EightPuzzle.java

```
package eightpuzzle;
public class EightPuzzle {
   private int gn=0;
   private Board start;
   private Board goal;
   public void initStart()
   {
      System.out.println("\n\n Enter start Board : ");
      start=new Board();
      start.initBoard();
      System.out.println("\n\nThe given start board is : ");
      start.display();
   }
   public void initGoal()
   {
      System.out.println("\n\n Enter goal Board : ");
      goal=new Board();
      goal.initBoard();
      System.out.println("\n\nThe given goal board is : ");
      goal.display();
   }
   public void solve()
   {
      Board cur = start;
      while(true)
      {
         System.out.println("\n\nBoard after "+gn+" moves : ");
         cur.display();
         if(cur.equals(goal))
         {
            System.out.println("\nGoal state achieved.");
            return;
         }
         gn++;
         cur = cur.nextMove(gn, goal);
      }
   }
   public static void main(String[] args) {
      EightPuzzle ep = new EightPuzzle();
      ep.initStart();
      ep.initGoal();
      System.out.println("\n\nThe board is solved as : \n");
      ep.solve();
```

```
    }
}
```

Board.java

```
package eightpuzzle;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import javax.swing.JOptionPane;
public class Board {
    public int board[][];
    private  int blankX,blankY;
    public Board()
    {
        this.board = new int[3][3];
    }
    public Board(Board b)
    {
        this.board = b.board;
        this.blankX = b.blankX;
        this.blankY = b.blankY;
    }
    public void initBoard()
    {
        Scanner inp = new Scanner(System.in);
        System.out.println("\nEnter one tile as '0' ie. Blank tile\n");
        for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
            {
                board[i][j] = Integer.parseInt(JOptionPane.showInputDialog("Enter the value of tile
["+i+"]["+(j)+"] : "));

                if(board[i][j] == 0)
                {
                    blankX=i;
                    blankY=j;
                }
            }
        }
    }
    public int[][] getBoard()
    {
        return board;
    }
```

```
  public void setBoard(int[][] board)
  {
     for(int i=0; i<3; i++)
     {
        for(int j=0; j<3; j++)
        {
           this.board[i][j] = board[i][j];
        }
     }
}
   public void setBlankX(int x)
  {
     blankX = x;
  }
   public void setBlankY(int y)
  {
     blankY = y;
  }
  public void display()
  {
     for(int i=0; i<3; i++)
     {
        for(int j=0; j<3; j++)
        {
           System.out.print("\t"+board[i][j]);
        }
        System.out.println();
     }
  }
  public Board nextMove(int gn, Board goal)
  {
     Board temp = new Board();
     Board next = new Board();
     int minFn = Integer.MAX_VALUE;
     System.out.println("\nPossible moves are : ");
     if(blankY>0)
     {
        temp.setBoard(board);
        temp.swap(blankX, blankY, blankX, blankY-1);
        int fn = (temp.getManhattanDistance(temp,goal))+gn;
        System.out.println("\nFor Fn = "+fn+" : ");
        temp.display();
        if(fn < minFn)
        {
           minFn = fn;
```

```
        next.setBoard(temp.board);
        next.setBlankX(blankX);
        next.setBlankY(blankY-1);
    }
}
if(blankY<2)
{
    temp.setBoard(board);
    temp.swap(blankX, blankY, blankX, blankY+1);
   int fn = (temp.getManhattanDistance(temp,goal))+gn;
    System.out.println("\nFor Fn = "+fn+" : ");
    temp.display();
    if(fn < minFn)
    {
        minFn = fn;
        next.setBoard(temp.board);
        next.setBlankX(blankX);
        next.setBlankY(blankY+1);
    }
}
if(blankX>0)
{
    temp.setBoard(board);
    temp.swap(blankX, blankY, blankX-1, blankY);
   int fn = (temp.getManhattanDistance(temp,goal))+gn;
    System.out.println("\nFor Fn = "+fn+" : ");
    temp.display();
    if(fn < minFn)
    {
        minFn = fn;
        next.setBoard(temp.board);
        next.setBlankX(blankX-1);
        next.setBlankY(blankY);
    }
}
if(blankX<2)
{
    temp.setBoard(board);
    temp.swap(blankX, blankY, blankX+1, blankY);
    int fn = (temp.getManhattanDistance(temp,goal))+gn;
    System.out.println("\nFor Fn = "+fn+" : ");
    temp.display();
    if(fn < minFn)
    {
        minFn = fn;
```

```
            next.setBoard(temp.board);
            next.setBlankX(blankX+1);
            next.setBlankY(blankY);
        }
    }
    return next;
}
public void swap(int i1, int j1, int i2, int j2)
{
    int temp = board[i1][j1];
    board[i1][j1] = board[i2][j2];
    board[i2][j2] = temp;

}
public boolean equals(Board b)
{
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
        {
            if(!(this.board[i][j] == b.board[i][j]))
            {
                return false;
            }
        }
    }
    return true;
}
public int getManhattanDistance(Board current , Board goal)
{
    int distance = 0;
    int search;
    int cloc1=0,cloc2=0,gloc1=0,gloc2=0;
    for(int j=0;j<3;j++)
    {
        for(int k=0;k<3;k++)
        {
            search = current.board[j][k];
            if(search!=0)
            {
                cloc1 = j;
                cloc2 = k;
                for(int a=0;a<3;a++)
                {
                    for(int b=0;b<3;b++)
```

```
                {
                    if(search == goal.board[a][b])
                    {
                        gloc1 = a;
                        gloc2 = b;
                    }
                }
            }
            distance = distance + Math.abs(cloc1 - gloc1)+Math.abs(cloc2 - gloc2);
        }
      }
    }
    return distance;
  }
}
```

Output -

Enter start Board :
Enter one tile as '0' ie. Blank tile
The given start board is :

1	3	4
8	0	5
7	2	6

Enter goal Board :
Enter one tile as '0' ie. Blank tile
The given goal board is :

1	2	3
8	0	4
7	6	5

The board is solved as :
Board after 0 moves :

1	3	4
8	0	5
7	2	6

Possible moves are :

For Fn = 8 :

1	3	4
0	8	5
7	2	6

For Fn = 8 :

1	3	4
8	5	0
7	2	6

For Fn = 8 :

1	0	4
8	3	5
7	2	6

For Fn = 6 :

1	3	4
8	2	5
7	0	6

Board after 1 moves :

1	3	4
8	2	5
7	0	6

Possible moves are :

For Fn = 8 :

1	3	4
8	2	5
0	7	6

For Fn = 6 :

1	3	4
8	2	5
7	6	0

For Fn = 8 :

1	3	4
8	0	5
7	2	6

Board after 2 moves :

1	3	4
8	2	5
7	6	0

Possible moves are :

For Fn = 8 :

1 3 4
8 2 5
7 0 6

For Fn = 6 :

1 3 4
8 2 0
7 6 5

Board after 3 moves :

1 3 4
8 2 0
7 6 5

Possible moves are :

For Fn = 8 :

1 3 4
8 0 2
7 6 5

For Fn = 6 :

1 3 0
8 2 4
7 6 5

For Fn = 8 :

1 3 4
8 2 5
7 6 0

Board after 4 moves :

1 3 0
8 2 4
7 6 5

Possible moves are :

For Fn = 6 :

1 0 3
8 2 4
7 6 5

For Fn = 8 :

1 3 4

8 2 0
7 6 5

Board after 5 moves :

1 0 3
8 2 4
7 6 5

Possible moves are :

For Fn = 8 :

0 1 3
8 2 4
7 6 5

For Fn = 8 :

1 3 0
8 2 4
7 6 5

For Fn = 6 :

1 2 3
8 0 4
7 6 5

Board after 6 moves :

1 2 3
8 0 4
7 6 5

Goal state achieved.