**package** hw3;

**import** api.Icon;

**import** api.Cell;

**import** api.Position;

**import** api.Piece;

/\*\*

\* Abstract superclass for implementations of the Piece interface.

\*

\* **@author** Halle N

\*/

**public** **abstract** **class** AbstractPiece **implements** Piece {

/\*\*

\* The position of the piece.

\*/

**private** Position position;

/\*\*

\* The cells in this piece.

\*/

**private** Cell[] cells;

/\*\*

\* Constructs a piece with the given position. Subclasses extending this class

\* MUST call setCells to initialize initial cell positions and icons.

\*

\* **@param** initialPosition

\* initial position for upper-left corner of bounding box

\*/

**protected** AbstractPiece(Position initialPosition) {

position = initialPosition;

}

/\*\*

\* Constructs a piece with the given position. Subclasses extending this class

\* MUST call setCells to initialize initial cell positions and icons.

\*

\* **@param** initialPosition

\* initial position for upper-left corner of bounding box

\* **@param** initialIcons

\* initial icons for this piece

\*/

**protected** AbstractPiece(Position initialPosition, Icon[] initialIcons) {

position = initialPosition;

}

/\*\*

\* Returns the position of this piece (upper-left corner of its bounding box).

\*

\* **@return**

\* position of this piece

\*/

@Override

**public** Position getPosition() {

**return** position;

}

/\*\*

\* Sets the cells in this piece, making a deep copy of the given array.

\*

\* **@param** givenCells

\* new cells for this piece

\*/

@Override

**public** **void** setCells(Cell[] givenCells) {

// deep copy the given array

cells = **new** Cell[givenCells.length];

**for** (**int** i = 0; i < givenCells.length; i++) {

cells[i] = givenCells[i];

}

}

/\*\*

\* Returns a deep copy of the Cell objects in this piece. The cell

\* positions are relative to the upper-left corner of its bounding box.

\*

\* **@return**

\* copy of the cells in this piece

\*/

@Override

**public** Cell[] getCells() {

// deep copy this object's cell array

Cell[] copy = **new** Cell[cells.length];

**for** (**int** i = 0; i < cells.length; i++) {

copy[i] = **new** Cell(cells[i]);

}

**return** copy;

}

/\*\*

\* Returns a new array of Cell objects representing the icons in this piece

\* with their absolute positions (relative positions plus position of bounding

\* box).

\*

\* **@return**

\* copy of the cells in this piece with absolute positions

\*/

@Override

**public** Cell[] getCellsAbsolute() {

Cell[] ret = **new** Cell[cells.length];

**for** (**int** i = 0; i < cells.length; i++) {

**int** row = cells[i].getRow() + position.row();

**int** col = cells[i].getCol() + position.col();

Icon b = cells[i].getIcon();

ret[i] = **new** Cell(b, **new** Position(row, col));

}

**return** ret;

}

/\*\*

\* Shifts the position of this piece down (increasing the row) by one. No bounds

\* checking is done.

\*/

@Override

**public** **void** shiftDown() {

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

}

/\*\*

\* Shifts the position of this piece left (decreasing the column) by one. No

\* bounds checking is done.

\*/

@Override

**public** **void** shiftLeft() {

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

}

/\*\*

\* Shifts the position of this piece right (increasing the column) by one. No

\* bounds checking is done.

\*/

@Override

**public** **void** shiftRight() {

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

}

/\*\*

\* Cycles the icons within the cells of this piece. Each icon is shifted forward

\* to the next cell (in the original ordering of the cells). The last icon wraps

\* around to the first cell.

\*/

@Override

**public** **void** cycle() {

Icon last = cells[cells.length - 1].getIcon();

**for** (**int** i = cells.length - 1; i > 0; i--) {

cells[i].setIcon(cells[i - 1].getIcon());

}

cells[0].setIcon(last);

}

/\*\*

\* Returns a deep copy of this object having the correct runtime type.

\*

\* **@return**

\* deep copy of this object

\*/

@Override

**public** Piece clone() {

**try**

{

// call the Object clone() method to create a shallow copy

AbstractPiece s = (AbstractPiece) **super**.clone();

// then make it into a deep copy (note there is no need to copy the position,

// since Position is immutable, but we have to deep-copy the cell array

// by making new Cell objects

s.cells = **new** Cell[cells.length];

**for** (**int** i = 0; i < cells.length; ++i)

{

s.cells[i] = **new** Cell(cells[i]);

}

**return** s;

}

**catch** (CloneNotSupportedException e)

{

// can't happen, since we know the superclass is cloneable

**return** **null**;

}

}

}