

Feladatmegoldás

Feladatok

1 Egy futsal csapatban egy kapus, és négy mezőnyjátékos (egy csatár, két szélső és egy védő) . A feladatunk, hogy egy olyan programot készítsünk, amely segítségével az elérhető játékosokból helyes felépítésű csapatot állíthatunk elő.

Minden játékosról ismert a neve és a pozíciója, amely a fent felsorolt értékek egyike lehet. Egy csapatba pontosan 5 játékos kell, a fentebb definiált módon. A program indításkor generáljon játékosokat, és a felhasználónak legyen lehetősége csapathoz rendelni őket.

Készítsük el a játékosokat reprezentáló **Player** osztályt az alábbi tagokkal.

- string name: a játékos neve
- Position pos: a játékos pozíciója (Goalkeeper, Forward, Winger, Defender)

Az adattagok beállítása a konstruktoron keresztül lehetséges. Definiáljuk felül az osztály **ToString** metódusát, amely a játékos nevét és pozícióját tartalmazó karakterláncot adja vissza.

A csapatot a **Team** osztály reprezentálja. Valósítsuk meg az alábbi tagokat az osztályban.

- Player[] players: a csapatban levő játékosok tömbje
- int NumberOfPlayers: a csapatban levő játékosok száma
- bool IsFull: a játékosok száma elérte az ötöt?
- bool IsIncluded(Player): eldönti, hogy a parameter szerepel-e már a csapatban
- bool IsAvailable(Player): eldönti, hogy a parameter pozíciója szabad-e
- void Include(Player): játékos hozzáadása a csapathoz

A főprogramban készítsük el az alábbi statikus metódust.

- Player[] RandomPlayers(int): adott mennyiségű játékos generálása

A **Main**-ben a véletlenszerűen generált játékosok közül választhat a felhasználó, és a csapatba rendelheti őket.

2 Készítsünk bőlényvadász játékot. A bőlénycsorda N bőlényből áll, amelyek egy $M \times M$ -es játéktéren a bal felső sarokból, a $(0,0)$ koordinátáról indulnak, és a játéktér jobb alsó sarkának irányába haladnak. Az N és M értékeket a felhasználó adhatja meg a játék kezdetén.

A játék körökre osztott. Minden körben minden bőlény véletlenszerűen lép egyet jobbra $(x + 1)$, lefelé $(y + 1)$ vagy átlósan jobbra lefelé $(x + 1, y + 1)$, de mindig a pálya határain belül maradván.

A felhasználó minden körben lövést ad le egy kiválasztott koordinátára, így eltalálva az ott lévő bőlények egyikét. Ha eltalált egy bőlényt, az kiesett a játékból. A bőlények akkor győznek, ha bármelyik elér a célba, a felhasználó pedig akkor, ha sikerül minden bőlényt kiiktatnia, mielőtt bármelyik a célba érne.

A játéktér a **Field** osztály reprezentálja.

- Tároljuk el a játéktér méretét (M) egy mezőben.
- A mező értékét a konstruktorban állítsuk be a paraméterként átadott értéknek megfelelően.
- Készítsünk egy **TargetX** és egy **TargetY** tulajdonságot, amelyek lekérdezésekor a játéktér cél koordinátáit (a pálya jobb alsó sarkának koordinátáit) kapjuk vissza.
- Definiáljunk egy **AllowedPosition** nevű metódust, amelynek két egész értéket fogad paraméterül, majd visszaadja, hogy az ezek által leírt koordináta a játéktér része-e. (Vagyis ha ide lépne egy bōlény, akkor még a játéktéren belül maradna-e.)
- A **Show()** nevű metódus meghívásakor rajzoljuk ki a játéktér körvonalát a képernyőre (például | és - karakterek segítségével).

Készítsük el a **Buffalo** osztályt a bōlények reprezentálására.

- Tároljuk el a bōlény aktuális pozícióját (x és y koordinátáit) egy-egy változóban. A bōlény állapotát (aktív/nem aktív) egy logikai típusú mezőben tároljuk.
- Készítsünk egy **X** és egy **Y** tulajdonságot, amelyek lekérdezésekor a bōlény aktuális koordinátáit kapjuk vissza.
- A **Move** metódus egyetlen paramétere egy **Field** típusú példány. A metódus meghívásakor valósítsuk meg a bōlény egy lépését a fenti szabályok szerint, de csak olyan mezőre léphet, amelyre a paraméterként kapott példány **AllowedPosition** igaz értékkel tér vissza.
- A **Deactivate** metódus meghívásakor a bōlény állapota nem aktív (hamis) értéket vesz fel.
- A **Show** metódus hívásakor a bōlény koordinátájának megfelelő helyre írjunk ki egy B karaktert zöld színnel, ha a bōlény aktív, vagy piros színnel, ha nem aktív.

Készítsük el a **Game** osztályt az alábbi tagokkal.

- Legyen egy **Field** típusú mezője, amely a játéktér leíró példányt tárolja. A bōlényeket tároljuk egy tömbben vagy listában.
- Az **IsOver** tulajdonság a játék aktuális állapotát adja meg: igaz értékkel tér vissza, ha a játék véget ért, és hamis értékkel, ha még folyamatban van (lásd a fenti szabályokat).
- A konstruktor a játéktér méretét és a bōlények számát várja paraméterül, majd ezeknek megfelelően létrehozza a játéktér és a szükséges számú bōlényt.
- A privát **VisualizeElements** metódus törli a képernyőt, majd kirajzolja a játéktér és a bōlényeket a képernyőre a **Show** metódusok meghívásával.
- A privát **Shoot** metódus egy x és egy y koordinátát kér a felhasználótól, majd minden olyan bōlényt deaktivál a megfelelő metódus meghívásával, amely a felhasználó által megadott pozícióban tartózkodik.
- A **Run** metódus meghívásakor rajzoljuk ki a játék állapotát a **VisualizeElements** meghívásával, majd hívjuk meg a **Shoot** metódust. Ismételjük ezeket a lépéseket az **IsOver** aktuális értékétől függően.

A főprogramban készítsünk egy példányt a **Game** osztályból, majd a **Run** metódus hívásával indítsuk el a játékot.