```java
import java.util.*;

class Direction {
  public String direction;
  public Direction(String dir)
  {
    direction = dir;
  }
  public int[] step(int x, int y)
  {
    //Step specified direction based on position
    if (direction.equals("left")) { return new int[]{x-1, y}; }
    else if (direction.equals("up")) { return new int[]{x, y-1}; }
    else if (direction.equals("right")) { return new int[]{x+1, y}; }
    else if (direction.equals("down")) { return new int[]{x, y+1}; }
    else if (direction.equals("upleft")) { return new int[]{x-1, y-1}; }
    else if (direction.equals("upright")) { return new int[]{x+1, y-1}; }
    else if (direction.equals("downleft")) { return new int[]{x-1, y+1}; }
    else if (direction.equals("downright")) { return new int[]{x+1, y+1}; }
    else { return new int[]{x, y}; }
  }
}

class Gomoku {
  public final int NUM_OF_X_TILES = 8;
  public final int NUM_OF_Y_TILES = 8;

  private char[][] gameBoard = new char[NUM_OF_X_TILES][NUM_OF_Y_TILES];

  //Map where the key is the player and the color is the value
  private HashMap<String, Character> players = new HashMap<String, Character>();

  private final char emptySpace = '¤';

  private char winningColor = emptySpace;

  public Gomoku() {
    //Reset game by filling board with empty spaces, resetting winner
    //and removing all players
    reset();
  }

  public char[][] getGameBoard() {
    return gameBoard;
  }

  public HashMap<String, Character> getActivePlayers() {
    return players;
  }

  public void addPlayer(String playerName, char color) {
    //Create new pair with player name and the color
    players.put(playerName, color);
    System.out.println("Gomoku: " + playerName + " joined as color " + color);
  }

  public void addTile(String playerName, int x, int y) {
```

```
58        //Assign specified position to user's color
59        gameBoard[x][y] = players.get(playerName);
60      }
61
62      public String getWinningMessage() {
63        String title = "Team " + winningColor + " has won!";
64        String winners = "\nWinners in team " + winningColor + ":";
65        String losers = "";
66        //Decide who is the loser based on the winner
67        switch (winningColor)
68        {
69          case 'x':
70            losers = "\nLosers in team o:";
71            break;
72          case 'o':
73            losers = "\nLosers in team x:";
74          default:
75            break;
76        }
77
78        //Go through all players
79        for (Map.Entry<String, Character> player : players.entrySet())
80        {
81          //Player is a winner
82          if (player.getValue() == winningColor)
83          {
84            winners+="\n" + player.getKey();
85          }
86          //Player is a loser
87          else
88          {
89            losers+="\n" + player.getKey();
90          }
91        }
92        return title + winners + losers;
93      }
94
95      public boolean decideWinner() {
96        boolean left = false;
97        boolean right = false;
98        boolean up = false;
99        boolean down = false;
100       for (int x = 0; x < NUM_OF_X_TILES; x++)
101       {
102         for(int y = 0; y < NUM_OF_Y_TILES; y++)
103         {
104           //Get color of tile
105           char color = gameBoard[x][y];
106
107           //A tile was found
108           if (color != emptySpace)
109           {
110             //Check possible directions for tile to get five in a row
                  in
111             if (x >= 4) { left = true; }
112             if (x <= (NUM_OF_X_TILES - 5)) { right = true; }
113             if (y >= 4) { up = true; }
```

```
114             if (y <= (NUM_OF_Y_TILES - 5)) { down = true; }
115
116             //Create list of possible directions to get five in a row in
117             List<Direction> directions = new ArrayList<Direction>();
118             if (left) { directions.add(new Direction("left")); }
119             if (right) { directions.add(new Direction("right")); }
120             if (up) { directions.add(new Direction("up")); }
121             if (down) { directions.add(new Direction("down")); }
122             if (up && left) { directions.add(new Direction("upleft")); }
123             if (up && right) { directions.add(new Direction("upright")); }
124             if (down && left) { directions.add(new Direction("downleft")); }
125             if (down && right) { directions.add(new Direction("downright")); }
126
127             //Go through possible directions
128             for (Direction dir : directions)
129             {
130               //Check if there's five in a row of the specified color...
131               //...at the specified position...
132               //...in the specified direction
133               if (fiveInARow(color, x, y, dir))
134               {
135                 //Set winner and return a team has won
136                 winningColor = color;
137                 return true;
138               }
139             }
140           }
141           //Reset possible directions for next tile
142           left = false;
143           right = false;
144           up = false;
145           down = false;
146         }
147       }
148       return false;
149     }
150
151     private boolean fiveInARow(char color, int x, int y, Direction dir) {
152       //Start at specified position
153       int[] steps = new int[] {x, y};
154       for (int i = 0; i < 5; i++)
155       {
156         //Check if there's not a correct color on the new tile
157         if (gameBoard[steps[0]][steps[1]] != color)
158         {
159           return false;
160         }
161         //Step in direction based on position
162         steps = dir.step(steps[0], steps[1]);
163       }
164       return true;
165     }
166
167     public boolean validColor(String color) {
168       //Valid colors are x or o
169       return color.equals("x") || color.equals("o");
170     }
```

```
171
172     public boolean validColor(char color) {
173       //Valid colors are x or o
174       return color == 'x' || color == 'o';
175     }
176
177     public boolean canAddTile(int x, int y) {
178       //x-cordinate is outside of board
179       if (x < 0 || x >= NUM_OF_X_TILES)
180       {
181         return false;
182       }
183       //y-cordinate is outside of board
184       else if (y < 0 || y >= NUM_OF_Y_TILES)
185       {
186         return false;
187       }
188       //Space is not occupied (0) means the tile can be added
189       return gameBoard[x][y] == emptySpace;
190     }
191
192     public boolean isActivePlayer(String userName) {
193       return players.containsKey(userName);
194     }
195
196     public void reset() {
197       //Fill gameboard with empty spaces (0)
198       for (int x = 0; x < NUM_OF_X_TILES; x++)
199       {
200         for(int y = 0; y < NUM_OF_Y_TILES; y++)
201         {
202           gameBoard[x][y] = emptySpace;
203         }
204       }
205       //Remove all players
206       players.clear();
207       winningColor = emptySpace;
208     }
209   }
210
```