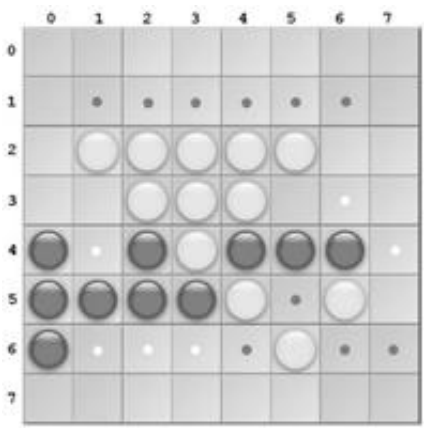
2. Programozás 40 pont

TUDNIVALÓK ÉS JAVASLATOK FELKÉSZÍTŐ, ILLETVE JAVÍTÓ TANÁROK SZÁMÁRA

Ebben a feladatban a vizsgázónak konzolos és/vagy grafikus alkalmazást kell készítenie a tanult programozási nyelv (Java vagy C#) és fejlesztői környezet felhasználásával.



A reversi játékot általában 8×8 mezőből álló négyzetrácsos táblán játsszák. Ebben a feladatban a tábla sorait és oszlopait is ***0-tól 7-ig*** azonosítjuk az ábra szerint. A játékot legjobb olyan korongokkal játszani, amelyek két oldala különböző színű (feladatunkban kék és fehér). A két játékos felváltva rakja le korongjait. A soron következő játékos csak olyan helyre rakhat, ahol meg tudja ***fordítani*** az ellenfél legalább egy korongját. Ez úgy lehetséges, hogy az éppen letett korong és a játékos másik korongja között egyenes vonalban vízszintesen, függőlegesen vagy átlósan kizárólag csak az ellenfél egy vagy több korongja található. Az egyes játékosok következő lehetséges lépéseit az ábrában kisebb körökkel szemléltetjük. Például a kék (sötét) játékos az 1;5 (sor;oszlop) mezőre azért rakhat szabályosan, mert a 4;2 mezőn lévő korongjával közrefog átlósan 2 db fehér korongot, így azok megfordulnak. A fehér játékos azért nem rakhat a 3;5 mezőre, mert nincs olyan korongja, amivel kizárólag csak kék (sötét) korongokat fogna közre, így erre a mezőre lépve nem tudna fordítani. Az ábra a forrás állomány adataival készült, így tanulmányozása a megoldáshoz segítséget nyújthat. A továbbiakban a reversi játékkal kapcsolatos feladatokat kell megoldania.

A megoldás során vegye figyelembe a következőket:

- *A program készítése során törekedjen az objektum orientált (OOP) megoldásra, amire a feladatsor ajánlásokat is tartalmaz. Amennyiben a programot ilyen módon nem tudja elkészíteni, akkor a feladatokat saját osztály létrehozása nélkül is megoldhatja, de így kevesebb pontot ér a megoldása. Ebben az esetben, ha a feladat jellemző vagy metódus létrehozását kéri, akkor Önnek saját alprogramot (függvényt, eljárást) kell készítenie, amely* ***paramétereken keresztül*** *kommunikál a hívó programmal!*
- *A megoldását úgy készítse el, hogy az azonos szerkezetű, de tetszőleges input adatok mellett is megfelelően működjön!*
- ***Az azonosítókat kis- és nagybetűkkel is kezdheti!***

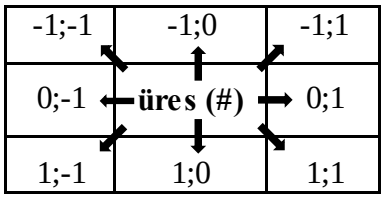
1. Készítsen ***grafikus*** felhasználói felületű programot a következő feladatok megoldására, amelynek a forráskódját `ReversiGUI` néven mentse el!
2. Hozzon létre saját osztályt `Tabla` azonosítóval és definiáljon benne egy karakter típusú mátrixot (kétdimenziós tömböt) `Allas` azonosítóval, melyben egy játék pillanatnyi állását tudja majd tárolni! A mátrix sorai és oszlopai 0-tól 7-ig legyenek indexelve!
3. Készítse el az osztály konstruktorát, ami a következő feladatokat hajtja végre:
   a. Inicializálja az `Allas` mátrixot 8×8-as mérettel!
   b. Feltölti az `Allas` mátrixot a „#”, „K” és „F” karakterekkel egy szöveges állományból. A feldolgozandó szöveges fájl nevét a konstruktor paramétereként adjuk át! A feladat megoldásához használandó `allas.txt` állomány 8 sora, soronként 8 karakterrel tárolja egy játék állását. A tábla üres mezőit a „#” karakter, a játékosok korongjait a „K” (kék) és „F” (fehér) karakterek kódolják.
   c. A konstruktort tetszőlegesen tovább bővítheti a későbbi feladatok megoldásához!
4. Hozzon létre egy `Tabla` típusú osztálypéldányt (objektumot), melynek a konstruktora az `allas.txt` forrás állomány nevét kapja aktuális paraméterként feldolgozásra!

5. Készítsen a grafikus tervező használatával és/vagy a `Megjelenit` metódus kódolásával az ***1. minta*** szerint grafikus felhasználói felületet, ami megjeleníti az `Allas` mátrixban eltárolt játék állását! A `Megjelenit` metódust a `Tabla` osztályban helyezze el! Színes mintákat a program futásáról a `minta*.jpg` állományokban talál! A grafikus felület kialakításhoz úgy válaszon osztályokat, hogy a 8-10. feladatok megoldhatók legyenek!

6. Definiáljon a `Tabla` osztályban metódust a következő algoritmus kódolásával! (Ha nem a `Tabla` osztályban kódolja a metódust, akkor az `Allas` mátrix is a függvény paramétere legyen!)

```
Függvény VanForditas(jatekos: Karakter, sor, oszlop,
                     iranySor, iranyOszlop: Egész): Logikai
   Változó aktSor, aktOszlop: Egész
   Változó ellenfel: Karakter
   Változó nincsEllenfel: Logikai
   aktSor:=sor + iranySor
   aktOszlop:=oszlop + iranyOszlop
   ellenfel:='K'
   Ha (jatekos='K') akkor
       ellenfel:='F'
   Elágazás vége

   nincsEllenfel:=igaz

   Ciklus amíg (aktSor>0 és aktSor<8 és
   aktOszlop>0 és aktOszlop<8 és t[aktSor, aktOszlop]=ellenfel)
       aktSor:=aktSor + iranySor
       aktOszlop:=aktOszlop + iranyOszlop
       nincsEllenfel:=hamis
   Ciklus vége

   Ha (nincsEllenfel vagy aktSor<0 vagy aktSor>7 vagy
   aktOszlop<0 vagy aktOszlop>7 vagy
   t[aktSor, aktOszlop]<>jatekos) akkor
       Térj vissza hamis
   Elágazás vége

   Térj vissza igaz
 Függvény vége
```

A metódus a megadott játékos, megadott lépését, megadott irányban határozza meg, hogy az adott irányban történhet-e fordítás. Az `iranySor` és `iranyOszlop` paraméterek a következők szerint határozzák meg a feltételezett fordítás irányát:

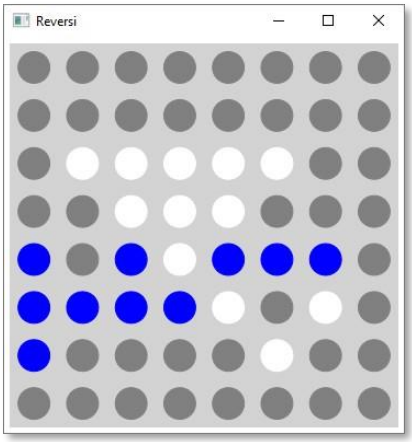


7. Készítsen a `Tabla` osztályban logikai értékkel visszatérő metódust `SzabalyosLepes()` azonosítóval, ami meghatározza egy megadott játékos megadott lépéséről, hogy szabályos lépés vagy nem szabályos lépés! Szabályosnak tekintünk egy lépést, ha a megadott cella üres és a nyolc irány valamelyikéből (lásd előző feladat) a megadott játékossal történhet fordítás. Megoldásában használja fel a korábban elkészített `VanForditas()` metódust is!

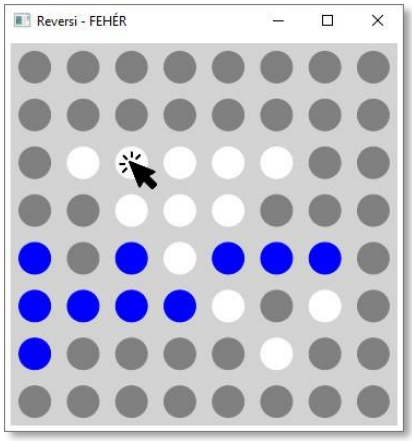
8. A program indulásakor a kék vagy fehér korongokra kattintva lehessen kiválasztani a játékban következő játékost! Az így kiválasztott játékos „színe” kerüljön az ablak címsorába a ***2. mintá***nak megfelelően! Ha indulás után üres (szürke) mezőre kattintunk, akkor ne történjen semmi!
9. Miután sikeresen kiválasztottunk egy játékost az előző feladatban leírtak szerint, határozzuk meg egy üres (szürke) mezőre kattintva, hogy az előzőleg kiválasztott játékos lépése szabályos-e erre a mezőre! Ezt az ablak címsorában jelezzük a ***3. és 4. minták***nak megfelelően! Megoldáshoz használja fel a `SzabalyosLepes()` metódust!
10. Legyen lehetőség minden üres cella ellenőrzésére és új játékos (szín) választására a fenti módszerek megismétlésével!

Minták:

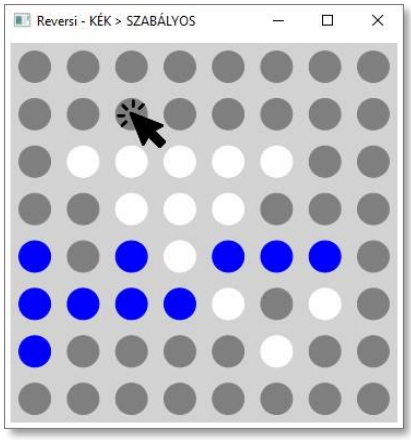
1. minta



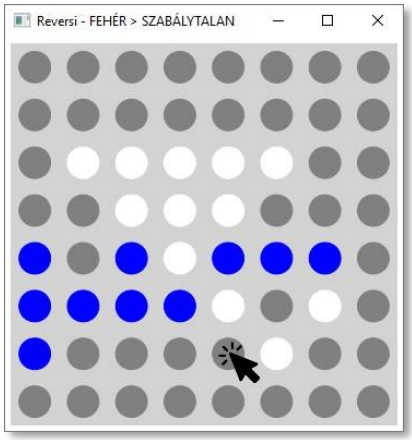
2. minta



3. minta



4. minta



Forrás:

http://mek.oszk.hu/00000/00056/html/138.htm (2017.06.10)