/\*\* =======================================================================

\* Class:CheckerBoard ExPrj Pg.n/a Author: Yin Linhai

\* Version:001Date:Feb 6, 2014

\*

\* Class, and tester for a checkerboard class which allows you to play checkers. (Does not have promotions yet)

\*

\* Course:Computer Science 201Teacher:Mr Blakey

\* School:Sir Winston Churchill High School, Calgary, Alberta, Canada

\* Language: Java SE 7.0Target Operating System: Java Virtual Machine

\* System:Intel Celeron 3GHz running under Windows 7 IDE: Eclipse 4.2

\*========================================================================\*/

**package** checkerBoard;

**public** **class** CheckerBoard {

//object constructor

**public** CheckerBoard(**int** size, **boolean** right, **boolean** backCapture) {

//to check board size limits

**if**(size>4 && size % 2 == 0) {

rightWhite = right;

backCap = backCapture;

boardSize = size;

checkerBoard = **new** **int**[size][size];

//for near right is white space

**if**(rightWhite) {

//fill in black spaces

**for**(**int** x = 0; x<checkerBoard.length; x+=2) {

**try** {

checkerBoard[(size/2)-1][x+1] = 1;

checkerBoard[(size/2)][x] = 1;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//fill in black pieces

**for**(**int** y = 0; y<((size-2)/2); y++) {

//splits for even, and odd y values to ensure checker pattern

**if**(y % 2 == 0) {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

//print section for white pieces

**for**(**int** y = (size/2)+1; y<size; y++) {

//Splitter like above

**if**(y % 2 == 0) {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

//for near right is dark space

} **else** {

//fill in black spaces

**for**(**int** x = 0; x<checkerBoard.length; x+=2) {

**try** {

checkerBoard[(size/2)][x+1] = 1;

checkerBoard[(size/2)-1][x] = 1;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//fill in black pieces

**for**(**int** y = 0; y<((size-2)/2); y++) {

//splits for even, and odd y values to ensure checker pattern

**if**(y % 2 == 0) {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

//print section for white pieces

**for**(**int** y = (size/2)+1; y<size; y++) {

//Splitter like above

**if**(y % 2 == 0) {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

// try block in case of odd sized board

**try** {

checkerBoard[y][x] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

}

} **else** {

String s = "The board has illegal dimensions";

ArrayIndexOutOfBoundsException e = **new** ArrayIndexOutOfBoundsException(s);

**throw** e;

}

}

**public** **void** configBoard(**int** size, **boolean** right, **boolean** backCapture) {

//to check board size limits

**if**(size>4 && size % 2 == 0) {

rightWhite = right;

backCap = backCapture;

boardSize = size;

checkerBoard = **new** **int**[size][size];

//for near right is white space

**if**(rightWhite) {

//fill in black spaces

**for**(**int** x = 0; x<checkerBoard.length; x+=2) {

**try** {

checkerBoard[(size/2)-1][x+1] = 1;

checkerBoard[(size/2)][x] = 1;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//fill in black pieces

**for**(**int** y = 0; y<((size-2)/2); y++) {

//splits for even, and odd y values to ensure checker pattern

**if**(y % 2 == 0) {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

//print section for white pieces

**for**(**int** y = (size/2)+1; y<size; y++) {

//Splitter like above

**if**(y % 2 == 0) {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

//for near right is dark space

} **else** {

//fill in black spaces

**for**(**int** x = 0; x<checkerBoard.length; x+=2) {

**try** {

checkerBoard[(size/2)][x+1] = 1;

checkerBoard[(size/2)-1][x] = 1;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//fill in black pieces

**for**(**int** y = 0; y<((size-2)/2); y++) {

//splits for even, and odd y values to ensure checker pattern

**if**(y % 2 == 0) {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

//print section for white pieces

**for**(**int** y = (size/2)+1; y<size; y++) {

//Splitter like above

**if**(y % 2 == 0) {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

// try block in case of odd sized board

**try** {

checkerBoard[y][x] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//for odd y values

} **else** {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

}

}

}

} **else** {

String s = "The board has illegal dimensions";

ArrayIndexOutOfBoundsException e = **new** ArrayIndexOutOfBoundsException(s);

**throw** e;

}

}

//resetting the board

**public** **void** resetBoard() {

//for near right is white space

**if**(rightWhite) {

//fill in black spaces

**for**(**int** x = 0; x<checkerBoard.length; x+=2) {

**try** {

checkerBoard[(boardSize/2)-1][x+1] = 1;

checkerBoard[(boardSize/2)][x] = 1;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

**break**;

}

}

//fill in black pieces

**for**(**int** y = 0; y<((boardSize-2)/2); y++) {

//splits for even, and odd y values to ensure checker pattern

**if**(y % 2 == 0) {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

//for odd y values

} **else** {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

}

}

//print section for white pieces

**for**(**int** y = (boardSize/2)+1; y<boardSize; y++) {

//Splitter like above

**if**(y % 2 == 0) {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

//for odd y values

} **else** {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

}

}

//for near right is dark space

} **else** {

//fill in black spaces

**for**(**int** x = 0; x<checkerBoard.length; x+=2) {

**try** {

checkerBoard[(boardSize/2)][x+1] = 1;

checkerBoard[(boardSize/2)-1][x] = 1;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

//fill in black pieces

**for**(**int** y = 0; y<((boardSize-2)/2); y++) {

//splits for even, and odd y values to ensure checker pattern

**if**(y % 2 == 0) {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

//for odd y values

} **else** {

//print for x

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 3;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

}

}

//print section for white pieces

**for**(**int** y = (boardSize/2)+1; y<boardSize; y++) {

//Splitter like above

**if**(y % 2 == 0) {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

//for odd y values

} **else** {

**for**(**int** x = 0; x<checkerBoard[y].length; x+=2) {

//try block in case of odd sized board

**try** {

checkerBoard[y][x+1] = 2;

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("not an even board");

}

}

}

}

}

}

//print out the board

**public** **void** printBoard() {

//go through the x values

**for**(**int** x = 0; x<checkerBoard.length; x++) {

//go through the y values

**for**(**int** y = 0; y<checkerBoard[x].length; y++) {

//print out

System.*out*.print(checkerBoard[x][y] + " ");

}

//to make it pretty

System.*out*.println();

}

System.*out*.println("\n");

}

**public** **void** move(**int** colour, **int** xOriginal, **int** yOriginal, **int** xNew, **int** yNew) {

//check if piece exists

**if**(checkerBoard[yOriginal][xOriginal] == colour) {

//check if it's within array's bounds

**if**(xNew>=0&&xNew<checkerBoard.length) {

**if**(yNew>=0&&yNew<checkerBoard[xNew].length) {

**if**(checkerBoard[yNew][xNew] == 1) {

//check which colour it is

//if red

**if**(colour == 2) {

//check if the move is valid

**if**(((xNew-xOriginal)==1||(xNew-xOriginal)==-1)&&(yNew-yOriginal)==-1) {

checkerBoard[yNew][xNew] = colour;

checkerBoard[yOriginal][xOriginal] = 1;

}

} **else** {

//if black

**if**(colour == 3) {

//check if the move is valid

**if**(((xNew-xOriginal)==1||(xNew-xOriginal)==-1)&&(yNew-yOriginal)==1) {

checkerBoard[yNew][xNew] = colour;

checkerBoard[yOriginal][xOriginal] = 1;

}

}

}

}

}

}

}

}

**public** **void** capturePiece(**int** colour, **int** colourOpposing, **int** xOriginal, **int** yOriginal, **int** xNew, **int** yNew, **int** xOpposing, **int** yOpposing) {

//check if piece exists

**if**(checkerBoard[yOriginal][xOriginal] == colour) {

//check if it's within array's bounds

**if**(xNew>=0&&xNew<checkerBoard.length) {

**if**(yNew>=0&&yNew<checkerBoard[xNew].length) {

//check if opponent exists

**if**(checkerBoard[yOpposing][xOpposing]==colourOpposing) {

//check if the space you're moving to is open

**if**(checkerBoard[yNew][xNew]==1) {

//which colour

//if red

**if**(colour == 2) {

//section if backwards capturing is allowed

**if**(backCap) {

//check if opponent is within reach

**if**(((xOpposing-xOriginal)==1||(xOpposing-xOriginal)==-1)&&((yOpposing-yOriginal)==-1||(yOpposing-yOriginal)==1)) {

//check if jump space is within reach

**if**(((xNew-xOriginal)==2||(xNew-xOriginal)==-2)&&((yNew-yOriginal)==-2||(yNew-yOriginal)==2)) {

checkerBoard[yNew][xNew] = colour;

checkerBoard[yOriginal][xOriginal] = 1;

checkerBoard[yOpposing][xOpposing] = 1;

}

}

} **else** {

//check if opponent is within reach

**if**(((xOpposing-xOriginal)==1||(xOpposing-xOriginal)==-1)&&(yOpposing-yOriginal)==-1) {

//check if jump space is within reach

**if**(((xNew-xOriginal)==2||(xNew-xOriginal)==-2)&&(yNew-yOriginal)==-2) {

checkerBoard[yNew][xNew] = colour;

checkerBoard[yOriginal][xOriginal] = 1;

checkerBoard[yOpposing][xOpposing] = 1;

}

}

}

} **else** {

//for black

**if**(colour == 3) {

//section if backward capturing is allowed

**if**(backCap) {

//check if opponent is within reach

**if**(((xOpposing-xOriginal)==1||(xOpposing-xOriginal)==-1)&&((yOpposing-yOriginal)==-1||(yOpposing-yOriginal)==1)) {

//check if space is within reach

**if**(((xNew-xOriginal)==2||(xNew-xOriginal)==-2)&&((yNew-yOriginal)==-2||(yNew-yOriginal)==2)) {

checkerBoard[yNew][xNew] = colour;

checkerBoard[yOriginal][xOriginal] = 1;

checkerBoard[yOpposing][xOpposing] = 1;

}

}

} **else** {

//check if opponent is within reach

**if**(((xOpposing-xOriginal)==1||(xOpposing-xOriginal)==-1)&&(yOpposing-yOriginal)==1) {

//check if space is within reach

**if**(((xNew-xOriginal)==2||(xNew-xOriginal)==-2)&&(yNew-yOriginal)==2) {

checkerBoard[yNew][xNew] = colour;

checkerBoard[yOriginal][xOriginal] = 1;

checkerBoard[yOpposing][xOpposing] = 1;

}

}

}

}

}

}

}

}

}

}

}

**private** **int**[][] checkerBoard;

**private** **boolean** backCap;

**private** **boolean** rightWhite;

**private** **int** boardSize;

}

**Tester**

**package** checkerBoard;

**public** **class** tester {

**public** **static** **void** main(String[] args) {

CheckerBoard check = **new** CheckerBoard(10, **true**, **true**);

check.printBoard();

check.move(2, 1, 6, 0, 5);

check.printBoard();

check.move(2, 0, 5, 1, 4);

check.printBoard();

check.capturePiece(3, 2, 2, 3, 0, 5, 1, 4);

check.printBoard();

check.resetBoard();

check.printBoard();

**try** {

check.configBoard(7, **false**, **false**);

check.printBoard();

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.*out*.println("Board has impossible dimensions");

}

}

}

**Output:**

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 1 0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0 1 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 1 0 1 0 1 0 1 0 1

2 0 1 0 1 0 1 0 1 0

0 1 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 2 0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0 1 0

0 1 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 3 0 3 0 3 0 3 0 3

3 0 1 0 3 0 3 0 3 0

0 1 0 1 0 1 0 1 0 1

3 0 1 0 1 0 1 0 1 0

0 1 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 3 0 3 0 3 0 3 0 3

3 0 3 0 3 0 3 0 3 0

0 1 0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0 1 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

0 2 0 2 0 2 0 2 0 2

2 0 2 0 2 0 2 0 2 0

Board has impossible dimensions