**Test Description**

**Students:**

**Safa 315182907**

**Klara 206442535**

**Nada 323000562**

**Sana 212207773**

**White Box Tests**

**id:1**

**תיאור :**

בדיקת קופסה לבנה אשר בודקת תנועה חריגה של הסוס ברמה 2 ומחזירה את המיקום החדש .

.**level2Move(*Directions* direction1 , *Directions* direction2 , *Directions* direction3 , String X , String y)**

* Knight – עצם של סוס אשר בודקים אם התנועה שלו חוקית לפי הרמה שנמצא בה
* direction 1- הכיוון הראשון שירצה לנוע בו
* direction 2 -הכיוון השני שירצה לנוע בו
* direction 3- הכיוון השלישי שירצה לנוע בו
* String x-קורדינטה x של המיקום הנוכחי שנמצא בו הסוס
* String y- קורדינטה y של המיקום הנוכחי שנמצא בו הסוס

**קלט שיוכנס:**

Knight.level2Move (*Directions*. ***RIGHT***, *Directions*. ***RIGHT***, *Directions*.***UP\_RIGHT***, "0","0")

כלומר בקלט הזה הסוס נמצא במקום 0,0 וצריך לנוע ימינה,ימינה,למעלה ימינה שבתנועה האחרונה הוא אמור לחצות את הלוח.

-יצירת אובייקט מסוג Location עם הקורדינטות (0,0) **loc=Location (0,0) ;**

-יצירת אובייקט מסוג סוס אשר מאותחל למיקום למעלה **knight1=new Knight (loc);**

-יצירת אובייקט מסוג Location אשר ישמור בתוכו את המיקום החדש של הסוס  **newLoc= new Location();**

-בודקים את החוקיות של מבנה התנועה :בקלט שנתון המבנה של התנועה חוקי לכן נבצע את המקרה האחרון

נקרא לכל תנועה לפי הסוג שלה

* **knight1.StrightMove(direction1)** => נלך למתודה הזו שתעדכן את המיקום של X ל 1 this.getLocation().setX(this.location.getX() + 1);
* **knight1.StrightMove(direction2)** => המתודה תעדכן את המיקום של X ל 2
* this.getLocation().setX(this.location.getX() + 1);
* **knight1.DiagonallyMove(direction3)** =>המתודה הזו תעדכן את X ל 3 ו Y ל 7
* this.location.setX(this.location.getX() + 1);
* this.location.setY(7);

-שומרים המיקום של knight1 בתוך newLoc **newLoc=knight1.getLocation();**

- ומחזירים אותו. **return newLoc;**

**פלט צפוי :**

**Location (3,7)**

**מחלקת שקילות :**

קלט של תנועות חוקיות

**id :2**

**תיאור :**

בדיקת קופסה לבנה אשר בודקת אם המלך יכול לזוז.

* **KingStrightMove(*Directions* dir)**
* king - עצם של מלך אשר בודקים החוקיות של התנועה שלו
* Dir – הכיוון שהמלכה רוצה לנוע אילו.

**קלט שיוכנס:**

dir=Directions. RIGHT

* נניח שהמלך נמצא במקום location= (0,1)

נקרא למתודה של **KingStrightMove(dir) -**במתודה הזו עוברים על כל מיני מקרים שהם לא חוקיים ,בקלט שהוכנס המתודה לא תכנס לאף תנאי כלומר שהמלך לא נמצא בצדדי הלוח

**פלט צפוי:**

**1**

**מחלקת שקילות:**

קלט של תנועה חוקית

**id :3**

**תיאור:**

בדיקת קופסה לבנה אשר מוחקת שאלה כלשהי באמצע המערך

**removeQuestion(Integer id)**

**-id מספר מזהה של שאלה**

**קלט שיוכנס:**

* קריאה למופע של SysData sysData = SysData.getInstance();
* ואז sysData. removeQuestion(2)
* המתודה עוברת על מערך השאלות בלולאה
* שומרת בתוך משתנה עזר את המיקום של השאלה בתוך המערך
* מעדכנת את מספר המזהה של כל השאלות העוקבות לשאלה שצריך למחוק
* מוחקת את השאלה מהמאגר

**פלט צפוי :**

**שאלה שהמזהה שווה ל 2 לא נמצאת כבר.**

**מחלקת שקילות:**

אין

**id:4**

**תיאור :**

בדיקת קופסה לבנה אשר בודקת תנועה של חלקים במשחק כאשר זזים בצורה אלכסונית

**DiagonallyMove(*Directions* dir)**

* Dir – הכיוון שהחלק רוצה לנוע אילו.

**קלט שיוכנס:**

**DiagonallyMove(Directions.DOWN\_RIGHT)**

-יוצרים אובייקט כלשהו מהמחלקות שיורשות מהמחלקת Piece שנמצאת בתוכה המתודה הזאת מכיוון שהמחלקת היא אבסטרקטית

-נניח שיצרנו מופע של Knight בשם knight שנמצא במקום (0,0)

**-אז נקרא למתודה knight. DiagonallyMove(Directions.DOWN\_RIGHT)**

-נלך למקרה של**\_RIGHT DOWN**  שבתוך המקרה הזה נלך למקרה האחרון שאומר שנוסיף אחד לקורדינטה X ו Y

**פלט צפוי :**

**מיקום של הסוס (1,1)**

**מחלקת שקילות:**

אין

**Black Box Tests**

**id:5**

**תיאור:**

בדיקת קופסה שחורה אשר מחזירה את כל התנועות האפשריות של סוס ברמה 2

**allValidMovesLevel3and4(Knight knight)**

**קלט שיוכנס:**

אובייקט מסוג סוס אשר נמצא במקום (2,2)

**פלט צפוי :**

**מערך אשר בתוכו יש את הערכים האלו מסוג Location**

**{(2,3), (1,3),(3,3),(1,2),(3,2),(2,1),(1,1),(3,1)}**

**מחלקת שקילות:**

אין

**id:6**

**תיאור :**

בדיקת קופסה שחורה אשר בודקת חוקיות תנועות של סוס ברמה 1 ומחזירה את המיקום של הסוס במידה שהן תקינות.

**level1Move(*Directions* direction1 , *Directions* direction2 , *Directions* direction3 , String X , String Y)**

* direction 1- הכיוון הראשון שירצה לנוע בו
* direction 2 -הכיוון השני שירצה לנוע בו
* direction 3- הכיוון השלישי שירצה לנוע בו
* String x-קורדינטה x של המיקום הנוכחי שנמצא בו הסוס
* String y- קורדינטה y של המיקום הנוכחי שנמצא בו הסוס

**קלט שהוכנס :**

**level2Move(***Directions*. ***Down* ,** *Directions*. ***DOWN\_RIGHT* ,** *Directions*. ***RIGHT* , “1”,”2”)**

**פלט צפוי :**

**throw exception “illegal Move”**

**מחלקת שקילות :**

קלט לא חוקי

**id:7**

**תיאור :**

בדיקת קופסה שחורה אשר בודקת עדכון של שאלה כלשהי.

**updateQuestion(Integer id, Question updated\_question)**

**-id מספר מזהה של השאלה שצריך לבצע עליה העדכון**

**- updated\_questionהשאלה המעודכנת**

**קלט שהוכנס:**

**sysData.updateQuestion(1, updated\_question)**

* קריאה למופע של SysData sysData = SysData.getInstance();
* יצירת מערך של תשובות בשם answers
* יצירת מופע חדש של שאלה
* answers,DifficultyLevel.EASY,”safa”) update\_question=new Question(1,”check”,

**פלט צפוי:**

**שאלה שמספר המזהה שווה ל 1 אכן התעדכנה**

**מחלקת שקילות:**

אין

**id:8**

**תיאור:**

בדיקת קופסה שחורה אשר בודקת המרחק הקצר של מלכה מהסוס.

**shortestDistance(Location loc1 ,Location loc2)**

**קלט שיוכנס:**

**-**נניח שהמלכה עם loc1 והסוס loc2

-loc1 =new Location (0,0)

Loc2 =new Location (0,4)

**פלט צפוי :**

**4**

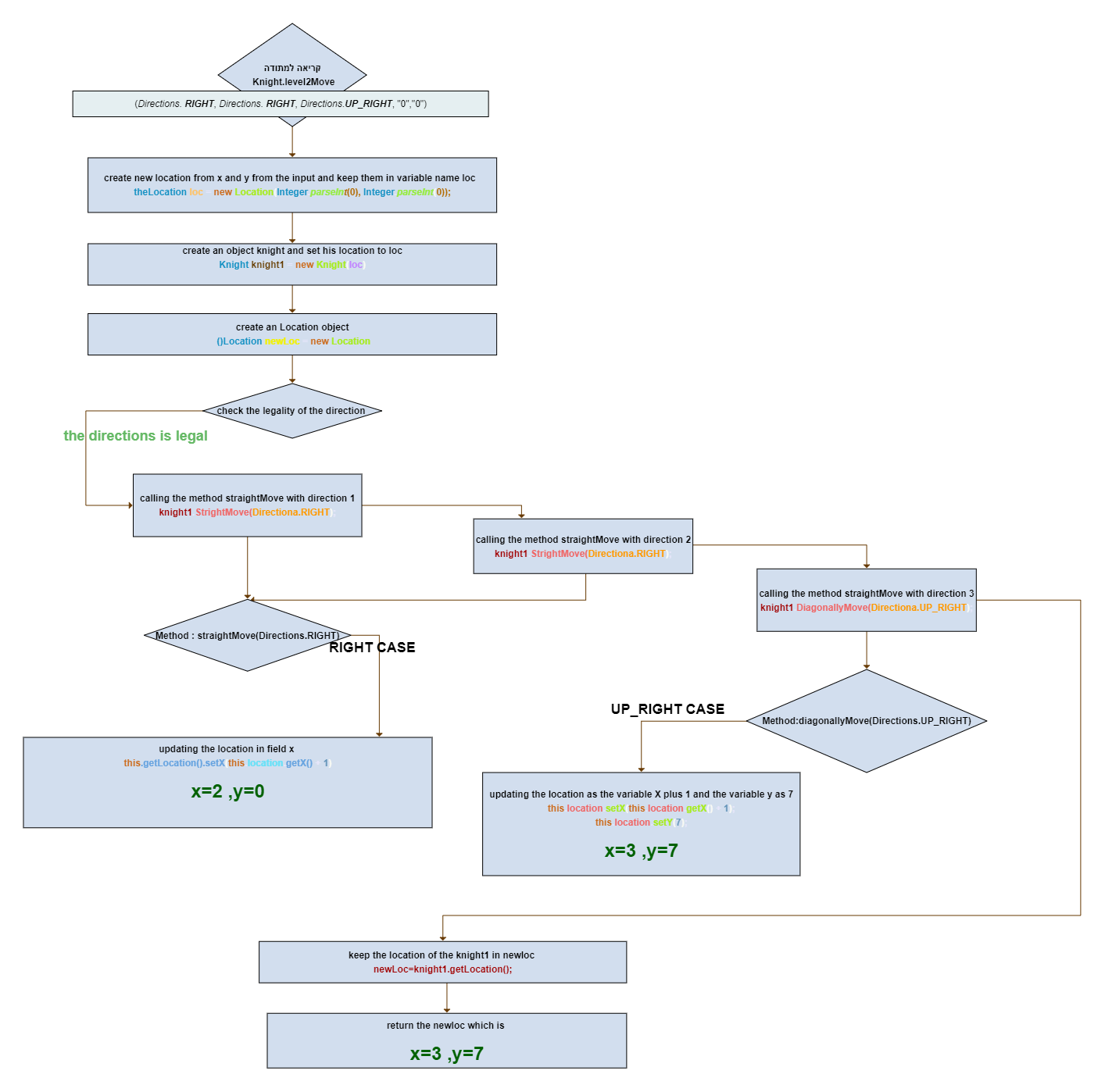
**מחלקת שקילות :**

אין

**Junit-Test**

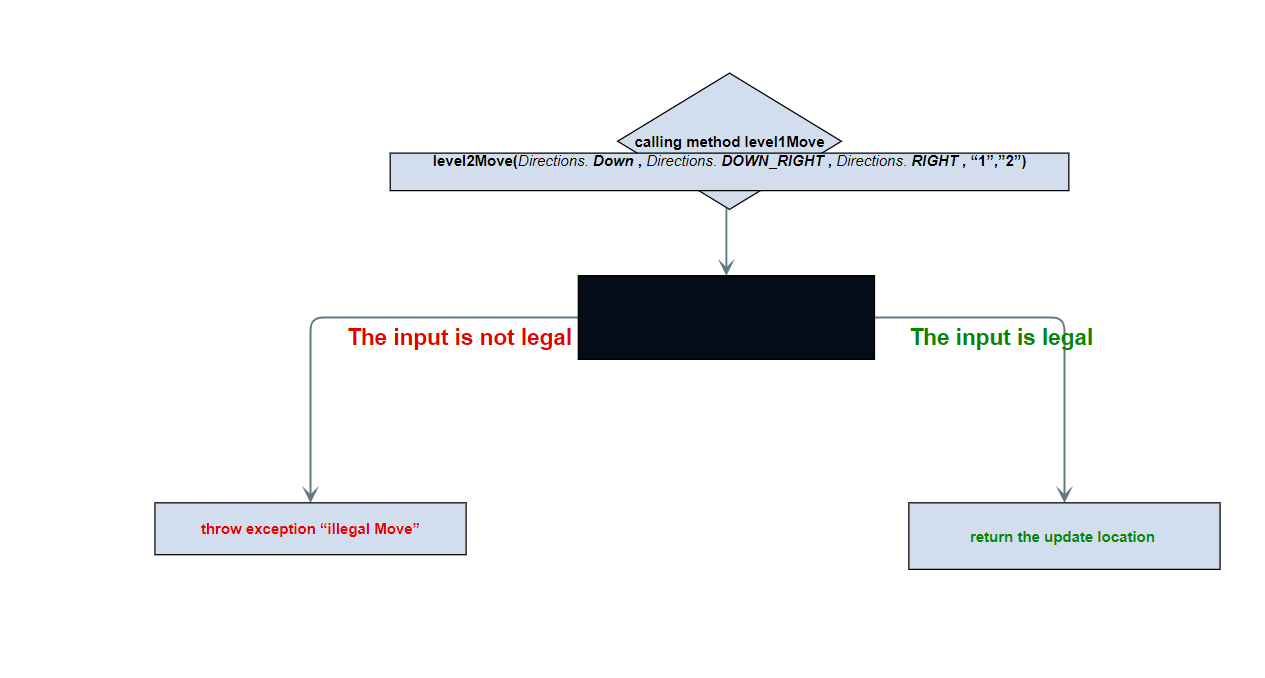
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Result | Actual Value | Expected Value | Description | Test ID |
| pass | שגיאה של  "illegal Move" | זריקת שגיאה של  "illegal Move" | בדיקה שבודקת מה קורה כאשר הסוס ינסה לזוז בתנועות לא חוקיות ברמה 1 | id:9 |
| pass | (2, 0) | (2, 0) | בדיקה של תנועות חוקיות אשר בודקת אם הסוס יגיע למקום הנכון ברמה 1 | id:10 |
| pass | (3, 1) | (3, 1) | בדיקה של תנועות רגילות אשר בודקת אם הסוס יגיע למקום הנכון ברמה 2 | id:11 |
| pass | מערך של  location  }(1, 4),(3, 4),(1,0), (3, 0), (4, 3) ,(4, 1),(0, 3),(0, 1){ | מערך של  location  }(1, 4),(3, 4),(1,0), (3, 0), (4, 3) ,(4, 1),(0, 3),(0, 1){ | בדיקה שבודקת אם יוחזר מיקום של כל התנועות האפשריות של הסוס ברמה 1 | id:12 |
| pass | 4 | 4 | בדיקה שבודקת אם יוחזר המרחק הקצר ביותר של המלכה מהסוס | id:13 |
| pass | 0 | 0 | בדיקה שבודקת חוקיות של תנועת המלכה | id:14 |
| pass | true | true | בדיקה שבודקת יצירת שני מופעים של מחלקת מסוג  singlton | id:15 |
| pass | true | true | בדיקה שבודקת הוספת שאלה מסוימת | id:16 |

**Graph for White Box, id:1**

* **the graph refers to specific input not in general**

**Graph for Black Box Test, id:6**

* **The graph refers to general input**



מצורף תמונות של הגרפים במידה והם לא ברורים