Dokumentáció

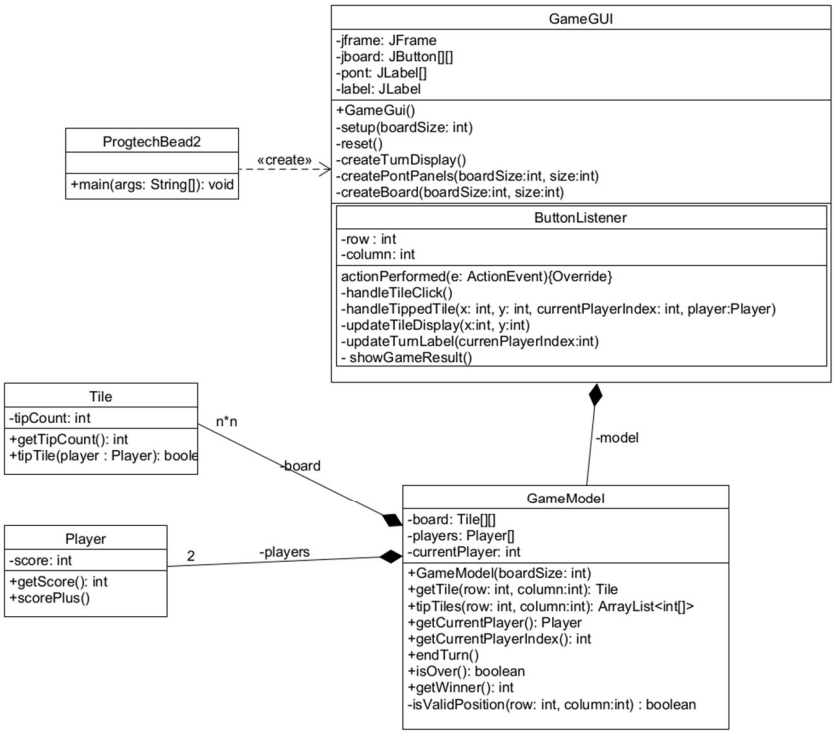
Feladat leírás:

6. 4-es játék

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy $n \times n$ mezőből álló tábla, amelynek mezői 0 és 4 közötti értékeket tartalmaznak. Kezdetben minden mezőn a 0 érték van. Ha a soron következő játékos a tábla egy tetszőleges mezőjét kiválasztja, akkor az adott mezőn és a szomszédos négy mezőn az aktuális érték eggyel nő felfelé, ha az még kisebb, mint 4. Aki a lépésével egy, vagy több mező értékét 4-re állítja, annyi pontot kap, ahány mezővel ezt megtette. A játékosok pontjait folyamatosan számoljuk, és a játékmezőn eltérő színnel jelezzük, hogy azt melyik játékos billentette 4-esre. A játék akkor ér véget, amikor minden mező értéke 4-et mutat. Az győz, akinek ekkor több pontja van.

A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (3×3, 5×5, 7×7), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.

UML diagram:



Metódusok rövid leírása:

reset() : reseteli a játékot. Visszaállítja alaphelyzetbe a játszott tábla mérettel.
createTurnDisplay() : Északi pozícióba jelenik meg, a "Egyes jatekos kore" felirat, ami valószínűleg a játék jelenlegi játékost mutatja.

creatPontPanels(boardSize: int, size:int) :
1. Létrehoz két JLabel objektumot egy tömbben (pont), hogy tárolja a két játékos pontszámát.

2. Létrehoz egy String tömböt (playerLabels), amely tartalmazza a két játékoshoz tartozó címkéket ("Egyes jatekos pontszama:" és "Kettes játékos pontszáma:").

3. Egy ciklusban végigiterál a két játékoson (0 és 1).

4. Minden játékoshoz létrehoz egy JPanel objektumot (pontPanel), beállítja annak méretét egy 160 pixel széles és a boardSize * size pixel magas értékre.

5. Létrehoz egy JLabel objektumot (pont[player]), amely kezdetben "0"-t tartalmaz.

6. Hozzáadja a pontPanel panelhez a megfelelő játékoscímkét (playerLabels[player]) és a játékos pontszámát (pont[player]).

7. A jframe-hez hozzáadja a pontPanel panelt. Az if (player == 0) kifejezés segítségével a "0" indexű játékost az ablak nyugati (bal) részéhez, a "1" indexű játékost pedig az ablak keleti (jobb) részéhez rendeli.

createBoard(int boardSize, int size):
1.Létrehoz egy JPanel objektumot (boardPanel), amelynek elrendezését (Layout) GridLayout-ra állítja, ahol a cellák számát a boardSize változó határozza meg.

2.Létrehoz egy kétdimenziós JButton tömböt (jboard), amely tárolja a játéktábla gombjait.

3.Létrehoz egy GameModel objektumot a boardSize mérettel.

4.Két egymásba ágyazott ciklusban (az egyik a sorokon, a másik az oszlopokon) inicializálja a játékmezőkhez tartozó gombokat (JButton), beállítja háttérszínüket, hozzárendel egy eseménykezelőt (ButtonListener), beállítja a gombok preferált méretét (size méretre), és beállítja a gombok szövegét a GameModel-ből lekért érték alapján.

5.A létrehozott gombokat eltárolja a jboard tömbben.

6.Hozzáadja a boardPanel panelt a jframe középen (BorderLayout.CENTER) az alkalmazás középső részébe.

7.Beállítja az jframe ablak preferált méretét úgy, hogy a tábla mérete és további 300 pixel széles és 50 pixel magas keretet is tartalmazzon.

setUp(): Meghívja a createBoard, createPontPanels, createTurnDisplay metódusokat.

ButtonListener : Egy belső osztályt definiál, amely az ActionListener interfészt implementálja. A handleTileClick metódus elvégzi a következő feladatokat a gombra való kattintás esetén:

1. Lekéri a jelenlegi játékos indexét és játékost.
2. Meghívja a tipTiles.

3. Iterál a "tipped" változó által visszaadott területeken, és az egyes mezők esetén meghívja a handleTippedTile metódust és frissíti a megjelenítést.

4. Frissíti a játékos váltásához tartozó feliratot.
5. Végrehajt egy játékos váltást a model-ben.
6. Ellenőrzi, hogy a játék véget ért-e, és ha igen, megjeleníti az eredményt, majd visszaállítja a játékot.

getTile(int row, int column) : Visszaadja a megadott sor és oszlop indexben lévő Tile-t. Hibát dob ha nem megfelelő a bekért két szám.

getCurrentPlayerIndex(), getCurrentPlayer() : Visszaadja az aktuális player-t és annak indexét

endTurn() : Megváltoztatja a currentPlayer értékét.

isOver() : A metódus ellenőrzi, hogy a játék befejeződött-e.

getWinner() : Visszaadja a győztes játékos indexét.

tipTiles(int row, int column) : Megnézi a lehetséges irányokat, a isValidPosition metódussal megnézi hogy valid e a új pozíciók. Meghívja a board tömbben található megfelelő mezőre a tipTile metódust a jelenlegi játékossal. A tippelés eredményétől függően (sikerült vagy nem) egy tipResult értéket állít be (1 vagy 0). Hozzáad egy új tömböt a tileCoordinates listához, amely tartalmazza az új sor- és oszloppozíciót, valamint a tipResult értékét az adott irányban. Végül visszaadja az ArrayList-et, amely tartalmazza az összes tippelt mező koordinátáit az összes irányban.

getScore() : Visszaadja a score értékét.

scorePlus() : növeli a score értékét.

getTipCount() : Visszaadja a tipCount értékét.

tipTile(Player player) : megnézi hogy a tipCount kisebb e mint 4, ha igen akkor növeli az értékét, majd megvizsgálja, hogy a tipCount egyenlő e 4-gyel, ha igen, akkor a player.scorePlus() metódust meghívja és igazzal tér vissza. Minden más esetben hamissal.

Tesztelés: