Programozási Technológia – 2. beadandó (2. feladat)
Abdurasitov Alekszandr
A49MZV

KITOLÁS

Feladatleírás

Kitolás

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelyen kezdetben a játékosoknak n fehér, illetve n fekete kavics áll rendelkezésre, amelyek elhelyezkedése véletlenszerű. A játékos kiválaszthat egy saját kavicsot, amelyet függőlegesen, vagy vízszintesen eltolhat. Eltoláskor azonban nem csak az adott kavics, hanem a vele az eltolás irányában szomszédos kavicsok is eltolódnak, a szélső mezőn lévők pedig lekerülnek a játéktábláról. A játék célja, hogy adott körszámon belül ($5n$) az ellenfél minél több kavicsát letoljuk a pályáról (azaz nekünk maradjon több kavicsunk). Ha mindkét játékosnak ugyanannyi marad, akkor a játék döntetlen.

A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 4×4, 6×6) és így a lépésszám (15, 20, 30) megadásával, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy melyik játékos győzött (ha nem lett döntetlen), majd kezdjen automatikusan új játékot.

Feladat elemzése

A feladatot **modell-view architektúrában** szükséges elkészíteni, ezek alapján a modellben a **tábla egy mátrixnak feleltethető meg**, aminek az elemei egy-egy felsoroló típusban definiált elemek. A **felsorolóban WHITE, BLACK és NOBODY** elemek fognak szerepelni, attól függően hol helyezkednek el az adott kövek. Ezek a kövek elhelyezkedése véletlenszerű, így **szükség lesz egy véletlenszám generáló segédfüggvényre,** mely alapján elhelyezhetjük a köveket. A generálófüggvénynek elég 0-2 intervallumban generálnia a véletlen számokat, ezek alapján a 0 – WHITE, 1 – BLACK, 2 – NOBODY, amelyik számot generálja a függvény, úgy a modell konstruktorában azt az elemet helyezzük el a mátrix x és y helyére.

Fehér és fekete kövek száma nem haladhatja meg a tábla N méretét így, ha az egyik kőnek a számossága = N-nel, abban az esetben a másik követ kell lehelyezni a helyére, függetlenül attól, hogy mit generált a véletlenszám segédfüggvény. Amennyiben mindkét kő számossága eléri a N-et, és még vannak további NULL elemei a mátrixnak, így azokat az elemeket feltöltjük NOBODY elemekkel.

A **View része a programnak** viszonylag egyszerű, ezt **2 részre bonthatjuk**. A **főablak**, ahol a felhasználó információt szerezhet a játék szabályairól, illetve kiválaszthatja a tábla méretét, illetve a **játékablak**, ami azután jelenik meg, miután a felhasználó kiválasztotta a tábla méretét, ahol a táblaméretével megegyező gombokat fog látni, mely **gombokat szintén egy mátrixba helyezzük, ezzel létrehozva a gombok mátrixát**, hogy könnyen lehessen összeegyeztetni a modell mátrixával.

Ezek után **minden egyes gombhoz egy eseménykezelőt rendelünk**, amelyik gombra kattintott a felhasználó, azt a rendszer leellenőrizi, hogy a saját köve szerepel-e benne, ha igen még egy gombra kattinthat a felhasználó, ami felé eltolhatja a kiválasztott követ, ha rossz helyre kattintott úgy a felhasználónak újra ki kell választania, melyik követ szeretné eltolni.

Programozási Technológia – 2. beadandó (2. feladat)
Abdurasitov Alekszandr
A49MZV

Megvalósítási terv

3 rétetű architektúrára lesz szükség. Az „indító” része a programnak „main” réteg, ahol a program main függvénye helyezkedik el, ahol meghívjuk a View rétegből a főablak konstruktorát és beállítjuk a láthatóságát.

A **view rétegben két java class helyezkedik el**, amely két külön ablaknak feleltethető meg. a **MainWindow (főablak)-ban** a felhasználó 3 gomb közül választhat, attól függően, hogy milyen játéktáblával szeretne játszani. **Mindhárom gombhoz egy eseménykezelőt rendelünk**, attól függően melyik gombra kattintott, **meghívjuk a játékablak konstruktorát** azzal a N számmal, amilyen méretű (NxN) táblát szeretne a felhasználó.

View réteg második ablaka a program fő része, a játékablak (GameWindow). A **játékablak konstruktorában 2 ciklussal hozzáadjuk a gombokat a képernyőre** egy ún. addButton függvény segítségével, mely függvényen belül létrehozzuk a gombokat, azokat behelyezzük a gombok mátrixába, és ezek **után minden egyes gombhoz egy eseménykezelőt rendelünk. Az eseménykezelő kétféleképpen működhet.** Ez alapján létrehozzunk egy clickedState nevű változót, mely 0 és 1 értéket vehet fel. Amennyiben a változó értéke 0, úgy a felhasználó kiválaszthat egy a saját kövei közül egyet, majd ezt a változót 1-re változtatjuk, így a következő kattintásánál egy szomszédos helyre kattinthat, ezáltal azt követ eltolhatja x vagy y irányban 1, vagy -1 értékkel. Ezek után a clickedState változót újra 0-ra változtatjuk. Mindezek mellett az **eseménykezelőben folyamatosan frissítjük a megjelenő szövegeket**, a gombok hátterét és az aktuális fennmaradt lépésszámot, illetve **minden lépésnél leellenőrizzük, hogy a játék véget ért-e**.

A játéknak 2 féleképpen lehet vége. Az első esetben, ha nem marad több lépési lehetőség, így kiszámítjuk a modellben, hogy hány kő maradt a táblán, akinek több köve maradt az nyert. ha kövek száma megegyezik, úgy a végeredmény döntetlen lesz. A második esetben, ha az egyik játékosnak az összes követ kitolták, abban az esetben az ellenfél játékos nyer.

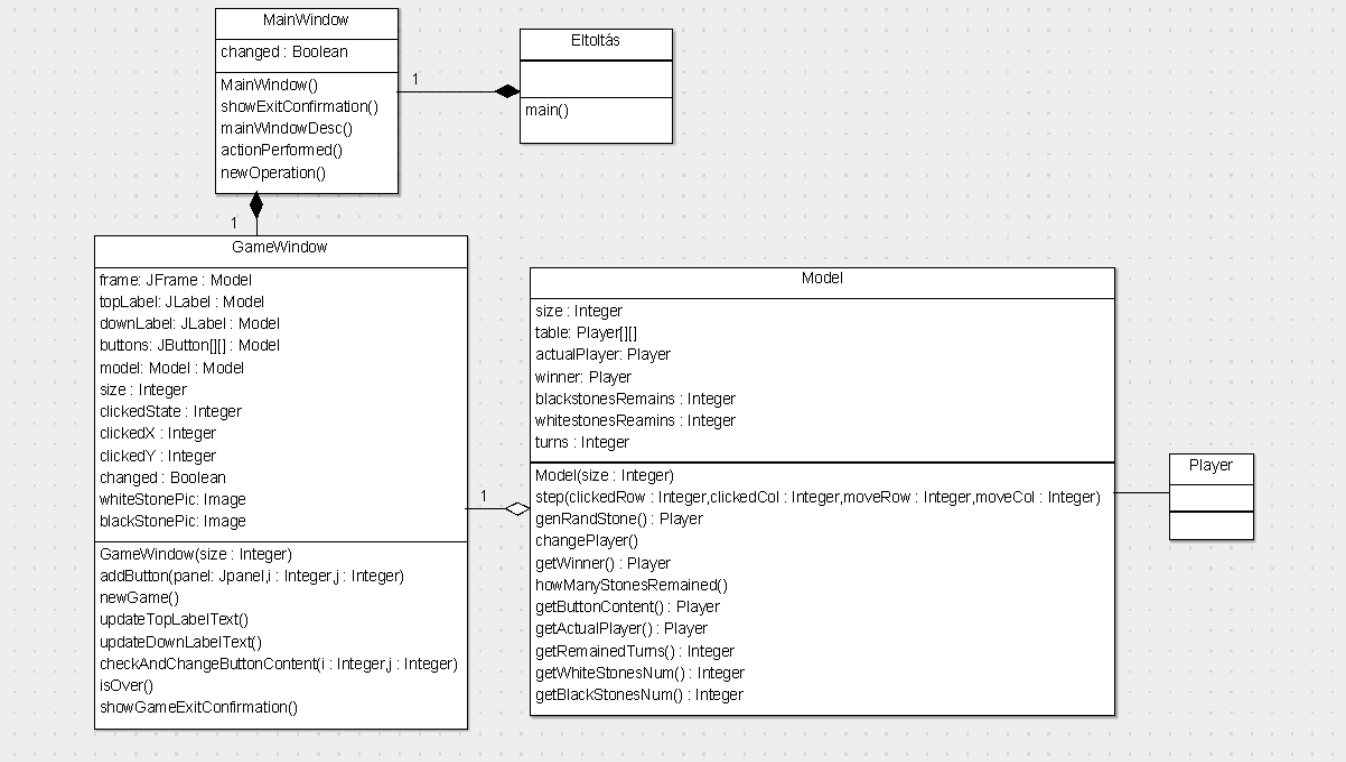
Ezek alapján a modell rétegben, a program működési egységét írjuk le. Attól függően milyen táblaméretet választott a felhasználó, olyan nagyságú felsoroló típusú mátrixot hozunk létre. A felsorolóban 3 elem van, a WHITE, BLACK és NOBODY. A konstruktorban figyelnünk kell, hogy minden egyes színű kőből, maximum N mennyiség lehet, ha ezt a mennyiséget eléri az egyik szín, abban az esetben a másik színt rakjuk a mátrixba, ha mindkét mennyiség eléri a N számosságot, úgy a többi null elemű mátrixra a NOBODY-t helyezzük el.

A modell főalgoritmusa egy **step()** nevű függvény, mely függvény **4 paramétert vár**, ahol az első két paraméter a kiválasztott kő x,y koordinátája, a második két paraméter pedig a második gomb koordinátái, ahová szeretné a felhasználó eltolni a kiválasztott követ. Ezek alapján **a függvényben leellenőrizzük, hogy melyik irányba fog történni az eltolás,** és az alapján minden egyes gombot eltoljuk *(lásd algoritmus)*. A függvény végén mindig kiszámoljuk hány köve maradt a fehér és a fekete játékosnak, hogy a játék végét könnyen lehessen jelezni.

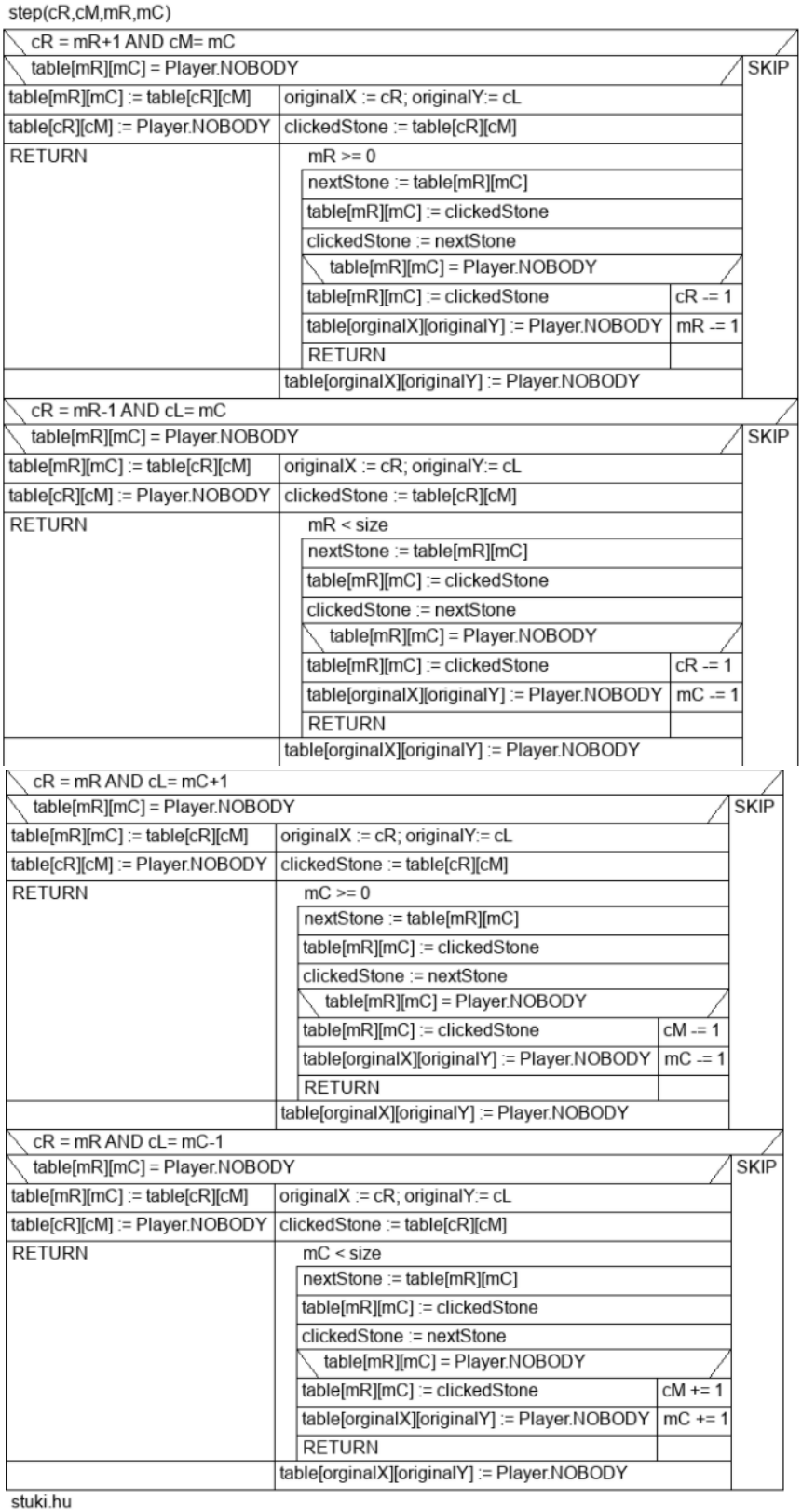
Ezenfelül a modellben létrehozzuk a megfelelő gettereket.

Programozási Technológia – 2. beadandó (2. feladat)
Abdurasitov Alekszandr
A49MZV

Ezek alapján az osztálydiagramm az alábbiak szerint fog kinézni:



A Modell step() algoritmusa



Esemény-eseménykezelők párosítások

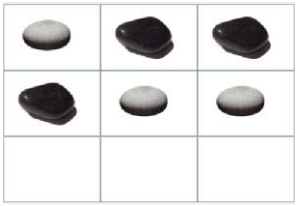
Megvalósítási terv alapján az alábbi ablakoknál szükségesek az eseménykezelők:

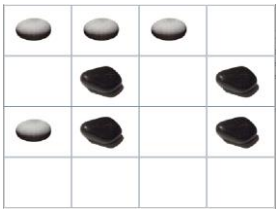
1. **Főablak (MainWindow.Java)**
   a. Három gomb helyezkedik el a főablaknál, ezek rendre a következők:
      i. 

         1. Meghívja a gameWindow konstuktorát 3-as paraméterrel, mely létrehozza a 3x3-as játéktáblát gombokkal. Létrehozza a gombok mátrixát, illetve elkészíti a Modellt.
      ii. 

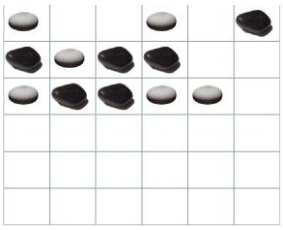
         1. Meghívja a gameWindow konstuktorát 4-as paraméterrel, mely létrehozza a 4x4-es játéktáblát gombokkal. Létrehozza a gombok mátrixát, illetve elkészíti a Modellt.
      iii. 

         1. Meghívja a gameWindow konstuktorát 6-as paraméterrel, mely létrehozza a 6x6-os játéktáblát gombokkal. Létrehozza a gombok mátrixát, illetve elkészíti a Modellt.
   b. Kilépés gombra kattintva megjelenik egy megerősíti üzenet, hogy biztosan ki szeretne-e lépni a felhasználó programból. Amennyiben igen a program bezárul.

2. **Játékablak (GameWindow.java)**
   a. Játékablak létrehozása utána az alábbi gombok jelennek meg gombok állapottól és a játéktábla méretétől függően.
      i. 
         1. Minden egyes gombhoz egy eseménykezelő van párosítva, mely kétféleképpen működhet. Amennyiben a kattintás állapota 0, úgy a felhasználó kiválaszthatja a saját kövét, ezek után a kattintás állapota 1-re változik.
         2. Kattintás 1-es állapotában a felhasználó kattinthat egy másik gombra, ahová el szeretné tolni a kövét. Amennyiben szabályok szerint megfelelő helyre kattintott, a modell step() metódusa lefut, ezek után minden egyes gombnak lekérdezzük az állapotát és frissítjük a gombok háttérét. Ezek után a kattintás állapota újra 0-ra változik, és az ellenfél léphet.

ii. 

1. Minden egyes gombhoz egy eseménykezelő van párosítva, mely kétféleképpen működhet. Amennyiben a kattintás állapota 0, úgy a felhasználó kiválaszthatja a saját kövét, ezek után a kattintás állapota 1-re változik.
2. Kattintás 1-es állapotában a felhasználó kattinthat egy másik gombra, ahová el szeretné tolni a kövét. Amennyiben szabályok szerint megfelelő helyre kattintott, a modell step() metódusa lefut, ezek után minden egyes gombnak lekérdezzük az állapotát és frissítjük a gombok háttérét. Ezek után a kattintás állapota újra 0-ra változik, és az ellenfél léphet

iii. 

1. Minden egyes gombhoz egy eseménykezelő van párosítva, mely kétféleképpen működhet. Amennyiben a kattintás állapota 0, úgy a felhasználó kiválaszthatja a saját kövét, ezek után a kattintás állapota 1-re változik.
2. Kattintás 1-es állapotában a felhasználó kattinthat egy másik gombra, ahová el szeretné tolni a kövét. Amennyiben szabályok szerint megfelelő helyre kattintott, a modell step() metódusa lefut, ezek után minden egyes gombnak lekérdezzük az állapotát és frissítjük a gombok háttérét. Ezek után a kattintás állapota újra 0-ra változik, és az ellenfél léphet

b. Kilépés gombra kattintva megjelenik egy megerősítő üzenet, hogy biztosan ki szeretne-e lépni a felhasználó a játékból. Amennyiben igen, bezáródik a játékablak és csak a főablak marad nyitva, ahol választhat egy új játéktáblát.

Budapest, 2020.11.08.

Készítette,
Abdurasitov Alekszandr
A49MZV