**import** java.awt.Color;

**import** java.util.Arrays;

**import** api.Cell;

**import** api.Icon;

**import** api.Piece;

**import** api.Position;

**import** hw3.CornerPiece;

**import** hw3.DiagonalPiece;

**import** hw3.IPiece;

**import** hw3.LPiece;

**import** hw3.SnakePiece;

/\*\*

\* Sample test code for Piece transform, shift, and cycle.

\* The comments in the code below specifically refer to

\* testing the LPiece, but you can adapt the code to test any lPiece.

\*

\* THIS CODE WILL NOT COMPILE UNTIL YOU HAVE IMPLEMENTED LPiece.

\*/

**public** **class** ExamplePieceTests

{

/\*\*

\* Helper method to create a lPiece at the given position

\* for testing. EDIT THIS CODE FOR THE lPiece YOU WANT TO TEST.

\*/

**private** **static** Piece makeLPiece(**int** row, **int** col)

{

Icon[] icons = {

**new** Icon(Color.RED),

**new** Icon(Color.GREEN),

**new** Icon(Color.BLUE),

**new** Icon(Color.MAGENTA),

};

**return** **new** LPiece(**new** Position(row, col), icons);

}

**private** **static** Piece makeIPiece(**int** row, **int** col)

{

Icon[] icons = {

**new** Icon(Color.RED),

**new** Icon(Color.GREEN),

**new** Icon(Color.BLUE),

};

**return** **new** IPiece(**new** Position(row, col), icons);

}

**private** **static** Piece makeCornerPiece(**int** row, **int** col)

{

Icon[] icons = {

**new** Icon(Color.RED),

**new** Icon(Color.GREEN),

**new** Icon(Color.BLUE),

};

**return** **new** CornerPiece(**new** Position(row, col), icons);

}

**private** **static** Piece makeDiagonalPiece(**int** row, **int** col)

{

Icon[] icons = {

**new** Icon(Color.RED),

**new** Icon(Color.GREEN),

};

**return** **new** DiagonalPiece(**new** Position(row, col), icons);

}

**private** **static** Piece makeSnakePiece(**int** row, **int** col)

{

Icon[] icons = {

**new** Icon(Color.RED),

**new** Icon(Color.GREEN),

**new** Icon(Color.BLUE),

**new** Icon(Color.MAGENTA),

};

**return** **new** SnakePiece(**new** Position(row, col), icons);

}

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

{

System.out.println("--------------------------------------");

System.out.println("LPIECE TESTING");

// As an example, start with a lPiece whose bounding square

// is located (10, 10)

Piece lPiece = makeLPiece(10, 10);

// expected (10, 10)

System.out.println("Position:");

System.out.println(lPiece.getPosition());

System.out.println();

// if testing the lPiece shown above in makePiece

// then expected [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

Cell[] lResult = lPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(lResult));

System.out.println();

// make sure you're really making a deep copy - modify the lResult

// and make sure the original didn't change

lResult[0].setRowCol(48, 137);

System.out.println("Making sure original positions don't change.");

System.out.println(Arrays.toString(lPiece.getCells()));

// expected [R(10, 10), G(10, 11), B(11, 11), M(12, 11)]

System.out.println("Absolute positions");

System.out.println(Arrays.toString(lPiece.getCellsAbsolute()));

System.out.println();

lPiece.shiftDown();

// expected (11, 10)

System.out.println("Position after shiftDown:");

System.out.println(lPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

System.out.println("Relative positions after shiftDown (should be unchanged):");

System.out.println(Arrays.toString(lPiece.getCells()));

System.out.println();

// expected [R(11, 10), G(11, 11), B(12, 11), M(13, 11)]

System.out.println("Absolute positions after shiftDown");

System.out.println(Arrays.toString(lPiece.getCellsAbsolute()));

System.out.println();

lPiece.shiftRight();

lPiece.shiftRight();

// expected (11, 12)

System.out.println("Position after shiftRight twice:");

System.out.println(lPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

System.out.println("Relative positions after shiftRight (should be unchanged):");

System.out.println(Arrays.toString(lPiece.getCells()));

System.out.println();

// expected [R(11, 12), G(11, 13), B(12, 13), M(13, 13)]

System.out.println("Absolute positions after shiftRight");

System.out.println(Arrays.toString(lPiece.getCellsAbsolute()));

System.out.println();

// transform for L-lPiece should flip it across vertical centerline

lPiece.transform();

// expected (11, 12)

System.out.println("Position after transform (should be unchanged):");

System.out.println(lPiece.getPosition());

System.out.println();

// expected [R(0, 2), G(0, 1), B(1, 1), M(2, 1)]

System.out.println("Relative positions after transform:");

System.out.println(Arrays.toString(lPiece.getCells()));

System.out.println();

// expected [R(11, 14), G(11, 13), B(12, 13), M(13, 13)]

System.out.println("Absolute positions after transform");

System.out.println(Arrays.toString(lPiece.getCellsAbsolute()));

System.out.println();

// try cycling, expected:

// [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

// [M(0, 0), R(0, 1), G(1, 1), B(2, 1)]

// [B(0, 0), M(0, 1), R(1, 1), G(2, 1)]

// [G(0, 0), B(0, 1), M(1, 1), R(2, 1)]

// [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

System.out.println("make new lPiece, cycle four times");

lPiece = makeLPiece(0, 0);

System.out.println(Arrays.toString(lPiece.getCells()));

lPiece.cycle();

System.out.println(Arrays.toString(lPiece.getCells()));

lPiece.cycle();

System.out.println(Arrays.toString(lPiece.getCells()));

lPiece.cycle();

System.out.println(Arrays.toString(lPiece.getCells()));

lPiece.cycle();

System.out.println(Arrays.toString(lPiece.getCells()));

System.out.println();

// check the clone method...

System.out.println("CHECKING CLONE METHOD");

lPiece = makeLPiece(0, 0);

Piece lCopy = lPiece.clone();

System.out.println(lCopy.getClass()); // should be LlPiece

lCopy.shiftDown();

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

System.out.println("after moving the copy, original should be unmodified:");

System.out.println(Arrays.toString(lPiece.getCellsAbsolute()));

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

// [R(2, 0), G(2, 1), B(3, 1), M(4, 1)]

lCopy.shiftDown();

System.out.println("After moving the copy, original should be unmodified");

System.out.println("Og: " + Arrays.toString(lPiece.getCellsAbsolute()));

System.out.println("Copy: " + Arrays.toString(lCopy.getCellsAbsolute()));

System.out.println();

// after transforming the copy, original should be unmodified

// expected [R(0, 0), G(0, 1), B(1, 1), M(2, 1)]

// [R(2, 2), G(2, 1), B(3, 1), M(4, 1)]

lCopy.transform();

System.out.println("After transforming the copy, original should be unmodified");

System.out.println("Og: " + Arrays.toString(lPiece.getCellsAbsolute()));

System.out.println("Copy: " + Arrays.toString(lCopy.getCellsAbsolute()));

System.out.println();

// cycle followed by transform should work

// expected [M(0, 2), R(0, 1), G(1, 1), B(2, 1)]

lPiece = makeLPiece(0, 0);

lPiece.cycle();

lPiece.transform();

System.out.println("cycle followed by transform should work");

System.out.println("Og: " + Arrays.toString(lPiece.getCellsAbsolute()));

// the copy should still be the same

// expected [R(2, 2), G(2, 1), B(3, 1), M(4, 1)]

System.out.println("the copy should still be the same");

System.out.println("Copy: " + Arrays.toString(lCopy.getCellsAbsolute()));

System.out.println("LPIECE TESTING");

System.out.println("--------------------------------------");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println("--------------------------------------");

System.out.println("IPIECE TESTING");

// As an example, start with a IPiece whose bounding square

// is located (10, 10)

Piece iPiece = makeIPiece(10, 10);

// expected [R(0, 1), G(1, 1), B(2, 1)]

Cell[] iResult = iPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(iResult));

System.out.println();

// expected (10, 10)

System.out.println("Position:");

System.out.println(iPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(1, 1), B(2, 1)]

Cell[] iResults = iPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(iResults));

System.out.println();

// make sure you're really making a deep copy - modify the lResult

// and make sure the original didn't change

// expected [R(0, 1), G(1, 1), B(2, 1)]

iResults[0].setRowCol(48, 137);

System.out.println("Making sure original positions don't change.");

System.out.println(Arrays.toString(iPiece.getCells()));

System.out.println();

// expected [R(10, 11), G(11, 11), B(12, 11)]

System.out.println("Absolute positions");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

System.out.println();

iPiece.shiftDown();

// expected (11, 10)

System.out.println("Position after shiftDown:");

System.out.println(iPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(1, 1), B(2, 1)]

System.out.println("Relative positions after shiftDown (should be unchanged):");

System.out.println(Arrays.toString(iPiece.getCells()));

System.out.println();

// expected [R(11, 11), G(12, 11), B(13, 11)]

System.out.println("Absolute positions after shiftDown");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

System.out.println();

iPiece.shiftRight();

iPiece.shiftRight();

// expected (11, 12)

System.out.println("Position after shiftRight twice:");

System.out.println(iPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(1, 1), B(2, 1)]

System.out.println("Relative positions after shiftRight (should be unchanged):");

System.out.println(Arrays.toString(iPiece.getCells()));

System.out.println();

// expected [R(11, 13), G(12, 13), B(13, 13)]

System.out.println("Absolute positions after shiftRight");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

System.out.println();

// transform for iPiece should do nothing

iPiece.transform();

// expected (11, 12)

System.out.println("Position after transform (should be unchanged):");

System.out.println(iPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(1, 1), B(2, 1)]

System.out.println("Relative positions after transform (unchanged from relative positions):");

System.out.println(Arrays.toString(iPiece.getCells()));

System.out.println();

// expected [R(11, 13), G(12, 13), B(13, 13)]

System.out.println("Absolute positions after transform (unchanged from absol positions)");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

System.out.println();

// try cycling, expected:

// [R(0, 1), G(1, 1), B(2, 1)]

// [B(0, 1), R(1, 1), G(2, 1)]

// [G(0, 1), B(1, 1), R(2, 1)]

// [R(0, 1), G(1, 1), B(2, 1)]

System.out.println("make new iPiece, cycle three times");

iPiece = makeIPiece(0, 0);

System.out.println(Arrays.toString(iPiece.getCells()));

iPiece.cycle();

System.out.println(Arrays.toString(iPiece.getCells()));

iPiece.cycle();

System.out.println(Arrays.toString(iPiece.getCells()));

iPiece.cycle();

System.out.println(Arrays.toString(iPiece.getCells()));

System.out.println();

// check the clone method...

System.out.println("CHECKING CLONE METHOD");

iPiece = makeIPiece(0, 0);

Piece iCopy = iPiece.clone();

System.out.println("New piece position = 0,0, new copy: ");

System.out.println("Copy: " + iCopy.getClass());

iCopy.shiftDown();

// after moving the copy, original should be unmodified

// expected [R(0, 1), G(1, 1), B(2, 1)]

System.out.println("Original shouldn't be modified after shifting copy: ");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

// after moving the copy, original should be unmodified

// expected [R(0, 1), G(1, 1), B(2, 1)]

// [R(2, 1), G(3, 1), B(4, 1)]

iCopy.shiftDown();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after two shifts down: ");

System.out.println(Arrays.toString(iCopy.getCellsAbsolute()));

System.out.println();

// after transforming the copy, original should be unmodified

// expected [R(0, 1), G(1, 1), B(2, 1)]

// [R(2, 1), G(3, 1), B(4, 1)]

iCopy.transform();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after transforming (no change): ");

System.out.println(Arrays.toString(iCopy.getCellsAbsolute()));

System.out.println();

// cycle followed by transform should work

// [B(0, 1), R(1, 1), G(2, 1)]

iPiece = makeIPiece(0, 0);

iPiece.cycle();

iPiece.transform();

System.out.println("New piece position = 0,0, same copy: ");

System.out.println("Absolute position of original after cycling once: ");

System.out.println(Arrays.toString(iPiece.getCellsAbsolute()));

// the copy should still be the same

// [R(2, 1), G(3, 1), B(4, 1)]

System.out.println("Absolute position of copy (no change): ");

System.out.println(Arrays.toString(iCopy.getCellsAbsolute()));

System.out.println("IPIECE TESTING");

System.out.println("--------------------------------------");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println("--------------------------------------");

System.out.println("CORNERPIECE TESTING");

// As an example, start with a CornerPiece whose bounding square

// is located (10, 10)

Piece cPiece = makeCornerPiece(10, 10);

// expected [R(0, 0), G(1, 0), B(1, 1)]

Cell[] cResult = cPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(cResult));

System.out.println();

// expected (10, 10)

System.out.println("Position:");

System.out.println(cPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 0), B(1, 1)]

Cell[] cResults = cPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(cResults));

System.out.println();

// make sure you're really making a deep copy - modify the lResult

// and make sure the original didn't change

// expected [R(0, 0), G(1, 0), B(1, 1)]

cResults[0].setRowCol(48, 137);

System.out.println("Making sure original positions don't change.");

System.out.println(Arrays.toString(cPiece.getCells()));

System.out.println();

// expected [R(10, 10), G(11, 10), B(11, 11)]

System.out.println("Absolute positions");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

System.out.println();

cPiece.shiftDown();

// expected (11, 10)

System.out.println("Position after shiftDown:");

System.out.println(cPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 0), B(1, 1)]

System.out.println("Relative positions after shiftDown (should be unchanged):");

System.out.println(Arrays.toString(cPiece.getCells()));

System.out.println();

// expected [R(11, 10), G(12, 10), B(12, 11)]

System.out.println("Absolute positions after shiftDown");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

System.out.println();

cPiece.shiftRight();

cPiece.shiftRight();

// expected (11, 12)

System.out.println("Position after shiftRight twice:");

System.out.println(cPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 0), B(1, 1)]

System.out.println("Relative positions after shiftRight (should be unchanged):");

System.out.println(Arrays.toString(cPiece.getCells()));

System.out.println();

// expected [R(11, 12), G(12, 12), B(12, 13)]

System.out.println("Absolute positions after shiftRight");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

System.out.println();

// transform for cPiece

cPiece.transform();

// expected (11, 12)

System.out.println("Position after transform (should be unchanged):");

System.out.println(cPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(0, 0), B(1, 0)]

System.out.println("Relative positions after transform):");

System.out.println(Arrays.toString(cPiece.getCells()));

System.out.println();

// expected [R(11, 13), G(11, 12), B(12, 12)]

System.out.println("Absolute positions after transform");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

System.out.println();

// try cycling, expected:

// [R(0, 0), G(1, 0), B(1, 1)]

// [B(0, 0), R(1, 0), G(1, 1)]

// [G(0, 0), B(1, 0), R(1, 1)]

// [R(0, 0), G(1, 0), B(1, 1)]

System.out.println("make new cPiece, cycle three times");

cPiece = makeCornerPiece(0, 0);

System.out.println(Arrays.toString(cPiece.getCells()));

cPiece.cycle();

System.out.println(Arrays.toString(cPiece.getCells()));

cPiece.cycle();

System.out.println(Arrays.toString(cPiece.getCells()));

cPiece.cycle();

System.out.println(Arrays.toString(cPiece.getCells()));

System.out.println();

// check the clone method...

System.out.println("CHECKING CLONE METHOD");

cPiece = makeCornerPiece(0, 0);

Piece cCopy = cPiece.clone();

System.out.println("New piece position = 0,0, new copy: ");

System.out.println("Copy: " + cCopy.getClass());

System.out.println();

cCopy.shiftDown();

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(1, 0), B(1, 1)]

System.out.println("Original shouldn't be modified after shifting copy: ");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(1, 0), B(1, 1)]

// expected [R(2, 0), G(3, 0), B(3, 1)]

cCopy.shiftDown();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after two shifts down: ");

System.out.println(Arrays.toString(cCopy.getCellsAbsolute()));

System.out.println();

// after transforming the copy, original should be unmodified

// expected [R(0, 0), G(1, 0), B(1, 1)]

// expected [R(2, 1), G(2, 0), B(3, 0)]

cCopy.transform();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after transforming: ");

System.out.println(Arrays.toString(cCopy.getCellsAbsolute()));

System.out.println();

// cycle followed by transform should work

// expected [B(0, 1), R(0, 0), G(1, 0)]

cPiece = makeCornerPiece(0, 0);

cPiece.cycle();

cPiece.transform();

System.out.println("New piece position = 0,0, same copy: ");

System.out.println("Absolute position of original after cycling once: ");

System.out.println(Arrays.toString(cPiece.getCellsAbsolute()));

// the copy should still be the same

// expected [R(2, 1), G(2, 0), B(3, 0)]

System.out.println("Absolute position of copy (no change): ");

System.out.println(Arrays.toString(cCopy.getCellsAbsolute()));

System.out.println();

System.out.println("CORNERPIECE TESTING");

System.out.println("--------------------------------------");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println("--------------------------------------");

System.out.println("DIAGONALPIECE TESTING");

// As an example, start with a DiagonalPiece whose bounding square

// is located (10, 10)

Piece dPiece = makeDiagonalPiece(10, 10);

// expected [R(0, 0), G(1, 1)]

Cell[] dResult = dPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(dResult));

System.out.println();

// expected (10, 10)

System.out.println("Position:");

System.out.println(dPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 1)]

Cell[] dResults = dPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(dResults));

System.out.println();

// make sure you're really making a deep copy - modify the lResult

// and make sure the original didn't change

// expected [R(0, 0), G(1, 1)]

dResults[0].setRowCol(48, 137);

System.out.println("Making sure original positions don't change.");

System.out.println(Arrays.toString(dPiece.getCells()));

System.out.println();

// expected [R(10, 10), G(11, 11)]

System.out.println("Absolute positions");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

System.out.println();

dPiece.shiftDown();

// expected (11, 10)

System.out.println("Position after shiftDown:");

System.out.println(dPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 1)]

System.out.println("Relative positions after shiftDown (should be unchanged):");

System.out.println(Arrays.toString(dPiece.getCells()));

System.out.println();

// expected [R(11, 10), G(12, 11)]

System.out.println("Absolute positions after shiftDown");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

System.out.println();

dPiece.shiftRight();

dPiece.shiftRight();

// expected (11, 12)

System.out.println("Position after shiftRight twice:");

System.out.println(dPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 1)]

System.out.println("Relative positions after shiftRight (should be unchanged):");

System.out.println(Arrays.toString(dPiece.getCells()));

System.out.println();

// expected [R(11, 12), G(12, 13)]

System.out.println("Absolute positions after shiftRight");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

System.out.println();

// transform for cPiece

dPiece.transform();

// expected (11, 12)

System.out.println("Position after transform (should be unchanged):");

System.out.println(dPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(1, 0)]

System.out.println("Relative positions after transform):");

System.out.println(Arrays.toString(dPiece.getCells()));

System.out.println();

// expected [R(11, 13), G(12, 12)]

System.out.println("Absolute positions after transform");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

System.out.println();

// try cycling, expected:

// [R(0, 0), G(1, 1)] original

// [G(0, 0), R(1, 1)]

// [R(0, 0), G(1, 1)]

System.out.println("make new dPiece, cycle two times");

dPiece = makeDiagonalPiece(0, 0);

System.out.println(Arrays.toString(dPiece.getCells()));

dPiece.cycle();

System.out.println(Arrays.toString(dPiece.getCells()));

dPiece.cycle();

System.out.println(Arrays.toString(dPiece.getCells()));

System.out.println();

// check the clone method...

System.out.println("CHECKING CLONE METHOD");

dPiece = makeDiagonalPiece(0, 0);

Piece dCopy = dPiece.clone();

System.out.println("New piece position = 0,0, new copy: ");

System.out.println("Copy: " + dCopy.getClass());

System.out.println();

dCopy.shiftDown();

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(1, 1)]

System.out.println("Original shouldn't be modified after shifting copy: ");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(1, 1)]

// expected [R(2, 0), G(3, 1)]

dCopy.shiftDown();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after two shifts down: ");

System.out.println(Arrays.toString(dCopy.getCellsAbsolute()));

System.out.println();

// after transforming the copy, original should be unmodified

// expected [R(0, 0), G(1, 1)]

// expected [R(2, 1), G(3, 0)]

dCopy.transform();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after transforming: ");

System.out.println(Arrays.toString(dCopy.getCellsAbsolute()));

System.out.println();

// cycle followed by transform should work

// expected [G(0, 1), R(1, 0)]

dPiece = makeDiagonalPiece(0, 0);

dPiece.cycle();

dPiece.transform();

System.out.println("New piece position = 0,0, same copy: ");

System.out.println("Absolute position of original after cycling once: ");

System.out.println(Arrays.toString(dPiece.getCellsAbsolute()));

// the copy should still be the same

// expected [R(2, 1), G(3, 0)]

System.out.println("Absolute position of copy (no change): ");

System.out.println(Arrays.toString(dCopy.getCellsAbsolute()));

System.out.println();

System.out.println("DIAGONALPIECE TESTING");

System.out.println("--------------------------------------");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println(" ");

System.out.println("--------------------------------------");

System.out.println("SNAKEPIECE TESTING");

// As an example, start with a DiagonalPiece whose bounding square

// is located (10, 10)

Piece sPiece = makeSnakePiece(10, 10);

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

Cell[] sResult = sPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(sResult));

System.out.println();

// expected (10, 10)

System.out.println("Position:");

System.out.println(sPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

Cell[] sResults = sPiece.getCells();

System.out.println("Relative positions in bounding square:");

System.out.println(Arrays.toString(sResults));

System.out.println();

// make sure you're really making a deep copy - modify the lResult

// and make sure the original didn't change

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

sResults[0].setRowCol(48, 137);

System.out.println("Making sure original positions don't change.");

System.out.println(Arrays.toString(sPiece.getCells()));

System.out.println();

// expected [R(10, 10), G(11, 10), B(11, 11), M(11, 12)]

System.out.println("Absolute positions");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

System.out.println();

sPiece.shiftDown();

// expected (11, 10)

System.out.println("Position after shiftDown:");

System.out.println(sPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

System.out.println("Relative positions after shiftDown (should be unchanged):");

System.out.println(Arrays.toString(sPiece.getCells()));

System.out.println();

// expected [R(11, 10), G(12, 10), B(12, 11), M(12, 12)]

System.out.println("Absolute positions after shiftDown");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

System.out.println();

sPiece.shiftRight();

sPiece.shiftRight();

// expected (11, 12)

System.out.println("Position after shiftRight twice:");

System.out.println(sPiece.getPosition());

System.out.println();

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

System.out.println("Relative positions after shiftRight (should be unchanged):");

System.out.println(Arrays.toString(sPiece.getCells()));

System.out.println();

// expected [R(11, 12), G(12, 12), B(12, 13), M(12, 14)]

System.out.println("Absolute positions after shiftRight");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

System.out.println();

// transform for sPiece

sPiece.transform();

// expected (11, 12)

System.out.println("Position after transform (should be unchanged):");

System.out.println(sPiece.getPosition());

System.out.println();

// expected [R(0, 1), G(0, 0), B(1, 0), M(1, 1)]

System.out.println("Relative positions after transform):");

System.out.println(Arrays.toString(sPiece.getCells()));

System.out.println();

// expected [R(11, 13), G(11, 12), B(12, 12), M(12, 13)]

System.out.println("Absolute positions after transform");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

System.out.println();

// try transform for all of sPiece

// expected:

// [R(0, 0), G(1, 0), B(1, 1), M(1, 2)] ORIGINAL

// [R(0, 1), G(0, 0), B(1, 0), M(1, 1)] 1

// [R(0, 2), G(0, 1), B(0, 0), M(1, 0)] 2

// [R(1, 2), G(0, 2), B(0, 1), M(0, 0)] 3

// [R(1, 1), G(1, 2), B(0, 2), M(0, 1)] 4

// [R(1, 0), G(1, 1), B(1, 2), M(0, 2)] 5

// [R(2, 0), G(1, 0), B(1, 1), M(1, 2)] 6

// [R(2, 1), G(2, 0), B(1, 0), M(1, 1)] 7

// [R(2, 2), G(2, 1), B(2, 0), M(1, 0)] 8

// [R(1, 2), G(2, 2), B(2, 1), M(2, 0)] 9

// [R(1, 1), G(1, 2), B(2, 2), M(2, 1)] 10

// [R(0, 1), G(1, 1), B(1, 2), M(2, 2)] 11

// [R(0, 0), G(1, 0), B(1, 1), M(1, 2)] 1

System.out.println("make new sPiece, transform 12 times");

sPiece = makeSnakePiece(0, 0);

System.out.println("OG: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("1: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("2: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("3: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("4: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("5: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("6: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("7: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("8: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("9: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("10: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("11: " + Arrays.toString(sPiece.getCells()));

sPiece.transform();

System.out.println("Loops back to original position: " + Arrays.toString(sPiece.getCells()));

System.out.println();

// try cycling, expected:

// [R(0, 0), G(1, 0), B(1, 1), M(1, 2)] original

// [M(0, 0), R(1, 0), G(1, 1), B(1, 2)]

// [B(0, 0), M(1, 0), R(1, 1), G(1, 2)]

// [G(0, 0), B(1, 0), M(1, 1), R(1, 2)]

// [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

System.out.println("make new sPiece, cycle 4 times");

sPiece = makeSnakePiece(0, 0);

System.out.println(Arrays.toString(sPiece.getCells()));

sPiece.cycle();

System.out.println(Arrays.toString(sPiece.getCells()));

sPiece.cycle();

System.out.println(Arrays.toString(sPiece.getCells()));

sPiece.cycle();

System.out.println(Arrays.toString(sPiece.getCells()));

sPiece.cycle();

System.out.println(Arrays.toString(sPiece.getCells()));

System.out.println();

// check the clone method...

System.out.println("CHECKING CLONE METHOD");

sPiece = makeSnakePiece(0, 0);

Piece sCopy = sPiece.clone();

System.out.println("New piece position = 0,0, new copy: ");

System.out.println("Copy: " + sCopy.getClass());

System.out.println();

sCopy.shiftDown();

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

System.out.println("Original shouldn't be modified after shifting copy: ");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

// after moving the copy, original should be unmodified

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

// expected [R(1, 0), G(2, 0), B(2, 1), M(2, 2)]

sCopy.shiftDown();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after two shifts down: ");

System.out.println(Arrays.toString(sCopy.getCellsAbsolute()));

System.out.println();

// after transforming the copy, original should be unmodified

// expected [R(0, 0), G(1, 0), B(1, 1), M(1, 2)]

// expected [R(2, 1), G(2, 0), B(3, 0), M(3, 1)]

sCopy.transform();

System.out.println("Absolute position of original: ");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

System.out.println("Absolute position of copy after transforming: ");

System.out.println(Arrays.toString(sCopy.getCellsAbsolute()));

System.out.println();

// cycle followed by transform should work

// expected [M(0, 1), R(0, 0), G(1, 0), B(1, 0)]

sPiece = makeSnakePiece(0, 0);

sPiece.cycle();

sPiece.transform();

System.out.println("New piece position = 0,0, same copy: ");

System.out.println("Absolute position of original after cycling once: ");

System.out.println(Arrays.toString(sPiece.getCellsAbsolute()));

// the copy should still be the same

// expected [R(1, 1), G(1, 0), B(2, 0), M(2, 1)]

System.out.println("Absolute position of copy (no change): ");

System.out.println(Arrays.toString(sCopy.getCellsAbsolute()));

System.out.println();

System.out.println("SNAKEPIECE TESTING");

System.out.println("--------------------------------------");

System.out.println(" ");

}

}