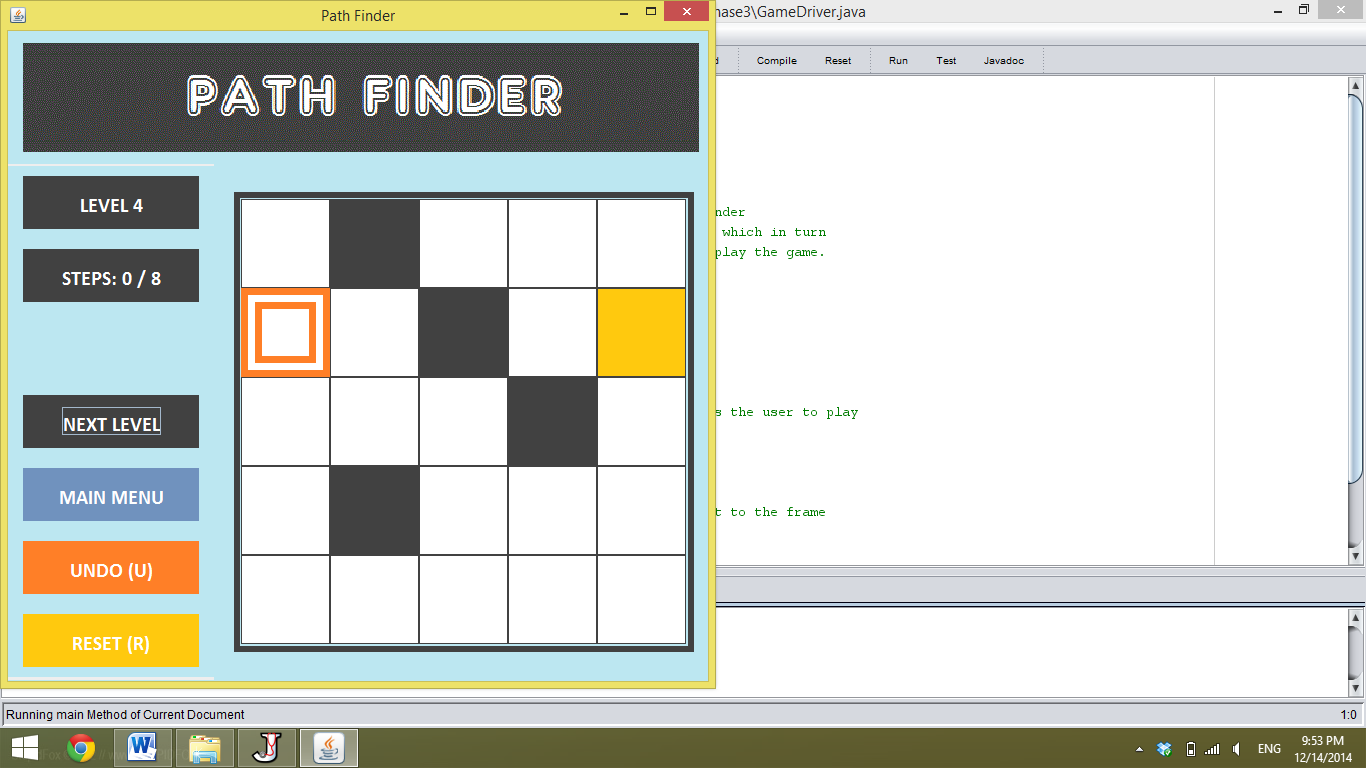
**Challenges**

**GridGraph.java**

* Constructor that reads in from a file
  + Reading in information from a file
  + Reading in multiple targets (while loop)
  + Reading in multiple blocks (another while loop)
* Constructor that constructs random grid given width & height
  + Uses the nextInt(int range) method in the Random class
  + Relies on a private method isTaken(GridCoordinate gc) to avoid “overlapping” blocks and targets
* Calculating the shortest path with 1 target
  + Had to modify breadth-first search (which returns a LinkedList<GridCoordinate> of all vertices visited)
  + Added an additional method pathLength(LinkedList<GridCoordinate> bfs) which takes in a bfs list and returns the length of the resulting path
    - Relies on the use of “previous” references

****

**A**

**B**

**D**

**N**

**C**

**I**

**G**

**E**

**F**

**H**

**Q**

**O**

**M**

**K**

**L**

**J**

**R**

**S**

**P**

**T**

BFS algorithm returns a linked list of vertices, but the size of this list doesn’t give us the length of the shortest path. Previous references are needed.

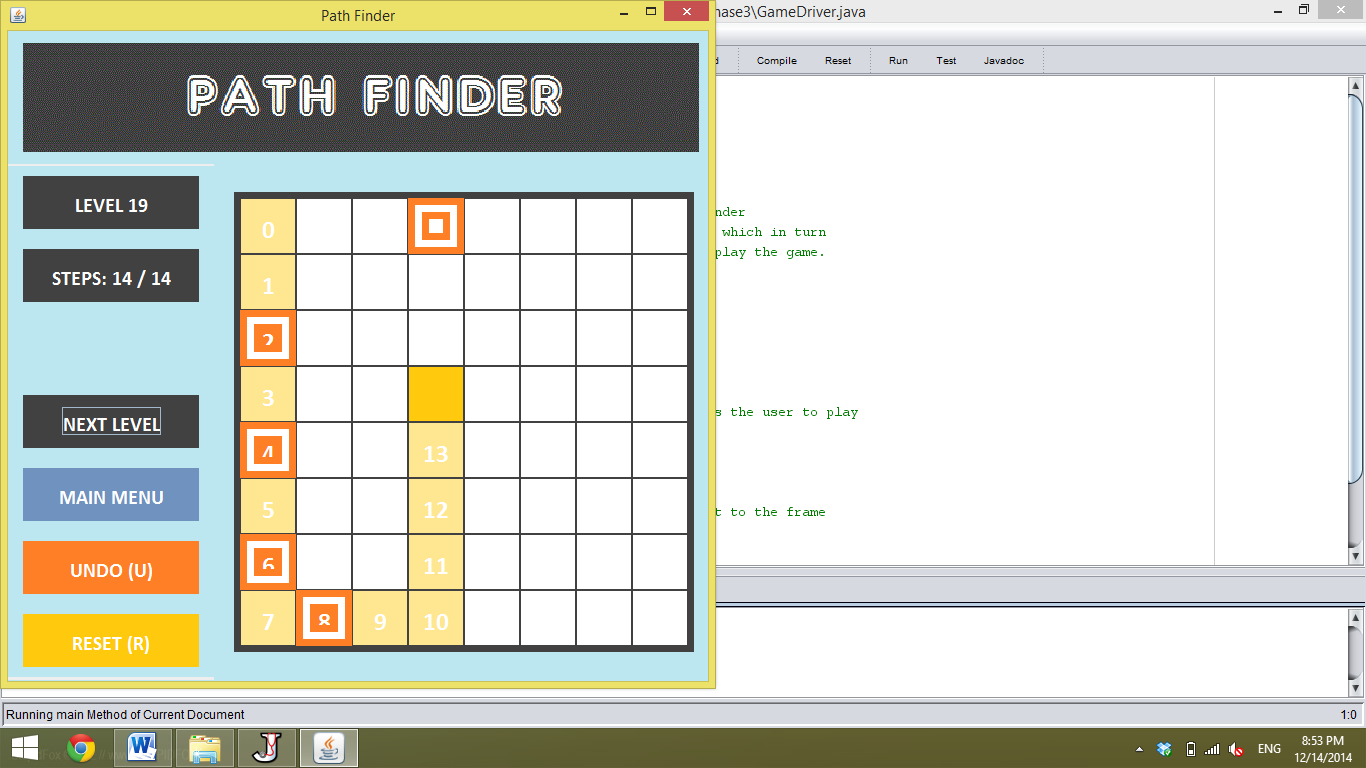
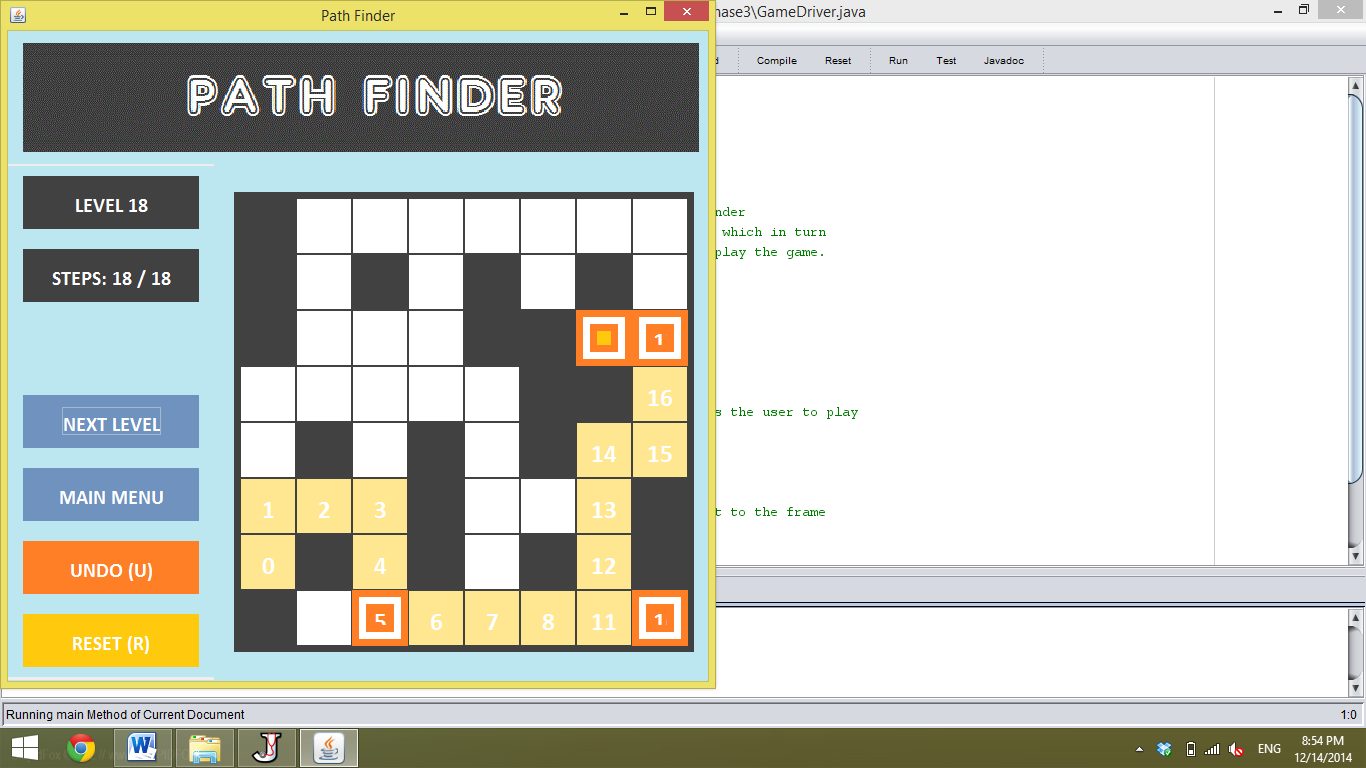
\*→A→A→A→B→C→E→F→F→H→I→J→K→L→M→N→O→O→P→Q

A→B→C→D→E→F→G→H→I→J→K→L→M→N→O→P→Q→R→S→T

Tracing through previous references results in:

A→C→F→I→K→M→O→Q→T which has length 8 (shortest path length)

* Calculating the shortest path with multiple targets
  + First attempt: travel through path in “steps” (choosing the current closest target as next step)
    - Failed: when targets “baited” computer down bad path



* + Correct strategy (recursion)
    - Determines the shortest path recursively (complicated and a bit confusing)

**Minor Bug**

**GridPanel.java**

* Undo method = doesn’t account for situation in which user “traces” over already existing path (very minor, purely visual problem)