Pseudo code implementation for the nine methods.

Ujvala Pradeep

1. void setAllDirections(Cell[][] grid, int routerRow, int routerCol)
    - Iterate through row indices in grid
    - Iterate through column indices in grid
    - Call setDirection function and calling the direction function by passing the i, routerRow,j and routerCol values to set the direction of each cell grid with respect to the router position.
      grid[i][j].setDirection(direction(routerRow,routerCol,i,j))

2. double fspl(double distance, double frequency)
    - initialize variable double freespl=20*20*Math.log10(distance)+20*Math.log10(frequency)+92.45
    - return freespl

3. double findMinSignal(Cell[][] grid)
    - assign min=grid[0][0].getSignal()
    - iterate through grid row indices
    - iterate through grid column indices
    - check min > grid[row][col].getSignal()
    - assign min= grid[i][j].getSignal()
    - return min

4. void printMinCellCoordinates(Cell[][] grid, double minSignal)
    - iterate through grid
    - check if grid[i][j] < minSignal
    - if true- print coordinates (i,j)

5. boolean isValid(Cell[][] grid)
    - iterate through the grid
    - check if grid[i][j].getEast==gird[i][j+1].getWest
    - if true, Boolean bool=true
    - if false, Boolean bool=false
    - check if grid[i][j].getNorth==grid[i+1][j].getSouth
    - if true, Boolean bool=true
    - if false, Boolean bool=false

6. boolean equivalent (Cell[][] grid1, Cell[][] grid2)

- check if grid1.length==row-grid2.length
- if true:
  - iterate through either grid values
  - check if grid1[i][j].getSignal()-grid2[i][j].getSignal()<=epsilon
  - if true, return true
  - else return false
- if false, return false

7. int attenRate(Cell[][] prev, int row, int col)
   - initialize string Direc=prev[row][col].getDirection;
   - check if Direc=='N';
   - if true:
     - assign neighbour_row=row-1;
     - assign neighbour_col=col;
     - get attenuation_rate_neighbour = prev[neighbour_row][neighbour_col].getRate()
     - get wall_type=prev[neighbour_row][neighbour_col].getSouth();
     - check if wall_type=="b"or " c" or "d" or "g" or "w" or "n"
     - assign wall_atten_value to corresponding value
     - atten_rate= attenuation_rate_neighbour+wall+atten_value;
   - repeat above steps for if Direc=="S", Direc=="E", Direc=="W"
   - if Direc=="NE" or Direc='NW' or Direc=='SE' or Direc=='SW';
     - get attenuation_rate_direction1 from the above steps
     - get attenuation_rate_direction2 from the above steps
     - if attenuation_rate_direction1 > attenuation_rate_direction2
     - assign atten_rate=attenuation_rate_direction1;
     - else if attenuation_rate_direction2 > attenuation_rate_direction1
     - assign atten_rate=attenuation_rate_direction2;
   - return atten_rate;

8. void read(Cell[][] grid, Scanner scnr)
   - Initialize int row =number of rows in grid
   - Initialize int col =number of columns in grid
   - Iterate through row indices in grid
   - Iterate through column indices in grid
   - Initialize String ch = scnr.next();
   - Call function grid[i][j].setWalls(ch) ;

9. void iterate(Cell[][] current, Cell[][] previous, int routerRow, int routercol)
    - iterate through current
    - while i!=routerRow and j!=routercol;
    - call and initialize double distance_current = current[i][j].getDistance()
    - initialize double current_fspl=fspl(distance_current,5);
    - initialize int atten_rate_current=attenRate(previous,i,j);
    - calculate int current_signal= 23-atten_rate-current_fspl;
    - get current_dierection=current[i][j].getDirection();
    - if i==routerRow and j==routerCol
    - current_signal=23
    -
    - write current_signal,distance_current, atten_rate_current,current_direction into text file signal.txt
    - write a blank line into signal.txt