve megfelel, amit a NevEllenorzo osztállyal tud ellenőrizni! Kapja el a NevEllenorzo osztály összes kivételét és a hibákat loggolja az Otuputra (a programozási nyelnek van erre beépített osztálya és metódusa).

8. feladat

A következő feladat minden réteget érinti. A tesztadatok betöltése után észrevehető, hogy az iskolaátlag kiszámítás és az osztály átlag kiszámítása nem működik. Fejlessze tovább az alkalmazást úgy, hogy ezek a funkciók is működjenek!

9. feladat

A szolgáltatás rétegben a feladata az „athelyezOsztalyba” metódus megírása! A metódust az „IskolaSzolgaltatasDiakokKezelese.cs” állományban találja. A feladat megoldása három egyszerű feladatból áll, amelyet az alsóbb rétegek segítségével tud megvalósítani:

- keresse meg az metódus paraméterei alapján az áthelyezendő diákot
- törölje abból az osztályból ahová járt
- helyezze át abba az osztályba, ahova járni fog

10. feladat

A munkát a controller rétegben folytatjuk. Az „IskolaVezerloDiakKezeles” osztályba kifejlesztendő metódus a „modositDiakotOsztalyba”. Ehhez tanulmányozni kell, a már megírt „hozzaadDiakotOsztalyhoz” metódust. Az ebben lévő kódok megértése és felhasználása alapján dolgozzon!

11. feladat

A view rétegben a feladat a „buttonHozzadOsztalyt_Click” metódus megírása! A forráskódot a „Forma1FormOsztalyokKezelese.cs” állományban találja!

12. feladat

A munkája végén dokumentálja a repository rétegben található osztályokat UML diagram segítségével. A diagramon az „Iskola”, „Osztaly” és „Diak” osztályok és kapcsolataik szerepeljenek! A diagramot a „systemdesign” mappába helyezze el!

13. feladat (szorgalmi, ha van ideje)

Ha a feladat megoldásával végzett, akkor érdekes feladat lehet a bug-ok megfigyelése és a kód javítása. Megfigyelhető, ha a teszt adatokat sikeresen betöltött nem lehetséges a diák átlagának módosítása! Javítsa ezt a hibát ha van rá idő!