**package** hw3;

**import** java.awt.Color;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**import** api.AbstractGame;

**import** api.Generator;

**import** api.Icon;

**import** api.Piece;

**import** api.Position;

**public** **class** BlockAddiction **extends** AbstractGame {

/\*\*

\* Constructs a game with the given height (rows) and width (columns).

\*

\* **@param** height

\* height of the game grid (number of rows)

\* **@param** width

\* width of the game grid (number of columns)

\* **@param** gen

\* instance of the Generator interface to be used for generating pieces in this game

\*/

**public** BlockAddiction(**int** height, **int** width, Generator gen) {

**super**(height, width, gen);

}

/\*\*

\* Constructs a game with the given height (rows) and width (columns). If preFillRows

\* is greater than zero, the given number of rows at the bottom of the grid will be

\* initialized in a checkerboard pattern using randomly generated icons from the given

\* generator.

\*

\* **@param** height

\* height of the game grid (number of rows)

\* **@param** width

\* width of the game grid (number of columns)

\* **@param** gen

\* instance of the Generator interface to be used for generating pieces in this game

\* **@param** preFillRows

\* number of rows at the bottom to be filled with a checkerboard pattern

\*/

**public** BlockAddiction(**int** height, **int** width, BasicGenerator gen, **int** preFillRows) {

**super**(height, width, gen);

**if** (preFillRows > 0) {

**for** (**int** row = 0; row < preFillRows; row++) {

**for** (**int** col = 0; col < width; col++) {

**if** ((row % 2 == 0 && col % 2 == 0) || (row % 2 == 1 && col % 2 == 1)) {

**super**.setBlock(row, col, gen.randomIcon());

}

}

}

}

}

/\*\*

\* Returns a list of locations for all cells that form part of a collapsible set. This

\* list may contain duplicates.

\*

\* **@return**

\* list of locations for positions to be collapsed

\*/

@Override

**public** List<Position> determinePositionsToCollapse() {

List<Position> neighborList = **new** ArrayList<Position>();

List<Position> completeList = **new** ArrayList<Position>();

**for** (**int** row = 0; row < **super**.getHeight(); row++) {

**for** (**int** col = 0; col < **super**.getWidth(); col++) {

**if** (**super**.getIcon(row, col) != **null**) {

Position block = **new** Position(row, col);

Color blockColor = **super**.getIcon(row, col).getColorHint();

// block above

**if** (row - 1 > 0) {

**if** (**super**.getIcon(row - 1, col) != **null** && **super**.getIcon(row - 1, col).getColorHint() == blockColor) {

Position blockAbove = **new** Position(row - 1, col);

neighborList.add(blockAbove);

}

}

// block below

**if** (row + 1 < **super**.getHeight()) {

**if** (**super**.getIcon(row + 1, col) != **null** && **super**.getIcon(row + 1, col).getColorHint() == blockColor) {

Position blockBelow = **new** Position(row + 1, col);

neighborList.add(blockBelow);

}

}

// block to the right

**if** (col + 1 < **super**.getWidth()) {

**if** (**super**.getIcon(row, col + 1) != **null** && **super**.getIcon(row, col + 1).getColorHint() == blockColor) {

Position blockRight = **new** Position(row, col + 1);

neighborList.add(blockRight);

}

}

// block to the left

**if** (col - 1 > 0) {

**if** (**super**.getIcon(row, col - 1) != **null** && **super**.getIcon(row, col - 1).getColorHint() == blockColor) {

Position blockLeft = **new** Position(row, col - 1);

neighborList.add(blockLeft);

}

}

**if** (neighborList.size() >= 2) {

completeList.addAll(neighborList);

completeList.add(block);

neighborList.removeAll(neighborList);

}

}

}

}

List<Position> sortedList = **new** ArrayList<Position>();

**for** (**int** i = 0; i < completeList.size(); i++) {

**if** (sortedList.contains(completeList.get(i))) {

**continue**;

} **else** {

sortedList.add(completeList.get(i));

}

}

Collections.sort(sortedList);

**return** sortedList;

}

}