**StudentMultiPuzzleSolver.java**

**Invariants:**

1. Puzzle will contain at most one solution (return either “*a solution* or “*no solution*”)
2. Puzzle will not contain cycles and/or loops
3. Puzzle will no larger than 20,000 x 20,000 cells

**1.Method Name:**  *StudentMultiPuzzleSolver*()

**Pre-conditions:**  must take an object of “*Puzzle”*

**Post-conditions:**  returns an object of class *StudentMultiPuzzleSolver*

**Exceptions:** Nil

**2.Method name:** *solve()*

**Pre-conditions:**

1. Puzzle cannot be null.
2. There is always a wall to south/east/north/west of cells on southern/eastern/northern/western border of the puzzle.
3. Position

**Post-conditions:** return either “*a solution i.e. List<Direction>*” or “*no solution i.e. null*”

**Exceptions:** Nil

**Approach 1:**  using A\* pathfinding algorithm.

**Approach 2:** using DFS

1. Create a *Fixed Thread pool* that creates a pool of threads.
2. Create a List that stores results of *Future* tasks that store the direction (solution to our problem).
3. Retrieve the start position.
4. Create tasks for threads, each having a possible choice
5. Execute the tasks until they are complete (a shutdown guarantees all the threads are complete).
6. Iterate through the List that stores *Future* results and return the solution if present otherwise null.