

סמל ביה"ס: 140129

שם ביה"ס: קריית נוער



# עובדת גמר

למילוי תפקיד של הדרישות לקבלת תואר

הנדסאי תוכנה

משחק רשות שחמט



מגייש

**בצלאל אברהמי 325004752**

בהנחיית מר אילן פרץ

שנה"ל תשפ"ב

2022

## תוכן עניינים

<b>5</b>	<b>הצעת הפרויקט</b>
<b>8</b>	<b>1. הצהרת הסטודנט ואישור הגשה</b>
<b>9</b>	<b>2. מבוא</b>
<b>10</b>	<b>3. מדריך למשתמש</b>
10	3.1 תיאור המשחק
20	3.2 דרישות טכניות
20	3.3 הרצת המשחק והנחיות שימוש
21	3.2.1 הרצת השרת והשימוש בו
22	3.2.2 הרצת הלקוב ווהשימוש בו
25	3.2.3 הפעלת המשחק וממשק משתמש גרפי ((UI))
<b>44</b>	<b>4. מדריך לתוכנה</b>
44	4.1 הדרישות ממערכת התוכנה
45	4.2 ארכיטקטורת המערכת
45	4.2.1 מודל שרת-לקוב Client-Server Model
45	4.2.2 פרוטוקול תקשורת TCP/IP
45	4.2.3 Sockets
46	4.2.4 שקעים
47	4.2.5Threads
48	4.3 תהליכיים עיקריים וזרימת המידע
48	4.3.1 תהליך ראשי
53	4.4 מחלקות הפרויקט
53	תרשי UML של המחלקות המשותפות ללקוב ולשרת
59	4.4.1 תרשימים UML של המחלקות מצד השרת
62	4.4.2 תיעוד המחלקות מצד השרת
81	4.4.3 תרשימים UML של המחלקות מצד הלקוב
89	4.4.4
89	תיעוד המחלקות מצד הלקוב
140	4.4.5 תיעוד המחלקות המשותפות ללקוב ולשרת
224	<b>4.5 אלגוריתמים ופעולות נבחרות</b>
224	4.5.1 אלגוריתם מינימקס עם גיזום
227	4.5.2 פונקציית הערכה
233	4.5.3 אלגוריתם 3 חישוב אלטרנטיבות לשם משתמש תפוא
236	4.5.4 אלגוריתם/פעולה 4 חישוב אינומים של כלים
240	4.5.5 AppSocket 5 פעולה
242	4.6 מבני נתונים עיקריים
<b>243</b>	<b>5. סיכום אישי</b>
243	5.1 אתגרים וקשיים איתם התמודדתי

243	5.2 מה הפרויקט תרם ל'
244	5.3 הצעות לשיפור
245	5.4 מסקנות
<b>245</b>	<b>6. תודות</b>
<b>245</b>	<b>7.ביבליוגרפיה</b>

## הצעת הפרויקט

סמל מוסד: 140129

שם מقلלה: קריית נוער – כנפי רוח, ירושלים.

שם הסטודנט: בצלאל אברהמי

ת"ז הסטודנט: 325004752

שם הפרויקט: משחק שחמט\Chess

תיאור הפרויקט:

משמעותו של המשחק הוא לחקלא אסטרטגי מופשט וענף ספורט המועד לשני שחקנים.

### מהלך המשחק

השחקן בכלים הלבנים (להלן: "לבן") הוא הראשון למשחק. כל שחקן מניע בתורו את אחד הכלים שברשותו (כלי אחד בלבד, להוציא במקורה של הצרחה), כאשר לכל כלי אוון תנועה הייחודי לו, כמפורט לעיל, וכך עד הכרעה. השחקן יכול להזיז כל כלי בתורו ולהזיזו لأنו שركירצה, ובבלבד שלא עבר על כללי תנועת הכלים המפורטים לעיל. כמו כן, אם המלך של השחקן מסוים נחשף לאיום מצד כלי של היריב (מצב המכונה "שח"), חובה על השחקן לעשות כל שביכולתו כדי לבטל את המצב הזה. אם לא ניתן לבטל את האיום, יפסיד במשחק, במצב המכונה מת. כל מהלך שאינו מביא לכך אוון מיידי לא יהיה חוקי. בנוסף לכך, כל מהלך שיביא באופן מיידי לכך שהמלך של השחקן המשחק יימצא במצב של שח גם אינו חוקי (מלך הוא הכלי היחיד שאסור להעמידו במכון תחת איום של כלי אחר).

הכאה: אם ניצב אחד מכליו של היריב בדרכו של הכלי הנע, באפשרותו להכות ("לأكل") אותו, בתנאי שמדובר במצב חוקי. בשחמט (שלא כמו בדמקה, למשל) ההכאה היא זכות ולא חובה. בהכאת כלי אחר, הכוונה לסלוקו מהלכה והצבת הכלי המכיה במקומו. את המלך אין מכחים. מקובל להכריז, במצב בו הוא חשוף לאיום ולא ניתן להגן עליו, על שחמט ("שח", בתוספת "מט"; יש שמקצרים ל-"מט"). כל הכלים מכחים דרך הילוכם (למשל, הרץ יכה רק על אלכסונו), מלבד הרגלי שזו קדימה אך מכיה באלכסון. מקרה מיוחד של הכאה הוא "הכאה דרך הילוכו".

### שלבי המשחק

השלב הראשון במשחק הוא הפתיחה. בשלב זה מפתח השחקן את כליו, החסומים מהחורי שורת הרגלים, על מנת לאפשר להם לשלוט על מרכזו הלאו ולאיים על ההגנה של היריב. שלב הפתיחה גם כולל את פינוי אחד האגפים על מנת לאפשר הצרחה שטטרתת לבצר את המלך בפינה ולהרחקו מהמרכז החשוב. פתיחה אופיינית כוללת בדרך כלל דילוג כפול עם הרגלי של המלך או המלכה, הוצאה אחד או שני הפרשים והוצאה אחד או שני הרכבים, ואו הצרחה. ישנן פתיחות הנקראות גמביט, בהם אחד הצדדים מקריב חומר - לרוב רגלי - על מנת להשתלט על המרכז. מספר הפתיחות הוא רב ומספר הוריאנטים שלهنן רב עוד יותר. שחקנים מקצוענים משננים בעל פה פתיחות רבות על מנת שלא לבצע טעויות בתחילת המשחק, דבר שייתן יתרון מוקדם ליריב.

השלב השני הוא מציצה, התרחש אחרי שרוב הכלים כבר פותחו, והקרב מתנהל במרחב הריפותו במרכז הלה, בדרך כלל תוך כדי חילופי כלים וניסיון לשבור את מערכם של הרגלים של הצד השני. בשלב זה אפשר למשוך מספר רב של טקטיקות לצבירה יתרון חומי ועמדתי (ראו בהמשך).

השלב השלישי הוא סיום, ושלב זה קורה כאשר נעשו מספר רב של חילופי כלים מבלי שהושג יתרון מכרייע לצד כלשהו. שלב זה כולל בדרך כלל מספר קטן של כלים לכל צד (מלך, קצין ומספר רגליים) וכן מנסים שחקנים להכריע אחד את השני בדרך כלל על ידי הכתרת אחד הרגלים לכלי כבד (כגון מלכה). בשלב זה, האiom במת על המלך קטן ולכון המלך הופך לכלי פעיל, המגן על הרגלים ותומך בהתקדמותם אל עבר השורה השמינית. מכיוון שהסיום דורש משחק מדויק, שחקנים רבים לומדים בקפידה עמדות סיום נפוצות.

### הגדרת הבעה האלגוריתמית:

הבעיה הכלכלית שפתרתי היא משחק שחמט בראש מרובה שחקנים (רשומים, אורחים ומוחשבים) ומשחקים, בצורה סימולטנית. בעיות יותר ספציפיות:

- התהברות בתור שחקן רשום \ אורח.
- ייצירת סטטיסטיות למשתמשים.
- שימוש במסד נתונים לשמרות משחקים.

### רקע תיאורי בתחום הפרויקט:

לימודי י"ג י"ד הנדסי תוכנה שבהם למדנו בין היתר: עבודה עם מסד נתונים בשפת SQL, תוכנות מונחה עצמים, מבני נתונים (רשימות מקשורות, מערכים...), מינימקס (כולל גיזום אלף בטה),

**Networking** (סוקטים, TCP\IP, מודל שרת ללקוח, מודל השכבות...), שימוש בת'רדים, מודל MVC, עיצוב ממשק גרפי למשתמש באמצעות `.java swing`.

#### תהליכיים עיקריים בפרויקט:

- הרשמה של משתמש חדש.
- התחברות של משתמש רשום או אורח.
- משחק נגד שחזור ממוחשב בעזרת שימוש באлогריתם מינימקס.
- שירות עובד במקביל לצורך ריבוי שחנים ומשחקים,
- שמירה בסיס נתונים (תוצאות משחקים שהסתינו ושמירת משחקים שהופסקו)
- הפקת סטטיסטיות ושאלות עדכון מסד הנתונים

#### תיאור טכנולוגיה וארქיטקטורה:

שימוש במודול שירות לניהול משחק בין לקוחות, שימוש בסוקטים להעברת נתונים בראש, שימוש בת'רדים לבניית נציגים וכמה משחקים במקביל.

#### שפת התכנות צד לקוח ושרת :

java (16.0.2) Win10

#### פרוטוקולי תקשורת:

התקשורת בין השירות ללקוחות מתבצע באמצעות העברת הודעות על סוקטים TCP/IP  
ההודעות מכילות תוכנה בשם "MessageType" שלפיה המქבל ידע איך לנוט את ההודעה, ועוד  
תכונות שנלוות לכל סוג הודעה.

#### תיאור מסד נתונים :

מסד נתונים טבליי מסוג Access עם כמה טבלאות ובניהם: משתמשים רשומים, משחקים שהסתינו, וממשחקים שעדיין לא הסתיימו וכו'... ובאמצעות שפת SQL נבצע פעולות על המסד.

#### לוחות זמינים :

מרץ	אפריל	מאי	יוני	יולי	אוגוסט	ספטמבר	אוקטובר	נובמבר	דצמבר	ינואר	פברואר	מרץ
												פתיחת המערכת
												תיכנון מבני הנתונים ואלגוריתמים
												כתיבת הקוד
												עיצוב ממשק המשתמש
												ניפוי שגיאות ובדיקות איות
												הרצה התוכנית

## 1. הצהרת הסטודנט ואישור הגשה

### הצהרת הסטודנט

שם הסטודנט: **בצלאל אברהמי** ת"ז: **325004752**

אני החתום מטה, מצהיר בזאת כי פרויקט הגמר וספר הפרויקט המצור'ב נעשו על ידי בלבד. פרויקט הגמר נעשה על סמך הנושאים שלמדתי באופן עצמאי, ועל בסיס הנחייתו של המנחה האישית. מקורות המידע בהם השתמשתי לביצוע פרויקט הגמר מפורטים בראשימת המקורות (ביבליוגרפיה) שנמצאת בסוף ספר הפרויקט.  
אני מודע לאחריות שהנני מקבל על עצמי על ידי חתימתה על הצהרה זו שכל הנאמר בה אמת ורק אמת.

תאריך: \_\_\_\_\_ חתימה הסטודנט: \_\_\_\_\_

### אישור המנחה האישית

הריני מאשר שהפרויקט בוצע בהנחייתו, בדקתי את קוד הפרויקט וקבעתי את ספר הפרויקט ומצאת כי הוא מוכן לצורך הגשת הסטודנט להגנה על פרויקט גמר.

תאריך: \_\_\_\_\_ חתימה: \_\_\_\_\_ שם המנחה: \_\_\_\_\_

### אישור ראש המגמה

הריני מאשר הגשת הסטודנט להגנה על פרויקט הגמר.

שם ראש המגמה: \_\_\_\_\_ תאריך: \_\_\_\_\_ חתימה: \_\_\_\_\_

## 2. מבוא

פרויקט הגמר מימוש תוכנה **למשחק רשות מבוסס משחק לוח לשני שחקנים** בשם "שחמט". הסבר על המשחק, הפעלתו והשימוש בו, נמצא במדריך למשתמש בפרק 4 כולל הצגת צלומי מסך והסבירים מפורטים לכל השלבים.

המשחק תוכנן ופותח על פי הדרישות מערכתי התוכנה הבאות, ואשר מפורטות בפרק 5:

- ריבוי משחקים במקביל בין שחקנים אנושיים שמתחרבים מרוחק באמצעות רשות האינטרנט.
- לכן מערכת התוכנה עושה שימוש בארכיטקטורה המבוססת על **מודול שירות/לקוח** (Client/Server) הכולל עבודה עם **ש愧ים** (Sockets) בפrotocol TCP/IP, ושימוש בתהליכיונים **Threads**.

**בסעיף 5.2 יש הסבר מעמיק על ארכיטקטורת המערכת ורכיבתה.**

- לאפשר לשחקן רשות (אנושי) ללחוץ לשחק מול שחקן ממוחשב עם בינה.
- לכן מערכת התוכנה עושה שימוש **באלגוריתם Minimax** למינימום השחקן הממוחשב.

**בסעיף 5.5.1, יש הסבר מעמיק על אלגוריתם מינימקס המממש את פעולה השחקן ממוחשב.**

- לאפשר התחברות (Login) של שחקנים אורחים ורשמיים, שמירת תוצאות משחקים שהסתימו, שמירת משחקים שנפסקו וטעינהם בהמשך, לקבלת סטטיסטיות ולאפשר פעולות עדכון.
- לכן מערכת התוכנה עושה שימוש **בבסיס נתונים מסוג Access** ובו שמורות טבלאות המכילות מידע על השחקנים הרשמיים, על תוצאות המשחקים, על מצב המשחקים ללא הסתיימו ועוד ...

**בסעיף 5.2.4 יש הסבר מעמיק יותר על בסיס הנתונים ותיאור כל הטבלאות שמורות בו.**

- לכתוב את קוד התוכנה בשפת Java תוך שימוש בעקרונות של תוכנות מונחה עצמים מתקדם.
- לכן נעשה שימוש במחלקות, **ירשה ופולימורפיים**, **משמעותים** (Interfaces), **אוסףים גנריים**, וכמוון **שימוש בתהליכיונים** (Threads) לצורך ניהול **משחקים במקביל**.

**בסעיף 5.4 יש הסבר מקיף על כל המחלקות בפרויקט (משמעותי API, תרשימי UML, תיעוד...).**

**בסעיף 5.2.5 יש הסבר מעמיק על השימוש בתהליכיונים ולמה היה צריך אותם.**

פרק 6 מסכם את החוויה האישית שלי במהלך העבודה על הפרויקט – התמודדות עם קשיים ובעיות שהיוו, הכלים והמיומנויות שרכשתי תוך כדי העבודה על הפרויקט ושאני לוקח איתני להמשך, המסקנות שהגיעהתי אליו לאחר סיום הפרויקט, הצעות לשיפור הפרויקט לו יהיה לי יותר זמן לעבוד עליו, ועוד.

## 3. מדריך למשתמש

בחלק זה יוסבר על המשחק, הלוח והכליים, חוקי המשחק והמהלכים. בנוסף נתאר את הקבצים המוגשים ביחד עם ספר פרויקט זה (שם חלק בלתי נפרד מהפרויקט כולו), וסביר איך להריץ את התוכנה ואייך לשימוש בה מנוקדת מבטו של **המשתמש**.

נתחיל בפירוט הדרישות הטכניות להריצה, נסביר בהסבר על קבצי ההרצה ואופן הרצתם, ולבסוף נסביר איך לשחק במשחק הילכה למעשה – נעשה זאת על ידי תיאור ממשק המשמש הגרפי (GUI) בלבד צלומי מסך להמחשה.

### 3.1 תיאור המשחק

#### • תיאור המשחק

שחמט הוא משחק לוח אסטרטגי מופשט ונוף ספורט המיועד לשני שחקנים. זה אחד מהמשחקים השכיחים והמורכבים ביותר הקיימים בתרבות האנושית. המשחק מקובל ברחבי העולם כתחריב וכספורט תחרותי אחד. אדם העוסק במשחק שחמט באופן מקצועי נקרא שחמטאי.

ニיצחון במשחק מושג, כאשר אחד המלכים מאויים בשח על ידי אחד מכל' היריב, אין יכול להזק את הכל' המאיים, לחסום את האiom על ידי כל' אחר או להימלט מהאיום לשבצת בה לא יהיה מאויין.

#### • לוח והכליים

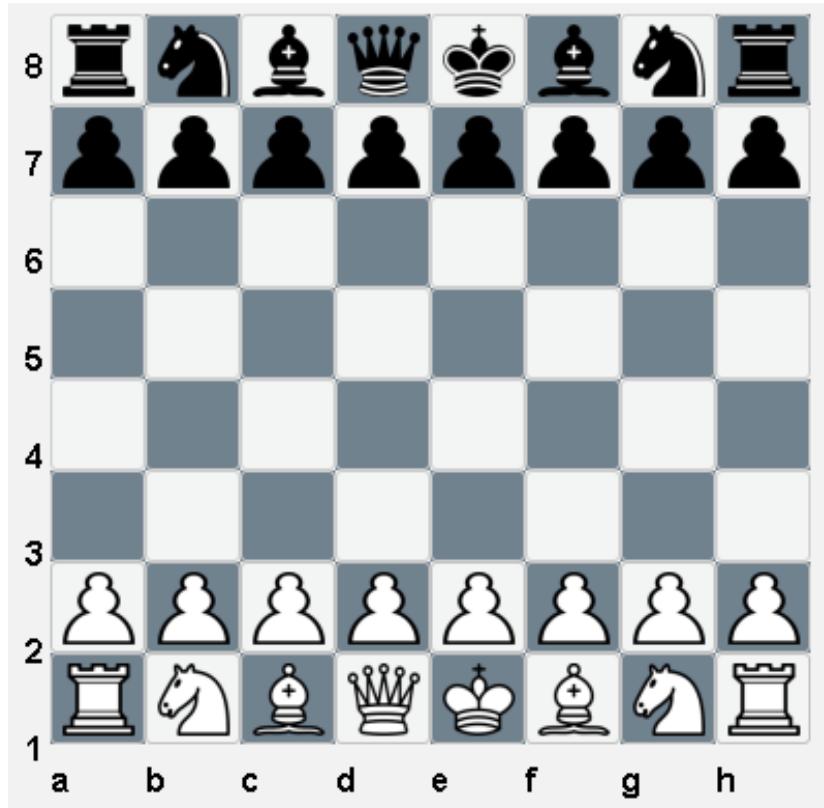
לוח המשחק בגודל 8X8. הכלים של כל שחקן הם: מלך(8X), צריה(2X), פרש(2X), רץ(2X), מלכה(1X) ומילר(1X). צבעם של הכלים נקבעים לפי צבע השחקן שלהם. לבן לשחקן הלבן, ושחור לשחקן השחור.

להלן הכלים הלבנים (לשחקן השחור יש אותם כלים בדגם, רק בצבע שחור):

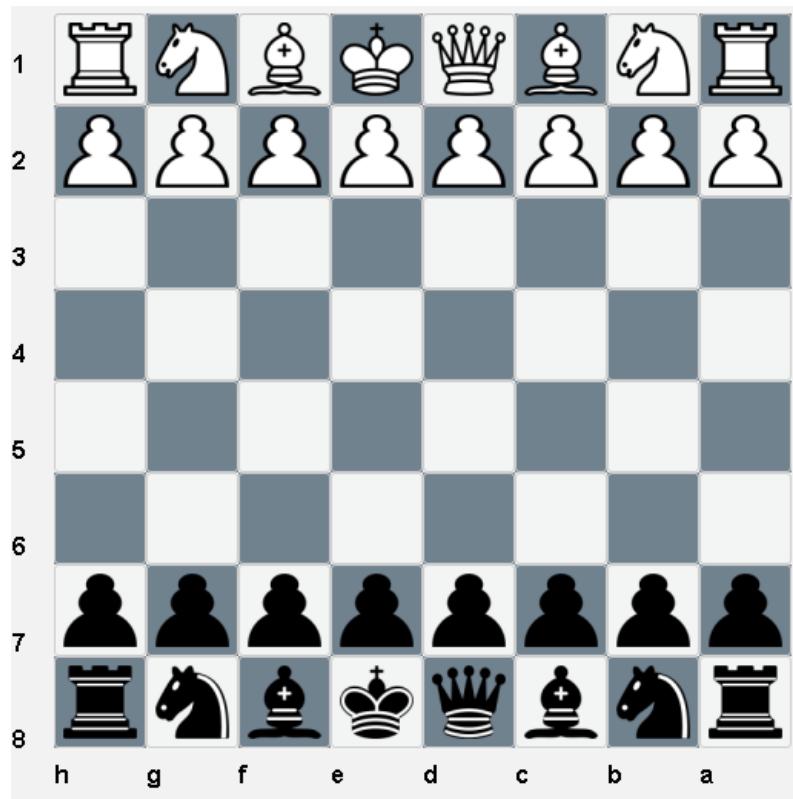


#### • מצב התחלתי

בהתבה שמתחלים מעמדת ההתלה הסטנדרטית השחקן הלבן משחק ראשון, והלוח 8X8 מסודר בצורה זו:



מנקודת המבט של השחקן השחור הלוח ייראה בדיקו אותו הדבר, רק שהכל יהיה הפוך. לדוגמה, עמדת זו תראה לשחקן השחור כז':

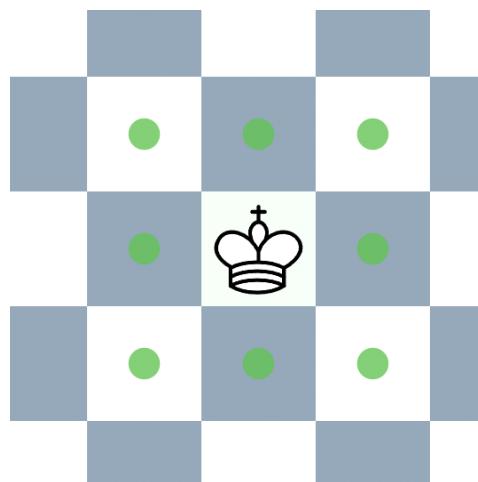


### • חוקי המשחק והמהלכים

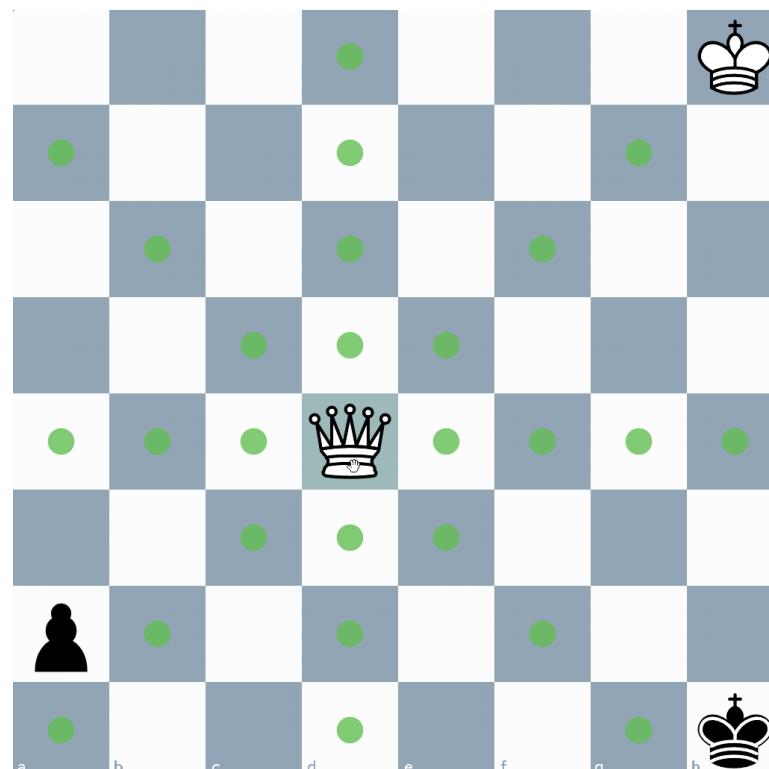
#### תנועת הכלים

בשחמט לכל כלי דרך תנועה שונה. לעיתים אי אפשר לנוע לtower משכצת שנמצאת בה כלי מהצעע של המשחק, או לעبور מעל כלי צזה ( מלבד הפרש). בנוסף, ישנה אפשרות של "הכאה" (מכונה גם: "אכילה" או "לקיחה"): להסיר כלי של היריב מהלוות. הכאה מתבצעת לרוב כדרך התנועה ( מלבד הרגלי) הכלי, נע תמיד למשכצת בה היה הכלי היריב ( מלבד המציב של "הכאה דרך היילוכו"). במסע לעולם לא יוזו יותר מכלי אחד ( מלבד "הצרכה"). הכלים בעלי תנועה "ארוכה" (מלכה, צריח ורץ) יכולים לנוע כרצונם כל עוד אין בדרך כל מיצבעם (שאז עליהם לעמוד לעצור לפניו) או כל מהצעע השני (שאז עליהם לעמוד לפניו, או להכות אותו).

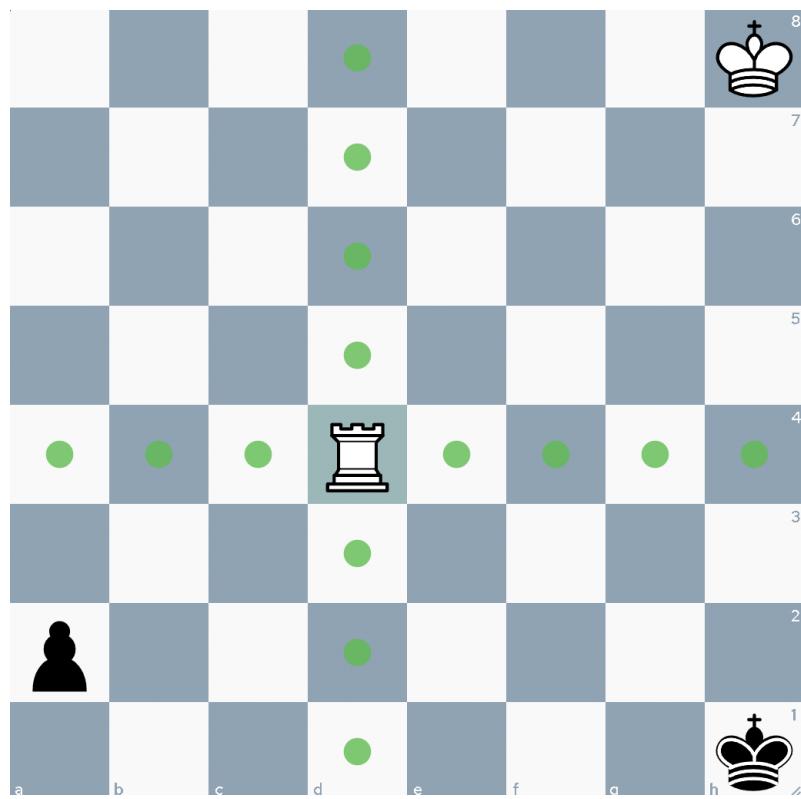
מלר: המלך יכול לנוע משכצת אחת לכל כיוון



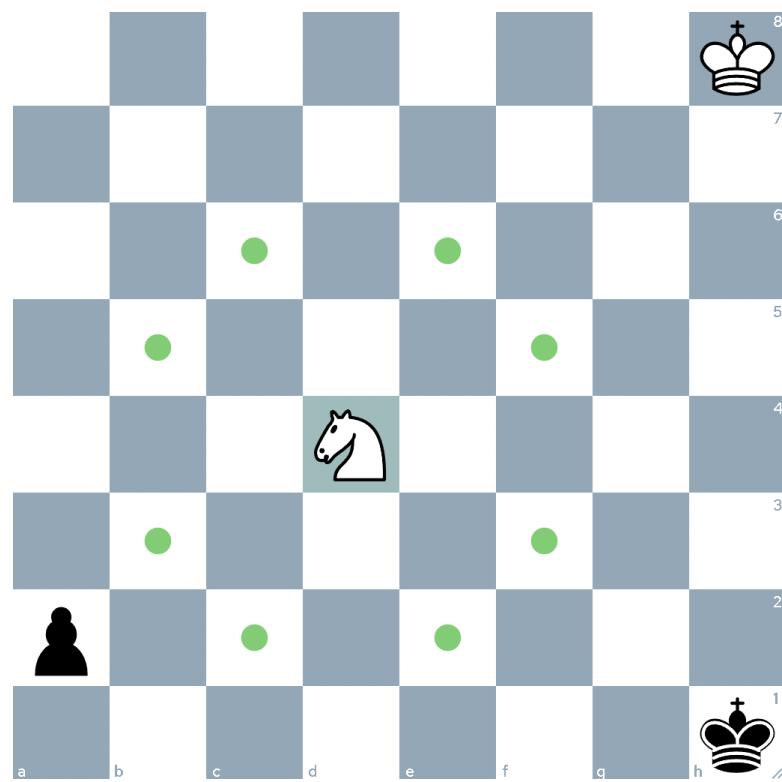
מלכה: המלכה יכולה לנוע מספר בלתי מוגבל של משבצות לכל כיוון.



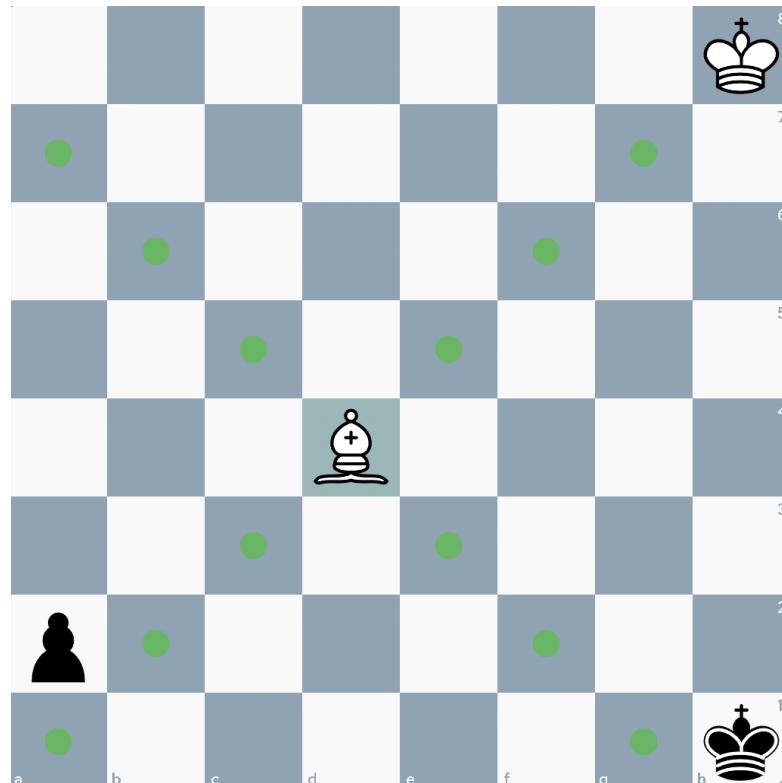
צריך: הצריח יכול לנוע מספר בלתי מוגבל של משבצות בתורות ובסורות.



פרש: הפרש יכול לנوع שתי ערוגות בטורים או בשורות ואז ערוגה נוספת בניצב לכיוון בו החל ללקת, כך מתקבלת צורת ה- L או האות R. לפרש (בלבד) מותר לדלג מעל כלים אחרים משני הצבעים במהלך תנועתו, אך הערוגה אליה הוא מגיע בסוף התנועה חייבת להיות פנימה או מאויישת על ידי כלי מהצבע הנגדי לצבעו.

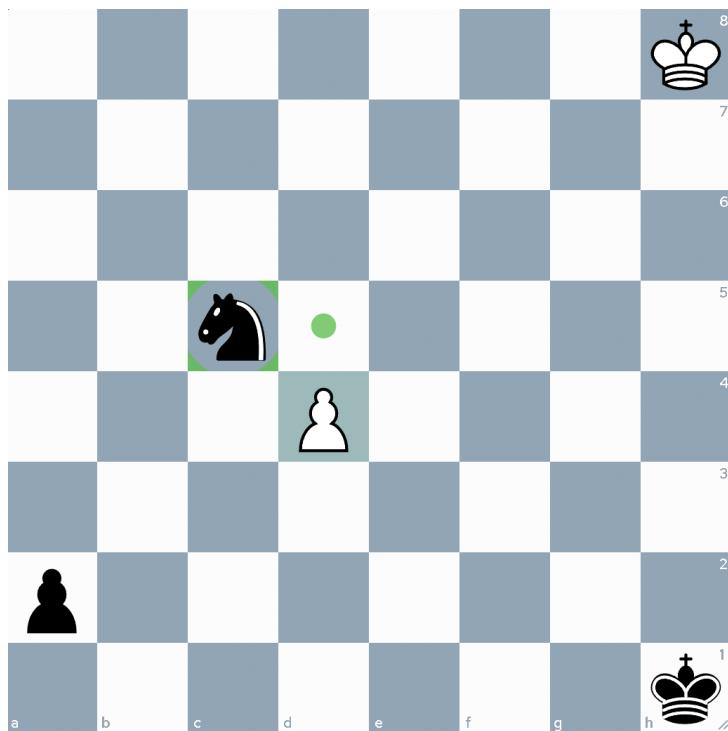


רץ: הרץ יכול לנوع מסוים בלתי מוגבל של משבצות באלאנסונים.



רגלי: הרגלי יכול לנوع רק קדימה. הרגלי יכול לנوع צעד אחד ישר קדימה, או להכות כל אחד באלאנסון קדימה (בניגוד לשאר כל המשחק שמכים בכיוון תנועתם). במסע הראשון של כל רגלי ניתנת לו הזכות (אך לא החובה) לצעוד שני צדים קדימה. כאשר מגיע הרגלי לשורה

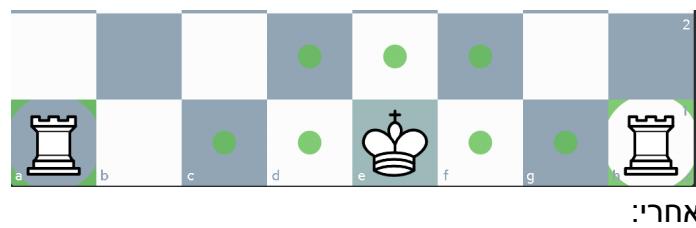
האחרונה של הלוח, מתרחש מצב הקריי "הכתרה". הרגלי הופר לכל אחד, על פי בחירת השחקן, באותו הצבע. ניתן להפוך את הרגלי לכל צל פרט למולך, אך לא ניתן לבחרו להשאיו רגלי. מהלך מיוחד של הרגלי הוא הכאה "דרך היילוכו": כאשר רגלי של היריב מתקדם שני צעדים בתור אחד, ובדרך עובר דרך משובצת המאפשרת על ידי אותו היריב רגלי יכול להכות את הרגלי של היריב כאילו זה התקדם רק צעד אחד. הכאה זו תקפה רק בתור שאחרי המהלך.

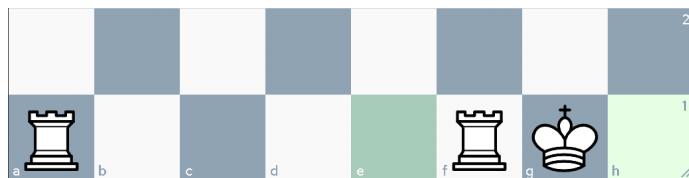


#### תנועות מיוחדות ההצרכה

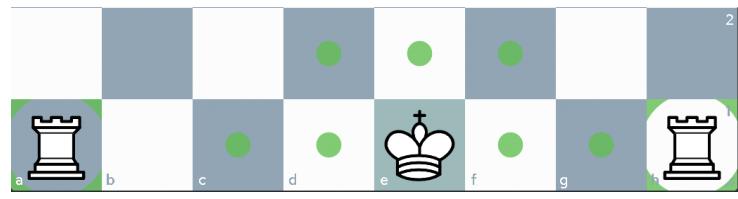
הצרכה מתבצעת כאשר מלך שודד לא זו במהלך המשחק זו שני צעדים לכיוון אחד מהצrichtים באותו צבע בתור 1' או טור 8' אשר אף הוא לא זו במהלך המשחק, והצריך מdag לכיוון המלך ונוחת ערוגה אחת אחרי המלך. הצרכה שבה משתמש הצריח הקרוב למלך נקראת הצרכה קטנה. הצרכה שבה משתמש הצריח השני מגף המלכה נקראת הצרכה גדולה.

#### הצרכה קטנה לפni:

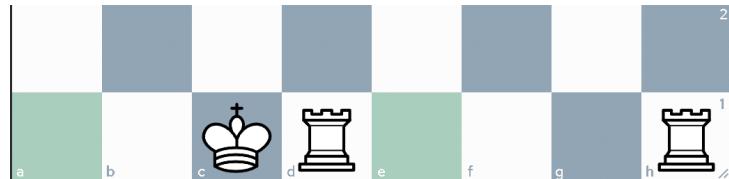




הצרכה גדולה:  
לפניכם:



אחריכם:



#### המצבים בהם זכות הצרכה מבוטלת:

- אם כלי האויב שולטים באחת העורוגות אותן המלך צריך לחצות כדי להגיע לעמדת הצרכה.
- כאשר המלך נמצא בשח, אם הוא עובר דרך משבצת מאוימת או כਮון מגיע למשבצת מאוימת.

שח

מצב בו המלך של הצד שתורו לשחק מאוים ככלומר אילו היה תור היריב הוא יכול ללקחת את המלך עם אחד מכליו.

#### הגדרת הפסד/ניצחון/תיקו

מטרת המשחק היא לנצח, וניצחון במשחק מושג כאשר אחד המלכים בשח על ידי אחד מכלוי היריב, אין יכול להקטן את הכלים המאימים, לחסום את האיים על ידי כלי אחר או להימלט מהאים למשבצת בה לא יהיה מאויים.

#### הכרעת משחק

הכרעת משחק תקרה כאשר אחד מהדברים הבאים קוראים:

- שחמט: מצב של שח בו אין לצד שתורו לשחק אפשרות לבצע מסע חוקי (כי אין לו מסע חוקי שמנוע מהיריב ללקחת את המלך), ואז המשחק הסתיים בהפסד של הצד שתורו לשחק.
- כנעה: מצב שבו שחקן נכנע. ואז המשחק הסתיים בהפסד של הצד של השחקן שנכנע.
- נגמר הזמן: מצב שבו נגמר הזמן לשחקן שתורו לשחק, וליריבו יש קומבינציית כלים שיכולים להנחתת מט.

#### פט (תיקו)

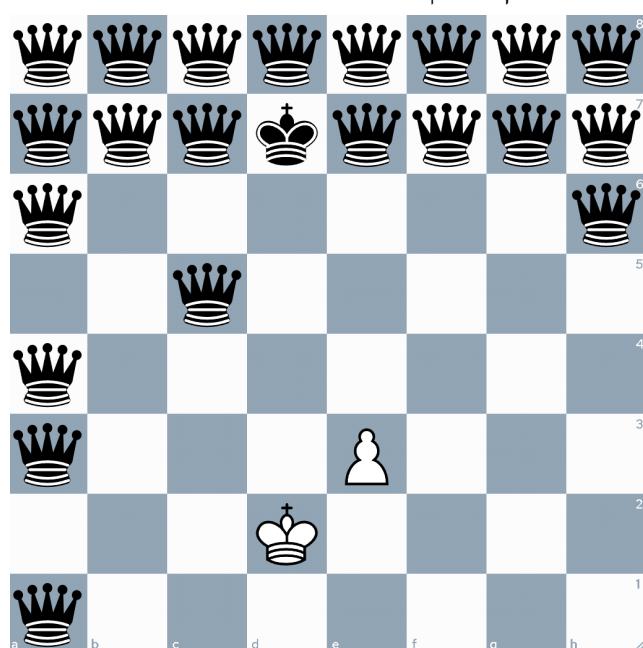
פט יקרה כאשר אחד מהדברים הבאים קוראים:

- תיקו בהסכמה (שחקן מציע תיקו ויריבו מסכים להצעה).

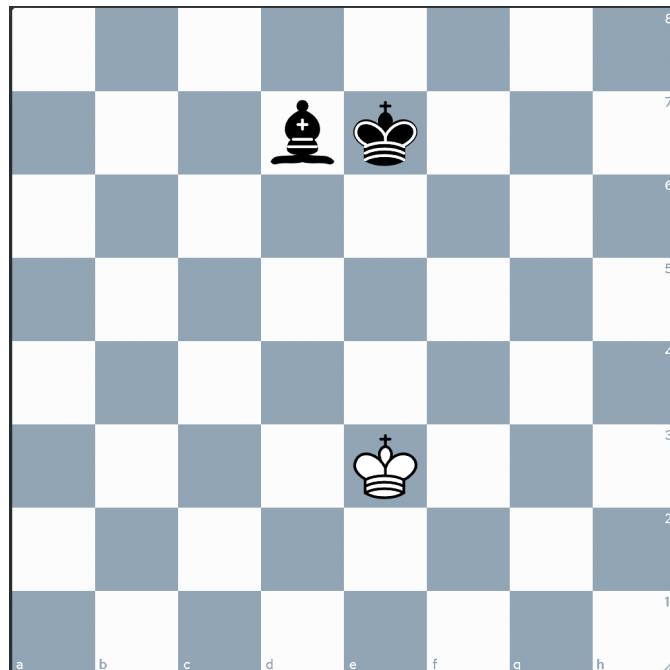
- כאשר אין אפשרות לבצע מהלך חוקי אבל המלך של הצד שטורו לשחק לא מואים בשח.
- כאשר אין לשני הצדדים שום דרך חוקית לסתת מט גם אם היריב טועה(למשל מלך מול מלך או מלך ופרש מול מלך).
- חזרה משולשת: כאשר העמدة חוזרת על עצמה שלוש פעמים או יותר עדת לחזור על עצמה 3 פעמים(המנוח עמדה אומר הצד שטורו לשחק זהה, וشعמדות הכלים זהות, וההאפשרויות העתידיות זהות).
- כאשר 50 המשומות האחרוניות של שני הצדדים בוצעו ללא הכהה או הצעת רגל או שזה עומד לקרוות במסע הבא בדומה לחזרה משולשת.
- כאשר נגמר הזמן לשחקן שטורו לשחק אך ליריבו אין קומבינציית כלים שמסוגלים להנחתת שחמט.

כמה דוגמאות לסופי משחק:

תورو של הלבן לשחק

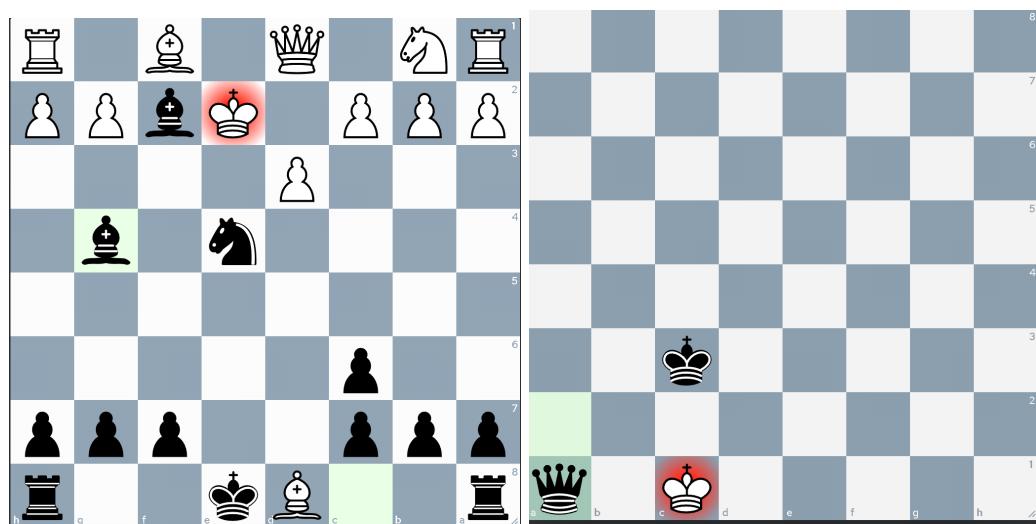


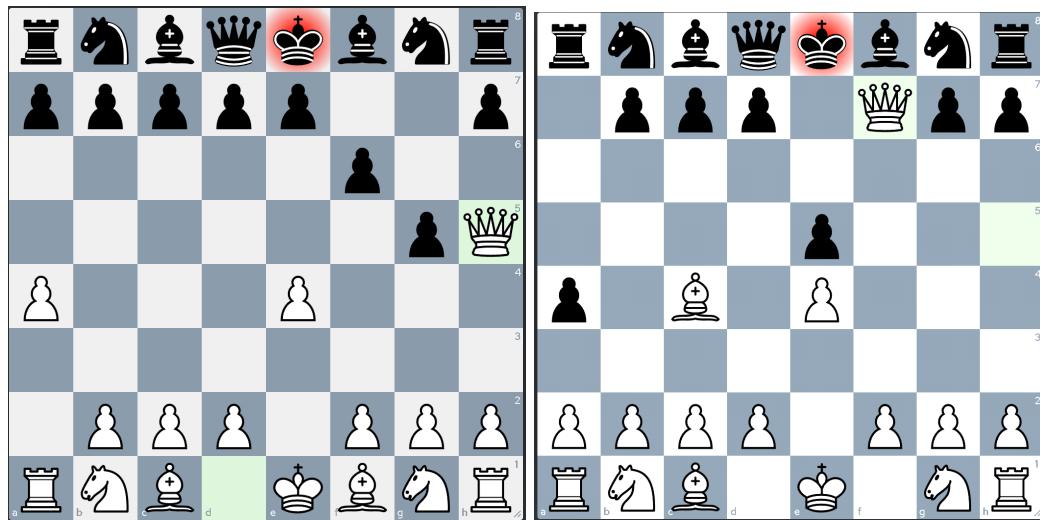
פט. לאחר מכן לשחקן הלבן מהלכים חוקיים.



פט. מאחר ולאף אחד מהשחקנים אין קומבינציית כלים שמסוגלים להנחתת מט.

כמה דוגמאות למט-ים:





### 3.2 דרישות טכניות

קוד התוכנה הורץ ונבדק על מחשבים עם המאפיינים הבאים, שהם גם הדרישות הטכניות להרצת קבצי הפROYיקט שעליהם :

- ✓ מחשב עם מערכת ההפעלה Windows 10 ו זיכרון ראשי עם לפחות 1GB.
- ✓ התקנת Environment Java Runtime בגרסה 16.0.2 ומעלה.
- ✓ קישוריות לרשות ופורטים פתוחים.
- ✓ עכבר + מקלדת.

### 3.3 הרצת המשחק והנחיות שימוש

בתיקיית הפROYיקט המצורף יש 3 תת-תיקיות:

- ▼ [תיקייה הגשה](#) [Bezalel Avrahami 325004752 Chess Network Game](#)
  - > [תת תיקייה לספר הפROYיקט](#)
  - > [תת תיקייה לקבצי ההרצה של הפROYיקט](#)
  - > [תת תיקייה לקוד המקור של הפROYיקט](#)

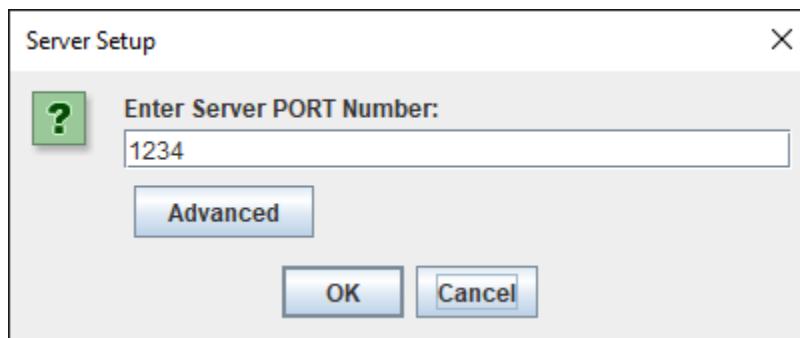
יש להיכנס לתת-התיקייה **Run Files** שבתוכה הקבצים הבאים:



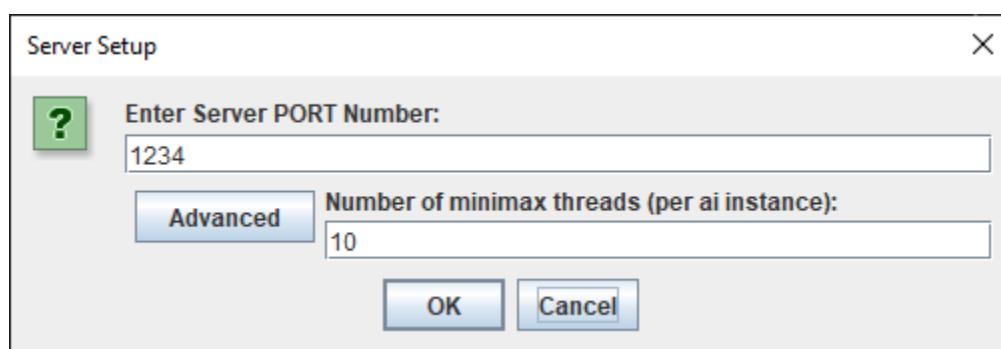
הרכבת המשחק מצריכה הריצה של שני מודולים נפרדים – מתחילה בהפעלת **השרת ע"י** הרצת הקובץ **ChessServer.jar**, שמיד ממתין להיבור לקובוחות (שחקני-רשת אונשים). הריצה זו חד-פעמית. כעת מפעילים את הלקוח ע"י הרצת הקובץ **ChessServer.jar** שמייצג שחקן-רשת יחיד. ניתן להריץ מספר לקובוחות כדי ליצור מספר שחקנים. עבור כל שני שחקנים, השרת "משדר" ויוצר עבורם משחק. בסעיפים הבאים נסביר בצורה מפורטת, באמצעות צלומי מסך, איך להריץ ולהשתמש בשרת ובלוקות, איך להתחיל משחק בין שני שחקנים וכמוהו איך לשחק.

### 3.2.1 הרצת השרת והשימוש בו

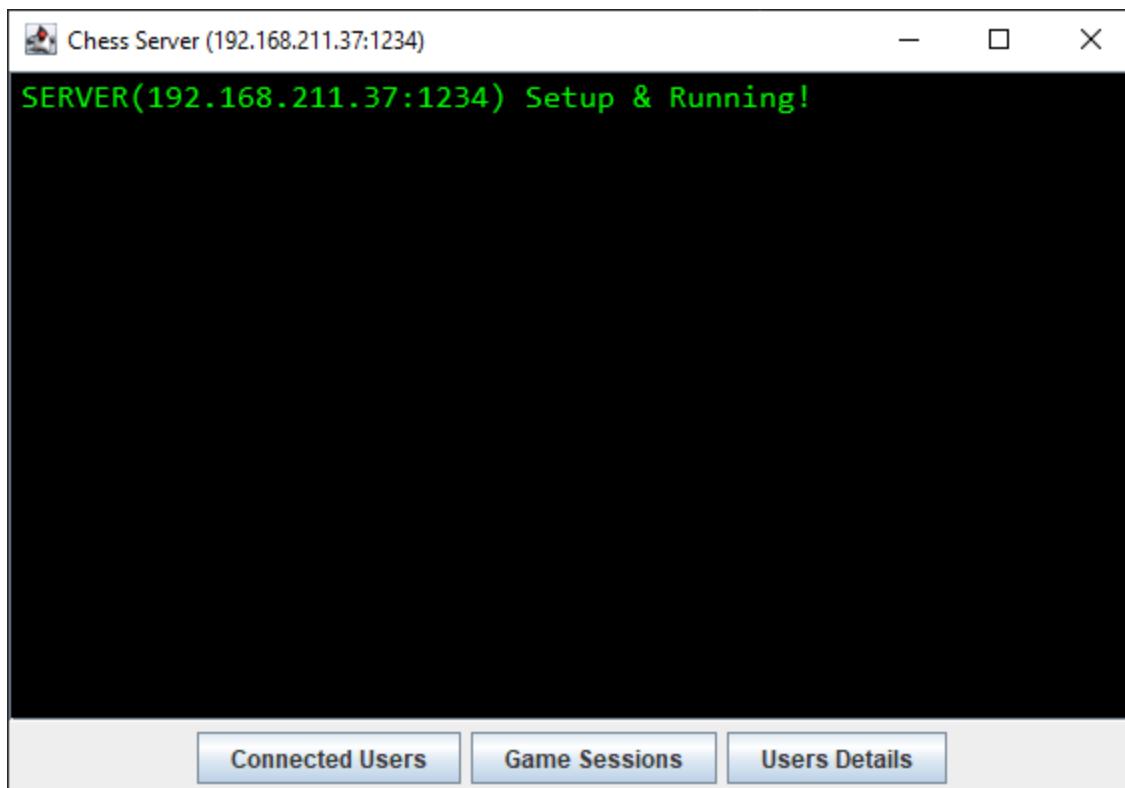
בהרצת השרת קופץ חלון אתחול לשרת, שבו תתקבקש לבחור פורט עלייו יירוץ השירות.



הפורט צריך להיות לא בשימוש. אם יוכנס פורט שנמצא בשימוש, י קופץ חלון שגיאה. בלחיצה על Advanced תוכל לבחור את מספר התהילכים שהמיןימקס ייצור עבור כל מופע של הבינה המלאכותית.



אחרי שהשרת אוחל בהצלחה, השרת מוכן לקבלת ליקווית בפורט שצוין.



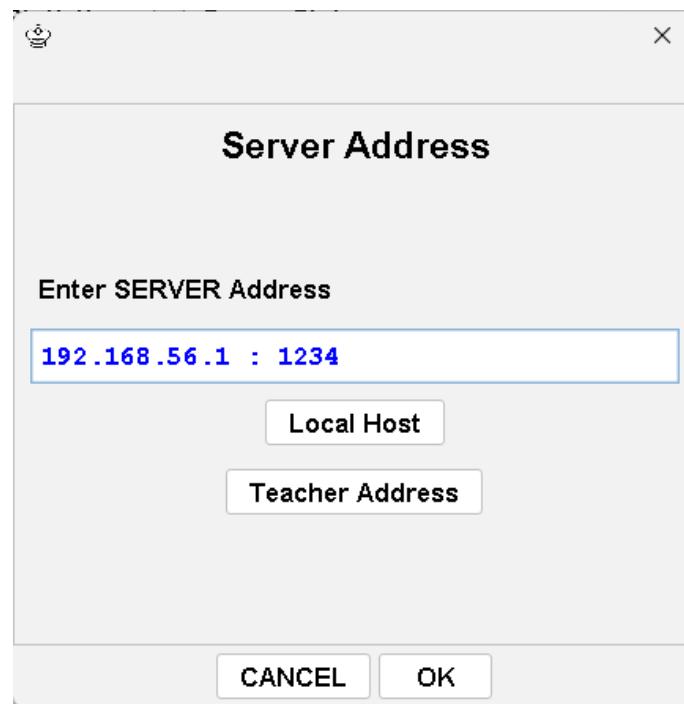
בלחיצה על הכפתור "Connected Users" תוצג רשימה של כל המשתמשים המתחברים.

בלחיצה על הכפתור "Game Sessions" תוצג רשימה של כל המשחקים שרצים בשרת כרגע.

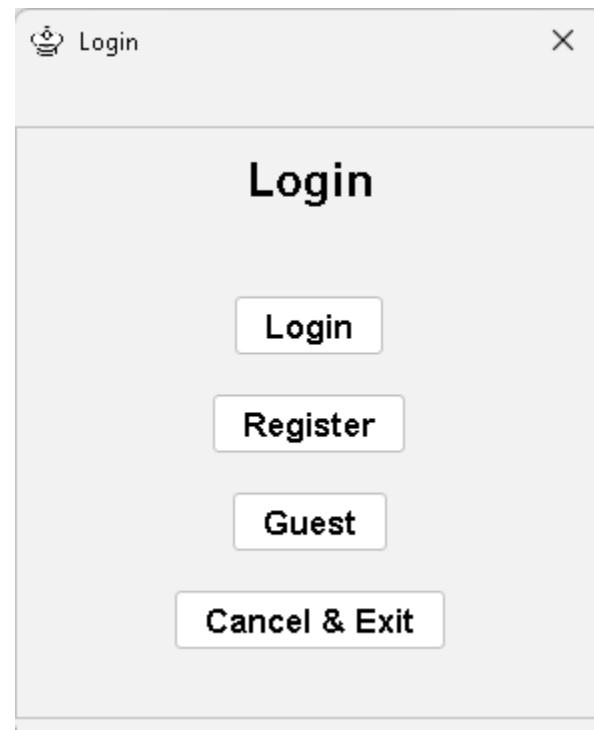
בלחיצה על הכפתור "Users Details" תוצג רשימה של כל שמות המשתמשים והסיסמאות של המשתמשים הרשומים.

### 3.2.2 הרצת הלקוּח והשימוש בו

בעת הרצת הלקוּח יוצג חלון להנחת כתובת השירות. הכתובת תהיה בפורמט: IP:PORT ישנים גם ערכיים דיפולטיים: localhost, לקרה שבו השירות מורץ על המחשב הנוכחי על פорт 1234, ושם teacher address שזו הכתובת של המחשב של הבוחן.



לאחר חיבור מוצלח לשרת בכתב שhwונת, יוצג חלון Login המאפשר חיבור כשחקן אורח או רשום או הרשמה או ביטול ויציאה.



בלחיצה על "Cancel & Exit", הילוקה ייסגר.  
בלחיצה על "Register", יוצג החלון הבא:

The screenshot shows a registration form titled "Register". It includes fields for "Username", "Password", and "Confirm Password". Each field has a red validation message below it. The "Username" field also features a placeholder "---" and an "eye" icon for password visibility.

Field	Validation Message
Username	5-10 Characters A-z 0-9 _.- Cannot Contain [Guest, User]
Password	5-10 Characters A-z 0-9 _.- Cannot Contain [Password]
Confirm Password	5-10 Characters A-z 0-9 _.-

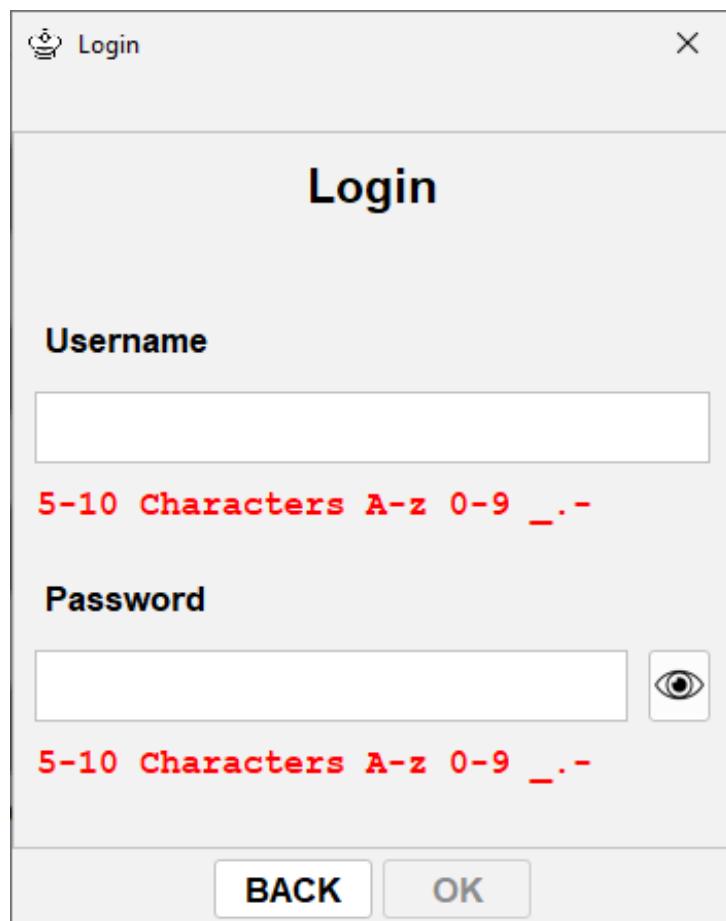
At the bottom are "BACK" and "OK" buttons.

ב"Username" צריך לבחור שם משתמש שעונה על הדרישות שכתובות (5-10 תווים..). לאחר שהוכנס שם משתמש חוקי, השתת יודא שם המשתמש לא קיים כבר בסיס הנתונים. במידה והוא כן, תוצג שגיאה.

ב"Password" צריך לבחור סיסמה שעונה על הדרישות (5-10 תווים...).

ב"Confirm Password" צריך לאמת את הסיסמה שהוקלדה.

בלחיצה על "Login", יוצג החלון הבא:



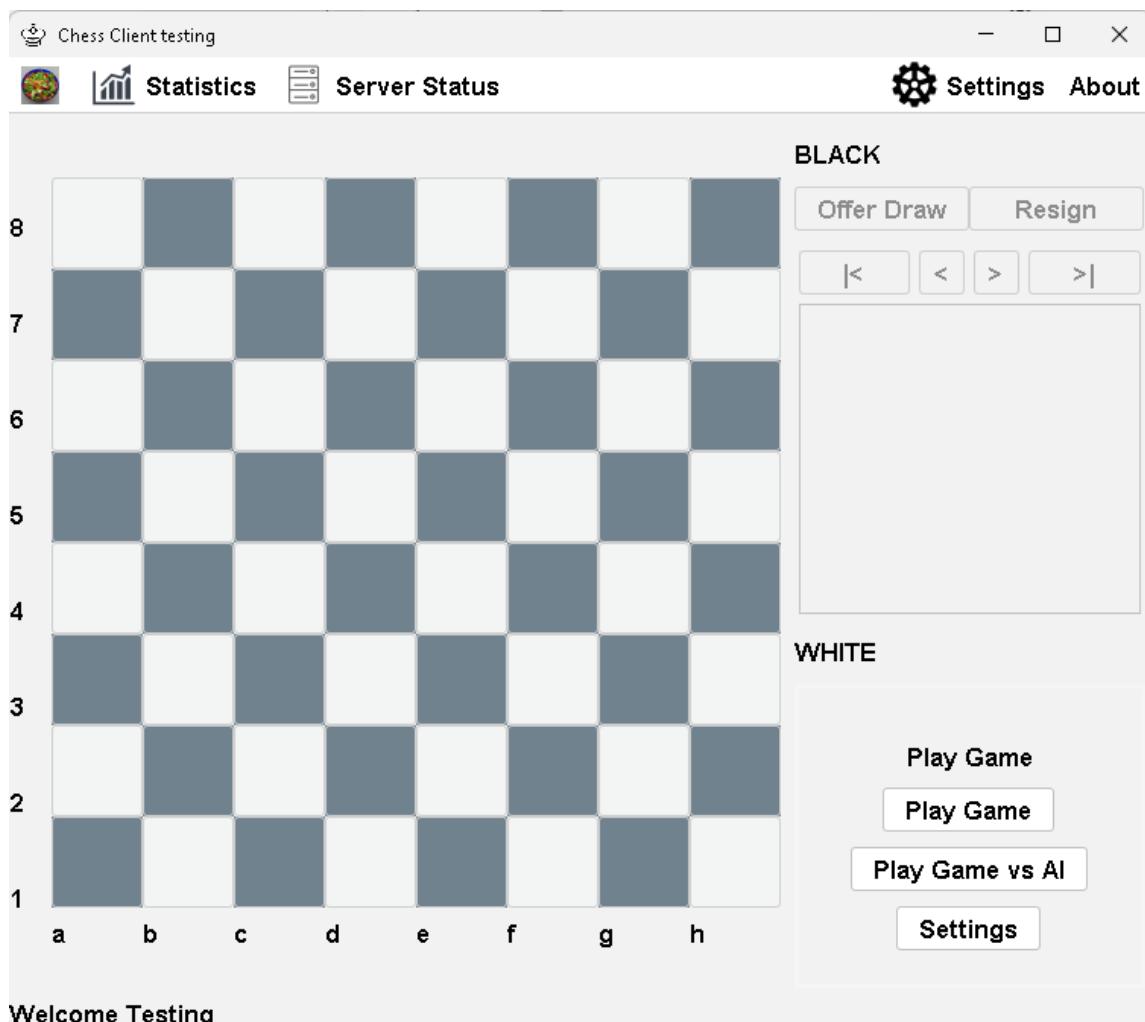
ב"Username" צריך להכניס את שם המשתמש.

ב"Password" צריך להכניס את הסיסמה.

לצורך הבדיקה ניתן להכנס עם שם המשתמש "testing" והסיסמה "123456". לחיצה על קונטROL+שיפט F+T תכניס את הפרטים האלה אוטומטית.

### 3.2.3 הפעלת המשחק וממשק משתמש גרפי (GUI)

לאחר כניסה של משתמש יוצג החלון הבא:



#### רכיבים גרפיים

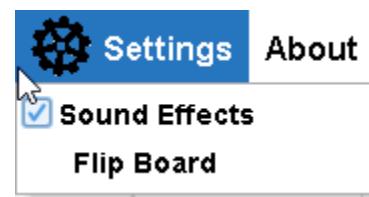
בתחתית החלון בצד שמאל נמצאת תווית הסטטוס. בה מוצג מידע על המצב הנוכחי. בצד ימין של החלון נמצאים התיעוד של המשחק, פעולות בקשר למשחק, והטיימרים של המשחק. ארכיב עליהם בהמשך.

#### התפריט העליון

בלחיצה על About יוצג:

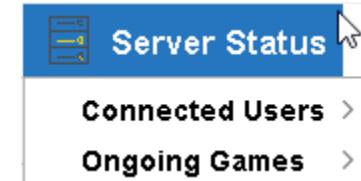


בלחיצה על Credits יפתח את חלונית הקרדיטים.  
בלחיצה על Rules ייפתחקובץ PDF של הוראות המשחק שחמט.  
בלחיצה על Settings :



ניתן לשולוט במצב האפקטים הקוליים. כשהחלהונית Sound Effects מסומנת בV האפקטים דלוקים וההפק. בלחיצה על הלוח יתאפשר לנקודות מבטו של השחקן השני.

בלחיצה על Server Status יוצג:



לחיצה על Connected Users תציג רשימה של כל השחקנים שמחוברים לשרת כרגע. לחיצה על Ongoing Games תציג רשימה של כל המשחקים שהושחקים כרגע בשרת.

### סטטיסטיקות

בלחיצה על Statistics יוצג:



עבור שחון שמחובר בתור אורה, Games In Range ו Stats By Time Of Day לא יהיה לחיצ. לאחר ואלו סטטיסטיקות שנבנו באמצעות משחקים שמורים בסיס הנתונים. ולאורה אין משחקים שמורים.

### Top Players

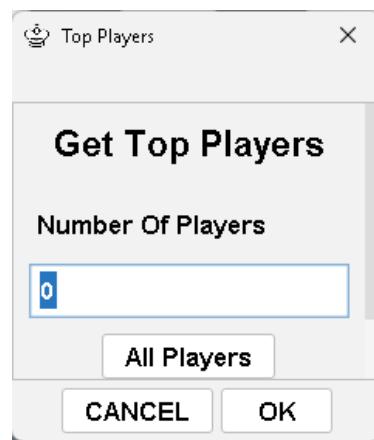
מטרת הסטטיסטיקה היא להציג רשימה של מספר מבוקש של השחקנים הטוביים ביותר וסטטיסטיקות עבור כל אחד. ממויננים מהטוב ביותר ומטה. היחס ככמה טוב כל שחון מתבצע בצורה זו:

$$\frac{ties*0.5+wins}{games}$$

לחיצה על Top Players תציג:



לחיצה על Top Players תציג:



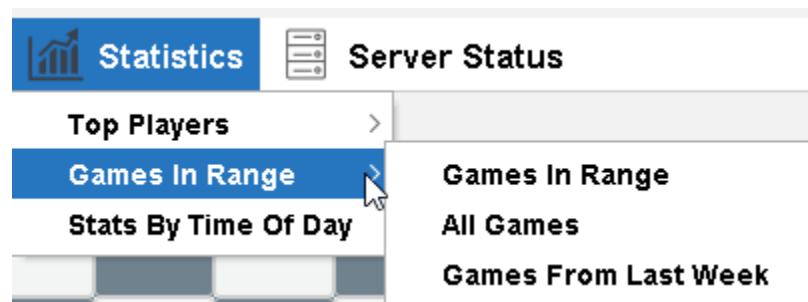
הערך הדיפולטיי, 0, יציג את כל השחקנים השמורים.  
Top All Players יציג את כל השחקנים השמורים.  
Top Five Players יציג את 5 השחקנים הטוביים ביותר.  
לדוגמה: Top All Players

Top All Players		
Top Players		
Username	Win-loss-tie Ratio	Num Of Games Played
bezalel6	0.500	4
testing	0.375	8
bezalel0	0.000	0
bezalel1	0.000	0
bezalel_6_	0.000	0

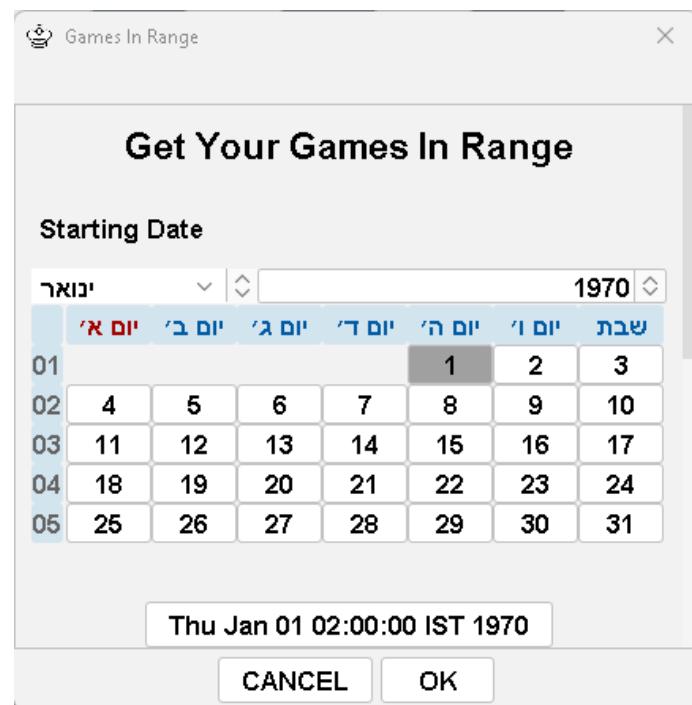
Total Games
8

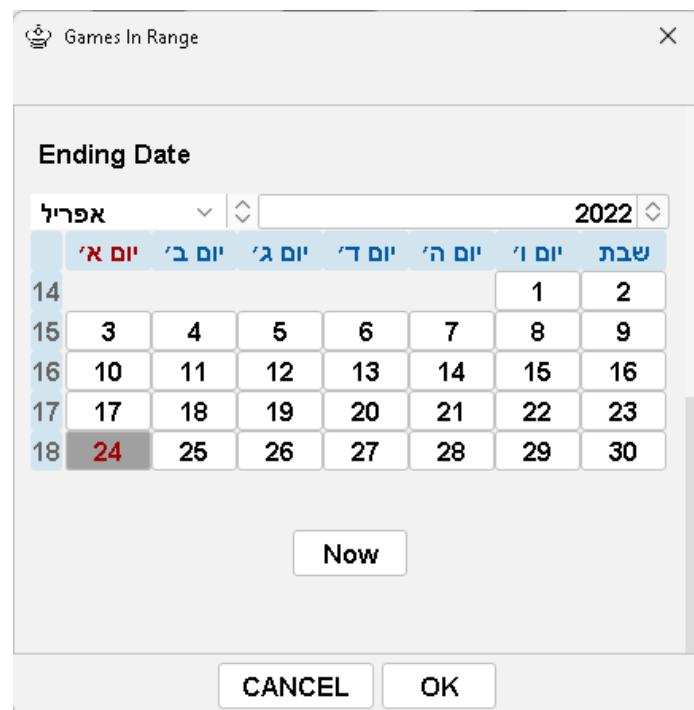
בטבלה העליונה ניתן לראות את רשימת המשתמשים והסטטיסティקיות שלהם. בטבלה התחתונה ניתן לראות את הסכום הכללי של השחקנים השמורים במסד הנתונים.  
[Games In Range](#)



מטרת הסטטיסטיקה היא לקבל את פרטי המשחקים השמורים של השחקן וסטטיסטיקות עבותם בתחום טווח תאריכים מסוים.

לחיצה על Games In Range יציג את החלון הבא שבו תتمكن להכניס טווח תאריכים שבמהלכו שוחקו המשחקים:





לאחר בחירה של טווח ולחיצה על OK יוצגו סטטיסטיות עבור המשחקים בטווח שנבחר.  
לחיצה על All Games: סטטיסטיות עבור כל המשחקים שנשמרו.  
לחיצה על Games From Last Week: סטטיסטיות עבור כל המשחקים ששוחקו בשבוע האחרון.

לדוגמא: All Games

All Games		
All Games For Testing In Selected Range		
Opponent	Winner	Created Date Time
MyAi	MyAi	2022-04-24 03:17:26.000000
GUEST#1	----tie----	2022-04-24 02:49:35.000000
GUEST#1	testing	2022-04-24 02:49:26.000000
GUEST#0	----tie----	2022-04-24 02:27:53.000000
GUEST#0	GUEST#0	2022-04-24 02:12:09.000000
GUEST#0	----tie----	2022-04-24 02:11:53.000000
MyAi	----tie----	2022-04-23 23:59:05.000000
GUEST#1	----tie----	2022-04-23 23:22:14.000000
GUEST#12	GUEST#12	2022-04-23 21:44:29.000000
bezalel6	bezalel6	2022-04-23 04:29:10.000000
MyAi	MyAi	2022-04-20 18:01:23.000000
MyAi	testing	2022-04-20 17:52:02.000000
MyAi	testing	2022-04-17 09:21:40.000000

Games Stats					
Total Games Played	Win-loss-tie Ratio	Wins	Losses	Ties	
19	0.394	5	9	5	
OK					

### Stats By Time Of Day



מטרת הסטטיסטיקה היא להציג לשחקן את הסטטיסטיות של המשחקים שלו לפי השעה ביום.  
לדוגמא:

Stats By Time Of Day | SUCCESS

### Stats By Time Of Day

23:00 - 24:00

Total Games Played	Win-loss-tie Ratio	Wins	Losses	Ties	
2	0.500	0	0	2	

21:00 - 22:00

Total Games Played	Win-loss-tie Ratio	Wins	Losses	Ties	
1	0.000	0	1	0	

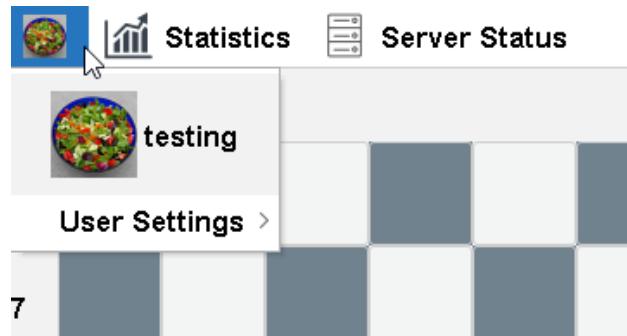
19:00 - 20:00

Total Games Played	Win-loss-tie Ratio	Wins	Losses	Ties	
0	0.000	0	0	0	

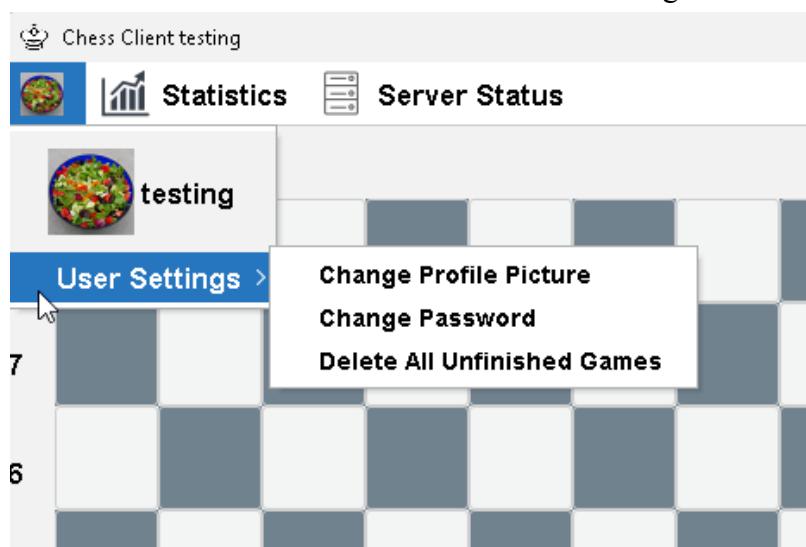
17:00 - 18:00

OK

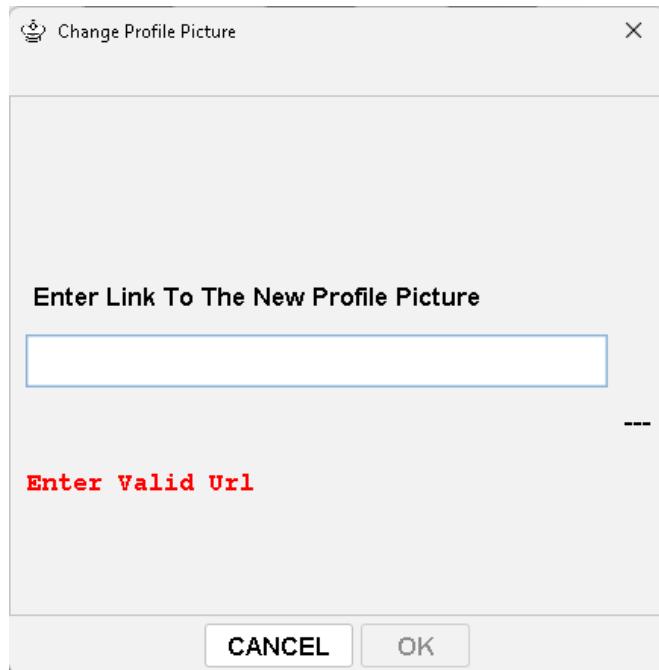
### הגדרות משתמש



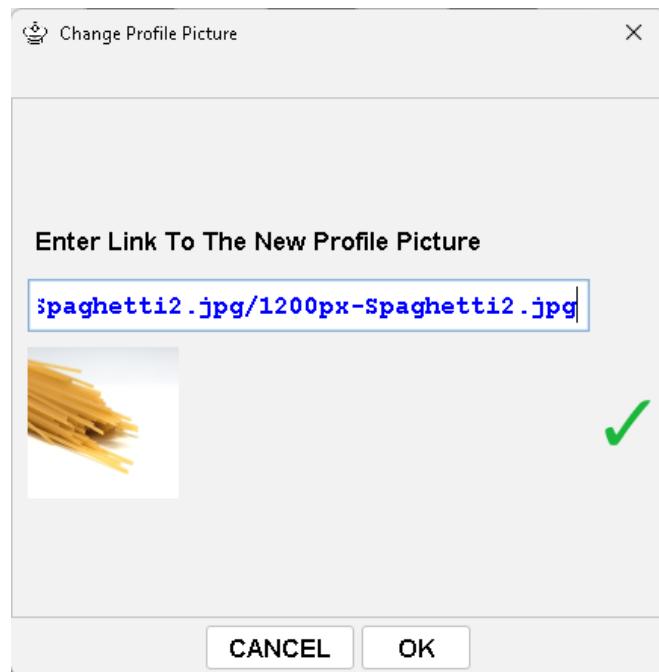
בעת לחיצה על תמונה ה פרופיל בצד שמאל לעלה, יוצג החלון עם שם המשתמש ותמונה הפרופיל של המשתמש (או תמונה אונומית במקורה של אורח או משתמש שלא הגדר את תמונה הפרופיל שלו).  
הכטור User Settings לא יהיה זמין עבור אורח.  
בלחיצה על משפטן User Settings של משתמש רשום:



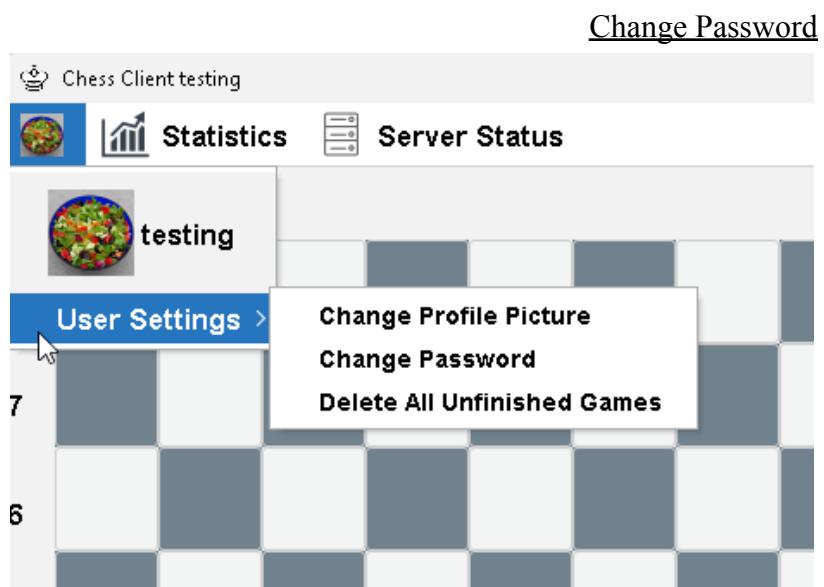
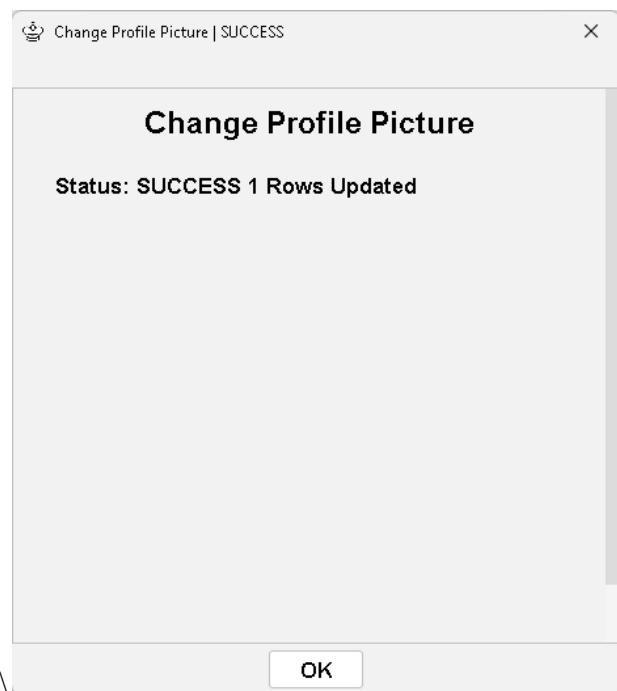
בלחיצה על Change Profile Picture יוצג החלון הבא:



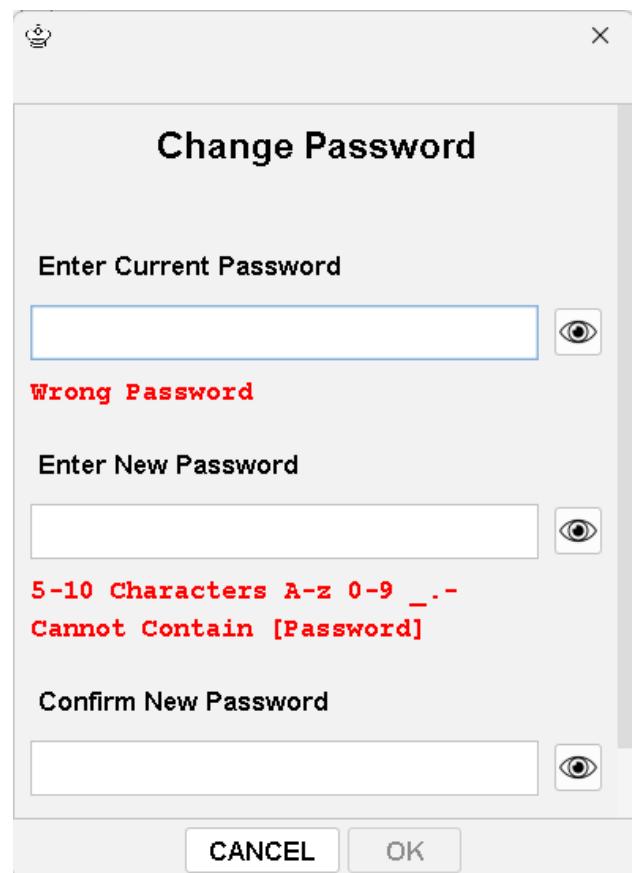
לאחר הכנסה של כתובת חוקית לתמונה:



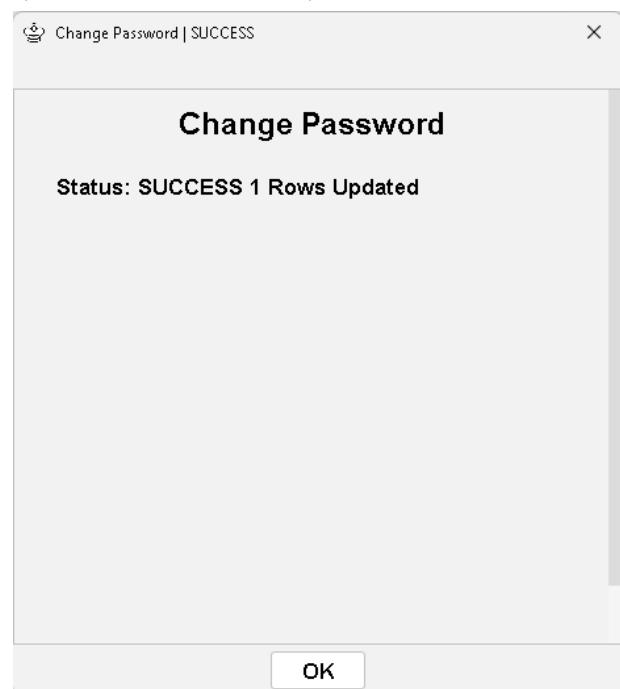
(התמונה תהיה התמונה שנמצאת בכתובת שהוכנסה)  
לאחר לחיצה על OK הכתובת החדשה תשלוח לשרת. לאחר שהשרת יעדכן את התמונה מסדר הנתונים  
בהצלחה, תוצג הודעה:

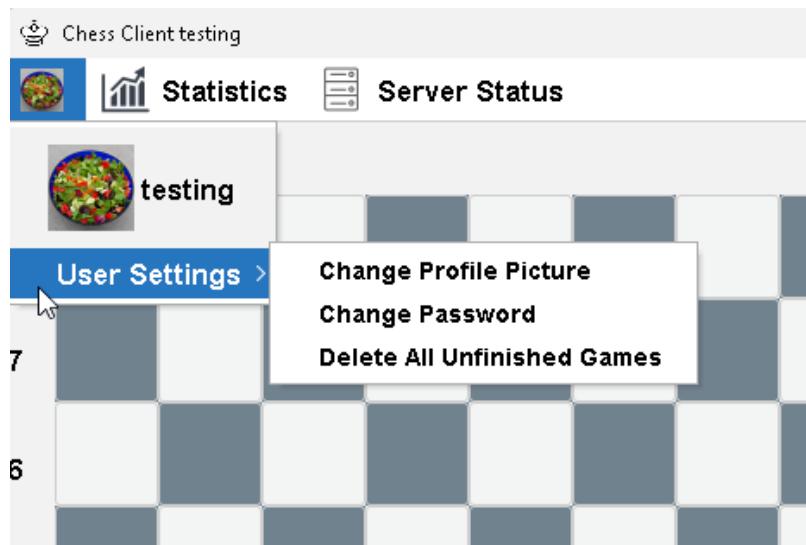


לאחר לחיצה על :Change Password



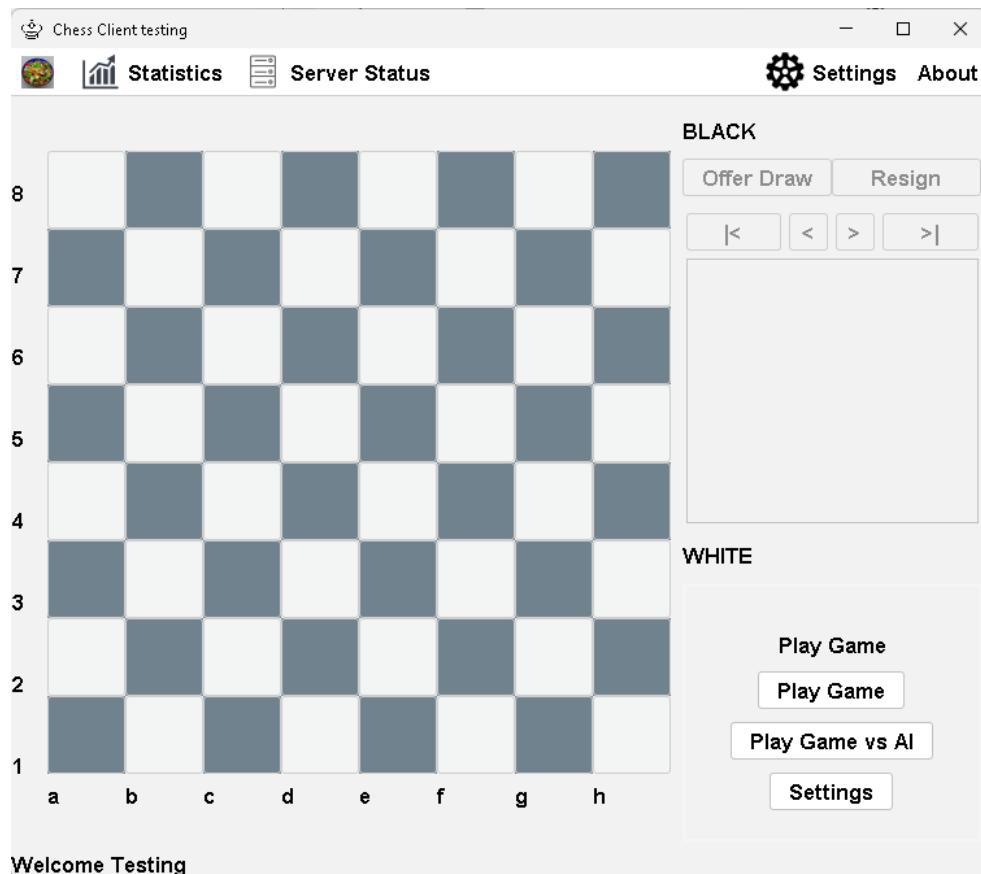
לאחר מילוי הסיסמה הנוכחית, הסיסמא החדש והאימות, ולהזיכת על OK:



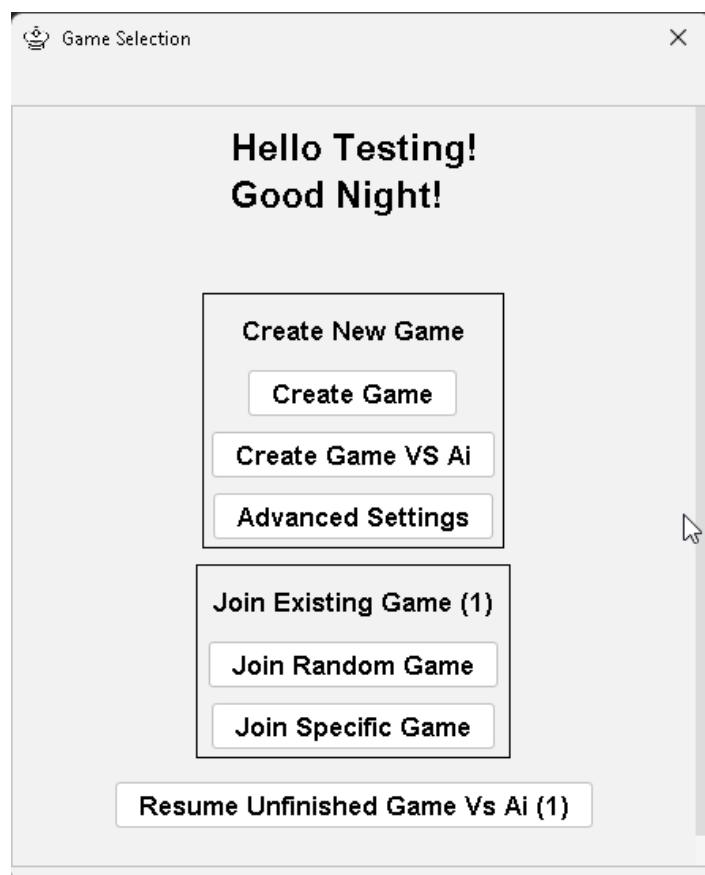


לאחר לחיצה על Delete All Unfinished Games, ימחקו כל המשחקים של השחקן שעדין לא נגמרו מול AI, ווונגן חלון שמציג את תגובת השרת לבקשת המהикаה.

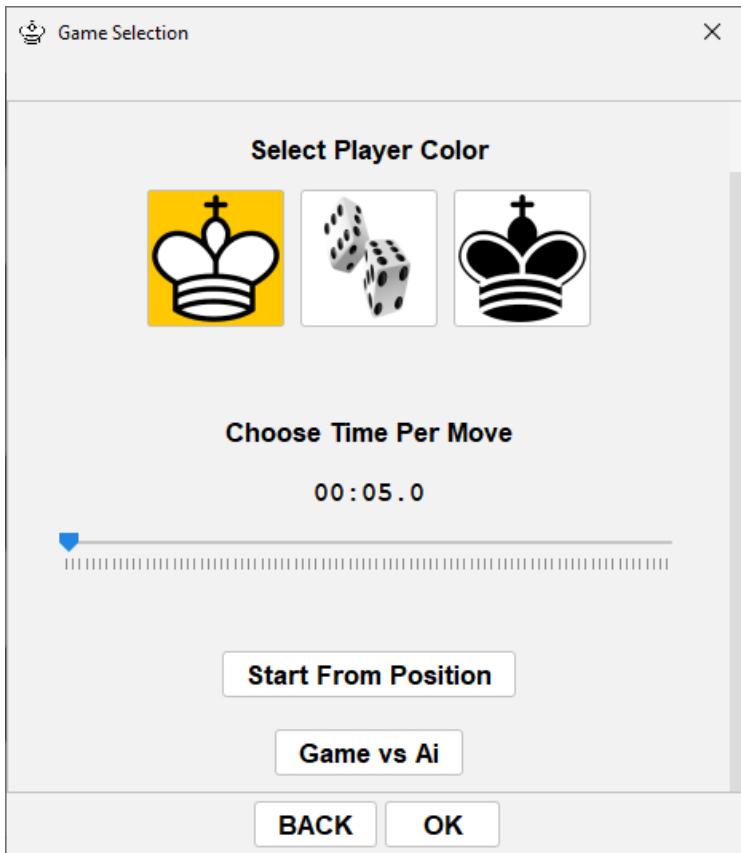
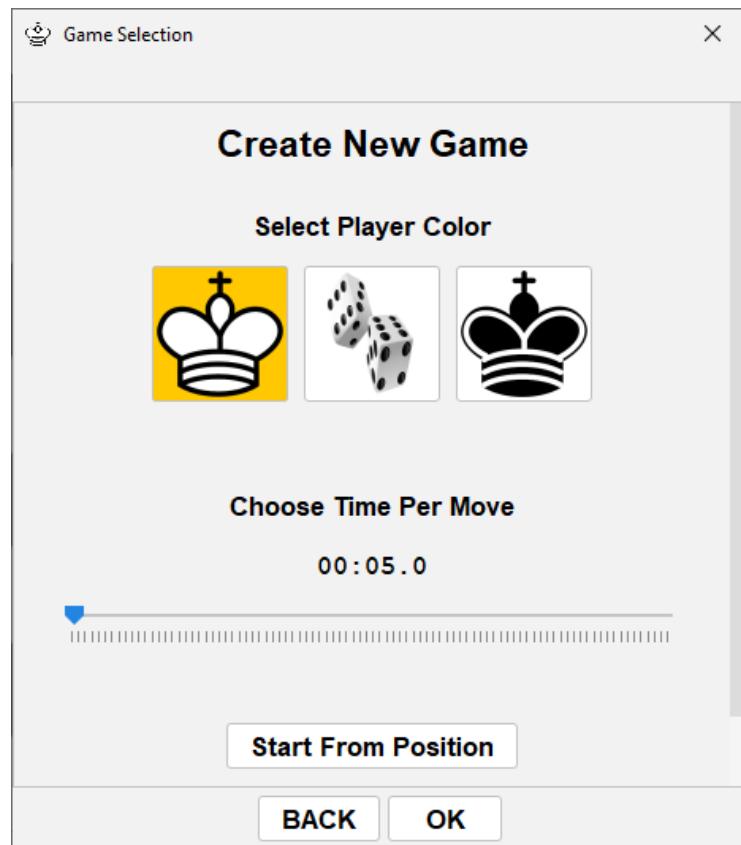
#### אתחול משחק



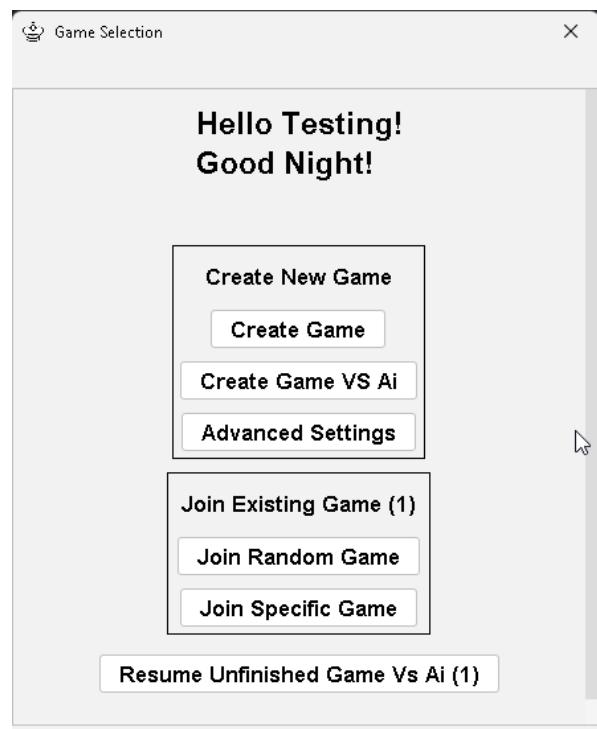
כדי להתחיל לשחק, אפשר לבחור בAI Play Game כדי להתחיל משחק מול הבינה המלאכותית מיד. או Play Game כדי לשחק עם שחפן רשות אחר. במידה וקיים משחק של שחפן אחר שממתין ליריב, המשחק יתחל מיד. אחרת, יש להמתין עד שחפן אחר ייצור משחק ואז יתחל המשחק. בלחיצה על "Settings" יפתח הדיאלוג הבא:



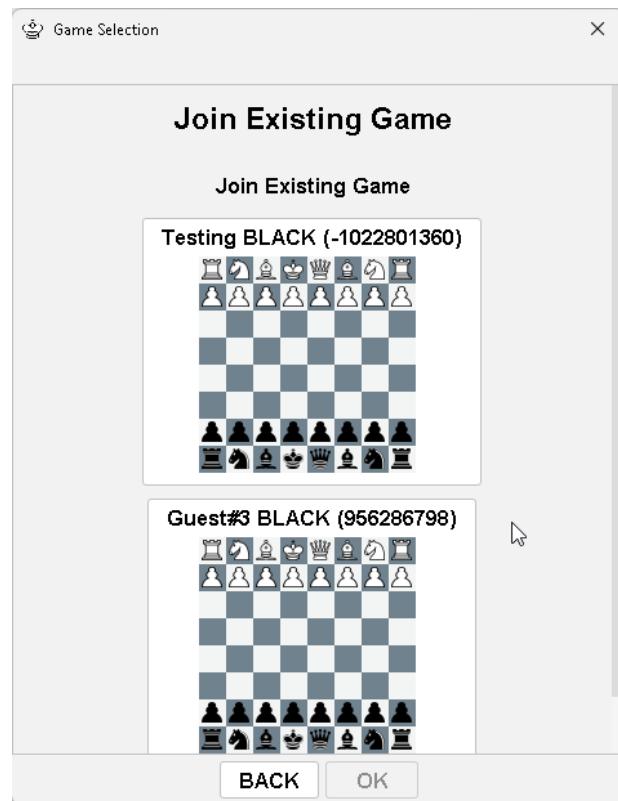
ייצור משחק סטנדרטי וימתין לשחקן אחר שיתחבר למשחק שנוצר. Create Game  
ייצור משחק מול הבינה המלאכותית ויתחיל מיד. Create Game VS AI  
יציג את החלון הבא: Advanced Settings



ניתן לבוחר איזה צבע אתה תהייה (לבן, שחור, או רנדומלי), כמה זמן יש לכל מהלך בשניות, להתחילה מעמדת מסויימת, ולשחק מול AI.  
בלחיצה על OK המשחק יוצע.



ניתן לנסה להתחבר למשחק של שחקן אחר שממתין ליריב. בהתאם לסוגרים ניתן לראות את כמות המשחקים שיש בשירות שממתינים לשחקן שני. יציג את החלון הבא:

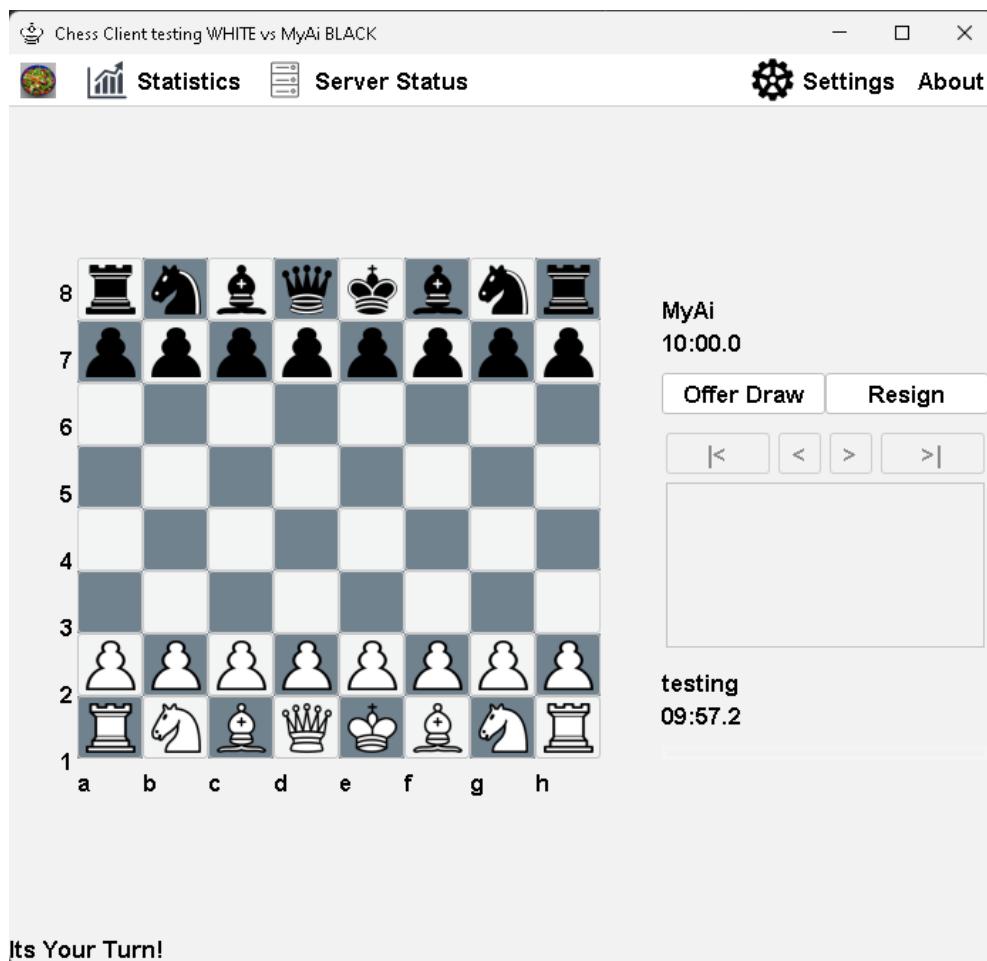


(רישימה תהיה של המחקים שזמינים בשרת בעת ההרצה) עבור כל משחק, ניתן לראות את שם המשתמש של היוצר של המשחק, והעמדה שמננה השחקן יתחיל לשחק. בבחירה של אחד המחקים ולחיצה על OK המשחק יתחל.

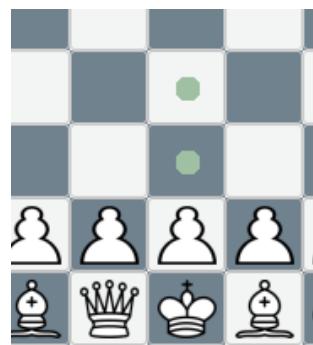
Resume Unfinished Game VS ai  
יציג רשימת משחקים שהופסקו מול הבינה המלאכותית. האופציה תהיה זמינה רק לשחקן שמור. לאחר ולאורחים לא נשמרים משחקים שהופסקו.

#### מהלך המשחק

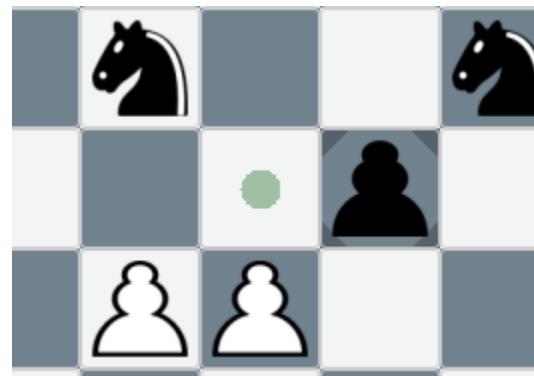
לאחר אתחול המשחק, ייטען הלוח עם העמדה שנבחרה(בכ"כ עמדת הפתיחה) והלוח יהיה מכובן בהתאם לצבע של שחקן הרשות. (צילומי המסך יהיו עבור השחקן הלבן מעמדת הפתיחה)



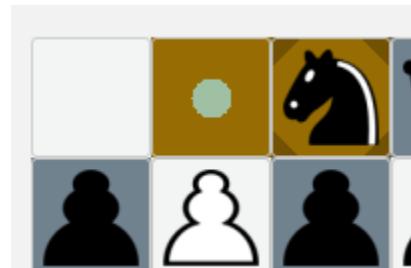
לפי תווית הסטטוס או הטימר שרצ ניתן לראות שתורנו, ולכן עליינו לבצע מהלך. בלחיצה על אחד מהכליים שלך, יסומנו כל המשבצות שאלייהם ניתן לכת באופן חוקי עם אותו כל. לדוגמה, בעת לחיצה על הרגלי בe2 בעמדת הנוכחתה:



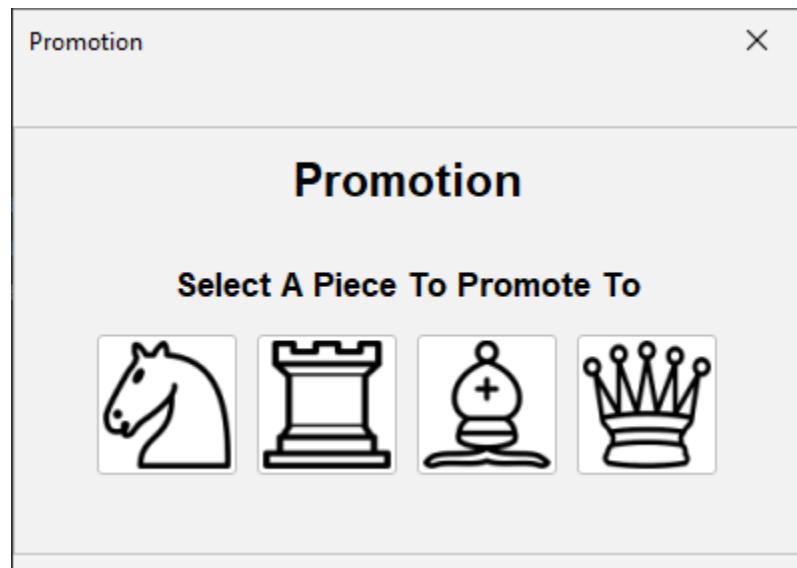
ניתן להחליף את הכלי שברצונך להזיז ע"י לחיצה שנייה על אותו הכלי (מה ש לבטל את הלחיצה הראשונה) או לחיצה על כלי אחר. לאחר שבחירה כל ויעד, הלוח יתעדכן ומהלך שלך נשלח לשרת. במקרה שבו נבחר כל שעבورو יש אפשרות אפשרית, היא מסומן בצורה זו:



במקרה שבו נבחר רגלי שבאפשרותו לעשות קידום, משבצות הקידום יסומנו بصورة זו:



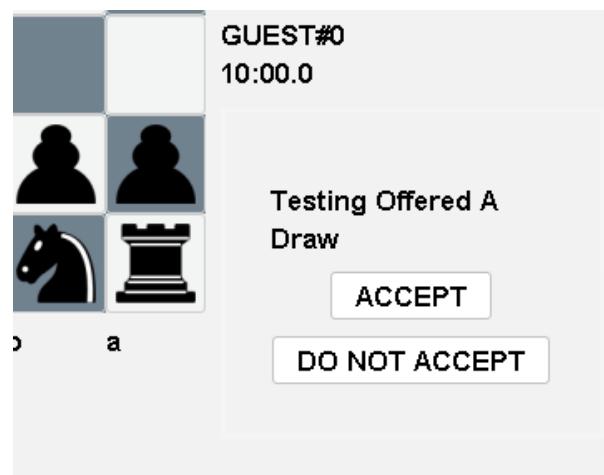
כששחקן מבצע מהלך קידום, תוצג החלונית:



שנה תתבקש לבחור סוג כלי לקדם אליו.

#### הצעת תיקו

כל עוד לא נגמר המשחק, כל שחקן יכול להציג תיקו לשחקן השני. לא משנה אם תורו. כדי להציג תיקוلوحצים על הכפתור Offer Draw. כל שחקן יכול להציג תיקו פעמי אחד בלבד במהלך כל משחק. לאחר שהיריב הציג תיקו, מוצגת הודעה بذلك ימין למטה. ההצעה תהיה זמינה למשך כל שאר המשחק והשחקן יוכל לקבל אותה מתי שירצה. אלא אם כן הוא מסרב לה בלחיצת על DO NOT ACCEPT. כדי לקבל את ההצעה, השחקן לוחץ על ACCEPT והמשחק יגמר מיד בתיקו.

כינעה

בכל שלב במהלך המשחק, לא משנה אם תורו, כל שחקן יכול ללחוץ על כפתור **Resign** כדי להכנע. לאחר לヒיצה על הכפתור, המשחק ייפסק מיד, והשחקן יהיה הפסד של השחקן הנכנע.

זמן

כשהטור שלך מתייחל השעון שלך מתייחל לרווז. כשהוא מגיע ל-10 השניות האחרונות הטימר הופך לאדום. אם נגמר לך הזמן לפני שביצעת מהלך, הפסדת.

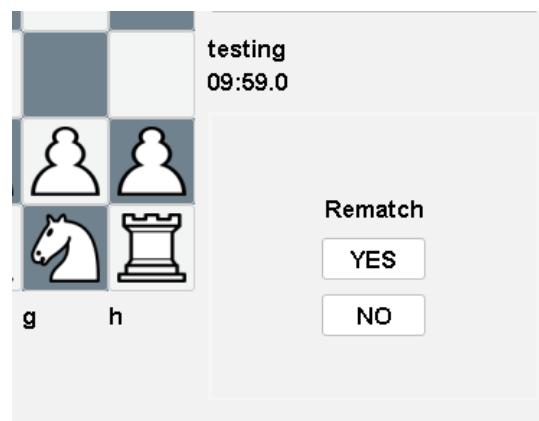
התנקות

במקרה של התנקות, המשחק נגמר מיד. אם המשחק היה בין 2 שחקנים אנושיים, המשחק מוכרע בתור הפסד של השחקן שהתנק. אם המשחק היה נגד AI, והשחקן האנושי היה מחובר בתור משתמש רשום, המשחק נשמר בתור משחק שלא נגמר, ומשתמש יכול לבחור להמשיך אותו מהנקודה שהופסקה בפעם הבאה שייתחבר.

סוף משחק

לפי התנאים של שמירת משחק. ז"א אם אחד מהשחקנים לפחות אינו AI ואינו אורח, המשחק ישמר במסד הנתונים.

בנוסף, ננעלים הכתופורים של הלוח, הטימרים נעצרים, ותוויות הסטטוס מתעדכנת בהתאם לפרטי סוף המשחק. אם אף אחד מהשחקנים לא התנק, הם יישאלו אם הם רוצים לשחק שוב אחד עם השני:



אם שניהם בוחרים YES, הם יתחילו לשחק עוד משחק. אם אחד מהם בוחר NO או מتنתק, השחקן חוזר למצב אתחול משחק.

## 4. מדריך לתוכנה

בחלק זה נסביר בצורה עמוקה את קוד התוכנה מנקודת מבטו של המתכנת. נתחיל בדרישות המערכת התוכנה, נמשיך עם פירוט מרכזי הפרויקט שבهم נעשה שימוש בכתיבת הקוד כגון: ארכיטקטורת שרת/לוקוח, פרוטוקול התקשרות TCP/IP, Sockets, Threads, תהליכונים ומסד הנתונים. כמו כן יפורטו המחלקות, מבני הנתונים ואלגוריתמים לפועלות נבחרות, בהם נעשה שימוש בפרויקט.

### 4.1 הדרישות ממיצוג התוכנה

- על המערכת להיות כתובה בשפת JAVA תוך שימוש בעקרונות תוכנות מונחה עצמים מתקדם
- על המערכת להשתמש בארכיטקטורת שרת לוקוח Client/Server
- על המערכת לבצע את התקשרות בין השרת לлокוח באמצעות Sockets עם פרוטוקול TCP/IP
- על המערכת לטפל בחיבור מסוים של לוקוחות ובמקביל (שימוש-ב-Threads)
- בעת חיבור לוקוח, על המערכת להציג מסך LOGIN שמאפשר לлокוח להתחבר משתמש רשום (עם שם משתמש וסיסמה) או כאורח או להירשם כמשתמש חדש. אם הוא מתחבר כמשתמש רשום, נתנוינו לבדוק מול מסד הנתונים. אם הוא נרשם כמשתמש חדש, שם המשתמש שהוא בחר יבדק מול מסד הנתונים, ולאחר מכן הוא בחר שם משתמש ייחודי, פרטייו יוכנסו למסד הנתונים.
- לאחר שלקוח המתחבר (כמשתמש רשום או כאורח), על המערכת לאפשר לו לבחור את סוג השחקן מולו הוא ירצה לשחק: שחקן אנושי (локוח אחר) או שחקן ממוחשב (רץ בשרת). אם בחר לשחק מול שחקן אנושי, הוא מקבל הודעה שעליו להמתין לחיבור לוקוח גוסף שייהיה בין הזוג שלו למשחק. במקרה זה המשחק לא מתחילה עד אשר מתחבר עוד לוקוח. אם בחר לשחק מול שחקן ממוחשב – המשחק מתחילה מיד.
- על המערכת לאפשר מספר לא מוגבל של משחקים. כל משחק הוא בין זוג שחקנים.
- כאשר משחק מסתיים, המערכת תודיע לשחקנים על סיום המשחק ותוצאתו (מי ניצח/פסיד או תיקו) ותאפשר התחלת משחק חדש ע"י הסכמה למשחק חוזר, או התחלת משחק חדש.
- אם משחק הוא בין שני שחקנים אנושיים (שהם שחקני רשות – לוקוחות) ושניהם ביצעו ההתabbrות כמשתמשים רשומים, אז תוצאות המשחקים שלהם ישמרו במסד הנתונים לצורך הפקת סטטיסטיות עבור כל שחקן.

- על המערכת לשמר בסיס נתונים לפחות 3 טבלאות: משתמשים רשומים (שם משתמש, סיסמה) משחקים (משתמש1, משתמש2, זמן המשחק-תאריך שעה, תוצאת המשחק) ועוד ...
- בכל שלב לקווי יכול להתנתק ע"י סגירת החלון שלו. על המערכת לאלהות זאת לבדוק: אם מדובר על לקווי שהוא באמצע משחק, אז יש להודיע לקווי השני (אם הוא אנושי) על סיום המשחק ולבצע סגירה והפסקת המשחק בצורה מסודרת (הקפצת הודעות מתאימות, סגירת סוקטים וכו').
- על המערכת להיות ניתוקים לא צפויים (כגון: נפילת לקווי/שרת) ולהקפיים הודעות מתאימות ולבצע סגירה מסודרת.

## 4.2 ארכיטקטורת המערכת

בסעיף זה נסביר על ארכיטקטורת המערכת המתארת את מרכיבי התוכנה שבהם נעשה שימוש לצורך כתיבת קוד הפרויקט. עבור כל מרכיב נסביר למה הוא משמש? ומה היה צריך אותו בימוש הפרויקט? למה בחרנו דוקא בו? ועוד.

### 4.2.1 מודל שרת-לקוֹו Client-Server Model

מודל שרת-לקוֹו הינו מודל תקשורת בין מחשבים ברשת האינטרנט המתבסס על שליחת בקשה ותגובה - הלקוח שולח בקשה Request והשרת מוחזיר תגובה Response. מודל זה מאפשר לקוֹו ללא צורך לביצוע הרבה חישובים, חוץ שליחת וקבלת הודעות. ורוב העומס מוטל על השרת. מה שמאפשר חיבור של מחשבים לא עוצמתיים והעברת אותה חוות משמש. וכך השתמשנו בו.

### 4.2.2 פרוטוקול תקשורת TCP/IP

פרוטוקול התקשרות TCP/IP הינו פרוטוקול תקשורת בטוח. ז"א שכל פריט מידע שנשלח, בטוח יגיע ליעדו. כל הודעה שנשלחת באמצעות הפרוטוקול עוברת חילוק לפאקטות, וכל אותן פאקטות נשלחות באופן נפרד לידי. כשהייד מקבל פאקטות, הוא מרכיב אותן מחדש להודעה המקורית, ומוכיח שכל הפאקטות הגיעו בשילמותן. ואם הן לא, הוא מבקש מהשולחה שישלים את החסר. מה שעולה בזמן יקר, אך מבטיחה את שלמות המידע. וכך השתמשתי בו. כי ההודעות שנשלחו בין הלקוח לשרת והשרת לקוֹו מכילות מידע קריטי שאסור שיאבוט. ובשימושים בפרויקט הזה, הזמן ש"מתבצע" על הפרוטוקול, זניח.

### 4.2.3 Sockets

SKU הינו עורך תקשורת שבאמצעותו ניתן לשלוח ולקבל נתונים בראשת. SKU יש שני ערוצים: `input` שדרכו קוראים מידע, ו-`output` שדרכו שולחים מידע. כל SKU פועל כמו לקבל רשות מוצלב, ז"א SKU שהתקבלה פונה ל-`output` של SKU בצד השני. והשתמשנו בו בפרויקט במחלקה `AppSocket` כדי לשלוח ולקבל הודעות בין השירות ללקוח. כמו כן, במחלקה `Server` יש SKU שתפקידו להזין לבקשת החברות של לקוחות חדשים לשרת ולקבל אותם. השתמשנו בסוקטים כדי שנוכל לתקשר בין השירות ללקוח דרך הרשת.

### 4.2.4 מסד הנתונים

מסד הנתונים שמור בתיקייה `assets` (או בתיקייה שבה נמצא קובץ `jar` אם מרכיבים אחד). אנו ניגשים אליו באמצעות הפעולה `getConnection` שיוצרת חיבור עם מסד הנתונים. אנו משתמשים במסד נתונים מסווג `.accdb`, `ucanaccess`, ולכן התקשרות עם מסד הנתונים מתבצעת באמצעות הדרייבר `druid`.

#### טבלה: Users

פקוד הטבלה הינו לאחסן את פרטיים של המשתמשים. הוא משתמש לאימות פרטם שהזנו בחזון, `Login`.  
לבדיקה אם שם משתמש קיים בהרשמה של שחקן חדש ולטעינה תמונה פרופיל של שחקנים.

username	password	CreatedDateTime	ProfilePic
bezalel_6_	123456	1649627990	
bezalel0	openSesame	1649102022	
bezalel1	openSesame	1649102024	
bezalel6	123456	1649102024	<a href="#">-2-500x500.jpg</a>
testing	123456	1650018895	<a href="#">tos/414768.jpg</a>

עמודות:

- שם משתמש: מפתח, מהרוות קצרה שמכילה את שם המשתמש של המשתמש
- סיסמא: מהרוות קצרה, שמכילה את הסיסמא של המשתמש
- dateTime: מהרוות קצרה, שמכילה את מספר הדשניות מאז 12 בלילה ב-1970/1/1 [לעת](#)
- mid שמייצג את תאריך הייצור של המשתמש
- תמונה פרופיל: מהרוות קצרה שמכילה קישור לתמונה הפרופיל של המשתמש. יכולה להיות ריקה.

#### טבלה: Games

פקוד הטבלה הינו לאחסן את פרטיים של משחקים שנגמרו. כדי שהשרת יוכל לייצר סטטיסטיות לפיהם.

GameID	Player1	Player2	SavedGame	Winner	CreatedDateTime
1060376175	testing	MyAi		MyAi	1650150915
1201131107	GUEST#0	testing	[0, 126, 0, 37]	testing	1650175535
-1225359578	bezalel6	MyAi		MyAi	1650010222
1427879759	bezalel6	GUEST#1		GUEST#1	1649405537
-161528818	bezalel6	MyAi		MyAi	1650176383
1684930198	bezalel6	MyAi		MyAi	1649438352
1728624763	bezalel6	MyAi		MyAi	1650010425

## עמודות:

- GameID: מפתח, מחרוזת קצרה, שמכילה את מזהה המשחק
  - Player1: מחרוזת קצרה, שמכילה את שם המשתמש של השחקן 1 במשחק
  - Player2: מחרוזת קצרה, שמכילה את שם המשתמש של השחקן 2 במשחק
  - SavedGame: מחרוזת ארוכה, שמכילה סיריליזציה של אובייקט מסווג ArchivedGameInfo.
  - עוד פרטים, ראה את דיאגרמת ה-UML של המחלקה בסעיף המוקצה לכך.
  - Winner: מחרוזת קצרה, שמכילה את שם המשתמש של השחקן שניצח במשחק, או "----tie----" במקרה של תיקו
  - CreatedDateTime: מחרוזת קצרה, שמכילה את מספר השנהות מאז 12 בלילה ב-1/1/1970 לעת
  - מיידע שמייצג את תאריך הייצור של המשחק

## טבלה UnfinishedGames

תפקיד הטבלה הינו לאחסן את פרטיו המשחיקים שהופסקו בין משתמשים שמורים לבינה מלאכותית, כדי שהשחקנים יוכל להמשיכם בפעם הבאה שיינסנו.

GameID	Player1	Player2	SavedGame	PlayerToMove	CreatedDateTime
50090790	testing	MyAi	[3, 0, 126, 0, 22]	WHITE	1065040398
35075526	bezalel6	MyAi	[3, 0, 126, 0, 22]	WHITE	330924504
23786598	bezalel_6_	MyAi	[3, 0, 126, 0, 9]	WHITE	449632704
28929482	testing	MyAi	[3, 0, 126, 0, 22]	WHITE	438569700
91275077	bezalel6	MyAi	[3, 0, 126, 0, 22]	WHITE	1536726564
30616875	bezalel1	MyAi	[3, 0, 126, 0, 22]	WHITE	1197635786
22122712	bezalel1	MyAi	[3, 0, 126, 0, 21]	WHITE	201214215

## עמודות:

- GameID: מפתח, מחרוזת קצרה, שמכילה את מידע המשחק שהופסק
  - Player1: מחרוזת קצרה, שמכילה את שם המשתמש של שחקן 1 במשחק
  - Player2: מחרוזת קצרה, שמכילה את שם המשתמש של שחקן 2 במשחק
  - SavedGame: מחרוזת ארוכה, שמכילה ייצוג מחרוזתי של סיריליזציה של אובייקט מסווג.
  - UnfinishedGame: לעוד פרטים, ראה את דיאגרמת ה-UML של המחלקה בסעיף המוקצה לכך.

- **PlayerToMove**: מחרוזת קצרה, שמכילה את שם המשתמש שתורו לשחק
  - **CreatedDateTime**: מחרוזת קצרה, שמכילה את מספר השניות מאז 12 בלילה ב-1970/1/1 [לעת](#)
- מידע** שמייצג את תאריך הייצור של המשחק

## Threads 4.2.5

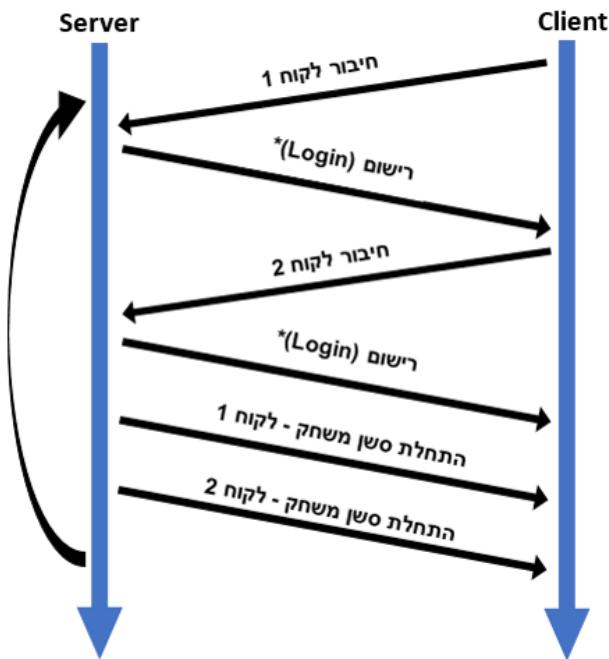
תהליכיון הינו אובייקט שמאפשר שזירה של קוד להרצתם במקביל לתהליך הראשי או תהליכיים אחרים. מה שמאפשר ביצוע מספר פעולות במקביל. חכונה שהשרות שלנו חייב. כדי שיכל לנוהל מספר משחקים ושחקנים במקביל. השתמשנו בו בפרויקט כדי לנוהל את כל המשחקים שרצים בשרת במקביל, ולקבל ל Kohotot דרך שקע במקביל. פעולה חוסמת, שהיתה מנדרלת לנו לגמרי את השרות ללא שימוש בתהליכיים. עוד שימוש היה במינימקס. שבו החיפוש מפוץ בין מספר תהליכיים, וכך מגיע לתוצאות טובות ממשמעותית. עוד שימוש היה גם אצל השרות וגם אצל הלוקה במחלקה AppSocket. שבו יש תהליך שכיל הזמן מזמן להודעות מהשרת או מהלוקה ומנתב אותו לעידן.

## 4.3 תהליכיים עיקריים וזרימת המידע

בסעיף זה נסביר את התהליכיים העיקריים שהם מרכיבת תוכנה ואת זרימת המידע ביניהם. את התהליכיים נציג בצורה גרפית באמצעות **תרשיימי רצץ** (Sequence Diagrams) שיתארו את זרימת המידע (פרוטוקולי השיח) בין השרות ללוקה הינה הchèל מהיבור הלוקחות, שידוך בין שני שחקנים להתחלה המשחק, התנהלות ומהלך המשחק ביניהם, סיום משחק ועוד.

### 4.3.1 תהליך ראשי

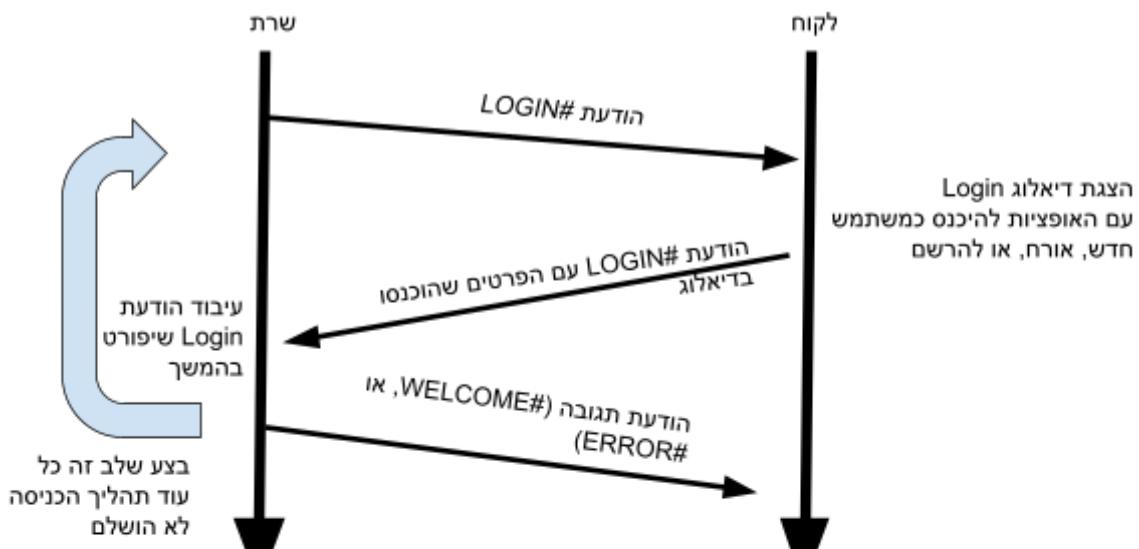
להלן תרשימים רצף עבור **התהליך הראשי** "משחק רשות – מרובה שחקנים ומשחקים":



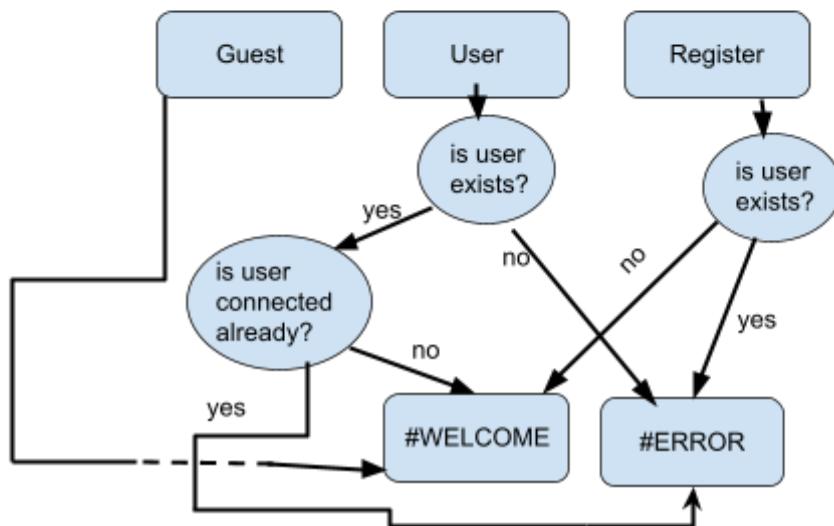
תרשים הרצף הנ"ל מתאר את פרוטוקול השיח בין השירות ללקוחות, עברו התהלייר הראשי בו השירות ממתין לחברות, בלאה אינסופית, ובמעבר כל זוג לקוחות שהתחברו בהצלחה, הוא משך ביניהם וтворע עבורם משחק.

שני החצים האנכיאים, מתארים שני הצדדים בשיח - השירות (משמאלה) והלקוח (מימין) - שמהווים את ציר הזמן. החצים השחורים האופקיים, הם ההודעות שעוברות מצד לצד, וראש החץ מעיד על כיוונם. מעל החצים יש כתוב המתאר את סוג ההודעה הנשלחת ומידע נוספת שנועבר עם ההודעה.

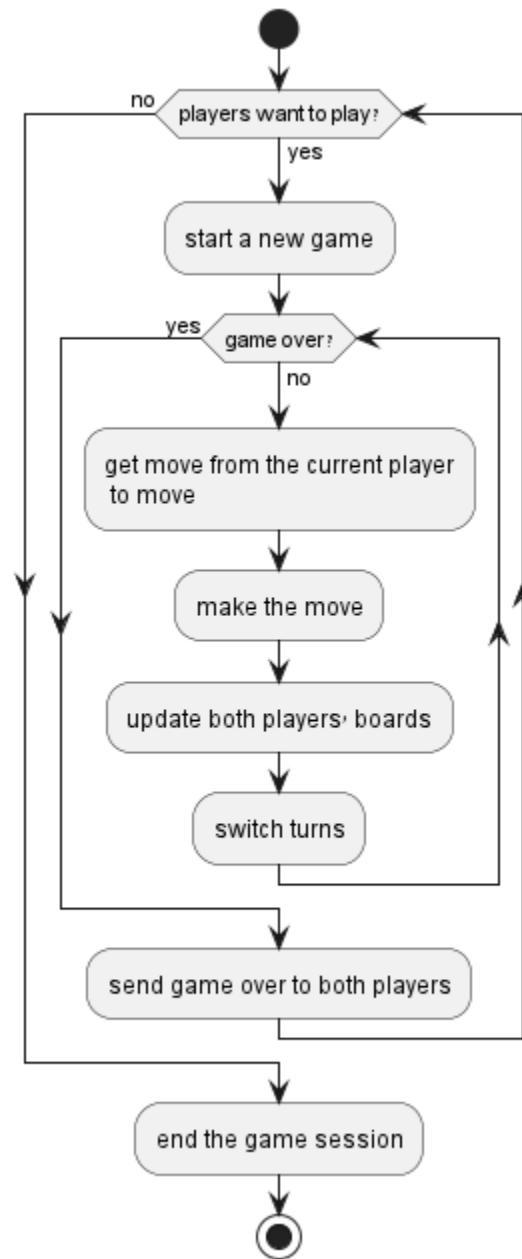
להלן תרשים רצף עבור תת-תהליך "התחברות ללקוח - **Login**:



עיבוד הודעת Login:

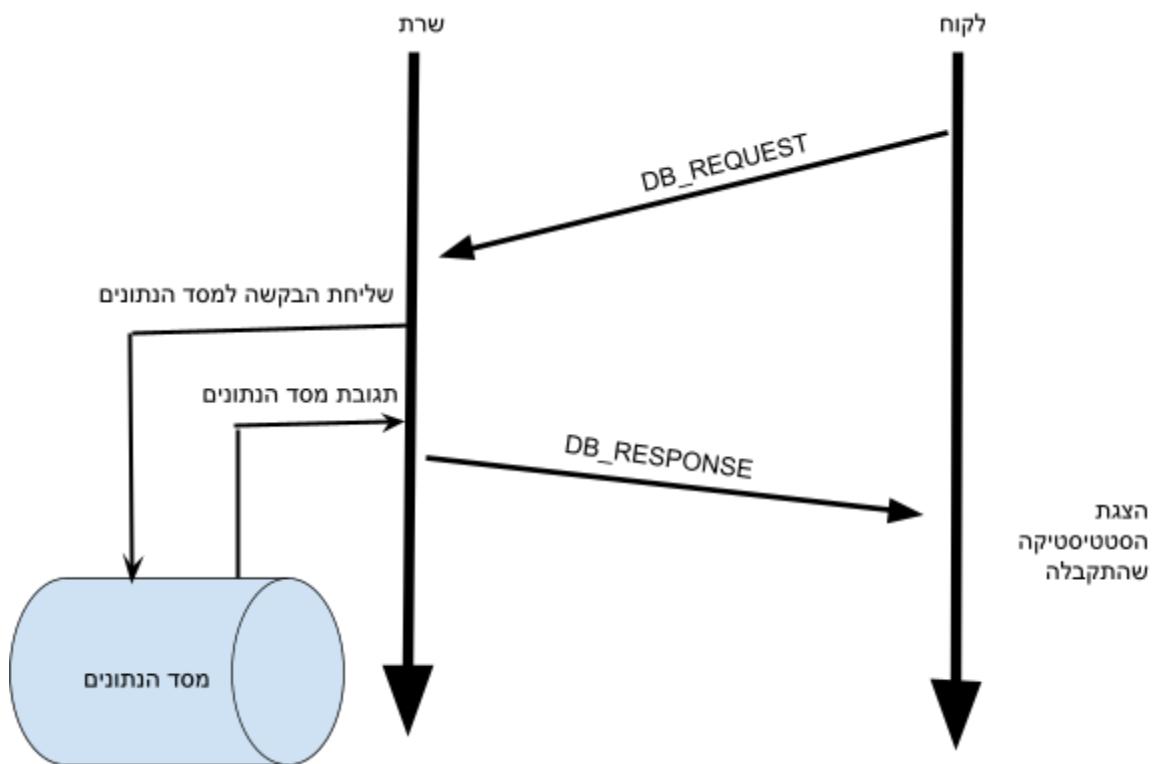


תהליך ה Game Session



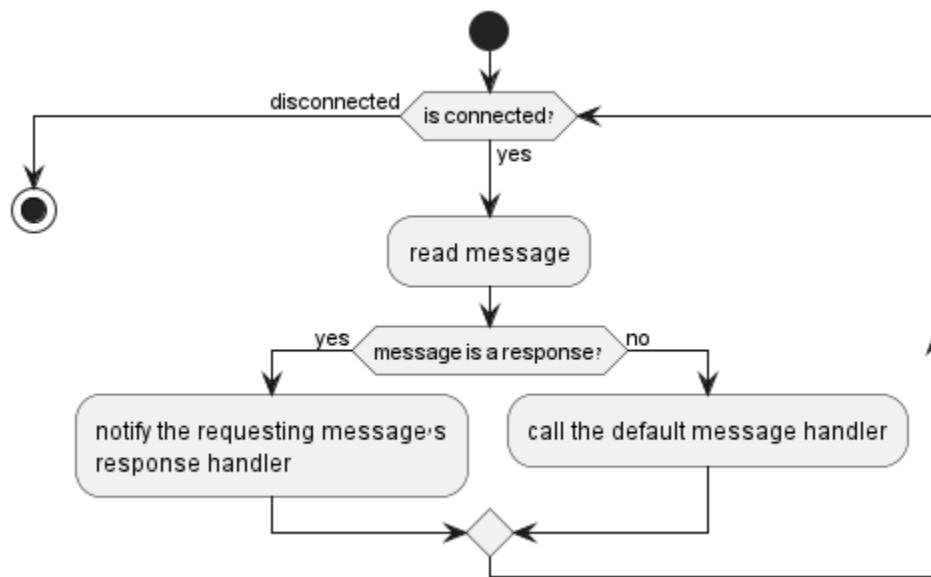
כששני השחקנים מתחילה למשחק זה עם זה, נוצר ביניהם סשן משחק. שבו מתחילה משחק חדש. בתוך לולאה: כל עוד לא נגמר המשחק, השרת מבקש מהשחקן שתורו למשחק את המהלך, מעודכן את הלוחות של 2 השחקנים, מחליף תור וחוזר חלילה. לאחר שנגמר המשחק, נשלחת הודעה סוף משחק ל-2 השחקנים, והם נשאלים האם הם רוצחים למשחק שוב. במידה וכן, נוצר ביניהם משחק חדש. אחרת, הסשן נגמר.

בקשת סטטיטיקת:



כששחקן שולח בקשה לקבלת סטטיטיקות מהשרות, השירות שולח בקשה למסד הנתונים, ממතין לקבלת תגובה, ושולח אותה חזרה לשחקן שביקש.

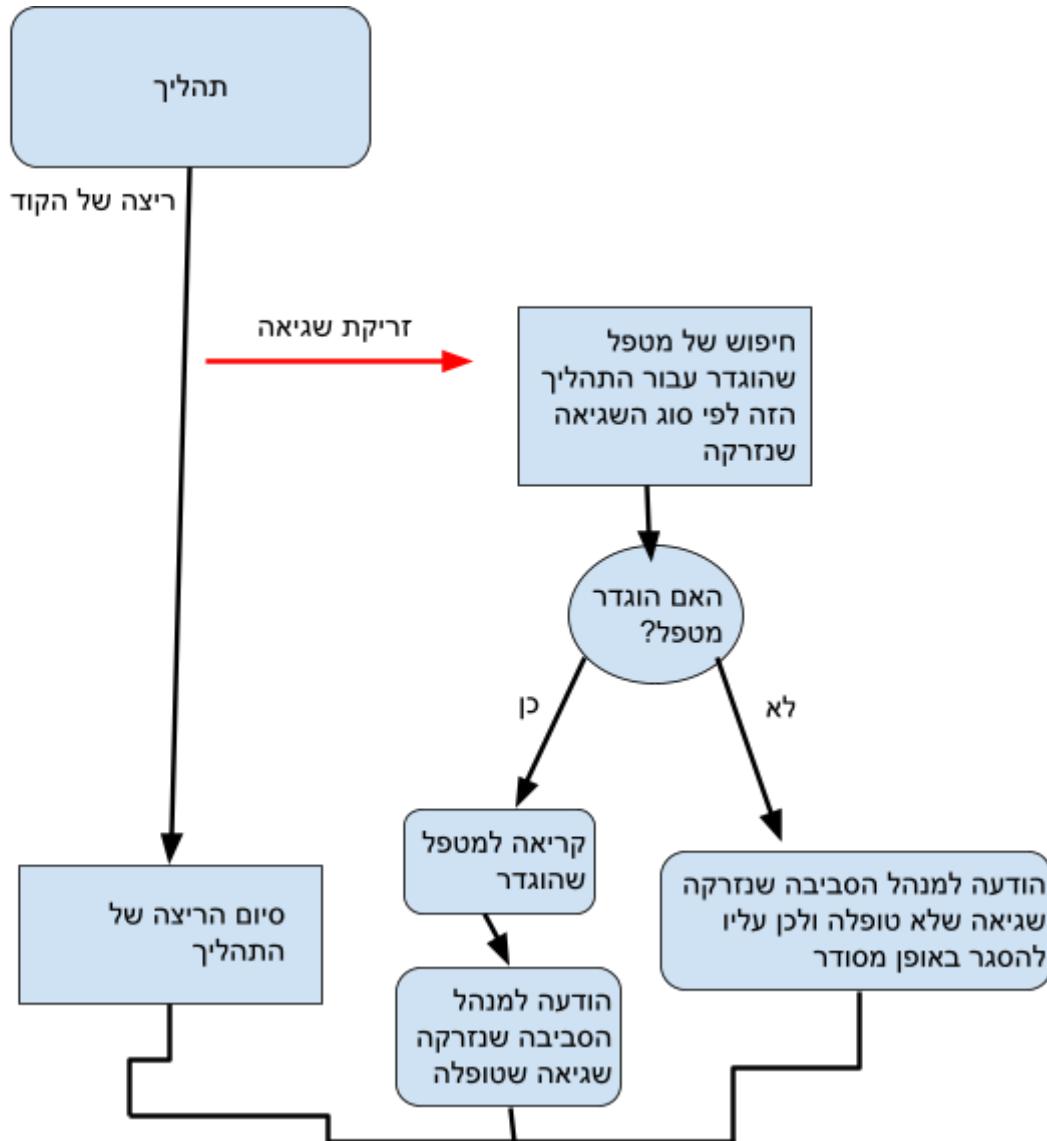
תהליך שליחת וקבלת הودעות



בתרשים לעיל מתואר תהליך קריית ההודעות וטיפול בהן. כשהודעה נקראית, היא מועברת למטפל בהודעות, שם (בתהליכי חדש, כדי לא לעצור את קריית ההודעות) ההודעה מנוטבת לעידה. במקרה שההודעה היא תגובה לבקשת, המטפל בתגובה שהועבר ביחד עם אותה בקשה נקרה. אחרת, המטפל הדיפולטיבי עברו אותה סוג הודעה נקרה.

#### תהליך טיפול בשגיאות שנורקות

את הטיפול בשגיאות ביצתי בתוכה מימוש של תהליכי 'מטופלים' ז"א תהליכי שיודיעים איך לטפל בשגיאות שועלות להיזרק.



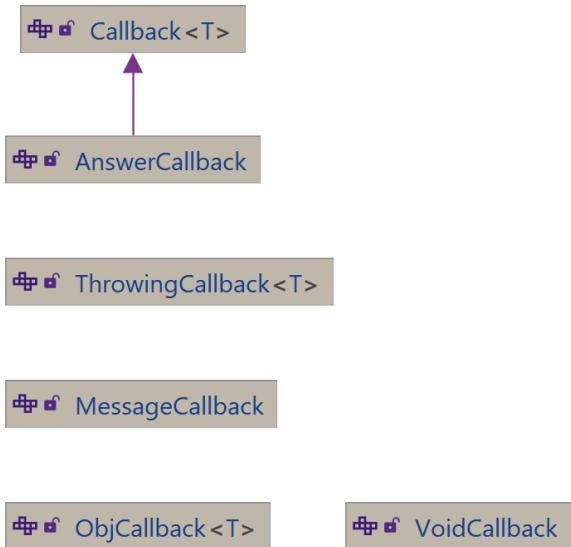
## 4.4 מחלקות הפרויקט

### תרשיimi UML של המחלקות המשותפות ללkoוח ולשרת

להלן תרשימי UML של המחלקות המשותפות ללkoוח ולשרת, הבניי מ-90 מחלקות שעלייהן נכתב פירוט מעמיק...

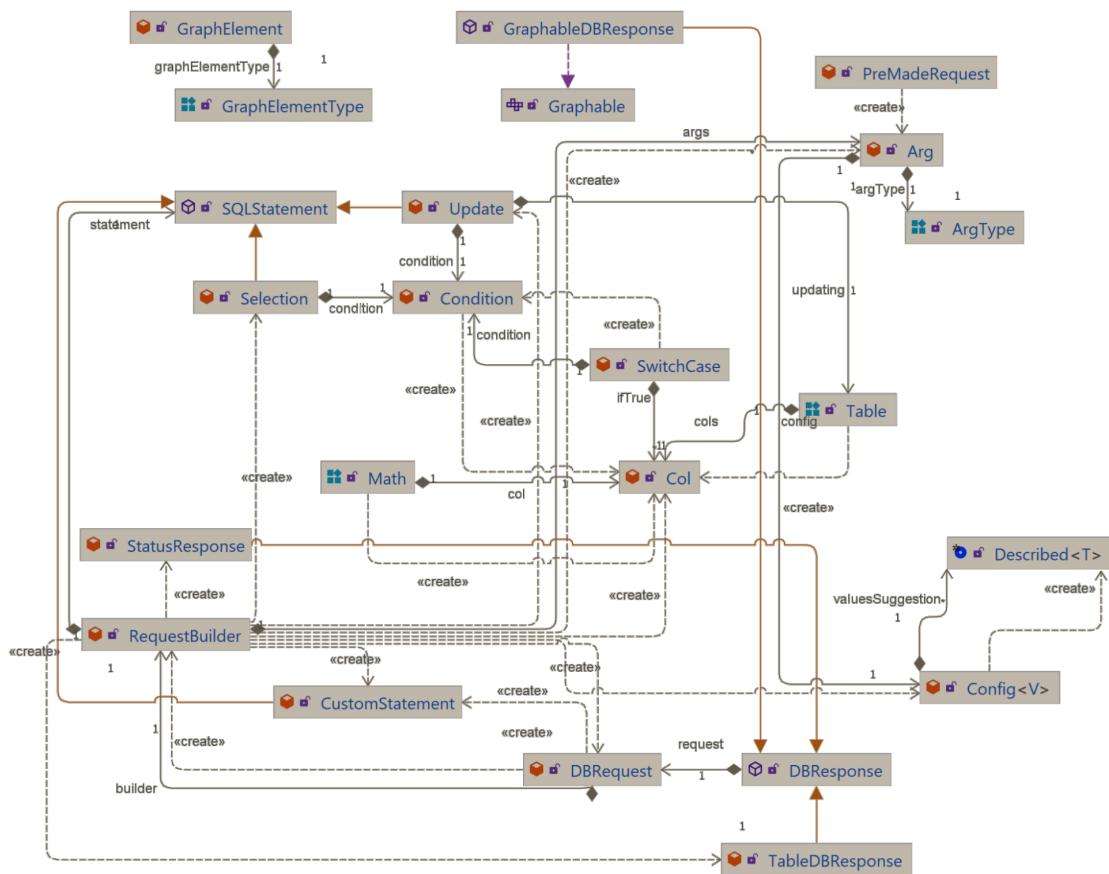
#### Callbacks

משמשים לקריאה א-סינכרונית של פעולות.



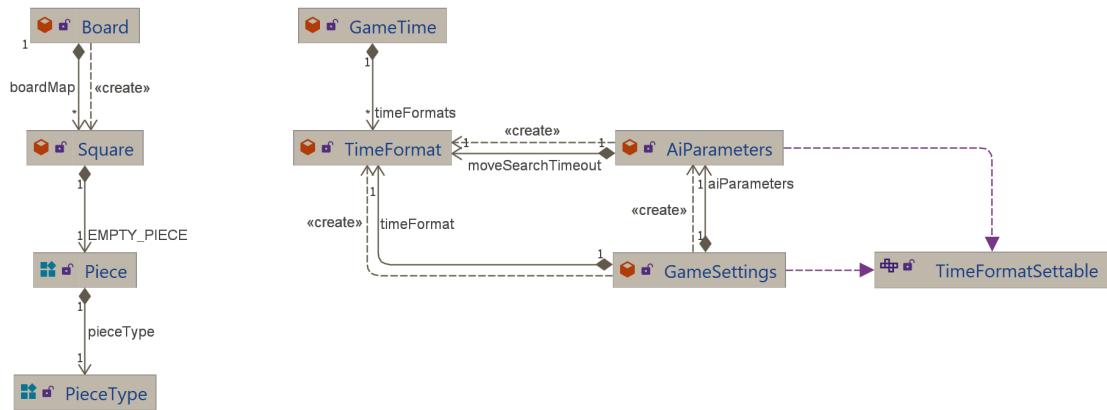
## DBActions

מחלקות הקשורות לasad הנתונים וביצוע פעולות עליון



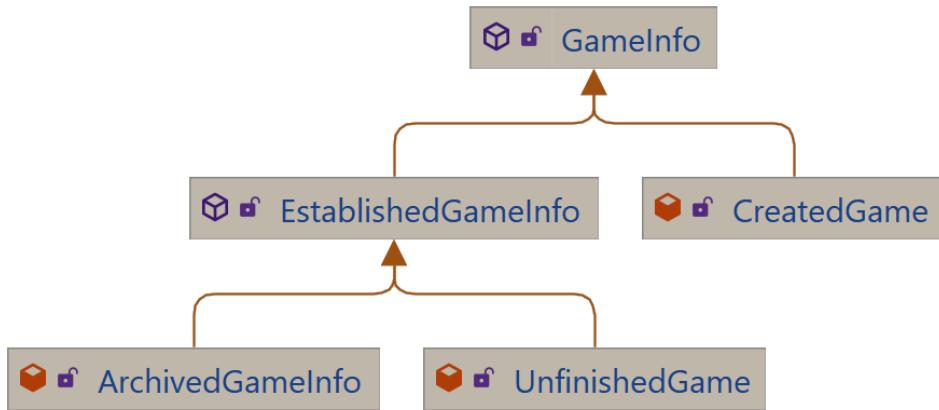
## GameSetup

## מחלקות הקשורות לאתחול וניהול המשחק



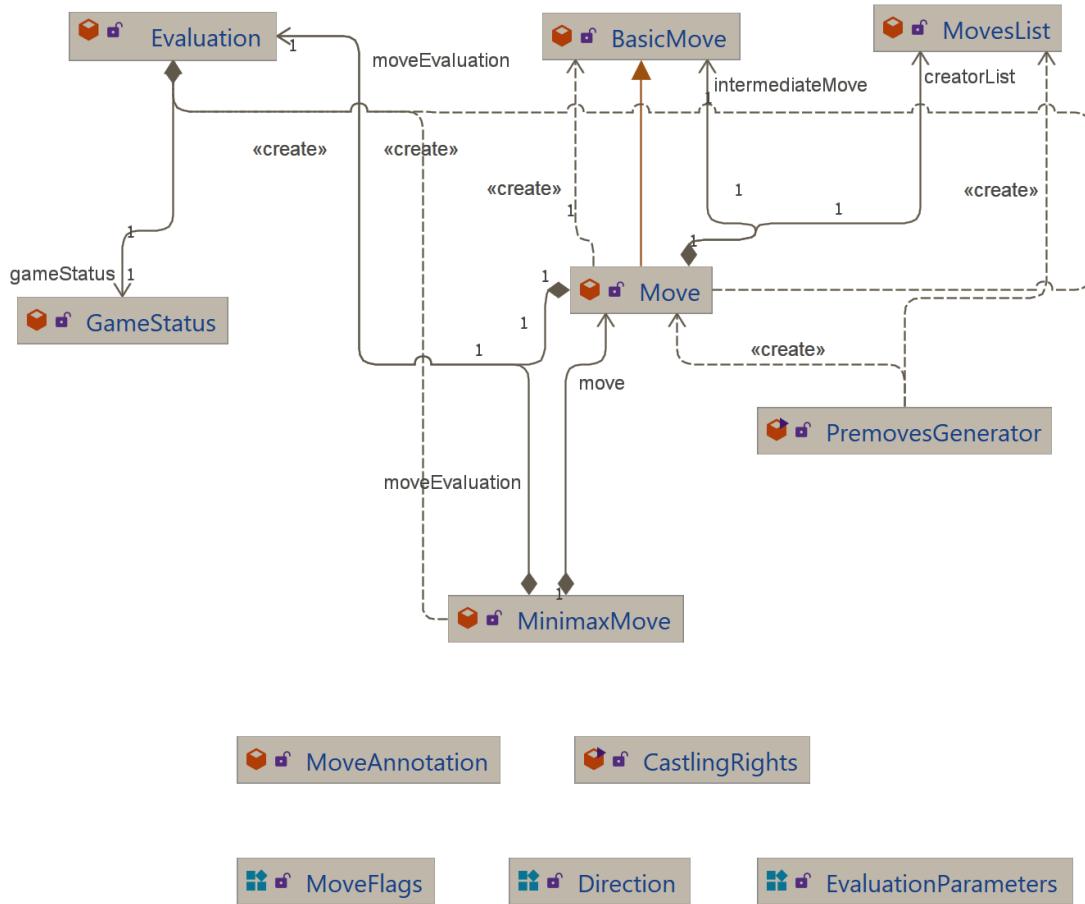
## SavedGames

ייצוג משחקים שמורים

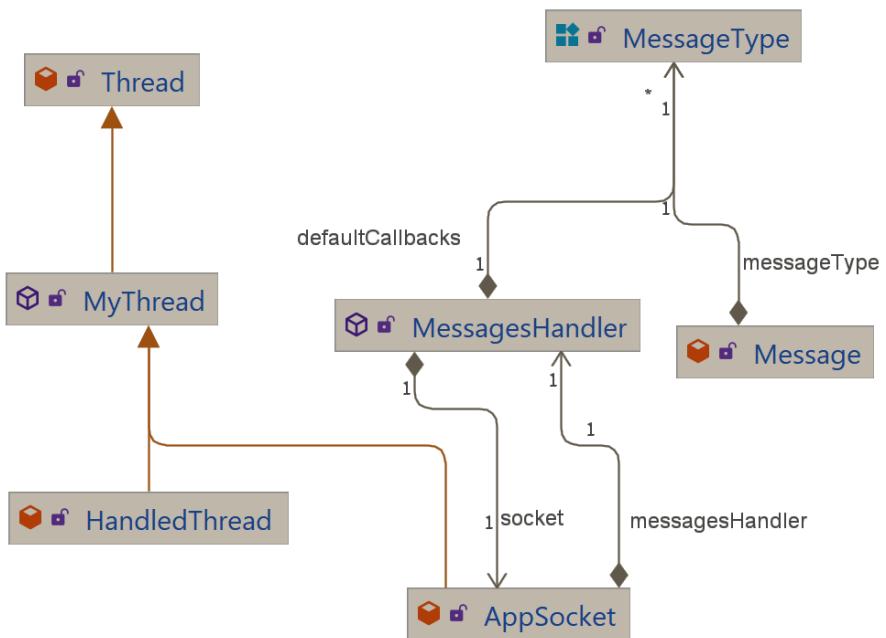


## Moves

יצוג, ייצור, וניהול מילכים.

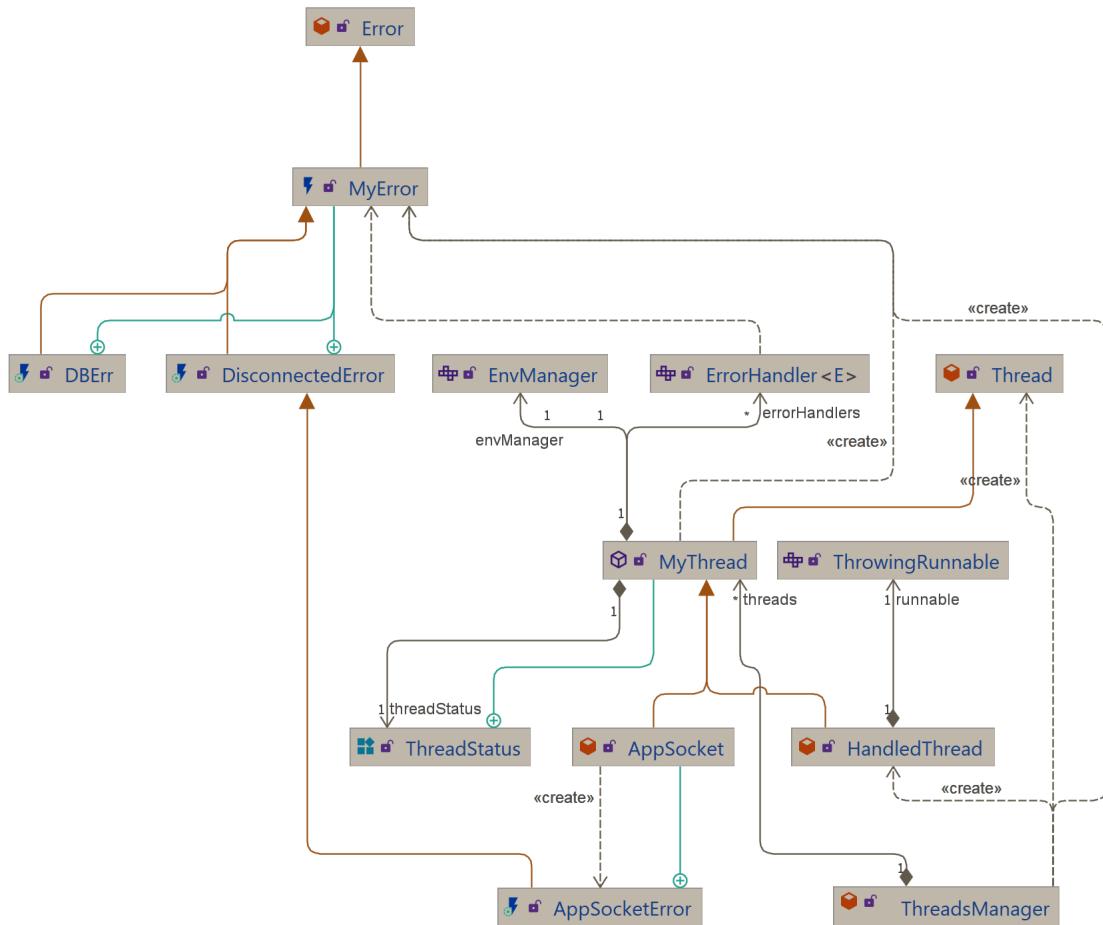


## Networking



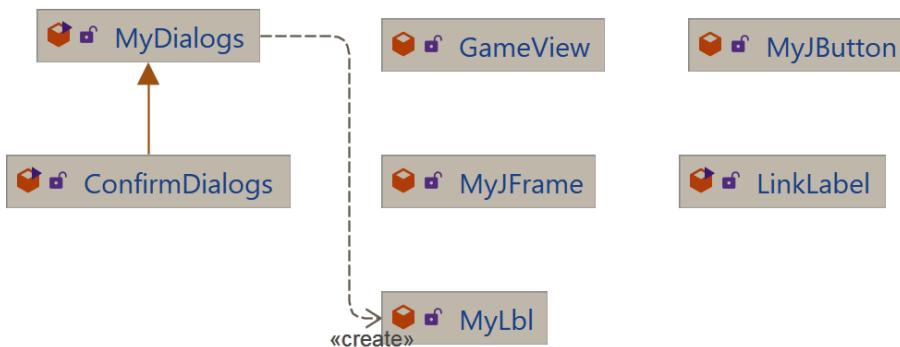
## Threads

תהליכיונים



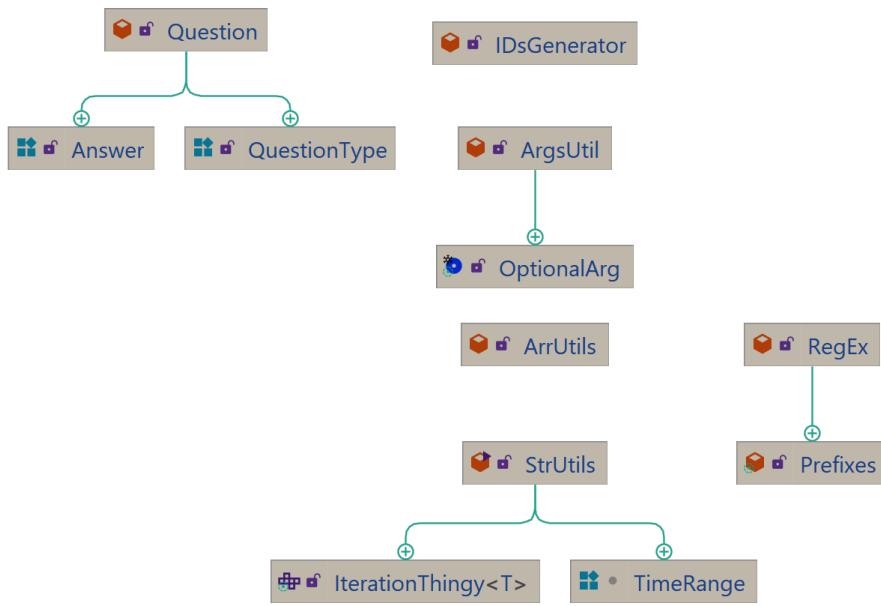
## UI

רכיבים גרפיים שימושתיים ללקוח ולשרה



## Utils&Misc

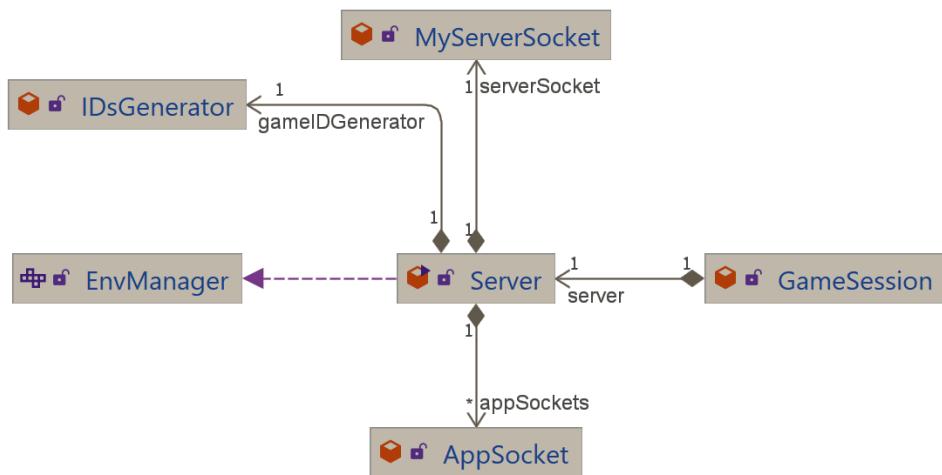
מחלקות שירות ושותנות



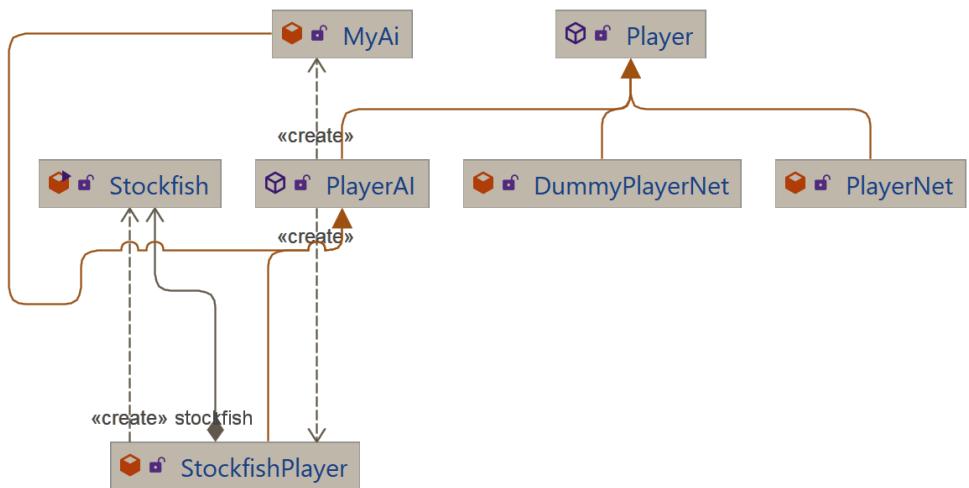
#### 4.4.1 תרשימים UML של המחלקות לצד השירות

להלן תרשימי UML של הצד השירות, הבניי 48 מחלקות שעלייהן אכתוב פירוט מעמיק...

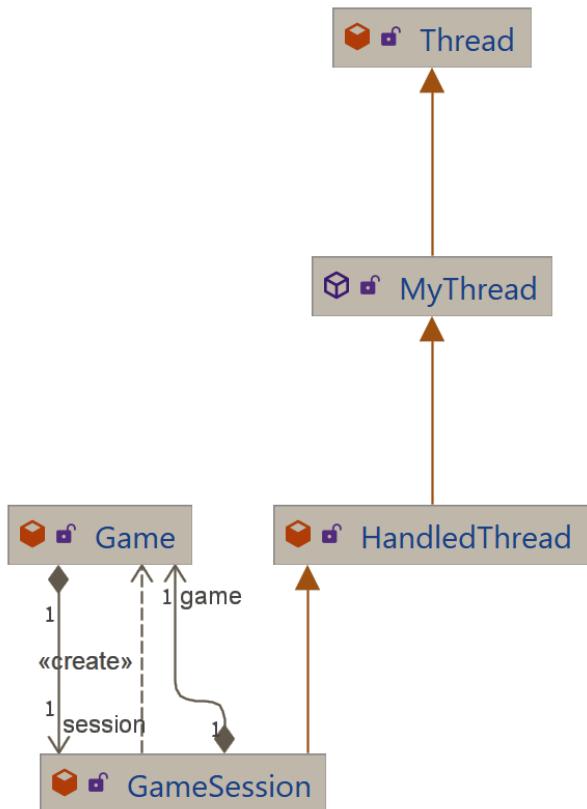
##### Server



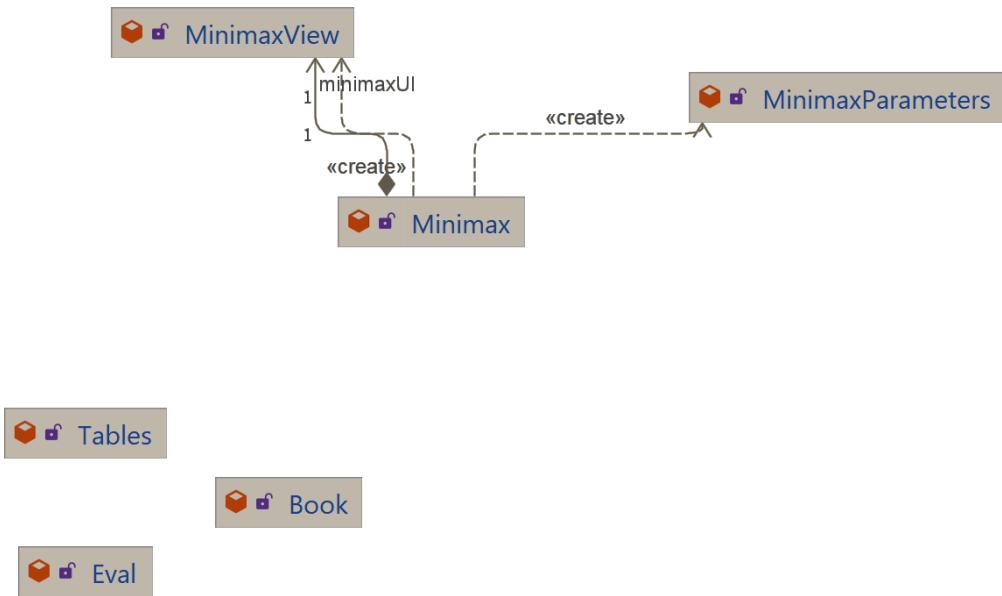
##### Players



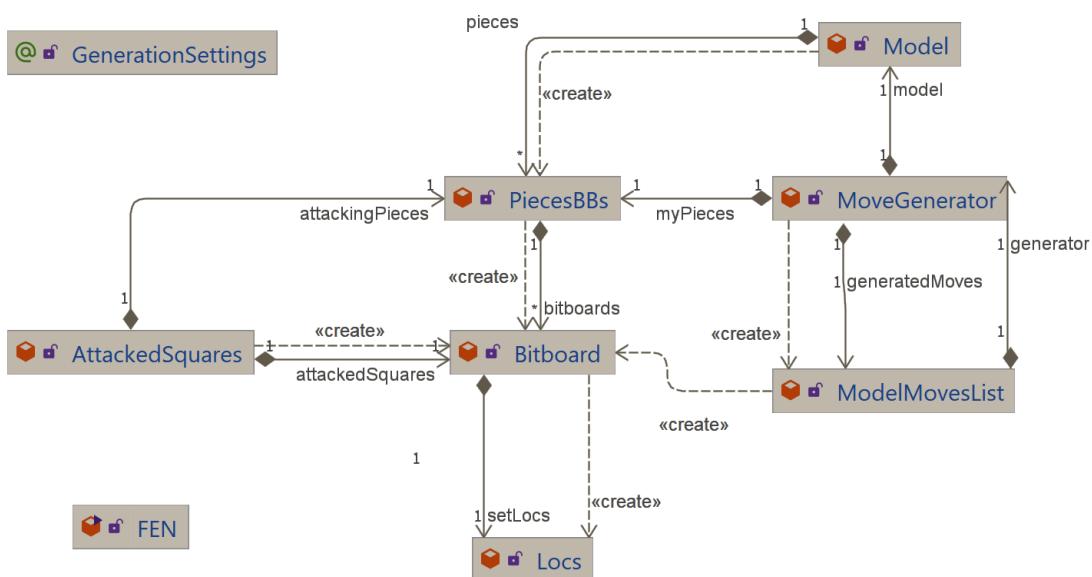
## Game



## Minimax



## Model



## 4.4.2 תייעוד המחלקות מצד השרת

Server		
Server()		
serverPort		int
serverSocket		MyServerSocket
frmWin		MyJFrame
appSockets		ArrayList <AppSocket>
syncedLists		ArrayList <SyncedItems<?>>
serverRunOK		boolean
SERVER_LOG_FGCOLOR		Color
areaLog		JTextArea
serverSetupOK		boolean

represents the chess server that is responsible for accepting incoming clients, manage them and the games they play.

### `Server()`

Constructor for ChessServer.

#### `void runServer()`

Run the server - wait for clients to connect and handle them. all processing from now on is done in a handled code to prevent catastrophic error throwing for any reason.

#### `void handleClient(AppSocket playerSocket)`

Handle client in a separate thread to allow for concurrent client handling.

**Parameters:** `playerSocket` - the socket to the player

`PlayerNet login(AppSocket appSocket)` throws `DisconnectedError`

login a new client

**Parameters:** `appSocket` - the app socket to the client **Returns:** the newly created player net **Throws:**

`DisconnectedError` - if the player disconnected while logging in

#### `Message responseToLogin(LoginInfo loginInfo)`

`throws DisconnectedError`

create a Response to a login message attempt.

**Parameters:** `loginInfo` - the login info **Returns:** the response message. a welcome message after a successful login, or an error message. **Throws:** `DisconnectedError` - if the player disconnected while logging in

#### `void gameSetup(Player player)`

ask the player for his preferred game settings and sets him up for a game (if possible).

**Parameters:** `player` - the player

#### `void playerDisconnected(Player player, String message)`

handle a player disconnected event. an attempt will be made to send a bye message to the disconnecting player. if the player is mid-game, the game session will be notified. if the player is waiting for a match, he will be removed from the queue. if the player has a game in the pool (waiting for other players to join to his game) it is removed.

**Parameters:** `player` - the player `message` - the disconnection description

את שאר הפעולות ניתן לראות בקוד עצמו

ServerMessagesHandler	
<b>ServerMessagesHandler(Server, AppSocket)</b>	
player	PlayerNet
server	Server
onCancelQuestion()	MessageCallback
onQuestion()	MessageCallback
onDBRequest()	MessageCallback
onUnplannedDisconnect()	void
onUsernameAvailability()	MessageCallback
createDisconnectedError()	DisconnectedError
onResign()	MessageCallback
setPlayer(PlayerNet)	void
onBye()	MessageCallback
...	

```
public class ServerMessagesHandler
extends MessagesHandler
Server messages handler.

ServerMessagesHandler(Server server, AppSocket appSocket)
Instantiates a new Server messages handler.
```

את שאר הפעולות ניתן לראות בקוד עצמו

UsernameSuggestions	
<b>UsernameSuggestions()</b>	
postUnderscore	MatchIterations
upper	MatchIterations
bothUnderscore	MatchIterations
lower	MatchIterations
preUnderscore	MatchIterations
maxSuggestions	int
numOfIterationsPerOptionGroup	int
createOptions(String, String, MatchIterations[])	ArrayList<ArrayList<String>>
createSuggestions(String)	ArrayList<String>
...	

```
public class UsernameSuggestions
Username suggestions utility class that generates username suggestions for players who are trying to
register and their username is used already
```

```
UsernameSuggestions()
ArrayList<String> createSuggestions(String username)
```

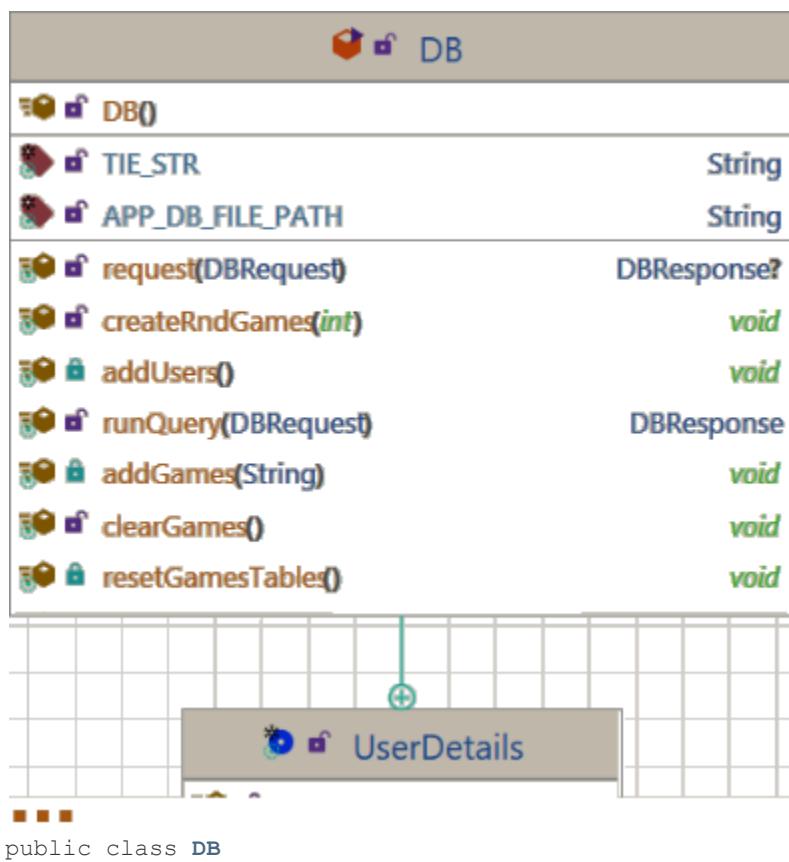
Create suggestions array list.

**Parameters:** username - the username **Returns:** the array list  
את שאר הפעולות ניתן לראות בקובץ עצמו

`UsernameSuggestions.MatchIterations`  
Match iterations.

`String createStr(MatchResult result, int iteration)`  
Create str string.

**Parameters:** result - the result iteration - should start at 1 **Returns:** string



utility class for communicating with the database.

`StatusResponse runUpdate(DBRequest request)`

Run a requested update on the db, and return the response for it. the response can be a success, with the number of updated rows, or an error.

**Parameters:** request - the request **Returns:** the status response

`Connection getConnection()`

create a connection to the db.

**Returns:** the created connection to the db.

`DBResponse runQuery(DBRequest request)`

Run a query on the db, and return the db's response. the response is built by individual requests allowing for maximum flexibility.

**Parameters:** request - the request **Returns:** the db response

`DBResponse request(DBRequest request)`

apply a db request, and return the db's response. a request will be forwarded to either:

`runUpdate(DBRequest) or runQuery(DBRequest)`

**Parameters:** request - the request **Returns:** the db's response to the request

את שאר הפעולות ניתן לראות בקוד עצמו

[Game](#)

 GameSession	
  GameSession(UnfinishedGame, Player, Player, Server)	
  GameSession(String, Player, Player, GameSettings, Server)	
  GameSession(EstablishedGameInfo, Player, Player, Server)	
  GameSession(GameInfo, Player, Player, Server)	
 game	Game
 gameID	String
 server	Server
 creator	Player
 p2	Player
 getSyncableItem()	SyncableItem
***	

```
public class GameSession
extends HandledThread
implements SyncableItem
```

Game session - represents a game session between two players. a session is running as long as both players are connected, and want to keep playing with each other.

```
GameSession(GameInfo gameInfo, Player creator, Player otherPlayer, Server
server)
```

Instantiates a new Game session.

```
GameSession(UnfinishedGame unfinishedGame, Player creator, Player otherPlayer,
Server server)
```

Instantiates a new Game session. resuming an unfinished game.

```
void handledRun()
```

run the game inside the HandledThread's 'container' with handlers setup for the relevant errors that might get thrown.

**Overrides:** handledRun in class HandledThread

את שאר הפעולות ניתן לראות בקוד עצמו

Game		
Game(Player, Player, GameSettings, GameSession)		
COLS		<i>int</i>
gameCreator		Player
originalSettings		GameSettings
session		GameSession
p2		Player
gameTime		GameTime
ROWS		<i>int</i>
isReadingMove		<i>boolean</i>
showGameView		<i>boolean</i>
...		

```
public class Game
```

Game - represents a game between two Players.

```
Game(Player gameCreator, Player p2, GameSettings gameSettings, GameSession session)
```

Instantiates a new Game.

```
GameStatus runNewGame()
```

Starts a new game, and eventually returning the game over status.

**Returns:** the game over status

```
GameStatus playTurn()
```

gets the current player's move, makes it, and returns the game status after making the move. might stop and not finish all those steps if it gets interrupted by a disconnected player or the current player timing out. in which case the appropriate game status will be returned.

**Returns:** the game status

את שאר הפעולות ניתן לראות בקובץ עצמוני

GameOverError		
GameOverError(GameStatus)		
gameOverStatus	GameStatus	

```
public class GameOverError  
extends MyError
```

Game over error - represents an error that will cause a game over. for example: a player disconnected.

```
GameOverError(GameStatus gameOverStatus)
```

Instantiates a new Game over error.

		<b>PlayerDisconnectedError</b>
		<b>PlayerDisconnectedError (Player)</b>
		<b>disconnectedPlayer</b> Player
		<b>createGameStatus ()</b> GameStatus
		<b>getDisconnectedPlayer ()</b> Player

```
public class PlayerDisconnectedError
extends DisconnectedError
an error that is thrown when a player is disconnected. NOTE: this error is not to be sent to the client, as it
holds a reference to a Player object

PlayerDisconnectedError (Player disconnectedPlayer)
Instantiates a new Player disconnected error.
```

את שאר הפעולות ניתן לראות בקובץ עצמו  
Model

		<b>Eval</b>
		<b>Eval(Model, PlayerColor, boolean)</b>
		<b>Eval(Model, PlayerColor)</b>
		<b>model</b> Model
		<b>opponentColor</b> PlayerColor
		<b>evaluation</b> Evaluation
		<b>endgameMaterialStart</b> double
		<b>evaluationFor</b> PlayerColor
		<b>playerToMove</b> PlayerColor
		<b>egWeight</b> double
		<b>PRINT_REP_LIST</b> boolean

```
...
public class Eval
implements Serializable
Eval - evaluate a given position for a player color. the evaluation is consistent to both players. meaning that
an evaluation for any position is going to be the same for both players, only multiplied by -1 for the other
player.
```

**privateEval (Model model, PlayerColor evaluationFor)**
Instantiates a new Eval.

```
privateEval (Model model, PlayerColor evaluationFor, boolean
onlyCheckForGameOver)
Instantiates a new Eval.
```

**Evaluation checkGameOver ()**

checks if the game is over in the current position.

**Returns:** if this position is a game over, the game over evaluation. otherwise an empty evaluation.  
**void calcEvaluation ()**

if the game isn't over, an evaluation is calculated.

```
int materialSum (PlayerColor playerColor)
```

the sum of all the pieces' values of a player

**Parameters:** playerColor - the player color **Returns:** the sum in centipawns

int **pieceTables**(PlayerColor clr)  
calculates piece tables evaluation for a player

**Parameters:** clr - the clr **Returns:** the int

את שאר הפעולות ניתן לראות בקובץ עצמו

Tables		
Tables()		
● MIDDLE_GAME	int	
● queen	PieceTable	
● king	PieceTable	
● pawn	PieceTable	
● knight	PieceTable	
● rook	PieceTable	
● pieceTables	PieceTable[]	
● ENDGAME	int	
● bishop	PieceTable	

public class **Tables**

represents all pieces value tables. the tables are used to calculate the Evaluation of a position.

**See Also:** for more information

**getPieceTable** public static Tables.PieceTable **getPieceTable**(PieceType pieceType)  
Gets a piece's table by piece type.

**Parameters:** pieceType - the piece type **Returns:** the piece table

את שאר הפעולות ניתן לראות בקובץ עצמו

PieceTable		
PieceTable(int[][], int[][])		
● tables	int[][][]	
● init(int[][])	int[][][]	
● reverse(int[][])	int[][]	
● getValue(double, PlayerColor, Location)	int	

#### Tables.PieceTable

represents a Piece table that has separate middlegame and endgame tables in centipawns for each square on the board.

**PieceTable**(int[][] middleGame, int[][] endGame)  
Instantiates a new Piece table.

int **getValue**(double egWeight, PlayerColor player, Location loc)  
את שאר הפעולות ניתן לראות בקובץ עצמו

 Book		
 Book()		
 book	File	
 pathToBook	String	
 getBookMove(Model)	String?	
 checkBook()	void	

```
public class Book
Book - utility class for interacting with the opening book. an opening book is a database containing lines of
openings.
```

**Book()**

**String getBookMove(Model model)**

looks for a game matching the current game inside the book games database. if one is found, the next move is saved. after going through all games in the database, if any matching game was found, a random move from the saved moves is returned. if no matching game was found, null is returned.

**Parameters:** model -- current game position **Returns:** a random move from every game found in the games' database, if one is found. null otherwise

את שאר הפעולות ניתן לראות בקובץ עצמו

 Minimax		
 Minimax(Model, long)		
 Minimax(Model, long, long)		
 transpositionHits	long	
 minimaxUI	MinimaxView	
 interrupt	MyError	
 DEBUG_MINIMAX	String	
 cpuUsageRecords	CpuUsages	
 scanTimeFlexibility	long	
 NUMBER_OF_THREADS	int	
 positionsReached	long	

```
public class Minimax
```

Minimax - represents my implementation of a multithreaded minimax algorithm.

**Move getBestMove(PlayerColor player)**

Gets best move using minimax.

**Parameters:** player - the player color to search for **Returns:** the best move the minimax found

MinimaxMove **minimaxRoot(Model model, int maxDepth, PlayerColor playerColor)** the entry point for the minimax.

**Parameters:** model -- current game position **maxDepth** -- the maximum depth the minimax can reach

**Returns:** best move for the current player to move

**void startMultithreaded(Model model, PlayerColor minimaxPlayerColor, int**

**maxDepth)**

**throws InterruptedException**

starts a multithreaded minimax search

**Parameters:** model -- current game position minimaxPlayerColor -- current player to move maxDepth -- the maximum depth the minimax can reach **Throws:** InterruptedException  
את שאר הפעולות ניתן לראות בקוד עצמו

MinimaxParameters	
<b>MinimaxParameters</b> (Model, boolean, int, int, PlayerColor, int)	
<b>MinimaxParameters</b> (Model, boolean, int, PlayerColor)	
<b>model</b>	Model
<b>b</b>	int
<b>isMax</b>	boolean
<b>a</b>	int
<b>maxDepth</b>	int
<b>currentDepth</b>	int
<b>minimaxPlayerColor</b>	PlayerColor
<b>isMax()</b>	boolean
...	

public class MinimaxParameters  
parameters used by the Minimax.

את שאר הפעולות ניתן לראות בקוד עצמו

MinimaxView	
<b>MinimaxView</b> (boolean, Minimax)	
<b>timer</b>	Timer
<b>EMPTY</b>	String
<b>currentDepthLbl</b>	JLabel
<b>reachedDepthTimeLbl</b>	JLabel
<b>format</b>	DecimalFormat
<b>bestMoveSoFarLbl</b>	JLabel
<b>chosenMoveLbl</b>	JLabel
<b>moveEvaluationLbl</b>	JLabel
<b>font</b>	Font
...	

public class MinimaxView  
extends JFrame

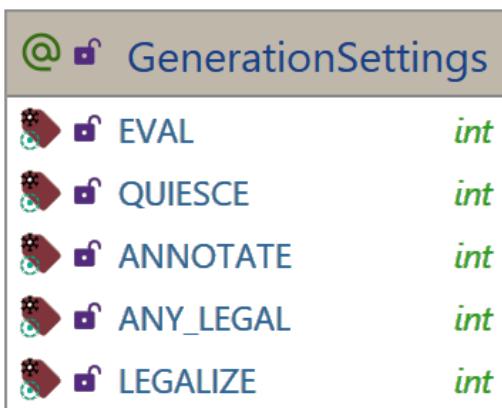
Minimax view - represents a debugging frame for watching the minimax 'think' in real time. can be activating by passing "DEBUG\_MINIMAX" as an argument when running the server

את שאר הפעולות ניתן לראות בקוד עצמו



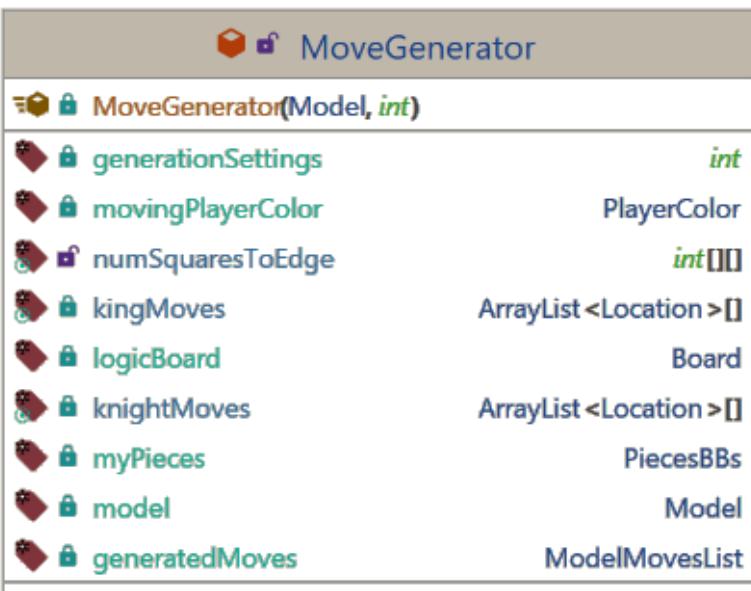
```
class QuietInterrupt
extends Throwable
```

Quiet interrupt - an interrupt meant to stop the search quietly, without throwing anything outside the minimax. the returned move will be the best move found until interrupted. NOTE: the move returned might be null as the search could've found no move yet.



```
public @interface GenerationSettings
```

Generation settings - a magic constant used to define the possible move-generation settings. example usecase: when evaluating a position, the first thing to look for is game overs. the most common of which are: checkmates, and stalemates. in order to efficiently check for them, the game looks for ANY legal move the current player can make. if one is found, it rules out both checkmate and stalemate. since both of them require the player having no legal move. and so, instead of generating all moves for all the player's pieces, the generation setting ANY\_LEGAL is used. which stops the moment it finds a legal move.



```
public class MoveGenerator
```

Generates moves from a given position and GenerationSettings

```
ModelMovesList generateMoves(Model model, int generationSettings)
```

Generate moves according to the given generationSettings

**Parameters:** model - the model generationSettings - the generation settings **Returns:** the list of moves generated

את שאר הפעולות ניתן לראות בקוד עצמו

AttackedSquares	
• AttackedSquares(PlayerColor, PiecesBBs, Bitboard)	
• AttackedSquares(Model, PlayerColor)	
attackingPlayerColor	PlayerColor
attackedPlayerColor	PlayerColor
attackingPieces	PiecesBBs
checkingAttacked	Location
attackedPlayerBB	Bitboard
attackedSquares	Bitboard
getPieceAttacksFrom(PieceType, Bitboard, PlayerColor, Model)	Bitboard
attack(PieceType, Bitboard, Direction[])	void
...	

```
public class AttackedSquares
```

Attacked squares - calculates which squares are attacked, by using bitwise operations on pieces Bitboards. saving a lot of time by batch calculating for each piece type instead of calculating by individual piece

Bitboard **getAttackedSquares**(Model model, PlayerColor attackingPlayerColor)  
Gets a bitboard of all the attacked squares.

**Parameters:** model - the model attackingPlayerColor - the attacking player color **Returns:** a bitboard of all the squares attacked by the attackingPlayerColor

```
void attack(PieceType pieceType, Bitboard attackingPiecesBB,
attackingDirections)
```

Attack - shift the given Bitboard according to the attacking directions. resulting in a quick, batch calculation of all the attacked squares by the given piece

**Parameters:** pieceType - the piece type attackingPiecesBB - the attacking pieces bb attackingDirections - the attacking directions that will be used to calculate the attack. if none will be passed, the directions used will be the piece type's attacking directions  
boolean **isAttacked**(Model model, Location loc, PlayerColor attackingPlayerColor)

Is the given loc threatened by the attackingPlayerColor.

**Parameters:** model - the model representing the current position loc - the loc attackingPlayerColor - the attacking player color **Returns:** true if the loc is attacked by the attacking player, false otherwise.

```
Bitboard getPieceAttacksFrom(PieceType pieceType, Bitboard pieceBB,
PlayerColor attackingPlayerColor, Model model)
```

Calculate a bitboard of the pieceType attacks from the pieceBB. every bit set on the pieceBB will be treated as a pieceType, and will attack like one.

**Parameters:** pieceType - the piece type pieceBB - the piece bb attackingPlayerColor - the attacking player color model - the model representing the current position **Returns:** a Bitboard representation of all the attacked squares

את שאר הפעולות ניתן לראות בקוד עצמו

Bitboard	
Bitboard(Bitboard, Bitboard, Direction, PlayerColor)	
Bitboard(Bitboard)	
Bitboard(Location)	
Bitboard(long)	
Bitboard()	
onSet	VoidCallback
lastSet	long
setLocs	Locs
lastSetLoc	Location
bitBoard	long
...	

```
public class Bitboard
implements Serializable
Bitboard - a bitboard representation of the chess board. every bit set is a chess piece on that square.
void set(Location loc, boolean state)
Sets the bit on the loc according to state. if state is true, the bit is set to 1 if state is false the bit is set to 0
Parameters: loc - the loc state - the state.
Bitboard shiftMe(PlayerColor playerColor, Direction direction)
performs a bitwise shift on this bitboard instance.
Parameters: playerColor - the player color. used for perspective
Returns: direction - the direction to shift in
Bitboard orEqual(long l)
Or equal bitboard. performs a bitwise or on this board
Parameters: l - the other board to or with
Returns: the changed bitboard
boolean isEmpty()
Is this board empty
Returns: is this board empty
Bitboard exclude(Bitboard other)
performs a bitwise and with 2's compliment of other. practically setting all other bits to zero
Parameters: other - the other
Returns: this changed bitboard
Bitboard and(long other)
performs a bitwise and on a new copy of this bitboard.
Parameters: other - the other
Returns: the new changed bitboard
את שאר הפעולות ניתן לראות בקוד עצמו
```

 FEN	
 FEN()	
 startingFen	String
 rndFens	String[]
 castling	String
 rndFen()	String
 extractEnPassantTargetLoc(String)	String?
 loadFEN(String, Model)	void
 isValidFen(String)	boolean
 generateFEN(Model)	String
 assertFen(String)	void

...  
 public class FEN  
 Fen - utility class used for loading a position from a FEN string, and creating a FEN string from a given position.

**See Also:** ...

String **generateFEN**(Model model)  
 Generate fen string.

**Parameters:** model - the model **Returns:** the fen for the given model's position

boolean **isValidFen**(String fen)  
 Is valid fen.

**Parameters:** fen - the fen **Returns:** true if the fen is valid and false otherwise

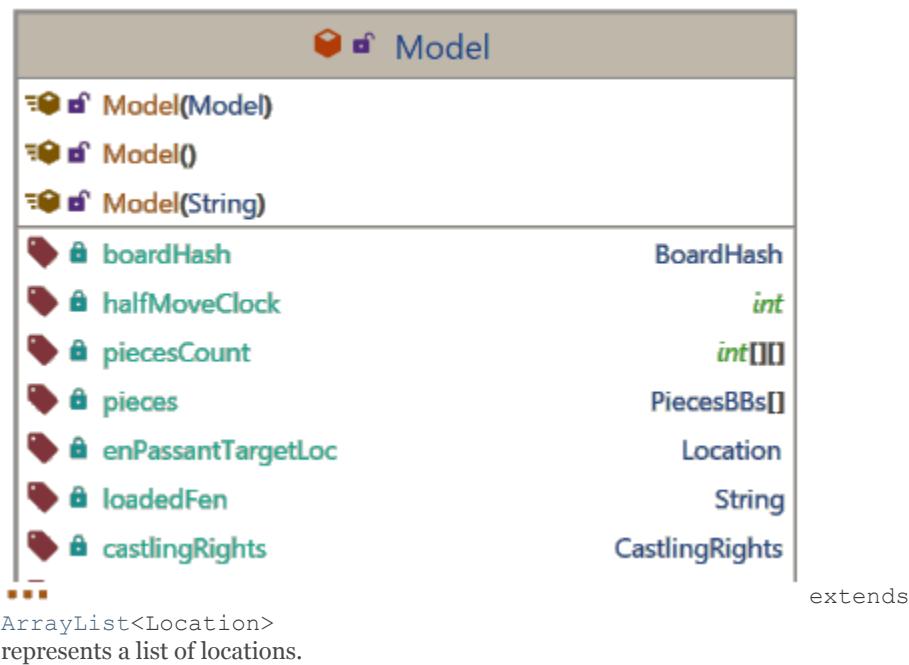
void **loadFEN**(String fen, Model model)  
 Loads a fen.

**Parameters:** fen - the fen model - the model

את שאר הפעולות ניתן לראות בקוד עצמו

 Locs	
 Locs()	
 Locs(Locs)	

<Location>, Collection<Location>, List<Location>, RandomAccess  
 public class Locs



```

public class Model
  implements Serializable
Model - handles all game logic.
  
```

**Model (String fen)**  
Instantiates a new Model, and setting it up from the fen.

**void setup (String fen)**  
initializes the model according to a given position's string representation.

**Parameters:** fen - the position's fen **See Also:** FEN  
**ModelMovesList generateAllMoves ()**  
Generate all moves for the current player to move.

**Returns:** a list with all the moves the current player can play  
**boolean isInCheck (PlayerColor playerColor)**  
Is playerColor in check.

**Parameters:** playerColor - the player color **Returns:** true if player is in check, false otherwise.  
**void applyMove (Move move)**  
applies move to the current position.

**Parameters:** move - the move  
**void undoMove (Move move)**  
Undoes the change made to the board by applyMove (Move) . essentially playing the moves in reverse.

**Parameters:** move - the move  
**את שאר הפעולות ניתן לראות בקוד עצמו**

 ModelMovesList	
  ModelMovesList(MoveGenerator, int)	
  uniqueMoves	HashMap<Integer, ArrayList<Move>>
  generationSettings	int
  generator	MoveGenerator
  rejectedPseudoLegal	ArrayList<Move>
  addAll(ModelMovesList, PieceType)	void
  isQuiescence(Move, PieceType)	boolean
  genMovingFromBB()	Bitboard
  add(Move, PieceType)	boolean
  prettyPrint()	void
...	

```
<Move>, Collection<Move>, List<Move>, RandomAccess
public class ModelMovesList
extends MovesList
Model moves list - represents a list of moves with a few features unique to the server side. calculating move
annotation is done using the Model. and MoveGenerator
```

**ModelMovesList**(MoveGenerator generator, int generationSettings)  
Instantiates a new Model moves list.

void initAnnotation()  
Initializes moves notation.

את שאר הפעולות ניתן לראות בקוד עצמו

 FoundLegalMove	
  FoundLegalMove()	

```
public class FoundLegalMove
extends Throwable
Found legal move. used for saving time when looking for any legal move.
```

PiecesBBs	
<b>PiecesBBs(int)</b>	
<b>size</b>	<i>int</i>
<b>prevAll</b>	Bitboard
<b>bitboards</b>	Bitboard[]
<b>getAll()</b>	Bitboard
<b>getBitboards()</b>	Bitboard[]
<b>getPieceType(Bitboard)</b>	PieceType
<b>getBB(PieceType)</b>	Bitboard
<b>toString()</b>	String

```
public class PiecesBBs
represents a collection of bitboards of pieces.

Bitboard getBB(PieceType pieceType)
Gets the bitboard of pieceType.

Parameters: pieceType - the piece type Returns: the bb
את שאר הפעולות ניתן לראות בקובץ עצמו
```

### Players

Player	
<b>Player(String)</b>	
<b>playerColor</b>	PlayerColor
<b>gameSession</b>	GameSession
<b>game</b>	Game
<b>username</b>	String
<b>partner</b>	Player
<b>createdGameID</b>	String
<b>isAi()</b>	<i>boolean</i>
<b>isSaveWorthy()</b>	<i>boolean</i>
<b>isGuest()</b>	<i>boolean</i>
...	

```
public abstract class Player
Player - represents a player capable of generating a selected move, respond to questions, and more.

Player(String id)
Instantiates a new Player.

error public abstract void error(String error)
alert the player of an Error.

Parameters: error - the error
getMove public abstract Move getMove()
ask the player to choose a move.
```

**Returns:** the chosen move

**waitTurn** public abstract void **waitTurn()**

Wait for your opponent to make his turn.

**gameOver** public abstract void **gameOver**(GameStatus gameStatus)  
notify player of a Game over.

**Parameters:** gameStatus - the game over status

**updateByMove** public abstract void **updateByMove**(Move move)  
notifies player of a change in the board. so he can Update his board.

**Parameters:** move - the move

**interrupt** public abstract void **interrupt**(MyError error)  
Interrupt a `getMove()` with an error.

**Parameters:** error - the error

את שאר הפעולות ניתן לראות בקוד עצמו

  PlayerNet	
   <b>PlayerNet</b> (AppSocket, LoginInfo, String)	
   <b>PlayerNet</b> (AppSocket, LoginInfo)	
   <b>loginInfo</b>	LoginInfo
   <b>profilePic</b>	String
   <b>socketToClient</b>	AppSocket
   <b>ID()</b>	String
   <b>updateByMove</b> (Move)	void
   <b>isGuest()</b>	boolean
   <b>disconnect</b> (String, boolean)	void
   <b>askQuestion</b> (Question, AnswerCallback)	void
...	

```
public class PlayerNet
extends Player
implements SyncableItem
Player net - represents a player connected to a client through an AppSocket.
```

את שאר הפעולות ניתן לראות בקוד עצמו

PlayerAI		
PlayerAI(AiParametrs)		
aiParameters	AiParametrs	
safetyNet		long
qNa	Map<QuestionType, Answer>	
moveSearchTimeout		long
error(String)		void
initGame(Game)		void
gameOver(GameStatus)		void
isAi()		boolean
createPlayerAi(AiParametrs)		PlayerAI
...		

```
public abstract class PlayerAI
extends Player
represents a player choosing moves using Artificial Intelligence .
```

את שאר הפעולות ניתן לראות בקוד עצמו

MyAi		
MyAi(AiParametrs)		
minimax	Minimax	
disconnect(String, boolean)	void	
gameOver(GameStatus)	void	
cancelQuestion(Question, String)	void	
getMove()	Move	
interrupt(MyError)	void	
initGame()	void	

```
public class MyAi
extends PlayerAI
My ai - represents an ai player using the Minimax algorithm to choose moves.
```

את שאר הפעולות ניתן לראות בקוד עצמו

 StockfishPlayer	
 StockfishPlayer(AiParamters)	
 stockfish	Stockfish
  initGame()	void
  cancelQuestion(Question, String)	void
  gameOver(GameStatus)	void
  disconnect(String, boolean)	void
  waitForMatch()	void
  updateByMove(Move)	void
  waitTurn()	void
  getMove()	Move
...	

```
public class StockfishPlayer
extends PlayerAI
represents a Stockfish Player, that always plays the best moves.
```

את שאר הפעולות ניתן לראות בקוד עצמו

 Stockfish	
 Stockfish(String)	
 Stockfish()	
  processReader	BufferedReader
  PATH	String
  engineProcess	Process
  processWriter	OutputStreamWriter
  setFen(String)	void
  perf(int)	StockfishPerft
  main(String[])	void
  getBestMove(String, int)	String
...	

```
public class Stockfish
an api for communicating with the world's highest rated chess bot. stockfish. the actual bot is saved in the
/assets/ folder. modified version of
```

`String getBestMove(String fen, int waitTime)`

This function returns the best move for a given position after calculating for 'waitTime' ms

**Parameters:** fen - Position string **waitTime** - in milliseconds **Returns:** Best Move in PGN format  
`float getEvalScore(String fen, int waitTime)`

Get the evaluation score of a given board position

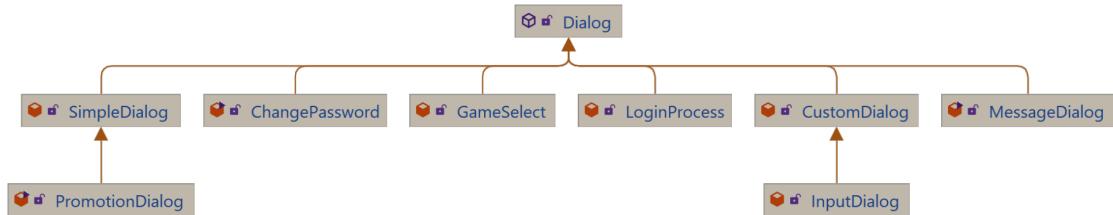
**Parameters:** fen - Position string **waitTime** - in milliseconds **Returns:** evalScore

את שאר הפעולות ניתן לראות בקוד עצמו

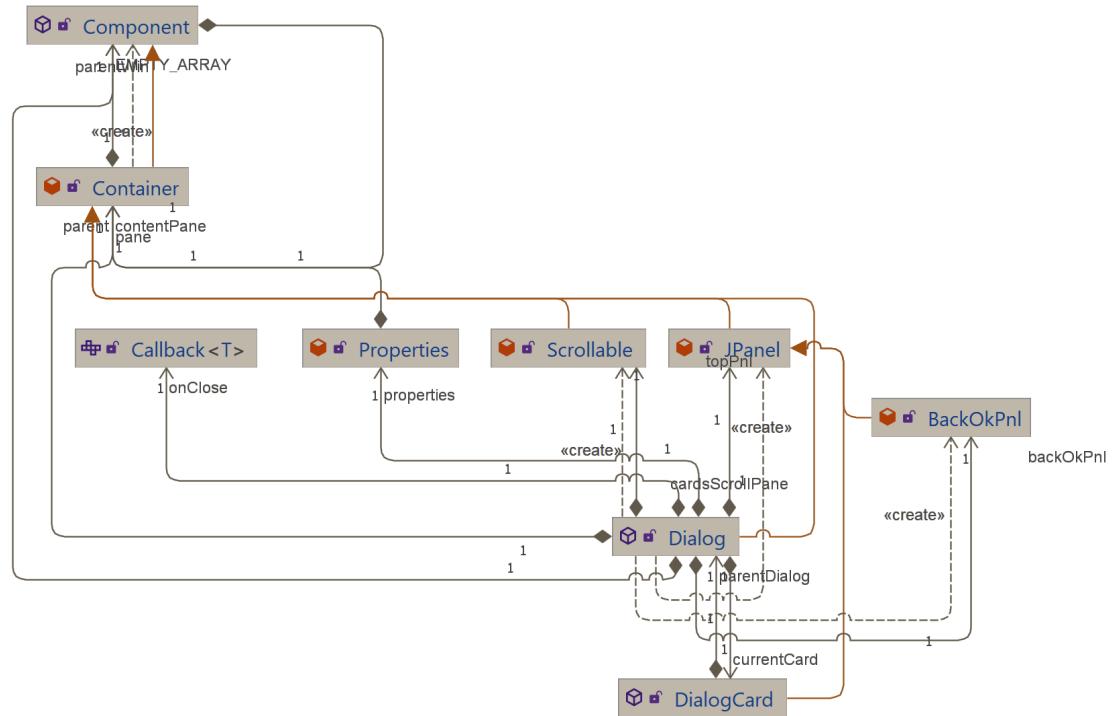
#### 4.4.3 תרשימים UML של המחלקות בצד הלוקוֹת

##### דיאלוגים:

תצוגה כללית של כל הדיאלוגים:

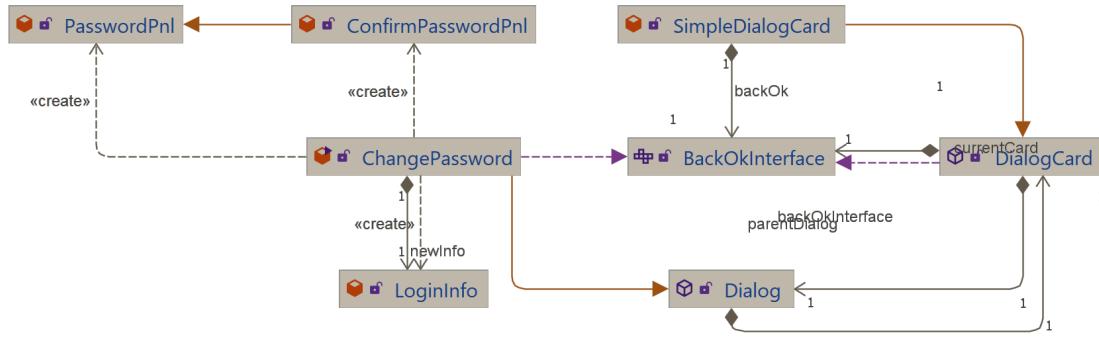


מחלקה דיאלוג:

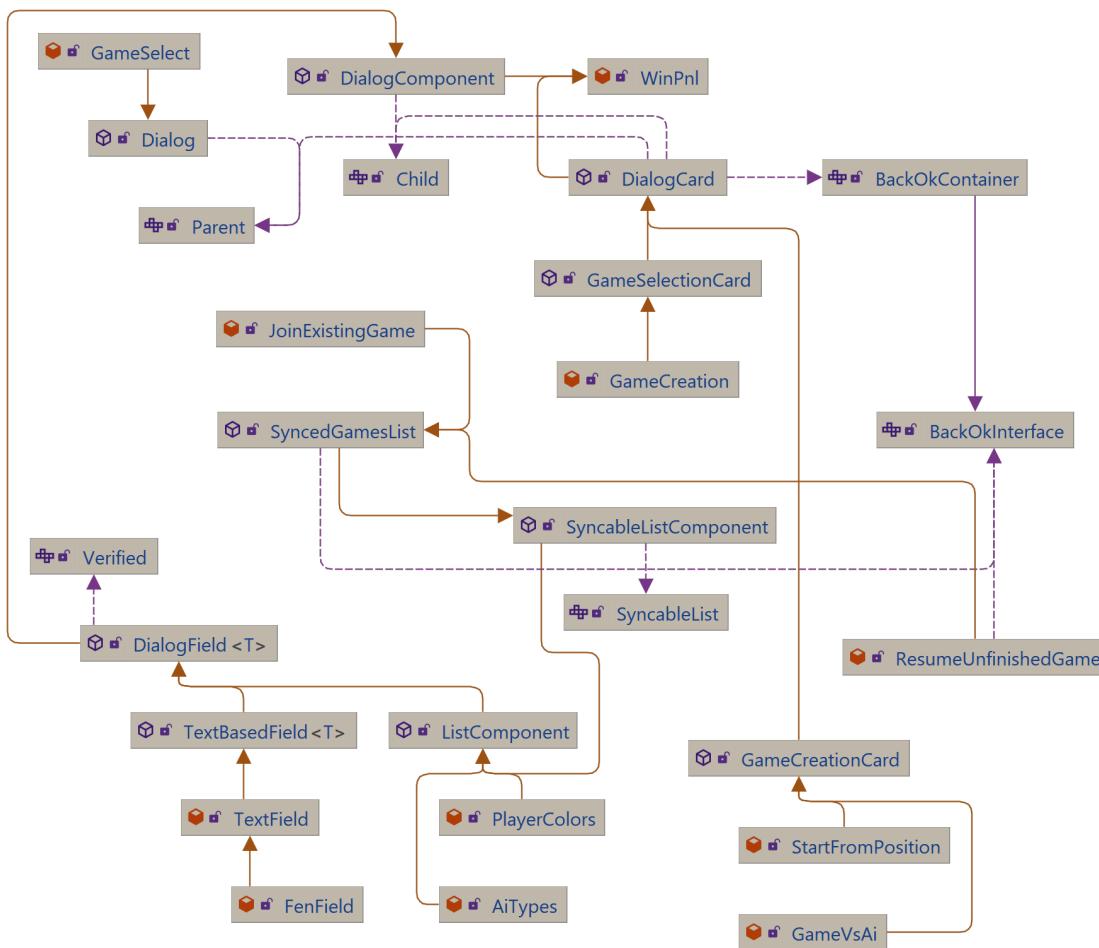


כל אחד מהדיאלוגים:

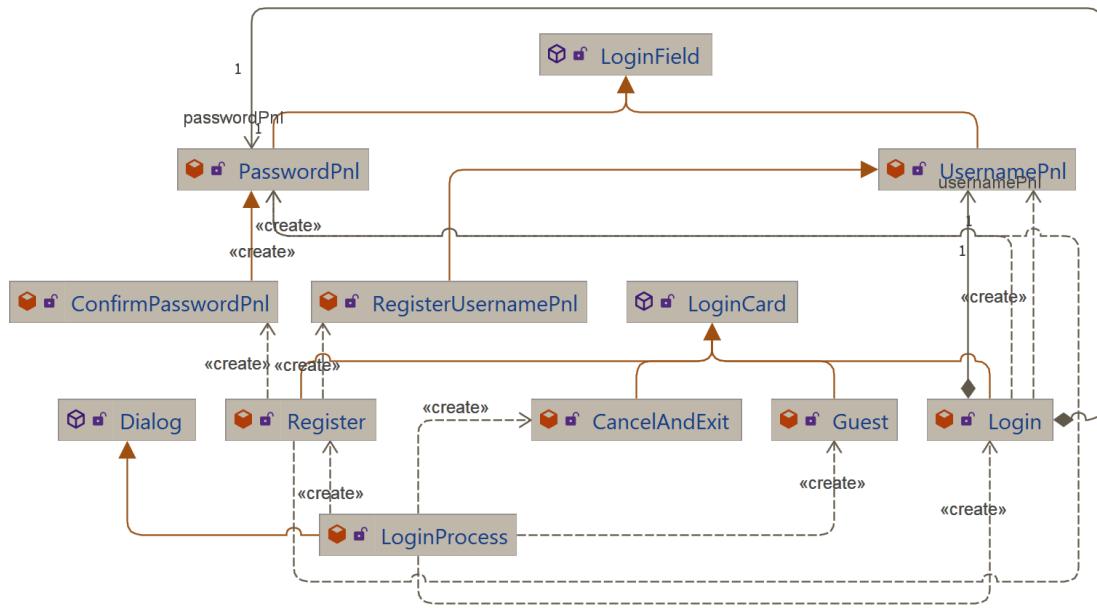
Change Password



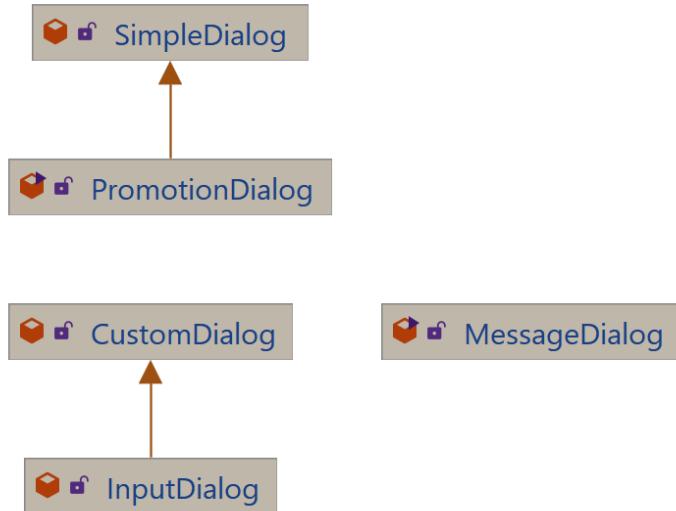
## Game Select



## Login Process

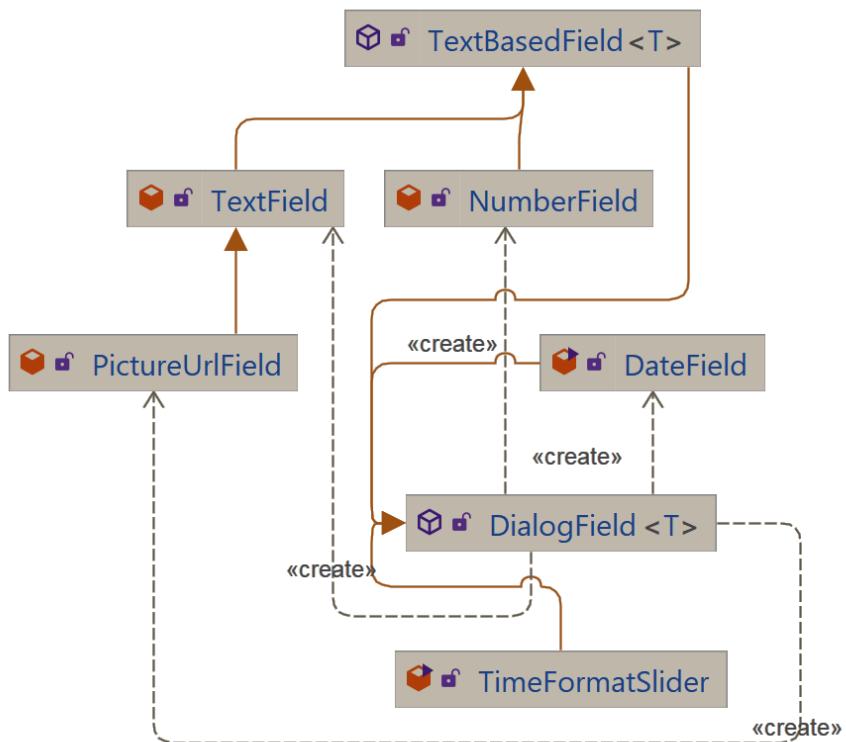


Simple Dialogs

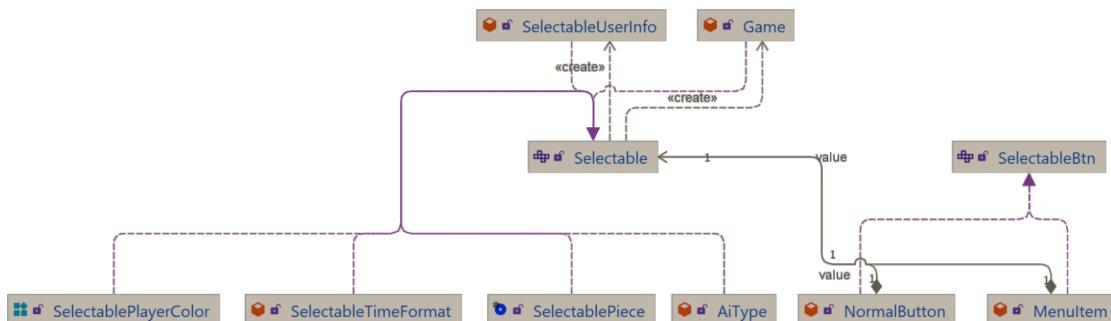


תכונות הדיאלוגים:

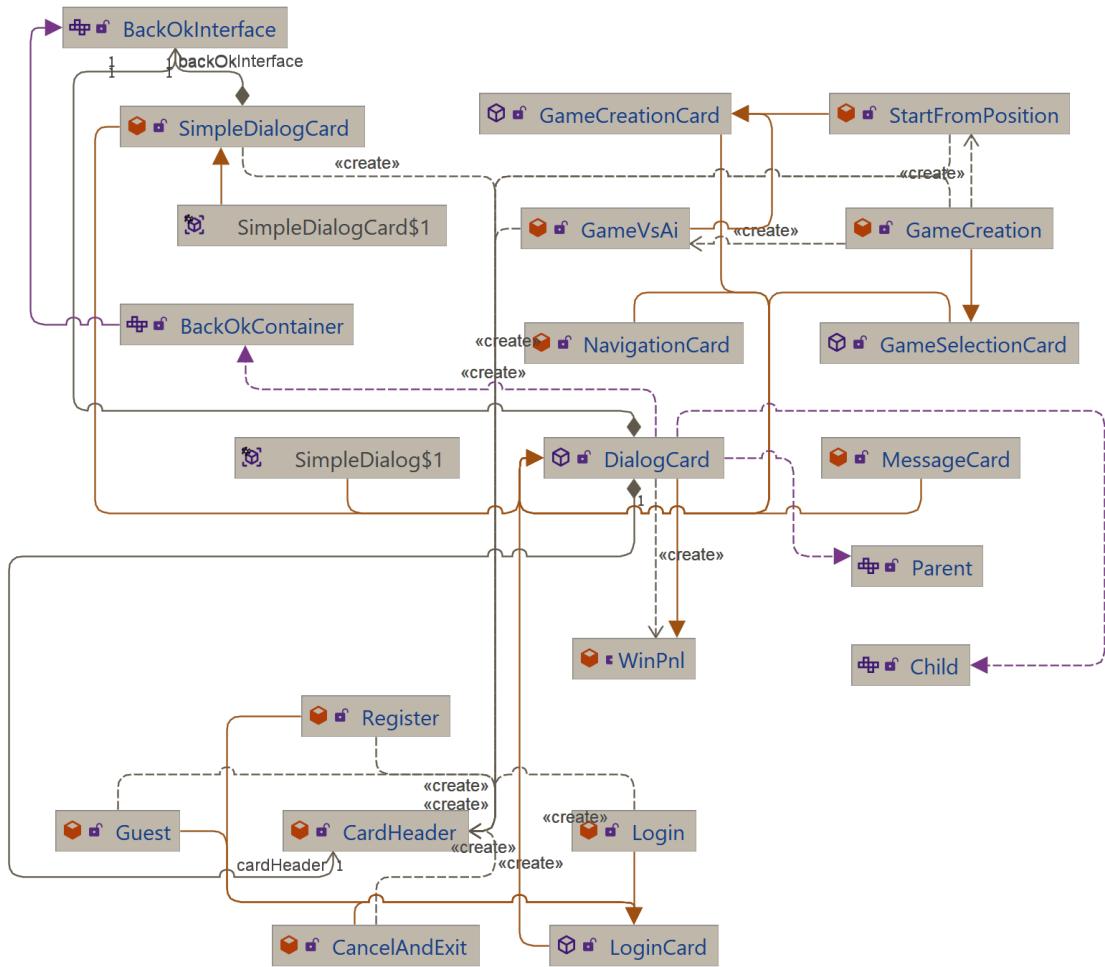
Fields



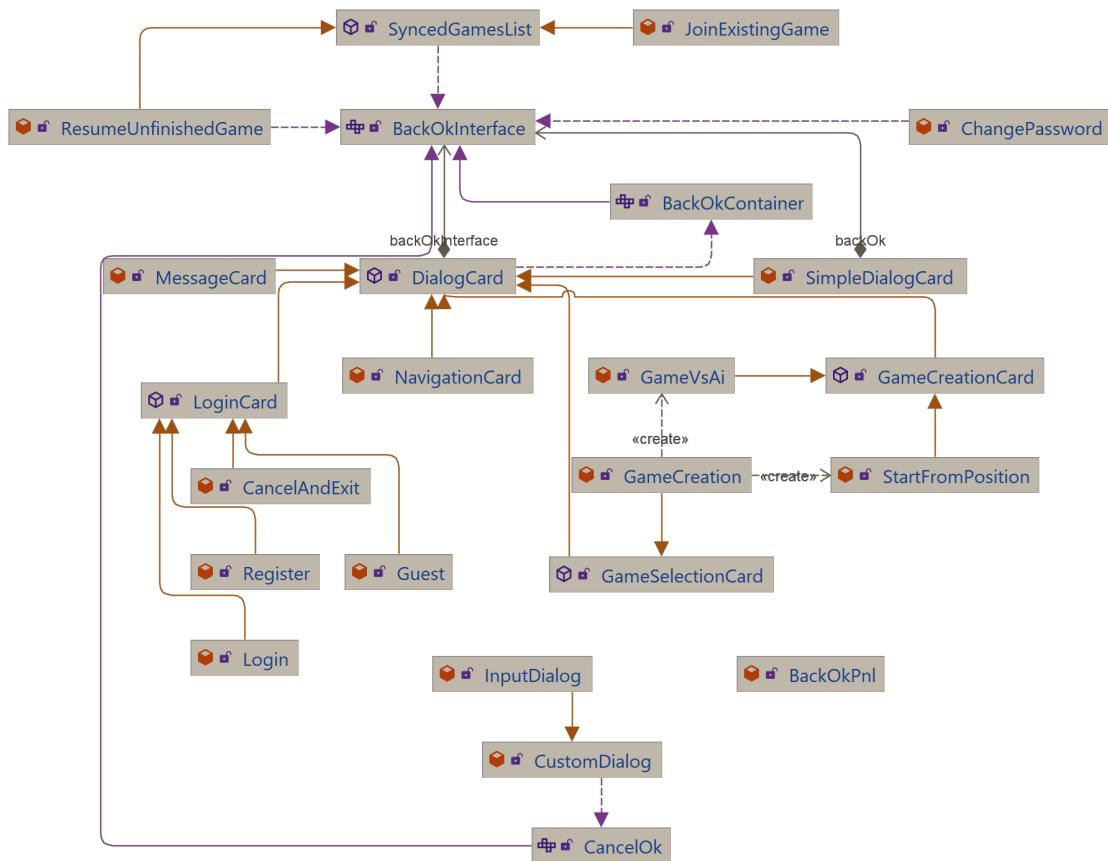
Selectables



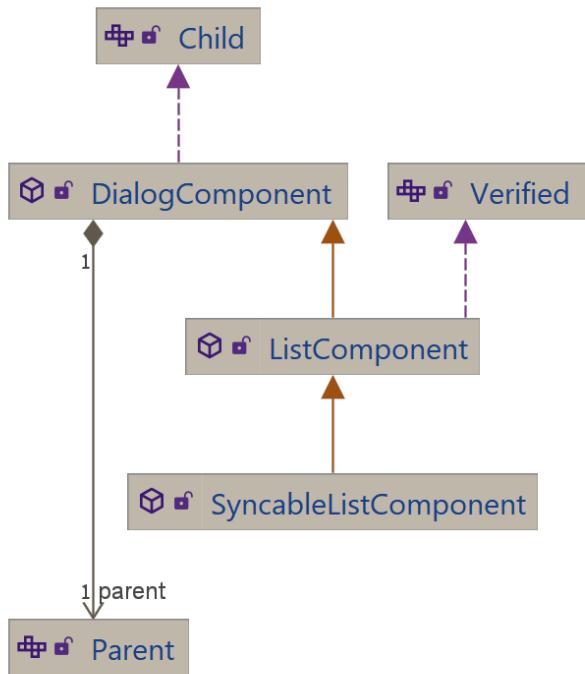
Cards



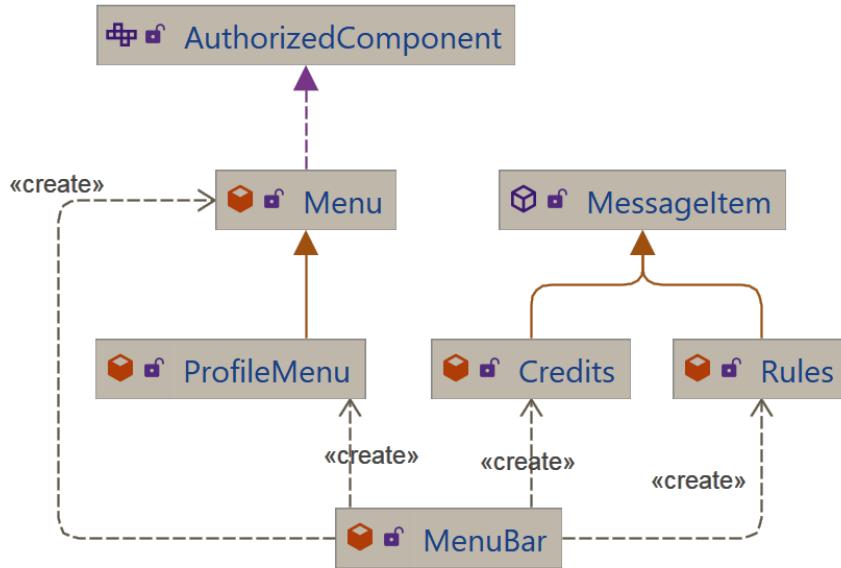
Back Ok



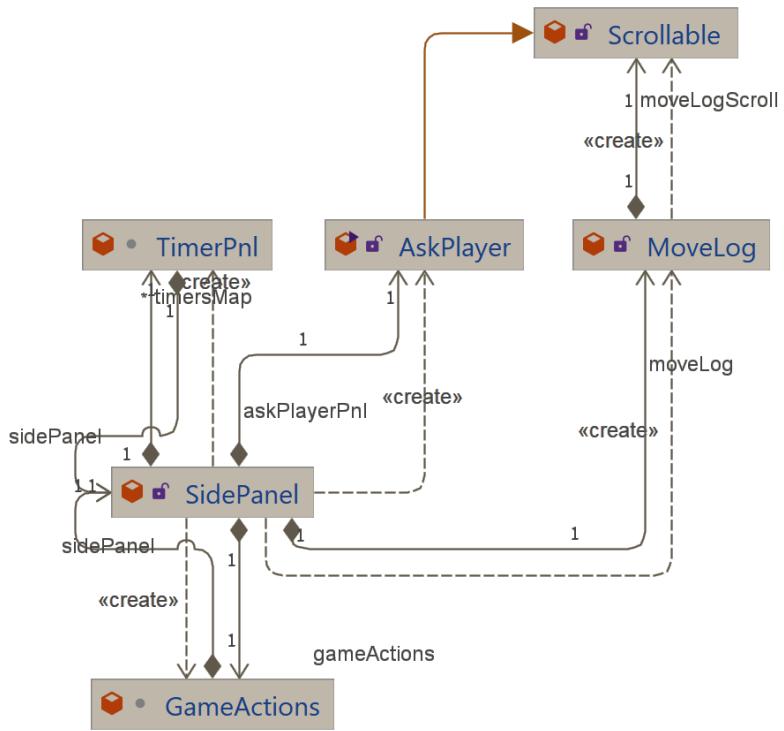
Components



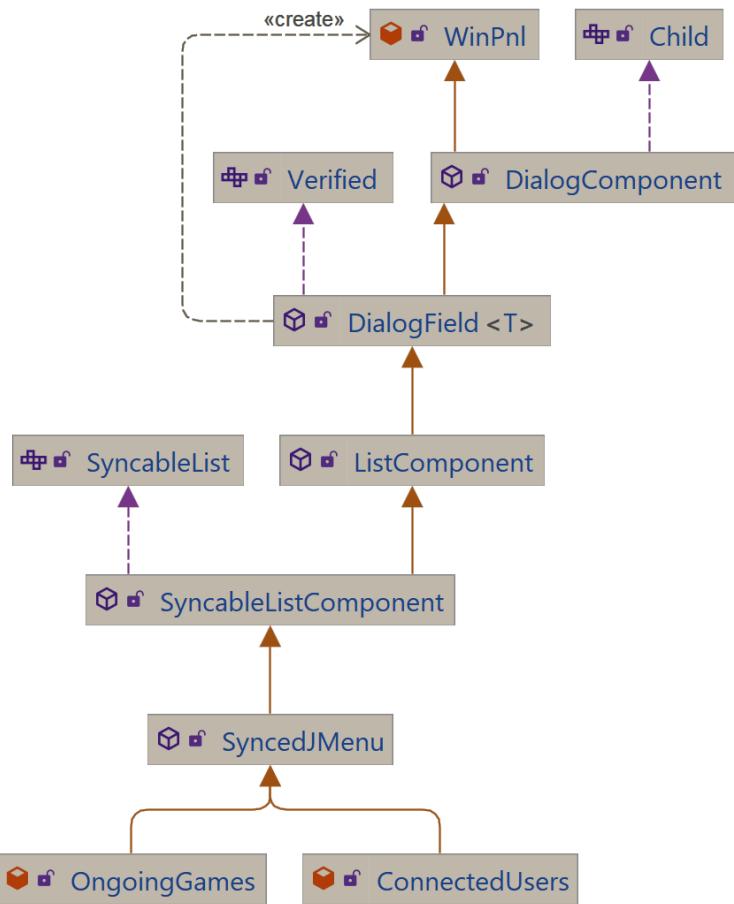
Menu Bar



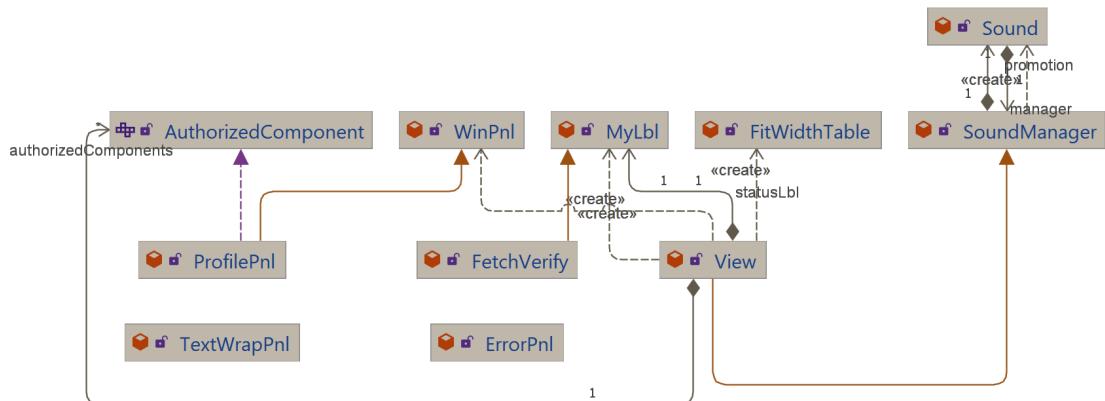
SidePanel



Synced JMenus



Misc



## תיעוד המחלקות בצד הלוקו

Client	
Client()	
START_FULLSCREEN	boolean
myColor	PlayerColor
firstClickLoc	ViewLocation
START_AT_ADDRESS	String
view	View
serverPort	int
clientSocket	AppSocket
teacherAddress	String
clientRunOK	boolean

```
public class Client
implements EnvManager
represents a chess Client that sets up the connection between the client and server and manages the View,
DBRequests, move selection, and a major part of the communication with the server through the
AppSocket.

Client()
Constructor for Chess Client.

void runClient()
start listening to messages from the server.

void boardButtonPressed(ViewLocation clickedLoc)
handle board button pressed.

Parameters: clickedLoc - the button's location
PieceType showPromotionDialog(PlayerColor clr)
Show promotion dialog and return the selected piece type.

Parameters: clr - the color pieces the player can choose from (white pieces for the white player...)
Returns: the chosen piece type
void request(RequestBuilder builder, Callback<DBResponse> onResponse,
Object... args)
send a db request to the server. RequestBuilders use arguments to create dynamic requests. some
arguments are expected to be filled by the player, and will show a dialog for getting that info, and some are
reliant on the environment to provide the value.

Parameters: builder - the request builder onResponse - a callback to call with a db response args - the
arguments for the request
את שאר הפעולות ניתן לראות בקוד עצמו
```

 ClientMessagesHandler	
 ClientMessagesHandler(Client, View)	
 view	View
 client	Client
 listeningLists	HashMap<SyncedListType, ArrayList<SyncableList>>
 syncedLists	HashMap<SyncedListType, SyncedItems<?>>
 onInitGame()	MessageCallback
 onUnplannedDisconnect()	void
 trySyncing(SyncableList)	void
 onLogin()	MessageCallback
 onGameOver()	MessageCallback
...	

```
public class ClientMessagesHandler
extends MessagesHandler
```

Client messages handler - represents a messages' handler that routes desired message types to their destinations.

**See Also:** MessagesHandler

```
void registerSyncableList(SyncableList list)
```

Register a list to be updated whenever a server message notifying of an update to that list type is received.

**Parameters:** list - the list to register

את שאר הפעולות ניתן לראות בקוד עצמו

 SoundManager	
 SoundManager()	
 tenSeconds	Sound
 soundEnabled	boolean
 promotion	Sound
 selfMove	Sound
 gameStart	Sound
 gameEnd	Sound
 capture	Sound
 castle	Sound
 check	Sound
...	

```
public class SoundManager
represents a Sound manager.
```

```
boolean isSoundEnabled()
```

Is sound enabled boolean.

**Returns:** the boolean

```
void setSoundEnabled(boolean soundEnabled)
```

Sets sound enabled.

**Parameters:** soundEnabled - the sound enabled

```
void moved(Move move, PlayerColor myClr)
play the right sound effect for the move played
```

**Parameters:** move - the move myClr - my clr

Sound	
Sound(String, SoundManager)	
manager	SoundManager
ASSETS_SOUND_EFFECTS	String
clip	Clip
playLoop()	void
stop()	void
play()	void

```
public class Sound
```

Sound - represents an audio clip that can be played on command.

```
void playLoop()
```

Play forever (loop).

```
void stop()
```

Stop playing.

```
void play()
```

Play once.

[View](#)

View	
View(Client)	
COLS	int
boardPnl	BoardPanel
sidePanel	SidePanel
winSize	Dimension
boardLock	Object
authorizedComponents	ArrayList<AuthorizedComponent>
CLIENT_WIN_TITLE	String
bottomPnl	JPanel
username	String

```
<BoardButton[]>
```

```
public class View
```

```
extends SoundManager
```

```
implements Iterable<BoardButton[]>
```

View - represents the GUI manager for the client. the view handles things like: showing and proper disposal of dialogs, the main game window, window resizes, and more..

```
public View(Client client)
```

Instantiates a new View.

**Parameters:** client - the client

```
void addListToRegister(SyncableList list)
```

Add list to register with the server once a connection has been made

**Parameters:** list - the list

```
void addAuthorizedComponent(AuthorizedComponent authorizedComponent)
```

Add an authorized component to the authorizedComponents list, so its state will change as the authorization status of the client is changed.

**Parameters:** authorizedComponent - the authorized component

```
void showMessage(String message, String title, MessageType messageType)
```

Show message.

**Parameters:** message - the message title - the title messageType - the message type

```
void authChange(LoginInfo loginInfo)
```

Auth change notify all the authorized components of a change in the authorization of the client

**Parameters:** loginInfo - the new login info

```
void initGame(GameTime gameTime, Board board, PlayerColor playerColor,
String otherPlayer)
```

Initialize the view for a new game.

**Parameters:** gameTime - the game time board - the board playerColor - this client's player color

otherPlayer - the other player's username

```
void askQuestion(Question question, AnswerCallback callback)
```

Ask the player a question.

**Parameters:** question - the question callback - callback to when the player selects one of the answers

```
void showDBResponse(DBResponse response, String respondingTo, String title)
```

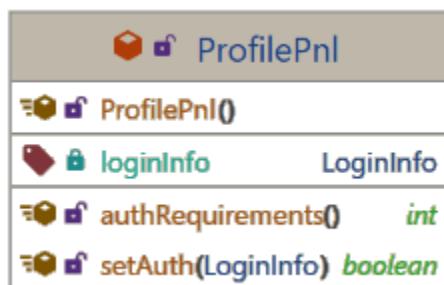
Show a db response from the server.

**Parameters:** response - the db response respondingTo - what is the server responding to title - the title

```
void dispose()
```

Dispose. closes the main window and all open dialogs

את שאר הפעולות ניתן לראות בקובץ עצמו



```
public class ProfilePnl
extends WinPnl
implements AuthorizedComponent
represents a Profile panel.
```

```
public ProfilePnl()
```

Instantiates a new Profile pnl.

את שאר הפעולות ניתן לראות בקובץ עצמו

MyTextArea	
MyTextArea(String)	
MyTextArea()	
textArea	JTextArea
scrollPane	JScrollPane
setWrap()	void
initializeUI()	void
setFont(Font)	void
setHeight(int)	void
setEditable(boolean)	void
setBackground(Color)	void
...	

```
public class FitWidthTable
extends JTable
represents a table that can fit its elements.
```

public FitWidthTable (@NotNull  
@NotNull Object[][] rowData, @NotNull  
@NotNull Object[] columnNames)

Instantiates a new Fit width table.

**Parameters:** rowData - the row data columnNames - the column names  
void **fit()**

calculates the optimal size for the contents of this table

את שאר הפעולות ניתן לראות בקוד עצמן

FetchVerify	
FetchVerify()	
FetchVerify(ImagenIcon, ImagenIcon, ImagenIcon)	
notVerifiedIcon	ImagenIcon
loadingIcon	ImagenIcon
verifiedIcon	ImagenIcon
startedLoading	ZonedDateTime
setIcon(Icon)	void
load()	void
nothing()	void
verify(boolean)	void
...	

```
public class FetchVerify
extends MyLbl
```

Fetch verify - represents a label with defined states: loading, nothing, verified and not verified. for each of the above states, the label has a predefined graphic to show with it. used to show the status of an online fetch.

את שאר הפעולות ניתן לראות בקוד עצמו

ErrorPnl	
ErrorPnl()	
ErrorPnl(String)	
jTextArea	JTextArea
emptyString	String
setText(String)	void
clear()	void
isEmpty()	boolean
setEmptyString(String)	void
getText()	String

```
public class ErrorPnl
extends JPanel
Error pnl - a panel that will show an error if one is found.
```

את שאר הפעולות ניתן לראות בקוד עצמו

Authorized Components

AuthorizedComponent	
setAuth(LoginInfo)	boolean
enableComp(boolean)	void
authRequirements()	int
auth(int)	boolean

```
public interface AuthorizedComponent
Authorized component - represents an object whose state depends on the authorization type of the client.
```

**setAuth** default boolean **setAuth**(LoginInfo loginInfo)  
updates the auth status of this client.

**Parameters:** loginInfo - the login info **Returns:** true if the login info provided satisfies this component's requirements. false otherwise.

**enableComp** default void **enableComp**(boolean e)  
enable/disable this component.

**Parameters:** e - did the current client's status satisfies the requirements  
int **authRequirements**()  
this component's authentication requirements

**Returns:** the int

את שאר הפעולות ניתן לראות בקוד עצמו

Menu		
Menu(String, int)		
authSettings	int	
childrenFont	Font	
add(String, VoidCallback)	JMenuItem	
authRequirements()	int	
setChildrenFont(Font)	void	
add(JMenuItem)	JMenuItem	

```
public class Menu
extends JMenuItem
implements AuthorizedComponent
Menu - represents an authorized jmenu.

Menu(String s, int authSettings)
Instantiates a new Menu with authSettings as the requirements
```

את שאר הפעולות ניתן לראות בקובץ עצמו

MenuItem		
MenuItem(String)		
MenuItem(String, int)		
authSettings	int	
authRequirements()	int	

```
public class MenuItem
extends JMenuItem
implements AuthorizedComponent
MenuItem - represents an authorized menu item.

MenuItem(String s)
Instantiates a new Menu item. with no authorization requirements

MenuItem(String s, int authSettings)
Instantiates a new Menu item with authSettings as the requirements.
```

את שאר הפעולות ניתן לראות בקובץ עצמו  
Board

BoardButton	
<b>BoardButton</b> (ViewLocation, MyColor, View)	
piece	Piece
wasUnlocked	boolean
view	View
iconSize	int
beforeLockBg	Color
ICON_MULTIPLIER	double
CHECK_COLOR	Color
statesCallbacks	Map<Integer, Callback<Graphics>>
hiddenIcon	Icon

```
public class BoardButton
extends MyJButton
represents a Board button.

boolean is(int state)
checks if this button's state is on.
```

**Parameters:** state - the state **Returns:** true if this state is on, false otherwise

```
void globalPaint(Graphics2D g2, Point mouseCoordinates, Component c)
allows a button to draw on the global, full board's 'canvas'
```

**Parameters:** g2 - the g 2 mouseCoordinates - the mouse coordinates c - the c

את שאר הפעולות ניתן לראות בקוד עצמו

@ State	
CHECK	int
PROMOTING	int
MOVING_FROM	int
CAN_MOVE_TO	int
HOVERED	int
SELECTED	int
DRAGGING	int
CAPTURE	int
MOVING_TO	int
CLICKED_ONCE	int

```
public @interface State
represents the different button States a board button can have.
```

	<b>ViewLocation</b>
	<b>ViewLocation(Location)</b>
	<b>viewLocation</b> Location
	<b>originalLocation</b> Location
	<b>toString()</b> String

```
public class ViewLocation
```

represents a location corrected for the shifted perspective of the view.

את שאר הפעולות ניתן לראות בקוד עצמו

	<b>ViewSavedBoard</b>
	<b>ViewSavedBoard(BoardPanel)</b>
	<b>savedSquares</b> ArrayList<SavedSquare>
	<b>disableAll()</b> void

<Square>

```
public class ViewSavedBoard  
extends Board
```

represents a capture of a view board's state. used for saving positions and scrolling through game logs.

**ViewSavedBoard(BoardPanel boardPanel)**

Instantiates a new View saved board.

את שאר הפעולות ניתן לראות בקוד עצמו

	<b>SavedSquare</b>
	<b>SavedSquare(BoardButton)</b>
	<b>isEnabled</b> boolean
	<b>piece</b> Piece
	<b>loc</b> ViewLocation
	<b>btnState</b> int
	<b>getLoc()</b> ViewLocation
	<b>setEnabled(boolean)</b> void
	<b>restore(BoardButton)</b> void

```
public class SavedSquare
```

represents a Saved square on the saved board.

את שאר הפעולות ניתן לראות בקוד עצמו

MyColor	
MyColor(Color)	
MyColor(int, int, int, int)	
MyColor(String)	
MyColor(int, int, int)	
movedClr()	Color

```
public class MyColor
extends Color
```

My color - represents a color that has a 'moved' property. used to highlight squares after a piece moved onto or from them .

```
Color movedClr()
Moved clr color.
```

**Returns:** the moved color

BoardPanel	
BoardPanel(int, int, View)	
whiteSquareClr	MyColor
boardOverlay	BoardOverlay
me	JPanel
buttonsPnl	JPanel
blackSquareClr	MyColor
rowsCoordinatesPnl	JPanel
btnMat	BoardButton[][]
colsCoordinatesPnl	JPanel
view	View

```
<BoardButton[]>, Accessible
public class BoardPanel
extends JPanel
implements Iterable<BoardButton[]>
represents the Board panel. holding all the BoardButtons.
```

```
BoardPanel(int rows, int cols, View view)
Instantiates a new Board panel.
```

```
void lock(boolean lock)
Lock all buttons.
```

**Parameters:** lock - the lock

```
void restoreBoardButtons(ViewSavedBoard savedBoard)
Restore board buttons.
```

**Parameters:** savedBoard - the saved board

```
getBtnMat public BoardButton[][] getBtnMat()
Get btn mat board button [ ][ ].
```

**Returns:** the board button [ ][ ]

את שאר הפעולות ניתן לראות בקוד עצמו

BoardOverlay	
BoardOverlay(View)	
arrows	ArrayList<Arrow>
view	View
pressedKey	Integer
currentBtn	BoardButton
startedAt	Point
mouseCoordinates	Point
keyClrMap	Map<Integer, Color>
NO_KEY	Integer
jlayer	JLayer<?>
...	

```
public class BoardOverlay
```

```
extends LayerUI<JPanel>
```

represents the Board's overlay. responsible for drawing arrows, detecting button clicks, and detecting held down buttons for selecting colors.

**BoardOverlay(View view)**

Instantiates a new Board overlay.

**Color currentColor()**

Current selected color. if no color is selected, the default color is returned.

**Returns:** the color

**Point centerPoint(Point point)**

Center point in the middle of a button.

**Parameters:** point - the point **Returns:** the point

**BoardButton getBtn(Point point)**

Gets a button that the given point is inside.

**Parameters:** point - the point **Returns:** the btn

**Location getLoc(Point point)**

converts a Point (x,y) to a Location

**Parameters:** point - the point **Returns:** the loc

את שאר הפעולות ניתן לראות בקובץ עצמו

• Arrow	
<b>• Arrow(Point, Point, Color)</b>	
• barb	<i>int</i>
• start	<i>Point</i>
• phi	<i>double</i>
• end	<i>Point</i>
• clr	<i>Color</i>
• draw(Graphics2D)	<i>void</i>
• setClr(Color)	<i>void</i>
• equals(Arrow)	<i>boolean</i>

```
class Arrow
```

represents an Arrow that can be painted on the screen.

```
Arrow(Point start, Point end, Color clr)
```

Instantiates a new Arrow.

```
void setClr(Color clr)
```

Sets clr.

**Parameters:** clr - the clr

```
void draw(Graphics2D g2)
```

Draws this arrow.

**Parameters:** g2 - the g 2

את שאר הפעולות ניתן לראות בקוד עצמו

### Dialog

• Dialog	
<b>• Dialog(DialogProperties)</b>	
• onCloseCallbacks	<i>ArrayList&lt;VoidCallback&gt;</i>
• backOkPnl	<i>BackOkPnl</i>
• onClose	<i>Callback&lt;Dialog&gt;</i>
• MAX_DIALOG_SIZE	<i>Size</i>
• cardsPnl	<i>JPanel</i>
• errorPnl	<i>ErrorPnl</i>
• DEFAULT_DIALOG_SIZE	<i>Size</i>
• cardStack	<i>Stack&lt;DialogCard&gt;</i>
• bottomPnl	<i>JPanel</i>

```
public abstract class Dialog
extends JDialog
implements Parent
```

my implementation of a Dialog. a dialog is made using dialog cards that are managed in a card layout. a dialog can communicate with the server directly using the provided AppSocket in the

`DialogProperties` the communication ability is mainly used for verifying availability of unique items. like when a new user registers, and needs to select a unique username.

**Dialog(DialogProperties dialogProperties)**  
Instantiates a new Dialog.

**void done()**  
the dialog is finished and should close.

**Specified by:** done in interface `Parent`

**void back()**  
the Back button has been clicked and the dialog should probably go to the previous card.

**Specified by:** back in interface `Parent`

**void scrollToTop()**  
Scroll to the top of the dialog.

**Specified by:** scrollToTop in interface `Parent`

**void verifyCurrentCard()**  
Verify the current card.

**See Also:** Verified

**void notifyClosed()**  
Notify all close listeners.

**void start(Callback<Dialog> onClose)**  
Start showing the dialog.

**Parameters:** onClose - a callback to call after closing the dialog

`NavigationCard navigationCardSetup(Size navCardSize, DialogCards dialogCards)`  
set the current card to as a navigation card for `dialogCards`.

**Parameters:** navCardSize - the nav card size `dialogCards` - the dialog cards **Returns:** the navigation card

**void cardsSetup(DialogCard startingCard, DialogCards dialogCards)**  
setup dialog with all its cards.

**Parameters:** startingCard - the starting card or null. if null, the first element in the array will replace it  
`dialogCards` - the dialog cards

את שאר הפעולות ניתן לראות בקוד עצמו

<b>DialogProperties</b>	
<b>DialogProperties(DialogDetails)</b>	
<b>DialogProperties(LoginInfo, AppSocket, MyJFrame, DialogDetail)</b>	
<b>loginInfo</b>	<code>LoginInfo</code>
<b>dialogDetails</b>	<code>DialogDetails</code>
<b>contentPane</b>	<code>Container</code>
<b>argConfig</b>	<code>Config&lt;?&gt;</code>
<b>socketToServer</b>	<code>AppSocket</code>
<b>parentWin</b>	<code>MyJFrame</code>
<b>toString()</b>	<code>String</code>
<b>setContentPane(Container)</b>	<code>void</code>
...	

`public class DialogProperties`

stores all kinds of important Dialog properties. like the current client's login info (if any), the socket to server so the dialogs can communicate directly with the server dialog details, and more.

את שאר הפעולות ניתן לראות בקוד עצמו

DialogDetails	
<code>DialogDetails(String[])</code>	
<code>DialogDetails (String, String, String)</code>	
<code>header</code>	String
<code>title</code>	String
<code>error</code>	String
<code>title()</code>	String
<code>error()</code>	String
<code>header()</code>	String

```
public record DialogDetails(String header, String title, String error)
```

represents a dynamic object with a few optional Dialog Details.

את שאר הפעולות ניתן לראות בקובץ עצמו

CanError<V>	
<code>errorDetails()</code>	String
<code>obj()</code>	V

```
public interface CanError<V>
```

represents a value that under some conditions might error. NOTE: the error detection is not done using this, only the details are acquired through it.

`V obj()`

get the value of this object.

**Returns:** the value of this object

`String errorDetails()`

get the Error details.

**Returns:** the error details

SimpleDialog	
<code>SimpleDialog(DialogProperties, Component[])</code>	
<code>SimpleDialog(DialogProperties, BackOkInterface, Component[])</code>	
<code>delayedSetup(BackOkInterface, Component[])</code>	<code>void</code>

```
public class SimpleDialog
```

extends Dialog

represents a Simple dialog, that can be created with simple components.

```
SimpleDialog(DialogProperties dialogProperties, BackOkInterface backOk,
Component... components)
```

Instantiates a new Simple dialog.

```
void setup(BackOkInterface backOk, Component... components)
setup this dialog.
```

**Parameters:** backOk - the back ok interface for this dialog components - the components for this dialog

Scrollable	
<code>Scrollable()</code>	
<code>Scrollable(JComponent, Dimension)</code>	
<code>Scrollable(JComponent)</code>	
<code>justScrolled</code>	<code>boolean</code>
<code>ogComp</code>	<code>JComponent</code>
<code>scrollToTop()</code>	<code>void</code>
<code>addComponent(Component, Object)</code>	<code>void</code>
<code>applyDefaults(JScrollPane, Dimension)</code>	<code>void</code>
<code>mySetSize(Dimension)</code>	<code>void</code>
<code>autoScrollToBottom()</code>	<code>ScrollNotifier</code>
<b>...</b>	

```
public class Scrollable
extends JScrollPane
my implementation of a Scrollable component.
```

`void scrollToTop()`  
Scroll to top.  
`void scrollTo(int position)`  
Scroll to.

**Parameters:** position - the position  
`void scrollToBottom()`  
Scroll to bottom.

את שאר הפעולות ניתן לראות בקובץ עצמו

SyncableList	
<code>syncedListType()</code>	<code>SyncedListType</code>
<code>sync(SyncedItems)</code>	<code>void</code>

```
public interface SyncableList
represents a Syncable list.
```

`SyncedListType syncedListType()`  
get this synced list's type.

**Returns:** the synced list type  
`void sync(SyncedItems items)`  
Sync this list with items.

**Parameters:** items - the items

WinPnl	
WinPnl(Header)	
WinPnl(int, Header)	
WinPnl()	
WinPnl(int)	
WinPnl(String, boolean)	
WinPnl(String)	
MAKE_SCROLLABLE	int
myInsets	Insets
insets	Insets
contentPnl	JComponent
...	

```
public class WinPnl
extends JPanel
```

Win pnl - my implementation of a panel. with a customizable layout that generally works similarly to a GridLayout, but allows for custom settings like a GridBagConstraints

void **setHeader**(Header header)

Sets header.

**Parameters:** header - the header

Component **add**(Component comp)

**Overrides:** add in class Container

void **add**(Component comp, GridBagConstraints gbc)

Add a component with custom settings.

**Parameters:** comp - the comp gbc - the gbc

**את שאר הפעולות ניתן לראות בקובץ עצמה**

[Back](#) [Ok](#)

BackOkInterface	
noInterface	BackOkInterface
getOkText()	String
onBack()	void
createSimpleInterface(VoidCallback) BackOkInterface	BackOkInterface
onOk()	void
getBackText()	String

```
public interface BackOkInterface
```

represents an object with navigation capabilities.

**createSimpleInterface** static BackOkInterface **createSimpleInterface**(VoidCallback onOk)

Create simple interface.

**Parameters:** onOk - the on ok **Returns:** the back ok interface

**getBackText** default String **getBackText()**

text for the back button. should return null for not creating the back button.

**Returns:** the text for the back button, or null for not creating it.

`getOkText default String getOkText()`

text for the ok button. should return null for not creating the ok button.

**Returns:** the text for the ok button, or null for not creating it.

`void onBack()`

called on click of the back button.

`void onOk()`

called on click of the ok button.



**All Superinterfaces:** BackOkInterface

public interface `CancelOk`

extends BackOkInterface

represents a cancel ok options navigation.

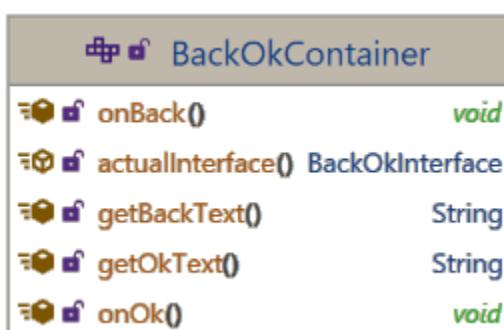
`getBackText default String getBackText()`

text for the back button. should return null for not creating the back button.

**Specified by:** `getBackText` in interface `BackOkInterface` **Returns:** the text for the back button, or null for not creating it.

`onCancel default void onCancel()`

On cancel.



**All Superinterfaces:** BackOkInterface

public interface `BackOkContainer`

extends BackOkInterface

represents a container for a Back ok interface.

`BackOkInterface actualInterface()`

Actual interface back ok interface.

**Returns:** the back ok interface

את שאר הפעולות ניתן לראות בקוד עצמו

BackOkPnl	
<b>BackOkPnl(BackOkInterface)</b>	
back	MyJButton
ok	MyJButton
enableOk(boolean)	void
createBtn(String, VoidCallback)	MyJButton
getOk()	MyJButton
enableBack(boolean)	void
getBack()	MyJButton

```
public class BackOkPnl
extends JPanel
represents a navigation panel for back ok navigating.
```

**BackOkPnl(BackOkInterface backOk)**  
Instantiates a new Back ok pnl.

את שאר הפעולות ניתן לראות בקובץ עצמו  
Cards

DialogCard	
<b>DialogCard(CardHeader, Dialog, BackOkInterface)</b>	
<b>DialogCard(CardHeader, Dialog)</b>	
cardID	String
parentDialog	Dialog
defaultValueBtns	ArrayList<MyJButton>
overrideableSize	boolean
navigationBtnsFont	Font
optionalHeader	Header
backOkInterface	BackOkInterface
navBtn	MyJButton
...	

```
public abstract class DialogCard
extends WinPnl
implements Child, Parent, AncestorListener, BackOkContainer
represents a Dialog card. a panel for displaying and managing DialogComponents.
```

**void navToMe()**

Navigate to this dialog card.

**String checkVerifiedComponents()**

Check verified components.

**Returns:** Error message if any not verified, null otherwise

**void displayed()**

called when this card gets Displayed.

**void askServer(Message msg, MessageCallback onRes)**

Send a message to the server.

**Specified by:** askServer in interface Parent **Parameters:** msg - the msg to send to the server onRes - the callback for when the server replies

```
void addDialogComponent(DialogComponent component)
Add dialog component.
```

**Parameters:** component - the component

את שאר הפעולות ניתן לראות בקובץ עצמו

MessageCard	
	MessageCard(Dialog, CardHeader, String, MessageType)
	createMsgPnl(String, MessageType, Size[]) MyTextArea
	displayed() void
	checkVerifiedComponents() String
	getBackText() String

```
public class MessageCard
extends DialogCard
```

Message card - represents a dialog card used for displaying messages.

את שאר הפעולות ניתן לראות בקובץ עצמו

MessageType	
	MessageType(ImageIcon, Font, Color)
	INFO
	font Font
	ServerError
	clr Color
	ERROR
	icon ImageIcon
	style(MyTextArea) void
	values() MessageType[]
	valueOf(String) MessageType

```
<MessageType>, Constable
public enum MessageType
```

Message type.

את שאר הפעולות ניתן לראות בקובץ עצמו

CardHeader	
CardHeader(String)	
CardHeader(ImageIcon, boolean, String)	
CardHeader(String, boolean, String)	
CardHeader(DialogProperties)	
CardHeader(String, boolean)	
CardHeader(String, ImageIcon, boolean, String)	
cardName	String
getCardName()	String
createHeader()	MyLbl

```
public class CardHeader
extends Header
represents a header for a dialog card.
```

את שאר הפעולות ניתן לראות בקוד עצמו

NavigationCard	
NavigationCard(CardHeader, Dialog, DialogCard[])	

```
public class NavigationCard
extends DialogCard
Navigation card - used for creating dialog cards that are just navigation cards to other cards.
NavigationCard(CardHeader cardHeader, Dialog parentDialog, linkTo)
Instantiates a new Navigation card.
```

SimpleDialogCard	
SimpleDialogCard(CardHeader, Dialog, BackOkInterface)	
backOk	BackOkInterface
onBack()	void
create(DialogComponent, Dialog)	SimpleDialogCard
create(CardHeader, Dialog, BackOkInterface, DialogComponent[])	SimpleDialogCard
onOk()	void

```
public class SimpleDialogCard
extends DialogCard
Simple dialog card - used for simple creation of a dialog card.
SimpleDialogCard create(CardHeader header, Dialog parent, BackOkInterface
backOk, components)
Create simple dialog card.
```

**Parameters:** header - the header parent - the parent backOk - the back ok components - the

**Returns:** the simple dialog card

את שאר הפעולות ניתן לראות בקוד עצמו

Components

DialogComponent	
DialogComponent(int, Header, Parent)	
DialogComponent(Header, Parent)	
parent	Parent
parent()	Parent
onUpdate()	void
setParent(Parent)	void

```
public abstract class DialogComponent
extends WinPnl
implements Child
Dialog component - represents a dialog component.
```

את שאר הפעולות ניתן לראות בקוד עצמו

Parent	
currentCard()	DialogCard
tryCancel()	boolean
scrollToTop()	void
back()	void
dialogWideErr(String)	void
onUpdate()	void
registerSyncedList(SyncableList)	void
tryOk(boolean)	boolean
keyAdapter()	MyAdapter
askServer(Message, MessageCallback)	void
***	

```
public interface Parent
Parent - represents a parent of components.

keyAdapter default MyAdapter keyAdapter()
Key adapter - gets the key adapter from the parent.

Returns: the adapter
void registerSyncedList(SyncableList list)
Register a synced list with the server.
```

**Parameters:** list - the list to register  
**void askServer(Message msg, MessageCallback onRes)**  
Send a message to the server.

**Parameters:** msg - the msg to send to the server onRes - the callback for when the server replies  
**void onUpdate()**  
On update to the hierarchy.

**scrollToTop** default void scrollToTop()  
Scroll to the top of the dialog.

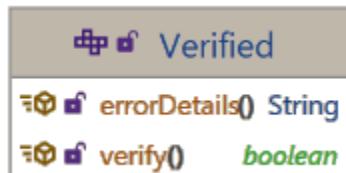
DialogCard **currentCard()**  
get the currently displayed dialog card

**Returns:** the currently displayed dialog card  
את שאר הפעולות ניתן לראות בקוד עצמו



public interface **Child**  
Child - represents an object with a parent.  
**Parent parent()**  
Parent.

**Returns:** the parent

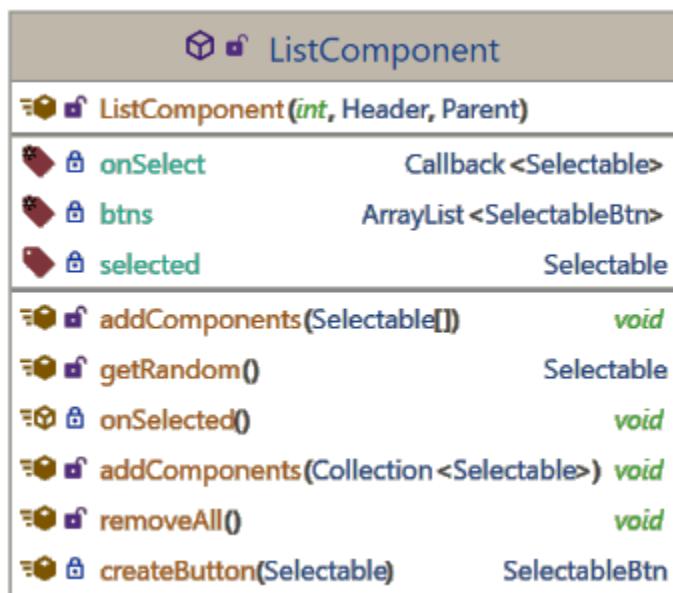


public interface **Verified**  
Verified - represents a verified component that according to the situation, might not verify.  
**boolean verify()**  
Verify check the current conditions, and try to verify.

**Returns:** true if this component verified successfully, false otherwise.

**String errorDetails()**  
gets the error details of this component. will only be called after not verifying.

**Returns:** the error details.



public abstract class **ListComponent**  
extends DialogField<Selectable>  
List component - represents a list component.  
**void addComponents( components)**  
Add components.

**Parameters:** components - the components  
**void addComponent(Selectable item)**  
Add component.

**Parameters:** item - the item

`SelectableBtn createButton(Selectable item)`  
Create button selectable btn.

**Parameters:** item - the item **Returns:** the selectable btn  
את שאר הפעולות ניתן לראות בקוד עצמו

SyncableListComponent	
<code>SyncableListComponent(Header, SyncedListType, Parent)</code>	
<code>listSize</code>	Size
<code>listType</code>	SyncedListType
<code>listItemSize</code>	Size
<code>setParent(Parent)</code>	void
<code>onUpdate()</code>	void
<code>syncedListType()</code>	SyncedListType
<code>sync(SyncedItems)</code>	void
<code>canUseIcon()</code>	boolean

```
public abstract class SyncableListComponent
extends ListComponent
implements SyncableList
represents a Syncable list component. a list that can be registered to be synced with the server. after
registering, the server will keep this list up-to-date after every change.
```

`SyncableListComponent(Header header, SyncedListType listType, Parent parent)`  
Instantiates a new Syncable list component.

`SyncedListType syncedListType()`  
get this synced list's type.

**Specified by:** syncedListType in interface SyncableList **Returns:** the synced list type  
`void sync(SyncedItems items)`  
Sync this list with items.

**Specified by:** sync in interface SyncableList **Parameters:** items - the items  
`boolean canUseIcon()`  
can the components of this list have icons

**Returns:** true if the components of this list can use icons, false otherwise.  
את שאר הפעולות ניתן לראות בקוד עצמו

[Dialog Fields](#)

DialogField <T>	
DialogField(int, Header, Parent)	
DialogField(Header, Parent)	
notEqualsTo	ArrayList<CanError<T>>
errLbl	ErrorPnl
config	Config<T>
onDefaultClickOk	boolean
cols	int
noRes	boolean
addSecondaryComp(Component)	void
valueBtnPresses(T)	void
...	

**Type Parameters:** T - the type of info this field will hold

```
public abstract class DialogField<T>
```

```
extends DialogComponent
```

```
implements Verified
```

Dialog field - represents a dialog field for the client to fill.

**See Also:** Serialized Form

**createField** public static DialogField<?> **createField**(Arg arg, Parent fieldParent)

Create field according to an argument. meant to create the required information fields when creating database requests.

**Parameters:** arg - the arg **fieldParent** - the field parent **Returns:** the dialog field

**setConfig** public final void **setConfig**(Config<T> config)

Sets the configuration of this field.

**Parameters:** config - the config

**createValBtn** private ValueBtn<T> **createValBtn**(Described<T> desc)

Create value button. used for creating default buttons for fields that allow default values.

**Parameters:** desc - the desc **Returns:** the value btn

boolean **verify()**

Verify check the current conditions, and try to verify.

**Specified by:** verify in interface **Verified** **Returns:** true if this component verified successfully, false otherwise.

**getValue** protected abstract T **getValue()**

Gets the current value of this field.

**Returns:** the value

**verifyField** protected abstract boolean **verifyField()**

Verify field.

**Returns:** true if the field has verified successfully, false otherwise.

**setValue** public abstract void **setValue**(T value)

Sets value.

**Parameters:** value - the value

T **getResult()**

Gets the result of this field.

**Returns:** this field's value if it is verified, null otherwise.

**DialogCard** **createCard()**

Create card out of this field.

**Returns:** the dialog card

את שאר הפעולות ניתן לראות בקובץ עצמו

<b>DateField</b>	
<b>DateField(Header, Parent)</b>	
  <b>jCalendar</b>	JCalendar
  <b>before</b>	DateField
  <b>err</b>	String
  <b>after</b>	DateField
  <b>verifyField()</b>	boolean
  <b>getDate()</b>	Date
  <b>errorDetails()</b>	String
  <b>toString()</b>	String
  <b>getValue()</b>	Date
...	

```
public class DateField
extends DialogField<Date>
a Date field.
```

`Date getValue()`

Gets the current value of this field.

**Specified by:** getValue in class DialogField<Date> **Returns:** the value

`boolean verifyField()`

Verify field.

**Specified by:** verifyField in class DialogField<Date> **Returns:** true if the field has verified successfully, false otherwise.

`void setValue(Date value)`

Sets value.

**Specified by:** setValue in class DialogField<Date> **Parameters:** value - the value

`void setBefore(DateField before)`

Sets the selected date to be before the provided date field.

**Parameters:** before - the before

`void setAfter(DateField after)`

Sets the selected date to be after the provided date field.

**Parameters:** after - the after

`String errorDetails()`

gets the error details of this component. will only be called after not verifying.

**Returns:** the error details.

את שאר הפעולות ניתן לראות בקובץ עצמו

TimeFormatSlider	
<b>TimeFormatSlider(Parent, TimeFormatComponent)</b>	
timeLbl	MyLbl
maxInSec	int
slider	JSlider
minInSec	int
setToMinValue()	void
getSlider()	JSlider
verifyField()	boolean
errorDetails()	String
getValue()	TimeFormat
...	

```
public class TimeFormatSlider
extends DialogField<TimeFormat>
a Time format slider.
```

**TimeFormatSlider (Parent parent, TimeFormatComponent timeFormatComponent)**  
Instantiates a new Time format slider.

TimeFormat **getValue ()**

Gets value.

**Specified by:** getValue in class DialogField<TimeFormat> **Returns:** the value

void **setValue (TimeFormat value)**

Sets value.

**Specified by:** setValue in class DialogField<TimeFormat> **Parameters:** value - the value  
את שאר הפעולות ניתן לראות בקוד עצמו

TextBasedField <T>	
<b>TextBasedField(String, Parent, RegEx)</b>	
<b>TextBasedField(Header, Parent, RegEx)</b>	
defaultTextFieldSize	Dimension
textField	JTextField
verifyRegEx	RegEx
verifyField()	boolean
errorDetails()	String
createTextField()	JTextField
verifyRegEx()	boolean
styleTextField(JTextField)	JTextField
...	

```
public abstract class TextBasedField<T>
extends DialogField<T>
a Text based field.
```

**TextBasedField (String str, Parent parent, RegEx verifyRegEx)**

Instantiates a new Text based field.

```
TextBasedField(Header dialogLabel, Parent parent, RegEx verifyRegEx)
```

Instantiates a new Text based field.

```
String getTextFieldText()
```

Gets text field text.

**Returns:** the text field text

```
boolean verifyRegEx()
```

Verify this field by its regex limitations.

**Returns:** true if the field passed its verification, false otherwise.

את שאר הפעולות ניתן לראות בקוד עצמן

<b>TextField</b>	
<b>TextField(Header, Parent, RegEx)</b>	
<b>TextField(String, Parent, RegEx)</b>	
<b>getValue()</b>	<b>String</b>
<b>setValue(String)</b>	<b>void</b>

```
public class TextField  
extends TextBasedField<String>
```

**See Also:** Serialized Form

```
String getValue()
```

Gets value.

**Specified by:** `getValue` in class `DialogField<String>` **Returns:** the value

```
void setValue(String value)
```

Sets value.

**Specified by:** `setValue` in class `DialogField<String>` **Parameters:** value - the value

Dialogs

<b>Header</b>	
<b>Header(String, boolean)</b>	
<b>Header(Icon, boolean)</b>	
<b>Header(String)</b>	
<b>Header(String, Icon, boolean)</b>	
<b>maximumSize</b>	<b>Size</b>
<b>center</b>	<b>boolean</b>
<b>insets</b>	<b>Insets</b>
<b>lbl</b>	<b>MyLbl</b>
<b>icon</b>	<b>Icon</b>
<b>text</b>	<b>String</b>
...	

```
public class Header  
extends JPanel
```

Header - represents a header panel holding an icon, text, both, or none.

את שאר הפעולות ניתן לראות בקוד עצמן

Change Password

❖ ChangePassword	
❖ ChangePassword(DialogProperties, String)	
❖ newInfo	LoginInfo
❖ password	String
❖ onBack()	void
❖ onOk()	void
❖ getPassword()	String
❖ getBackText()	String

```
public class ChangePassword
extends Dialog
implements BackOkInterface
Change password dialog.
```

**ChangePassword(DialogProperties dialogProperties, String currentPassword)**  
Instantiates a new Change password.

**String getPassword()**  
Gets password.

**Returns:** the password  
את שאר הפעולות ניתן לראות בקוד עצמו  
Game Selection

❖ GameSelect	
❖ GameSelect(DialogProperties)	
❖ gameSettings	GameSettings
❖ getGameSettings()	GameSettings

```
public class GameSelect
extends Dialog
```

represents a Game selection dialog, when the client can create a custom game, join an existing game, and resume an unfinished game.

**GameSelect(DialogProperties dialogProperties)**  
Instantiates a new Game select.

**GameSettings getGameSettings()**  
Gets game settings.

**Returns:** the game settings  
Cards

❖ GameSelectionCard	
❖ GameSelectionCard(CardHeader, Dialog, GameSettings, GameType)	
❖ gameSettings	GameSettings
❖ gameType	GameType
❖ onOk()	void

```
public abstract class GameSelectionCard
extends DialogCard
a Game selection card.
```

**GameSelectionCard**(**CardHeader** cardHeader, **Dialog** parentDialog, **GameSettings** gameSettings, **GameType** gameType)  
Instantiates a new Game selection card.

את שאר הפעולות ניתן לראות בקוד עצמו

  JoinExistingGame	
  JoinExistingGame( <b>Parent</b> , <b>GameSettings</b> )	
  createCard()	DialogCard

```
public class JoinExistingGame
extends SyncedGamesList
represents a synchronized list of games that are available to join to.

JoinExistingGame(Parent parent, GameSettings gameSettings)
Instantiates a new Join existing game.

DialogCard createCard()
Create card out of this field.
```

**Overrides:** createCard in class DialogField<Selectable> **Returns:** the dialog card

  ResumeUnfinishedGame	
  ResumeUnfinishedGame( <b>Parent</b> , <b>GameSettings</b> )	
  getOkText()	String

```
public class ResumeUnfinishedGame
extends SyncedGamesList
implements BackOkInterface
Resume unfinished game - a list of all unfinished games this player can resume.

ResumeUnfinishedGame(Parent parent, GameSettings gameSettings)
Instantiates a new Resume unfinished game.
```

את שאר הפעולות ניתן לראות בקוד עצמו

  GameCreationCard	
  GameCreationCard( <b>CardHeader</b> , <b>Dialog</b> , <b>GameSettings</b> )	
  checkbox	Checkbox
  gameSettings	GameSettings
  navCols()	int
  setEnabledState( <b>boolean</b> )	void
  onOk()	void
  createNavPnl()	WinPnl
  onBack()	void

```
public abstract class GameCreationCard
extends DialogCard
represents a Game creation card.

GameCreationCard(CardHeader cardHeader, Dialog parentDialog, GameSettings gameSettings)
Instantiates a new Game creation card.
```

**setEnabledState** protected abstract void **setEnabledState**(**boolean** state)  
set should use the value in this card.

**Parameters:** state - the state. if true the value from this card will be used  
את שאר הפעולות ניתן לראות בקוד עצמו

GameVsAi	
<b>GameVsAi(Dialog, GameSettings)</b>	
<b>aiParameters</b>	AiParameters
<b>setEnabledState(boolean)</b>	void

```
public class GameVsAi
extends GameCreationCard
Game vs ai card.
```

את שאר הפעולות ניתן לראות בקוד עצמו

GameCreation	
<b>GameCreation(Dialog, GameSettings)</b>	

```
public class GameCreation
extends GameSelectionCard
Game creation card.

GameCreation(Dialog parentDialog, GameSettings gameSettings)
Instantiates a new Game creation.
```

StartFromPosition	
<b>StartFromPosition(Dialog, GameSettings)</b>	
<b>iconLbl</b>	JLabel
<b>getPreferredSize()</b>	Dimension
<b>setEnabledState(boolean)</b>	void
<b>onUpdate()</b>	void

```
public class StartFromPosition
extends GameCreationCard
Start from position - start from a custom position.
```

את שאר הפעולות ניתן לראות בקוד עצמו

Position	
Position(String)	
Position(String, String)	
StartingPosition	
Promotion	
fen	String
M1	
name	String
QueenGambit	
QueenVsPawn	
values()	Position[]
...	

```
<Position>, Constable
public enum Position
```

Position - represents pre-made default positions.

את שאר הפעולות ניתן לראות בקוד עצמו  
Components

AiTypes	
AiTypes(Parent, AiParameters)	
aiParameters	AiParameters
onSelected()	void

```
public class AiTypes
extends ListComponent
Ai types.

AiTypes (Parent parent, AiParameters aiParameters)
Instantiates a new Ai types.
```

את שאר הפעולות ניתן לראות בקוד עצמו

FenField	
FenField(Parent, GameSettings)	
gameSettings	GameSettings
setFen(String)	void
errorDetails()	String
onUpdate()	void

```
public class FenField
extends TextField
a Fen field. a FEN is a string representation of a chess position.

void setFen(String fen)
Sets fen.
```

**Parameters:** fen - the fen  
את שאר הפעולות ניתן לראות בקוד עצמו

PlayerColors	
<b>PlayerColors(Dialog, GameSettings)</b>	
<b>gameSettings</b>	GameSettings
<b>onSelected()</b>	void

```
public class PlayerColors
extends ListComponent
Player colors.

PlayerColors(Dialog parent, GameSettings gameSettings)
Instantiates a new Player colors.
```

את שאר הפעולות ניתן לראות בקוד עצמו

SyncedGamesList	
<b>SyncedGamesList(Header, SyncedListType, Parent, GameSettings,</b>	
<b>gameSettings</b>	GameSettings
<b>gameType</b>	GameType
<b>onBack()</b>	void
<b>onSelected()</b>	void
<b>onOk()</b>	void

```
public abstract class SyncedGamesList
extends SyncableListComponent
implements BackOkInterface
represents a Synchronized list of games.
```

את שאר הפעולות ניתן לראות בקוד עצמו  
Login Process

LoginProcess	
<b>LoginProcess(DialogProperties)</b>	
<b>loginInfo</b>	LoginInfo
<b>getLoginInfo()</b>	LoginInfo

```
public class LoginProcess
extends Dialog
represents the dialog for the Login process.
```

**LoginInfo getLoginInfo()**  
Gets the chosen login info.

**Returns:** the chosen login info  
Cards

LoginCard	
<b>LoginCard(CardHeader, Dialog, LoginInfo, LoginType)</b>	
loginInfo	LoginInfo
loginType	LoginType
onOk()	void

```
public abstract class LoginCard
extends DialogCard
represents a Login card.

LoginCard(CardHeader cardHeader, Dialog parentDialog, LoginInfo loginInfo,
LoginType loginType)
Instantiates a new Login card.
```

את שאר הפעולות ניתן לראות בקוד עצמו

CancelAndExit	
<b>CancelAndExit(Dialog, LoginInfo)</b>	
navToMe()	void

```
public class CancelAndExit
extends LoginCard
Cancel and exit card.

void navToMe()
Navigate to this dialog card.
```

**Overrides:** navToMe in class DialogCard

Guest	
<b>Guest(Dialog, LoginInfo)</b>	
navToMe()	void

```
public class Guest
extends LoginCard
Guest login.

void navToMe()
Navigate to this dialog card.

Overrides: navToMe in class DialogCard
```

Login	
Login(LoginProcess, LoginInfo)	
usernamePnl	UsernamePnl
removeAdapters ArrayList<Set<Integer>>	
passwordPnl	PasswordPnl
refreshValues(LoginProcess)	void
displayed()	void
onBack()	void

```
public class Login
extends LoginCard
represents the normal Login card. enables a login with a username and a password.
```

את שאר הפעולות ניתן לראות בקובץ עצמו

Register	
Register(LoginProcess, LoginInfo)	

```
public class Register
extends LoginCard
represents a Register card.
```

#### Components

LoginField	
LoginField(String, RegEx, LoginInfo, Parent)	
LoginField(Header, RegEx, LoginInfo, Parent)	
loginInfo	LoginInfo

```
public abstract class LoginField
extends TextField
a Login field.
```

UsernamePnl	
UsernamePnl(boolean, LoginInfo, Parent)	
onUpdate()	void

```
public class UsernamePnl
extends LoginField
represents a Username field.
```

את שאר הפעולות ניתן לראות בקובץ עצמו

RegisterUsernamePnl	
RegisterUsernamePnl(LoginInfo, Parent)	
isLoading	boolean
availabilityMap	Map<String, Boolean>
suggestionsMap	Map<String, ArrayList<String>>
lastCheckTime	ZonedDateTime
suggestionsPnl	WinPnl
lastCheckedStr	String
minLoadTime	int
fetchVerifyPnl	FetchVerify
executor	ExecutorService

public class RegisterUsernamePnl  
 extends UsernamePnl  
 Register username field. a field that after verifying the regex requirements for usernames, checks if the username is available.

boolean **verifyField()**  
 Verify field.

**Overrides:** verifyField in class TextBasedField<String> **Returns:** true if the field has verified successfully, false otherwise.

boolean **checkAvailable()**

Check if the current username is available on the server.

**Returns:** true if the username is available, false otherwise.

void **setSuggestions**(ArrayList<String> suggestions)

Sets username suggestions.

**Parameters:** suggestions - the suggestions

void **fetch**(String un)

Fetch result from the server.

**Parameters:** un - the un

את שאר הפעולות ניתן לראות בקוד עצמו

  PasswordPnl	
  PasswordPnl(boolean, LoginInfo, Parent)	
  PasswordPnl(String, boolean, LoginInfo, Parent)	
  showPasswordBtn	MyJButton
  iconRatio	double
  setForeground(Color)	void
  setBtnIcon(boolean)	void
  getPasswordField()	JPasswordField
  onUpdate()	void
  getPassword()	String
  showText()	void
...	
public class PasswordPnl extends LoginField represents a Password field.	
void showText()	
Show text.	
void hideText()	
Hide text.	
String getPassword()	
Gets password.	
<b>Returns:</b> the password	
את שאר הפעולות ניתן לראות בקוד עצמו	

  ConfirmPasswordPnl	
  ConfirmPasswordPnl (String, LoginInfo, Supplier<String>, Parent)	
  ConfirmPasswordPnl (LoginInfo, Supplier<String>, Parent)	
  matchWith	Supplier<String>
  noMatchErr	String
  isMatching()	boolean
  verifyRegEx()	boolean
  errorDetails()	String
  setMatchWith(Supplier<String>, String)	void

public class ConfirmPasswordPnl  
extends PasswordPnl  
Confirm password field. a field for repeating a password for confirmation.  
boolean isMatching()  
Is matching with the actual password field.

**Returns:** true if the password is matching with the other field. false otherwise.

    את שאר הפעולות ניתן לראות בקוד עצמו

Promotion

	Promotion
	Promotion(PlayerColor)
	list
	getResult()
	Piece

```
public class Promotion
extends SimpleDialog
```

represents a Promotion dialog for choosing a piece to promote to.

```
Promotion(PlayerColor playerColor)
```

Instantiates a new Promotion dialog.

```
Piece getResult()
```

Gets the dialog's result.

**Returns:** the chosen piece type

	PromotionList
	PromotionList(Promotion, PlayerColor)
	promotion
	Promotion
	onSelected()
	void

```
public class PromotionList
extends ListComponent
```

represents a list of all the pieces a player can promote to.

```
PromotionList(Promotion promotion, PlayerColor playerColor)
```

Instantiates a new Promotion list.

את שאר הפעולות ניתן לראות בקובץ עצמו

#### Other Dialogs

	CustomDialog
	CustomDialog(DialogProperties, Arg[])
	map Map<DialogField<?>, Integer>
	noRes boolean
	resultsArrSize int
	onBack()
	onOk()
	getResults()
	setup(ArrayList<DialogField<?>>) void
	setup(DialogField<?>[]) void

```
public class CustomDialog
```

```
extends Dialog
```

```
implements CancelOk
```

represents a Custom dialog, mostly used to get input for DBRequests. (like getting a date ranges for a games in range request).

```
CustomDialog(DialogProperties dialogProperties, args)
Instantiates a new Custom dialog with fields to satisfy the given args.
```

```
void setup(DialogField<?>... components)
initializes this Custom Dialog .
```

**Parameters:** components - the components

```
getResults public Object[] getResults()
```

Get the results with all the values from all the fields.

**Returns:** the results

את שאר הפעולות ניתן לראות בקוד עצמו

  InputDialog	
  <a href="#">InputDialog(DialogProperties)</a>	
  <a href="#">InputDialog(DialogProperties, ArgType)</a>	
  <a href="#">getInput()</a>	String
  <a href="#">getPreferredSize()</a>	Dimension

```
public class InputDialog
```

```
extends CustomDialog
```

represents a simple Input dialog.

```
InputDialog(DialogProperties dialogProperties)
```

Instantiates a new Input dialog.

```
InputDialog(DialogProperties dialogProperties, ArgType inputType)
```

Instantiates a new Input dialog.

```
String getInput()
```

Gets input.

**Returns:** the input

את שאר הפעולות ניתן לראות בקוד עצמו

  MessageDialog	
  <a href="#">MessageDialog(DialogProperties, String, String, MessageType)</a>	
  <a href="#">messageType</a>	MessageType
  <a href="#">onXClick()</a>	void
  <a href="#">getMessageType()</a>	MessageType

```
public class MessageDialog
```

```
extends Dialog
```

represents a Message dialog, used for displaying messages.

```
MessageType getMessageType()
```

Gets message type.

**Returns:** the message type

את שאר הפעולות ניתן לראות בקוד עצמו

### Selectables

  Selectable	
  <a href="#">getText()</a>	String
  <a href="#">getIcon()</a>	ImageIcon
  <a href="#">createSelectables(SyncedItems&lt;?&gt;, boolean)</a>	ArrayList<Selectable>

```
public interface Selectable
```

represents an object the client can select in some way (clicking, hovering...). a visual representation of an existing logical structure, like PlayerColor

```
createSelectables static ArrayList<Selectable> createSelectables (SyncedItems<?>
list, boolean canUseIcon)
```

Create a list of selectables from a list of SyncedItems. used to convert a list of logical items to a list of displayable objects.

**Parameters:** list - the list **canUseIcon** - the can use icon **Returns:** the collection

```
ImageIcon getIcon()
```

Gets the icon of this selectable.

**Returns:** null to show no icon, or the icon

```
String getText()
```

Gets the text of this selectable.

**Returns:** null / empty string to show no text, or the text

SelectableAiType	
<b>SelectableAiType(AiType)</b>	
aiType	AiType
aiType()	AiType
getText()	String
selectableAiTypes()	ArrayList<Selectable>
getIcon()	ImageIcon?

**Record Components:** aiType - The Ai type.

```
public record SelectableAiType (AiType aiType)
```

implements Selectable  
a selectable Ai type.

את שאר הפעולות ניתן לראות בקוד עצמו

SelectableGame	
<b>SelectableGame(GameInfo, Size)</b>	
gameInfo	GameInfo
txt	String
icon	ImageIcon
getIcon()	ImageIcon
ID()	String
getText()	String
getGameInfo()	GameInfo
equals(Object)	boolean

```
public class SelectableGame
implements Selectable, SyncableItem
a selectable Game.
```

את שאר הפעולות ניתן לראות בקוד עצמו

SelectablePlayerColor	
<b>SelectablePlayerColor(PlayerColor)</b>	
icon	ImageIcon
playerColor	PlayerColor
WHITE	
RANDOM	
BLACK	
values()	SelectablePlayerColor[]
valueOf(String)	SelectablePlayerColor
getIcon()	ImageIcon
getText()	String
...	

```
<SelectablePlayerColor>, Constable, Selectable
public enum SelectablePlayerColor
```

implements Selectable  
represents a Selectable player color.

את שאר הפעולות ניתן לראות בקוד עצמו

SelectableUserInfo	
<b>SelectableUserInfo(UserInfo)</b>	
userInfo	UserInfo
ID()	String
userInfo()	UserInfo
getIcon()	ImageIcon
getText()	String

**Record Components:** userInfo - The User info.  
public record SelectableUserInfo(UserInfo userInfo)

implements Selectable, SyncableItem  
represents a Selectable user info.

את שאר הפעולות ניתן לראות בקוד עצמו  
Selectable Buttons

SelectableBtn	
<b>comp()</b> Component	
getValue()	Selectable
select(boolean)	void

public interface SelectableBtn  
represents a button with a Selectable value.

```
Selectable getValue()  
Gets the selectable value of this button.
```

**Returns:** the selectable value of this button.  
**void select(boolean select)**  
set the selection state of this button.

**Parameters:** select - true to select this button and false to unselect it  
**comp default Component comp()**  
get the visual Component of this button.

**Returns:** the component

<b>NormalButton</b>	
<b>NormalButton(Selectable, Callback&lt;Selectable&gt;)</b>	
<b>selectedClr</b>	Color
<b>value</b>	Selectable
<b>selected</b>	boolean
<b>normalClr</b>	Color
<b>select()</b>	void
<b>getValue()</b>	Selectable
<b>select(boolean)</b>	void
<b>unselect()</b>	void

```
public class NormalButton  
extends MyJButton  
implements SelectableBtn  
represents a Normal selectable button.  
  
void select()  
Select this button.  
  
void unselect()  
Unselect this button.
```

את שאר הפעולות ניתן לראות בקוד עצמו

<b>MenuItem</b>	
<b>MenuItem(Selectable, Callback&lt;Selectable&gt;)</b>	
<b>normalClr</b>	Color
<b>value</b>	Selectable
<b>selected</b>	boolean
<b>selectedClr</b>	Color
<b>getValue()</b>	Selectable
<b>select()</b>	void
<b>select(boolean)</b>	void
<b>unselect()</b>	void

```
public class MenuItem  
extends JMenuItem
```

```

implements SelectableBtn
represents a Menu item with a selectable value.

MenuItem(Selectable value, Callback<Selectable> onSelect)
Instantiates a new Menu item.

void select()
Select.

void unselect()
Unselect.

את שאר הפעולות ניתן לראות בקובץ עצמן
Icon Manager

```

IconManager	
<b>IconManager()</b>	
 userIcon	MyImage
 TIE	int
 greenCheck	MyImage
 capturingIcon	MyImage
 LOGIN_PROCESS_SIZES	Size
 dynamicStatisticsIcon	DynamicIcon
 WON	int
 showPassword	MyImage
 dynamicSettingsIcon	DynamicIcon
...	

```

public class IconManager
Icon manager - utility class for loading icons.

ImageIcon copyImage(ImageIcon og)
Copy image.

```

**Parameters:** og - the original image **Returns:** the new copy  
**MyImage loadOnline(String path, \_size)**  
Load image from an online source.

**Parameters:** path - the path to the image **\_size** - the optional size of the image **Returns:** the loaded image if it loaded successfully. null otherwise  
**ImageIcon scaleImage(ImageIcon img, Dimension... \_size)**  
Scale an image.

**Parameters:** img - the image to scale **\_size** - the optional size. if one is not passed, the image will scale to 80 **Returns:** the image icon  
**MyImage loadImage(String relativePath, \_size)**  
Load an image from a relative path.

**Parameters:** relativePath - the relative path **\_size** - the optional size **Returns:** the loaded image  
**ImageIcon getPieceIcon(Piece piece)**  
Gets a piece icon.

**Parameters:** piece - the piece **Returns:** the piece icon  
את שאר הפעולות ניתן לראות בקובץ עצמן

  MyImage	
 	<code>MyImage(Image, String)</code>
 	<code>MyImage(URL)</code>
 	<code>MyImage(Imagen)</code>
 	<code>MyImage(URL, String)</code>
 	<code>MyImage(String, String)</code>
 	<code>paintInMiddle(Graphics, Component) void</code>

```
public class MyImage
extends ImageIcon
My implementation of an icon.
void paintInMiddle(Graphics g, Component c)
Paint this image in the middle of a Component.
```

**Parameters:** g - the graphics c - the component to paint in the middle of

  DynamicIcon	
 	<code>DynamicIcon(Imagen, Imagen)</code>
 	<code>DynamicIcon(String, Size)</code>
 	<code>defaultHoverFilename</code> String
 	<code>normal</code> Imagen
 	<code>defaultNormalFilename</code> String
 	<code>hover</code> Imagen
 	<code>getNormal()</code> Imagen
 	<code>set(AbstractButton)</code> void
 	<code>getHover()</code> Imagen

```
public class DynamicIcon
Dynamic icon - represents an icon that changes depending on hover.
void set(AbstractButton btn)
Set a button's icon to be this icon.
```

**Parameters:** btn - the button

את שאר הפעולות ניתן לראות בקוד עצמו

Size		
▪ Size(Dimension)		
▪ Size()		
▪ Size(int, int)		
▪ Size(int)		
▪ Size(Size)		
▪ DEFAULT_SIZE	int	
▪ maxDiff(Dimension)	int	
▪ padding(int)	Size	
▪ isValid()	boolean	
▪ createMinCombo(Dimension)	Dimension	
...		

**Direct Known Subclasses:** Size.RatioSize

```
public class Size
extends Dimension
Size - some general improvements to Dimension.
```

**See Also:** Serialized Form

את שאר הפעולות ניתן לראות בקוד עצמו

GameIconsGenerator		
▪ GameIconsGenerator()		
▪ maxFenLen	int	
▪ generate(String, PlayerColor, Dimension)	ImageIcon	

```
public class GameIconsGenerator
```

Game icons generator - utility class for generating icons of positions.

```
ImageIcon generate(String fen, PlayerColor orientation, Dimension iconSize)
Generate icon for the given position fen, from the orientation of the given color.
```

**Parameters:** fen - the fen orientation - the orientation iconSize - the icon size **Returns:** the image icon  
**MenuBar**

MenuBar		
▪ MenuBar(Client, View)		
▪ client	Client	
▪ menuFont	Font	
▪ menuItemsFont	Font	

```
public class MenuBar
```

```
extends JMenuBar
```

Menu bar - represents the menu bar of the main View window.

```
MenuBar(Client client, View view)
```

Instantiates a new Menu bar.

**RequestMenuItem**

**RequestMenuItem(PreMadeRequest, Client, Font)**

```
public class RequestMenuItem
extends MenuItem
represents a Request menu item, that when clicked, will send a PreMadeRequest to the server.
RequestMenuItem(PreMadeRequest preMadeRequest, Client client, Font font)
Instantiates a new Request menu item.
```

**ProfileMenu**

**ProfileMenu(Client, View)**

		menuFont	Font
		menuItemsFont	Font
		profilePnl	ProfilePnl

**setAuth(LoginInfo) boolean**

```
public class ProfileMenu
extends Menu
represents the profile menu. showing the username and profile pic, and when clicked will show all the
profile actions available to the current client.
ProfileMenu(Client client, View view)
Instantiates a new Profile menu.
```

את שאר הפעולות ניתן לראות בקוד עצמו

**MessageItem**

**MessageItem(String, Client)**

		client	Client
		msg()	String
		category(String, String[])	String
		onClick()	void
		title()	String

```
public abstract class MessageItem
extends JMenuItem
Message item - represents a message item that will show a static message on click.
MessageItem(String btnText, Client client)
Instantiates a new Message item.
```

את שאר הפעולות ניתן לראות בקוד עצמו

**Credits**

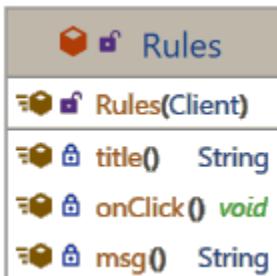
**Credits(Client)**

		title()	String
		msg()	String

```
public class Credits
```

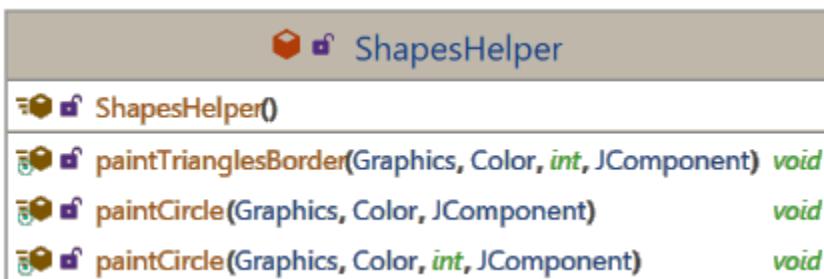
```
extends MessageItem
Credits message item.
```

את שאר הפעולות ניתן לראות בקוד עצמו



```
public class Rules
extends MessageItem
Rules message item.
```

את שאר הפעולות ניתן לראות בקוד עצמו



```
public class ShapesHelper
```

The utility class Shapes helper. can paint some shapes.

```
void paintTrianglesBorder(Graphics g, Color color, int size, JComponent component)
```

Paint a triangles border. paints a triangle at each of the component's corners. (naturally, if the provided component is a circle, your computer will blow up)

**Parameters:** g - the graphics color - the color of the triangles size - the size component - the component

```
void paintCircle(Graphics g, Color color, int diameterRatio, JComponent component)
```

Paint circle inside a component.

**Parameters:** g - the graphics color - the color diameterRatio - the diameter ratio: smaller is larger component - the component

את שאר הפעולות ניתן לראות בקוד עצמו

Side Panel

SidePanel		
SidePanel(boolean, Client)		
font	Font	
moveLog	MoveLog	
client	Client	
gameActions	GameActions	
askPlayerPnl	AskPlayer	
timeControl	long	
currentlyRunningClr	PlayerColor	
currentRunMillis	long	
timersMap	Map<PlayerColor, TimerPnl>	
...		

```
public class SidePanel
extends JPanel
```

Side panel - represents the side panel of the main view window. containing the TimerPnls, MoveLog and GameActions.

void namesSync()

sync the names on the timers to they're corresponding colors.

void initGame(PlayerColor myClr, String myUn, String oppUn, GameTime gameTime)

initialize the side panel for a new game.

**Parameters:** myClr - the color this client is playing as myUn - this client's username oppUn - the opponent's username gameTime - the game time

void sync(GameTime gameTime)

Sync timers.

**Parameters:** gameTime - the game time

void startRunning(PlayerColor clr)

Start running a player's clock.

**Parameters:** clr - the clr to start running

את שאר הפעולות ניתן לראות בקוד עצמו

GameActions		
GameActions(SidePanel)		
resignBtn	MyJButton	
offerDrawBtn	MyJButton	
sidePanel	SidePanel	
enableBtns(boolean)	void	
resignBtnClicked()	void	
offerDrawBtnClicked()	void	
enableDrawOfferBtn(boolean)	boolean	

```
public class GameActions
```

```

extends JPanel
Game actions - a panel where a player can execute game-related actions. like resigning.
GameActions(SidePanel sidePanel)
Instantiates a new Game actions.

void resignBtnClicked()
Resign btn clicked.

void offerDrawBtnClicked()
Offer draw btn clicked.

boolean enableDrawOfferBtn(boolean e)
Enable draw offer btn boolean.

```

**Parameters:** e - the e **Returns:** was the draw offer btn enabled before  
את שאר הפעולות ניתן לראות בקוד עצמו

MoveLog	
<b>MoveLog0</b>	
boardPanel	BoardPanel
forward	MyJButton
moveLogScroll	Scollable
currentMoveIndex	int
end	MyJButton
start	MyJButton
blackMoves	JPanel
movesBtns	ArrayList <MyJButton>
back	MyJButton

```

public class MoveLog
extends JPanel

```

Move log - represents a scrollable list of moves annotations in pgn format that gets built as the game progresses. at any point, the client can click on any of the previous moves in the game's history, and that position will be loaded.

**MoveLog()**  
Instantiates a new Move log.

void **forward()**  
go one move Forward if possible.

void **back()**  
go one move Backwards if possible.

KeyAdapter **createNavAdapter()**  
Create nav adapter for navigating moves with the arrow keys.

**Returns:** the key adapter

void **switchTo**(int index)  
Switch to a certain move.

**Parameters:** index - the index

void **scroll()**  
Scroll to the current move index.

את שאר הפעולות ניתן לראות בקוד עצמו

AskPlayer	
<b>AskPlayer()</b>	
justAdded	boolean
flashesDelay	int
content	WinPnl
shownQuestions	ArrayList<QuestionPnl>
noBorder	Border
numOfFlashesDone	int
borderThickness	int
currentCrlIndex	int
size	Size

```
public class AskPlayer
extends Scrollable
```

Ask player - represents a panel for asking the player questions. when a question is asked, it will show the question and flash to get the player's attention.

```
void ask(Question question, AnswerCallback callback)
Ask question.
```

**Parameters:** question - the question to ask callback - the callback to call once the player clicked an answer

```
void replaceWithMsg(Question replacing, String msg)
Replace a question with a message.
```

**Parameters:** replacing - the replacing msg - the msg

```
void removeQuestion(QuestionType questionType)
Remove question.
```

**Parameters:** questionType - the question type to remove

את שאר הפעולות ניתן לראות בקובץ עצמו

QuestionPnl	
<b>QuestionPnl(AskPlayer, Question, AnswerCallback)</b>	
question	Question
askPlayer	AskPlayer
createBtn(Answer, AnswerCallback)	MyJButton
getPreferredSize()	Dimension
setReplacement(String)	void
createBtn(Answer)	MyJButton

```
public class QuestionPnl
extends WinPnl
```

Question pnl - represents a single question panel.

```
QuestionPnl(AskPlayer askPlayer, Question question, AnswerCallback callback)
Instantiates a new Question pnl.
```

```
void setReplacement(String headerMsg)
Sets replacement for this question's message.
```

**Parameters:** headerMsg - the header msg  
את שאר הפעולות ניתן לראות בקוד עצמו

TimerPnl	
<b>TimerPnl(SidePanel, PlayerColor)</b>	
nameLbl	MyLbl
sidePanel	SidePanel
makeNoise	long
color	PlayerColor
timerLbl	MyLbl
tenSecondsClr	Color
played10s	boolean
<b>setTimer(long)</b>	void
<b>setName(String)</b>	void
...	

```
class TimerPnl
extends JPanel
Timer pnl - represents a panel with a timer label that will turn red as it gets to less than 10 seconds, and
will play a sound if this client has less than 10 seconds.
```

Instantiates a new Timer pnl.

**Parameters:** sidePanel - the side panel color - the PlayerColor assigned to this timer panel  
void **setName(String name)**

**Overrides:** `setName` in class `Component`

את שאר הפעולות ניתן לראות בקוד עצמו

Synced Jmenus

SyncedJMenu	
<b>SyncedJMenu(Header, SyncedListType)</b>	
<b>getMenu()</b>	JMenu
<b>createButton(Selectable)</b>	SelectableBtn
<b>add(Component)</b>	Component
<b>onSelected()</b>	void
<b>removeContentComponent(Component)</b>	void

```
public abstract class SyncedJMenu
extends SyncableListComponent
a synchronized jmenu.

public SyncedJMenu(Header header, SyncedListType listType)
Instantiates a new Synced j menu.
```

**Parameters:** header - the header listType - the list type  
את שאר הפעולות ניתן לראות בקוד עצמו

MyMenu	
MyMenu(Header)	
header	Header
done()	void
addOnClose(VoidCallback)	void
registerSyncedList(SyncableList)	void
currentCard()	DialogCard
onUpdate()	void
back()	void
backOkPnl()	BackOkPnl
askServer(Message, MessageCallback)	void
...	

```
class MyMenu
extends JMenu
implements Parent
My implementation of a jmenu used for a SyncedJMenu.

public MyMenu(Header header)
Instantiates a new Menu.
```

**Parameters:** header - the menu's header  
את שאר הפעולות ניתן לראות בקוד עצמו

OngoingGames	
OngoingGames()	
canUseIcon()	boolean

```
public class OngoingGames
extends SyncedJMenu
a synchronized list of all Ongoing games.

public OngoingGames()
Instantiates a new Ongoing games.
```

את שאר הפעולות ניתן לראות בקוד עצמו

ConnectedUsers	
ConnectedUsers()	
canUseIcon()	boolean

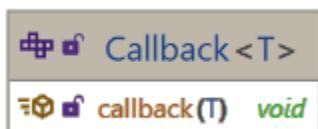
```
public class ConnectedUsers
extends SyncedJMenu
a synchronized list of all Connected users.

public ConnectedUsers()
Instantiates a new Connected users.
```

#### 4.4.5 תיעוד המחלקות המשותפות לקוות ולשרת

- המחלקה `Connection` מייצגת...
- המחלקה `Location` מייצגת...

##### Callbacks

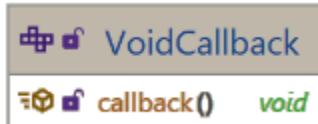


**Type Parameters:** `T` - the object's type

`public interface Callback<T>` `Callback` - represents an asynchronous callback with an object. some actions to execute at an unknown point in the future. used for things like button clicks.

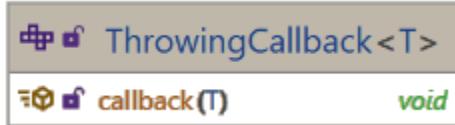
`void callback(T obj)` `Callback`.

**Parameters:** `obj` - the parameter of type `Callback`



`public interface VoidCallback` represents a callback with no object attached to it.

את שאר הפעולות ניתן לראות בקוד עצמו



**Type Parameters:** `T` - the callback type

`public interface ThrowingCallback<T>` represents a callback that might throw an exception .

```
void callback(T obj) Callback.
```

**Parameters:** obj - the obj **Throws:** Exception - the exception that might get thrown

### LazyHashSupplier<T>

**All Superinterfaces:** Serializable, Supplier<T>

```
public interface LazyHashSupplier<T>
```

extends Supplier<T>, Serializable represents a supplier for a hash that will later be fully calculated.

### MessageCallback

**All Superinterfaces:** Callback<Message>

```
public interface MessageCallback
```

extends Callback<Message> represents a message callback.

### DB Actions

#### Arg

Arg	
<code>Arg(ArgType, Config&lt;?&gt;)</code>	
<code>Arg(ArgType)</code>	
<code>Arg(ArgType, boolean, Config&lt;?&gt;)</code>	
<code>argType</code>	ArgType
<code>config</code>	Config<?>
<code>isUserInput</code>	boolean
<code>escape</code>	boolean
<code>ids</code>	IDsGenerator
<code>replnStr</code>	String
<code>toString()</code>	String
<code>...</code>	

```
public class Arg
```

implements Serializable represents an argument that will later be replaced with a value. used for creating db requests.

public Arg(ArgType argType, boolean escape, Config<?> config) Instantiates a new Arg.

**Parameters:** argType - the arg type escape - should this argument's value be escaped config - the config

את שאר הפעולות ניתן לראות בקוד עצמו

ArgType	
ArgType(boolean)	
ArgType()	
Password	
PictureUrl	
● isUserInput boolean	
ServerAddress	
Text	
Username	
DateRange	
Url	
...	

<ArgType>, Constable

```
public enum ArgType
```

argument type.

את שאר הפעולות ניתן לראות בקוד עצמו

Config<V>	
Config(String, Described<V>)	
Config(String, boolean, Described<V>)	
Config(String, V)	
Config(String, V, String)	
Config()	
Config(String)	
canUseDefault	boolean
defaultValue	Described<V>
description	String
valuesSuggestion	ArrayList<Described<V>>
...	

**Type Parameters:** v - the type of the argument

```
public class Config<v>
```

implements Serializable an Arg's configuration consisting of: a description: describing the argument's requirements. a default Described<v> value (optional) a list of Described<v> suggestions (optional)

**See Also:** Serialized Form

```
public Config(String description, Described<v> defaultValue) Instantiates a new Config.
```

**Parameters:** description - the description defaultValue - the default value

את שאר הפעולות ניתן לראות בקוד עצמו

Described<T>	
Described(T, String)	
Described(T)	
description	String
obj	T
description0	String
obj0	T

```
public record Described<T>(T obj, String description)
```

implements Serializable represents a described object of type Described.

```
public Described(T obj, String description) Creates an instance of a Described record class.
```

**Parameters:** obj - the value for the obj record component description - the value for the description record component

את שאר הפעולות ניתן לראות בקובץ עצמו

### DB Request

```
public class DBRequest
```

implements Serializable represents a database request. a db request can have a subRequest for requests that need to use the db separately, but are still contained under one request. like a summary line at the end of a games request. the summary is calculated on a separate db call after the main stat has finished, but both should show up as one to the client.

```
public DBRequest(SQLStatement sqlStatement) Instantiates a new Db request.
```

**Parameters:** sqlStatement - the sql statement

**DBRequest** **getSubRequest**() Gets sub request.

**Returns:** the sub request

```
String getRequest() Gets request.
```

**Returns:** the request

את שאר הפעולות ניתן לראות בקובץ עצמו

Type
Type0
Query
Update
valueOf(String) Type
values() Type[]

<Type>, Constable

```
public enum Type
```

represents a db request type.

```
private Type()
```

#### Enum Constant Details

##### Query

```
public static final
```

```
Type
```

##### Query

db query. retrieving some data from the db

##### Update

```
public static final
```

```
Type
```

##### Update

db update. changing some data in the db.

את שאר הפעולות ניתן לראות בקובץ עצמו

PreMadeRequest	
PreMadeRequest(Supplier<RequestBuilder>, int, VariationCreator... variations)	
DeleteUnfGames	PreMadeRequest
ChangeProfilePic	PreMadeRequest
builderBuilder	Supplier<RequestBuilder>
Games	PreMadeRequest
StatsByTimeOfDay	PreMadeRequest
TopPlayers	PreMadeRequest
requestVariations	PreMadeRequest[]
statistics	PreMadeRequest[]
authSettings	int
...	

```
public class PreMadeRequest
```

represents a db request with a Supplier<RequestBuilder> able to generate unique request builders. a remade request also has an AuthSettings for limiting access to the requests. some PreMadeRequests might want to provide some variations of an existing PreMadeRequest, and may implement the VariationCreator interface.

**See Also:** Variation, VariationCreator

**PreMadeRequest** **PreMadeRequest**(Supplier<RequestBuilder> builderBuilder, int authSettings, VariationCreator... variations) Instantiates a new Pre made request.

**Parameters:** builderBuilder - the supplier of builders authSettings - the auth requirements for this request variations - the variations for this request

**getRequestVariations** public PreMadeRequest[] **getRequestVariations()** Get the variations for this request

**Returns:** the variations for this request

**RequestBuilder** **createBuilder()** or **createBuilder()** Create a new unique request builder

**Returns:** the request builder

Variation	
<b>▪ Variation(String, Object[], Arg[])</b>	
buildingArgs	Object[]
variationArgs	Arg[]
variationName	String

```
public class Variation
```

Variation - represents a variation of a premade request.

**Variation Variation**(String variationName, Object[] buildingArgs, Arg[] variationArgs) Instantiates a new Variation.

**Parameters:** variationName - the variation name  
**Parameters:** buildingArgs - the arguments that will be used to build the original request  
**Parameters:** variationArgs - the arguments required by the variation

VariationCreator	
<b>▪ Variation create(RequestBuilder)</b>	
create	RequestBuilder

public interface VariationCreator represents a creator of a variation.

Variation **create**(RequestBuilder actualBuilder) Create a variation.

**Parameters:** actualBuilder - the builder of the original request **Returns:** the variation

[DB Response](#)

DBResponse	
DBResponse(Status, DBRequest)	
addedRes	DBResponse
request	DBRequest
status	Status
print()	void
isSuccess()	boolean
setAddedRes(DBResponse)	void
clean()	DBResponse
toString()	String
isAnyData()	boolean
...	

```
public abstract class DBResponse
```

implements Serializable Db response - represents a database response to a request .

protected **DBResponse**(Status status, DBRequest request) Instantiates a new Db response.

**Parameters:** status - the status request - the request

את שאר הפעולות ניתן לראות בקובץ עצמו

Status	
Status()	
ERROR	
SUCCESS	
valueOf(String)	Status
values()	Status[]

```
<Status>, Constable
```

```
public enum Status
```

Status - represents a response status.

```
private Status()
```

Status SUCCESS Success status.

Status ERROR Error status.

את שאר הפעולות ניתן לראות בקוד עצמו

StatusResponse	
	StatusResponse(Status, DBRequest, int)
	StatusResponse(Status, String, DBRequest, int)
	details String
	updatedRows int
	getDetails() String
	isAnyData() boolean
	clean() DBResponse

```
public class StatusResponse
```

extends DBResponse  
represents a db status response with a DBResponse.Status and the number of rows updated as a result of the request.

```
public StatusResponse(Status status, DBRequest request, int updatedRows)
Instantiates a new Status response.
```

**Parameters:** status - the status request - the request updatedRows - the updated rows

את שאר הפעולות ניתן לראות בקובץ עצמו

TableDBResponse	
TableDBResponse(String[], String[][], DBRequest)	
TableDBResponse(String[], String[][], Status, DBRequest)	
TableDBResponse()	
columns	String[]
calcedLengths	int[]
rows	String[][]
getFirstRow()	String[]
getRows()	String[][]
getColumnIndex(Col)	int
isAnyData()	boolean
...	

```
public class TableDBResponse
```

extends DBResponse represents a db response with the requested data structured in a table.

```
public TableDBResponse(String[] columns, String[][] rows, DBRequest request)
Instantiates a new Table db response.
```

**Parameters:** columns - the columns rows - the rows request - the request

את שאר הפעולות ניתן לראות בקובץ עצמו

Statements

SQLStatement		
<b>SQLStatement(Type)</b>		
<b>type</b>	Type	
<b>statement</b>	String	
<b>replace(String, String)</b>	void	
<b>createIfNotCreated()</b>	String	
<b>createStatement()</b>	String	
<b>getStatement()</b>	String	
<b>toString()</b>	String	

```
public abstract class SQLStatement
```

implements Serializable represents an sql statement. with a Type.

```
public SQLStatement(Type type) Instantiates a new Sql statement.
```

**Parameters:** type - the sql statement type

```
void replace(String replacing, String replaceWith) Replace string in the statement. used
to replace argument's placeholders with actual values.
```

**Parameters:** replacing - the replacing replaceWith - the replace with

```
protected abstract String createStatement() Create statement string.
```

**Returns:** the created sql string

```
String getStatement() Gets the statement.
```

**Returns:** the statement

את שאר הפעולות ניתן לראות בקוד עצמו

Selection	
<code>Selection(Object, Condition, Object[])</code>	
<code>Selection(Object, Object[])</code>	
<code>Condition condition</code>	Condition
<code>Object[] select</code>	Object[]
<code>String selectPrefix</code>	String
<code>String postFix</code>	String
<code>String selectFrom</code>	String
<code>String createStatement()</code>	String
<code>void top(Object)</code>	void
<code>Selection nestMe(Col[])</code>	Selection
<code>...</code>	

```
public class Selection
```

extends SQLStatement  
It represents a selection sql statement.

```
public Selection(Object selectFrom, Condition condition, Object[] select)
Instantiates a new Selection
```

**Parameters:** selectFrom - where to select from condition - the condition for selecting select - select what

Selection `nestMe( outerSelect )` create a Selection with this selection statement nested inside.

**Parameters:** outerSelect - the columns to select from the now nested selection **Returns:** the new nested selection

`void top(Object top)` get a certain number of results from the top.

**Parameters:** top - the number of results

`void join(String joinType, Table joinWith, Condition condition, groupBy) Join`  
this selection with another Table.

**Parameters:** joinType - the join type. left or right. a left join represents a join operation where every record is selected from the original selection, and any matching record from the joinWith table. a right join represents a join operation where every record is selected from the joinWith table and any matching record from the original selection, joinWith - the table to join with condition - the condition to join by groupBy - the columns to group both tables by

void **orderBy**(Col col, String order) Order this selection in an ascending or descending order.

**Parameters:** col - the col to order by order - the order

את שאר הפעולות ניתן לראות בקובץ עצמו

	<b>Delete</b>
	<b>Delete(Table, Condition)</b>
	<b>deletingFrom Table</b>
	<b>condition Condition</b>
	<b>createStatement() String</b>

```
public class Delete
```

extends SQLStatement  
It represents a deletion statement.

```
public Delete(Table deletingFrom, Condition condition) Instantiates a new Deletion statement. any record in the deletingFrom that matches the condition will be deleted.
```

**Parameters:** deletingFrom - the table this statement will delete from condition - the condition

את שאר הפעולות ניתן לראות בקובץ עצמו

[Update](#)

 Update	
 <a href="#">Update(Table, Condition, NewValue[])</a>	
 condition	Condition
 newValues	NewValue[]
 updating	Table
 createStatement()	String

```
public class Update
```

extends SQLStatement  
It represents an update statement.

```
public Update(Table updating, Condition condition, NewValue... newValues)
Instantiates a new Update.
```

**Parameters:** updating - the Table to be updated condition - the condition newValues - the new values

את שאר הפעולות ניתן לראות בקוד עצמו

 NewValue	
 <a href="#">NewValue (Col, Object)</a>	
 col	Col
 value	Object
 value()	Object
 col()	Col
 toString()	String

**Record Components:** col - The Column to update. value - The New Value.

```
public record NewValue(Col col, Object value)
```

implements `Serializable` represents a new value for a certain column in a table in the db.

**See Also:** Serialized Form

```
public NewValue(Col col, Object value) Creates an instance of a NewValue record class.
```

**Parameters:** `col` - the value for the `col` record component `value` - the value for the `value` record component

את שאר הפעולות ניתן לראות בקוד עצמו

### Table

<b>Table</b>		
<b>Table</b> (Col[])		
UnfinishedGames		
cols	Col[]	
Users		
Games		
values()	Table[]	
escapeValues(Object[], boolean, boolean)	String	
tableAndValues()	String	
valueOf(String)	Table	

<Table>, Constable

```
public enum Table
```

represents a table in the db.

```
private Table( cols ) Instantiates a new Table.
```

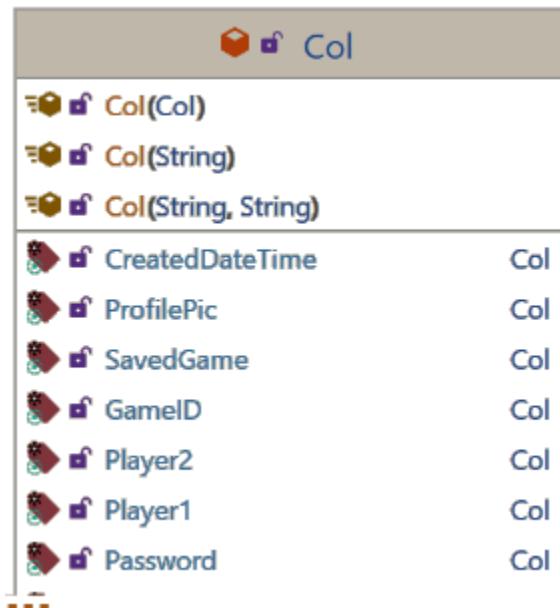
**Parameters:** cols - the columns in this table

Table **Games** Games table.

Table **UnfinishedGames** Unfinished games table.

Table **Users** Users table.

את שאר הפעולות ניתן לראות בקוד עצמו



```
public class Col
```

implements Serializable represents a column. either existing column in the db (the constant columns GameID, SavedGame ...) or created columns.

Col **count**(String as, Object countWhat) a column that will show the number of times countWhat evaluates to true

**Parameters:** as - an alias for the counting function. will show up as the column's name in the result.  
countWhat - after each record match with countWhat the counter will increment by 1. **Returns:** the counting column

Col **countIf**(String as, Condition condition) Count by a certain condition. unlike count(String, Object), that counts matches with records, this counts times a condition was true. offering more flexibility. the tradeoff is some performance.

**Parameters:** as - an alias. will show up as the column's name in the result. condition - the condition  
**Returns:** the counting col

CustomCol **sum**(String as, Col... colsToSum) a summary column of numeric columns.

**Parameters:** as - an alias. will show up as the column's name in the result. colsToSum - the cols to sum  
**Returns:** the summing col

Col **switchCase**(String as, cases) represents a Switch case column

**Parameters:** as - an alias. will show up as the column's name in the result. cases - the cases **Returns:** the switch case col **See Also:** SwitchCase

Col **time()** Time col.

**Returns:** a new col representing time

Col **date()** Date col.

**Returns:** a new col representing datetime

Col **of**(Table table) get this column, but 'belong' to a Table. meaning it's path will show with the normal col. for example: both Table.Games and Table.UnfinishedGames have the column GameID. should the need present itself, both tables might find themselves in the same statement. and specifying Which of the available GameIDs is accessed will be necessary. sure enough, by using **of**(Table) specifying the parent table is possible.

**Parameters:** table - the table **Returns:** the col

את שאר הפעולות ניתן לראות בקובץ עצמו

CustomCol	
CustomCol (String)	
CustomCol (String, String)	
nested()	String

```
public class CustomCol
```

extends Colol represents a Custom column that will keep its name, even on nesting.

public CustomCol (String colName) Instantiates a new Custom col.

**Parameters:** colName - the col name

את שאר הפעולות ניתן לראות בקוד עצמו

Math()	
Div	
Plus	
col	
Mult	
formatNum (Object)	String
apply (Object)	void
formatNum (Object, String)	String
str (Object)	String
asFloat (Object)	String
...	

<Math>, Constable

```
public enum Math
```

Math - allows for math operations on columns and some math-related utilities for columns.

```
private Math()
```

```
Math Plus add
```

```
Math Mult multiply
```

```
Math Div divide
```

**nullIf0** public static String **nullIf0**(Object val) to avoid dividing by 0, if the value is equal to 0, it will be replaced with null, and then (if set up correctly) will be handled. one way of handling with nulls is by using **zeroIfNull()**

**Parameters:** val - the val **Returns:** the string

```
String formatNum(Object num) Format a num with a default 3 decimal places.
```

**Parameters:** num - the num **Returns:** the string

```
String formatNum(Object num, String format) cast col to be in a number format.
```

**Parameters:** num - the num format - the format **Returns:** the formatted string

```
String asFloat(Object num) cast value as float.
```

**Parameters:** num - the num **Returns:** the formatted value

```
void zeroIfNull() if the col's value is null, it will be replaced with a 0.
```

Col **execute**(Col col, Col **execute**(Col col, Object value, boolean changeSelf)  
Execute this operation on the passed col, or on a copy of it.

**Parameters:** col - the col value - the value for the right side of this operation changeSelf - true to

change the column passed as a parameter. **Returns:** the changed column

את שאר הפעולות ניתן לראות בקוד עצמו

SwitchCase	
<code>SwitchCase(Condition, Col)</code>	
<code>condition</code>	Condition
<code>ifTrue</code>	Col
<code>defaultCase(Col)</code>	SwitchCase
<code>toString()</code>	String
<code>equals(Col, String, Col)</code>	SwitchCase
<code>condition()</code>	Condition
<code>ifTrue()</code>	Col

**Record Components:** condition - The Condition. ifTrue - The If true.

```
public record SwitchCase(Condition condition, Col ifTrue)
```

Switch case - represents a case that is meant to be used inside a switch case column. if the `condition` is true, the `ifTrue` col will display in the switch case col

```
public SwitchCase(Condition condition, Col ifTrue) Instantiates a new Switch case.
```

**Parameters:** condition - the condition ifTrue - the column that will show if the condition is true

SwitchCase `equals(Col col, String value, Col ifTrue)` if the passed `col` is equal to `value`, the `ifTrue` column will display.

**Parameters:** col - the col value - the value ifTrue - the if true **Returns:** the created switch case

SwitchCase `defaultCase(Col ifTrue)` if none of the previous cases matched, this column will show.

**Parameters:** ifTrue - the if true **Returns:** the switch case

`Col ifTrue () or ifTrue()` If true col.

**Returns:** the col

את שאר הפעולות ניתן לראות בקובץ עצמו

Condition	
<code>Condition(String, Object[])</code>	
<code>parms</code>	<code>Object[]</code>
<code>str</code>	<code>String</code>
<code>equals(Object, Object)</code>	<code>Condition</code>
<code>add(Condition, Relation, boolean)</code>	<code>Condition</code>
<code>toString()</code>	<code>String</code>
<code>getStr()</code>	<code>String</code>
<code>notEquals(Object, Object)</code>	<code>Condition</code>
<code>between(Object, Object, Object)</code>	<code>Condition</code>
<code>setStr(String)</code>	<code>void</code>
...	

```
public class Condition
```

implements Serializable represents a condition.

`public Condition(String str, Object... parms)` Instantiates a new Condition.

**Parameters:** str - the str parms - the parms

`Condition equals(Object col, Object value)` Equals condition. matches the toString of both objects.

**Parameters:** col - the col value - the value **Returns:** the condition

Condition `noNulls()` returns a condition that will add a not null condition for any columns included in this condition. example for when some columns' values will be null: say a general list of games and statistics is generated for any registered user. if there are registered users who haven't played a game yet, a join with the `Table.Games` table will produce a null value. since the player has never played a game. using a `noNulls()` condition will save a lot of headache by failing immediately.

**Returns:** the condition

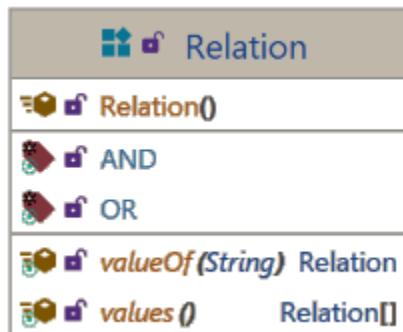
Condition `add(Condition condition, Relation relation, boolean wrap)` Add a condition to this condition.

**Parameters:** condition - the condition relation - the relation between the conditions wrap - the wrap  
**Returns:** THIS condition

Condition `between(Object col, Object start, Object end)` Between condition.

**Parameters:** col - the col start - the start end - the end **Returns:** the condition

את שאר הפעולות ניתן לראות בקובץ עצמו



<Relation>, Constable

```
public enum Relation
```

represents relations between conditions.

```
private Relation()
```

Relation **AND** And relation.

Relation **OR** Or relation.

את שאר הפעולות ניתן לראות בקובץ עצמו

 RequestBuilder	
  RequestBuilder(DBRequest Variation)	
  RequestBuilder(SQLStatement String, String, String, Arg[])	
  RequestBuilder(SQLStatement String, String, Arg[])	
  RequestBuilder(SQLStatement String, Arg[])	
  postDescription	String
  args	Arg[]
  builtArgsVals	String[]
  name	String
  subBuilder	RequestBuilder
  shouldSync	ArrayList<SyncedListType>

**Direct Known Subclasses:** RequestBuilder.GraphableSelection

```
public class RequestBuilder
```

implements Serializable represents a DBRequest that needs some context to be complete. for example: say there is a db request for getting how many times a user has won, that request will need to get a specific user as context. a request has an array of Args, with every argument it requires. when an actual DBRequest is built, all the values for the Args are passed to build(Object...), and it will create a request with all the arguments replaced with their values.

**See Also:** Serialized Form

```
public RequestBuilder(SQLStatement statement, String name, String postDescription, String preDescription, Arg[] args) Instantiates a new Request builder.
```

**Parameters:** statement - the statement name - the name of the request postDescription - a description for after building. values may be depended on the arguments, since they will be replaced with their actual values by then. preDescription - a description for before building the request. should not depend on arguments. args - the arguments required to build this request.

`RequestBuilder createVariation(Supplier<RequestBuilder> og, VariationCreator variationCreator)` creates a variation of another `RequestBuilder`.

**Parameters:** `og` - a supplier of the original request builder. a supplier is used to make sure all values are different. `variationCreator` - the variation creator **Returns:** the request builder

`p1_OR_p2 private static Condition p1_OR_p2(Object un, on p1_OR_p2(Object un, Table playersOf) creates a condition that will evaluate to true if the un is Col.Player1 or Col.Player2 of playersOf (specifying which Table is necessary to avoid ambiguity)`

**Parameters:** `un` - the username `playersOf` - the table **Returns:** the condition

`DBRequest build(Object... argsVals) st build(Object... argsVals)` Build a full DBRequest by replacing all the temporary argument values with their actual values.

**Parameters:** `argsVals` - the args vals **Returns:** the created db request

את שאר הפעולות ניתן לראות בקובץ עצמן

## Games

### Evaluation

Evaluation		
<code>Evaluation(Evaluation)</code>		
<code>Evaluation(PlayerColor)</code>		
<code>Evaluation(GameStatus, PlayerColor)</code>		
<code>Evaluation(int, GameStatus, PlayerColor)</code>		
<code>MAKE_DETAILED</code>		<code>boolean</code>
<code>gameStatus</code>		<code>GameStatus</code>
<code>perspective</code>		<code>PlayerColor</code>
<code>detailedEval</code>	<code>Collection&lt;EvaluationDetail&gt;</code>	
<code>TIE_EVAL</code>		<code>int</code>
<code>WIN_EVAL</code>		<code>int</code>

```
public class Evaluation
```

implements Serializable represents a position's evaluation relative to a PlayerColor.

Evaluation book() Book evaluation.

**Returns:** the evaluation

void addDetail(EvaluationParameters parm, int value) Add a detail to the evaluation.

**Parameters:** parm - the kind of detail value - the value

boolean isChecked() Is this evaluation a check.

**Returns:** true if this evaluation is a check

Evaluation setPerspective(PlayerColor playerColor) Sets the perspective the evaluation should be in. if this evaluation was made for the opponent, it is flipped.

**Parameters:** playerColor - the player color **Returns:** this changed evaluation

void flipEval() Flip the evaluation. will multiply the eval by -1.

את שאר הפעולות ניתן לראות בקובץ עצמו

EvaluationDetail		
<i>EvaluationDetail (EvaluationParameters, int)</i>		
eval	int	
parm	EvaluationParameters	
parm0	EvaluationParameters	
eval0	int	
toString()	String	

```
public record EvaluationDetail(EvaluationParameters parm, int eval)
```

implements Serializable represents an Evaluation detail.

public EvaluationDetail(EvaluationParameters parm, int eval) Creates an instance of a EvaluationDetail record class.

**Parameters:** parm - the value for the parm record component eval - the value for the eval record component

את שאר הפעולות ניתן לראות בקובץ עצמו

EvaluationParameters	
说实	EvaluationParameters(double)
说实	EvaluationParameters()
说实	MATERIAL
说实	FORCE_KING_TO_CORNER
说实	weight double
说实	PIECE_TABLES
说实	values() EvaluationParameters[]
说实	valueOf(String) EvaluationParameters

<EvaluationParameters>, Constable

```
public enum EvaluationParameters
```

represents an evaluation parameter, a metric on which a position might get evaluated by, an evaluation parameter has a weight that decides how much influence it has on the final evaluation of a position.

```
private EvaluationParameters(double weight)
```

EvaluationParameters **MATERIAL** pieces values evaluation parameter.

EvaluationParameters **PIECE\_TABLES** piece tables evaluation parameter.

EvaluationParameters **FORCE\_KING\_TO\_CORNER** Force king to corner evaluation parameter.

את שאר הפעולות ניתן לראות בקובץ עצמו

GameStatus		
GameStatus(SpecificStatus)		
GameStatus(GameStatus)		
GameStatus(PlayerColor, SpecificStatus)		
customStr	String	
winningPlayerColor	PlayerColor	
checkedKingLoc	Location	
gameStatusType	GameStatusType	
specificStatus	SpecificStatus	
depth	int	
checkmate(PlayerColor, Location)	GameStatus	
...		

```
public class GameStatus
```

implements Serializable represents a game status. a game status has a SpecificStatus for all the details about the specific game status and a GameStatusType for the general info about the game status. is it game over? and if so, is it a tie.

GameStatus **checkmate**(PlayerColor winningPlayerColor, Location matedKing)  
Checkmate game status.

**Parameters:** winningPlayerColor - the winning player color matedKing - the mated king **Returns:** the game status

GameStatus **gameGoesOn**() Game goes on game status.

**Returns:** the game status

`GameStatus tieByAgreement()` Tie by agreement game status.

**Returns:** the game status

`GameStatus stalemate()` Stalemate game status.

**Returns:** the game status

`GameStatus fiftyMoveRule()` Fifty move rule game status.

**Returns:** the game status

את שאר הפעולות ניתן לראות בקובץ עצמו

GameStatusType		
		<code>GameStatusType(String)</code>
		<code>GameStatusType(String, String)</code>
		<code>TIE</code>
		<code>WIN_OR_LOSS</code>
		<code>gameOverStr</code> <code>String</code>
		<code>CHECK</code>
		<code>annotation</code> <code>String</code>
		<code>GAME_GOES_ON</code>
		<code>UNFINISHED</code>
		<code>isGameOver()</code> <code>boolean</code>
<code>...</code>		

`<GameStatusType>, Constable`

`public enum GameStatusType`

`implements Serializable` represents the Game status type.

GameStatusType **TIE** Tie.

GameStatusType **CHECK** a player is in check.

GameStatusType **GAME\_GOES\_ON** the Game goes on.

GameStatusType **WIN\_OR\_LOSS** a player won.

GameStatusType **UNFINISHED** the game is unfinished.

boolean **isGameOver()** Is game over.

**Returns:** true if is game over. false otherwise

את שאר הפעולות ניתן לראות בקוד עצמו

SpecificStatus	
	<b>SpecificStatus()</b>
	<b>SpecificStatus(GameStatusType)</b>
	<b>TimedOut</b>
	<b>PlayerDisconnectedVsAi</b>
	<b>ServerStoppedGame</b>
	<b>ThreeFoldRepetition</b>
	<b>Resignation</b>
	<b>GameGoesOn</b>
	<b>Stalemate</b>
	<b>TieByAgreement</b>
...	

<SpecificStatus>, Constable

public enum **SpecificStatus**

represents a specific game status with all its details.

```
private SpecificStatus () Instantiates a new Specific status.
```

```
SpecificStatus Checkmate Checkmate.
```

```
SpecificStatus TimedOut Timed out.
```

```
SpecificStatus TimedOutVsInsufficientMaterial Timed out vs insufficient material.
```

```
SpecificStatus Resignation Resignation.
```

```
SpecificStatus GameGoesOn Game goes on .
```

את שאר הפעולות והקבועים ניתן לראות בקובץ עצמו

### Game Setup

### Board Setup

<b>Piece</b>	
<b>Piece(PieceType, PlayerColor)</b>	
	<b>pieceType</b>
	<b>PlayerColor</b>
	<b>W_P</b>
	<b>W_B</b>
	<b>PIECES_ICONS</b>
	<b>W_Q</b>
	<b>W_N</b>
	<b>B_N</b>
	<b>B_K</b>

```
<Piece>, Constable
```

```
public enum Piece
```

represents a combination of a PieceType and PlayerColor.

```
private Piece(PieceType pieceType, PlayerColor playerColor)
```

Piece **W\_P** represents a White Pawn.

Piece **W\_R** represents a White Rook.

Piece **W\_B** represents a White Bishop.

Piece **W\_N** represents a White Knight.

את שאר הפעולות והקבועים ניתן לראות בקוד עצמו

<b>PieceType</b>		
		<b>PieceType</b> (String, String, int, boolean)
		<b>PieceType</b> (String, String, int)
		COLORLESS_PIECES_FENS String[]
		asInt int
		value int
		QUEEN
		MINOR_PIECES PieceType[]
		KING
		UNIQUE_MOVES_PIECE_TYPES PieceType[]
		NUM_OF_PIECE_TYPES int
...		

<PieceType>, Constable

```
public enum PieceType
```

implements Serializable represents the Piece Type.

PieceType **PAWN** Pawn Piece Type.

PieceType **ROOK** Rook piece type.

PieceType **BISHOP** Bishop piece type.

PieceType **KNIGHT** Knight piece type.

PieceType **QUEEN** Queen piece type.

PieceType **KING** King piece type.

`boolean isAttack(Direction direction, int maxDistance)` Is this piece type attacking a square a maxDistance away in a direction.

**Parameters:** direction - the direction maxDistance - the max distance to the square **Returns:** true if this piece can attack the square

את שאר הפעולות ניתן לראות בקוד עצמו

 Board	
  <b>Board(Board)</b>	
  <b>Board(String)</b>	
  <b>Board()</b>	
   <b>boardMap Map&lt;Location, Square&gt;</b>	
  <b>startingFen</b>	<b>String</b>
  <b>getPiece(Location)</b>	<b>Piece</b>
  <b>fenSetup(String)</b>	<b>void</b>
  <b>print()</b>	<b>void</b>
  <b>iterator()</b>	<b>Iterator&lt;Square&gt;</b>
  <b>getSquare(Location)</b>	<b>Square</b>
<b>...</b>	

<Square>

```
public class Board  
  
implements Iterable<Square>, Serializable represents a logic board with 64 Squares
```

public **Board()** Instantiates a new empty Board.

public **Board(String fen)** Instantiates a new Board and sets it up with a fen.

**Parameters:** fen - the fen

void **setPiece(Location loc, Piece piece)** Sets a piece on a square.

**Parameters:** loc - the square's loc piece - the piece

Piece **getPiece(Location loc)** Gets piece from a square.

**Parameters:** loc - the square's loc **Returns:** the piece or null if no piece is on that square

boolean **isSquareEmpty(Location loc)** Is a square empty.

**Parameters:** loc - the loc **Returns:** true if the square is empty

את שאר הפעולות ניתן לראות בקוד עצמו

Square		
Square(Location)		
EMPTY_PIECE_STR	String	
loc	Location	
EMPTY_PIECE	Piece	
piece	Piece	
getPiece()	Piece	
getPieceIcon()	String	
getFen()	String	
toString()	String	
isEmpty()	boolean	
...		

```
public class Square
```

implements Serializable represents a square on the logic board, with a Location and a Piece

public **Square**(Location loc) Instantiates a new Square.

**Parameters:** loc - the location of the square

void **setEmpty**() Set this square to be empty.

Piece **getPiece**() ce **getPiece**() Gets the piece on this square.

**Returns:** the piece if this square isn't empty, or EMPTY\_PIECE if it is.

void **setPiece**(Piece piece) Sets a piece on this square.

**Parameters:** piece - the piece

boolean **isEmpty**() Is this square empty.

**Returns:** true if this square is empty

את שאר הפעולות ניתן לראות בקובץ עצמו

GameSettings		
<code>GameSettings(AISettings)</code>		
<code>GameSettings(GameSettings)</code>		
<code>GameSettings(PlayerColor, TimeFormat, String, AISettings, GameType)</code>		
<code>serialVersionUID</code>		<i>long</i>
<code>fen</code>		<i>String</i>
<code>gameType</code>		<i>GameType</i>
<code>gameID</code>		<i>String</i>
<code>playerToMove</code>		<i>PlayerColor</i>

```
public class GameSettings
```

implements Serializable, TimeFormatComponent  
nt represents Game settings. the starting position, which PlayerColor turn is it to move, the TimeFormat, the AISettings and the GameType

public GameSettings(PlayerColor playerToMove, TimeFormat timeFormat, String fen, AISettings AISettings, GameType gameType) Instantiates a new Game settings.

**Parameters:** playerToMove - the player to move timeFormat - the time format fen - the fen AISettings - the ai parameters gameType - the game type

את שאר הפעולות ניתן לראות בקובץ עצמו

Condition	
<code>Condition(String, Object[])</code>	
<code>parms</code>	<code>Object[]</code>
<code>str</code>	<code>String</code>
<code>equals(Object, Object)</code>	<code>Condition</code>
<code>add(Condition, Relation, boolean)</code>	<code>Condition</code>
<code>toString()</code>	<code>String</code>
<code>getStr()</code>	<code>String</code>
<code>notEquals(Object, Object)</code>	<code>Condition</code>
<code>between(Object, Object, Object)</code>	<code>Condition</code>
<code>setStr(String)</code>	<code>void</code>
<code>...</code>	

<GameType>, Constable

```
public enum GameType
```

represents a Game's type.

```
private GameType()
```

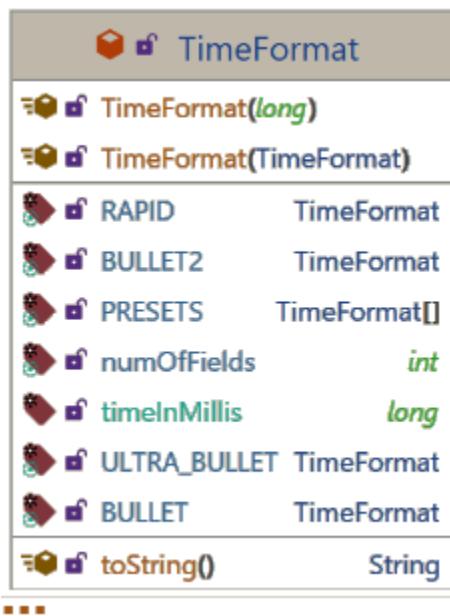
GameType JOIN\_EXISTING Join existing game.

GameType RESUME Resume game.

GameType CREATE\_NEW Create new game.

GameType QUICK\_MATCH Quick match.

את שאר הפעולות ניתן לראות בקובץ עצמו



```
public class TimeFormat
```

implements Serializable represents a player's Time format. how much time does he have for each move.

public **TimeFormat**(long timeInMillis) Instantiates a new Time format.

**Parameters:** timeInMillis - the time in millis

את שאר הפעולות ניתן לראות בקובץ עצמו

GameTime		
GameTime(TimeFormat[])		
GameTime(GameTime)		
gameTime	<code>long[]</code>	
currentlyRunning	PlayerColor	
timeFormats	TimeFormat[]	
lastStart	<code>long</code>	
clean()	GameTime	
startRunning(PlayerColor)	<code>void</code>	
getTimeFormat(PlayerColor)	TimeFormat	
getCurrentlyRunning()	PlayerColor	
...		

```
public class GameTime
```

implements Serializable represents the current state of both player's time.

public GameTime( TimeFormats ) Instantiates a new Game time with the formats for both players.  
possible inputs: [TimeFormat] will set both players' times to that time. [TimeFormat, TimeFormat] will  
set white player's time to the first one and black's to the second.

**Parameters:** timeFormats - the time formats

TimeFormat **getTimeFormat**(PlayerColor clr) at **getTimeFormat**(PlayerColor clr) Gets the time  
format for a player.

**Parameters:** clr - the clr **Returns:** the time format

void **startRunning**(PlayerColor playerColor) Start running a player's clock.

**Parameters:** playerColor - the player color

long **getTimeLeft**(PlayerColor playerColor) Gets the time left for a player.

**Parameters:** playerColor - the player color **Returns:** the time left in milliseconds

את שאר הפעולות ניתן לראות בקובץ עצמו

AISettings		
		<code>AISettings(AISettings)</code>
		<code>AISettings(AiType, TimeFormat)</code>
		<code>AISettings()</code>
		<code>aiType</code> <code>AiType</code>
		<code>moveSearchTimeout</code> <code>TimeFormat</code>
		<code>setAiType(AiType)</code> <code>void</code>
		<code>isEmpty()</code> <code>boolean</code>
		<code>getMoveSearchTimeout()</code> <code>TimeFormat</code>
		<code>setMoveSearchTimeout(TimeFormat)</code> <code>void</code>
		<code>set(TimeFormat)</code> <code>void</code>
<code>...</code>		

<TimeFormat>

```
public class AISettings
```

implements Serializable, ParentOf<TimeFormat> represents Ai Settings. the AiType and how much time can it think.

public **AISettings**(AiType aiType, TimeFormat moveSearchTimeout) Instantiates a new Ai parameters.

**Parameters:** aiType - the ai type moveSearchTimeout - the move search timeout

TimeFormat **getMoveSearchTimeout()** at **getMoveSearchTimeout()** Gets move search timeout.

**Returns:** the move search timeout

void **setMoveSearchTimeout**(TimeFormat moveSearchTimeout) Sets move search timeout.

**Parameters:** moveSearchTimeout - the move search timeout

את שאר הפעולות ניתן לראות בקובץ עצמו

AiType	
■	AiType()
■	MyAi
■	Stockfish
■	valueOf(String) AiType
■	values() AiType[]

<AiType>, Constable

public enum **AiType**

Ai type.

private **AiType**()

AiType **Stockfish** Stockfish ai type.

AiType **MyAi** My ai type.

את שאר הפעולות ניתן לראות בקובץ עצמו

Location		
	Location()	
WHITE_STARTING_ROW	<i>int</i>	
D	<i>int</i>	
blackSquares	<i>long</i>	
asInt	<i>int</i>	
F6		
H5		
B8		
F5		
B2		
...		

<Location>, Constable

```
public enum Location
```

an enum consisting of 64 values representing all 64 squares on the board. used to access squares on the board an enum is used for performance reasons.

`private Location()` Instantiates a new Location.

Location **A8**

Location **B8**

Location **C8**

את השאר ניתן לראות בקוד

`Location getLoc(Location loc, int numOfMult, Direction direction)` Gets the location relative to loc in the direction given and the distance is determined by the numOfMult

**Parameters:** loc - the loc **numOfMult** - the num of mult **direction** - the direction **Returns:** the location if the calculated index is inside the bounds(0...63). null otherwise

`Location getLoc(Location loc, int add)` Gets the location that is exactly add squares from loc  
 NOTE: add should be in bitboard format

**Parameters:** loc - the loc add - the number of squares to add **Returns:** the location if the calculated index is inside the bounds(0...63). null otherwise **See Also:** Bitboard

`Location getLoc(int locIndex)` Gets location corresponding to the locIndex provided (0..63)

**Parameters:** locIndex - the locIndex **Returns:** the location if the provided index is inside the bounds(0...63). null otherwise

את שאר הפעולות ניתן לראות בקובץ עצמו

PlayerColor		
<code>PlayerColor(int, int)</code>		
indexOf2	<code>int</code>	
NUM_OF_PLAYERS	<code>int</code>	
BLACK		
startingRow	<code>int</code>	
NO_PLAYER		
asInt	<code>int</code>	
diff	<code>int</code>	
WHITE		
PLAYER_COLORS	<code>PlayerColor[]</code>	
...		

<PlayerColor>, Constable

public enum `PlayerColor`

represents a player color.

`private PlayerColor(int startingRow, int diff)` Instantiates a new Player color.

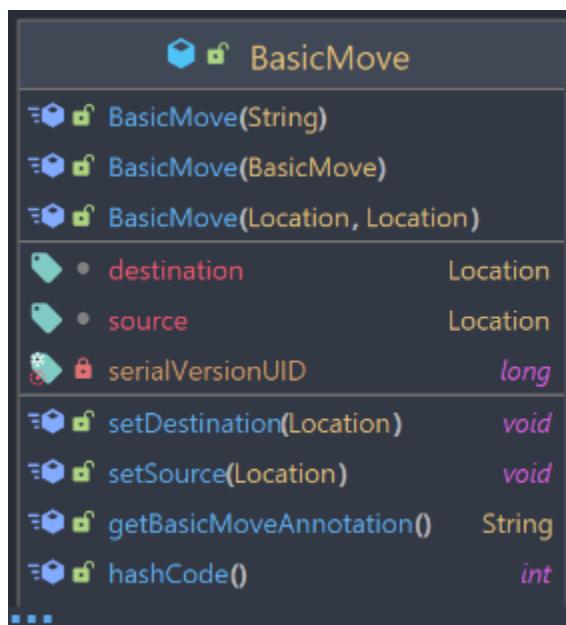
**Parameters:** startingRow - the starting row diff - the diff

PlayerColor **WHITE** White.

PlayerColor **BLACK** Black.

PlayerColor **NO\_PLAYER** No player.

את שאר הפעולות ניתן לראות בקובץ עצמו



```
public class BasicMove
```

implements Serializable Basic move - represents a basic move. with a source and a destination.

public **BasicMove**(Location source, Location destination) Instantiates a new Basic move.

**Parameters:** source - the source destination - the destination

את שאר הפעולות ניתן לראות בקובץ עצמו

 Move

  Move(ThreefoldStatus)	
  Move(Location, Location)	
  Move(Move)	
  Move(Location, Location, PieceType)	
  disabledCastling	<i>byte</i>
  isReversible	<i>boolean</i>
  movingPlayerColor	PlayerColor
  NOT_CAPTURING	PieceType
  threefoldStatus	ThreefoldStatus
  createdListHashSupplier	LazyHashSupplier<Long>
<b>...</b>	

&lt;Move&gt;

```
public class Move
```

```
extends BasicMove
```

implements Comparable<Move> represents a "heavy" move. with a lot of info. one of the fields it has is intermediateMove (aka Zwischenzug) which is a move to play before this move. like moving the king in castling, or moving a pawn once in a double pawn push.

```
public Move(Location source, Location destination) Instantiates a new Move.
```

**Parameters:** source - the source destination - the destination

Move castling(Location source, Location destination, Side side) creates a Castling move.

**Parameters:** source - the source destination - the destination side - the castling side **Returns:** the castling move

boolean isCheck() Is this move a check.

**Returns:** true if this move is a check

`PieceType getCapturingPieceType()` Gets the type of piece this move captures.

**Returns:** the piece if one is captured, null otherwise.

`boolean isReversible()` Is this move reversible.

**Returns:** true if this move is reversible **See Also:** chessprogramming.org/Reversible\_Moves

את שאר הפעולות ניתן לראות בקוד עצמו

MoveFlag		
	<code>MoveFlag(Side)</code>	
	<code>MoveFlag()</code>	
	<code>isCastling</code>	<code>boolean</code>
	<code>NormalMove</code>	
	<code>Promotion</code>	
	<code>DoublePawnPush</code>	
	<code>EnPassant</code>	
	<code>castlingSide</code>	<code>Side</code>
	<code>ShortCastle</code>	
	<code>CASTLING_FLAGS</code>	<code>MoveFlag[]</code>

<Move.MoveFlag>, Constable

```
public static enum Move.MoveFlag
```

extends Enum<Move.MoveFlag> Move flag - which type of move this is.

`Move.MoveFlag NormalMove` Normal move move flag.

`Move.MoveFlag EnPassant` En passant move flag.

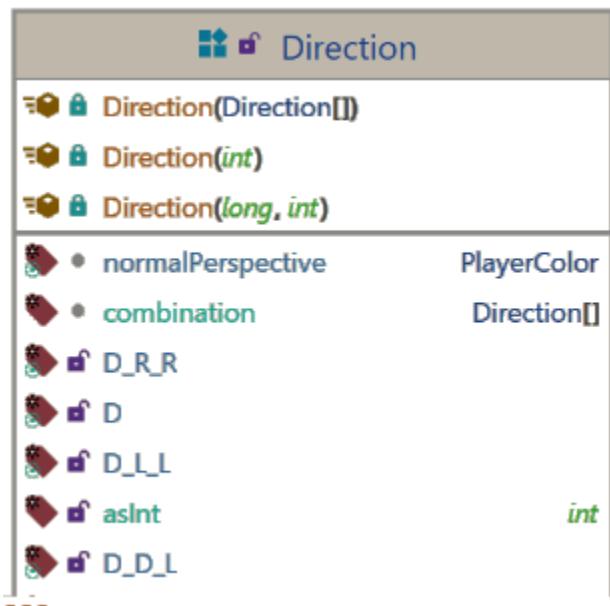
`Move.MoveFlag DoublePawnPush` Double pawn push move flag.

`Move.MoveFlag Promotion` The Promotion.

`Move.MoveFlag ShortCastle` Short castle move flag.

`Move.MoveFlag LongCastle` Long castle move flag.

את שאר הפעולות ניתן לראות בקובץ עצמו



`<Direction>, Constable`

```
public enum Direction
```

Direction - represents a moving direction on a board. sort of like a vector. has a `offset` that is added to a certain location or bitboard, in order to achieve movement in that direction. the general direction map looks like this:

-9      -8      -7

-1 loc 1

7 8 9

Direction **U** one square up the board.

Direction **D** one square down the board.

Direction **L** one square to the left.

Direction **R** one square to the right.

#### את השאר ניתן לראות בקוד

Direction **perspective**(PlayerColor playerColor) gets the correct perspective for the provided player color. this is necessary because for example: a white pawn push(**U**) is the exact opposite of a black pawn push (**D**). so the perspective needs to be in relation to the moving color.

**Parameters:** playerColor - the player color **Returns:** the direction

**opposite** public abstract Direction **opposite()** the Opposite direction to this one.

**Returns:** the direction

#### את שאר הפעולות ניתן לראות בקוד עצמו

CastlingRights		
CastlingRights(CastlingRights)		
CastlingRights(byte)		
CastlingRights()		
rights	byte	
PLAYER_MASKS	byte[]	
NO_CASTLING_ABILITY	String	
RIGHTS	byte[]	
FENS	String[]	
toString()	String	
disableCastling(PlayerColor, Side)	byte	
...		

```
public class CastlingRights
```

implements Serializable represents castling rights for both players in a position, using a byte. two bits for each side for each player. 4 bits total. (a byte is the smallest available. so 4 bytes go to waist). examples: white and black can castle both sides: 1 1 1 1 white can castle king side and black can queen side: 1 0 0 1 white can't castle black can both: 1 1 0 0 white can both black can't: 0 0 1 1 white can king side black can't: 0 0 0 1

public CastlingRights() Instantiates a new Castling rights. with a default value of 0. (no one can castle)

void enableCastling(PlayerColor playerColor, CastlingRights.Side side) Enable castling for a player on a side.

**Parameters:** playerColor - the player color side - the side

boolean isEnabled(PlayerColor playerColor, CastlingRights.Side side) Is a player's castling right enabled.

**Parameters:** playerColor - the player color side - the side **Returns:** true if the player can castle

byte disableCastling(PlayerColor playerColor) disable a player's castling ability to both sides.

**Parameters:** playerColor - the player color **Returns:** a byte with bits set where the disabling changed

את שאר הפעולות ניתן לראות בקוד עצמו

Side	
Side(int, int, int)	
castledKingCol	int
asInt	int
rookStartingCol	int
SIDES	Side[]
kingTravelDistance	int
QUEEN	
KING	
castledRookCol	int
mult	int

<CastlingRights.Side>, Constable

public static enum CastlingRights.Side

extends Enum<CastlingRights.Side> Castling side.

private **Side**(int castledKingCol, int rookStartingCol, int castledRookCol)  
Instantiates a new Side.

**Parameters:** castledKingCol - the castled king col rookStartingCol - the rook starting col  
castledRookCol - the castled rook col

CastlingRights.Side **KING** King side.

CastlingRights.Side **QUEEN** Queen side.

Location **kingFinalLoc**(Location currentKingLoc) on **kingFinalLoc**(Location currentKingLoc)  
calculate the King's final castled location.

**Parameters:** currentKingLoc - the current king loc **Returns:** the location

את שאר הפעולות ניתן לראות בקובץ עצמו

BitData		
<b>BitData()</b>		
notAFile	<i>long</i>	
notHFile	<i>long</i>	
everything	<i>long</i>	

```
public class BitData
```

utility class for storing useful board constants in bitboard format

MovesList		
<b>MovesList(MovesList)</b>		
<b>MovesList()</b>		
hash	<i>long</i>	
finalHash	<i>long</i>	
<b>finalizeHash()</b>	<i>void</i>	
<b>findMove(BasicMove, CompareMoves)</b>	<i>Move</i>	
<b>findMove(BasicMove)</b>	<i>Move</i>	
<b>add(Move)</b>	<i>boolean</i>	
<b>getFinalHash()</b>	<i>long</i>	
<b>createSimpleStr()</b>	<i>String</i>	
...		

<Move>, Collection<Move>, List<Move>, RandomAccess

```
public class MovesList
```

```
extends ArrayList<Move>
```

`implements Serializable` represents a list of moves, with a calculated hash used to find threefold repetitions.

את שאר הפעולות ניתן לראות בקוד עצמו

MoveAnnotation		
<code>MoveAnnotation()</code>		
<code>CAPTURE_ANN</code>		<code>String</code>
<code>annotate(Move, Piece, String)</code>		<code>String</code>
<code>annotate(Move, Piece)</code>		<code>String</code>
<code>basicAnnotate(BasicMove)</code>		<code>String</code>

```
public class MoveAnnotation
```

utility class for annotating moves.

`String annotate(Move move, Piece movingPiece)` Annotate move in the PGN format

**Parameters:** move - the move movingPiece - the moving piece **Returns:** the annotation

`String basicAnnotate(BasicMove move)` Basic annotate a move. format:[source:destination]

**Parameters:** move - the move **Returns:** the string

את שאר הפעולות ניתן לראות בקוד עצמו

<b>MinimaxMove</b>	
<b>MinimaxMove(Evaluation)</b>	
<b>MinimaxMove(MinimaxMove)</b>	
<b>MinimaxMove(Move, Evaluation)</b>	
<b>moveEvaluation</b>	<i>Evaluation</i>
<b>move</b>	<i>Move</i>
<b>moveDepth</b>	<i>int</i>
<b>isDeeperAndBetterThan(MinimaxMove)</b>	<i>boolean</i>
<b>getShortPrintingStr()</b>	<i>String</i>
<b>setMoveEvaluation(Evaluation)</b>	<i>void</i>
<b>equals(Object)</b>	<i>boolean</i>
<b>...</b>	

&lt;MinimaxMove&gt;

```
public class MinimaxMove
```

implements Comparable<MinimaxMove>, Serializable represents a Minimax move with a score and depth.

public **MinimaxMove** (Move move, Evaluation moveEvaluation) Instantiates a new Minimax move.

**Parameters:** move - the move moveEvaluation - the move evaluation

boolean **isDeeperAndBetterThan** (MinimaxMove other) Is deeper and better than given minimax move.

**Parameters:** other - the other **Returns:** true if this evaluation is better

את שאר הפעולות ניתן לראות בקובץ עצמו

[Saved Games](#)

GameInfo	
<b>GameInfo(String, String, GameSettings)</b>	
gameId	String
gameSettings	GameSettings
serialVersionUID	long
creatorUsername	String
getStartingColor()	PlayerColor
getJoiningPlayerColor()	PlayerColor
getGameDesc()	String
toString()	String
isCreator(String)	boolean
...	

```
public class GameInfo
```

implements Serializable, SyncableItemem represents a Game info.

```
public GameInfo(String gameId, String creatorUsername, GameSettings gameSettings) Instantiates a new Game info.
```

**Parameters:** gameId - the game id creatorUsername - the creator username gameSettings - the game settings

את שאר הפעולות ניתן לראות בקוד עצמו

<b>EstablishedGameInfo</b>	
<b>EstablishedGameInfo</b> (String, String, String, GameSettings, Stack<Move>)	
<b>moveStack</b>	Stack<Move>
<b>createdAt</b>	Date
<b>opponentUsername</b>	String
<b>setCreatedAt(Date)</b>	void
<b>toString()</b>	String
<b>getMoveStack()</b>	Stack<Move>
<b>getGameDesc()</b>	String
<b>getCreatedAt()</b>	Date

public abstract class

**EstablishedGameInfo**

extends GameInfofo represents a game that was established between two players.

protected **EstablishedGameInfo**(String gameId, String creatorUsername, String opponentUsername, GameSettings gameSettings, Stack<Move> moveStack) Instantiates a new Established game info.

**Parameters:** gameId - the game id creatorUsername - the creator username opponentUsername - the opponent username gameSettings - the game settings moveStack - the move stack

את שאר הפעולות ניתן לראות בקוד עצמו

<b>UnfinishedGame</b>	
<b>UnfinishedGame</b> (String, String, GameSettings, String, PlayerColor)	
<b>playerColorToMove</b>	PlayerColor
<b>playerToMove</b>	String
<b>isCreatorToMove()</b>	boolean

public class

**UnfinishedGame**

extends EstablishedGameInfofo represents an Unfinished game.

את שאר הפעולות ניתן לראות בקוד עצמו

  ArchivedGameInfo	
  ArchivedGameInfo(String, String, String, GameSettings, String, String)	
  winner	String
  toString()	String
  getWinner()	String
	public class
<b>ArchivedGameInfo</b>	

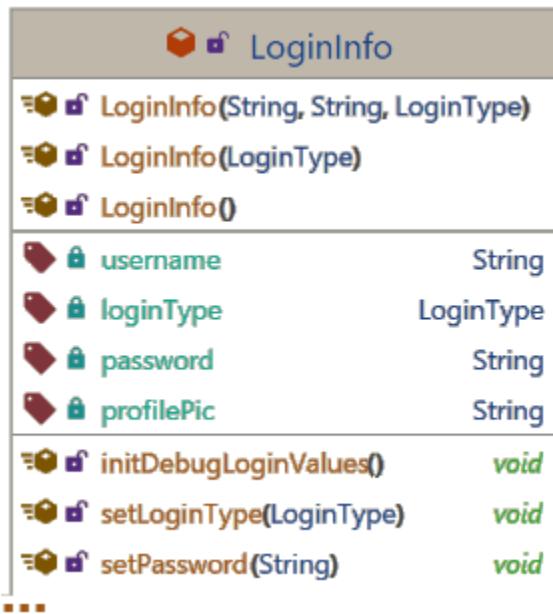
extends EstablishedGameInfo to represents a game that was finished, and is intended to be archived in the db.

`String getWinner()` Gets the winner of this game.

**Returns:** the winner's username if the game is decisive. "----tie----" otherwise

את שאר הפעולות ניתן לראות בקובץ עצמו

[Login](#)



```
public class LoginInfo
```

```
implements Serializable represents Login info .
```

```
public LoginInfo(String username, String password, LoginType loginType)
Instantiates a new Login info.
```

**Parameters:** username - the username password - the password loginType - the login type

את שאר הפעולות ניתן לראות בקובץ עצמו

**LoginType**

<b>NOT_SET_YET</b>
<b>CANCEL</b>
<b>REGISTER</b>
<b>LOGIN</b>
<b>GUEST</b>
<b>values()</b> <code>LoginType[]</code>
<b>toString()</b> <code>String</code>
<b>valueOf(String)</b> <code>LoginType</code>
<b>asUser()</b> <code>boolean</code>
...

<LoginType>, Constable

```
public enum LoginType
```

represents a Login Type.

`LoginType LOGIN` Login.

`LoginType REGISTER` Register.

`LoginType GUEST` Guest.

`LoginType CANCEL` Cancel.

`LoginType NOT_SET_YET` Not set yet.

`boolean asUser()` is this login type of user. not a guest.

**Returns:** true if this login type is of a user

את שאר הפעולות ניתן לראות בקוד עצמו

@ ⚡ AuthSettings		
⚡ 🔒 GUEST	<i>int</i>	
⚡ 🔒 ANY_LOGIN	<i>int</i>	
⚡ 🔒 NEVER_AUTH	<i>int</i>	
⚡ 🔒 USER	<i>int</i>	
⚡ 🔒 NO_AUTH	<i>int</i>	

```
public @interface AuthSettings Auth settings. represents all types of login authentication.
```

### Networking

📦 ⚡ AppSocket		
⚡ 📦 AppSocket(String, <i>int</i> )		
⚡ 📦 AppSocket(Socket)		
🔒 🔒 messagesHandler	MessagesHandler	
🔒 🔒 msgIS	ObjectInputStream	
🔒 🔒 msgSocket	Socket	
🔒 🔒 msgOS	ObjectOutputStream	
🔒 🔒 didDisconnect	<i>boolean</i>	
⚡ 📦 requestMessage(Message)	Message	
⚡ 📦 stopReading()	<i>void</i>	
⚡ 📦 close(MyError)	<i>void</i>	
...		

```
public class AppSocket
```

extends MyThread  
ad represents a communications socket able to send and receive Messages to and from another AppSocket.

```
public AppSocket(String ip, int port) Instantiates a new App socket to a specified address
```

**Parameters:** ip - the ip port - the port **Throws:** IOException - the io exception

`void close (MyError err)` Close this socket, and interrupt every method waiting for a message.

**Parameters:** `err` - the error to interrupt with

`void requestMessage (Message requestMsg, MessageCallback onRes)` send a Request and call a MessageCallback when a response message is received.

**Parameters:** `requestMsg` - the request message `onRes` - the callback to call when a response is received

`void respond (Message msg, Message respondingTo)` Respond to a request message.

**Parameters:** `msg` - the response `respondingTo` - the request message

`void writeMessage (Message msg)` Write a message.

**Parameters:** `msg` - the message

Message `requestMessage (Message requestMsg)` ge `requestMessage (Message requestMsg)` send a request and block this thread until a response is read. then return the response.

**Parameters:** `requestMsg` - the request message **Returns:** the response

את שאר הפעולות ניתן לראות בקובץ עצמו

MessagesHandler	
<b>MessagesHandler(AppSocket)</b>	
customCallbacks	Map<String, MessageCallback>
waiting	Vector<CompletableFuture<Message>>
socket	AppSocket
chronologicalSemaphore	Semaphore
didDisconnect	boolean
defaultCallbacks	Map<MessageType, MessageCallback>
isExpectingDisconnect	boolean
<b>onDisconnected()</b>	void
<b>onAnyDisconnection()</b>	void
...	

```
public abstract class MessagesHandler
```

Messages handler - handles all types of messages by using a hash map to make all routing as fast as possible. when a request is sent to the server, a callback is passed with it. when a response is received, that callback is called with the response message. if a message that isn't a response is received, the handling is differed to the default callbacks map. which is where most of the implementation of this abstract class comes in. the individual message type handling

void **interruptBlocking**(MyError err) interrupt every thread waiting for a response with an error.

**Parameters:** err - the error

Message **blockTilRes**(Message request) ge **blockTilRes**(Message request) send a request and Block this thread until a response is received

**Parameters:** request - the request **Returns:** the response

void **noBlockRequest**(Message request, MessageCallback onRes) send a non-blocking request.

**Parameters:** request - the request onRes - the callback to call once a response is received

void **processMessage**(Message message) Process a message read from the socket.

**Parameters:** message - the message

את שאר הפעולות ניתן לראות בקובץ עצמו

Message	
Message(MessageType, String, String)	
Message(MessageType, String)	
Message(MessageType)	
Message(MessageType, Message)	
playerColor	PlayerColor
preMoves	ArrayList<Move>
respondingToMsgId	String
loginInfo	LoginInfo
username	String
gameStatus	GameStatus

```
public class Message
```

implements Serializable represents a Message that is used to carry information over the network.

```
public Message(MessageType messageType, Message respondingTo) Instantiates a new response Message.
```

**Parameters:** messageType - the message type respondingTo - the request message this message is responding to

```
public Message(MessageType messageType, String subject) Instantiates a new Message.
```

**Parameters:** messageType - the message type subject - the subject

Message askForLogin() create a request for a login message.

**Returns:** the message

Message `returnLogin(LoginInfo loginInfo, Message respondingTo)` respond to a login request.

**Parameters:** `loginInfo` - the login info `respondingTo` - the responding to **Returns:** the message

את שאר הפעולות ניתן לראות בקוד עצמו

Message Type	
	<code>MessageType(boolean)</code>
	<code>MessageType()</code>
	<code>WELCOME_MESSAGE</code>
	<code>UPDATE_BY_MOVE</code>
	<code>ERROR</code>
	<code>GAME_OVER</code>
	<code>CANCEL_QUESTION</code>
	<code>INTERRUPT</code>
	<code>WAIT_FOR_MATCH</code>
	<code>WAIT_TURN</code>

<MessageType>, Constable

```
public enum MessageType
```

represents a Message type.

`MessageType.LOGIN` Login message type.

`MessageType.RESIGN` Resign message type.

`MessageType.WELCOME_MESSAGE` Welcome message type.

את השאר ניתן לראות בקוד

Sync

SyncableItem	
ID	String
getSyncableItem()	SyncableItem

public interface `SyncableItem` represents an item that can be synchronized.

`getSyncableItem` default `SyncableItem getSyncableItem()` Gets the syncable item. some items might not be compatible for syncing themselves, so they may override this function and create a syncable representation of themselves.

**Returns:** the syncable item representing the current state of this obj

`String ID()` a syncable item must have a unique id.

**Returns:** the id

SyncedItems<E>	
<code>SyncedItems(SyncedListType)</code>	
<code>SyncedItems(SyncedListType, Callback &lt;SyncedItems&lt;E&gt;&gt;)</code>	
<code>serialVersionUID</code>	<code>long</code>
<code>syncedListType</code>	<code>SyncedListType</code>
<code>onUpdate</code>	<code>Callback &lt;SyncedItems&lt;E&gt;&gt;</code>
<code>put(E)</code>	<code>SyncableItem</code>
<code>updated()</code>	<code>void</code>
<code>addAll(Collection&lt;SyncableItem&gt;)</code>	<code>void</code>
<code>put(String, E)</code>	<code>E?</code>
<code>forEachItem(Callback &lt;E&gt;)</code>	<code>void</code>
...	

**Type Parameters:** E - the type of syncable elements in this collection

```
<String,E>, Map<String,E>

public class SyncedItems<E extends SyncableItem>

extends ConcurrentHashMap<String,E> represents a collection of Synced items.
```

**See Also:** Serialized Form

```
public SyncedItems(SyncedListType syncedListType, Callback<SyncedItems<E>>
onUpdate) Instantiates a new Synced items.
```

**Parameters:** syncedListType - the synced list type onUpdate - the callback to call when a change is made to the collection (removal or adding of elements)

את שאר הפעולות ניתן לראות בקובץ עצמו

SyncedListType	
SyncedListType()	
CONNECTED_USERS	
ONGOING_GAMES	
JOINABLE_GAMES	
RESUMABLE_GAMES	
valueOf(String)	SyncedListType
values()	SyncedListType[]

```
<SyncedListType>, Constable
```

```
public enum SyncedListType
```

represents a Synced list type.

SyncedListType **RESUMABLE\_GAMES** games that have been paused and may be resumed.

SyncedListType **JOINABLE\_GAMES** games a player can join.

SyncedListType **CONNECTED\_USERS** the connected users to the server.

SyncedListType **ONGOING\_GAMES** the games that are being played on the server.

את שאר הפעולות ניתן לראות בקוד עצמו

UserInfo	
•	UserInfo(String)
•	UserInfo(String, String)
•	profilePic      String
•	id                String
•	ID()             String

```
public class UserInfo

implements SyncableItem, Serializable represents syncable User information.
```

את שאר הפעולות ניתן לראות בקוד עצמו

Threads

MyThread	
<b>MyThread()</b>	
<b>errorHandlers</b> Map<Class<MyError>, ErrorHandler<MyError>>	
<b>ignoreerrs</b>	boolean
<b>envManager</b>	EnvManager
<b>threadStatus</b>	ThreadStatus
<b>setEnvManager(EnvManager)</b>	void
<b>isIgnoreerrs()</b>	boolean
<b>stopRun()</b>	void
<b>currentThread(Callback&lt;MyThread&gt;)</b>	void
<b>ignoreerrs()</b>	void
...	

```
public abstract class MyThread
```

extends Thread represents My implementation of an error handling thread. when an error is thrown inside a MyThread, a map of error handlers is searched to find a handler set to deal with the type of error thrown. if one is found, it is called. and the EnvManager will be notified of a handled error. if a handler is not found, the EnvManager will be notified that an unhandled error has occurred, and he will then begin shutting down.

void **setEnvManager**(EnvManager manager) Sets the manager of this environment.

**Parameters:** manager - the manager

**addHandler** public <E extends MyError>

void **addHandler**(Class<E> errClass, ErrorHandler<E> onErr) Add an error handler.

**Type Parameters:** E - the type of errors this handler will handle **Parameters:** errClass - the class of the error onErr - the error handler

void **stopRun()** Stop the run of this thread. if this thread's state is set to MyThread.ThreadStatus.RUNNING it will be interrupted.

**handledRun** protected abstract void **handledRun()** run this thread in a handled manner.

**Throws:** `Throwable` - the error that might get thrown

את שאר הפעולות ניתן לראות בקוד עצמו

ThreadStatus	
	<b>ThreadStatus()</b>
	<b>NOT_STARTED</b>
	<b>RUNNING</b>
	<b>DONE</b>
	<b>next()</b> ThreadStatus
	<b>values()</b> ThreadStatus[]
	<b>valueOf(String)</b> ThreadStatus

<MyThread.ThreadStatus>, Constable

```
public static enum MyThread.ThreadStatus
extends Enum<MyThread.ThreadStatus> Thread status.
```

MyThread.ThreadStatus NOT\_STARTED Not started.

MyThread.ThreadStatus RUNNING Running.

MyThread.ThreadStatus DONE Done.

את שאר הפעולות ניתן לראות בקוד עצמו

HandledThread	
HandledThread(ThrowingRunnable)	
HandledThread()	
runnable	ThrowingRunnable
handledRun()	void
setRunnable(ThrowingRunnable)	void
runInHandledThread(ThrowingRunnable)	HandledThread

```
public class HandledThread
```

extends MyThread represents a thread that can handle errors.

public **HandledThread**(ThrowingRunnable runnable) Instantiates a new Handled thread.

**Parameters:** runnable - the runnable

HandledThread **runInHandledThread**(ThrowingRunnable runnable) Run in a handled thread

**Parameters:** runnable - the runnable **Returns:** the handled thread

את שאר הפעולות ניתן לראות בקוד עצמו

[Error Handling](#)



```
public class MyError
```

extends `Error` represents my implementation of an error.

`public MyError(Throwable cause)` Instantiates a new error with a cause.

**Parameters:** `cause` - the cause

`public MyError(String message)` Instantiates a new error with an error message.

**Parameters:** `message` - the error message

`public MyError(String message, Throwable cause)` Instantiates a new error.

**Parameters:** `message` - the error message `cause` - the cause

`String getShortDesc()` Gets a short description of this error.

**Returns:** the short description of this error

את שאר הפעולות ניתן לראות בקוד עצמו

**⚡ ⚡ DisconnectedError**

- ⚡ ⚡ **DisconnectedError(Throwable)**
- ⚡ ⚡ **DisconnectedError()**
- ⚡ ⚡ **DisconnectedError(String)**

```
public class DisconnectedError  
  
extends MyError represents a disconnection error.
```

**⚡ ⚡ DBError**

- ⚡ ⚡ **DBError(String, Throwable)**
- ⚡ ⚡ **DBError(Throwable)**

```
public class DBError  
  
extends MyError represents a database error.
```

**✚ ✚ EnvManager**

- ⚡ ⚡ **handledErr(MyError) void**
- ⚡ ⚡ **criticalErr(MyError) void**

```
public interface EnvManager represents an object that acts as an Environment Manager. logging  
handled errors as they occur, and safely shutting down if an unhandled error is thrown.
```

void **handledErr**(MyError err) notifies manager of a managed error

**Parameters:** err - the error thrown

`void criticalErr(MyError err)` notifies manager of an un-handleable error. triggering a shutdown

**Parameters:** err - the error thrown

ErrorHandler	
<code>ignore(ThrowingRunnable)</code>	<code>boolean</code>
<code>handle(MyError)</code>	<code>void</code>

`public interface ErrorHandler` represents a callback that can handle a certain type of errors.

`ignore` static boolean `ignore(ThrowingRunnable runnable)` Ignore any error that might get thrown while running the runnable.

**Parameters:** runnable - the runnable to run **Returns:** true if the runnable threw, false otherwise

`void handle(MyError err)` handle an error that was thrown.

**Parameters:** err - the error thrown

ThrowingRunnable	
<code>run()</code>	<code>void</code>

`public interface ThrowingRunnable` represents a runnable that might throw an error.

`void run()` Run.

**Throws:** `Throwable` - the throwable that might get thrown

UI

FontManager		
FontManager()		
defaultLinkLbl	Font	
statistics	Font	
xLarge	Font	
dbResponseTable	Font	
normal	Font	
statusLbl	Font	
backOk	Font	
error	Font	
large	Font	
...		

```
public class FontManager
```

represents the Font manager.

GameView		
GameView()		
font	Font	
textArea	JTextArea	
update(Board)	void	

```
public class GameView
```

extends JFrame represents a debugging window for viewing a game's status.

void **update**(Board board) Update the view with a new board position.

**Parameters:** board - the board

LinkLabel	
LinkLabel(String, VoidCallback)	
hoverClr	Color
normalClr	Color
setText(String)	void

```
public class LinkLabel
```

extends MyLbl represents a link label.

את שאר הפעולות ניתן לראות בקוד עצמו

MyJFrame	
MyJFrame()	
onClose	Closing<?>
resizeDelayInMs	int
myAdapter	MyAdapter
doXClick()	void
toggleFullscreen()	void
setOnExit(Closing<?>)	void
getMyAdapter()	MyAdapter
setOnResize(VoidCallback)	void
addBorderRec(Container)	void
...	

```
public class MyJFrame
```

extends JFrame represents my implementation of a window.

void **toggleFullscreen()** Toggle fullscreen.

```
void setOnExit(Closing<?> onClose) Sets on exit.
```

**Parameters:** onClose - the on close

```
void addResizeEvent(JRootPane rootPane, VoidCallback onResize) Add a callback to call on a resize of a root pane. When a resize event is invoked, a delay of 100ms will limit the number of unnecessary calls while a resize is taking place, and will offer a good tradeoff between snappiness, and performance.
```

**Parameters:** rootPane - the root pane to add the callback to. onResize - the callback to call on a resize.

את שאר הפעולות ניתן לראות בקוד עצמו

<b>Closing&lt;T&gt;</b>	
	<b>header</b> String
	<b>icon</b> ImageIcon
	<b>title</b> String
	<b>tryClose()</b> void
	<b>show()</b> T
	<b>checkClosingVal(T)</b> boolean
	<b>closing(T)</b> void

**Type Parameters:** T - the input type

```
public interface Closing<T> represents a user input confirmation for closing a window.
```

```
default void tryClose() show the input dialog and if the input verifies, close the window.
```

T **show()** Show the input dialog.

**Returns:** the input value

```
boolean checkClosingVal(T val) check an input value supplied by the user.
```

**Parameters:** val - the value **Returns:** true if the window can close

void **closing**(T val) close the window.

**Parameters:** val - the value the user submitted in the input dialog

StringClosing	
initialValue()	String
show()	String
checkClosingVal(String)	boolean

**All Superinterfaces:** Closing<String>

public interface **StringClosing**

extends Closing<String> represents a string input for closing a window.

**show** default String **show()** Show input dialog.

**Specified by:** show in interface Closing<String> **Returns:** the input

**checkClosingVal** default boolean **checkClosingVal**(String val) Check if the input should close the window.

**Specified by:** checkClosingVal in interface Closing<String> **Parameters:** val - the input value  
**Returns:** true if the window should close

String **initialValue**() the initial value of the input field.

**Returns:** the initial value of the input field

BooleanClosing	
	closing(Boolean)
	void
	checkClosingVal(Boolean)
	boolean
	show()
	Boolean
	closing()
	void

All Superinterfaces: Closing<Boolean>

public interface BooleanClosing

extends Closing<Boolean> represents a boolean input closing dialog. with yes\no options

**show** default Boolean **show()** Show the input dialog.

**Specified by:** show in interface Closing<Boolean> **Returns:** true if the user selected yes

**checkClosingVal** default boolean **checkClosingVal**(Boolean val) **Description copied from interface:** Closing check an input value supplied by the user.

**Specified by:** checkClosingVal in interface Closing<Boolean> **Parameters:** val - the value **Returns:** true if the window can close

**closing** default void **closing**(Boolean val) **Description copied from interface:** Closing close the window.

**Specified by:** closing in interface Closing<Boolean> **Parameters:** val - the value the user submitted in the input dialog

void **closing()** the user selected 'yes' and the window should close

MyAdapter	
MyAdapter()	
pressedKeys	Set<Integer>
heldDownMap	Map<Integer, HeldDown>
lastPressedKey	Integer
coolDown	long
actions	Map<Set<Integer>, VoidCallback>
lastPressedTime	long
addAction(VoidCallback, Integer[])	Set<Integer>
keyPressed(KeyEvent)	void
keyReleased(KeyEvent)	void
...	

```
public class MyAdapter
```

extends KeyAdapter represents a key adapter

void **addHeldDown**(MyAdapter.HeldDown heldDown) Add action on held down.

**Parameters:** heldDown - the action

**addAction** public Set<Integer> **addAction**(VoidCallback action, Integer... keys)  
Add action set.

**Parameters:** action - the action keys - the keys **Returns:** the set

void **removeAction**(Set<Integer> action) Remove action.

**Parameters:** action - the action

את שאר הפעולות ניתן לראות בקובץ עצמו

FontManager		
FontManager()		
dbResponseTable	Font	
small	Font	
error	Font	
sidePanel	Font	
coordinates	Font	
statistics	Font	
defaultLinkLbl	Font	
large	Font	
dbResponseTableHeader	Font	
...		

```
public class FontManager
```

fonts manager

MyLbl		
MyLbl(String, Font)		
MyLbl(String, Icon, int)		
MyLbl()		
MyLbl(String)		
MyLbl(Icon, Font)		
modifier	StringModifier	
setText(String)	void	
underline()	void	
setAllSizes(Dimension)	void	

```
public class MyLbl
```

extends JLabel represents a label.

את שאר הפעולות ניתן לראות בקוד עצמו

## Utils

StrUtils	
<b>StrUtils()</b>	
<b>htmlNewLines(String)</b>	String
<b>clean(String)</b>	String
<b>formatDate(String)</b>	String
<b>uppercase(String)</b>	String
<b>formatDate(Date)</b>	String
<b>createTimeStr(long)</b>	String
<b>dontCapWord(String)</b>	String
<b>fixHtml(String)</b>	String
<b>countMatches(String, String)</b>	int
...	

```
public class StrUtils
```

utility class for `String` related utilities

`boolean isAbsoluteUrl(String urlString)` Is the given string an absolute url.

**Parameters:** urlString - the url string **Returns:** true if the string is an absolute url

`int countMatches(String str, String regex)` Count number of matches in a string.

**Parameters:** str - the str **regex** - the **Returns:** the number of matches

`boolean isEmpty(String str)` Is a string empty.

**Parameters:** str - the string **Returns:** true if the string is empty

`String strINN(Object... objs)` create a string of the `Object.toString()` of any object that isn't

null.

**Parameters:** objs - the objs **Returns:** the string

את שאר הפעולות ניתן לראות בקובץ עצמו

RegEx	
RegEx(String, String, String[])	
RegEx(String, String, boolean, String[])	
Password	RegEx
details	String
Username	RegEx
DontSaveGame	RegEx
URL	RegEx
Numbers	RegEx
Icon	RegEx
useDontMatch	boolean

```
public class RegEx

implements Serializable Regex.
```

את שאר הפעולות ניתן לראות בקובץ עצמו

MathUtils	
MathUtils()	
log(double, int)	double
log2(double)	double

```
public class MathUtils
```

Math utility class.

```
log2 public static double log2(double num) Log in base 2.
```

**Parameters:** num - the number **Returns:** the result

```
double log(double num, int base) Log.
```

**Parameters:** num - the number base - the base **Returns:** the result



```
public class ArrUtils
```

Array Utility Class.

```
concat<T> T[] concat(T[] array1, T... array2) concatenate two arrays credit
```

**Type Parameters:** T - the type of the arrays **Parameters:** array1 - the first array array2 - the second array **Returns:** an array containing all the objects from both arrays

```
ArrayList<T> createList(Supplier<T> objCreator, int size) Create a list of objects in a given size, using a supplier to create each object.
```

**Type Parameters:** T - the type of the objects **Parameters:** objCreator - the supplier size - the size of the returned list **Returns:** the list of the objects

```
T exists(T[] arr, int... index) if an item exists, it will be returned. if it doesn't null will be returned
```

**Type Parameters:** T - the type of the array **Parameters:** arr - the array index - the index **Returns:** the item if it exists, null otherwise.

ArgsUtil	
ArgsUtil(String[])	
streamSupplier Supplier<Stream<String>>	
equalsSign(String) OptionalArg	
create(String[]) ArgsUtil	
plainTextIgnoreCase(String) OptionalArg	

```
public class ArgsUtil
```

represents a utility for integrating with `String[]` arguments

`ArgsUtil create(String[] args)` Create args util.

**Parameters:** args - the args **Returns:** the args util

`equalsSign` public `ArgsUtil.OptionalArg equalsSign(String preEqualStr)` for any arg of this format: `preEqualStr=%argval%`

**Parameters:** preEqualStr - the pre equal str **Returns:** the optional arg value(assuming there is one) `%argval%` in the example above

את שאר הפעולות ניתן לראות בקובץ עצמו

OptionalArg	
OptionalArg(String)	
str String	
getBoolean(Boolean) Boolean	
exists() boolean	
checkExists() void	
str() String	
getString() String	
getInt() int	
getInt(int) int	

```
public static record ArgsUtil.OptionalArg(String str)
```

represents an argument that might have been passed.

`String getString()` Gets the string value of this argument.

**Returns:** the string value of the argument if it was found, null otherwise.

`boolean exists()` was this argument found

**Returns:** true if the argument was found

את שאר הפעולות ניתן לראות בקובץ עצמו

## 4.5 אלגוריתמים ופעולות נבחרות

### 4.5.1 אלגוריתם מינימקס עם גיזום

#### הצגת האלגוריתם – מטרתו ואופן פעולה

מטרת האלגוריתם הוא למצוא את המהלך הכי טוב בכל עמדה נתונה. והוא עושים זאת באמצעות ניקוד כל אחד מהלוחות שנוצרים לאחר כל אחד מההלכים האפשריים, ובבחירה המהלך שהוביל ללוח הכי טוב עבור השחקן שתורו לשחק.

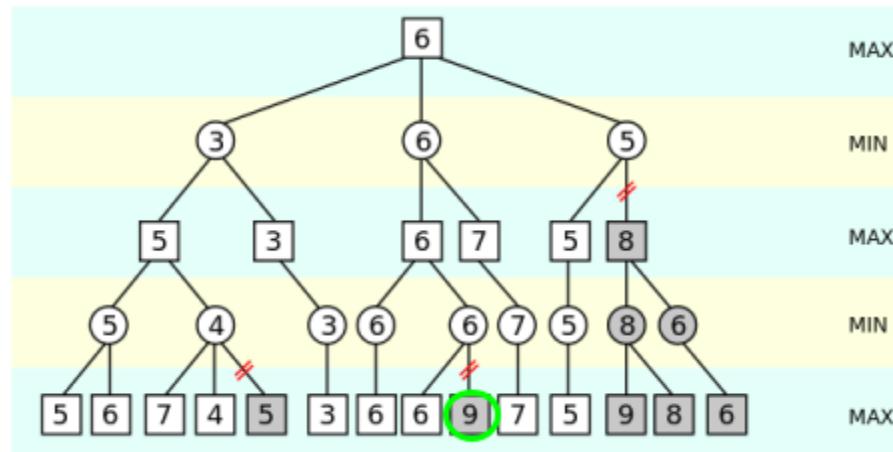
עבור עמדה נתונה, האלגוריתם עובר על כל המהלים האפשריים לשחקן הנוכחי, עושה אותם, ואז עושה את כל המהלים של היריב. בסוף משחק (תיקו, ניצחון או הפסד) הוא מחזיר ערך עבור העמדה הסופית. אותו ערך יהיה מאד גבוהה עבור ניצחון, מאד נמוך עבור הפסד, וניטרלי עבור תיקו. ואז חוזר במצבה רקורסיבית, ובוחר את המהלים שהשחקנים ישחקו במצבה אידיאלית עבור כל אחד. הרעיון הוא ש כדי למצוא את המהלך הכי טוב, חיברים להניח שהיריב ישחק במצבה מושלמת. וכך ניתן למצוא את המהלך הכי טוב בכל עמדה אפשרית.

גיזום

כדי להעריך את העמדות בצורה הכי מדויקת שאפשר, אנו רוצים ליעיל את החיפוש ככל האפשר, ולהסוך בחיפושים שלא נצרכים. לשם כך ננסה לגלוות متى חיפוש عمדה מסוימת יהיה מיותר, ולגוזם את החיפוש זהה. זו לא לטרואה לחפש בענף זה כלל, וכך להסוך זמן יקר.

### אלפה בטא

גיזום אלפא בטא הינו גיזום שמטרתו לחסוך חיפוש שכ"מ ייפסל כשיוזר מהركורסיה. לדוגמה:



כשהעץ הגיע ל-9 שסמן בירוק, ברור שהוא לא ייבחר. מאוחר והוא יותר גדול מהערך המינימלי שנמצא עד עכשיו. וברור שהערך שיוזר יהיה קטן שווה ל-6. וה-9 נמצא בשכבה מקסימום, ולכן הערך שיוזר ממנו יהיה גדול שווה מ-9. ואפשר לפолос את החיפוש בענף הזה למורי.

### באיזה מקום בפרויקט נעשה שימוש באלגוריתם

האלגוריתם ממומש בקוד הפרויקט ע"י המחלקה Minimax.java שנעוזרת באלגוריתם להערכת מצב (פונקציית הערכה שתוסבר בסעיף הבא) שמנקד לווה משחק – עד כמה הוא טוב לשחקן.

### הציג קוד האלגוריתם

להלן (עיקר) קוד האלגוריתם:

```
private Evaluation minimax(MinimaxParameters parms) {
    if (interrupt != null) {
        throw interrupt;
    }
    if (isOvertime() || parms.currentDepth >= parms.maxDepth || Eval.isGameOver(parms.model)) {
        Evaluation evaluation = Eval.getEvaluation(parms.model,
            parms.minimaxPlayerColor);
        evaluation.setEvaluationDepth(parms.currentDepth);
        return evaluation;
    }
}
```

```

Evaluation bestEval = null;
ArrayList<Move> possibleMoves =
MoveGenerator.generateMoves(parms.model, GenerationSettings.LEGALIZE);

if (parms.currentDepth < MOVE_ORDERING_DEPTH_CUTOFF)
    sortMoves(possibleMoves, true);

for (int i = 0, possibleMovesSize = possibleMoves.size(); i <
possibleMovesSize; i++) {
    Move move = possibleMoves.get(i);

    if (interrupt != null) {
        throw interrupt;
    }
    parms.model.applyMove(move);

    Evaluation eval = minimax(parms.nextDepth());
    parms.model.undoMove(move);

    if (bestEval == null || eval.isGreaterThan(bestEval) ==
parms.isMax) {
        bestEval = eval;
    }

    if (parms.prune(bestEval)) {
        break;
    }
}

return bestEval;
}

```

### ניתוח יעילות האלגוריתם

המספר המהלים החוקיים שיויצאים מכל עמדת חוקית (הוא ממוצע 33 (במשחקים כמו שחמט שבhem מס' מהלים החוקיים עבר כל עמדה לא אחד, מחשבים ממוצע). מה שMOVAIL למספר אסטרונומי של עדות לבחון כבר בעומקים יחסית נוכחים. לדוגמה, אם משתמש ממוצע, בעומק 6 יש כבר יותר ממיליארד עדות לבחון. ועם כח המחשב שלנו היום, להמשיך לרוחב בעומק עד שmaguiim לסופם משחק זה כמעט אף פעם לא אפשרי. וכך חיברים להגביל את החיפוש בצורה מסוימת.

הhiposh המינימקס 'יציר' עץ DFS (הhiposh לא באמת יוצר עץ, אלא שהוא עובד בצורה של צלילה שמאלה, ובחזרה הוא פונה ימינה. בדומה לעץ DFS) אם הינו מנהים לאלגוריתם לפועל ככה, ומגבילים את החיפוש בזמן מסוים (כדי שהhiposh גם יעוצר מתיישהו), הינו נתקלים בבעיה. בשל טבע החיפוש, העץ עלול להיות רק עדות שיצאו מהלך אחד\כמה בצורה הרבה יותר עמוקה (ומדויקת), ולנוטש לגמרי\כמעט לגמרי עדות שיצאו מהצד הימני של העץ. כל זה בגל הנטייה שלו ליכת עד הסוף שמאלה, ורק בחזרה לפנות ימינה. והמצב שעלול להווצר הוא שבעוד העץ עמוק שמאלה, נגמר לו הזמן. והוא חייב להחזיר תוצאה. כדי לפתור בעיה זו, נפעיל את המינימקס בצורה איטרטיבית. ז"א שקדם כל נרי'ן אותו עד לעומק 1,

הוא ירד שמאליה, יחזור, יעבור ימינה, ויחזר תוצאה. כעת, אם נשאר זמן, אפשר לשלווה אותו לעומק 2, ירד שמאליה, יחזור, יעבור ימינה וכן הלאה. ככה שהוא לא יוכל להפלות את הענפים הימניים לרעה (המקסימום שהוא יכול להפלות אותו לרעה יהיה בעומק 1, במקרה שבו הוא נאלץ לעצור באמצע החיפוש).

## 4.5.2 פונקציית הערקה

### הציגת האלגוריתם – מטרתו ואופן פעולה

כדי לנוהל את המשחק, צריכה להיות דרך לאחות סיום משחק. וכשען המינימקס נבנה, כל מצב סופי מקבל ציון הערקה כדי שאלגוריתם המינימקס יוכל לקבל החלטה. אם המצב הסופי הוא סיום משחק – ניצחון/פסד/תיקו – אז הניקוד ברור וקל. אך אם המצב לא סיום משחק, יש צורך לנתק את המצב זהה המשקף עד כמה הוא טוב עבור השחקן שתורו לשחק. כמעט בשום עמדה לא יהיה ניתן לרדת עד לסוף העז. בשלב כלשהו חיברים לעצור ולנסות "לשער" מהו ניקוד העמדה. לשם כך אנו משתמשים במחלקה Eval. שמחשבת ניקוד עבור עמדה מסוימת. לכלcoli יש ערך שנמדד ע"פ מידות centipawns (מאיות רגלי). הרגלי שווה 100, וכל שאר ערכי ה *evaluation* מנוקדים בהתאם למידה זו. ז"א שאם למשל על마다 מסויימת קיבל ניקוד של 500, ניתן להסיק שהיתרונו שיש לשחקן שווה ערך לעמדה שבה יש לו 5 رجالים יותר מיריבו. לכל אחת מהמטריקות שבהם השתמשתי יש משקל (עלול להיות שונה ממה שמצוין כאן בשל מטיב הקוד), שקובע באיזה מידה הם משפיעים על הציון הסופי. המטריקות:

- **מספר הכלים**  
נתחיל במטריקה הבסיסית ביותר. ערך הכלים. גגלי שווה 100, צריח שווה בערך 5 رجالים, וכן ערכו 500. פרש שווה קצת יותר מ 3 رجالים וכן ערכו 310. רץ שווה קצת יותר מהפרש, וכן ערכו 320. והמלכה שווה 900. אפשר לטעון שהמלך שווה לאינסוף, וכן הוא יכול Zion מאד מוגזם. חישוב הערך הוא תהליך פשוט של סכום הכלים של כל אחד מסוגי הכלים של כל אחד מסוגי השחקנים.

משקל מספר הכלים הינו 1.5.

- **טבלאות כלים**  
לכל אחד מהכלים ישנו משבצות שעלייהם הוא יותר ופחות טוב. (זו כמובן הכלילה, שלא יכולה להיות נכונה בכ-100% מהפעמים, אך חיברים לעשות זאת. ובסקול הכלל הרעיון הוא שהמטריקות יcssו על ה"শטחים המתים" אחת על השניה). לשם כך חישבו ציונים עבור כל משבצת עבור כל כלי, וכל הכלים של השחקנים משתקלים לפי ערך המשבצת שבה הם נמצאים. לאחר מכן משבצת מסוימת עבור כל מטיים משתנה מאוד לקרה, ובמיוחד במהלך המשחק, השלב הסופי של המשחק, ישנה טבלת ציונים ייחודית שמותאמת לסוף המשחק, וכל שהוא מתקרב, המשקל שלו עולה.  
משקל טבלאות הכלים הינו 1.1.

- **אלוץ המלך לפינה**  
לקראת סוף המשחק, בד"כ, המלך "חוצה" להתקדם למרדף. כדי להיות יותר מועל, כדי שייהיו לו הרבה מהלכים, ולא יוכל להילך בקהלות. לכן ככל שהמלך מתקדם לסופו, ישנו יותר ויותר משקל לאילוץ המלך של היריב לפינה. לאחר והוא בד"כ הרבה הרבה פחות פעיל כשהוא בפינה. ובנוספ על כן, בד"כ הרבה יותר קל לעשות מעט מלך שנמצא בפינה.  
משקל אלוץ המלך לפינה הינו 1.5.

ישן עוד מטריקות שנייסטי, אך כרגע אשר עם השלושה שצ'ינתי.

### הציג קוד האלגוריתם

להלן (עיקר) קוד האלגוריתם:

בדיקה האם נגמר המשחק:

```

private Evaluation checkGameOver() {
    if (!model.anyLegalMove(playerToMove)) {
        if (model.isInCheck(playerToMove)) {
            return new Evaluation(GameStatus.checkmate(playerToMove.getOpponent(),
                model.getKing(playerToMove)), playerToMove);
        }
        return new Evaluation(GameStatus.stalemate(),
            playerToMove);
    }

    if (model.getHalfMoveClock() >= 100) {
        return new Evaluation(GameStatus.fiftyMoveRule(),
            playerToMove);
    }

    if (checkRepetition()) {
        return new Evaluation(GameStatus.threeFoldRepetition(), playerToMove);
    }

    if (checkForInsufficientMaterial()) {
        return new Evaluation(GameStatus.insufficientMaterial(), playerToMove);
    }

    return new Evaluation(playerToMove);
}

private boolean checkRepetition() {
    var stack = model.getMoveStack();

```

```

if (stack.size() < 8)
    return false;

ArrayList<Long> list = new ArrayList<>();
for (int i = stack.size() - 1; i >= 0; i -= 2) {
    var move = stack.get(i);
    if (!move.isReversible())
        break;
    long l = move.getCreatedListHashSupplier().get();
    list.add(l);
}

if (list.size() < 4)
    return false;
for (int i = 0; i < list.size(); i++) {
    int matches = 0;
    long current = list.get(i);
    for (int j = i + 1; j < list.size(); j++) {
        if (list.get(j) == current) {
            matches++;
            if (matches == 2) { //a repetition is found
                return true;
            }
        }
    }
}
return false;
}

private boolean checkForInsufficientMaterial() {
    return insufficientMaterial(PlayerColor.WHITE, model) &&
        insufficientMaterial(PlayerColor.BLACK, model);
}

```

```
private static boolean insufficientMaterial(PlayerColor playerColor, Model model) {
    return (
        model.getNumberOfPieces(playerColor, PieceType.PAWN) ==
0 &&
        model.getNumberOfPieces(playerColor,
PieceType.MINOR_PIECES) <= 1 &&
        model.getNumberOfPieces(playerColor,
PieceType.MAJOR_PIECES) == 0);
}
```

במקרה שבו לא היה סוף משחק

```
private void calcEvaluation() {
    evaluation = new Evaluation(PlayerColor.WHITE);

    if (model.isInCheck()) {
        evaluation.getGameStatus().setInCheck(model.getKing());
    }

    evaluation.addDetail(EvaluationParameters.MATERIAL,
materialSum(PlayerColor.WHITE) -
materialSum(PlayerColor.BLACK));

    evaluation.addDetail(EvaluationParameters.PIECE_TABLES,
materialSum(PlayerColor.WHITE) -
materialSum(PlayerColor.BLACK));
}
```

```

evaluation.addDetail(EvaluationParameters.FORCE_KING_TO_CORNER
, forceKingToCorner(egWeight, PlayerColor.WHITE) -
forceKingToCorner(egWeight, PlayerColor.BLACK));
}

private int materialSum(PlayerColor playerColor) {
    int ret = 0;
    int[] piecesCount = model.getPiecesCount(playerColor);
    for (int i = 0, piecesCountLength = piecesCount.length;
i < piecesCountLength; i++) {
        int count = piecesCount[i];
        ret += count * PieceType.getPieceType(i).value;

    }
    return ret;
}

private int forceKingToCorner(double egWeight, PlayerColor
playerColor) {

    if (egWeight == 0)
        return 0;
    int ret = 0;
    Location opK = model.getKing(playerColor.getOpponent());
    int opRow = opK.row, opCol = opK.col;
    int opDstToCenterCol = Math.max(3 - opCol, opCol - 4);
    int opDstToCenterRow = Math.max(3 - opRow, opRow - 4);
}

```

```
        int opKDstFromCenter = opDstToCenterCol +  
opDstToCenterRow;  
        ret += opKDstFromCenter;  
  
        Location myK = model.getKing(playerColor);  
  
        int myRow = myK.row, myCol = myK.col;  
  
        int kingsColDst = Math.abs(myCol - opCol);  
        int kingsRowDst = Math.abs(myRow - opRow);  
        int kingsDst = kingsColDst + kingsRowDst;  
        ret += 14 - kingsDst;  
  
    return (int) (ret * egWeight);  
}
```

### ניתוחיעילות האלגוריתם

בדיקה האם נגמר המשחק:

כמוות המהלך שהשחקן שטורו לזרז יכול לבצע  $n$

$O(n)$

אם לא נגמר המשחק:

בדיקה השח:

$O(n)$

ספרת כלים:

$O(n)$

טבלאות כלים:

$O(n)$

אילוץ מלך לפינה:

$O(1)$

### 4.5.3 אלגוריתם 3 חישוב אלטרנטיבות לשם משתמש תפוא

כשלקוח מנסה להירשם בתור שחקן חדש, ובוחר שם משתמש שכבר קיים בשרת, רציתי לחשב כל מיני אלטרנטיבות זמינות לשם המשתמש שהוא ניסה לבחר ולחשוף לו אתן.

לשם כך נכתבת מחלוקת UsernameSuggestions. שמקבלת שם משתמש, ומנסה ליצור אלטרנטיבות זמינות סבירו. ההליך יוצרת האלטרנטיבות מתבצע בצורה זו: מהרזה RegEx מוצאת התבנית, ובאמצעות שימוש המשק MatchIterations והפעולה createStr שמקבלת כפרמטר את המחרוזת שנמצאה, ואת מספר האיטרציה הנוכחית של ייצור אלטרנטיבות (יצירת האלטרנטיבות מתבצעת בלולאה לפי קבוע שקבע כמוינט). אלטרנטיבות צרכות להיווצר עבור כל אחד מיוצר האלטרנטיבות. לדוגמה: אם התבנית RegEx מוצאת מספר במחרוזת, וווצר האלטרנטיבות מעלה אותו באחד. כדי ליזור קוד דינامي יותר, יוצר האלטרנטיבות לא עלה את המספר באחד, אלא במספר האיטרציה הנוכחי. כדי שנוכל לצורך, לדוגמה עבור 0-test-0: גם את test-1 וגם את 2-test-1 וכו'. במקום רק את test-1), וכך ניתן ליצור הרבה הרבה סוגים אלטרנטיבות בקהלות.

אחד הקשיים באלגוריתם זה היה חיפוש לרוחב של שמות משתמשים שלא נמצאים בשימוש, כדי להציג ללקוח רשימת אופציות מגוננת ככל האפשר. הייתה יכול לבדוק את כל האופציות ולקחת מילוק מגוון ככל האפשר מהאופציות הזמינות, אבל מאחר ואני רוצה לשמר על מספר הקריאות למינימום למסגרת האפשרי, אנסה למצוא את הקומבינציה הכי מגוננת שעונה על מספר האופציות הנדרשות עם מינימום קריאות למסגרת הנתונים.

#### הצגת קוד האלגוריתם

להלן קוד האלגוריתם:

```
public static ArrayList<String> createSuggestions(String username) {
    ArrayList<ArrayList<String>> options = new ArrayList<>();
    options.addAll(createOptions("([0-9]+)|([0-9])", username,
        (matchGroup, i) -> {
            int num = Integer.parseInt(matchGroup.group()) + i;
            return num + "";
        }, (matchGroup, i) -> {
            int num = Integer.parseInt(matchGroup.group()) - i;
            return num + "";
        }, preUnderscore, postUnderscore, bothUnderscore));
    options.addAll(createOptions("[A-Z]", username, preUnderscore,
        postUnderscore, bothUnderscore));
    options.addAll(createOptions("(^|[a-z])|(([0-9]|_)|[a-z])",
        username, upper, lower));
}
```

```

        options.addAll(createOptions("(.*)^", username, (matchGroup, i) ->
matchGroup.group() + Math.abs(new Random().nextInt())));
    }

    return createResults(options);
}

private static ArrayList<ArrayList<String>>
createOptions(@Language("RegExp") String regex, String username,
MatchIterations... matchIterations) {
    ArrayList<ArrayList<String>> options = new ArrayList<>();
    Pattern pattern = Pattern.compile(regex);
    pattern.matcher(username).results().forEach(match -> {
        ArrayList<String>[] arr = new
ArrayList[matchIterations.length];
        Arrays.setAll(arr, i -> new ArrayList<>());
        for (int i = 1; i <= numOfIterationsPerOptionGroup; i++) {
            for (int j = 0; j < matchIterations.length; j++) {
                MatchIterations iterations = matchIterations[j];
                String currentIterationStr =
iterations.createStr(match, i);
                String str = username.substring(0,
match.start()).concat(currentIterationStr).concat(username.substring(matc
h.end()));
                arr[j].add(str);
            }
        }
        options.addAll(Arrays.stream(arr).toList());
    });
    return options;
}
private static ArrayList<String>
createResults(ArrayList<ArrayList<String>> optionsLists) {
    ArrayList<String> suggestions = new ArrayList<>();
    int[] indices = new int[optionsLists.size()];
    Arrays.fill(indices, 0);
    int numOfSearchedLists = 0;
    outer:
    while (numOfSearchedLists < optionsLists.size()) {

```

```

        for (int i = 0; i < optionsLists.size(); i++) {
            ArrayList<String> options = optionsLists.get(i);
            boolean keepSearching = true;
            if (options.isEmpty()) {
                keepSearching = false;
                numOfSearchedLists++;
            }
            while (keepSearching && indices[i] < options.size()) {
                String suggestion = options.get(indices[i]++;
                if (!suggestions.contains(suggestion) &&
!DB.isUsernameExists(suggestion)) {
                    suggestions.add(suggestion);
                    if (suggestions.size() >= maxSuggestions) {
                        break outer;
                    }
                    keepSearching = false;
                }
                if (indices[i] >= options.size()) {
                    keepSearching = false;
                    numOfSearchedLists++;
                }
            }
        }
        return suggestions;
    }

    private interface MatchIterations {
        String createStr(MatchResult result, int iteration);
    }
}

```

ניתוח יעילות האלגוריתם

מספר הפעמים שתבנית תמצא במחuzeות  $n$

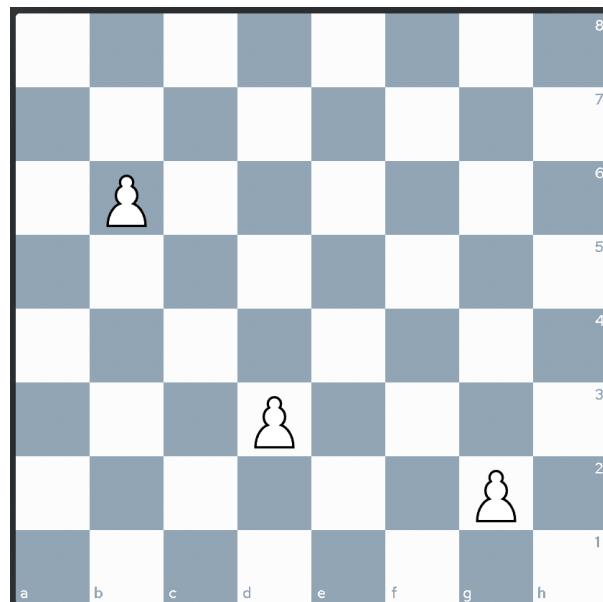
מספר הניסיונות שייצרו קבוע, ולכן זמן הריצה זהה לינארי, זמן הריצה של בדיקת הניסיונות מול מסד הנתונים הינו:

$O(n^2)$

#### 4.5.4 אלגוריתם/פעולה 4 חישוב אינומים של כלים

אחד האלגוריתמים החשובים במשחק שחמט, זו הבדיקה האם משכנת מסוימת מאוימת. כדי לבדוק האם למלך מותר ללכת אליה, או האם שחקו נמצא בשח. כדי ליעיל את החישוב הזה השתמשתי בלחוחות ביטים (אובייקט שמייצג את הלוח בצורה בינארית. האם יש או אין כלי עboro כל משכנת). עבור כל סוג כלי והצעע שלו שמרתני לוח ביטי. ואז באמצעות היסט, שבעצם מזינו את כל הלוח לפי אופסט שהוא כיוון תקיפה של אותו כלי, אני יכול לקבל את כל המשכנות שאותו סוג כל תוקף. ואז בדיקה של & עם ייצוג ביטורי של מיקום של משכנת (מספר ארוך עם בית דלוק באינדקס המשכנת. המשכנת הכיו שמאלית למעלה (A8) מוצגת ע"י בית 0, והמשכנת הכיו ימנית למטה (H1) ע"י בית 63) תגיד לי האם אותה משכנת מאוימת ע"י אותו סוג כל או לא. ככה שאני לא צריך לחשב את המשכנות עבור כל כלי, אלא בקבוצות של סוגי כלים. גם החישוב נעשה בצורה הרבה יותר יעילה.

לדוגמה:



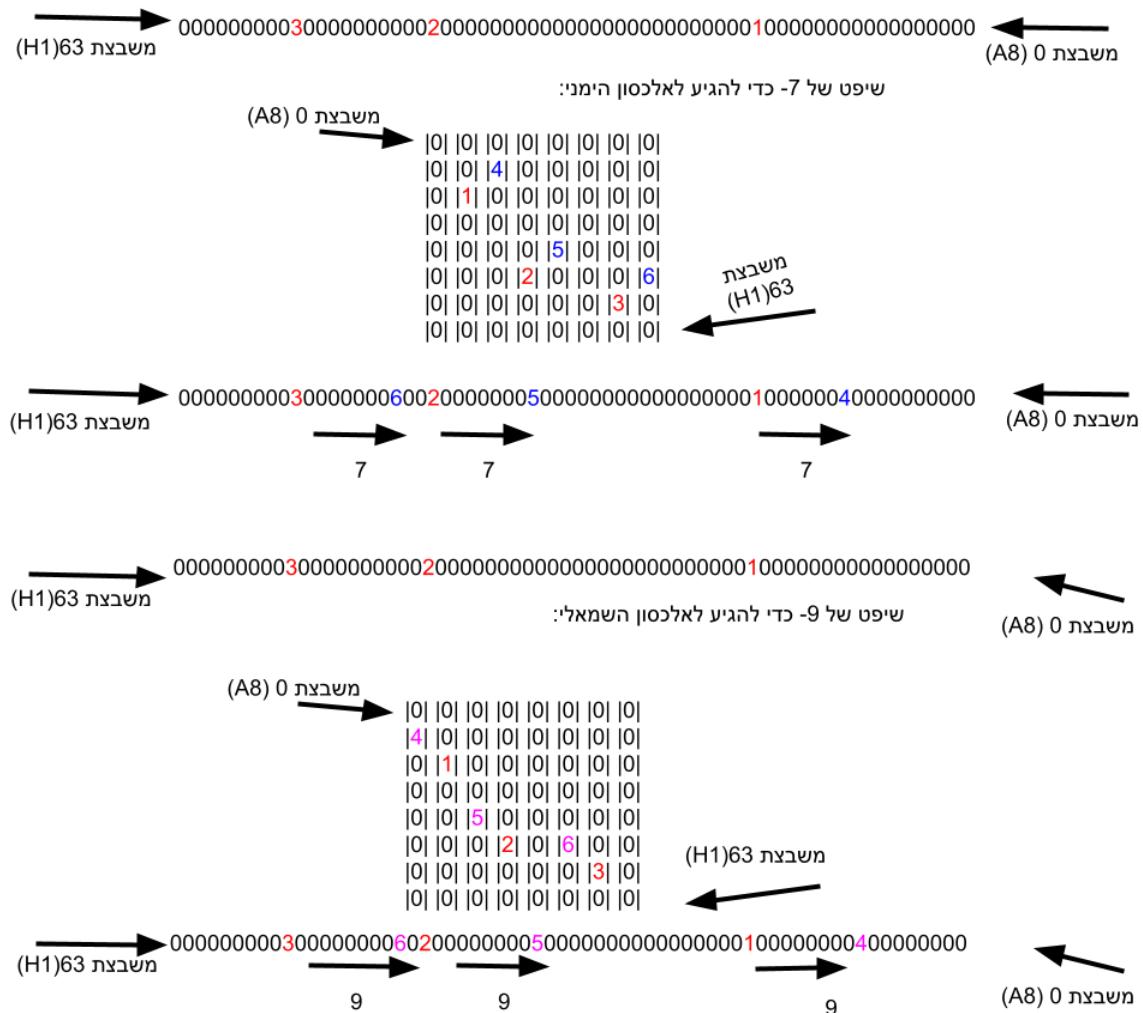
להלן הביטוי של הרגלים של השחקן הלבן של הלוח הזה נראה ככזה:  
0000000001000000000001000  
או בגרסה קצרה יותר ידידותית לבני אדם:

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

כדי לחשב את המשבצות אותן הרגלים תוקפים, צריך לעשות שיפט-ים בכיוונים שבהם רגליים תוקפים. הרעיון הוא שהתזוזה היא איחוד. חוץ ממקרי קצה (ליטרלי) שבהם הכלי נמצא בקצה הלוח ולא יכול לזרז בכיוון ההפוך, הכלי שנמצא ב3d עובד בדיקון כמו אותו סוג הכלי שנמצא ב4h או כל משbatchת אחרת. היחידים שבהם זה שונה הם הרגלים והמלכים. שלהם יש מהלכים מיוחדים לפי המיקום שלהם בלוח (הצרכה, קידום של רגלי לכלי אחר, צעד כפול, וEn Passant). אך כל אותם מהלכים לא משנה את צורת האכילה של אותן כלים.

הצרכה וצעד כפול: לא יכולה להיות אכילה. קידום וEn Passant: לא משנה את צורת האכילה של הרגלים. הם עדין תוקפים את המשבצות ישר באלכסון מהם.

במקרה של הדוגמא, רגליים תוקפים ב2 צורות. האלכסון לימין ולשמאלו. ולכן החישוב יראה כך: (המספרים השונים נועדו לצורך הדוגמה. כמובן שבאמת 4,5 ו-6 הם אחדים. ו-1,2 ו-3 לא נמצאים כלל. מאחר שהם נמצאים רק בלוח המקורי)



כל התוצאות נשמרות בתוך ביטבורד סופי. שבסקרה של הדוגמא זו ייראה כך:



מצגת קוד האלגוריתם

להלן (עיקר) קוד האלגוריתם:

```

private boolean isAttacked(Location loc) {
    PieceType[] piece_types = PieceType.ATTACKING_PIECE_TYPES;
    this.checkingAttacked = loc;
    for (PieceType pieceType : piece_types) {
        attack(pieceType);
        if (attackedSquares.isSet(loc))
            return true;
    }
    return false;
}

private void attack(PieceType pieceType, Bitboard attackingPiecesBB,
Direction... attackingDirections) {
    if (attackingDirections.length == 0)
        attackingDirections = pieceType.getAttackingDirections();

    for (int i = 0, attackingDirectionsLength =
attackingDirections.length; i < attackingDirectionsLength &&
(checkingAttacked == null || !attackedSquares.isSet(checkingAttacked));
i++) {
        Direction direction = attackingDirections[i];
        Bitboard pieceBB = attackingPiecesBB.cp();

        Bitboard adjustedOpponent =
attackedPlayerBB.shift(attackingPlayerColor, direction);
        do {
            attackedSquares
                .orEqual(pieceBB.shiftMe(attackingPlayerColor,
direction)
                    .exclude(attackingPieces.getAll())
                    .exclude(adjustedOpponent));
        } while (pieceType.isSliding && pieceBB.notEmpty()
&& (checkingAttacked == null ||
!attackedSquares.isSet(checkingAttacked))
);
    }
}

```

```

    }
}

```

### ניתוח יעילות האלגוריתם

מספר סוגי הכלים השונים שיש לשחקן החקוף =  $n$

מספר הכלויינים שבהם כלי תוקף =  $y$

עבור כל סוג כלי שיש לשחקן החקוף, עבור כל כיוון שבו הוא תוקף, נעשה מקסימום 64 שיפטים כדי לחשב את המשבצות שהוא תוקף באותו כיוון.  
לכן האלגוריתם פועל ביעילות של:

$O(n * y * 64)$



$O(n * y)$

### **AppSocket 5.5.5 אלגוריתם/פעולה**

את התקשרות בין הרשות ללקוח מימושי בצורה הבא:

המחלקה AppSocket מייצגת תהליך שכלי הזמן קורא הודעות. ומאפשרת שליחת בקשות (Request) ולקראת שבו מצופה לקבלת תגובה (Response) מהצד השני, מועבר גם Callback שאליו נקרא כשהודעתת תגובה תקרה. או לחלופין, במקרים מסוימים, התהליך ירצה להסום עד לקבלת תגובה. ולכן יש אופציה לשולח בקשה ולהמתין עד לקבלת תגובה.

כשהודעה מתתקבלת, היא מועברת ל-`MessageHandler`, והוא מנתב אותה בתהליך חדש בצורה זו: אם ההודעה היא תגובה לבקשת קודמת, `Callback` שהועבר ביחד עם אותה בקשה נקרא. אם לא, הטיפול בהודעות עובר למטפל שנבחר בצורה זו: להודעות יש סוג. ולכל סוג הودעה, יש `Handler` שמתפל באוטו סוג ה הודעות, שיכל להדרס ע"י כל אחד מההמשימות של `Handlers`. לדוגמה: במקרה שבו הרשות נתקל בשגיאה לא צפוייה, הוא ישלח הודעה מסוג שגיאיה, ואotta הודעה תנוטב מצד הל庫ו למטפל של הודעות מסוג שגיאיה. שתציג את הודעתת השגיאה ותשגור את הלוקו. אך במקרים שבהם שגיאת היא תגובה צפוייה, לדוגמה: כשלוקו מנסה להיכנס עם משתמש רשום, מועבר מטפל ביחיד עם הודעתת הכניסה, והטיפול בשגיאות (או בחיבור מוצלח) יבוצע בעורתו.

### הציג קוד האלגוריתם

להלן (עיקר) קוד האלגוריתם:

```

private void processMessage(Message message) {
    onAnyMsg(message);
    String respondingTo = message.getRespondingToMsgId();
    MessageCallback callback;
    synchronized (customCallbacks) {
        if (respondingTo != null &&
            customCallbacks.containsKey(respondingTo)) {
            callback = customCallbacks.remove(respondingTo);
        } else {
            callback = defaultCallbacks.get(message.getMessageType());
        }
    }
    callback.callback(message);
}

\

public Message blockTilRes(Message request) {
    CompletableFuture<Message> future = new CompletableFuture<>();
    waiting.add(future);

    Message msg = null;
    noBlockRequest(request, future::complete);
    try {
        msg = future.get();
    } catch (InterruptedException e) {
    } catch (ExecutionException e) {
        e.printStackTrace();
    }
    waiting.remove(future);

    if (msg == null)
        throw new MyError.DisconnectedError();

    if (msg.getMessageType() == MessageType.THROW_ERROR) {
        onThrowError().callback(msg);
    }

    return msg;
}

```

```

public void noBlockRequest(Message request, MessageCallback onRes) {
    customCallbacks.put(request.messageID, onRes);
    socket.writeMessage(request);
}

```

### ניתוח יעילות האלגוריתם

האלגוריתם פועל בזמן לינארי.

## 4.6 מבני נתונים עיקריים

מבנה הנתונים העיקריים שבהם השתמשתי הם:

- **ArrayList**

משמש לאחסון רשימה אובייקטים, וגישה אליהם ע"י אינדקס בזמן לינארי.  
השתמשתי בו בפרויקט:

לאחסון שחנים מחוברים, משחקים שרצו, מהלכים ששחקן יכול לבצע, ועוד.

- **HashMap**

משמש לגישה בזמן לינארי לאובייקטים שנשמרו ע"פ מפתח מסוימים. השתמשתי בו מטעמי נוחות ויעילות.

אחד השימושים היה לאחסון AppSocket-callbackים בסוגם כדי לנטר כל הודעה למטפל שלא עבר סוגיה הידועות שונות. דבר שהוסך זמן תקורה קריטי לעיבוד ההודעות.

- **מערכות**

השתמשתי בהם בפרויקט כדי לאחסן אוסףם בסדר כלשהו. לדוגמה אחסון לוחות Bitboard-ים של הכלים נעשה בתחום מערך שסודר ע"פ סוג הכלים (האינדקס שלהם).

- **מטריצות**

השתמשתי בפרויקט לאחסן טבלאות כלים כמו שתואר ב4.5 פונקציית הערכה. או לאחסן מספר כלים לפי כלי ושחקן. השתמשתי בהם כדי לשמר מידע על עמדת (שורות&עמודות) ולגשש אליהם בצורה יעילה.

- **Bitboard**

ייצוג לוח שחמט בצורה בינארית. האם יש כלי על אותה משבצת או לא. השתמשתי בו כדילייצג את הלוח במקרים סוג הכלי על אותה משבצת ידוע, או לא משנה. תיארתי את השימוש העיקרי בו ב4.5.4 אלגוריתם/פעולה 4 חישוב אינומים של כלים.

## 5. סיכום אישי

### 5.1 אתגרים וקשיים איתם התמודדתי

- **ניהול זמן**

בזמן העבודה על הפרויקט, דבר שקרה המון היה שהשकעת הזמן זמן בפרטיהם שליליים מאוד, והזנחה חלקיים הרבה יותר משמעותיים. כדי לפתור את זה השתדלתי לעצור את מה שאני עושה אחת לכמה זמן, ולשאול את עצמי האם אני מנצח את הזמן שלי בצורה ייעלה.

- **אובססיה לגבי לעשות משהו בצורה 'מושלמת'**

כשניגשתי לפתור אייזו בעיה או למשתמש איזה אלגוריתם, המחשבה שאני חייב לעשות את זה בצורה מושלמת כל הזמן חוזרת לי בראש. מה שמאז אחד הוא דבר די חביבי, אבל מהצד השני יכול לגרום ביכולת שלי בכלל לגשת לבעה. כי אני אהיה עסוק מדי בלנסות לפתור אותה בדרך "המושלמת", או לנסתות לחשב בכלל על מה הוא הפתרון המושלם. מה שבסופו של דבר גורם לבזבוז זמן ותסכול. כדי להתגבר על זה, אני משתמש קודם כל מהצורה הכיו פשטוטה שלהם, ולהגיע לפתרון קיים. ורק אחר כך לחשב על דרכם אולי למטרת הפתרון הראשוני.

- **KISS**

ראשי תיבות של *keep it super simple stupid* (יש שאומרים *keep it simple*). ביטוי שנועד להזכיר למפתחים לשמר על דברים בצורה פשוטה, בלי לסבך אותם לשוווא. ולבטל כל מרכיבות שאינה חיובית. זה דבר שקצת התקשייתי לעמוד בו במהלך כתיבת הפרויקט (מספר העמודים בספר עלול לרמז על כך) בהסתכלות אחרת, זה נראה כאילו הנטיה הראשונית שלי היא בדיקת הפה. לסבך כמה שאפשר. ככה שיצא לפעמים שמבייל לשים לב, יצרתי אייזו מערכת מורכבת ומסועפת כדי לפתור בעיה פשוטה. כדי להתגבר על זה, השתדלתי להכריח את עצמי לעצור לרגע לפני שאני מתחילה לעבוד, ולשאול את עצמי האם חשבתי על הדרכם הכי פשוטה שבה אני יכול לעשות את זה.

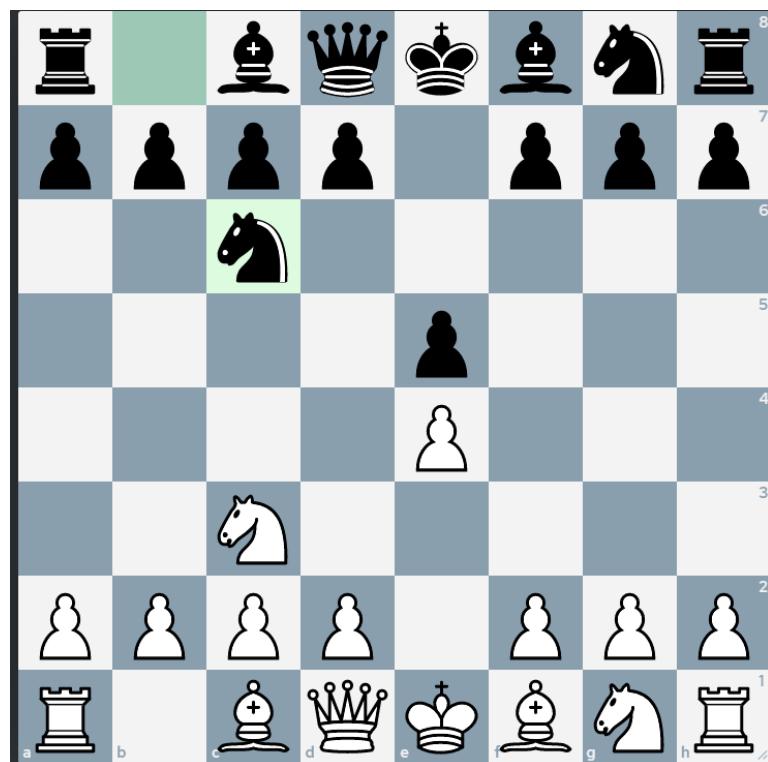
### 5.2 מה הפרויקט תרם לי

- כתיבת טסדים לקוד כדי לוודא את פועלתו, ולוודא שהדרישות נשמרו לאחר כל שינוי.
- כתיבת קוד גנרי הכל האפשר כדי לעבוד בצורה דינמית, ולמנוע חוזה על קוד.

- כתיבת תיעוד לקוד בזמן הכתיבה כדי לעזור בקריאה של הקוד, ובמציאת טעויות עיצוב ולתקן אותן בזמן הכתיבה, במקום הרבה יותר.
- שימוש בססן כדי לעקוב אחר גרסאות שונות של הקוד, ולהזור בזמן בכל רגע נדרש.

### 5.3 הצעות לשיפור

- **שיפור Eval:** Eval דיביסי, וمبוסס על מטריקות יחסית פשוטות. עיקר היתרונו של המחשב הוא בחישוב טקטיות שבובוסות על הובלה בכלים (מספר מהלכים שזכרים ברגלי, כל, או דברים בסגנון), ולא בשחק עמדתי (השגת עמידה טובה). מה שיכולה לשים אותו בעמדה שמננה יהיה קשה לו להתחושש. לכן הייתי רוצה להשתמש בעוד מטריקות שישפרו את הערכת העמדות של המחשב.
- **שיפורimax באמצעות טרנספוזיציות:** במהלך החיפוש בעץ המינימקס, ישנן הרבה עמדות שמופיעות לאחר סדר מהלכים שונה. לדוגמה:



אפשר להגיע לאחר המהלך: רגלי ל-e4, רגלי ל-e5, פרש ל-c3 ופרש ל-c6.  
לאוთה עמדה בדיקת אפשרות גם לאחר המהלך: פרש ל-c3, פרש ל-c6, רגלי ל-e4 ורגלי ל-e5.

לאחר שאוთה עמדה הושגה בחיפוש בפעם השנייה, המשך החיפוש באותו ענף יהיה בזבוז של זמן. מאוחר וכבר חישבנו את תוצאה הענף הזה. כדי להסוך בחיפוש המיותר הזה, הייתי רוצה להשתמש בטבלת טרנספוזיציות. שומרת מידע על עמדות שבהם חישנו כבר. דבר שיכל ליעיל את החיפוש ממשמעותית.

## 5.4 מסקנות

בשנה וחצי שבהם עבדנו על הפרויקט, בנוסף לשיעורים, השקעתינו בו את רוב הזמן הפנו אליו. כך שיצא פרויקט די גדול (כנ' אמר... סליה על זה), והרחבתי את הדעת שלי אל מחוץ לתחום הנלמד. עוד דבר שיצא לי לעשות ממנו לא מעט בשנה וחצי האחרון, היה לעזור לחבריי לכיתה בהבנה של התחומיים הנלמדים. ואני חשב שיש לכך חלק גדול בהבנה שלי בחומר. כי כשאתה צריך להסביר משהו למישהו, נדרש הבנה מסוימת בחומר. ואני רוצה לחשב שעמדתי בדרישה זוו רוב הזמן. החלק הכי מספק בפרויקט זהה היא ההרגשה שאני מסוגל להסביר כל פיסת קוד בפרויקט. אני לא אומר שהיא מושלמת, רחוק מכך, אבל אני יודע לבדוק מה הייתה מטרתה.

## 6. תודה

ראשית כל, אני רוצה להודות למורה אילן פרין, שהתמודד עם הנוכחות המציקה שלי בשיעורים, ענה על שאלותי, ודאג שכולנו נשיג את המטלות הנדרשות. למורה גיא יחזקאל, שהתמודד עם הנוכחות המציקה שלי בשיעורים, ותמיד שמה לעזר. אפילו שהפרויקט לא היה בתחום אהוריותו. למורה אסף בנימין, שהתמודד עם הנוכחות המציקה שלי בשיעורים, ולא יותר לנו עד שנבין את החומר. לחבריי לחדר ולכיתה, על התמיכה לאורך השנהתיים.

## 7.ביבליוגרפיה

- [שם למדתי הרבה. בעיקר על מימוש המשחק, והערכת עמדות שעלייה הרחבה ב4.5.2.](#)  
[פונקציית הערכה.](#)

[chessprogramming.org](http://chessprogramming.org)

- אחת הסיבות העיקריות שבחרתי בשחמט היהת הסרטון הזה, והרבה חלקיים מהפרויקט

נבנו בהשראתו

[youtube.com/watch?v=U4ogK0MIzqk](https://youtube.com/watch?v=U4ogK0MIzqk).

- גיזום אלף בטא.

[youtube.com/watch?v=l-hh51ncgDI](https://youtube.com/watch?v=l-hh51ncgDI)

- נוסחא לחישוב כמה טוב שחן כדי לייצר סטטיסטיות.

[cutt.ly/pHiD1Uv](https://cutt.ly/pHiD1Uv)

- ייצוג האריכים ושבועות מסך הנתונים לפי unix timestamp.

[currentmillis.com](https://currentmillis.com)

- בינה מלאכותית בקוד פתוח שמנוה למדדי דרכים לשיפור יעלות הקוד, ובנוסף, לאחר זה קוד פתוח, יכולתי להוריד את כל הבינה המלאכותית, ולהציג אותה בתור אופציה לשחק נגודה.

[stockfishchess.org](https://stockfishchess.org)

- כדי לתקשר עם Stockfish השימושי בAPI זה.

[cutt.ly/aHiDKjb](https://cutt.ly/aHiDKjb)

- כללי המשחק בעברית.

[cutt.ly/5HiDDn9](https://cutt.ly/5HiDDn9)

- השימושי בגרסה מעט שונה מזו כדי להציג תוצאות שאללה מסך הנתונים.

[cutt.ly/oG26V2Q](https://cutt.ly/oG26V2Q)