5. heti feladatok

1. órai feladat Készítsen egy olyan alkalmazást, amivel szimulálni tudják élelmiszerek ásott veremben történő tárolását.

- Hozzon létre egy **FoodIngredient** osztályt.
  - Az osztályban tárolja el egy élelmiszer fajta nevét, tárolt mennyiségét és a mennyiség egységét.
  - Mennyiségi egységként liter, kilogramm, darab és csomag legyen megadható.
  - Legyen az osztálynak egy felülírt `string ToString()` metódusa.
  - Tesztelje az osztály `string ToString()` metódusát.
- Kivételek kezelése érdekében hozzon létre három kivétel osztályt.
  - A **StackException** osztályban tárolja el, hogy melyik **IngredientStack** objektum dobta el a kivételt.
  - A **StackEmptyException** osztály legyen a **StackException** osztály leszármazottja.
  - A **StackFullException** osztály legyen a **StackException** osztály leszármazottja. Tárolja el, hogy melyik az az élelmiszer, amit nem sikerült betenni a verembe.
- Készítsen egy **IngredientStack** osztályt a verem reprezentálására.
  - A veremben lévő élelmiszereket tárolja el egy tömbben, aminek a méretét a konstruktorban állítsa be. Új elemeket mindig a tömb elejétől nézve első üres helyre szúrjon be, amikor pedig kivesz valamit a veremből, a kivett elem a tömbben eltárolt utolsó elem legyen.
  - Legyen az osztálynak egy publikus `bool Empty()` metódusa, ami információt szolgáltat arról, hogy a verem üres-e.
  - Legyen az osztálynak egy publikus `void Push(FoodIngredient newItem)` metódusa, ami betesz egy új elemet a verem tetejére. Ha a verem már tele volt, akkor sikertelen a verembe helyezés, ezért dobjon el egy **StackFullException** kivételt.
  - Legyen az osztálynak egy publikus `FoodIngredient Pop()` metódusa, ami kiveszi a legfelül lévő elemet a veremből. Ha a verem üres volt, akkor dobjon el egy **StackEmptyException** kivételt.
  - Legyen az osztálynak egy publikus `FoodIngredient Top()` metódusa, ami megadja, hogy mi van a verem tetején, de azt nem veszi ki onnan. Ha üres a verem, akkor `null` értékkel térjen vissza.
  - Tesztelje teljes körűen a publikus metódusokat.
- Készítsen egy **IngredientStackHandler** osztályt, ami egy példányosításkor megadott verem kezelését valósítja meg.
  - Legyen az osztálynak egy publikus `FoodIngredient[] AddItems(FoodIngredient[] foodIngredients)` metódusa. A metódus beteszi a verembe a paraméterként megkapott élelmiszerek közül azokat, amik befférnek a verembe. Visszatérési értéke tartalmazza azokat az élelmiszereket, amik már nem fértek be a verembe.
  - Tesztelje a metódus működését.
  - Próbálja meg úgy is tesztelni a metódus működését, hogy közben kimockolja a háttérben használt **IngredientStack** osztályt. Ennek érdekében, ha szükséges, hozzon létre megfelelő interfészt is.

2. órai feladat Sportóra által rögzített edzések adatait csv formátumú fájlban menti el a sportórához adott alkalmazás. Egy csv fájl tartalma, például az alábbi.

```
type,distance,time,elevation,heart_rate
Cycling,"35,8","1:28:03","314","142"
Running,"11,95","1:12:04","15","138"
Hiking,"24,3","4:47:02","687","116"
Swimming,"2,4","0:58:35","0","143"
```

Ahogy a példából is látszik négyféle sportág edzéseinek rögzítésére képes a sportóra. A számértékkel rendelkező adatok a csv állományban mindig idézőjelek között vannak. A távolság lebegőpontos, az emelkedés és a pulzus pedig egész számként van eltárolva. Az idő minden esetben `h:mm:ss` formátumú, ahol az óra akár egynél több számjegyet is tartalmazhat.

Írjon csv fájl feldolgozó alkalmazást, ami be tudja olvasni az egyes sorokban tárolt adatokat. Az esetleges hibás formátumú sorok esetén az alkalmazás írja ki a konzolra, hogy hiba történt, majd az adott sor figyelmen kívül hagyása mellett folytatódjon a fájl feldolgozása.

Egy lehetséges megvalósítási módhoz ötletek:

- Hozzon létre egy **Time** osztályt, amelyben óra, perc és másodperc adatok legyenek eltárolva. Írjon ehhez az osztályhoz egy statikus `Parse` metódust. Ha a parszolás közben azt tapasztalja, hogy nem megfelelő a bemenet formátuma, akkor dobjon el egy **TimeException** kivételt. Ha az óra érték negatív, akkor dobjon el egy **HourException** kivételt, ami a **TimeException** leszármazottja. Ha a perc vagy másodperc érték negatív vagy 59-nél, akkor is dobjon el megfelelő kivételeket.
- Hozzon létre egy **Workout** osztályt, amelyben el tudja tárolni egy edzés adatait. Legyen ennek az osztálynak is egy saját parszere, ami egy sorban lévő sztringet fel tud dolgozni. Ha közben valami problémát tapasztal dobjon el **WorkoutException** kivételt.
- Legyen egy **CSVProcessor** osztálya, aminek példányosításkor megadja a csv fájl elérési útját. Az osztály `Workout[] AllItems()` metódusa végzi a csv fájl feldolgozását, lekezeli az esetleg előálló kivételeket és visszaadja az értelmezhető sorokban lévő edzések tömbjét.

Tesztelje az egyes publikus metódusokat.

3. gyakorló feladat Készítsen el egy akasztófa játékot, ahol a nem elvárt inputokat kivételekkel kezelje le az alábbiak szerint.

- Hozzon létre egy **HangingTreeGameException** kivétel osztályt, mely egy szövegesen tárolt hibakódot tartalmazzon. Ez a hibakód az osztály konstruktorán keresztül legyen beállítható és későbbiekben ez az osztályon kívül csak olvasható formában legyen elérhető.
- Hozzon létre egy **BusinessLogicViolationException** kivétel osztályt, mely öröklődjön a **HangingTreeGameException** osztályból. Az ős hibakódja mellett egy felhasználónak szánt üzenetet is tartalmazzon. Ez az üzenet is (a hibakód mellett) az osztály konstruktorán keresztül legyen beállítható és későbbiekben ez az osztályon kívül csak olvasható formában legyen elérhető.
- A lehetséges szavakat a program indulásakor egy fájlból olvassa be a programban tárolt tömbbe.
  - Amennyiben a fájl nem található, akkor egy **FileNotFoundException**-t váltson ki, melyet a `System.IO` névtér alatt talál meg.
  - Ha a fájl létezik, de nem található benne egyetlen szó sem, akkor egy **NoPossibleWordsHangingTreeGameException** kivételt váltson ki. Ez a kivétel a **HangingTreeGameException** osztályból öröklődjön. A hibakód legyen minden esetben *HGT01*, amelyet a **NoPossibleWordsHangingTreeGameException** definiál.
- Válasszon ki egy szót (későbbiekben *keresett szó* ként hivatkozzuk) véletlenszerűen a beolvasott szavak tömbjéből.
- Üdvözölje a játék a felhasználót a konzolon, majd jelenítse meg, hogy hány karakterből álló szót kell kitalálnia.
- Hozzon létre egy akkora méretű karakterek tömbjét, amekkora a keresett szó hossza.
- Állítsa be a felhasználó életét 6-ra.
- Egy hátultesztelős ciklus segítségével hajtsa végre az alábbi műveleteket
  - A ciklus addig menjen, ameddig nem fogyott el a felhasználó élete (nem érte el a 0 értéket) és nem találta ki a keresett szót (nincs már „padló" (_) karakter a szóban).
  - Jelenítse meg a keresett szó már kitalált karaktereit a következő formában: a _ _ a
  - Egy `char GetValidChar()` metódus segítségével kérjen be egy karaktert a konzolról. (A metódus tényleges működését lásd később.)
  - Vizsgálja meg, hogy a bekért karaktert kipróbálta-e már korábban. Amennyiben igen, akkor váltson ki egy **BusinessLogicViolationException** *HGT21* hibakóddal és egy „Ezt a karaktert már korábban tippelte. Ez ebben a játékban nem megengedett, kezdje elölről egy másik játékkal" üzenettel.
  - Vizsgálja meg, hogy a bekért karakter szerepel-e a keresett szóban. Amennyiben igen, akkor minden egyes előfordulását vezesse fel a keresett szót reprezentáló karakter tömbben. Ellenkező esetben csökkentse a játékos életeinek számát eggyel.
- Jelenítse meg a keresett szót.
- Hogyha kitalálta a keresett szót, akkor gratuláljon a felhasználónak, ellenkező esetben hogyha felhasználó életeinek száma 0 értesítse arról, hogy a játék véget ért, mert elfogyott az élete. Megjegyzés: a harmadik eset az az, amikor ugyan arra a karakterre másodjára tippelt, ekkor azonnal bannolja a rendszer és egy új játékot kell kezdenie, de erről a kivétel üzenete értesíti.

A program futását először tesztelje úgy, hogy nincs kivétel kezelés a programkód köré építve, majd helyezzen el egy hibakezelő blokkot az alábbiak szerint:

- **BusinessLogicViolationException** kivétel esetén a konzolra írja ki a felhasználónak szánt üzenetet.
- **HangingTreeGameException** kivétel esetén a hibakódot jelenítse meg a konzolon.

A `char GetValidChar()` metódus működésének részletezése:

- Egy ciklus segítségével mindaddig kérjen be inputot a konzolról, míg egy megfelelő értéket nem kap a felhasználótól. Az esetleges **BusinessLogicViolationException** típusú kivételeket helyben kezelje le és a keletkezett kivételek üzenetét a konzolon jelenítse meg. Megjegyzés: ehhez szükséges lehet egy logikai értéket nyilvántartani, amelyet az esetleges kivétel elkapásakor tud módosítani.
- A konzolról való bekérést egy *char GetChar()* metódusban valósítsa meg az alábbiak szerint:
  - *Console.ReadLine()* segítségével olvasson be egy stringet a felhasználótól.
  - Hogyha a string hossza 0, akkor váltson ki egy **BusinessLogicViolationException** kivételt *HGT22* hibakóddal és „Nem adott meg karaktert" üzenettel.
  - Hogyha a string hossza nagyobb mint 1, akkor váltson ki egy **BusinessLogicViolationException** kivételt *HGT23* hibakóddal és „Egy karaktert adjon meg" üzenettel.
  - Hogyha a string nem a következő elemek valamelyikét tartalmazza („qwertzuiopőúasdfghjkléáűíyxcvbnmöüó-"), akkor váltson ki egy **BusinessLogicViolationException** kivételt *HGT24* hibakóddal és „Érvénytelen karakter" üzenettel.
  - A metódus a már érvényes karaktert adja vissza a megvizsgált stringből.