ספר פרויקט

מגישים: עומר אסייג – 215487174 , טל רוזנוולד – 328330451

שם הפרויקט: משחק הזיכרון

**תיאור הנושא:**

התוכנית היא משחק זיכרון שבו יש התאמות בין המספרים השונים. על-מנת ליישם את החלק הגרפי, התוכנית נעזרת בממשק הגרפי שב-Java. בזמן שהתוכנית ממלאת את המטריצה הלוגית (הבונה את הלוח) ומוצאת איבר שמופיע יותר מפעמיים נקבל קטע קול של "אתה פויה!". תוכנית מציגה חלון גרפי בו יש לוח לחצנים, בלחיצה על אחד מהכפתורים מופיע חלון קופץ עם הערך המייצג את הכפתור, ומופיע ל-0.1 שניות הערך המייצג את הכפתור. בעת שהערכים בין שני כפתורים מתאימים; אז תופיע הודעה: "!Match". בלחיצה על ה-X יופיע חלון קופץ בה יש את הכיתוב: "!Game Is Over", וגם צליל של ירייה.

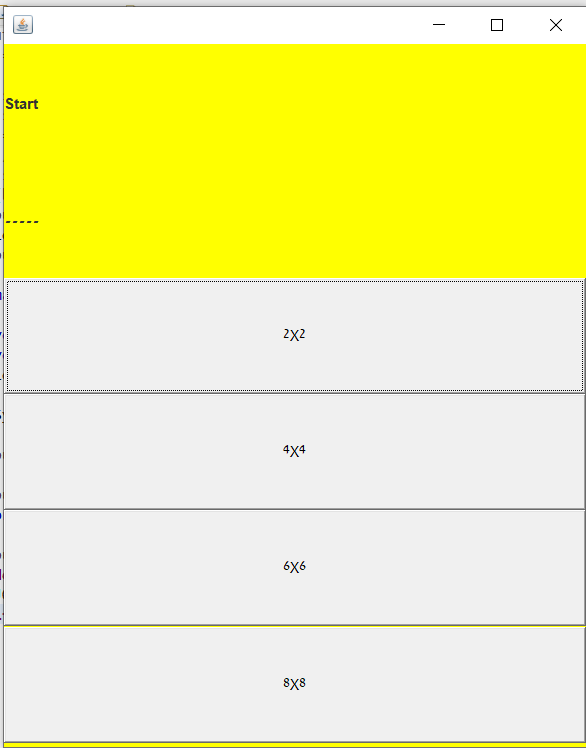
**פירוט הדרישות מהמערכת:**

* יופיע חלון גרפי עם בחירה של ארבע כפתורים (2X2, 4X4, 6X6, 8X8)
* יופיע חלון גרפי עם לוח.
* על-גבי הלוח, יהיו מוצגים מספר כפתורים בגודל שהוזן (בחלון הגרפי הראשון).
* יהיה ניתן ללחוץ על המספרים שבמשבצות.
* בלחיצה על כל כפתור, יופיע חלון קופץ (הודעה המציגה את הערך של הכפתור שבמשבצת שנבחרה).
* תראה את ערכם של הזוג שנבחר למשך 0.1 שניות.
* אם ילחצו שני כפתורים עם ערך זהה- אז, תופיע הודעה עם שמראה את האחוז הנמצא מהלוח (כולל אותו הזוג). יהיה קטע קול "המלווה" את ההודעה. אם לא, יהיה קטע קול המתריע על חוסר ההתאמה (לאחר חשיפתם).
* במידה וכל הכפתורים נמצאו אחרי ההודעה המופיעה אחרי ההתאמה, גם תופיע הודעה המודיעה שניצחת. לאחר שהודעה זו תיסגר, יהיה קטע קול המודיע על כך שניצחת ותיסגר המשחק.
* הלוח ייסגר בלחיצה על האיקס, אך לפני זה יופיע חלון קופץ המתריע על הסגירה וקטע קול של ירייה.
* החלונות הקופצים ייסגר בלחיצה על האיקס / על הOK (רק בחלון קופץ 1 איןOK)

**מבני נתונים:**

* מערך דו ממדי של מספרים שלמים.
* מערך חד ממדי של מספרים שלמים.
* **עמדת פיתוח:**
* שפת פיתוח Java.
* פלטפורמת פיתוח: .Eclipse IDE
* Intel® Core™ i5
* RAM: 8 GB
* מערכת הפעלה: Windows 10.

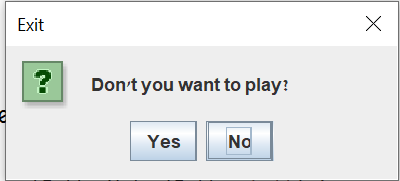
**מדריך למשתמש**



**מסך מספר 1:**

כאשר מריצים את הקוד המשך הזה מופיע, ממסך זה ניתן לבחור את הגודל שבו אתה רוצה את הלוח משחק בלחיצה על אחד מהכפתורים האפורים, לדוגמה הכפתור של 2×2 אשר החץ מצביע עליו. כאשר ניבחר הגודל יופיע חלון קופץ מספר 3 אשר אומר כי יוצרים לוח משחק חדש, ואחריו המסך יופיע בגודל המבוקש כמו שניתן לראות במסך מספר 2. במהלך יצירת הלוח ישמע קטע קול "אתה פויה".

בנוסף לכך ניתן לסגור חלון זה ובעצם כל חלון אחר בידי לחיצה על הכפתור עם ה-X כמו שהחץ מצביע, כאשר הכפתור נלחץ חלון קופץ מספר 1 מופיע.



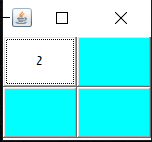
**חלון קופץ מספר 1:**

חלון זה קופץ כאשר לוחצים על כפתור ה- Xהחלון הזה קופץ. צריך ללחוץ על Yesאו על X אם רוצים לסגור את החלון כמו שהחיצים הסגולים מראה. אם לא רוצים לסגור את החלון והאיקס נילחץ בטעות יש ללחוץ על No, כמו שהחץ כחול מראה. לאחר שלוחצים על Yes יקפוץ חלון קופץ מספר 2, ואחריו החלון יסגור את עצמו.

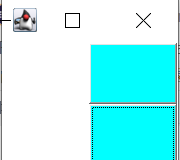


**מסך מספר 2 גרסה א:**

לאחר שבוחרים גודל מופיע החלון הזה, במקרה זה החלון מציג את הגודל של 2X2. חלון זה הוא החלון עליו משחקים. בשביל לשחק יש ללחוץ על אחד מהמלבנים הקטנים כמו האחד שמסומן בחץ. כאשר לוחצים הוא הופך ללבן עם מספר עליו. המטרה היא למצוא שניים בעלי אותו מספר. כאשר שניים בעלי אותו מספר נלחצו הם נעלמים.

בשביל לסגור חלון זה יש ללחוץ על הכפתור עם X, כמו שהחץ מציג, לאחר הלחיצה יקפוץ חלון קופץ מספר 1.

**מסך מספר 2 גרסה ב:**

ככה ניראה המסך לאחר שנלחץ מספר אחד.

**מסך מספר 2 גרסה ג:**

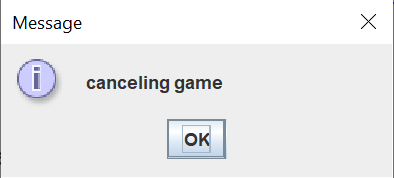
ככה ניראה המסך לאחר שזוג נימצא.

לאחר שנימצא זוג יקפוץ חלון קופץ מספר 4, חלון זה יגיד כמה אחוזים מתוך הלוח נפתרו.

כשאר כל הזוגות נמצאו וחלון קופץ מספר 4 קפץ, חלון קופץ מספר 5 יקפוץ, ואחריו המסך יסגור את עצמו.

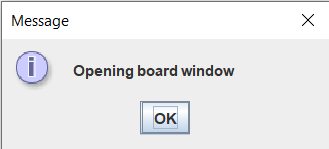
כאשר ימצא זוג ישמע קטע קול "winner.wav".

במידה ולא נימצא זוג ישמע קול No.wav"".

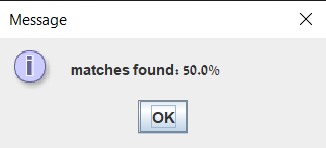


**חלון קופץ מספר 2:**

חלון זה מופיע כאשר לוחצים על "כן" בחלון קופץ מספר 1. חלון זה אומר שהמסך של המשחק הולך להיסגר לאחר סגירת חלון זה. בשביל לסגור חלון זה יש ללחוץ על "אוקי" או על X, כמו שהחיצים מראים. בסגירת מסך מספר 1 ישמע קול "boom"

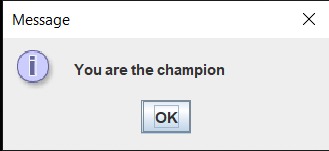


**חלון קופץ מספר 3:**

חלון זה מופיע לאחר שבוחרים בגודל המסך מספר 1, לאחר שסוגרים אותו מסך מספר 2 יופיע בגודל הנבחר. בשביל לסגור חלון זה יש ללחוץ על "אוקי" או על X, כמו שהחיצים מראים.

**חלון קופץ מספר 4:**

חלון זה קופץ כל פעם שנימצא זוג והוא מראה כמה אחוזים מתוך הזוגות נמצאו. בשביל לסגור חלון זה יש ללחות על "אוקי" או על X, כמו שהחיצים מראים.

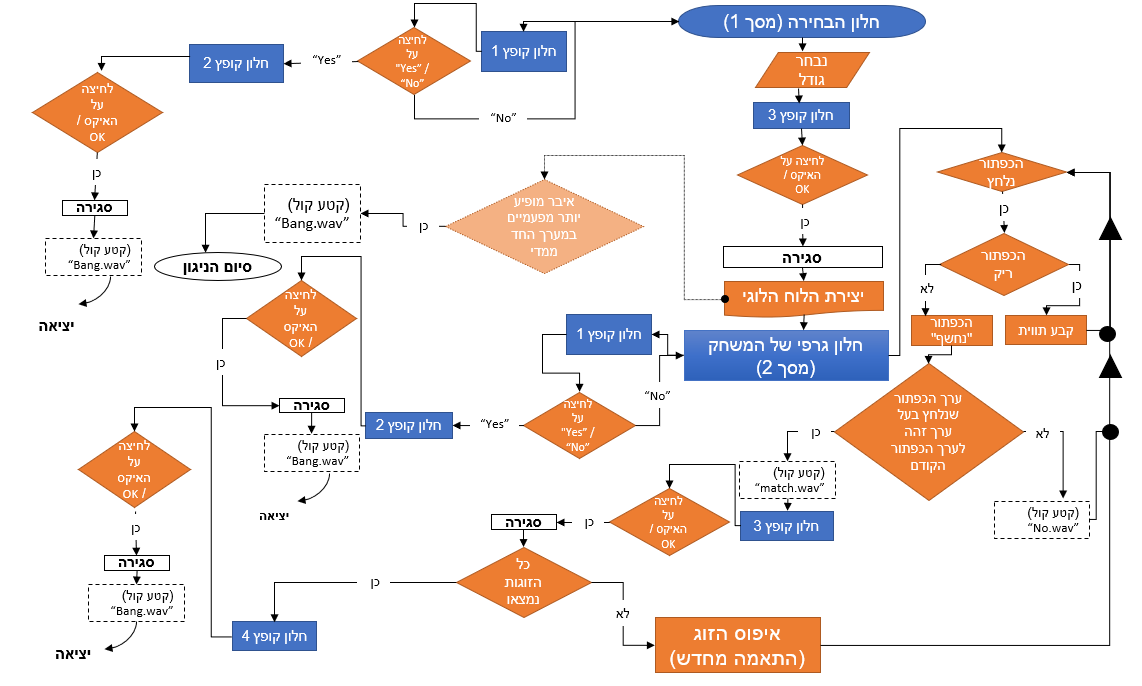


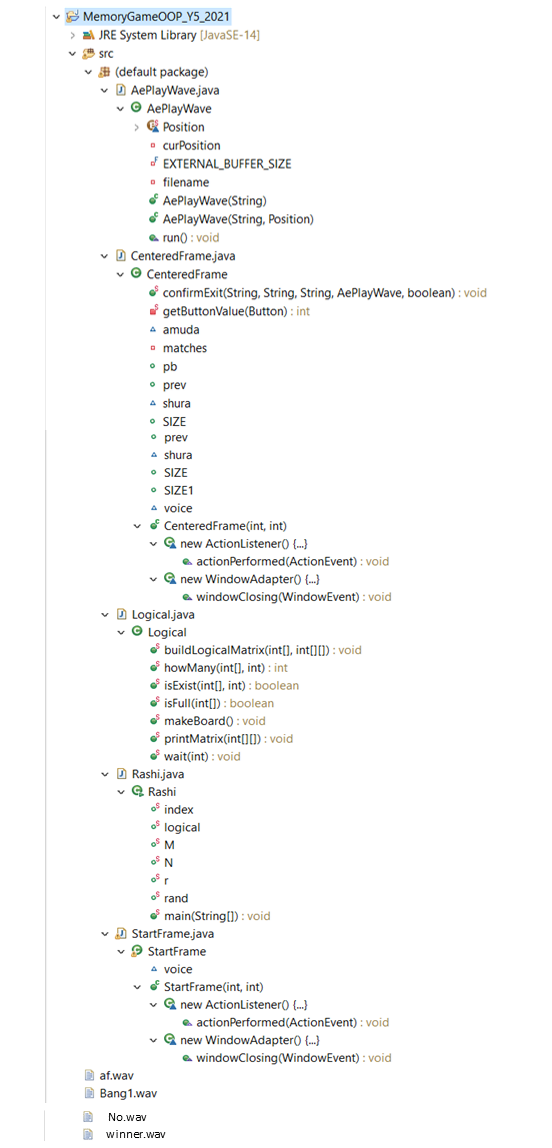
**חלון קופץ מספר 5:**

חלון זה קופץ לאחר שכל הזוגות נמצאו וחלון קופץ מספר 4, אשר מראה 100.0 אחוז נסגר, לאחר סגירתו של חלון זה הלוח של המשחק ייסגר אוטומטית. בשביל לסגור חלון זה יש ללחוץ על "אוקי" או על X, כמו שמראים החיצים.

**מדריך למפתח:**

מפת תוכנה:





: Rashiמחלקת

(המחלקת הראשית)

מחלקה זו יוצרת ומציגה את הלוח החדש. המחלקה פותחת את החלון הגרפי.

הגדרת משתנים:

יצירה של משתנים שונים שישמשו לאורך כל הקוד. **public** **static** **int** *N*; //shura

**public** **static** **int** *N*; //shura

**public** **static** **int** *M*; // amuda

**public** **static** **int** *r*; // מספר אקראי

**public** **static** **int** *index*=0;

**public** **static** **int** []*rand*;

**public** **static** **int** [][]*logical*;

שורה 3: יוצרת משתנה מסוג מספר שלם בשם N.

שורה 4: יוצרת משתנה מסוג מספר שלם בשם M.

שורה 5: יוצרת את משתנה r אשר מייצג מספר רנדומלי שלם.

שורה 6: יוצרת משתנה מסוג מספר שלם בשם index ומגדירה את גודלו כ0.

שורה 7: יוצרת מערך של מספרים שלמים בשם rand.

שורה 8: יוצרת מערך דו מימדי של מספרים שלמים בשם Logical.

main-

הפעולה הראשית מפעילה את הפעולה האחראית על יצירת המטריצה הלוגית. על ידי יצירת העצם

נוצר חלון גרפי חדש המשמש למשחק.

**public** **static** **void** main(String []args)

{

StartFrame start = **new** StartFrame(8, 10);

start.setVisible(**true**);

}

שורה 10: מגדירה את הפעולה כפעולה הראשית.

שורה 13: יוצאת חלון גרפי בעזרת סטרארט פריים.

שורה 14: הופכת את החלון הגרפי לנראה.

:Logicalמחלקת

המחלקה מכילה פעולות סטטיות אשר "אחראיות" על יצירת הממשק למשחק.

מחלקה זו "מנהלת" את ממשק המשחק הלוגי: בונה את בסיסי הנתונים (המטריצה הלוגית והמערך החד-ממדי) תוך בדיקת תקינותם הלוגית. התנאים הנבדקים: המטריצה הלוגית מלאה, המערך ה"מאחסן" את הזוגות מכיל בתוכו רק זוגות סדורים (לא מעל פעמיים למספר).

Wait-

הפעולה הזו מקבלת מספר מסוג שלם של מילי-שניות. "מפסיקה" את הקוד למשך מכפלת המספר המוזן בפונקציה בארבע מאות.

**public** **static** **void** wait(**int** miliSec)

{

**try** {

Thread.*currentThread*();

Thread.*sleep*(miliSec \* 400);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

**שורה 7:** מקבל פרמטר מסוג שלם.

**שורה 10-11:** מזמן את פונקציית התהליכון ו"מחכה" פרק זמן של מכפלת הפרמטר ב-400 (מילי-שניות).

**שורה 12-13:** על-ידי הבלוק הנוכחי, התוכנית בודקת אם נוצרה איזשהו שגיאה בפעולה הנוכחית.

printMatrix-

מקבלת מערך דו-ממדי של מספרים מסוג שלם. מדפיסה את הערכים של המערך הדו-ממדי.

**public** **static** **void** printMatrix(**int** [][]grid)

{

System.***out***.println();

**for**(**int** r=0; r<grid.length; r++)

{

**for**(**int** c=0; c<grid[r].length; c++)

System.***out***.printf("%-5d",grid[r][c]);

System.***out***.println(); // \n new line

}

}

**שורה 17:** מקבל מערך דו-ממדי של מספרים מסוג שלם.

**שורה 20-24:** עוברת על המערך ומדפיסה את ערכי לפי מיקום בשורה ובעמודה.

buildLogicalMatrix-

הפונקציה מקבלת מערך חד-ממדי של מספרים מסוג שלם, ומערך דו ממדי של מספרים מסוג שלם. עוברת על המערך הדו-ממדי ומשימה לכל איבר במערך הדו-ממדי, איבר שמהמערך החד ממדי.

**public** **static** **void** buildLogicalMatrix(**int** []a,**int** [][]grid)

{

**int** index=0;

**int** r;// = grid.length;

**int** c; // = grid[r].length;

**for**(r=0; r<grid.length; r++)

{

**for**(c=0; c<grid[r].length; c++)

{

grid[r][c]=a[index++];

//printMatrix(grid);

}

}

}

**שורה 28:** מקבל כפרמטר מערך חד ממדי עם מספרים מסוג שלם, ומערך חד ממדי עם מספרים מסוג שלם.

**שורה 29-31:** מוגדר משתנה המייצג את המיקום במטריצה ושני משתנים אשר מייצגים את מיקום השורה והעמודה במטריצה.

**שורה 32-34:** עוברת על המטריצה ומשימה באיבר את הערך במערך במיקום העולה על אחד ממיקום המשתנה ה"מאחסן" את השורה.

isFull-

מקבלת מערך חד ממדי של מספרים מסוג שלם. הפעולה על כל משתנה במערך עד לרגע שנמצא שאחד הערכים שווה לאפס, או שכבר נעבר על כל אורך המערך. במידה והפעולה מצאה שאיבר במערך ערכו שווה לאפס – יוחזר פסוק שקר. אחרת, יוחזר פסוק אמת.

**public** **static** **boolean** isFull(**int** []a)

{

**for** (**int** i=0; i<a.length; i++)

**if** (a[i]==0)

**return**(**false**);

**return**(**true**);

}

**שורה 43:** מקבלת מערך חד ממדי עם מספרים מסוג שלם.

**שורה 44-45:** עובר על המערך, ובודק: האם כל המקומות מלאים (שונים מאפס).

**שורה 46-47:** אם יש ערך אפס במערך יוחזר פסוק-שקר. אחרת, יוחזר פסוק אמת.

howMany-

מקבלת מערך חד ממדי של מספרים שלמים ומשתנה מסוג שלם. עוברת על המערך, ומחזירה את מספר הפעמים אשר המשתנה מופיע במערך.

**public** **static** **int** howMany(**int** []a, **int** lookFor)

{

**int** mone=0;

**for** (**int** i=0 ; i<a.length ; i++)

**if** (a[i]==lookFor)

mone++;

**return**(mone);

}

**שורה 51:** מקבלת מערך חד ממדי של מספרים מסוג שלם, פרמטר מסוג שלם.

**שורה 52:** מאפס משתנה המונה את מספר הפעמים שמספר מסוים המופיע במערך.

isExist-

הפעולה מקבלת מערך חד-ממדי של מספרים מסוג שלם ומספר מסוג שלם.

עוברת על המערך ומשווה כל אחד מאיברי המערך למשתנה. אם נמצאה התאמה- יוחזר פסוק-אמת. אם לא, יוחזר פסוק שקר.

**public** **static** **boolean** isExist(**int** []a, **int** lookFor)

{

**for** (**int** i=0 ; i<a.length ; i++)

**if** (a[i]==lookFor)

**return**(**true**);

**return**(**false**);

}

**שורה 60:** מקבלת כפרמטרים: מערך חד ממדי של מספרים שלמים, ומשתנה מסוג שלם.

**שורה 61-62:** עובר על המערך ובודק האם המשתנה שקיבלנו במערך.

**שורה 63-64:** אם האיבר במערך- יוחזר פסוק אמת. אחרת, פסוק שקר.

makeBoard-

הפעולה ה"אחראית" על בניית הלוח. עובר על המערך החד-ממדי המכיל את המספרים באופן אקראי, כל עוד המערך אינו מלא (משתמש בפונקציה האחראית על כך). משתמש בפונקציה מובנית של ג'אווה להגרלת מספרים אקראיים; שמטרתה ערך מסוים מבין טווח הערכים שבין אחד לעד מחצית מכמות הריבועים. אחר כך, הפעולה בודקת כמה פעמים מופיע המספר המתקבל במערך החד ממדי: במקרה בו מופיע אפס או אחד פעמים – מכניסה את המספר במקום הבא במערך. במקרה בו מופיע פעמיים, יישמע קטע קול ("אתה פויה!"). כאשר המערך החד ממדי מלא מופעלת הפונקציה האחראית על בנייתה של המטריצה מתוך המערך החד ממדי ומדפיסה את המטריצה לבסוף.

**public** **static** **void** makeBoard()

{

AePlayWave voice;

System.***out***.println("logical matrix At start:");

Logical.*printMatrix*(Rashi.*logical*); // MyPrint

System.***out***.println(Arrays.*deepToString*(Rashi.*logical*)+"\n"); // JavaPrint

System.***out***.println(Arrays.*toString*(Rashi.*rand*)); // JavaPrint

**while** (!(Logical.*isFull*(Rashi.*rand*)))

{

//System.out.println("Enter number between 1.."+((Rashi.N\*Rashi.M)/2)+":");

//Rashi.r=Rashi.kelet.nextInt();

Random rand = **new** Random();

Rashi.*r* = rand.nextInt(Rashi.*N*\*Rashi.*M*)/2;

Rashi.*r*++; // avoid zero!

//System.out.println("r is: "+Rashi.r);

**switch** (Logical.*howMany*(Rashi.*rand*,Rashi.*r*))

{

**case** 0:

**case** 1:Rashi.*rand*[Rashi.*index*++]=Rashi.*r*;

**break**;

**case** 2: // NOP!

System.***err***.println(Rashi.*r*+ " allready exist!");

voice=**new** AePlayWave("af.wav");

voice.start(); // run()

*wait*(1);

**break**;

**default**: // NOP!

**break**;

}

System.***out***.println(Arrays.*toString*(Rashi.*rand*)); // JavaPrint

}

System.***out***.println("Final: "+Arrays.*toString*(Rashi.*rand*)); //JavaPrint

Logical.*buildLogicalMatrix*(Rashi.*rand*,Rashi.*logical*);

System.***out***.println("logical matrix At end:");

Logical.*printMatrix*(Rashi.*logical*); // MyPrint

System.***out***.println(Arrays.*deepToString*(Rashi.*logical*)+"\n"); // JavaPrint

}

**שורה 68:** עובר על המערך החד ממדי המכיל את המספרים באופן אקראי, כל עוד המערך החד ממדי אינו מלא.

**שורה 70-71:** שם במיקום הבא במערך מספר אקראי בטווח שבין הערך של מחצית כמות המספרים במטריצה לאפס, "מקדם" את המספר באחד כל פעם (במטרה לוודא כי המספר אפס לא יהיה במערך; אפס מייצג מקום ריק).

**שורה 72:** בודקת את כמות הפעמים אשר הופיע המספר ה"מוגרל" כבר במערך.

**שורה 73-75:** אם המספר הופיע אפס או פעם אחת תוסיף את המספר במיקום הבא במערך, ותצא מבלוק הסוויץ'.

**שורה 76-81:** אם המספר כבר הופיע פעמיים, אז הוא יכנס למערך ויושמע קטע קול ("אתה פויה!") ותצא מהבלוק.

**שורה 82-83:** במקרה אחר, הפונקציה תצא מהבלוק.

: CenteredFrameמחלקת

המחלקה היורשת ממחלקת "פריים". תפקידה ליצור את לוח המשחק ולנהלו.

confirmExit:

הפעולה מוודאה שהשחקן באמת רצה לסגור את המשחק ושהוא לא בטעות לחץ על לאיקס.

**public** **static** **void** confirmExit(String question, String framei, String message, AePlayWave voice, **boolean** isMainFrame)

{

// To confirm closing window.

Frame f = **new** Frame();

Object[] options = {"Yes", "No"};

**int** n = JOptionPane.*showOptionDialog*(f, question, "Exit", JOptionPane.***YES\_NO\_OPTION***, JOptionPane.***QUESTION\_MESSAGE***, **null**, options, options[1]);

**boolean** flag=**false**;

**switch** (n) {

**case** 1:

flag = **true**;

**break**;

**default**:

flag = **false**;

**break**;

}

**if** (!flag)

{

System.***out***.println(framei+"Frame is Dead!");

**if** (isMainFrame)

{

JOptionPane.*showMessageDialog*(**null**, message);

voice=**new** AePlayWave("Bang1.wav");

voice.start(); // run()

Logical.*wait*(5);

}

**else**

Logical.*wait*(1);

System.*exit*(0);

}

}

**שורה 24:** פונקציה השייכת למחלקה זו. מקבלת כפרמטרים: שלוש מחרוזות- השאלה המופנית למשתמש, תחילתי של שמו, תוכן ההודעה שתופיע לאחר שנבחר האם לבטל את החלון. עצם קטע הקול. גם התנאי הבוליאני הבודק האם היא הפריים הראשי.

**שורה 27:** יוצר "פריים" חדש.

**שורה 28:** נוצר מערך חד ממדי שמכיל אובייקט המהווים את האפשרויות.

**שורה 29:** הודעה הקולטת שנלחץ (בהתאם לאפשרויות ובהתאם למיקומן).

**שורה 30:** מוגדר ערך בוליאני המהווה את איזה אפשרות נבחרה (אילו אפשרות במיקום 1 נבחרה אז זה לא יתקיים- כלומר, אין יציאה מהמסך).

**שורות 31-37:** בודקות אם המשתמש לחץ אפשרות במיקום 1, במערך החד ממדי המופיע לעיל, על כן או לא.

**שורה 39:** בודקת אם נלחץ כן (האפשרות במיקום 0).

**שורה 41:** מודפס שמו המלא של הפריים שנסגר (התראה על כך שהוא נסגר).

**שורה 42-44:** התנאי הבוליאני שבדק האם החלון הוא הראשי, אם הוא מתקיים מופיעה הודעה שהוזנה כפרמטר.

**שורות 46-45:** משמיעות את קול הבום הנשמע בזמן סגירת החלון.

**שורה 47:** מחכה 2 שניות.

**שורה 50:** אחרת, מחכה 0.4 שניות.

**שורות 51:** סוגר את החלון הנוכחי.

: CenteredFrame בנאי

הפעולה הבונה. מייצרת את החלון הגרפי של לוח המשחק. פונה לפעולה המחכה לפעולת סגירה שהמשתמש יבצע על הלוח, כמשתנה שולח פעולה של סגירת החלון הגרפי שתפקידה להפעיל קטע קול בעת סגירת החלון ("בום!", צליל ירייה) ולסגור את החלון של הלוח. לאחר כך, הפעולה מחלקת את הלוח לגודל סטטי. הפעולה עוברת על לוח המשחק ופונה אל המערך הדו-ממדי אשר מכיל את השמות המחרוזת ומכניסה לכל כפתור את הערך הנמצא באותו המיקום במטריצה כערך מחרוזת (השם המדובר). אחר ההוספת הכפתור, מגדירה לכפתורים שיצרה הפעולה ה"אחראית" על "מה יקרה" ברגע שנלחץ הכפתור המדובר. ברגע שנלחץ על הכפתור מופיעה על הודעה שבה מופיע התווית שהכפתור שנלחץ. אחרי שהפונקציה מכניסה את כל הכפתורים הפעולה מממשת את מה שבנתה לחלון גרפי (הנראה על פני מסך המחשב) ומגדירה לו גודל.

**public** CenteredFrame(**int** N, **int** M)

{

**this**.SIZE=N;

**this**.SIZE1=M;

Panel p=**new** Panel();

addWindowListener(**new** WindowAdapter()

{

**public** **void** windowClosing(WindowEvent e)

{

*confirmExit*("Want get out of this game?", "", "Frame is Dead!", voice, **true**);

}

});

p.setLayout(**new** GridLayout(SIZE,SIZE1));

**for** (shura=0 ; shura<SIZE ; shura++)

**for** (amuda=0 ; amuda<SIZE1 ; amuda++)

{

Button butt; // null null

butt=**new** Button();

butt.setBackground(Color.***CYAN***.brighter());

butt.getAccessibleContext().setAccessibleName(Rashi.*logical*[shura][amuda]+"");

p.add(butt);

// מאזין על כל כפתור

butt.addActionListener((ActionListener) **new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**int** buttonValue = *getButtonValue*(butt);

String buttonValueStr = String.*valueOf*(buttonValue);

**if** (prev == **null**) // the first value of the pair

{

butt.setBackground(Color.***WHITE***);

butt.setLabel(buttonValueStr);

prev = butt;

} **else** { // the second value of the pair

butt.setBackground(Color.***WHITE***);

**if** ((buttonValue == *getButtonValue*(prev)) && (prev != butt)) { // there is a match

butt.setLabel(buttonValueStr);

Logical.*wait*(1);

butt.setBackground(Color.***WHITE***);

matches++;

voice=**new** AePlayWave("match.wav");

voice.start(); // run()

Logical.*wait*(1);

System.***out***.println("match found");

butt.getLabel();

butt.setVisible(**false**);

prev.setVisible(**false**);

butt.setLabel(buttonValueStr);

**double** pr = (**double**)(matches)/((SIZE \* SIZE1) / 2);

JOptionPane.*showMessageDialog*(**null**, "matches found: "+(pr\*100)+"%");

**if** (matches == (SIZE \* SIZE1) / 2) {

JOptionPane.*showMessageDialog*(**null**, "You are the champion");

voice=**new** AePlayWave("winner.wav");

voice.start(); // run()

Logical.*wait*(4);

System.*exit*(0);

}

} **else** {

butt.setLabel(buttonValueStr);

Logical.*wait*(1);

butt.setLabel("");

prev.setLabel("");

voice=**new** AePlayWave("No.wav");

voice.start(); // run()

Logical.*wait*(4);

butt.setBackground(Color.***CYAN***.brighter());

prev.setBackground(Color.***CYAN***.brighter());

}

prev = **null**;

}

**if** (butt.isVisible())

System.***out***.println("button pressed: "+*getButtonValue*(butt));

}

});

}

add(p);

setSize(SIZE\*60,SIZE\*60);

}

**private** **static** **int** getButtonValue(Button butt) {

**return** Integer.*parseInt*(butt.getAccessibleContext().getAccessibleName());

}

}

**שורות 56-57**: מגדירות את האובייקט החדש.

**שורה 59**: יוצרת חלון ("פנל") חדש.

**שורה 60:** יוצרת "אירוע" חדש המכיל את הדברים בשורה הבאה.

**שורה 64:** משתמשת פעולה קונפיירם אגזיט על התנאים של החלון החדש.

**שורה 67:** יוצר רשת צירים. פונקציה ש"מצביעה" על ה"פנל" הגרפי. הפונקציה מקבלת אובייקט חדש: שהוא מקבל שני פרמטרים, שני הפרמטרים שווים ונקלטים מהמשתמש.

**שורות 69-140:** רצות על המערך הדו סיפרתי שבוא יש את הערכים שת הכפתורים.

**שורה 72:** מגדירה את הכפתור הקודם כנאל.

**שורה 73:** יוצרת כפתור חדש.

**שורה 74:** מגדירה את הצבע הרקע של הכפתור ציאן לטיפה יותר בהיר (באמצעות הפעולה בריטר שמבהירה את הצבע המקור שקיים בג'אווה)

**שורה 75:** שמה את הכפתור במקום שלו.

**שורה 76:** מוסיפה את הפתור לp.

**שורה 80:** מקבלת את ערך של הכפתור ומגדירה אותו כמשתנה.

**שורות 82-86:** שומרות את הערך של הפתור הראשון.

**שורה 84:** מגדירה את צבע הרקע של הכפתור כלבן.

**שורה 85:** מגדירה את ערך הכפתור כ-buttonValueStr.

**שורה 86:** מגדירה את prev ככפתור.

**שורה 88:** מגדירה את רקע הכפתור השני לבן.

**שורה 89:** בודקת אם הערך של שני הכפתורים שווה.

**שורה 90:** מגדירה את ערך הכפתור השני.

**שורה 91:** גורמת לקוד לחקות חלקיק שניה.

**שורה 92:** מגדירה את הרקע של החלון עצמו לבן

**שורה 94:** מעלה את matches באחד.

**שורה 96:** מגדירה קטע קול חדש.

**שורה 97:** משמיע את קטע קול match.wav.

**שורה 98:** גורמת לקוד לחקות חלקיק שניה.

**שורה 100:** מדפיסה "match found".

**שורה 102:** מגדירה את הטווית של הכפתור כנאל.

**שורה 104:** מסתירה את הכפתור

**שורה 105:** מסתירה את הכפתור הנימצא ב- prev.

**שורה 107:** מגדירה את הטווית של הכפתור כ- buttonValueStr.

**שורה 108:** מחשבת כמה אחוזים מהלוח המשתמש פתר ושומרת את התוצאה ב- .pr

**שורה 109:** יוצרת חלון קופץ המציג כמה אחוזים נפתרו מהלוח.

**שורה 110:** בודקת עם כל החלון פתור.

**שורה 111:** יוצרת חלון קופץ שאומר " You are the champion".

**שורה 113:** מגדירה קטע קול חדש

**שורה 114:** משמיע את קטע הקול winner.wav.

**שורה 115:** גורמת לפעולה לחקות 4 חלקי שניה.

**שורה 117:** סוגר את החלון הראשי.

**שורה 119:** במידה והכפתורים לא שווים.

**שורה 120:** מגדירה את הטווית של הכפתור כ- buttonValueStr.

**שורה 121:** גורמת לקוד לחקות חלקיק שניה.

**שורה 122:** מגדירה את הטווית של הכפתור כנאל

**שורה 123:** מגדירה את הטווית של prev כנאל.

**שורה 125:** מגדירה קטע קול חדש

**שורה 126:** מנגנת את קטע הקול No.wav

**שורה 127:** גורמת לקוד לחקות 4 חלקיקי שינה.

**שורה 129:** מגדירה את הרקע של הכפתור בצבע ציאן.

**שורה 130:** מגדירה את הרקע של prev בצבע ציאן.

**שורה 132:** מגדירים את prev כנאל.

**שורה 134:** בודקת אם ניתן לראות את הכפתור.

**שורה 135:** מדפיסה "הכפתור הנילחץ" ואת המספר של הכפתור הנלחץ.

**שורה 141:** מוסיף את החלון ("פנל"). כך, שמופיע על-גבי המסך.

**שורה 142:** מגדירה את הגודל של החלון הראשי.

getButtonValue-

מגדירה את הערך של הכפתור.

**private** **static** **int** getButtonValue(Button butt) {

**return** Integer.*parseInt*(butt.getAccessibleContext().getAccessibleName());

}

**שורה 145:** הפונקציה "שייכת" למחלקה שבה היא נמצאת. מקבלת כפרמטר את הכפתור המוזן בה.

**שורה 146:** מחזירה את הערך של הכפתור כמספר שלם, כלומר את שמו של הכפתור.

מחלקת StartFrame :

המחלקה היורשת ממחלקת "פריים". תפקידה להתחיל את יצירת שני המסכים של המשחק: המסך הראשון, הוא המסך האחראי על בחירת גודלו של הלוח. השני, מפעיל את מחלקה של פריים המשחק. כך, מתחיל המשחק לפעול.

-StartFrame בנאי

הפעולה הבנאית. המקבלת שני פרמטרים מסוג שלם: הם מידות הלוח הנוצר (אשר כל אחד מהם בסוף פעולה זו מוכפל ב-60).

**public** StartFrame(**int** size0, **int** size1) {

Panel p=**new** Panel();

addWindowListener(**new** WindowAdapter()

{

**public** **void** windowClosing(WindowEvent e)

{

CenteredFrame.*confirmExit*("Don't you want to play?", "Start ", "canceling game", **null**, **true**);

}

});

p.setLayout(**new** GridLayout(6,6));

p.setBackground(Color.***YELLOW***.brighter()); // Color it's static final variable from class Color. (pay attention - delete afterwards)

JLabel label1 = **new** JLabel("Start");

JLabel label2 = **new** JLabel("-----");

p.add(label1);

p.add(label2);

**for**(**int** i =0; i<4; i++) {

Button b = **new** Button();

p.add(b);

**if**(i==0)

b.setLabel("2X2");

**else** **if**(i==1)

b.setLabel("4X4");

**else** **if**(i==2)

b.setLabel("6X6");

**else**

b.setLabel("8X8");

b.addActionListener((ActionListener) **new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

Rashi.*N*=Character.*getNumericValue*(b.getLabel().charAt(0)); //first num

Rashi.*M*=Character.*getNumericValue*(b.getLabel().charAt(2));//second num

JOptionPane.*showMessageDialog*(**null**, "Opening board window");

Logical.*wait*(1);

dispose();

Rashi.*rand* = **new** **int**[Rashi.*N*\*Rashi.*M*];

Rashi.*logical*=**new** **int**[Rashi.*N*][Rashi.*M*];

Logical.*makeBoard*();

CenteredFrame board = **new** CenteredFrame(Rashi.*N*,Rashi.*M*);

board.setVisible(**true**);

}

});

}

add(p);

setSize(size0\*60,size1\*60);

}

**שורה 12:** בנאי (המופיע ראשון). בונה אובייקט בשם המחלקה הנ"ל, עם מידה אחת (מסוג שלם) ומידה שניה (מסוג שלם).

**שורה 13:** יוצרת חלון חדש.

**שורה 15-17:** יוצרת "מאזין" על החלון ל"אירוע" חדש המכיל את הדברים בשורה הבאה. (סגירת החלון, לחיצה על כפתור האיקס)

**שורה 19:** משתמשת פעולה קונפיירם אגזיט על התנאים של החלון החדש.

**שורה 23:** יוצר רשת צירים. פונקציה ש"מצביעה" על ה"פנל" הגרפי.

**שורה 24:** מגדירה את הצבע הרקע של המסך שיפתח לצהוב לטיפה יותר בהיר (באמצעות הפעולה בריטר שמבהירה את הצבע המקור שקיים בג'אווה).

**שורה 25-26:** יוצרות שני עצמים שהינם המכילים טקסט.

**שורה 27-28:** מוסיף את רכיבים אלו לפנל.

**שורה 30-34:** עובר ארבע פעמים, על-מנת ליצור ארבעה כפתורים. ביצירה של כל כפתור, הוא מוסף ישירות לפנל. (בהתאם לכל פעם נוצר אחד מהגדלים המבוקשים)

**שורה 43-44:** מוסיף "מאזין" אשר מבצע פעולות מסוימות כאשר הכפתור נלחץ. (נוצר לכל כפתור בלוח, "מאזין" זה)

**שורה 45-46:** מגדיר את המשתנים N ו-M (מהמחלקה הראשית) כגודל הכפתור הנבחר (באמצעות פונקציה של האובייקט "תו" שלוקח ערך של תו- ואם היא מספר תחזור כמספר שהתו מייצג). למשל, אם ילחץ "2X2" ערכם של שניהם יהיה שווה לשתיים.

**שורה 47:** מופיע חלון קופץ שמודיע על פתיחת חלון של לוח המשחק.

**שורה 48:** מחכה 0.4 שניות (השהייה קצרה).

**שורה 49:** מעלים את חלון הבחירה.

**שורה 50-51:** מגדיר את גודלם של שני המערכים שהוגדרו במחלקה הראשית (מערך חד ממדי ומערך דו ממדי של מכילים ערכים של מספרים שלמים). גודל המערך החד ממדי יהיה בגודל מכפלת המשתנים N ו-M. המערך הדו ממדי יהיה גודל שורתו יהיה N, מספר העמודות יהיה M.

**שורה 52:** מפעילה את הפעולה "האחראית" על יצירת המטריצה הלוגית (בהתאם לתנאים שהלוח "חייב לעמוד" בהם).

**שורה 53:** יוצר אובייקט של מחלקה של עצם מסך המשחק, ומפעיל את הפעולה הבנאית: היא מקבלת את N ו-M.

**שורה 54:** הפעולה שפועלת על עצם זה. מאפשרת את הצגה של לוח המשחק.

**שורה 58:** מוסיף את החלון ("פנל"). כך, שמופיע על-גבי המסך.

**שורה 59:** קובע את מידת החלון (אורך ורוחב). כך, שהפרמטרים של הפעולה מקבלת מהווים את מידות החלון (כפול 60) בהתאמה.

: AePlayWaveמחלקת

המחלקה ה"יורשת" ממחלקת התהליכון. אחראית על בחירתם של קבצי אודיו, קריאתם וניגונם.

-AePlayWaveבנאי

הפעולה הבנאית. מקבלת את קובץ האודיו שנרצה לנגן, וגם את המיקום ("הפוזיציה"): ימין, שמאל או רגיל.

**public** AePlayWave(String wavfile)

{

filename = wavfile;

curPosition = Position.***NORMAL***;

}

**public** AePlayWave(String wavfile, Position p)

{

filename = wavfile;

curPosition = p;

}

**שורה 23-27:** בנאי (המופיע ראשון). בונה אובייקט קובץ אודיו עם שם שמקבל וposition שתמיד normal.

**שורה 29-33:** בנאי (המופיע שני). בונה אובייקט קובץ אודיו עם שם, שמקבל וposition שמקבל.

run-

הפעולה "אחראית" על פתיחת קובץ האודיו, טעינתו וניגונו. היות והקובץ הוא חיצוני, הפונקציה "בודקת" כי אין שגיאות בהעלאתו וטעינתו של הקובץ. במידה ויש שגיאה, היא "נתפסת" והפעולה מסתיימת (בכדי למנוע שגיאה שתסיים את המשחק (טרם ביצועו).

**public** **void** run()

{

File soundFile = **new** File(filename);

**if** (!soundFile.exists())

{

System.***err***.println("Wave file not found: " + filename);

**return**;

}

AudioInputStream audioInputStream = **null**;

**try** {

audioInputStream = AudioSystem.*getAudioInputStream*(soundFile);

} **catch** (UnsupportedAudioFileException e1) {

e1.printStackTrace();

**return**;

} **catch** (IOException e1) {

e1.printStackTrace();

**return**;

}

AudioFormat format = audioInputStream.getFormat();

SourceDataLine auline = **null**;

DataLine.Info info = **new** DataLine.Info(SourceDataLine.**class**, format);

**try** {

auline = (SourceDataLine) AudioSystem.*getLine*(info);

auline.open(format);

} **catch** (LineUnavailableException e) {

e.printStackTrace();

**return**;

} **catch** (Exception e) {

e.printStackTrace();

**return**;

}

**if** (auline.isControlSupported(FloatControl.Type.***PAN***)) {

FloatControl pan = (FloatControl) auline

.getControl(FloatControl.Type.***PAN***);

**if** (curPosition == Position.***RIGHT***)

pan.setValue(1.0f);

**else** **if** (curPosition == Position.***LEFT***)

pan.setValue(-1.0f);

}

auline.start();

**int** nBytesRead = 0;

**byte**[] abData = **new** **byte**[EXTERNAL\_BUFFER\_SIZE];

**try** {

**while** (nBytesRead != -1) {

nBytesRead = audioInputStream.read(abData, 0, abData.length);

**if** (nBytesRead >= 0)

auline.write(abData, 0, nBytesRead);

}

} **catch** (IOException e) {

e.printStackTrace();

**return**;

} **finally** {

auline.drain();

auline.close();

}

}

}

**שורה 38-41:** יוצר אובייקט חדש של קובץ ומכניס את שמו. אם הקובץ אינו "נמצא"- אז הפעולה תסתיים.

**שורה 45-54:** הפעולה מכניסה את הקובץ לאובייקט מסוג "AudioInputStream" אשר מטפל בקבצי אודיו. במקרה בו יש שגיאה בפורמט הקובץ (הפורמט אינו מזוהה), או במקרה בו יש שגיאת קריאה/ כתיבה של הקובץ – אז, הפעולה תסתיים.

**שורה :56-78** הפעולה יוצרת גישה אל כרטיס הקול של המחשב ופותחת ערוץ מולו לצורך הניגון. במידה וערוץ הקול אינו זמין, מסיבה כלשהי - הפעולה תמה. גם במצב שגיאה אחר, הפעולה תסתיים.

**שורה :80-96** הפעולה מנגנת את קובץ האודיו. הקובץ נקרא בחלקים של byte כגודל הbuffer ומנגנת את מה שנקרא לתוך הbuffer וחוזרת על התהליך עד שכל קובץ האודיו נקרא. במקרה בו תהיה שגיאה בקריאה/ כתיבה של הקובץ – אז, הפעולה תסתיים. לסיום, הערוץ מול כרטיס הקול "ינוקה" וייסגר.