Függvények

BorderBuilder:

public boolean build(ArrayList<Field> board, ArrayList<Point> coordinates, ArrayList<BufferedImage> images):

Ez a függvény a coordinates paraméterként kapott listába rak bele olyan koordinátákat, ahol majd folytatni lehet a pálya építését. Ehhez kipróbálja az összes board listában lévő mező szomszédját, hogy arrafelé lehet-e folytatni. Kezdetben belerakja az összes mező minden olyan szomszédjának a pozícióját, ahol nincs mező, ezután a megszünteti az ismétlődést, végül megnézi, hogy a maradék közül ténylegesen melyiken lehet folytatni.

Field:

public void addNeighbor(Field f):

Ez a függvény a paraméterként kapott mezőt megpróbálja hozzáadni a saját szomszédjai közé. Ez akkor sikerül, ha a mező közepét jelképező pozíció alapján egymás mellett vannak.

public int getIndex(Field f):

Ez a metódus a paraméterként kapott mezőről mondja meg, hogy magához képest hol helyezkedik el. Ha a két mező szomszédos, akkor egy indexet ad vissza, hogy hányadik szomszédja a mező. Ha pedig nem szomszédosak, akkor -1-et.

public boolean equals(Field f):

Megmondja, hogy a paraméterként kapott mező és a saját pozíciója megegyezik-e.

public Point getNeighborsPosition(int i):

Paraméterként kap egy egyész számot, majd visszaadja a szám által hivatkozott szomszédjának a helyét. Ha nincs az adott pozícióban szomszédja akkor is visszaadja, hogy mi lenne a pozíciója a mezőnek.

public void deleteNeighbor(Field f):

Amennyiben a szomszédjai között van a paraméterként kapott mező, akkor törli a saját szomszédjai közül a mező.

public Field getFurthestNeighbor(int i):

Ez a rekurzív függvény kap egy egész számot paraméterként, ami a valamelyik szomszédjának a sorszámát jelöli. Ha az adott szomszédja létezik a mezőnek akkor azon meghívja magát és ezt adja vissza. Ha pedig nincs az adott irányban szomszédja vagy van rajta valami, akkor saját magát adja vissza.

ExitButtonActionListener:

public void actionPerformed(ActionEvent e):

Befejezi a program futását.

FieldBuilder:

public boolean build(ArrayList<Field> board, ArrayList<Point> coordinates, ArrayList<BufferedImage> images):

Ez a metódus két meglévő mező alapján csinál még kettőt, hogy a négy mező kiadja a rombusz alakzatot, amit a játékosok le tudnak helyezni. Ehhez először a paraméterként kapott boardtól elkéri a két utolsó elemet, ha a második mező valamelyik egyébként már a táblán lévő mező pozíciójába esik, akkor leveszi a tábláról és visszatér hamissal. Ha nem akkor hozzáadja a két mezőt a másik szomszédjai közé, ezután megnézi, hogy a második az elősnek melyik oldalán van. Ez alapján egy megfelelően forgatott képet tesz a harmadik paraméterként kapott listába és létrehoz két megfelelő pozícióban lévő mezőt. Ha ezek valamelyik másik már a táblán lévő mező pozíciójába esnek, akkor törli az utolsó képet és a két új mezőt, majd az első két mező közül is törli a másodikat még az első szomszédjai közül is és visszatért hamissal. Ha ez sem történik meg, akkor hozzáadja az összes mező szomszédjaihoz az utolsó három mezőt és hozzájuk is az összes többi mezőt.

FieldBuilder2:

public boolean build(ArrayList<Field> board, ArrayList<Point> coordinates, ArrayList<BufferedImage> images):

Ez a függvény visszaállítja az első paraméterként kapott lista alapján a harmadik paraméter képeit. Ezt úgy teszi, hogy egy FieldBuildert és egy mezőket tartalmazó ideiglenes listát. Az ideiglenes listába bele teszi board minden négyes csoportjának első két elemét kezdve az elsővel. Mikor belerakta az egyik csoport elemeit, akkor meghívja ezzel a listával és a paraméterként kapott képek listájával a FieldBuilder megfelelő metódusát, majd törli az ideiglenes lista tartalmát és rakja bele a következő csoport elemeit.

Game:

public Game(Player player1, Player player2):

Konstruktor, beállítja az adattagokat, létrehoz egy pályát és inicializálja a játék kinézetét.

public Game(Map map, ArrayList<Player> players):

Másik konstruktor, de ez akkor hívódik meg , ha a pálya már létre lett hozva így azzal nem kell törődni. A játékosokat pedig ebben az esetben egy listában kapja meg.

protected void paintComponent(Graphics g):

Ez a játék ősosztályának, a JPanelnek az egyik metódusának a felüldefiniálása. Ez rajzolja ki a játékosokat, a mezőket és a lehetséges lépéseket jelképező képeket.

public void addNumbers():

Ez a függvény a játék egyik adattagját feltölti képekkel.

InnerBorderBuilder:

public boolean build(ArrayList<Field> board, ArrayList<Point> coordinates, ArrayList<BufferedImage> images):

Törli a paraméterként kapott koordinátákat. Majd végigmegy a tábla mezőin és ha valamelyiknél a bárányok száma 0 és van szomszédja, ami null, akkor a mező pozícióját hozzáadja a koordináták listájához. Ezután pedig megszünteti az ismétlődést.

LoadGameButtonActionListener:

public void actionPerformed(ActionEvent e):

A beérkezett esemény hatására visszaszerializálja az adatokat a fájlokból, majd létrehoz egy pályát és a egy játékot.

MainMenu:

public MainMenu():

Konstruktor, ami a főmenü kinézetét inicializálja.

MainMenuButtonActionListener:

public void actionPerformed(ActionEvent e):

Létrehoz egy főmenü példányt és bezárja a jelenlegi ablakot.

Map:

public Map(Point o):

Konstruktor, inicializálja az adattagokat.

public Point getBasis(Point p):

Megadja, hogy a paraméterként kapott pontnak milyen vektor mutat az origóból. Majd az értéket kerekíti, hogy egy hatszög közepét jelölje.

public Point roundBasis(double q, double r):

A paraméterként kapott két bázisvektort kerekíti, hogy azok valamelyik hatszög közepébe mutassanak.

public Point getFieldCenter(Point vector):

Megadja, hogy a paraméterként kapott bázisvektor által mutatott hatszögnek hol van a középpontja a képernyőn.

public Point getImageCenter(int i):

Megmutatja, hogy a táblán lévő i-dik négyes csoportnak hol a középpontja a képernyőn, ezt az alapján teszi, hogy mind a négy mezőnek megkérdezi a középpontját az előző függvényt használva, majd átlagolja ezeket.

MouseListener1:

public void mouseClicked(MouseEvent me):

Egy felüldefiniálás, hogy paraméterként kapott esemény hatására a program a megfelelő módon működjön. Minden esemény lekezelése után újra rajzolja a játékot.

private void createTable():

Egy metódus, ami az egyik lehetséges viselkedést írja le. Ez akkor hívódik meg, ha a pálya még nincs teljesen kész, nem elég nagy a mérete. Két viselkedése van ebben az esetben is a programnak, az egyik az, amikor nem jött esemény arra vonatkozóan, hogy a négyes csoportok közül az elsőt hova rakjuk, ekkor ezt eltárolja és segítségként egy builderrel megadja, hogy a második mezőt hova lehet majd tenni. A másik pedig, ha megadtuk már az első mezőt és azt folytatjuk, ekkor a megfelelő builderrel felépítjuk a két mező által meghatározott mező négyest.

private void putDownSheep():

Ez a metódus felrakja a játékosokat a táblára, ha az egyes játékos van soron, akkor az ő mezőihez adja hozzá a választott mezőt, ha pedig a kettes az övéihez.

private void play():

Az utolsó lehetséges viselkedést írja le. Mikor a pálya is kész van és a játékosok is kiválasztották a kezdő mezőiket. Ha még nem választottuk ki honnan akarunk lépni, az adott játékostól azt az eseményt várja majd, ha ez megvolt egy builder segítségével kirajzolja, hova léphet. Majd következik a tényleges lépés. Ekkor az eredeti helyen lévő bárányok számát lefelezi és hozzáadja ahhoz a mezőhöz, ahova léptünk. A lépést követően pedig leellenőrizzük, hogy nem állt-e elő az a helyzet, hogy valamelyik játékos nem tud lépni. Amennyiben ez a helyzet akkor a játéknak vége és létrehoz egy Result példányt, amiben gratulál a nyertesnek.

NewGame:

public NewGame():

Konstruktor, inicializálja a menü kinézetét, ezen belül adhatják meg a játékosok a színüket és a nevüket.

NewGameButtonActionListener:

public void actionPerformed(ActionEvent ae):

Felül definiálás, a kívánt működés érdekében. Létrehoz egy NewGame példányt és bezárja az ablakot.

OkButtonActionListener:

public void actionPerformed(ActionEvent ae):

Felül definiálás, a beérkezett esemény hatására bezárja az ablakot.

Player:

public Player(Color c, String name):

Konstruktor, beállítja az adattagokat és a játékosok képének az elérési útvonalát, a paraméterként kapott szín alapján.

public boolean canStep():

Visszatérési értékként adja, hogy a játékos tud-e lépni a paraméterként kapott táblán. Ehhez egy buildert hoz létre, majd megnézi a builder segítségével minden egyes olyan mezőre, amin az adott játékos báránya van, hogy tud-e valamelyikről valamilyen irányban lépni.

PossibleNewFieldBuilder:

public boolean build(ArrayList<Field> board, ArrayList<Point> coordinates, ArrayList<BufferedImage> images):

Törli a koordinátákat és az utolsó elemet leveszi a tábláról, ha ez valamelyik másik listaelemmel megegyezik, akkor visszatér hamissal. Egyébként visszarakja az utolsó elemet. Majd elkéri a pozícióját és létrehoz egy ideiglenes listát, amiben mezők lesznek. Ezt megtölti az tábla utolsó elemének szomszédjaival, illetve azokkal az elemekkel, amelyikek által meg lehet határozni egy négyes csoportot az tábla utolsó elemével együtt. Ezután, ha valamelyik mező az ideiglenes tárolóban megegyezik egy mezővel, ami a táblán van akkor azokat törli a tárolóból. Végül pedig létrehoz egy buildert és ennek segítségével megnézi, melyek azok a mezők, amiken ténylegesen lehet folytatni a játékot, a többit kiszedi. Ha nincs ilyen mező akkor hamissal tér vissza, ha pedig van akkor az utolsó mezőt a tábláról hozzáadja a többi mező szomszédjai közé és fordítva, majd igazzal tér vissza.

PossibleStepBuilder:

public boolean build(ArrayList<Field> board, ArrayList<Point> coordinates, ArrayList<BufferedImage> images):

Törli a koordinátákat, majd leveszi az utolsó elemet a tábláról és megnézi annak melyik irányba melyik a legtávolabbi szomszédja. Amennyiben ez saját maga akkor nem adja hozzá a koordinátákhoz, ha másik mező akkor hozzáadja a pozícióját. Ha a coordinates üres marad akkor hamissal tér vissza, egyébként igazzal.

Result:

public Result(Player p):

Konstruktor, ami megkapja a győztest és elmenti ezt, majd inicializálja a kinézetet.

public void paint(Graphics g):

Az ős egyik metódusának felül definiálása, a kívánt kinézet érdekében. Kirajzolja a nyertes képét és a nevét kiírja.

SaveButtonActionListener:

public void actionPerformed(ActionEvent e):

Az ősosztály függvényének felül definiálása a megfelelő működés érdekében. Szerializálja a játékosokat és a pályát.