Órai feladat – Interfész

Bevezetés

A lenti leírás egy egyszerű számkitalálós játék elkészítését mutatja be. A játék célja, hogy egy megadott intervallumon belüli véletlenül kiválasztott számot kell eltalálni. Interfészek segítségével különböző stratégiákat fogunk ehhez készíteni (véletlen tippelés, bejárás, logaritmikus keresés, emberi játékos, stb.), majd pedig ezekből statisztikákat készíteni, hogy melyik módszer mennyire volt eredményes.

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.

Feladatok

Játékot vezérlő osztály és alapvető interfészei

Készíts egy **IJatekos** interfészt, amely az alábbi metódusokat írja elő:

- **Nyert()** – akkor kell majd meghívni, ha nyert a játékos (nincs visszatérési értéke)
- **Veszitett()** – akkor kell majd meghívni, ha veszített a játékos (nincs visszatérési értéke)

Készíts egy **ITippelo** interfészt, amely kiterjeszti az **IJatekos**-t az alábbiakkal:

- **JatekIndul(alsoHatar, felsoHatar)** – a játék indulásakor kell meghívni, megadva az eltalálni kívánt szám minimális és maximális értékét (nincs visszatérési értéke)
- **KovetkezoTipp()** – egy egész számot visszaadó metódus. Ezen keresztül fogja a játékos megadni a következő tippet

Készíts egy **SzamKitalaloJatek** nevű osztályt az alábbiak szerint:

- **MAX_VERSENYZO** – versenyzők maximális száma (konstans érték, legyen 5)
- **versenyzok** – egy *MAX_VERSENYO* méretű, *ITippelo*-t megvalósító elemeket tartalmazó tömb
- **versenyzoN** – az aktuálisan a tömbben lévő versenyzők száma
- **VersenyzoFelvetele(ITippelo)** – egy versenyzőt felvesz a tömbbe
- **alsoHatar, felsoHatar** – két egész szám, amelyek a konstruktoron keresztül kapnak értéket. A játék célja az lesz, hogy ki kell találni egy számot a két határ között
- **VersenyIndul()** – a hívást követően dob egy véletlen számot *alsoHatar* és *felsoHatar* között, ezt tárolja el egy **cel** nevű mezőben. Ezt követően hívja meg az összes felvett játékos *JatekIndul* metódusát
- **MindenkiTippel()** – ez fog végrehajtani egy kört az alábbiak szerint: egymást követően minden játékosnak meghívja a *KovetkezoTipp()* metódusát. Amennyiben sikerült eltalálni a *cel* értéket, akkor hívjuk meg a játékos *Nyert()* metódusát. Ha többen is eltalálták a számot egy körön belül, akkor mindannyian nyertek. Amennyiben legalább egy játékos nyert, akkor a kör végén (vagy közben) minden rossz tippet megadó játékosnak meg kell hívni a *Veszitett()* metódusát. A *MindenkiTippel* metódus visszatérési értéke legyen igaz akkor, ha volt nyertes és hamis, ha senki se találta el a számot.
- **Jatek()** – ez a metódus egy teljes játékot játszik le. Hívja meg először a *VersenyIndul()* metódust, majd pedig egy ciklusban addig hívja a *MindenkiTippel()* metódust, amíg az nem ad vissza igazat (tehát amíg valaki(k) nem nyert(ek)).

Vegyük észre, hogy sikerült elkészíteni a játék teljes logikáját és működését anélkül, hogy akár egyetlen játékost is készítettünk volna. Az interfészek csak azt határozzák meg, hogy ezeknek a játékosoknak milyen funkcióik lesznek majd, de hogy ezt hogy valósítják meg, azt nem.

Néhány egyszerűbb játékstratégia megvalósítása

Készíts egy **GepiJatekos** nevű absztrakt osztályt, amely megvalósítja az *ITippelo* interfészt. Ez majd a leendő gépi stratégiák őse lesz:

- **alsoHatar**, **felsoHatar** – két egész szám, amelyek között majd tippelnie kell
- **JatekIndul(alsoHatar, felsoHatar)** – beállítja a fenti két értéket
- **nyertDB**, **veszitettDB** – két egész szám, amelyek tárolják az eredményeket
- **Nyert()** – növeli a *nyertDB* értékét
- **Veszitett()** – növeli a *veszitettDB* értékét
- **KovetkezoTipp()** – az interfész miatt meg kellene valósítani, de ezen a szinten még nem tudjuk, ezért legyen absztrakt metódus

Készíts egy **VeletlenTippelo** nevű osztályt, ami a *GepiJatekos* leszármazottja:

- **KovetkezoTipp()** – valósítsuk meg ezt a metódust úgy, hogy mindig egy véletlenszámot ad vissza *alsoHatar* és *felsoHatar* között

Készíts egy **BejaroTippelo** nevű osztályt, ami a *GepiJatekos* leszármazottja:

- **aktualis** – egy egész szám, ami a következő tippet mutatja mindig
- **JatekIndul(alsoHatar, felsoHatar)** – hívja meg az ős metódusát, majd állítsa be az *aktualis* mező értékét *alsoHatar*-ra
- **KovetkezoTipp()** – visszaadja az *aktualis* értékét, majd annak értékét növeli 1-gyel

Teszt1: Most már ki tudjuk próbálni a rendszert. Hozz létre egy *SzamKitalaloJatek* objektumot és egy-egy *VeletlenTippelo* és *BejaroTippelo*-t. Ezeket kapcsold össze és játssz le egy játékot. Hogy lássuk mi történik, ideiglenesen célszerű a képernyőre kiírni a folyamat részeredményeit. Például egy 10-20 közötti játékot indítva egy lehetséges kimenet:

```
VersenyIndul
  cel:15
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:16
  Interfesz.BejaroTippelo tippje:10
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:13
  Interfesz.BejaroTippelo tippje:11
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:18
  Interfesz.BejaroTippelo tippje:12
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:18
  Interfesz.BejaroTippelo tippje:13
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:10
  Interfesz.BejaroTippelo tippje:14
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:12
  Interfesz.BejaroTippelo tippje:15
```

Mivel a Random objektum inicializációjakor a gépi időt veszi alapul, ezért előfordulhat, hogy az egymás után gyorsan létrehozott Random objektumok mindig ugyanazokat a pszeudovéletlen számokat adják. Ilyenkor

célszerű lehet a programon belül egyetlen darab Random objektumot használni, pl. egy erre a célra létrehozott statikus osztályban.

Hatékonyabb játékstratégia megvalósítása

Egészítsük ki úgy a játékot, hogy az okosabb játékosokat is támogassa annyival, hogy megmondjuk nekik, hogy a tippnél a keresett szám kisebb vagy nagyobb.

Ennek érdekében készítsünk egy **IOkosTippelo** nevű interfészt, az kiterjeszti az *ITippelo* interfészt az alábbiak szerint:

- **Kisebb()** – ezt a metódust hívjuk akkor, ha a tippnél kisebb a keresett szám
- **Nagyobb()** – ezt a metódust hívjuk akkor, ha a tippnél nagyobb a keresett szám

A fenti működéshez módosítanunk kell a *SzamKitalaloJatek* osztályt is:

- **MindenkiTippel()** – módosítsuk úgy a metódust, hogy hibás tipp esetén ellenőrizzük, hogy az aktuális játékos megvalósítja-e az *IOkosTippelo* interfészt. És ha igen, akkor a hibának megfelelően hívjuk meg a *Kisebb* vagy *Nagyobb* metódusát (kasztolni kell majd).

Készítsünk egy **LogaritmikusKereso** osztályt, ami az *IGepiJatekos* leszármazottja, de ezen felül megvalósítja az *IOkosTippelo* interfészt is.

- **KovetkezoTipp()** – felülírja ennek a metódusnak a működését úgy, hogy mindig az *alsoHatar* és *felsoHatar* közötti középső értéket adja vissza
- **Kisebb()** – mivel tudjuk, hogy ez akkor hívódik meg, amikor az előző tippnél kisebb a keresett szám, ezért a *felsoHatart* ennek megfelelően kisebbre állítja
- **Nagyobb()** – mivel tudjuk, hogy ez akkor hívódik meg, amikor az előző tippnél nagyobb a keresett szám, ezért az *alsoHatar* értékét ennek megfelelően nagyobbra állítja

Teszt2: Egy *LogaritmikusKereso* játékossal kiegészítve a játékot már hasonló eredményt várhatunk:

```
VersenyIndul
  cel:46
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:49
  Interfesz.BejaroTippelo tippje:10
  Interfesz.LogaritmikusKereso tippje:30
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:15
  Interfesz.BejaroTippelo tippje:11
  Interfesz.LogaritmikusKereso tippje:40
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:33
  Interfesz.BejaroTippelo tippje:12
  Interfesz.LogaritmikusKereso tippje:45
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:22
  Interfesz.BejaroTippelo tippje:13
  Interfesz.LogaritmikusKereso tippje:48
MindenkiTippel
  Interfesz.VeletlenTippelo tippje:17
  Interfesz.BejaroTippelo tippje:14
  Interfesz.LogaritmikusKereso tippje:46
```

Emberi játékos megvalósítása

A fentiek alapján már egyszerűen készíthetünk egy emberi játékost is. Feltételezzük, hogy okos lesz a játékosunk, ezért készítsünk egy **EmberiJatekos** osztályt, ami megvalósítja az *IOkosTippelo*-t. Ez csak a bemenetet/kimenetet fogja biztosítani a játékos felé, ezért nincs is saját mezője. Az interfész által megadott metódusokat az alábbiak szerint valósítsuk meg:

- **JatekIndul(alsoHatar, felsoHatar)** – írja ki a képernyőre, hogy kezdődik a játék megadott határokkal
- **Kisebb()** – írja ki a képernyőre, hogy „az előző tippnél kisebb a keresett szám”
- **Nagyobb()** – írja ki a képernyőre, hogy „az előző tippnél nagyobb a keresett szám”
- **KovetkezoTipp()** – írja ki a képernyőre, hogy „add meg a következő tippet”. Kérjen be egy számot, és ez legyen a függvény visszatérési értéke
- **Nyert()** – írja ki a képernyőre, hogy „nyertél”
- **Veszitett()** – írja ki a képernyőre, hogy „veszítettél”

Teszt3: egy *EmberiJatekos* felvétele után már mi is játszhatunk (egy csillag jel került az *EmberiJatekos* üzenetei elé, hogy megkülönböztethetőek legyenek a napló soroktól, kék színnel jelöltem azt, amit én írtam be a konzolra):

```
VersenyIndul.
  cel:32
*Jatek indul az alábbi határok között: [10,50]
MindenkiTippel
*Add meg a következő tippet:19
  Interfesz.EmberiJatekos tippje:19
*Az előző tippnél nagyobb a keresett szám
  Interfesz.VeletlenTippelo tippje:43
  Interfesz.BejaroTippelo tippje:10
  Interfesz.LogaritmikusKereso tippje:30
MindenkiTippel
*Add meg a következő tippet:26
  Interfesz.EmberiJatekos tippje:26
*Az előző tippnél nagyobb a keresett szám
  Interfesz.VeletlenTippelo tippje:33
  Interfesz.BejaroTippelo tippje:11
  Interfesz.LogaritmikusKereso tippje:40
MindenkiTippel
*Add meg a következő tippet:29
  Interfesz.EmberiJatekos tippje:29
*Az előző tippnél nagyobb a keresett szám
  Interfesz.VeletlenTippelo tippje:29
  Interfesz.BejaroTippelo tippje:12
  Interfesz.LogaritmikusKereso tippje:35
MindenkiTippel
*Add meg a következő tippet:31
  Interfesz.EmberiJatekos tippje:31
*Az előző tippnél nagyobb a keresett szám
  Interfesz.VeletlenTippelo tippje:50
  Interfesz.BejaroTippelo tippje:13
  Interfesz.LogaritmikusKereso tippje:32
*Veszítettél!
```

Statisztikák készítése

Szeretnénk összehasonlítani az egyes stratégiák hatékonyságát, ehhez készítsünk egy **IStatisztikatSzolgaltat** nevű interfészt az alábbi csak olvasható egész tulajdonságokkal:

- **HanyszorNyert** – egy számot ad majd vissza, hogy hányszor nyert az adott objektum
- **HanyszorVesztett** – egy számot ad majd vissza, hogy hányszor veszített az adott objektum

Egészítsük ki a *GepiJatekos* osztályt úgy, hogy megvalósítja ezt az interfészt is. Az ehhez szükséges adatokat szerencsére már tároljuk, ezért csak két tulajdonságot kell létrehozni:

- **HanyszorNyert** – az interfészben megadott tulajdonság adja vissza a *nyertDB* értékét
- **HanyszorVeszitett** – az interfészben megadott tulajdonság adja vissza a *veszitettDB* értékét

Egészítsük ki a *SzamKitalaloJatek* osztályt az alábbiak szerint:

- **Statisztika(korokSzama)** – a metódus a paraméterként megadott számú játékot játssza le egymást követően (a *Jatek* metódust kell csak meghívni többször). Ezt követően nézze végig a versenyzőket, és ha valamelyik megvalósítja az *IStatisztikatSzoglaltat* interfészt, akkor annak adatait írja ki a képernyőre.

Teszt4: emberi játékost ne használjuk, csak a három gépi stratégiát. A *Statisztika* eljárást futtatva 1000 játékra hasonlót láthatunk (ahogy az várható, a logaritmikus kereső nyeri a legtöbbet):

```
0. jatekos (Interfesz.VeletlenTippelo), NY:88   V:912
1. jatekos (Interfesz.BejaroTippelo), NY:97     V:903
2. jatekos (Interfesz.LogaritmikusKereso), NY:836       V:164
```

Érdemes megnézni, hogy ha nincs logaritmikus kereső, akkor a bejáró fog nyerni, mivel hasonlóak az esélyei mint a véletlen tippelőnek, csak kétszer nem próbálkozik ugyanazzal a számmal

Kaszinó mód beépítése

Egészítsük ki a játékot azzal, hogy legyen egy maximális tipp szám, ami alatt el kell találni a számot. Ha ez alatt senki se találta el, akkor a „kaszinó" nyert. Ehhez készítsünk egy **SzamKitalaloJatekKaszino** nevű osztályt, ami a *SzamKitalaloJatek* leszármazottja és azt kiegészíti az alábbiakkal:

- **kaszinoNyert, kaszinoVeszitett** – egész számok, amelyek tartalmazzák, hogy hányszor nyert, illetve veszített a kaszinó
- **korokSzama** – egész számot tartalmazó mező, ami a maximális körök számát tartalmazza
- **konstruktor** – az *alsoHatar* és *felsoHatar* adatok mellett itt lehessen megadni a *korokSzama* értékét is. Az első két értéket adja tovább az ős konstruktorának
- **Jatek()** – módosítsuk úgy a játékot, hogy a *VersenyIndul* hívása utáni ciklusban maximum *korokSzama*-szor hívja meg a *MindenkiTippel* metódust (vagy persze addig, amíg valaki nyert). Ha a ciklusból azért léptünk ki, mert valamelyik játékos eltalálta a számot, akkor növeljük a *kaszinoVeszitett* értékét, egyébként pedig növeljük a *kaszinoNyert* értékét
- az osztály megvalósítja az *IStatisztikatSzolgaltat* interfészt is, ahol a *HanyszorNyert* metódus a *kaszinoNyert* értékét, a *HanyszorVeszitett* metódus pedig a *kaszinoVeszitett* értékét adja vissza
- **Statisztika(korokSzama)**: hívja meg az ős azonos nevű metódusát, majd annak lefutását követően írja ki a képernyőre a kaszinó nyerési adatait is

Vegyük észre, hogy a SzamKitalaloJatekKaszino és a gépi játékosok is megvalósítják a IStatisztikatSzolgaltat interfészt. Pedig egymástól meglehetősen távoli osztályok közös ős nélkül, de egy bizonyos szempont szerint hasonló működést várunk tőlük (számolják az eredményeket). És az interfészen keresztül közösen kezelhetők.

Teszt5: készítsünk egy *SzamKitalaloJatekKaszino* objektumot 1-100 közötti határokkal és 6-os kör limittel. Így hasonló eredményt kaphatunk (7 lépésnél a logaritmikus keresés már mindig meg fogja találni az eredményt időben):

```
0. jatekos (Interfesz.VeletlenTippelo), NY:54   V:609
1. jatekos (Interfesz.BejaroTippelo), NY:51     V:612
2. jatekos (Interfesz.LogaritmikusKereso), NY:565       V:98
Kaszino, NY:337 V:663
```

Opcionális kiegészíti lehetőségek

Naplózás

Ha valakit (jogosan) zavar, hogy az üzleti logikát megvalósító osztályokból írunk a konzolra, akkor célszerű ezt is egy interfésszel megvalósítani. Készítsünk egy **INaploz** interfészt, aminek egy

naplobaIr(szöveg) metódusa van. A *SzamKitalaloJatek*-nak legyen egy ilyen típusú mezője, és a konzolra írás sorokat mindenhol cseréljük ki arra, hogy ha van ebben a mezőben bármi, akkor annak meghívjuk a *naplobaIr* metódusát, ha nincs, akkor pedig nem naplózunk.

Ezt követően készíthetünk különböző *INaploz* megvalósításokat, pl. olyat ami a konzolra írja az üzeneteket, de olyat is, ami fájlba.