Project 5: Make a square

Goal of the Project:

To create a program that fits 4 Tetris pieces in a 4x4 square board utilizing multithreading, It will first choose a shape to start with, then create a number of threads that match the available rotations for that shape, then recursively pick the next shape and create enough threads for that shape too. What every thread does is that it tries to fit the shape to the board and translate it across the board till a vacant place is found. The program will attempt to find a solution (or none, if none exists) for the given shapes.

Problem Definition:

Make a square with size 4X4 by using 4 pieces. The pieces can be rotated, and all pieces should be used to form a square. Example sets of pieces. There may be more than one possible solution for a set of pieces, and not every arrangement will work even with a set for which a solution can be found. Examples using the above set of pieces…

Input:

The first line contains a number of pieces. Each piece is then specified by listing a single line with two integers, the number of rows and columns in the piece, followed by one or more lines which specify the shape of the piece. The shape specification consists of 0 or 1 characters, with 1 character indicating the solid shape of the puzzle (the 0 characters are merely placeholders). For example, piece A above would be specified as follows:

2 3

111

101

Output:

Your program should report all solutions in the format shown by the examples below. A 4x4 square should be created, with each piece occupying its location in the solution. The solid portions of piece #1 should be replaced with `1' characters, of piece #2 with `2' characters.

Sample output that represents the figure above could be:

1112

1412

3422

3442

For cases which have no possible solution simply report "No solution possible".

You must provide many sample inputs (many text files) during the discussion time, to test your project on many samples. Each text file represents one problem to be solved.

Algorithm:

The algorithm, done by recursively running a new thread using the executor, and passing a copy of the current board, next shape to get fit, a list of all shapes, a list that holds the solutions and the executor service that manages the threads.

The ThreadMaker class takes in a list of shapes and then calls generateShapePermutations() which returns a list of all unique orders in which shapes can be tackled, then it is randomized to remove any ordering, then a loop is made to give depth id to shapes in each permutation and then submit the list of shapes, the solution list and the shared executor service to the TetrisBoardSolver.

The TetrisBoardSolver constructor creates a thread of each rotation of the current shape, and passes a copy of the current board, a clone of the current shape rotated, the list of shapes, solutions and the executor service.

Each thread tries to fit the shape with its current rotation using the method in TetrisBoard called placeTetrisShape() which tries to fit the shape on the given board using the private method fitTetrisShape() and a that method returns false, then the shape is translated within the boundaries of the board, then calls the fit method again till either a place is found or the board finishes.

If a placement is found for a certain shape and that shape is not the last shape to be fit then a new thread is made, but if it's the last shape and that shape is fit, a solution is added to the concurrent list of solutions and then executor shutdownNow() method is called which stops the creation of new threads.

In the ThreadMaker class there is a method from the executor class called awaitTermination() waiting for shutdownNow() to be called or for the timeout period to pass which is 200ms, the code is too light so it is not reached normally.

Pseudocode:

```
fitTetrisShape():
      if current shape can't fit in current place return false
      else change board elements to equal shape id return true
placeTetrisShape():
      if fitTetrisShape returns false move shape within board borders
      if fitTetrisShape is false and there are no more moves return false
      else return true
```

Project 5: Make a square

```
ThreadMaker(shapes):
      Initialize executor service with number of processors
      For each permutation:
            Give id to shapes
            Call TetrisBoardSolver and pass the shapes
      Await for any solution to get passed

TetrisBoardSolver(board,shapes,executor):
      For each rotation:
            Make new thread and pass copies
            Handle when the executor service rejects its execution

Run():
      Fit current shape:
            If fit and not the last shape:
                  Make new threads and pass next shape with all rotations
            If fit and is last shape:
                  Add solution
                  Shutdown threads
            If not fit:
                  Return

Controller:
      Calls ThreadMaker and awaits for termination
      If solutions exist print first one
      If no solutions print no solution found
```

Code Overview:

ThreadMaker(Shapes)

```java
package com.threadedsquaresolver.board;

import java.util.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.RejectedExecutionException;
import java.util.concurrent.TimeUnit;

import com.threadedsquaresolver.shapes.*;

public class ThreadMaker {

    public TetrisBoard ansBoard;
    public List<TetrisBoard> solutionList;
    public ExecutorService executor;
    public static final int noThreads = Runtime.getRuntime().availableProcessors();

    public ThreadMaker(ArrayList<TetrisShape> shapes) throws InterruptedException {…

    private ArrayList<ArrayList<TetrisShape>> generateShapePermutations(ArrayList<TetrisShape> shapes) {…

    private void generatePermutations(ArrayList<TetrisShape> nums, ArrayList<ArrayList<TetrisShape>> results,…

    private boolean isNotUnique(ArrayList<ArrayList<TetrisShape>> results, ArrayList<TetrisShape> result) {…
}
```

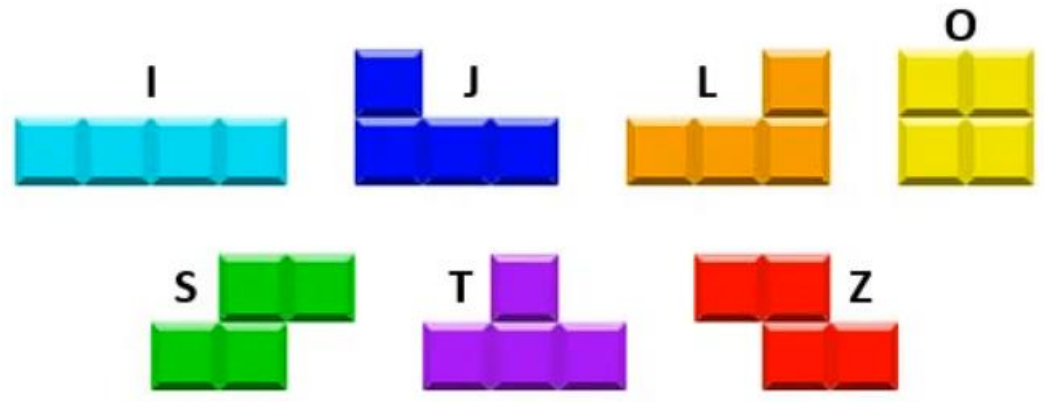
TetrisBoardSolver(Shapes,solutionList,executor)

```java
package com.threadedsquaresolver.board;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.RejectedExecutionException;

import com.threadedsquaresolver.shapes.*;

public class TetrisBoardSolver extends Thread {
    public TetrisBoard currentBoard;
    public TetrisShape currentShape;
    private ArrayList<TetrisShape> shapes;
    public List<TetrisBoard> solutionList;
    public ExecutorService executor;

    public TetrisBoardSolver(ArrayList<TetrisShape> shapes, List<TetrisBoard> solutionList, ExecutorService executor) {…

    public TetrisBoardSolver(TetrisBoard board, TetrisShape shape, ArrayList<TetrisShape> shapes,…

    @Override
    public void run() {…

    public static void printArray(int[][] array) {…
}
```

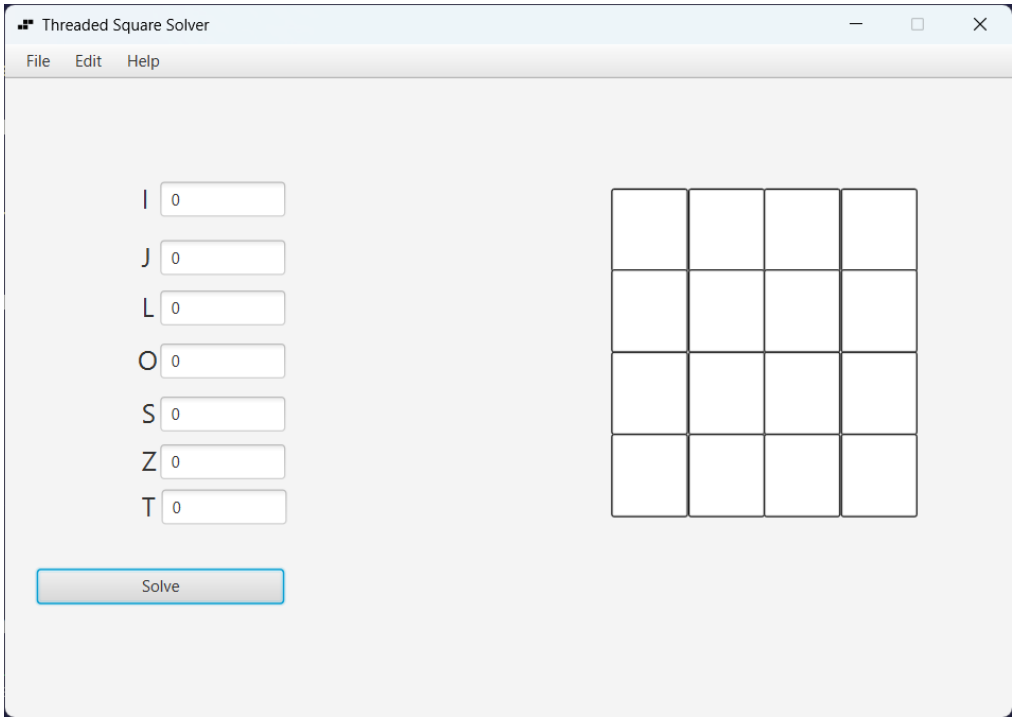
`TetrisBoard()`

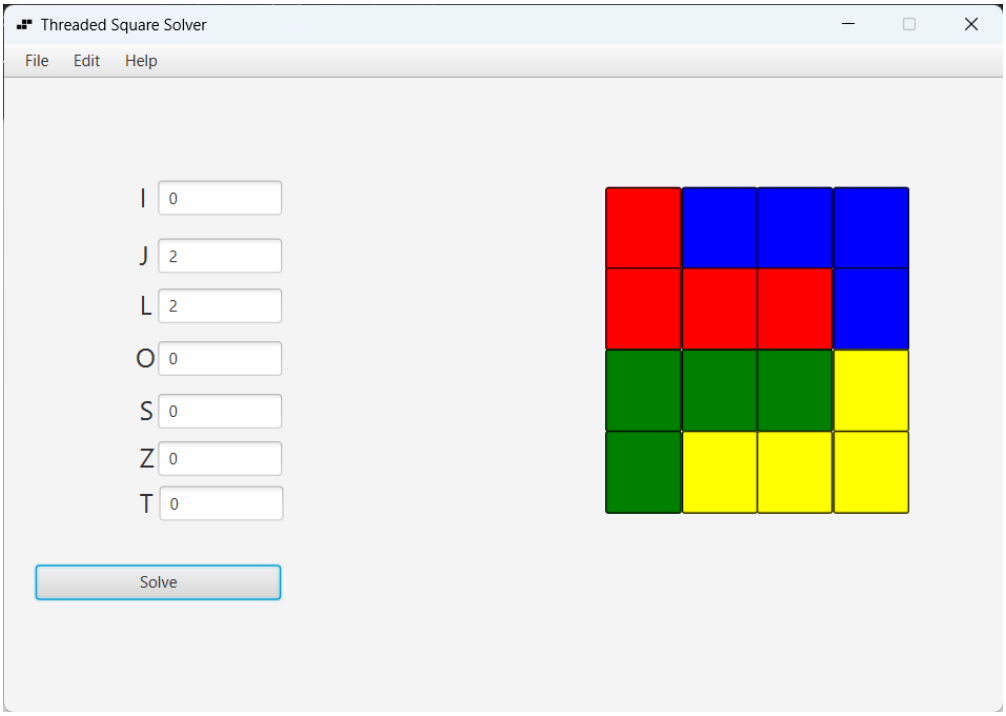
```
1   package com.threadedsquaresolver.board;
2
3   import java.util.ArrayList;
4   import java.util.Arrays;
5
6   import com.threadedsquaresolver.shapes.*;
7
8   public class TetrisBoard {
9       private int[][] board;
10      private boolean finished = false;
11      private ArrayList<TetrisShape> shapes;
12
13  >   public TetrisBoard() {···
19
20  >   public TetrisBoard(TetrisBoard board) {···
30
31  >   public int[][] getBoard() {···
34
35  >   public ArrayList<TetrisShape> getArrayList() {···
38
39  >   public String getString(ArrayList<TetrisShape> arr) {···
46
47  >   private boolean fitTetrisShape(TetrisShape shape, int i, int j) {···
66
67  >   public boolean placeTetrisShape(TetrisShape shape) {···
88  }
```

Set Pieces:



GUI Overview:





Project 5: Make a square