**Tests**

**White Box Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Test ID*** | ***Description*** | ***Excepted Result*** |  |
| TestGettingQuestionByDifficulty | Sysdata - ArrayList<Question> fetchQuestionsArr(E\_Difficulty diff) - checking that there is at least 1 question for each difficulty after reading from the JSON, splitting all questions by difficulty level using a for loop to sort, then returning array lists by question difficulty making sure they aren't empty | Expecting 3 Array lists with at least 1 question each of different level in each |  |
| TestDeleteHistory | Sysdata -void deleteGameHistory() -delete the game history scores, for loop on maximum of 10 entries in the Array List that keep the history scores, however if it is empty it will stop and overwrite the history score JSON. | Game history scores will be empty -empty arraylist and JSON. |  |
| TestReturnedSoldier | Game – ReturnSoldier(Soldier s) – Checks if the correct soldier returned to the board in the correct location, the method will receive the soldier and checks what is it color, it will check the location entered and if it’s valid location using if/else statements. | The soldier returned to valid location (no other object is in this location). |  |
| TestRemoveSoldier | Game – removeSoldier(Soldier s) – Checks if the correct soldier removed from the relevant arraylist, the method receives the soldier, checks it’s color and remove it from the arraylist. | The soldier location in the arraylist will be empty(set isAlive as false) |  |
| TestQueenDead | Soldier – checkQueenDead(int I, int j, square[][] Board, Soldier\_Color\_AtSquare color) – Checks if the queen is dead. The method receives the location chosen, the board and the color of the board, checks if in this location the queen is dead | return 0 and a relevant message that the queen is dead and sets the color of the queen to Empty |  |

**TestGettingQuestionByDifficulty**

q.getDifficulty().equals(diff))

**1**

true

**3**

**2**

choosenQuestions.add(q)

return choosenQuestions

**4**

**TestDeleteHistory**

**1**

i > 0

**2**

GameHistory.remove(i-1)

i--

Return true

**3**

**TestRemoveSoldier:**

**1**

s!=null

true

false

**3**

**2**

return false

s.getcolor.equals(Black)

**4**

white

black

i++

**5**

**6**

i++

BlackPieces[i]==s

i++

**7**

**8**

WhitePieces[i]==s

i++

**9**

BlackPieces[i].setIsAlive(false)

**10**

WhitePieces[i].setIsAlive(false)

**11**

s.setIsAlive(false)

Return true

**12**

**TestQueenDead**

(Board[x-1][y-1].getSoldierColor() != color) && i!= x-1 && j!=y-1

**1**

Board[x-1][y-1].setSoldierColor(Soldier\_COLOR\_AtSquare.EMPTY

setIsAlive(false)

**3**

**2**

(Board[x-1][y+1].getSoldierColor() != color) && i!= x-1 && j!=y+1

**4**

**5**

(Board[x+1][y+1].getSoldierColor() != color) && i!= x+1 && j!=y+1

**6**

**7**

(Board[x+1][y-1].getSoldierColor()!= color) && i!= x+1 && j!=y-1)

**8**

**9**

**10**

Return -1

(System.out.println("Queen is dead!")

Return 0

**11**

**Black Box Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Test ID*** | ***Description*** | ***Expected Results*** |  |
| TestDeleteQuestion | Sysdata - boolean removeQuestion(Question q) -Check if question is deleted | If question exists it will be deleted and should return true, otherwise error message and returns false. |  |
| TestAddQuestion | Sysdata - boolean addQuestion(Question q) -Check if question is added successfully | If question is already exist it will gave a warning message, otherwise it will add the question to the question array |  |
| TestNoNameInput | NickNameSetUpController -void StartGame() -Testing if a correct error message showing when user trying to play without inserting a name | Game won't run and an error message will appear for the user |  |
| TestYellowSquares | BoardGameController – CheckAndDoYellowSquares() – the method checks if there is always three yellow squares in the board, if not create randomly. | Method will create legal(no other object in this square) three random yellow squares. |  |

**TestDeleteQuestion**

q!=null

**1**

false true

**2**

**3**

System.out.println("There is no such a question") }

remove(QuestionsArray.indexOf(q))

Return true

**4**

return false

**5**

**TestAddQuestion**

Question.equals(q)

**1**

false true

System.out.println(“We already have this question”)

**3**

**2**

QuestionsArray.add(q)

Catch(Exception e)

**4**

Return true

**6**

**5**

Return false

**7**

**TestNoNameInput**

Whitename = null

Blackname = aaa }

!whitename.Isempty() && !blackname.isEmpty()

**1**

**3**

**2**

temp=new MainBoardController()

**4**

Catch(Exception e)

**TestYellowSquares**

x = rand.nextInt(8)

y = rand.nextInt(8)

Rectangle yellowr

g.getboard()[x][y] != null

**1**

**2**

g.getboard()[x][y].getSquareColor != null

**3**

g.getBoard()[x][y].getSquareColor()==SQUARE\_COLOR.BLACK && g.getBoard()[x][y].getSoldierColor()==Soldier\_COLOR\_AtSquare.EMPTY

**4**

**5**

**6**

g.getBoard()[x][y].setSquareColor(SQUARE\_COLOR.YELLOW

yellowr=getRectangle(x,y)

yellowr != null

**7**

**8**

yellowr.setFill(Color.rgb(255,255,0))

System.out.println(numofYellowSquares)

**JUnit Tests**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Test ID*** | ***Description*** | ***Excepted Results*** | ***Actual Result*** |
| TestDeleteHistory | Sysdata -void deleteGameHistory() -delete the game history scores, for loop on maximum of 10 entries in the Array List that keep the history scores, however if it is empty it will stop and overwrite the history score JSON | Game history scores will be empty -empty arraylist and JSON | All tests returned the correct instance and passed as true |
| TestGettingQuestionByDiffuculty | Sysdata -ArrayList<Question> fetchQuestionsArr(E\_Difficulty diff) -checking that there is at least 1 question for each difficulty after reading from the JSON, splitting all questions by difficulty level using a for loop to sort, then returning array lists by question difficulty making sure they aren't empty | Expecting 3 Array lists with at least 1 question each of different level in each | The method returned 3 array lists that weren't empty |
| TestDeleteQuestion | Sysdata – boolean removeQuestion(Question q) – delete the given question from the QuestionsArray if exist and write the changes to the JSON file | The entered question will be removed and the JSON will be over written | The question has been removed from the QuestionsArray and from the JSON file. |
| TestEditQuestion | Sysdata – boolean editQuestion(Question old, Question new) – Edit an existing question with a new one. | The old question in the Question Array will be changed with the new one | The Question has been changed with the new one in the QuestionsArray. |
| TestPlayerWithDifferentColor | Game – AddPlayer(String wp, String bp) – The method will add two players to the game. The test method checks if the players have different colors(black and white) | Two players will be added to the game with different colors | Two players have been added to the game, one is white player and the second is a black one. |