/\*\*

\* This class represents a simple two-dimensional map composed of square cells.

\* Each cell specifies the cost of traversing that cell.

\*\*/

public class Map2D

{

/\*\* The width of the map. \*\*/

private int width;

/\*\* The height of the map. \*\*/

private int height;

/\*\*

\* The actual map data that the pathfinding algorithm needs to navigate.

\*\*/

private int[][] cells;

/\*\* The starting location for performing the A\* pathfinding. \*\*/

private Location start;

/\*\* The ending location for performing the A\* pathfinding. \*\*/

private Location finish;

/\*\* Creates a new 2D map, with the specified width and height. \*\*/

public Map2D(int width, int height)

{

if (width <= 0 || height <= 0)

{

throw new IllegalArgumentException(

"width and height must be positive values; got " + width +

"x" + height);

}

this.width = width;

this.height = height;

cells = new int[width][height];

// Make up some coordinates for start and finish.

start = new Location(0, height / 2);

finish = new Location(width - 1, height / 2);

}

/\*\*

\* This helper method checks the specified coordinates to see if they are

\* within the map's boundaries. If the coordinates are not within the map

\* then the method throws an <code>IllegalArgumentException</code>.

\*\*/

private void checkCoords(int x, int y)

{

if (x < 0 || x > width)

{

throw new IllegalArgumentException("x must be in range [0, " +

width + "), got " + x);

}

if (y < 0 || y > height)

{

throw new IllegalArgumentException("y must be in range [0, " +

height + "), got " + y);

}

}

/\*\* Returns the width of the map. \*\*/

public int getWidth()

{

return width;

}

/\*\* Returns the height of the map. \*\*/

public int getHeight()

{

return height;

}

/\*\*

\* Returns true if the specified coordinates are contained within the map

\* area.

\*\*/

public boolean contains(int x, int y)

{

return (x >= 0 && x < width && y >= 0 && y < height);

}

/\*\* Returns true if the location is contained within the map area. \*\*/

public boolean contains(Location loc)

{

return contains(loc.xCoord, loc.yCoord);

}

/\*\* Returns the stored cost value for the specified cell. \*\*/

public int getCellValue(int x, int y)

{

checkCoords(x, y);

return cells[x][y];

}

/\*\* Returns the stored cost value for the specified cell. \*\*/

public int getCellValue(Location loc)

{

return getCellValue(loc.xCoord, loc.yCoord);

}

/\*\* Sets the cost value for the specified cell. \*\*/

public void setCellValue(int x, int y, int value)

{

checkCoords(x, y);

cells[x][y] = value;

}

/\*\*

\* Returns the starting location for the map. This is where the generated

\* path will begin from.

\*\*/

public Location getStart()

{

return start;

}

/\*\*

\* Sets the starting location for the map. This is where the generated path

\* will begin from.

\*\*/

public void setStart(Location loc)

{

if (loc == null)

throw new NullPointerException("loc cannot be null");

start = loc;

}

/\*\*

\* Returns the ending location for the map. This is where the generated

\* path will terminate.

\*\*/

public Location getFinish()

{

return finish;

}

/\*\*

\* Sets the ending location for the map. This is where the generated path

\* will terminate.

\*\*/

public void setFinish(Location loc)

{

if (loc == null)

throw new NullPointerException("loc cannot be null");

finish = loc;

}

}