```
// PARTIAL DEFINITION OF SOME CLASSES/STRUCTS FOR THE DEVELOPMENT OF THE 2ND PROJECT (ROBOT GAME)

////////////////////////////////////////////////////////////////////////////
struct Movement
{
  int dRow, dCol; // displacement, taking into account the chosen movement
};

////////////////////////////////////////////////////////////////////////////
struct Position
{
  int row, col;
};

////////////////////////////////////////////////////////////////////////////
class Player {
public:
  Player(int row, int col, char symbol);
  int getRow() const;
  int getCol() const;
  char getSymbol() const;
  bool isAlive() const;
  void setAsDead();
  bool move(Movement delta);
private:
  int row, col;
  bool alive;
  char symbol;
};

////////////////////////////////////////////////////////////////////////////
class Robot {
public:
  enum State { ALIVE, STUCK, DEAD };
  Robot(int row, int col);
  int getID() const;
  char getSymbol() const; // get char representation of robot (R if alive, r if dead)
  int getRow() const;
  int getCol() const;
  Position getPosition() const;
  bool isAlive() const;
  void setRow(int x);
  void setCol(int y);
  void setPosition(const Position &pos);
  void setAsDead();
  //other methods
private:
  static int robotCounter; //used to attribute automatically the id to the robots
  int id;
  int row, col;
  bool alive;
  // other attributes (?)
};

////////////////////////////////////////////////////////////////////////////
class Post {
public:
  Post(int row, int col, char type);
  int getRow() const;
  int getCol() const;
  char getSymbol() const; // get char representation of Post
  bool isElectrified() const;
  //other methods
private:
  int row, col;
  char type;  // '*' - electrified; '+'- non-electrified
  // other attributes (?)
  // could also have a state, like the robot(?)
};

////////////////////////////////////////////////////////////////////////////
class Maze {
public:
  Maze(int numRows, int numCols);
  bool addPost(const Post& post, const Position& position)
  int getnumRows() const;
  int getnumCols() const;
  // other methods
private:
  int numRows, numCols;
  // data structure that represents the posts
};
```

```
////////////////////////////////////////////////////////////////////////
class Game {
public:
  Game(const string & filename);
  // This constructor should initialize the Maze, the vector of Robots, and the Player,
  // using the chars read from the file
  bool play(); // implements the game loop; returns true if player wins, false otherwise
  bool isValid();
private:
  void showGameDisplay() const;
  bool collide(Robot& robot, Post& post); // check if robot collided with post (and possibly set it as dead)
  bool collide(Robot& robot, Player& player); // check if human and robot collided (and possibly set human as dead)
  // other methods, for example:
  // to check if player is trying to move to a valid place
  // to apply a valid play and check collisions
  // to check if two robots collide (and possibly set them as dead)
  // etc.
private:
  Maze maze;
  Player player;
  vector<Robot> robots;
  //other attributes
};
```