פנטגו

<u>שם הפרויקט:</u> פנטגו (Pentago)

<u>שם התלמיד:</u> תמיר דוידי

<u>שם המורה:</u> זאב וולפר

תוכן עניינים

מבוא.....3

מסכי המשחק.....4-5

חלוקה למחלקות.....6

תיאור המחלקות.....6-7

תרשימי UML של המחלקות.....8-20

היוריסטיקה.....21-22

רפלקציה.....23

קוד מתועד.....24-51

מבוא

המשחק שאני בחרתי לכתוב בשפת C# הוא **פנטגו** (PENTAGO).

תיאור כללי

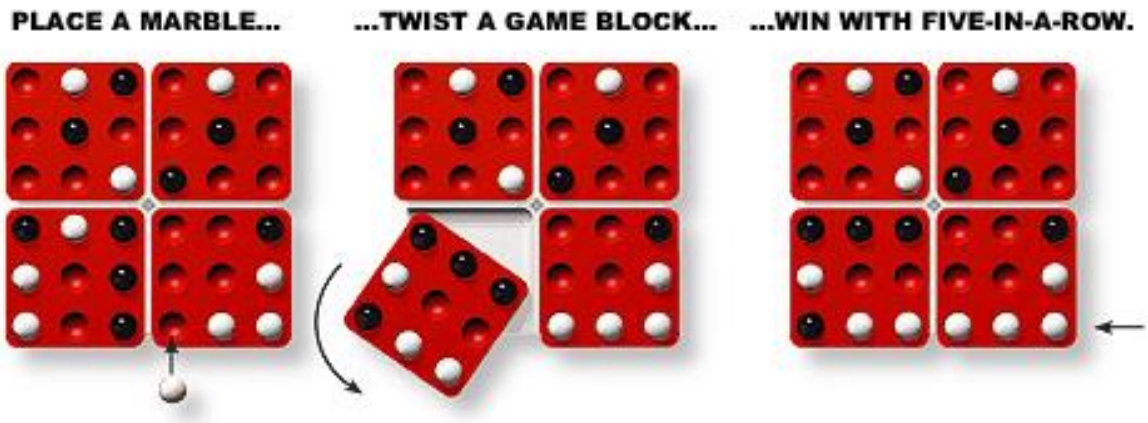
פנטגו הוא משחק לוח לשני שחקנים, מסוג משחקי חשיבה. הוא מבוסס על משחק איקס עיגול אך מורכ ביותר. ייחודו של המשחק בכלליו הפשוטים מחד, ובמחשבה המורכבת מאידך.
אורכו של המשחק הוא כ – 15 דקות.
המשתמש משתמש באבנים לבנות, ואילו המחשב משתמש באבנים שחורות.

לוח המשחק

לוח המשחק מכיל 36 אבני משחק - 18 לבנות ו-18 שחורות, וכן לוח של 6X6 משבצות, כשהלוח מורכב מארבעה לוחות של 3X3, וכל לוח ניתן לסיבוב בנפרד.

הוראות המשחק

המשתתף שם אבן משחק. ומסובב את אחד מחלקי הלוח בתשעים מעלות ימינה או שמאלה, לפי בחירתו– זאת באמצעות לחיצה על אחד משמונת הכפתורים המאפשרים את סיבוב חלקי הלוח. לאחר מכן מגיע תור המחשב שגם הוא ממקם אבן בלוח ומסובב את אחד מחלקי הלוח 90 מעלות ימינה או שמאלה לפי בחירתו.המנצח הוא זה שסיים שורה של 5 אבני משחק, במאוזן במאונך או באלכסון.



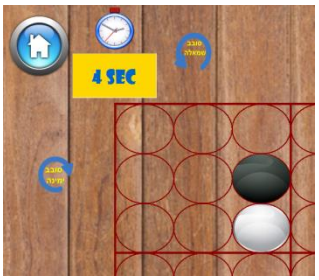
מצב התחלתי

בתפריט הראשי (הטופס הראשי) מקלידים את שם השחקן, לאחר מכן לוחצים על כפתור ההפעלה של המשחק. ומתחילים לשחק!

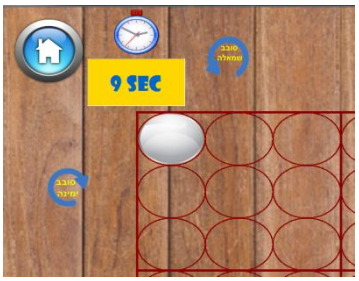
כפתור ההפעלה

הקלקה על כפתור זה מעבירה את השחקן לטופס המשחק (form game) ובו מופיע לוח 6 על 6 ריק ומרגע זה המשחק מתחיל.

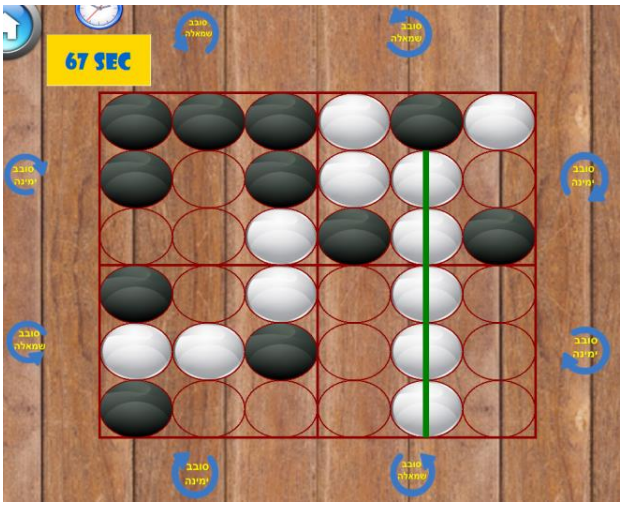
2.דוגמה למצב לאחר הקלקה
על לחצן סיבוב ופעילות הבינה



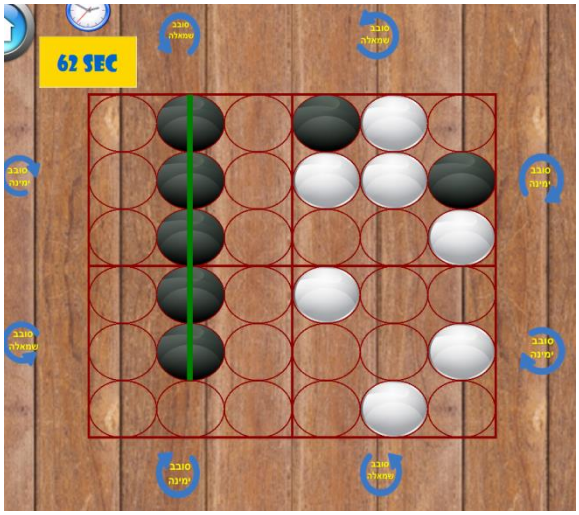
1.דוגמא להצבת אבן בלוח:



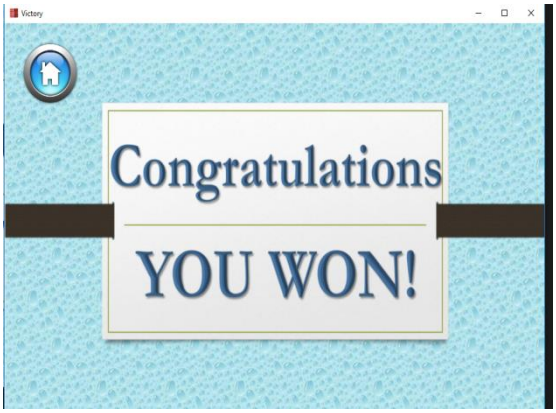
דוגמה למצב ניצחון המשתמש



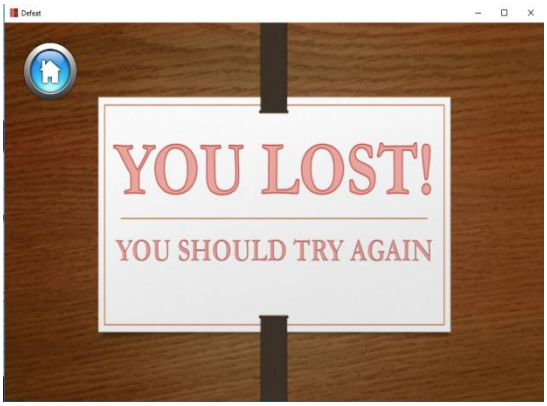
דוגמה למצב ניצחון המחשב



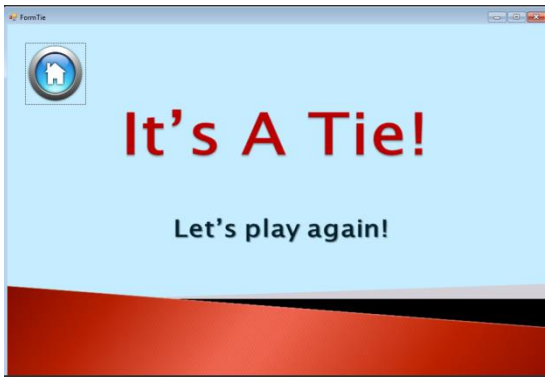
טופס ניצחון



טופס הפסד



טופס תיקו



טופס הוראות

טופס טבלת שיאים

TIME	PLAYER
25	TAMIR
37	TOMER
44	LIRAN
49	EDEN
53	HEN
55	BAR
61	OR

מסכי המשחק

תחילה המשתמש מופנה למסך "תפריט ראשי", ממנו הוא יכול לפנות אל המסכים: טבלת שיאיוהוראות. בתפריט הראשי מופיע כפתור "Exit" שלחיצה עליו מביאה ליציאה מהמשחק. כמו כן, לאחר הקלדת שם השחקן במקום המיועד יש ללחוץ על "כפתור ההפעלה של המשחק" שלאחר לחיצה עליו ניתן לעבור למסך המשחק. בנוסף לכך, כאשר מסתיים המשחק, מופיע מסך המודיע על ניצחון, הפסד, או תיקו בהתאמה למצב.
בכל אחד מן המסכים הללו מופיע כפתור המאפשר לחזור למסך "התפריט הראשי. בתמונות מעלה ניתן לראות מסכים אלו.

מצב סופי

המשחק מסתיים לאחר שאו המשתמש או המחשב הצליח למקם חמש מאבני המשחק שלו בשורה, בטור, או באלכסון. אם לא המחשב ולא המשתמש ניצחו, כל התאים תפוסים ולא ניתן לבצע תור, מוכרז תיקו. לאחר שהסתיים המשחק יוצג מסך נצחון, הפסד או תיקו (בהתאם ללוח). ממסך התוצאה (ניצחון, הפסד או תיקו) ניתן לעבור באמצעות לחיצה על כפתור לתפריט הראשי ומשם ניתן לעבור באמצעות כפתור נוסף לטבלת השיאים. בטבלה זו יופיעו שמותיהם של המשתמשים שהצליחו לנצח את המחשב בזמן הקצר ביותר והזמנים שלקח לכל אחד מהם, מהנמוך לגבוה.

חלוקה למחלקות

הפרויקט מחולק למחלקות במטרה לייעל את תהליך כתיבת הקוד, למנוע כפילויות וליצור סדר וארגון של המידע הרב הנמצא בתוכנה. לכל מחלקה יש תפקיד ייחודי שלשמה היא נוצרה, והיא מכילה פעולות ותכונות שבאמצעותה המחלקה מבצעת את תפקידה.

תיאור המחלקות

המחלקות שבהן השתמשתי בתהליך בניית המשחק:

- **Cell-** מחלקה המכילה את כל המידע המתאר תא בודד בו ניתן להציב אבן.כמו כן, מכילה פעולות פנימיות שבאמצעותן התא מממש את תפקידו.

- **Quarter -** מחלקה המכילה את כל המידע המתאר רבעון (אחד מארבעת חלקי הלוח), מכיל בין היתר מערך דו מימדי שמכיל את כל התאים (Cells) שבו וכן פעולות פנימיות שמסייעות לו למלא את תפקידו.

- **Board -** מחלקה המכילה את כל המידע המתאר לוח. מכילה בין היתר מערך דו מימדי שמכיל את ארבעת הרבעונים שבו, וכן פעולות פנימיות שמסייעות לו למלא את תפקידו ולבצע פעולות הנוגעות אליו כשם הצבת אבן בדיקת נצחון וכדומה.

- **ArtificialInteligence -** מחלקה המכילה את כל המידע הדרוש למחשב כדי לחשב את הצעד המיטבי, זה שהכי מקרב אותו אל הניצחון. כמו כן המחלקה מכילה פעולות המאפשרות לה לבצע את מהלך זה.

- **RotationArrow -** מחלקה המכילה את כל הפעולות והמידע הדרושים לכפתורי הסיבוב של חלקי הלוח כדי לבצע את תפקידם.

- **Score -** מחלקה המכילה את כל המידע המתאר תוצאת ניצחון של המשתמש במשחק: הזמן שלקח עד שהמשתמש ניצח, ושם השחקן. באמצעות מידע זה טופס ה"טבלת שיאים" ינהל את רשומות השיאים.

- **PlacingOption -** מחלקה המכילה את כל המידע המתאר מהלך אפשרי בלוח: התא הנבחר, וכן הניקוד של הלוח אם יבוצע המהלך.

- **PlacingOptions -** מחלקה המכילה רשימה של מהלכים אפשריים לביצוע וכן פעולה שמוצאת את המהלך המיטבי מביניהם.

- **LineSequence -** מחלקה המכילה את המידע על רצפים מאותו צבע בשורה אחת: אורך הרצף והאם חסום מצידיו.

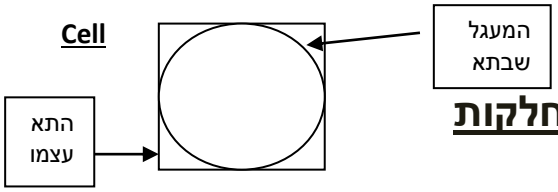
- **ToorSequence -** מחלקה המכילה את המידע על רצפים מאותו צבע בטור אחד: אורך הרצף והאם חסום מצידיו.

- **SlantLeftSequence -** מחלקה המכילה את המידע על רצפים מאותו צבע באלכסון שמאלי (נטוי לשמאל) אחד: אורך הרצף והאם חסום מצידיו.

- **SlantRightSequence -** מחלקה המכילה את המידע על רצפים מאותו צבע באלכסון ימני (נטוי לימין) אחד: אורך הרצף והאם חסום מצידיו.

טפסים

- **MainMenu –** התפריט הראשי של המשחק ממנו ניתן לגשת אל ההוראות, טבלת השיאים והמשחק עצמו.
- **FormGame –** טופס המכיל את המשחק עצמו (לוח המשחק והחיצים).
- **FormVictory –** טופס המציג הודעת ניצחון בפני המשתמש במידה שניצח.
- **FormDefeat** – טופס המציג הודעת הפסד בפני המשתמש במידה שהפסיד.
- **RecordsForm** – טופס המכיל את טבלת השיאים במשחק.
- **FormTie** – טופס המציג הודעה כי המשחק הסתיים והתוצאה היא תיקו כאשר זה המצב.
- **FormInstructions** – טופס המכיל את הוראות המשחק.



תרשימי UML של המחלקות

תכונות Cell	
`Status status`	משתנה ששומר את תכולת התא: האם ריק, יש בו אבן שחורה או לבנה
`int x`	ערך האיקס של הקצה שמאלי עליון של התא
`int y`	ערך הוואי של הקצה שמאלי עליון של התא
`int width`	רוחב התא
`int height`	גובה התא
`int radius`	רדיוס העיגול החסום בתא
`Point center`	מרכז העיגול החסום בתא

פעולות Cell	
`public Cell(int x, int y, int width, int height)`	פעולה בונה לתא
`public Cell(Cell cell)`	פעולה בונה מעתיקה
`public void Draw(Graphics g, Pen pen)`	מציירת את המסגרת של התא
`public bool IsInCircle(int x, int y)`	בודקת האם נקודה נמצאת בתוך המעגל החסום בתא
`public double Distance(int x1, int y1, int x2, int y2)`	מחזירה את המרחק בין שתי נקודות
`public void PlaceBlack(Graphics g)`	ממקמת אבן שחורה בתא
`public void PlaceWhite(Graphics g)`	ממקמת אבן לבנה בתא
`public bool PlaceBall(Graphics g, Board.Turn turn)`	ממקמת אבן בתא: מחזירה שקר כאשר הלחיצה על תא שיש בו אבן. מוודאת שהתא ריק לפני הצבת אבן חדשה, ומחזירה אמת כאשר התא ריק.
`public void DrawStatus(Graphics g)`	משחזרת את תכולת התא לאחר שנמחק
`public bool Inside(int x, int y)`	בודקת אם נקודה נמצאת בתוך התא

תכונות Quarter	
`int x`	ערך האיקס של הקצה שמאלי עליון של הרבעון
`int y`	ערך הוואי של הקצה שמאלי עליון של הרבעון
`int width`	רוחב הרבעון
`int height`	גובה הרבעון
`Cell[,] nineCells`	מערך דוד מימדי של תשעת התאים שברבעון
`FormGame formGame`	טופס המשחק
`Cell.Status[,] nineCellsStatus`	מערך דו מימדי של תכולת תשעת התאים ברבעון

Quarter פעולות	
public Quarter(int x, int y, int width, int height, FormGame formGame)	פעולה בונה לרבעון
public Quarter(Quarter quarter)	פעולה בונה מעתיקה
public void Draw(Graphics g, Pen penBoardFrame,Pen penCells)	מציירת את המסגרת של הרבעון ואת התאים שבו
public void DrawStatus(Graphics g)	ציור תכולת תאי הרבעון
public void SetNineCellsStatus()	עדכון ערכי תכולת תשעת התאים ברבעון
public void Rotate90DegreesLeft()	מסובבת את הרבעון 90 מעלות שמאלה
public void Rotate90DegreesRight()	מסובבת את הרבעון 90 מעלות ימינה
public bool MouseClick(Graphics g, int x0, int y0, Board.Turn turn)	הפעולה מחזירה אמת כאשר הלחיצה נעשתה על מקום שבו תא ריק. מחזירה שקר כאשר הלחיצה על תא שיש בו כבר כדורית. בודקת באיזה תא התרחשה הלחיצה ומעבירה אליו את הטיפול בלחיצה.
public bool IsInCellCircle(int x0, int y0)	מחזירה אמת כאשר הלחיצה הייתה בתוך המעגל החסום באחד מן התאים ברבעון. מחזירה שקר כאשר הלחיצה הייתה בתוך אחד התאים אך לא בתוך המעגל שבו.
public bool Inside(int x, int y)	בודקת אם נקודה נמצאת בתוך הרבעון

Board תכונות	
int x	ערך האיקס של הקצה שמאלי עליון של הלוח
int y	ערך הוואי של הקצה שמאלי עליון של הלוח
int width	רוחב הלוח
int height	גובה הלוח
Quarter[,] quarters	מערך דו מימדי של ארבעת חלקי הלוח - הרבעונים
Turn turn	פרמטר שמראה של מי התור - שלי או של המחשב או שהמשחק נגמר
TurnClick turnClick	פרמטר שמראה אם עכשיו צריכה להגיע לחיצה על תא בלוח או על חץ סיבוב
Cell[,] allCells;	מערך דוד מימדי של כל התאים בלוח
RotationArrow rightUpTurnRight	חץ סיבוב ימין למעלה שמסובב ימינה
RotationArrow rightUpTurnLeft	חץ סיבוב ימין למעלה שמסובב שמאלה
RotationArrow rightDownTurnRight	חץ סיבוב ימין למטה שמסובב ימינה
RotationArrow rightDownTurnLeft	חץ סיבוב ימין למטה שמסובב שמאלה
RotationArrow leftDownTurnRight	חץ סיבוב שמאל למטה שמסובב ימינה
RotationArrow leftDownTurnLeft	חץ סיבוב שמאלה למטה שמסובב שמאלה
RotationArrow leftUpTurnRight	חץ סיבוב שמאל למעלה שמסובב ימינה
RotationArrow leftUpTurnLeft	חץ סיבוב שמאל למעלה שמסובב שמאלה

Graphics g	משתנה הגרפיקה
Pen penBoardFrame	עט שמצייר את מסגרת הלוח
Pen penCells	עט שמצייר את מסגרת התאים בלוח
Pen penWin	עט שמצייר את קו הניצחון
FormVictory formVictory	טופס ניצחון
FormDefeat formDefeat	טופס הפסד
FormTie formTie	טופס תיקו

פעולות Board	
public Board(int x, int y, int width, int height, FormGame formGame, Graphics g, Pen penBoardFrame, Pen penCells, FormVictory formVictory, FormDefeat formDefeat, FormTie formTie)	פעולה בונה לוח
public Board(Board board)	פעולה בונה מעתיקה
public void Draw(Graphics g)	מציירת את הלוח
public void MouseClick(Graphics g, int x0, int y0)	מנהל אירוע לחיצה על הלוח
public bool IsTie()	פעולה שבודקת אם מצב הלוח הוא תיקו
public void DrawStatus(Graphics g)	מציירת את מצב בלוח לאחר מחיקת הלוח
public bool IsFiveEqualStatus(Cell c1, Cell c2, Cell c3, Cell c4,Cell c5)	האם חמישה תאים זהים בתכולתם - האם כולם שחורים או כולם לבנים
public void DrawVictoryLineInLine(Cell cStart, Cell cEnd)	מצייר את קו הניצחון בשורה
public void DrawVictoryLineInColumn(Cell cStart, Cell cEnd)	מצייר את הקו ניצחון בטור
public void DrawVictoryLineInSlantLeft(Cell cStart, Cell cEnd)	באלכסון שמאלי מצייר את הקו ניצחון
public void DrawVictoryLineInSlantRight(Cell cStart, Cell cEnd)	באלכסון ימני מצייר את הקו ניצחון
public bool IsVictoryInOneLineSpecificColor(Cell.Status statusCompetitor, int IndexLine)	האם יש ניצחון בשורה אחת בצבע מסוים
public bool IsVictoryInAllLines(Cell.Status statusCompetitor)	האם יש ניצחון בצבע מסוים באחת מן השורות בלוח
public bool IsVictoryInOneColumnSpecificColor(Cell.Status statusCompetitor, int IndexColumn)	האם יש ניצחון בטור אחד בצבע מסוים
public bool IsVictoryInAllColumns(Cell.Status statusCompetitor)	האם יש ניצחון באחד מן הטורים בלוח בצבע מסוים
public bool IsVictoryLeftMainUpSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון השמאלי הראשי העליון
public bool IsVictoryLeftMainDownSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון השמאלי ראשי תחתון
public bool IsVictoryLeftDownSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון השמאלי משני תחתון
public bool IsVictoryLeftUpSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון השמאלי משני עליון
public bool IsVictoryLeftSlant(Cell.Status statusCompetitor)	האם יש ניצחון באחד מהאלכסונים השמאליים הנוטים שמאלה
public bool IsVictoryRightMainUpSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון הראשי ימני עליון
public bool IsVictoryRightMainDownSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון הראשי ימני תחתון
public bool IsVictoryRightDownSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון המשני ימני תחתון
public bool IsVictoryRightUpSlant(Cell.Status statusCompetitor)	האם יש ניצחון באלכסון המשני ימני עליון
public bool IsVictoryRightSlant(Cell.Status statusCompetitor)	האם יש ניצחון באחד מהאלכסונים הנוטים ימינה
public bool IsVictoryAllSlants(Cell.Status statusCompetitor)	האם יש ניצחון באחד מן האלכסונים בכלל
public bool IsTotalVictory(Cell.Status statusCompetitor)	האם יש ניצחון או בשורה או בטור או באלכסון

public void WinLoseTieForms(int time, string playerName)	הפעולה בודקת אם הסתיים המשחק ובהתאם פותחת טופס ניצחון /הפסד או תיקו
public int RecursiaLengthLine(int i, int j, Cell.Status status)	**רקורסיה שמחשבת אורך של רצף שורה בצבע מסוים שיוצא מתא מסוים**
public LineSequence MaxFreeSequanceLine(Cell.Status status)	מחזיר את הרצף הכי ארוך שבשורה בצבע מסוים
public int RecursiaLengthToor(int i, int j, Cell.Status status)	**רקורסיה שמחשבת אורך של רצף טור שיוצא מתא מסוים בצבע מסוים**
public ToorSequence MaxFreeSequanceToor(Cell.Status status)	מחזיר את הרצף הכי ארוך שבטור
public int RecursiaLengthLeftSlant(int i, int j, Cell.Status status)	**רקורסיה שמחשבת אורך של רצף אלכסון שמאלי בצבע מסוים שיוצא מתא מסוים**
public int RecursiaLengthRightSlant(int i, int j, Cell.Status status)	**רקורסיה שמחשבת אורך של רצף אלכסון ימני בצבע מסוים שיוצא מתא מסוים**
public SlantLeftSequence MaxFreeSequenceSlantLeft(Cell.Status status)	מחזיר את הרצף הכי ארוך באלכסון שמאלי
public SlantRightSequence MaxFreeSequenceSlantRight(Cell.Status status)	מחזיר את הרצף הכי ארוך באלכסון ימני
public void CopyQuartersToAllCells()	מעדכן את המערך של כל תאי הלוח מהנתונים שברבעונים
public bool Inside(int x, int y)	בודק אם נקודה בתוך הלוח

תכונות **Score**	
int time	זמן שלקח למשתמש לנצח
string playerName	שמו של המשתמש

פעולות **Score**	
public Score(int time, string playerName)	פעולה בונה תוצאה

תכונות **PlacingOption**	
int rate	הניקוד למהלך - אפשרות
int i	ערך I של המקום בו נמקם אבן בלוח
int j	ערך J של המקום בו נמקם אבן בלוח

פעולות **PlacingOption**	
public PlacingOption(int i, int j, int rate)	פעולה בונה מהלך
public PlacingOption()	פעולה בונה ריקה

תכונות PlacingOptions	
`List<PlacingOption> options`	רשימה של מהלכים אפשריים לביצוע על ידי המחשב

פעולות PlacingOptions	
`public PlacingOptions()`	פעולה בונה ריקה
`public PlacingOption MaxOption()`	פעולה שמחזירה את המהלך עם הניקוד הגבוה ביותר ברשימה
`public void Add(PlacingOption placingOption)`	מוסיפה בסוף הרשימה מהלך חדש

תכונות LineSequence	
`bool blockedRight`	האם הרצף חסום מימין
`bool blockedLeft`	האם הרצף חסום משמאל
`int oreh`	האורך של הרצף

פעולות LineSequence	
`public LineSequence(int oreh, bool blockedRight, bool blockedLeft)`	פעולה בונה
`public bool TotalBlocked()`	האם הרצף חסום שני צידיו
`public bool TotalFree()`	האם הרצף פתוח משני צידיו
`public bool HalfFree()`	האם הרצף חסום רק מצד אחד

תכונות ToorSequence	
`bool blockedUp`	האם הרצף חסום מימין
`bool blockedDown`	האם הרצף חסום משמאל
`int oreh`	האורך של הרצף

פעולות ToorSequence	
`public ToorSequence(int oreh, bool blockedUp, bool blockedDown)`	פעולה בונה
`public bool TotalBlocked()`	האם הרצף חסום שני צידיו
`public bool TotalFree()`	האם הרצף פתוח משני צידיו
`public bool HalfFree()`	האם הרצף חסום רק מצד אחד

תכונות SlantRightSequence	
bool blockedRightUp	האם הרצף חסום מימין למעלה
bool blockedLeftDown	האם הרצף חסום משמאל למטה
int oreh	האורך של הרצף

פעולות SlantRightSequence	
public SlantRightSequence(int oreh, bool blockedRightUp, bool blockedLeftDown)	פעולה בונה
public bool TotalBlocked()	האם הרצף חסום שני צידיו
public bool TotalFree()	האם הרצף פתוח משני צידיו
public bool HalfFree()	האם הרצף חסום רק מצד אחד

תכונות SlantLeftSequence	
bool blockedLeftUp	האם הרצף חסום משמאל למעלה
bool blockedRightDown	האם הרצף חסום מימין למטה
int oreh	האורך של הרצף

פעולות SlantLeftSequence	
public SlantLeftSequence(int oreh, bool blockedLeftUp, bool blockedRightDown)	פעולה בונה
public bool TotalBlocked()	האם הרצף חסום שני צידיו
public bool TotalFree()	האם הרצף פתוח משני צידיו
public bool HalfFree()	האם הרצף חסום רק מצד אחד

תכונות RotationArrow	
PictureBox rotationArrow	התמונה של החץ סיבוב
Direction direction	כיוון החץ - ימינה או שמאלה
Quarter quarter	הרבעון אליו משויך החץ
Graphics g	משתנה הגרפיקה
FormGame formGame	טופס המשחק

Board board	הלוח
Pen penBoardFrame	עט ציור מסגרת הלוח
Pen penCells	עט ציור התאים בלוח
FormVictory formVictory	טופס הניצחון
FormDefeat formDefeat	טופס ההפסד
FormTie formTie	טופס התיקו
ArtificialIntelligence AI	הבינה המלאכותית
int time	הזמן מתחילת המשחק
string playerName	שם השחקן

פעולות **RotationArrow**	
public RotationArrow(int x, int y, int width, int height, FormGame formGame, Image image, RotationArrow.Direction direction, Quarter quarter, Graphics g, Board board, Pen penBoardFrame, Pen penCells, FormVictory formVictory, FormDefeat formDefeat, FormTie formTie)	פעולה בונה
private void FormGame_MouseClick(object sender, MouseEventArgs e)	ניהול הקלקה על לחצן סיבוב
public void Activate()	פעולה המפעילה את לחצן חץ הסיבוב ללא הקלקה עליו
public bool Inside(int x, int y)	בודקת אם נקודה נמצאת בתוך תמונת חץ הסיבוב
public static void SetTime(int time0)	מגדירה את זמן המשחק כסטטי
public static void SetPlayerName(string playerName0)	מגדריה את שם השחקן כסטטי

תכונות **ArtificialIntelligence**	
Board board	הלוח הנוכחי במשחק
List<Board> boards	רשימה של 8 לוחות שעשויים להיווצר בעקבות לחיצה על כל אחד מחיצי הסיבוב

פעולות ArtificialIntelligence	
public ArtificialIntelligence(Board board, Graphics g, FormVictory formVictory, FormDefeat formDefeat)	פעולה בונה
public void RotateBoards3()	פעולה מסובבת את כל חלקי הלוח בהתאם לחץ הסיבוב המתאים לו
public bool ThereIsFiveSequence(int oreh, LineSequence maxLineSequence, ToorSequence maxToorSequence, SlantRightSequence maxSlantRightSequence, SlantLeftSequence maxSlantLeftSequence)	האם קיים רצף באורך נתון
public bool AllKindsTotalBlocked(LineSequence maxLineSequence, ToorSequence maxToorSequence, SlantRightSequence maxSlantRightSequence, SlantLeftSequence maxSlantLeftSequence)	אם כל סוגי הרצפים (שורה,טור , אלכסונים)המקסימלים חסומים משני צידיהם
public bool AtLeastOneTotalFreeSequence(int oreh, LineSequence maxLineSequence, ToorSequence maxToorSequence, SlantRightSequence maxSlantRightSequence, SlantLeftSequence maxSlantLeftSequence)	האם יש רצף באורך מסויים ופתוח משני הכיוונים לפחות בסוג רצף אחד
public bool AtLeastOneSequenceHalfFree(int oreh, LineSequence maxLineSequence, ToorSequence maxToorSequence, SlantRightSequence maxSlantRightSequence, SlantLeftSequence maxSlantLeftSequence)	האם יש רצף באורך מסויים ופתוח למחצה לפחות בסוג רצף אחד
public bool SpaceSlantVictory(Board board, Cell.Status status)	אם יש מצב של אלכסון של שלושה ברבעון אחד , הרביעי נמצא במרכז הרבעון המנוגד לו ווקצוותיו פנויים כך שסיבוב רבעון זה יצור חמישיה ברצף. זה מצב ניצחון בטוח
public int Rate(Board board, Cell.Status status)	הפעולה מעריכה את מצב הלוח ונותנת לו ניקוד בהתאם לכמה הוא קרוב לניצחון לצבע מסוים
public void RatePlaceRotate(Graphics g, Board realBoard, FormVictory formVictory, FormDefeat formDefeat)	הפעולה מבצעת תור עבור המחשב:מציבה אבן ומסובבת את אחד מחלקי הלוח
public void PlacingBest(Graphics g, Board realBoard, FormVictory formVictory, FormDefeat formDefeat)	ממקמת גולה במקום המיטבי בלוח
public void RandomPlacingRotating(Graphics g)	פעולה לבדיקה עצמית בלבד ללא שימוש מעשי . הפעולה ממקמת גולה במיקום אקראי בלוח

UML של טפסים

תכונות FormDefeat	
MainMenu mainMenu	התפריט הראשי – הטופס הראשי

פעולות FormDefeat	
public FormDefeat(MainMenu mainMenu)	פעולה בונה
private void pictureBox1_Click(object sender, EventArgs e)	לחיצה על כפתור הבית המעביר את המשתמש חזרה לתפריט הראשי
private void FormDefeat_FormClosing(object sender, FormClosingEventArgs e)	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס

תכונות FormGame	
FormVictory formVictory	טופס ניצחון
FormDefeat formDefeat	טופס הפסד
FormTie formTie	טופס תיקו
Graphics g	משתנה גראפיקה
Board board	לוח המשחק
Brush brush	בראש של הצביעה
Pen penBoardFrame	עט ציור מסגרת הלוח
Pen penCells	עט ציור תאי הלוח
int xBoard	איקס עוגן הלוח
int yBoard	וואי עוגן הלוח
int widthBoard	רוחב הלוח
int heightBoard	גובה הלוח
MainMenu mainMenu	טופס תפריט ראשי
static string playerName	שם השחקן
int time	זמן שעבר מתחילת המשחק

פעולות FormGame	
public FormGame(MainMenu mainMenu)	פעולה בונה
private void HomeButton_Click(object sender, EventArgs e)	לחיצה על כפתור הבית המעביר את המשתמש חזרה לתפריט הראשי
private void FormGame_FormClosing(object sender, FormClosingEventArgs e)	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס
public void FormGame_MouseClick(object sender, MouseEventArgs e)	מנהלת את ההקלקה בטופס
private void FormGame_Paint(object sender, PaintEventArgs e)	אחרי כל תור הלוח מתעדכן בהתאמה
private void timeCounter_Tick(object sender, EventArgs e)	הטיימר מציג לשחקן כמה זמן בשניות עבר מתחילת המשחק
public static void SetPlayerName(string playerName0)	שימוש במשתנה סטטי שם משתמש

תכונות FormInstructions	
MainMenu mainMenu	התפריט הראשי – הטופס הראשי

פעולות FormInstructions	
public FormInstructions(MainMenu mainMenu)	פעולה בונה
private void homeButton_Click(object sender, EventArgs e)	לחיצה על כפתור הבית המעביר את המשתמש חזרה לתפריט הראשי
private void FormInstructions_FormClosing(object sender, FormClosingEventArgs e)	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס

תכונות FormTie	
MainMenu mainMenu	התפריט הראשי – הטופס הראשי

פעולות FormTie	
public FormTie(MainMenu mainMenu)	פעולה בונה
private void homeButton_Click(object sender, EventArgs e)	לחיצה על כפתור הבית המעביר את המשתמש חזרה לתפריט הראשי
private void FormTie_FormClosing(object sender, FormClosingEventArgs e)	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס

תכונות FormVictory	
MainMenu mainMenu	התפריט הראשי – הטופס הראשי

פעולות FormVictory	
public FormVictory(MainMenu mainMenu)	פעולה בונה
private void homeButton_Click(object sender, EventArgs e)	לחיצה על כפתור הבית המעביר את המשתמש חזרה לתפריט הראשי
private void FormVictory_FormClosing(object sender, FormClosingEventArgs e)	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס

תכונות MainMenu	
RecordsForm formRecords	טופס טבלת שיאים
FormInstructions formInstructions	טופס הוראות
FormGame formGame	טופס המשחק

פעולות MainMenu	
public MainMenu()	פעולה בונה
private void instructionsButton_Click(object sender, EventArgs e)	מעבר לפורם הוראות
private void MainMenu_FormClosing(object sender, FormClosingEventArgs e)	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס
private void exitButton_Click(object sender, EventArgs e)	יציאה באמצעות לחצן פנימי
private void startButton_MouseClick(object sender, MouseEventArgs e)	כניסה למשחק לאחר הקלדת שם השחקן
private void HistoryButton_Click(object sender, EventArgs e)	מעבר לטופס שיאים
private void nameBox_Click(object sender, EventArgs e)	הקלקה על תיבת הטקסט

תכונות RecordsForm	
`MainMenu mainMenu`	טופס תפריט ראשי
`string playerName`	שם שחקן
`List<Score> scores`	רשימה של כל התוצאות
`Label[] times`	רשימה של כל הזמנים
`Label[] names`	רשימה של כל שמות השחקנים

פעולות RecordsForm	
`public RecordsForm(MainMenu mainMenu, string playerName)`	פעולה בונה
`private void homeButton_Click(object sender, EventArgs e)`	לחיצה על כפתור הבית המעביר את המשתמש חזרה לתפריט הראשי
`private void FormInstructions_FormClosing(object sender, FormClosingEventArgs e)`	סוגר את המשחק לאחר לחיצה על לחצן האיקס בקצה הטופס
`public void ResetScoresList(string fileName)`	הפעולה קוראת מקובץ טקסט את התוצאות מהמשחקים הקודמים
`public void SortScores()`	מארגנת את רשימת התוצאות לפי זמן הניצחון הקצר אל הארוך
`public void SetLabels()`	מעדכן את הטקסט בלייבלים לפי המיון

היוריסטיקה

אני בחרתי לכתוב את המשחק פנטגו כמשחק של "אדם נגד המחשב". המחשב מסוגל להיות יריב ראוי לשחקן אחר בעקבות הכנסת בינה מלאכותית לתוכנה. בקוד המשחק ישנה מחלקה שלמה אשר שמורה לייעוד זה – ArtificialIntellegence. בשורש של מחלקה זו נמצאת הפעולה ההיוריסטית. פעולה היוריסטית היא פעולה אשר מקבלת כפרמטר מצב במשחק – במקרה של המשחק שלי, לוח – ונותנת לו ניקוד, ערך עבור כמה מצב זה קרוב לניצחון. למשל, במשחק שלי אם פעולה זו תקבל לוח שבו רצף של חמש אבנים זהות באותה שורה זו ליד זו (מצב ניצחון) הפעולה תחזיר את ערכה המקסימלי. לכל מצב ניתן ניקוד אחר בהתאם לכמה הוא מקרב את המחשב לניצחון. לדוגמה, מצב בו יש ארבע אבנים זהות ברצף זו לצד זו יהיה בעל ערך גבוה יותר מאשר מצב בו יש רק שלוש אבנים ברצף שכזה. כמו כן, גם לקצה ברצף יש משמעות: רצף אבנים זהות שבקצותיו יש אבנים של היריב ינוקד פחות מאשר אותו רצף אשר בקצותיו אין אבנים נוספות ואינו נמצא בקצה הלוח. זאת מאחר שרצף אשר קצותיו חסומים לא ניתן יהיה ניתן להרחיב לרצף המספיק לניצחון, ואילו רצף אשר פתוח מקצותיו ניתן יהיה. בנוסף, רצף אשר יהיה חסום רק מצידו האחד ינוקד פחות מרצף זהה אשר פתוח משני קצותיו, אך יותר מרצף זהה אשר חסום משני קצותיו.
אופן תהליך החלטת המחשב על ביצוע צעד:
תחילה על המחשב להחליט איפה יניח אבן בלוח. לשם כך המחשב עובר על כל תאי הלוח הפנויים, ומשתמש בפונקציה ההיוריסטית כדי להעריך כמה "שווה" לו להציב במיקום זה אבן. עבור כל תא המחשב יוצר עצם מסוג "אופציה" (option) שמכיל את מיקום התא ואת ערך הלוח למחשב (כמה המחשב קרוב לניצחון) פחות ערך הלוח למשתמש (כמה היריב קרוב לניצחון) עבור המצב שבו האבן תוצב במיקום זה. המחשב מכניס את כל האופציות שיווצרו מכל התאים לרשימה. מתוך הרשימה המחשב מחשב את המהלך שבעל הניקוד הגבוה ביותר – זה שהכי מקרב אותו לניצחון וכמו כן מרחיק את היריב מניצחונו שלו ומבצע אותו.
לאחר ש"החליט" היכן יניח אבן, על המחשב לקבוע איזה מן רבעונים "ירצה" לסובב ולאיזה כיוון. דהיינו, על המחשב להחליט איזה מן כפתורי הסיבוב ירצה להפעיל. במחלקת הבינה קיימת רשימה ובה שמונה לוחות שמהווים את (המצבים) שיווצרו כתוצאה מהפעלה של כל אחד מכפתורי סיבוב הרבעונים. המחשב נותן לכל אחד מהלוחות הללו ניקוד כך: כמה המחשב קרוב לניצחון פחות כמה המשתמש קרוב לניצחון. המחשב יבחר להשתמש בכפתור הסיבוב אשר יצור את הלוח עם הניקוד הגבוהה ביותר עפ"י החישוב לעיל. כך המחשב יבצע את המהלך שהכי יקרב אותו לניצחון והכי ירחיק את היריב מניצחונו שלו.

לאחר שהמחשב מבצע את תורו, תוך הנחת אבן בלוח וסיבוב אחד מהרבעונים, מגיע תור היריב (בהנחה שהמשחק לא הסתיים בשל נצחון אחד מהשניים) ולאחריו שוב תור המחשב וכך חוזר חלילה עד נצחון אחד משניהם או הגעה למצב תיקו.

מבנים מיוחדים

כחלק מתהליך כתיבת התוכנה השתמשתי בכמה וכמה צורות כתיבה מורכבות ומבנים מיוחדים. למשל, במחלקה Board השתמשתי ב**רקורסיה** בפעולות:

```
1.public int RecursiaLengthLine(int i, int j, Cell.Status status)
2.public int RecursiaLengthToor(int i, int j, Cell.Status status)
3.public int RecursiaLengthLeftSlant(int i, int j, Cell.Status status)
4.public int RecursiaLengthRightSlant(int i, int j, Cell.Status status)
```

בפעולה הראשונה המטרה הייתה להחזיר את אורך הרצף שיוצא מתא מסוים בצבע מסוים בלוח באותה השורה.
בפעולה השניה המטרה הייתה להחזיר את אורך הרצף שיוצא מתא מסוים בצבע מסוים בלוח באותו טור.
בפעולה השלישית המטרה הייתה להחזיר את אורך הרצף שיוצא מתא מסוים בצבע מסוים בלוח באותו אלכסון נוטה שמאלה.
ובפעולה הרביעית המטרה הייתה להחזיר את אורך הרצף שיוצא מתא מסוים בצבע מסוים בלוח באותו אלכסון הנוטה ימינה.

בנוסף לרקורסיה השתמשתי במבני נתונים מורכבים כמו **רשימה מקושרת**, **עצמים מורכבים ו**מערכים**. לדוגמה: במחלקת הבינה (ArtificialIntellegence) השתמשתי ברשימה מקושרת של לוחות אשר מייצגים כל אחד את המצבים שיווצרו בעקבות הפעלת כל אחד מכפתורי סיבוב הרבעונים בהתאמה. יתרה מזאת, במחלקה לוח השתמשתי במערך דו מימדי שמכיל את כל תאי הלוח. עצם מורכב הוא למעשה הלוח, כאשר כל לוח מורכב מארבעה רבעונים שכל אחד מהם מורכב מתשעה תאים. עצם מורכב נוסף שבניתי הוא ה"אופציה" (placingOption) אשר מייצגת אפשרות להצבת אבן בלוח על ידי המחשב. עצם זה הוא חלק מעצם נוסף (placingOption) אשר מהווה רשימה של אופציות.

רפלקציה

לפני תחילת הפרויקט, התבקשתי לבחור משחק שאותו ארצה לבנות. גם לאחר חשיבה מרובה לא הצלחתי להגיע להחלטה, לכן פניתי אל המורה שלנו בבקשה לעזרה בבחירה. זאב, המורה, הציע לי את המשחק "פנטגו". הפשטות שבחוקים מצד אחד, והמורכבות החשיבתית הרבה שדרושה כדי לנצח שבו את ליבי וכך בחרתי את המשחק. אני מאוד נהנתי לבנות את התוכנה: אהבתי להתמודד עם האתגרים השזורים לאורך כתיבת הקוד, את המחשבה הרבה הכרוכה בכך ואת המאמצים הרבים הכרוכים בלנסות לתרגם משחק לוח לשפת תכנות. לעיתים נתקלתי בקשיים בעיצוב, אך באמצעות סיוע מחברים הצלחתי להתגבר עליהם.

קוד מתועד

Board

```
class Board
    {
        private int x;      //הלוח בעוגן איקס ערך
        private int y;      // הלוח בעוגן הוואי ערך
        private int width;  //  הלוח רוחב
        private int height;  /// הלוח גובה
        private Quarter[,] quarters;  // הרבעונים - הלוח חלקי ארבעת של מימדי דו מערך
        public enum Turn { Black, White, gameOver }; // התור מי של לי שמראה פרמטר -
נגמר שהמשחק או המחשב של או שלי
        private Turn turn;
        public enum TurnClick { cell, rotationArrow, gameOver }; //שמראה פרמטר
סיבוב חץ על או בלוח תא על לחיצה להגיע צריכה עכשיו אם
        private TurnClick turnClick;
        private Cell[,] allCells;    // בלוח התאים כל של מימדי דוד מערך
        private RotationArrow rightUpTurnRight;  //ימינה שמסובב למעלה ימין סיבוב חץ
        private RotationArrow rightUpTurnLeft;   //שמאלה שמסובב למעלה ימין סיבוב חץ
        private RotationArrow rightDownTurnRight;//ימינה שמסובב למטה ימין סיבוב חץ
        private RotationArrow rightDownTurnLeft;//שמאלה שמסובב למטה ימין סיבוב חץ
        private RotationArrow leftDownTurnRight;//ימינה שמסובב למטה שמאל סיבוב חץ
        private RotationArrow leftDownTurnLeft;//שמאלה שמסובב למטה שמאלה סיבוב חץ
        private RotationArrow leftUpTurnRight;//ימינה שמסובב למעלה שמאל סיבוב חץ
        private RotationArrow leftUpTurnLeft;//שמאלה שמסובב למעלה שמאל סיבוב חץ
        private Graphics g; //הגרפיקה משתנה
        private Pen penBoardFrame;   //הלוח מסגרת את שמצייר עט
        private Pen penCells;  // בלוח התאים מסגרת את שמצייר עט
        private Pen penWin;  // הניצחון קו את שמצייר עט
        private FormVictory formVictory; //ניצחון טופס
        private FormDefeat formDefeat;  // הפסד טופס
        private FormTie formTie;  // תיקו טופס

        //בונה פעולה
        public Board(int x, int y, int width, int height, FormGame formGame,
Graphics g, Pen penBoardFrame, Pen penCells, FormVictory formVictory,
FormDefeat formDefeat, FormTie formTie) // constructor function
       {
            this.formVictory = formVictory;
            this.formDefeat = formDefeat;
            this.formTie = formTie;
            this.g = g;
            this.x = x;
            this.y = y;
            this.width = width;
            this.height = height;
            this.penBoardFrame = penBoardFrame;
            this.penCells = penCells;
            this.penWin = new Pen(Color.Green, 10);

            int difX = (int)(this.width / 2); //  width of a quarter
            int difY = (int)(this.height / 2);  // height of a quarter

            this.quarters = new Quarter[2, 2];   //הלוח חלקי 4 את מגדירים
            this.quarters[0, 0] = new Quarter(this.x, this.y, difX,
difY,formGame);
            this.quarters[0, 1] = new Quarter(this.x + difX, this.y, difX,
difY, formGame);
```

```
            this.quarters[1, 0] = new Quarter(this.x, this.y + difY, difX,
difY, formGame);
            this.quarters[1, 1] = new Quarter(this.x + difX, this.y + difY,
difX, difY, formGame);
            this.turn = Turn.White;        ///המחשב לא) השחקן של הראשון התור הגדרת
            this.turnClick = TurnClick.cell;
            allCells = new Cell[6, 6];
            for (int i = 0; i < 3; i++)                ///הכולל המערך איברי את מעדכן
תור כל אחרי
            {
                for (int j = 0; j < 3; j++)
                {
                    allCells[i, j] = this.quarters[0, 0].GetNineCells()[i, j];
                }
            }
            for (int i = 0; i < 3; i++)
            {
                for (int j = 3; j < 6; j++)
                {
                    allCells[i, j] = this.quarters[0, 1].GetNineCells()[i, j-
3];
                }
            }
            for (int i = 3; i < 6; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    allCells[i, j] = this.quarters[1, 0].GetNineCells()[i-3,
j];
                }
            }
            for (int i = 3; i < 6; i++)
            {
                for (int j = 3; j < 6; j++)
                {
                    allCells[i, j] = this.quarters[1, 1].GetNineCells()[i-3, j-
3];
                }
            }
////הסיבוב חצי הגדרת
            rightUpTurnRight = new RotationArrow((int)(this.allCells[1,
5].GetX() * 1.19), this.allCells[1, 5].GetY(), 85, 85, formGame,
Properties.Resources.rightUpTurnRight2, RotationArrow.Direction.Right,
this.quarters[0, 1], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            rightUpTurnLeft = new RotationArrow((int)(this.allCells[0,
4].GetX() * 0.97), (int)(this.allCells[0, 4].GetY() * 0.2), 105, 105, formGame,
Properties.Resources.rightUpTurnLeft2, RotationArrow.Direction.Left,
this.quarters[0, 1], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            rightDownTurnRight = new RotationArrow((int)(this.allCells[4,
5].GetX() * 1.19), this.allCells[4, 5].GetY(), 90, 90, formGame,
Properties.Resources.rightDownTurnRight2, RotationArrow.Direction.Right,
this.quarters[1, 1], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            rightDownTurnLeft = new RotationArrow((int)(this.allCells[5,
4].GetX() * 0.985), (int)(this.allCells[5, 4].GetY() * 1.16), 95, 95, formGame,
Properties.Resources.rightDownTurnLeft2, RotationArrow.Direction.Left,
this.quarters[1, 1], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            leftDownTurnRight = new RotationArrow((int)(this.allCells[5,
1].GetX() * 0.98), (int)(this.allCells[5, 1].GetY() * 1.165), 90, 90, formGame,
```

```
Properties.Resources.LeftDownTurnRight2, RotationArrow.Direction.Right,
this.quarters[1, 0], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            leftDownTurnLeft = new RotationArrow((int)(this.allCells[4,
0].GetX() * 0.3), (int)(this.allCells[4, 0].GetY() * 0.97), 95, 95, formGame,
Properties.Resources.LeftDownTurnLeft2, RotationArrow.Direction.Left,
this.quarters[1, 0], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            leftUpTurnRight = new RotationArrow((int)(this.allCells[1,
0].GetX() * 0.3), (int)(this.allCells[1, 0].GetY() * 0.98), 87, 87, formGame,
Properties.Resources.LeftUpTurnRight2, RotationArrow.Direction.Right,
this.quarters[0, 0], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);
            leftUpTurnLeft = new RotationArrow((int)(this.allCells[0, 1].GetX()
* 0.98), (int)(this.allCells[0, 1].GetY() * 0.21), 97, 97, formGame,
Properties.Resources.LeftUpTurnLeft2, RotationArrow.Direction.Left,
this.quarters[0, 0], g, this, this.penBoardFrame, this.penCells, formVictory,
formDefeat, formTie);


        }

        public void Draw(Graphics g)            ////הלוח את מציירת
        {

            int difX1=(int)(this.width / 2);  // הלוח מרוחב חצי הוא רבעון רוחב
            int difY1 = (int)(this.height / 2); //הלוח מגובה חצי הוא רבעון גובה
            for (int i = 0; i < quarters.GetLength(0); i++)
            {
                for (int j = 0; j < quarters.GetLength(1); j++)
                {
                    this.quarters[0, 0].SetAll(this.x, this.y, difX1, difY1);
////הרבעונים גודל את מעדכן
                    this.quarters[0, 1].SetAll(this.x+difX1, this.y, difX1,
difY1);
                    this.quarters[1, 0].SetAll(this.x, this.y + difY1, difX1,
difY1);
                    this.quarters[1, 1].SetAll(this.x + difX1, this.y + difY1,
difX1, difY1);

                    this.quarters[i, j].Draw(g, this.penBoardFrame,
this.penCells);  /// שבהם לתאים - יותר קטנות ליחידות הציור פקודת את שולח
                }

            }
        }
public void MouseClick(Graphics g, int x0, int y0)   /// הלוח על לחיצה אירוע מנהל
        {
            bool locationInCircle = false;  // העגול התא בתוך נעשתה הלחיצה האם
            bool locationInBoard = true; // הלוח בתוך נעשתה הלחיצה האם
            bool locationFree = true;
            if (this.turn == Turn.White)  // המשתמש תור עכשיו אם
            {
                if (this.turnClick == TurnClick.rotationArrow) // השחקן על כעת אם
הסיבוב כפתור על ללחוץ
                {
                    MessageBox.Show("you need to click on a rotation arrow
first"); // סיבוב כפתור על קודם ללחוץ המשתמש שעל שתגיד הודעה תצא אז
                    return;
                }
```

```
                if (!this.Inside(x0, y0))  ///if click wasnt in board and wasnt
in arrow
                {
                    locationInBoard = false;
                    MessageBox.Show("click was neither in the board nor in
rotation arrow"); // בלוח ולא סיבוב חץ על לא שהלחיצה למשתמש שתגיד הודעה
                }

                bool tempInCircle=false;
                for (int i = 0; i < 2; i++)
                {
                    for (int j = 0; j < 2; j++)
                    {
                        if (this.quarters[i, j].Inside(x0, y0)) // נמצאת הלחיצה אם
מהרבעונים באחד
                        {
                            tempInCircle = this.quarters[i,
j].IsInCellCircle(x0, y0); // התא של העיגול בתוך נעשתה הלחיצה אם בודק
                            if (tempInCircle) //כן אם
                                locationInCircle = true;
                        }
                    }
                }

                if (!locationInCircle) //העיגול של בתא נעשתה לא הלחיצה אם
                    MessageBox.Show("click inside the circle"); // שגיאה הודעת הצג

                else // התא של העיגול ובתוך המתאים בתור נעשתה הלחיצה אם
                for (int i = 0; i < 2; i++)
                {
                    for (int j = 0; j < 2; j++)
                    {
                        if (this.quarters[i, j].Inside(x0, y0)) //רבעון באיזה בודקת
הלחיצה נעשה
                            locationFree = this.quarters[i, j].MouseClick(g, x0,
y0, this.turn); // התא אם .זה בתא אבן וממקמת אמת מחזירה פעולה, ריק תא על נעשתה הלחיצה אם
שקר מחזירה אז מלא.
                    }
                }


                if (locationFree && locationInBoard && locationInCircle) //אם
בלוח אבן ומוקמה תור נעשה
                {
                    this.turnClick = TurnClick.rotationArrow;   //המשתמש על כעת
סיבוב כפתור כל ללחוץ יהיה
                }

                locationInCircle = false;
                tempInCircle = false;
            }

        }
```

```
public bool IsTotalVictory(Cell.Status statusCompetitor) //האם יש ניצחון או בשורה או
באלכסון או בטור
        {
            if (IsVictoryAllSlants(statusCompetitor) ||
IsVictoryInAllLines(statusCompetitor) ||
IsVictoryInAllColumns(statusCompetitor))
                return true;
            else
                return false;
        }


public void WinLoseTieForms(int time, string playerName) //הסתיים אם בודקת הפעולה
תיקו או הפסד /ניצחון טופס פותחת ובהתאם המשחק
        {
            if (IsTotalVictory(Cell.Status.White))   //ניצחון בדוק
            {
                        //טקסט בקובץ השחקן ושם לנצח שלקח הזמן את רשום אז מנצח השחקן אם

                File.AppendAllText("timeTable.txt",
string.Format(time.ToString() + Environment.NewLine));
                File.AppendAllText("timeTable.txt", string.Format(playerName +
Environment.NewLine));

                turn = Turn.gameOver;
                turnClick = TurnClick.gameOver;
                System.Threading.Thread.Sleep(700);//המחשב של התגובה את מאט זה
                formVictory.Show();
            }
            else if (IsTotalVictory(Cell.Status.Black))
            {
                turn = Turn.gameOver;
                turnClick = TurnClick.gameOver;
                System.Threading.Thread.Sleep(700);//המחשב של התגובה את מאט זה
                formDefeat.Show();
            }

            if (IsTie())    // תיקו אם בודק
            {
                turn = Turn.gameOver;
                turnClick = TurnClick.gameOver;
                System.Threading.Thread.Sleep(700);//המחשב של התגובה את מאט זה
                formTie.Show();
            }
        }
```

רקורסיות

```
public int RecursiaLengthLine(int i, int j, Cell.Status status)
///////////////רקורסיה! שמחשבת אורך של שורה בצבע מסים מתא שיוצא מסים///////////
        {
            if (status == Cell.Status.Black)
            {
                if (j == this.allCells.GetLength(1) || this.allCells[i,
j].GetStatus() != Cell.Status.Black)
                    return 0;
                return 1 + RecursiaLengthLine(i, j + 1, status);
            }

            else if (status == Cell.Status.White)
            {
```

```
                if (j == this.allCells.GetLength(1) || this.allCells[i,
j].GetStatus() != Cell.Status.White)
                    return 0;
                return 1 + RecursiaLengthLine(i, j + 1, status);
            }
            else return -1;
        }

public int RecursiaLengthToor(int i, int j, Cell.Status status)
///////////////רקורסיה! מסוים בצבע מסוים מתא שיוצא טור של אורך שמחשבת/////////////
        {
            if (status == Cell.Status.Black)
            {
                if (i == this.allCells.GetLength(0) || this.allCells[i,
j].GetStatus() != Cell.Status.Black)
                    return 0;
                return 1 + RecursiaLengthToor(i + 1, j, status);
            }
            else if (status == Cell.Status.White)
            {
                if (i == this.allCells.GetLength(0) || this.allCells[i,
j].GetStatus() != Cell.Status.White)
                    return 0;
                return 1 + RecursiaLengthToor(i + 1, j, status);
            }
            else return -1;
        }

        public int RecursiaLengthLeftSlant(int i, int j, Cell.Status status)
/////////רקורסיה! מסוים מתא שיוצא מסוים צבע שמאלי אלכסון של אורך שמחשבת/////////////
        {
            if (status == Cell.Status.Black)
            {
                if (i == this.allCells.GetLength(0) || j ==
this.allCells.GetLength(1) || this.allCells[i, j].GetStatus() !=
Cell.Status.Black)
                    return 0;
                return 1 + RecursiaLengthLeftSlant(i + 1, j + 1, status);
            }

            else if (status == Cell.Status.White)
            {
                if (i == this.allCells.GetLength(0) || j ==
this.allCells.GetLength(1) || this.allCells[i, j].GetStatus() !=
Cell.Status.White)
                    return 0;
                return 1 + RecursiaLengthLeftSlant(i + 1, j + 1, status);
            }
            else return -1;
        }
```

```
public int RecursiaLengthRightSlant(int i, int j, Cell.Status status)
//////////מסוים מתא שיוצא מסוים צבע ימני אלכסון של אורך שמחשבת !רקורסיה////////////
        {
            if (status == Cell.Status.Black)
            {
                if (i == this.allCells.GetLength(0) || j == -1 ||
this.allCells[i, j].GetStatus() != Cell.Status.Black)
                    return 0;
                return 1 + RecursiaLengthRightSlant(i + 1, j - 1, status);
            }
            else if (status == Cell.Status.White)
            {
                if (i == this.allCells.GetLength(0) || j == -1 ||
this.allCells[i, j].GetStatus() != Cell.Status.White)
                    return 0;
                return 1 + RecursiaLengthRightSlant(i + 1, j - 1, status);
            }
            else return -1;
        }

public LineSequence MaxFreeSequanceLine(Cell.Status status)  //הכי הרצף את מחזיר
מסוים בצבע שבשורה ארוך
        {
            LineSequence Ls = new LineSequence(0, false, false);
            int max = 0;
            int oreh = 0;
            bool blockedRight = false;
            bool blockedLeft = false;

            for (int i = 0; i < this.allCells.GetLength(0); i++)
            {

                for (int j = 0; j < this.allCells.GetLength(1); j++) //תא עובר
בלוח תא
                {

                    oreh = RecursiaLengthLine(i, j, status); //הרצף אורך את בודק
בשורה ספציפי מתא שיוצא

                    if (status == Cell.Status.Black)
                    {                                       ///חסום הרצף אם בדיקה
מצידיו
                        if (j + oreh == 6 || this.allCells[i, j +
oreh].GetStatus() == Cell.Status.White)
                            blockedRight = true;

                        if (j == 0 || this.allCells[i, j - 1].GetStatus() ==
Cell.Status.White)
                            blockedLeft = true;
                    }

                    if (status == Cell.Status.White)
                    {                                           ///הרצף אם בדיקה
מצידיו חסום
                        if (j + oreh == 6 || this.allCells[i, j +
oreh].GetStatus() == Cell.Status.Black)
                            blockedRight = true;

                        if (j == 0 || this.allCells[i, j - 1].GetStatus() ==
Cell.Status.Black)
                            blockedLeft = true;
                    }
```

```
                    if (oreh > max) //ארוך הכי הרצף למשתנה הכנס כה עד הגדול הוא האורך אם
                    {
                        max = oreh;
                        Ls.SetOreh(oreh);
                        Ls.SetBlockedRight(blockedRight);
                        Ls.SetBlockedLeft(blockedLeft);
                    }

                    oreh = 0;

                    blockedRight = false;
                    blockedLeft = false;


                }
            }
            return Ls;
        }

public bool IsTie() // תיקו הוא הלוח מצב אם שבודקת פעולה
       {
           for (int i = 0; i < 6; i++)
           {
               for (int j = 0; j < 6; j++)
               {
                   if (allCells[i, j].GetStatus() == Cell.Status.Empty) //אם
שקר החזר ריק תא קיים
                       return false;
               }
           }
           return true; //אמת החזר מלאים התאים כל אם
       }

        public void DrawStatus(Graphics g) // הלוח מחיקת לאחר בלוח מצב את מציירת
        {
           for (int i = 0; i < 2; i++)
           {
               for (int j = 0; j < 2; j++)
               {
                   this.quarters[i, j].DrawStatus(g);
               }
           }
        }

        public bool IsFiveEqualStatus(Cell c1, Cell c2, Cell c3, Cell c4,Cell
c5) // לבנים כולם או שחורים כולם האם - בתכולתם זהים תאים חמישה האם
        {
            if ((c1.GetStatus() == c2.GetStatus()) && (c1.GetStatus() ==
c3.GetStatus()) &&
                (c1.GetStatus() == c4.GetStatus()) && (c1.GetStatus() ==
c4.GetStatus()) &&
                (c1.GetStatus() == c5.GetStatus()) &&
(c1.GetStatus()!=Cell.Status.Empty))
                return true;
            else
                return false;
        }
```

```
        public void DrawVictoryLineInLine(Cell cStart, Cell cEnd)    ///את מצייר
בשורה ניצחון הקו
        {
            g.DrawLine(this.penWin, cStart.GetX(), cStart.GetY() +
cStart.GetHeight() / 2, cEnd.GetX() + cEnd.GetWidth(), cEnd.GetY() +
cEnd.GetHeight() / 2);
        }

        public void DrawVictoryLineInColumn(Cell cStart, Cell cEnd)   /// בטור
ניצחון הקו את מצייר
        {
            g.DrawLine(this.penWin, cStart.GetX() + cStart.GetWidth() / 2,
cStart.GetY(), cEnd.GetX() + cEnd.GetWidth() / 2, cEnd.GetY()+
cEnd.GetHeight());
        }

        public void DrawVictoryLineInSlantLeft(Cell cStart, Cell cEnd)   ///
ניצחון הקו את מצייר שמאלי באלכסון
        {
            g.DrawLine(this.penWin, cStart.GetX(), cStart.GetY(), cEnd.GetX() +
cEnd.GetWidth(), cEnd.GetY() + cEnd.GetHeight());
        }

        public void DrawVictoryLineInSlantRight(Cell cStart, Cell cEnd)  //
ניצחון הקו את מצייר ימני באלכסון
        {
            g.DrawLine(this.penWin, cStart.GetX()+ cStart.GetWidth(),
cStart.GetY(), cEnd.GetX() , cEnd.GetY() + cEnd.GetHeight());
        }


        public bool IsVictoryInOneLineSpecificColor(Cell.Status
statusCompetitor, int IndexLine)// מסוים בצבע אחת בשורה ניצחון יש האם
        {
            if ((IsFiveEqualStatus(this.allCells[IndexLine, 0],
this.allCells[IndexLine, 1], this.allCells[IndexLine, 2],
this.allCells[IndexLine, 3], this.allCells[IndexLine, 4]) &&
                (this.allCells[IndexLine, 0].GetStatus() == statusCompetitor)))
            {
                DrawVictoryLineInLine(this.allCells[IndexLine, 0],
this.allCells[IndexLine, 4]);
                return true;
            }
            else if (IsFiveEqualStatus(this.allCells[IndexLine, 1],
this.allCells[IndexLine, 2], this.allCells[IndexLine, 3],
this.allCells[IndexLine, 4], this.allCells[IndexLine, 5]) &&
                (this.allCells[IndexLine, 1].GetStatus() == statusCompetitor))
            {
                DrawVictoryLineInLine(this.allCells[IndexLine, 1],
this.allCells[IndexLine, 5]);
                return true;
            }
            else
                return false;


        }
        public bool IsVictoryInAllLines(Cell.Status statusCompetitor) // יש האם
בלוח השורות מן באחת מסוים בצבע ניצחון
        {
            for (int i = 0; i < 6; i++)
```

```
            {
                if (IsVictoryInOneLineSpecificColor(statusCompetitor, i))
                    return true;
            }
            return false;
        }


        public bool IsVictoryInOneColumnSpecificColor(Cell.Status
statusCompetitor, int IndexColumn) // מסוים בצבע אחד בטור ניצחון יש האם
        {
            if (IsFiveEqualStatus(this.allCells[0, IndexColumn],
this.allCells[1, IndexColumn], this.allCells[2, IndexColumn], this.allCells[3,
IndexColumn], this.allCells[4, IndexColumn]) &&
                (this.allCells[0, IndexColumn].GetStatus() ==
statusCompetitor))
            {
                DrawVictoryLineInColumn(this.allCells[0, IndexColumn],
this.allCells[4, IndexColumn]);
                return true;
            }
            else if ((IsFiveEqualStatus(this.allCells[1, IndexColumn],
this.allCells[2, IndexColumn], this.allCells[3, IndexColumn], this.allCells[4,
IndexColumn], this.allCells[5, IndexColumn])) &&
                (this.allCells[1, IndexColumn].GetStatus() ==
statusCompetitor))
            {
                DrawVictoryLineInColumn(this.allCells[1, IndexColumn],
this.allCells[5, IndexColumn]);
                return true;
            }
            else
                return false;


        }


public bool IsVictoryInAllColumns(Cell.Status statusCompetitor) //ניצחון יש האם
מסוים בצבע בלוח הטורים מן באחד
        {
            for (int j = 0; j < 6; j++)
            {
                if (IsVictoryInOneColumnSpecificColor(statusCompetitor,j))
                    return true;
            }
            return false;
        }


        public bool IsVictoryLeftMainUpSlant(Cell.Status statusCompetitor) //
העליון הראשי השמאלי באלכסון ניצחון יש האם
        {
            if (IsFiveEqualStatus(this.allCells[0, 0], this.allCells[1, 1],
this.allCells[2, 2], this.allCells[3, 3], this.allCells[4, 4]))
            {
                if (statusCompetitor == this.allCells[0, 0].GetStatus())
                {
                    DrawVictoryLineInSlantLeft(this.allCells[0, 0],
this.allCells[4, 4]);
                    return true;
                }
```

```
            }
            return false;
        }

        public bool IsVictoryLeftMainDownSlant(Cell.Status statusCompetitor)
//תחתון ראשי השמאלי באלכסון ניצחון יש האם
        {
            if (IsFiveEqualStatus(this.allCells[1, 1], this.allCells[2, 2],
this.allCells[3, 3], this.allCells[4, 4], this.allCells[5, 5]))
            {
                if (statusCompetitor == this.allCells[1, 1].GetStatus())
                {
                    DrawVictoryLineInSlantLeft(this.allCells[1, 1],
this.allCells[5, 5]);
                    return true;
                }
            }
            return false;
        }

        public bool IsVictoryLeftDownSlant(Cell.Status statusCompetitor) //האם
תחתון משני השמאלי באלכסון ניצחון יש
        {
            if (IsFiveEqualStatus(this.allCells[1, 0], this.allCells[2, 1],
this.allCells[3, 2], this.allCells[4, 3], this.allCells[5, 4]))
            {
                if (statusCompetitor == this.allCells[1, 0].GetStatus())
                {
                    DrawVictoryLineInSlantLeft(this.allCells[1, 0],
this.allCells[5, 4]);
                    return true;
                }
            }
            return false;
        }

        public bool IsVictoryLeftUpSlant(Cell.Status statusCompetitor) //יש האם
עליון משני השמאלי באלכסון ניצחון
        {
            if (IsFiveEqualStatus(this.allCells[0, 1], this.allCells[1, 2],
this.allCells[2, 3], this.allCells[3, 4], this.allCells[4, 5]))
            {
                if (statusCompetitor == this.allCells[0, 1].GetStatus())
                {
                    DrawVictoryLineInSlantLeft(this.allCells[0, 1],
this.allCells[4, 5]);
                    return true;
                }
            }
            return false;
        }

public bool IsVictoryLeftSlant(Cell.Status statusCompetitor) //באחד ניצחון יש האם
שמאלה הנוטים השמאליים מהאלכסונים
        {
            if (IsVictoryLeftMainUpSlant(statusCompetitor) ||
IsVictoryLeftMainDownSlant(statusCompetitor) ||
                IsVictoryLeftUpSlant(statusCompetitor) ||
IsVictoryLeftDownSlant(statusCompetitor))
                return true;
            else
                return false;
```

```
        }


        public bool IsVictoryRightMainUpSlant(Cell.Status statusCompetitor)
//עליון ימני הראשי באלכסון ניצחון יש האם
        {
            if (IsFiveEqualStatus(this.allCells[0, 5], this.allCells[1, 4],
this.allCells[2, 3], this.allCells[3, 2], this.allCells[4, 1]))
            {
                if (statusCompetitor == this.allCells[0, 5].GetStatus())
                {
                    DrawVictoryLineInSlantRight(this.allCells[0, 5],
this.allCells[4, 1]);
                    return true;
                }
            }
            return false;
        }

        public bool IsVictoryRightMainDownSlant(Cell.Status statusCompetitor)
//תחתון ימני הראשי באלכסון ניצחון יש האם
        {
            if (IsFiveEqualStatus(this.allCells[1, 4], this.allCells[2, 3],
this.allCells[3, 2], this.allCells[4, 1], this.allCells[5, 0]))
            {
                if (statusCompetitor == this.allCells[1, 4].GetStatus())
                {
                    DrawVictoryLineInSlantRight(this.allCells[1, 4],
this.allCells[5, 0]);
                    return true;
                }
            }
            return false;
        }
```

ArtificialIntellegnce

```
class ArtificialIntelligence
    {
        private Board board; //במשחק הנוכחי הלוח
        private List<Board> boards; //על לחיצה בעקבות להיווצר שעשויים לוחות 8 של רשימה
הסיבוב מחיצי אחד כל


        public ArtificialIntelligence(Board board, Graphics g, FormVictory
formVictory, FormDefeat formDefeat)
        { //בונה פעולה

            this.board = board;

            this.boards = new List<Board>();

            //boards.Add(this.boardLUTurnRight);
            //boards.Add(this.boardLUTurnLeft);
            //boards.Add(this.boardRUTurnRight);
            //boards.Add(this.boardRUTurnLeft);
            //boards.Add(this.boardRDTurnRight);
            //boards.Add(this.boardRDTurnLeft);
            //boards.Add(this.boardLDTurnRight);
            //boards.Add(this.boardLDTurnLeft);

            for (int i = 0; i < 8; i++)
                boards.Add(new Board(board));

        }


public void RotateBoards3()  //  לו המתאים הסיבוב לחץ בהתאם הלוח חלקי כל את מסובבת פעולה
        {
            boards[0].GetQuarters()[0, 0].Rotate90DegreesRight(); // מסובבת הפעולה
את ימינה עליון הימני הלוח
            boards[1].GetQuarters()[0, 0].Rotate90DegreesLeft();  // מסובבת הפעולה
את שמאלה עליון הימני הלוח
            boards[2].GetQuarters()[0, 1].Rotate90DegreesRight(); // מסובבת הפעולה
את ימינה תחתון הימני הלוח
            boards[3].GetQuarters()[0, 1].Rotate90DegreesLeft();   // הפעולה
שמאלה תחתון הימני הלוח את מסובבת
            boards[4].GetQuarters()[1, 1].Rotate90DegreesRight(); // מסובבת הפעולה
את ימינה תחתון השמאלי הלוח
            boards[5].GetQuarters()[1, 1].Rotate90DegreesLeft();  // מסובבת הפעולה
את שמאלה תחתון השמאלי הלוח
            boards[6].GetQuarters()[1, 0].Rotate90DegreesRight(); // מסובבת הפעולה
את ימינה עליון השמאלי הלוח
            boards[7].GetQuarters()[1, 0].Rotate90DegreesLeft();  // מסובבת הפעולה
את שמאלה עליון השמאלי הלוח
            for (int i = 0; i < boards.Count; i++)
                boards[i].CopyQuartersToAllCells();
        }

public int Rate(Board board, Cell.Status status) //לו ונותנת הלוח מצב את מעריכה הפעולה
מסוים לצבע לניצחון קרוב הוא לכמה בהתאם ניקוד
        {
            LineSequence maxLineSequence = board.MaxFreeSequanceLine(status);
            ToorSequence maxToorSequence = board.MaxFreeSequanceToor(status);
            SlantRightSequence maxSlantRightSequence =
board.MaxFreeSequenceSlantRight(status);
```

```
            SlantLeftSequence maxSlantLeftSequence =
board.MaxFreeSequenceSlantLeft(status);


            if (ThereIsFiveSequence(5, maxLineSequence, maxToorSequence,
maxSlantRightSequence, maxSlantLeftSequence) ||
                SpaceSlantVictory(board, status))
                return 1000; ///ברצף חמש יש אם
            else if (AllKindsTotalBlocked(maxLineSequence, maxToorSequence,
maxSlantRightSequence, maxSlantLeftSequence))
                return 1;  ////חסום הכל אם

            else if (AtLeastOneTotalFreeSequence(4, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 80;    ////אחד רצף בסוג לפחות הכיוונים משני ופתוח 4 קבוע אורך
            else if (AtLeastOneSequenceHalfFree(4, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 70;   //אחד רצף בסוג לפחות אחד מכיוון ופתוח 4 קבוע אורך

            else if (AtLeastOneTotalFreeSequence(3, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 60;    ////אחד רצף בסוג לפחות הכיוונים משני ופתוח 3 קבוע אורך
            else if (AtLeastOneSequenceHalfFree(3, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 50; // אחד רצף בסוג לפחות אחד מכיוון ופתוח 3 קבוע אורך

            else if (AtLeastOneTotalFreeSequence(2, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 40;    ////אחד רצף בסוג לפחות הכיוונים משני ופתוח 2 קבוע אורך
            else if (AtLeastOneSequenceHalfFree(2, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 30;  // אחד רצף בסוג לפחות אחד מכיוון ופתוח 2 קבוע אורך

            else if (AtLeastOneTotalFreeSequence(1, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 20;    ////אחד רצף בסוג לפחות הכיוונים משני ופתוח 1 קבוע אורך
            else if (AtLeastOneSequenceHalfFree(1, maxLineSequence,
maxToorSequence, maxSlantRightSequence, maxSlantLeftSequence))
                return 10;   // אחד רצף בסוג לפחות אחד מכיוון ופתוח 1 קבוע אורך
            else
                return 5;

        }

public void RatePlaceRotate(Graphics g, Board realBoard, FormVictory
formVictory, FormDefeat formDefeat)
        {//הלוח מחלקי אחד את ומסובבת אבן מציבה:המחשב עבור תור מבצעת הפעולה
            PlacingBest(g, realBoard, formVictory, formDefeat); //במקום אבן מציבה
המיטבי
            for (int i = 0; i < boards.Count; i++)   //של המצבים לוחות את מעדכן
הגולה בהצבת החצים כפתורי
            {
                boards[i] = new Board(realBoard);
            }
            RotateBoards3();    //החצים מצבי לוחות את מסובב

            int max = -100; //מינימלי התחלתי ערך
            int[] maxValueOfBoards = new int[8];    //המקסימלי הניקוד את שמכיל מערך
סיבוב כפתור לכל לוח - לוח לכל

            for (int i = 0; i < maxValueOfBoards.Length; i++)
            {
```

```
                maxValueOfBoards[i] =Rate(boards[i], Cell.Status.Black) -
Rate(boards[i], Cell.Status.White) ;   //// מיטבי (לוח) מצב מצא למחשב
                if (maxValueOfBoards[i] > max)
                    max = maxValueOfBoards[i];
            }


            for (int i = 0; i < maxValueOfBoards.Length; i++)
            {
                if (max == maxValueOfBoards[i])   ///המיטבי הלוח זה אם
                {                                  //המתאים הרבעון את סובב
                    if (i == 0)
                        realBoard.GetLeftUpTurnRight().Activate();
                    else if (i == 1)
                        realBoard.GetLeftUpTurnLeft().Activate();
                    else if (i == 2)
                        realBoard.GetRightUpTurnRight().Activate();
                    else if (i == 3)
                        realBoard.GetRightUpTurnLeft().Activate();
                    else if (i == 4)
                        realBoard.GetRightDownTurnRight().Activate();
                    else if (i == 5)
                        realBoard.GetRightDownTurnLeft().Activate();
                    else if (i == 6)
                        realBoard.GetLeftDownTurnRight().Activate();
                    else if (i == 7)
                        realBoard.GetLeftDownTurnLeft().Activate();

                    return;
                }
            }

        }

public void PlacingBest(Graphics g, Board realBoard, FormVictory formVictory,
FormDefeat formDefeat) ///בלוח המיטבי במקום גולה ממקמת
        {
            Board testBoard = new Board(realBoard);
            int testRate = 0;
            PlacingOptions pOptions = new PlacingOptions();
            PlacingOption bestOption = new PlacingOption();

            for (int i = 0; i < realBoard.GetAllCells().GetLength(0); i++)
////בה ומציב המקסימלי את מוצא הצבה לכל שווי ונוקד בלוח ההצבה אפשרויות על תא תא עובר
            {
                for (int j = 0; j < realBoard.GetAllCells().GetLength(1); j++)
                {
                    if (realBoard.GetAllCells()[i, j].GetStatus() ==
Cell.Status.Empty)
                    {

testBoard.GetAllCells()[i,j].SetStatus(Cell.Status.Black);
                        testRate = Rate(testBoard, Cell.Status.Black) -
Rate(testBoard, Cell.Status.White);        //המצב את מנקד
                        testBoard.GetAllCells()[i,
j].SetStatus(Cell.Status.Empty);

                        pOptions.Add(new PlacingOption(i, j, testRate));
////האופציות לרשימת זה במיקום ההצבה אופצית את מוסיף
                    }

                }
```

```
            }

            bestOption = pOptions.MaxOption();

            realBoard.GetAllCells()[bestOption.GetI(),
bestOption.GetJ()].SetStatus(Cell.Status.Black);

            realBoard.GetAllCells()[bestOption.GetI(),
bestOption.GetJ()].PlaceBlack(g);   //מתאים שנמצא במקום הגולה את מציב

            System.Threading.Thread.Sleep(700);

            if (board.IsTotalVictory(Cell.Status.White))     // בודק אם יש ניצחון או
הפסד
            {
                System.Threading.Thread.Sleep(700);//המחשב של התגובה את מאט זה
                formVictory.Show();
                return;
            }
            else if (board.IsTotalVictory(Cell.Status.Black))
            {
                System.Threading.Thread.Sleep(700);//המחשב של התגובה את מאט זה
                formDefeat.Show();
                return;
            }


        }


public bool SpaceSlantVictory(Board board, Cell.Status status)
        { //אם יש מצב של אלכסון שלושה של אחד ברבעון , הרביעי נמצא במרכז הרבעון המנוגד לו
ברצף חמישיה יצור זה רבעון שסיבוב כך פנויים ווקצוותיו
            //בטוח ניצחון מצב זה
            if (board.RecursiaLengthLeftSlant(0, 0, status) == 3)
            {
                if (board.GetAllCells()[4, 4].GetStatus() == status)
                    if (board.GetAllCells()[3, 5].GetStatus() ==
Cell.Status.Empty || board.GetAllCells()[5, 3].GetStatus() ==
Cell.Status.Empty)
                        return true;
            }

            else if (board.RecursiaLengthLeftSlant(3, 3, status) == 3)
            {
                if (board.GetAllCells()[1, 1].GetStatus() == status)
                    if (board.GetAllCells()[2, 0].GetStatus() ==
Cell.Status.Empty || board.GetAllCells()[0, 2].GetStatus() ==
Cell.Status.Empty)
                        return true;
            }

            else if (board.RecursiaLengthRightSlant(3, 2, status) == 3)
            {
                if (board.GetAllCells()[1, 4].GetStatus() == status)
                    if (board.GetAllCells()[2, 5].GetStatus() ==
Cell.Status.Empty || board.GetAllCells()[0, 3].GetStatus() ==
Cell.Status.Empty)
                        return true;
            }
```

```
            else if (board.RecursiaLengthRightSlant(0, 5, status) == 3)
            {
                if (board.GetAllCells()[4, 1].GetStatus() == status)
                    if (board.GetAllCells()[3, 0].GetStatus() ==
Cell.Status.Empty || board.GetAllCells()[5, 2].GetStatus() ==
Cell.Status.Empty)
                        return true;
            }

            return false;


public bool ThereIsFiveSequence(int oreh, LineSequence maxLineSequence,
ToorSequence maxToorSequence, SlantRightSequence maxSlantRightSequence,
SlantLeftSequence maxSlantLeftSequence)
        {//נתון באורך רצף קיים האם
            if (maxLineSequence.GetOreh() == oreh || maxToorSequence.GetOreh()
== oreh || maxSlantLeftSequence.GetOreh() == oreh ||
maxSlantRightSequence.GetOreh() == oreh)
                return true;
            else
                return false;

        }

        public bool AllKindsTotalBlocked(LineSequence maxLineSequence,
ToorSequence maxToorSequence, SlantRightSequence maxSlantRightSequence,
SlantLeftSequence maxSlantLeftSequence)
        { //צידיהם משני חסומים המקסימלים (אלכסונים, טור,שורה) הרצפים סוגי כל אם
            if (maxLineSequence.TotalBlocked() &&
maxToorSequence.TotalBlocked() && maxSlantLeftSequence.TotalBlocked() &&
maxSlantRightSequence.TotalBlocked())
                return true;
            else
                return false;
        }

        public bool AtLeastOneTotalFreeSequence(int oreh, LineSequence
maxLineSequence, ToorSequence maxToorSequence, SlantRightSequence
maxSlantRightSequence, SlantLeftSequence maxSlantLeftSequence)
        {
            if ((maxLineSequence.GetOreh() == oreh &&
maxLineSequence.TotalFree()) ||         ////// משני ופתוח מסויים באורך רצף יש האם
אחד רצף בסוג לפחות הכיוונים
                (maxToorSequence.GetOreh() == oreh &&
maxToorSequence.TotalFree()) ||
               (maxSlantLeftSequence.GetOreh() == oreh &&
maxSlantLeftSequence.TotalFree()) ||
                (maxSlantRightSequence.GetOreh () == oreh &&
maxSlantRightSequence.TotalFree()))
                return true;
            else
                return false;
        }

        public bool AtLeastOneSequenceHalfFree(int oreh, LineSequence
maxLineSequence, ToorSequence maxToorSequence, SlantRightSequence
maxSlantRightSequence, SlantLeftSequence maxSlantLeftSequence)
        { //// אחד רצף בסוג לפחות למחצה ופתוח מסויים באורך רצף יש האם
            if ((maxLineSequence.GetOreh() == oreh &&
maxLineSequence.HalfFree()) ||
```

```
                (maxToorSequence.GetOreh() == oreh &&
maxToorSequence.HalfFree()) ||
                (maxSlantLeftSequence.GetOreh() == oreh &&
maxSlantLeftSequence.HalfFree()) ||
                (maxSlantRightSequence.GetOreh() == oreh &&
maxSlantRightSequence.HalfFree()))
               return true;

            return false;
        }
```

rotationArrow

```
class RotationArrow
    {

        private System.Windows.Forms.PictureBox rotationArrow; //החץ של התמונה
סיבוב
        public enum Direction { Left, Right }; //שמאלה או ימינה - החץ כיוון
        private Direction direction;
        private Quarter quarter; // החץ משויך אליו הרבעון
        private Graphics g; //הגרפיקה משתנה
        private FormGame formGame; //המשחק טופס
        private Board board; //הלוח
        private Pen penBoardFrame; //הלוח מסגרת ציור עט
        private Pen penCells; //בלוח התאים ציור עט
        private FormVictory formVictory; //הניצחון טופס
        private FormDefeat formDefeat; //ההפסד טופס
        private FormTie formTie; //התיקו טופס
        private ArtificialIntelligence AI; //המלאכותית הבינה
        static int time; //המשחק מתחילת הזמן
        static string playerName; //השחקן שם



        public RotationArrow(int x, int y, int width, int height, FormGame
formGame, Image image, RotationArrow.Direction direction, //בונה פעולה
            Quarter quarter, Graphics g, Board board, Pen penBoardFrame, Pen
penCells, FormVictory formVictory,
            FormDefeat formDefeat, FormTie formTie) //בונה פעולה
        {
            this.formVictory = formVictory;
            this.formDefeat = formDefeat;
            this.formTie = formTie;
            this.formGame = formGame;
            this.penBoardFrame = penBoardFrame;
            this.penCells = penCells;
            this.board = board;
            this.g = g;
            this.quarter=quarter;
            this.direction = direction;
            this.AI = AI = new ArtificialIntelligence(board, g, formVictory,
formDefeat);
            this.rotationArrow = new System.Windows.Forms.PictureBox();
            this.rotationArrow.BackColor = Color.Transparent;
            this.rotationArrow.BackgroundImage = image;
            this.rotationArrow.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Stretch;
```

```
            this.rotationArrow.Location = new System.Drawing.Point(x, y);
            this.rotationArrow.Name = "pictureBox";
            this.rotationArrow.Size = new System.Drawing.Size(width, height);
            formGame.Controls.Add(this.rotationArrow);
            this.rotationArrow.Visible = true;
            this.rotationArrow.MouseClick += new
System.Windows.Forms.MouseEventHandler(this.FormGame_MouseClick);

        }

        private void FormGame_MouseClick(object sender, MouseEventArgs e) //ניהול
סיבוב לחצן על הקלקה
        {
            if (board.GetTurnClick() == Board.TurnClick.rotationArrow)   // אם
החץ של הקלקה התור
            {
                this.board.SetTurnClick(Board.TurnClick.cell);
                this.board.SetTurn(Board.Turn.Black);

                if (direction == RotationArrow.Direction.Right)    //הלוח את סובב
לכיוון בהתאם
                    this.quarter.Rotate90DegreesRight();
                else if (direction == RotationArrow.Direction.Left)
                    this.quarter.Rotate90DegreesLeft();

                this.formGame.Refresh();           //בהתאם הלוח את צייר

                board.WinLoseTieForms(time, playerName); //טופס הצג הסתיים המשחק אם
מתאים

                System.Threading.Thread.Sleep(700); //כדי שניה לכמעט הכל עוצר
מולו קורה מה יבין שהמשתמש

                if (board.GetTurnClick() != Board.TurnClick.gameOver &&
board.GetTurn() != Board.Turn.gameOver) //נגמר לא המשחק אם
                {
                    AI.RatePlaceRotate(g, this.board, formVictory, formDefeat);
//ופועלת עבורה המיטבי הצעד את מחשבת מלאכותית הבינה

                    board.SetTurn(Board.Turn.White);   // ללבן התור את מעביר -
למשתמש

                    board.WinLoseTieForms(time, playerName); //הצג הסתיים המשחק אם
מתאים טופס
                }

            }

            else
                MessageBox.Show("click on a cell first"); //השחקן על תור עכשיו אם
שגיאה הודעת תצא תא על לקליק


        }
```

Quarter

```
class Quarter
    {
        private int x; //עוגן של איקס
        private int y; //עוגן של וואי
        private int width; //רבעון רוחב
        private int height; // רבעון גובה
        private Cell[,] nineCells; //שברבעון התאים תשעת של מימדי דוד מערך
        private FormGame formGame; //המשחק טופס
        private Cell.Status[,] nineCellsStatus; //התאים תשעת תכולת של מימדי דו מערך
ברבעון


        public Quarter(int x, int y, int width, int height, FormGame formGame)
// constructor function
        {
            this.x = x;
            this.y = y;
            this.width = width;
            this.height = height;
            this.formGame = formGame;
            this.nineCellsStatus = new Cell.Status[3, 3];
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    this.nineCellsStatus[i, j] = Cell.Status.Empty; //תחילה מגדיר
את כל התאים ברבעון כריקים
                }

            }
            nineCells = new Cell[3, 3];
            int difX = (int)(this.width) / 3;  // width of a cell
            int difY = (int)(this.height) / 3; // height of a cell
            this.nineCells[0, 0] = new Cell(this.x, this.y, difX, difY);
            this.nineCells[0, 1] = new Cell(this.x+difX, this.y, difX, difY);
            this.nineCells[0, 2] = new Cell(this.x + 2*difX, this.y, difX,
difY);
            this.nineCells[1, 0] = new Cell(this.x, this.y+ difY, difX, difY);
            this.nineCells[1, 1] = new Cell(this.x+ difX , this.y + difY, difX,
difY);
            this.nineCells[1, 2] = new Cell(this.x + 2*difX, this.y + difY,
difX, difY);
            this.nineCells[2, 0] = new Cell(this.x, this.y + 2*difY, difX,
difY);
            this.nineCells[2, 1] = new Cell(this.x+difX, this.y + 2 * difY,
difX, difY);
            this.nineCells[2, 2] = new Cell(this.x + 2*difX, this.y + 2 * difY,
difX, difY);
        }


public void Draw(Graphics g, Pen penBoardFrame,Pen penCells) //את המציירת פעולה
הרבעון
        {
            int difX1 = (int)(this.width) / 3; //שליש הוא ברבעון תא כל ואורך רוחב
עצמו הרבעון ורוחב מאורך
            int difY1 = (int)(this.height) / 3;
            for (int i = 0; i < nineCells.GetLength(0); i++)
            {
```

```
                for (int j = 0; j < nineCells.GetLength(1); j++) //אורכי הגדרת
ברבעון התאים ורוחבי
                {
                    this.nineCells[0, 0].SetAll(this.x, this.y, difX1, difY1);
                    this.nineCells[0, 1].SetAll(this.x + difX1, this.y, difX1,
difY1);
                    this.nineCells[0, 2].SetAll(this.x + 2 * difX1, this.y,
difX1, difY1);
                    this.nineCells[1, 0].SetAll(this.x, this.y + difY1, difX1,
difY1);
                    this.nineCells[1, 1].SetAll(this.x + difX1, this.y + difY1,
difX1, difY1);
                    this.nineCells[1, 2].SetAll(this.x + 2 * difX1, this.y +
difY1, difX1, difY1);
                    this.nineCells[2, 0].SetAll(this.x, this.y + 2 * difY1,
difX1, difY1);
                    this.nineCells[2, 1].SetAll(this.x + difX1, this.y + 2 *
difY1, difX1, difY1);
                    this.nineCells[2, 2].SetAll(this.x + 2 * difX1, this.y + 2
* difY1, difX1, difY1);


                    nineCells[i, j].Draw(g,penCells); //ברבעון התאים ציור
                    nineCells[i, j].DrawStatus(g); //ברבעון התאים תכולת ציור
                }

            }
            g.DrawRectangle(penBoardFrame, this.x, this.y, this.width,
this.height); //הרבעון מסגרת ציור
        }


public void SetNineCellsStatus() //ברבעון התאים תשעת תכולת  ערכי עדכון
        {
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    nineCellsStatus[i, j] = this.nineCells[i, j].GetStatus();
                }
            }
        }

        public void Rotate90DegreesLeft() // rotates the quarter 90 degrees to
the left
        {
            SetNineCellsStatus();

            nineCells[0, 0].SetStatus(nineCellsStatus[0, 2]);
            nineCells[0, 1].SetStatus(nineCellsStatus[1, 2]);
            nineCells[0, 2].SetStatus(nineCellsStatus[2, 2]);
            nineCells[1, 0].SetStatus(nineCellsStatus[0, 1]);
           //nineCells[1, 1] stays the same//
            nineCells[1, 2].SetStatus(nineCellsStatus[2, 1]);
            nineCells[2, 0].SetStatus(nineCellsStatus[0, 0]);
            nineCells[2, 1].SetStatus(nineCellsStatus[1, 0]);
            nineCells[2, 2].SetStatus(nineCellsStatus[2, 0]);

            SetNineCellsStatus();
            //DrawStatus(g);

        }
```

```
        public void Rotate90DegreesRight() // rotates the quarter 90 degrees to
the right
        {
            SetNineCellsStatus();

            nineCells[0, 0].SetStatus(nineCellsStatus[2, 0]);
            nineCells[0, 1].SetStatus(nineCellsStatus[1, 0]);
            nineCells[0, 2].SetStatus(nineCellsStatus[0, 0]);
            nineCells[1, 0].SetStatus(nineCellsStatus[2, 1]);
            //nineCells[1, 1] stays the same//
            nineCells[1, 2].SetStatus(nineCellsStatus[0, 1]);
            nineCells[2, 0].SetStatus(nineCellsStatus[2, 2]);
            nineCells[2, 1].SetStatus(nineCellsStatus[1, 2]);
            nineCells[2, 2].SetStatus(nineCellsStatus[0, 2]);

            SetNineCellsStatus();
            //this.DrawStatus(g);
        }

        public bool MouseClick(Graphics g, int x0, int y0, Board.Turn turn)
//ריק תא על הייתה הלחיצה כאשר אמת מחזירה הפעולה
        {
//כדורית כבר בו שיש תא על הלחיצה כאשר שקר מחזירה הפעולה
            for (int i = 0; i < 3; i++)
//בלחיצה הטיפול את אליו ומעבירה הלחיצה התרחשה בה תא באיזה בודקת הפעולה
            {
                for (int j = 0; j < 3; j++)
                {
                    if (this.nineCells[i, j].Inside(x0, y0))
                    {
                        return nineCells[i, j].PlaceBall(g, turn);
                    }
                }

            }
            return true; ;
        }
```

Cell

```
    class Cell
    {
        public enum Status { Black, White, Empty }; //התא תכולת
        private Status status;
        private int x; //העוגן ששל איקס
        private int y; //העוגן של וואי
        private int width; //התא רוחב
        private int height; //התא גובה
        private int radius; //שבתא העיגול רדיוס
        private Point center; //שבתא העיגול מרכז

        public Cell(int x, int y, int width, int height) // constructor
function
        {
            this.x = x;
            this.y = y;
            this.width = width;
            this.height = height;
            this.status = Status.Empty;
            this.radius = (int)(width / 2);
```

```
            this.center = new Point(x + (int)(width / 2), y + (int)(height /
2));
        }

        public double Distance(int x1, int y1, int x2, int y2)      //////מחזירה
נקודות שתי בין המרחק את
        {
            return  Math.Sqrt(Math.Pow(x2 - x1, 2) + Math.Pow(y2 - y1, 2));
        }


        public void PlaceBlack(Graphics g) // fills the cell in black
        {
            g.DrawImage(Properties.Resources.blackGula, this.x, this.y,
this.width, this.height);
            this.status = Cell.Status.Black;
        }

        public void PlaceWhite(Graphics g) // fills the cell in white
        {
            g.DrawImage(Properties.Resources.whiteGula, this.x, this.y,
this.width, this.height);
            this.status = Cell.Status.White;
        }

        public bool PlaceBall(Graphics g, Board.Turn turn) // places a ball in
a cell
        {                                                       //כאשר שקר מחזירה
כדורית כבר בו שיש תא על הייתה הלחיצה
            if (this.status != Status.Empty)                    //ריק שהתא מוודא
כדורית בו שמציבים לפני
            {                                                   // כאשר אמת מחזירה
ריק תא על הייתה הלחיצה
                MessageBox.Show("click only in empty cells");
                return false;
            }
            else
            {
                if (turn == Board.Turn.White)
                {
                    this.PlaceWhite(g);
                }
                if (turn == Board.Turn.Black)
                {
                    this.PlaceBlack(g);
                }
                return true;
            }
        }

        public void DrawStatus(Graphics g)           ///////////תכולת את משחזרת
שנמחק לאחר התא
        {
            if (this.status == Cell.Status.Black)
            {
                PlaceBlack(g);
            }
            if (this.status == Cell.Status.White)
            {
                PlaceWhite(g);
            }
        }
```

FormGame

```
public partial class FormGame : Form
    {     ////we are white, computer is black
        FormVictory formVictory; //ניצחון טופס
        FormDefeat formDefeat; //הפסד טופס
        FormTie formTie; //תיקו טופס
        Graphics g; //גראפיקה משתנה
        Board board; //המשחק לוח
        Brush brush; //הצביעה של בראש
        Pen penBoardFrame;  //הלוח מסגרת ציור עט
        Pen penCells; //הלוח תאי ציור עט
        int xBoard; //הלוח עוגן איקס
        int yBoard; //הלוח עוגן וואי
        int widthBoard; //הלוח רוחב
        int heightBoard; //הלוח גובה
        MainMenu mainMenu; //ראשי תפריט טופס
        int time; //המשחק מתחילת שעבר זמן
        static string playerName; //השחקן שם


        public FormGame(MainMenu mainMenu) //בונה פעולה
        {
            InitializeComponent();
            formVictory = new FormVictory(mainMenu);
            formDefeat = new FormDefeat(mainMenu);
            formTie = new FormTie(mainMenu);
            xBoard = (int)(ClientRectangle.Width / 5);
            yBoard=(int)(ClientRectangle.Height / 5);
            widthBoard=(int)(3*((int)(ClientRectangle.Width/5)));
            heightBoard = (int)(3 * ((int)(ClientRectangle.Height / 5)));
            g = CreateGraphics();
            brush = new SolidBrush(Color.DarkRed);
            penBoardFrame = new Pen(brush, 4);
            penCells = new Pen(brush, 2);
            board = new Board(xBoard, yBoard, widthBoard, heightBoard, this, g,
penBoardFrame, penCells, formVictory, formDefeat, formTie);
            this.mainMenu = mainMenu;
            time = 0;

        }

public void FormGame_MouseClick(object sender, MouseEventArgs e) //את מנהלת
בטופס ההקלקה
        {
            board.MouseClick(g, e.X, e.Y);
            RotationArrow.SetTime(time);
            board.WinLoseTieForms(time, playerName); //המשחק הסתיים אם בדיקה
        }



        private void FormGame_Paint(object sender, PaintEventArgs e) //כל אחרי
בהתאמה מתעדכן הלוח תור
        {
            g = CreateGraphics();
            xBoard = (int)(ClientRectangle.Width / 5);
            yBoard = (int)(ClientRectangle.Height / 5);
            widthBoard = (int)(3 * ((int)(ClientRectangle.Width / 5)));
            heightBoard = (int)(3 * ((int)(ClientRectangle.Height / 5)));
            board.SetX(xBoard);
```

```
            board.SetY(yBoard);
            board.SetWidth(widthBoard);
            board.SetHeight(heightBoard);
            board.SetCenterCellsRadius(xBoard, yBoard, widthBoard,
heightBoard);
            board.SetLocationArrows();
            board.Draw(g);
            board.DrawStatus(g);
        }


        private void HomeButton_Click(object sender, EventArgs e) //כפתור על לחיצה
הראשי לתפריט המשתמש את מעבירה הבית
        {
            this.Hide();
            mainMenu.Show();

        }

        private void FormGame_FormClosing(object sender, FormClosingEventArgs
e) //מהתוכנה יוצא היציאה כפתור על לחיצה
        {
            Application.Exit();
        }



        private void timeCounter_Tick(object sender, EventArgs e) //מציג הטיימר
המשחק מתחילת עבר בשניות זמן כמה לשחקן
        {
            time++;
            timerLabel.Text =  time.ToString() + " sec";
        }

        public static void SetPlayerName(string playerName0) //סטטי במשתנה שימוש
משתמש שם
        {
            playerName = playerName0;
        }
```

MainMenu

```
public partial class MainMenu : Form
    {
        RecordsForm formRecords; //שיאים טבלת טופס
        FormInstructions formInstructions; //הוראות טופס
        FormGame formGame; //המשחק טופס
private void instructionsButton_Click(object sender, EventArgs e) //לפורם מעבר
הוראות
        {
            this.Hide();
            formInstructions = new FormInstructions(this);
            formInstructions.Show();
        }

        private void exitButton_Click(object sender, EventArgs e) // באמצעות יציאה
פנימי לחצן
        {
            Application.Exit();
        }
```

```
        private void startButton_MouseClick(object sender, MouseEventArgs e)
//השחקן שם הקלדת לאחר למשחק כניסה
        {
            if (nameBox.Text == "enter your name" || nameBox.Text == "")
                MessageBox.Show("Enter Your Name First");
            else
            {
                this.Hide();
                FormGame.SetPlayerName(nameBox.Text);
                RotationArrow.SetPlayerName(nameBox.Text);
                formGame = new FormGame(this);
                formGame.Show();
            }
        }

private void nameBox_Click(object sender, EventArgs e) //הטקסט תיבת על הקלקה
        {
            nameBox.Text = "";
        }
```

PlacingOption

```
class PlacingOption
    {
        private int rate;  //אפשרות - למהלך הניקוד
        private int i; // ערך I של המקום בו נמקם אבן בלוח
        private int j;// ערך J של המקום בו נמקם אבן בלוח

        public PlacingOption(int i, int j, int rate) //בונה פעולה
        {
            this.i = i;
            this.j = j;
            this.rate = rate;
        }
```

PlacingOptions

```
class PlacingOptions
    {
        private List<PlacingOption> options; // ידי על לביצוע אפשריים מהלכים של רשימה
המחשב

        public PlacingOptions() //בונה פעולה
        {
            this.options=new List<PlacingOption>();
        }


        public PlacingOption MaxOption() //ביותר הגבוה הניקוד עם המהלך את שמחזירה פעולה
ברשימה
        {
            if (this.options.Count == 0)
                return null;

            int iMax = -1;
            int jMax = -1;
            int maxRate = int.MinValue;
            PlacingOption maxPlacingOption = new PlacingOption(iMax,jMax,
maxRate);

            for (int i = 0; i < options.Count; i++)
            {
                if (options[i].GetRate() > maxRate)
                {
```

```
                maxRate = options[i].GetRate();
                iMax = options[i].GetI();
                jMax = options[i].GetJ();
                maxPlacingOption.SetIJRate(iMax, jMax, maxRate);
            }
        }

        return maxPlacingOption;

    }

    public List<PlacingOption> GetOptions() //הרשימה את מחזירה
    {
        return this.options;
    }

    public void Add(PlacingOption placingOption) //חדש מהלך הרשימה בסוף מוסיפה
    {
        this.options.Add(placingOption);
    }
```

RecordsForm

```
public partial class RecordsForm : Form
    {
        MainMenu mainMenu; //ראשי תפריט טופס
        string playerName; //שחקן שם
        List<Score> scores; //התוצאות כל של רשימה
        Label[] times; //הזמנים כל של רשימה
        Label[] names; //השחקנים שמות כל של רשימה

        public RecordsForm(MainMenu mainMenu, string playerName) //בונה פעולה
        {
            InitializeComponent();
            this.mainMenu = mainMenu;
            this.playerName = playerName;
            scores = new List<Score>();
            times = new Label[] { time0, time1, time2, time3, time4, time5,
time6 };
            names = new Label[] { playerName0, playerName1, playerName2,
playerName3, playerName4, playerName5, playerName6 };
            ResetScoresList("timeTable.txt");
            SortScores();
            SetLabels();
        }

public void ResetScoresList(string fileName)
        {                                                   //טקסט מקובץ קוראת הפעולה
הקודמים מהמשחקים התוצאות את

            string time;
            string playerName;


            using (StreamReader streamReader = File.OpenText(fileName))
            {
                //time = "";                 ///מספר ואז שם קודם
                //playerName = "";

                //while (playerName != null)
                //{
                //    playerName = streamReader.ReadLine();
                //    if (playerName == null)
```

```
        //        return;
        //    scores.Add(new Score(playerName));
        //    time = streamReader.ReadLine();
        //    scores.Last().SetTime(int.Parse(time));
        //}


        time = "";                  ///שם ואז מספר קודם
        playerName = "";

        while (time != null)
        {
            time = streamReader.ReadLine();
            if (time == "" || time == null)
                return;
            scores.Add(new Score(int.Parse(time)));
            playerName = streamReader.ReadLine();
            scores.Last().SetPlayerName(playerName);
        }


    }
}

public void SortScores()  ///הארוך אל הקצר הניצחון זמן לפי התוצאות רשימת את מארגן
{
    Score temp;

    for (int i = 0; i < scores.Count; i++)
    {
        for (int j = i + 1; j < scores.Count; j++)
        {
            if (scores[i].GetTime() > scores[j].GetTime())
            {
                temp = scores[i];
                scores[i] = scores[j];
                scores[j] = temp;
            }
        }
    }
}

public void SetLabels() ///המיון לפי בלייבלים הטקסט את מעדכן
{
    for (int i = 0; i < times.Length; i++)
    {
        times[i].Text = scores[i].GetTime().ToString();
        names[i].Text = scores[i].GetPlayerName();
        if (i % 2 == 0)
        {
            times[i].BackColor = Color.FromArgb(209, 215, 232);
            names[i].BackColor = Color.FromArgb(209, 215, 232);
        }
        else
        {
            times[i].BackColor = Color.FromArgb(233, 237, 245);
            names[i].BackColor = Color.FromArgb(233, 237, 245);
        }
```