

סמל ביה"ס: 140129

שם ביה"ס: קריית נוער



עובדת גמר

למילוי חלקי של הדרישות לקבלת תואר

הנדסאי תוכנה

משחק רשות שחמט



מגייש

בצלאל אברהמי 325004752

בהנחיית מר אילן פרץ

שנה"ל תשפ"ב

2022

תוכן עניינים

3	הצעת פרויקט גמר
6	1. הצהרת הסטודנט ואישור הגשה
7	2. מבוא
8	3. מדריך למשתמש
8	3.1 תיאור המשתמש
21	3.2 דרישות טכניות
21	3.3 הרצת המשחק והנחיות שימוש
22	3.2.1 הרצת השרת והשימוש בו
23	3.2.2 הרצת הלקוּח והשימוש בו
26	3.2.3 הפעלת המשחק וממשק משתמש גרפי ((GUI))
46	4. מדריך לתוכנה
46	4.1 הדרישות מערכית התוכנה
47	4.2 ארქיטקטורת המערכת
47	4.2.1 מודל שרת-לקוח Client-Server Model
47	4.2.2 פרוטוקול תקשורת TCP/IP
47	4.2.3 Sockets
48	4.2.4 asad
49	4.2.5 Threads
50	4.3 תהליכיים עיקריים וזרימת המידע
50	4.3.1 תהליך ראשי
55	4.4 מחלקות הפרויקט
55	4.4.1 תרשיimi UML של המחלקות המשותפות ללקוּח ולשרת
60	4.4.2 תרשיim UML של המחלקות בצד השרת
63	4.4.3 תיעוד המחלקות בצד השרת
85	4.4.4 תרשיim UML של המחלקות בצד הלקוּח
93	4.4.5 תיעוד המחלקות בצד הלקוּח
148	4.4.5 תיעוד המחלקות המשותפות ללקוּח ולשרת
154	Enum Constant Details
154	Query
154	Update
236	4.5 אלגוריתמים ופעולות נבחנות
236	4.5.1 אלגוריתם מינימקס עם גיזום
239	4.5.2 פונקציית הערכה
246	4.5.3 אלגוריתם 3 חישוב אלטרנטיבות לשם משתמש תפוא
250	4.5.4 אלגוריתם/פעולה 4 חישוב איזומים של כלים
255	4.5.5 AppSocket 5/פעולה

257	4.6 מבני נתונים עיקריים
258	5. סיכום איש
258	5.1 אתגרים וקשיים איתם התמודדתי
258	5.2 מה הפרויקט תרם לי
259	5.3 הצעות לשיפור
260	5.4 מסקנות
260	6. תודות
260	7.ביבליוגרפיה

הצעת פרויקט גמר

בס"ד



הצעת פרויקט גמר (שאלון 714918) יד הנדסאי תוכנה – תשפ"ב 2022

מספר מסך: 140129

שם מכללה: קריית נוער – כנפי רוח, ירושלים.

שם הסטודנט: בצלאל אברהם

ת"ז הסטודנט: 325004752

שם הפרויקט: משחק שחמט\Chess

תיאור הפרויקט:

מיומש משחק הלוח שחמט שהוא משחק לוח אסטרטגי מופשט וענף ספורט המועד לשני שחקנים.

מהלך המשחק

השחקן ב כלים הלבנים (להלן: "הלבן") הוא הראשון למשחק. כל שחקן מניע בתורו את אחד הכלים שברשותו (כלי אחד בלבד, להוציא במקרה של הצרחה), כאשר לכל כלי אופן תנועה הייחודי לו, כמפורט לעיל, וכך עד להכרעה. השחקן יכול להזיז כל כלי בתורו ולהזיזו לאן שירצה, וב惟ד שלא יעבור על כללי תנועת הכלים המפורטים לעיל. כמו כן, אם המלך של שחקן מסוים נחשף לאיום מצד כלי של היריב (מצב המכונה "שח"), חובה על השחקן לעשות כל שביכולתו כדי לבטל את המזב הזה.

אם לא יצליח לבטל את האיום, יפסיד במשחק, במצב המכונה מט. כל מהלך שאינו מביא לכך באופן מיידי לא יהיה חוקי. בנוסף לכך, כל מהלך שיביא באופן מיידי לכך שהמלך של השחקן המשחק יימצא במצב של שח גם אינו חוקי (המלך הוא הכליל היחיד שאסור להעמידו במצבו תחת איום של כליל אחר).

הכאה: אם ניצב אחד מכליו של היריב בדרכו של הכליל הנע, באפשרותו להכות ("לא יכול") אותו, בתנאי שמדובר במצב חוקי. בשחמט (שלא כמו בדמקה, למשל) ההכהה היא זכות ולא חובה. בהכהת כליל אחר, הכוונה לסלוקו מהלווה והצבת הכליל המכה במקומו. את המלך אין מכינים. מקובל להכריז, במצב בו הוא חשוף לאיום ולא ניתן להגן עליו, על שחמט ("שח", בתוספת "מט"; יש שמקצתרים לו-"מט"). כל הכלים מכנים בדרך הילוכם (למשל, הרץ ייכה רק על אלכסון), מלבד הרגלי שזו קדימה אך מכנה באלכסון. מקרה מיוחד של הכאה הוא "הכהה דרך הילוכו".

שלבי המשחק

השלב הראשון במשחק הוא הפתיחה. בשלב זה מפתח השחקן את כליו, החסומים מאחוריו שורת הרגלים, על מנת לאפשר להם לשולוט על מרוץ הלוח ולאיים על ההגנה של היריב. שלב הפתיחה גם כולל את פינוי אחד האגפים על מנת לאפשר הצרחה שטטרת לבצר את המלך בפינה ולהרחיקו מהמרכז החשוף. פתיחה אופיינית יכולה בדרך כלל לדילוג כפול עם הרגלי של המלך, הוצאת אחד או שני הפרשים והוצאת אחד או שני הרכזים, ואו הצרחה. ישנן פתיחות הנקראות גמبيיט, בהם אחד הצדדים מקריב חומר - לרוב רגלי - על מנת להשתלט על המרכז. מספר הפתיחות הוא רב ומספר ה兜ריאנטים שלו נרחב עוד יותר. שחקנים מקצועניים משננים בעלפה פתיחות רבות על מנת שלא לבצע טעויות בתחילת המשחק, דבר שייתן יתרון מוקדם ליריב.

השלב השני הוא מציצה, התראות אחורי שרוב הכלים כבר פותחו, והקרבת מתנהל מרבית חרייפותו במרכזה הלוח, בדרך כלל תוך כדי חילופי כלים וניסיון לשבור את מערכ הרגלים של הצד השני. בשלב זה אפשר למשר מספר רב של טקטיקות לצבירות יתרון חומיי ועמדתי (ראו בהמשך).

השלב השלישי הוא סיום, ושלב זה קורה כאשר נעשו מספר רב של חילופי כלים אבלתי שהושג יתרון מכריע לצד כלשהו. שלב זה כולל בדרך כלל מספר קטן של כלים לכל צד (מלך, קצין ומספר رجالים) ובו מנסים השחקנים להכריע אחד את השני בדרך כלל על ידי הכתרת אחד הרגלים לכלי בלבד (כגון מלכה). בשלב זה, האיום במט על המלך קטן ולכון המלך הופך לכלי פעיל, המגן על הרגלים ותומך בהתקומות אל עבר השורה השמינית. מכיוון שהטיום דורש משחק מדויק, שחקנים רבים לומדים בקפידה עמדות סיום נפוצות.

הגדרת הבעה האלגוריתמית:

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

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

רקע תיאורי בתחום הפרויקט:

לימודי י"ג י"ד הנדסי תוכנה שבhem למדנו בין היתר: עבודה עם מסד נתונים בשפת SQL, תכונות מונחה עצמים, מבני נתונים (רשימות מקשורות, מערכיות...), מינימקס (כולל גיזום אלף בטה), Networking (סוקטים, TCP\IP, מודול שרת לקוב, מודול השכבות...), שימוש בת'דים, מודל MVC, עיצוב ממשק גרפי למשתמש באמצעות swing.java

תהליכיים עיקריים בפרויקט:

- הרשמה של משתמש חדש.
- התהברות של משתמש רשום או אורח.
- משחק נגד שחקן מוחשב בעזרת שימוש באלגוריתם מינימקס.
- שרת עובד במקביל לצורכי ריבוי שחקנים ומשחקים.
- שימוש במסד נתונים (توزאות משחקים שהסתינו ושמירת משחקים שהופסקו)
- שימירה במסד נתונים עדכון מasad הנתונים
- הפקת סטטיסטיות ושאלות עדכון מasad הנתונים

תיאור טכנולוגיה וארQUITטורה:

שימוש במודול שרת לכוח ניהול משחק בין לוחות, שימוש בסוקטים להעברת נתונים ברשות, שימוש בת'דים לניהול של כמה שחקנים וכמה משחקים במקביל.

שפת התכונות צד לקוח ושרת:

נכתב בשפת **java (16.0.2) Win10**

פרוטוקולי תקשורת:

התקשרות בין השרת ללוחות מתבצע באמצעות העברת הודעות על סוקטים TCP/IP ההודעות מכילות תוכנה בשם "MessageType" שלפיה מקבל ידע איך לנוט את ההודעה, ועוד תוכנות שנלוות לכל סוג הודעה.

תיאור מסד נתונים:

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

לוחות זמינים:

מרץ	אפריל	מאי	יוני	יולי	אוגוסט	ספטמבר	אוקטובר	נובמבר	דצמבר	ינואר	פברואר	מרץ
												ניתוחה המערכת
												תכנון מבני הנתונים ו알גוריתמים
												כתיבת הקוד
												עיצוב משתמש
												ניסוי 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 זו שכל הנאמר בה אמת ורק אמת.

תאריך:

חתימתה הסטודנט:

אישור המנהה האישית

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

שם המנהה: _____ תאריך: _____ חתימה: _____

אישור ראש המגמה

הריני מאשר הגשת הסטודנט להגנה על פרויקט הגמר.

שם ראש המגמה: _____ תאריך: _____ חתימה: _____

2. מבוא

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

המשחק תוכנן ופותח על פי הדרישות מערכת 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 שרת/לקוח** (Client/Server) **Threads**.
הכולל עבודה עם **שקעים** (Sockets) בפרוטוקול TCP/IP, ושימוש בתהיליכונים בסעיף 5.2 יש הסבר מעמיק על ארכיטקטורת המערכת ומרכיביה.

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

בסעיף 5.5.1, יש הסבר מעמיק על אלגוריתם מינימקס המממש את פועלות השחקן הממוחשב.

- לאפשר התחברות (Login) של שחקנים א/orחים ורשומים, שמירת תוכאות משחקים שהסתינו, שמירת משחקים שנפסקו וטעינתם בהמשך, לקבלת סטטיסטיקות ולאפשר פעולות עדכון.
- לכן מערכת התוכנה עושה שימוש **בבסיס נתונים מסוג Access** ובו שמורות טבלאות המכילות מידע על השחקנים הרשומים, על תוכאות המשחקים, על מצב המשחקים שלא הסתיימו ועוד ...

בסעיף 5.2.4 יש הסבר עמוק יותר על מסד הנתונים ותיאור כל הטבלאות שמורות בו.

- לכתוב את קוד התוכנה בשפת Java תוך שימוש בעקרונות של תכנות מונחה עצמים מתקדם.
 - לכן נעשה שימוש במחלקות, **ירשה ופולימורפיים**, **משקדים (Interfaces)**, **אוסףים גנריים**, וכמוון **שימוש בתהליכיונים (Threads)** לצורך ניהול **משחקים במקביל**.
- בסעיף 5.4 יש הסבר מكيف על כל המחלקות בפרויקט (ממשקי API, תרשימי UML, תיעוד ...).
- בסעיף 5.2.5 יש הסבר עמוק על השימוש בתהליכיונים ולמה היה צריך אותם.

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

3. מדריך למשתמש

בחילק זה יוסבר על המשחק, הלוח והכלים, חוקי המשחק והמלחלים. בנוסף נתאר את הקבצים המוגושים ביחד עם ספר פרויקט זה (שהם חלק בלתי נפרד מהפרויקט כולם), ונסביר איך להריץ את התוכנה ואיך להישתמש בה **מנקודת מבטו של המשתמש**.

נתחיל בפירוט הדרישות הטכניות להריצה, נסביר בהסביר על קבצי ההרצה ואופן הרצטם, ולבסוף נסביר איך לשחק במשחק הלכה למעשה – נעשה זאת על ידי תיאור ממשק המשתמש הגרפי (GUI) בלבד צלומי מסך להמחשה.

3.1 תיאור המשחק

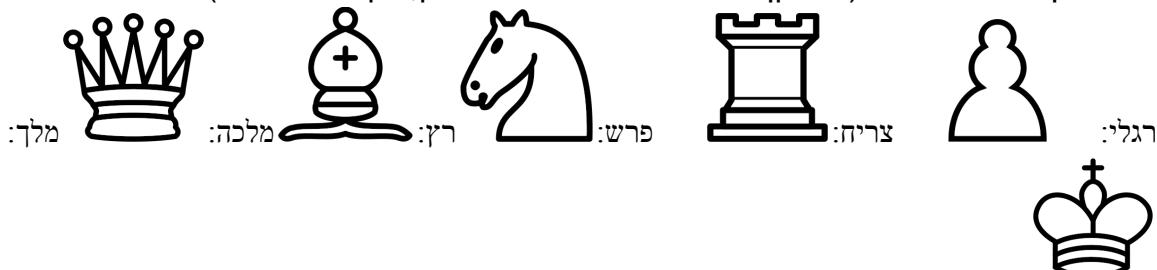
• תיאור המשחק

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

ניצחון במשחק מושג, כאשר אחד המלכים מאויים בשח על ידי אחד מכל היריב, אין יכול להכוט את הכלי המאיים, לחסום את האיים על ידי כלי אחר או להימלט מהאיים למשבצת בה לא יהיה מאויים.

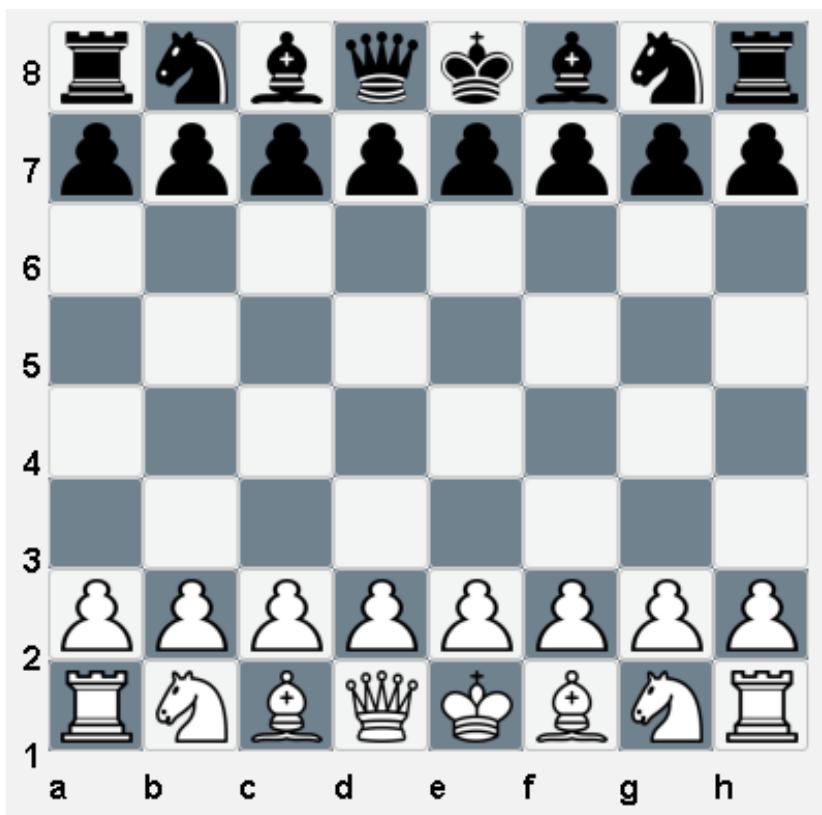
• הולח והכלים

לוח המשחק בגודל 8X8. הכלים של כל שחקן הם: מלך(8X), צריה(2X), פרש(2X), רץ(2X), מלכה(X) ומלר(1X). צבעם של הכלים נקבעים לפי צבע השחקן שלהם. לבן לשחקן הלבן, ושחור לשחקן השחור. להלן הכלים הלבנים (לשחקן השחור יש אותם כלים בדיאוק, רק בצבע שחור):

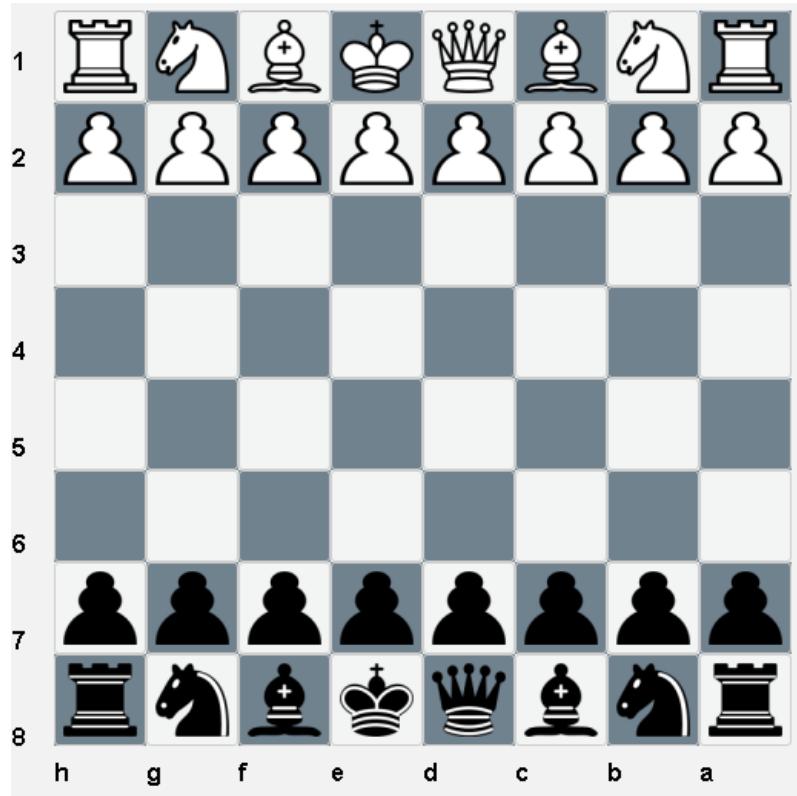


• מצב התחלתי

בהנחה שמתחילה מעמדת ההתחלה הסטנדרטית השחקן הלבן משחק ראשון, והלוח 8X8 מסודר כך:



מנקודת המבט של השחקן השחור ייראה ייראה בדיקו אותו הדבר, רק שהכל יהיה הפוך. לדוגמה, עמדה זו תראה לשחקן השחור כך:

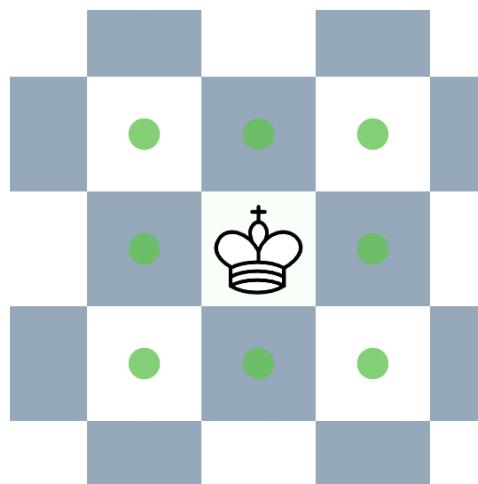


• חוקי המשחק והמהלכים

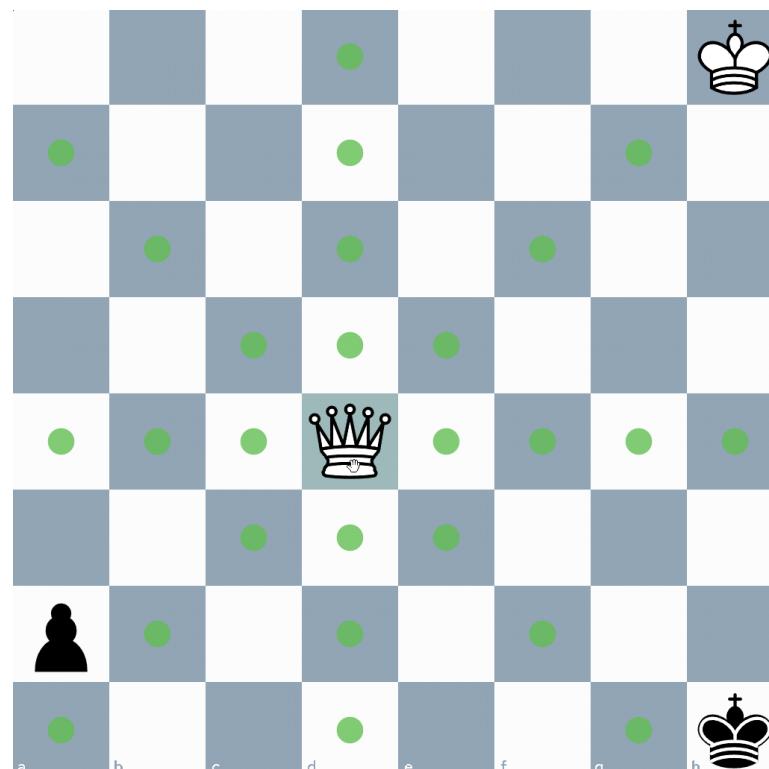
תנועת הכלים

בשחמט לכל כלי דרך תנועה שונה. לעיתים אי אפשר לנוע לתוך משבצת שנמצאה בה כלי מהצעע של המשחק, או לעبور מעל כלי צזה (מלבד הפרש). בנוספ', ישנה אפשרות של "הכהה" (מכונה גם: "אכילה" או "לקיחה"): להסיר כלי של היריב מהלוות. הכהה מתבצעת לרוחב כדרך התנועה (מלבד הרגלי) הכלי, נע תמיד למשבצת בה היה הכלי היריב (מלבד המצב של "הכהה דרך היילוכו"). במשמעותו לא יוזהו יותר מכלי אחד (מלבד "הצרכה"). הכלים בעלי תנועה "ארוכה" (מלכה, צריך ורץ) יכולים לנוע כרצונם כל עוד אין בדרך כלי מכבעם (שاذ עליהם לעצור לפניו) או כלי מהצעע השני (שاذ עליהם לעצור לפניו, או להכות אותו).

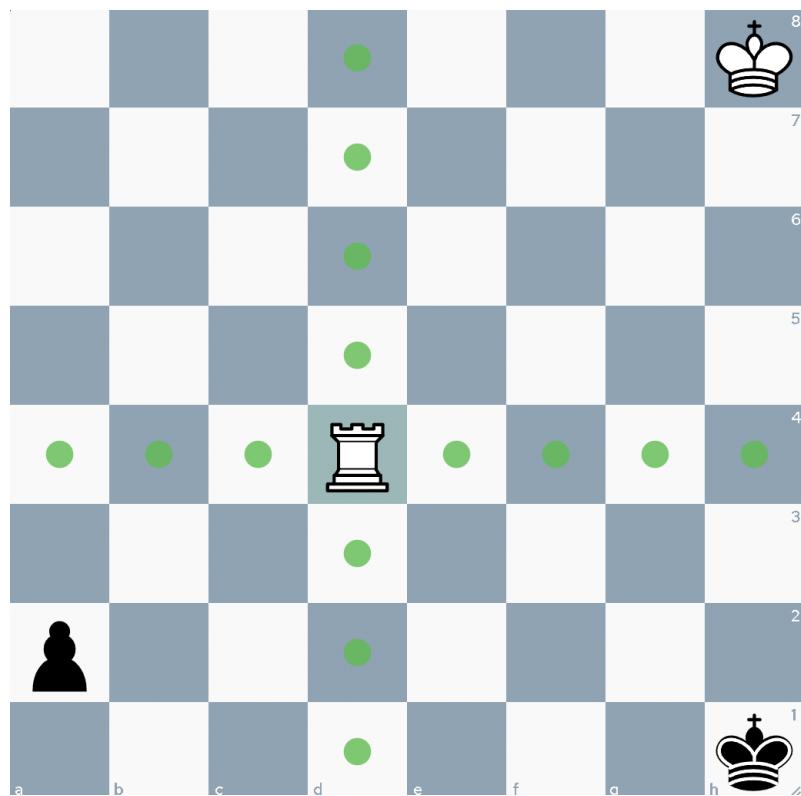
מלר: המלך יכול לנוע משבצת אחת לכל כיוון



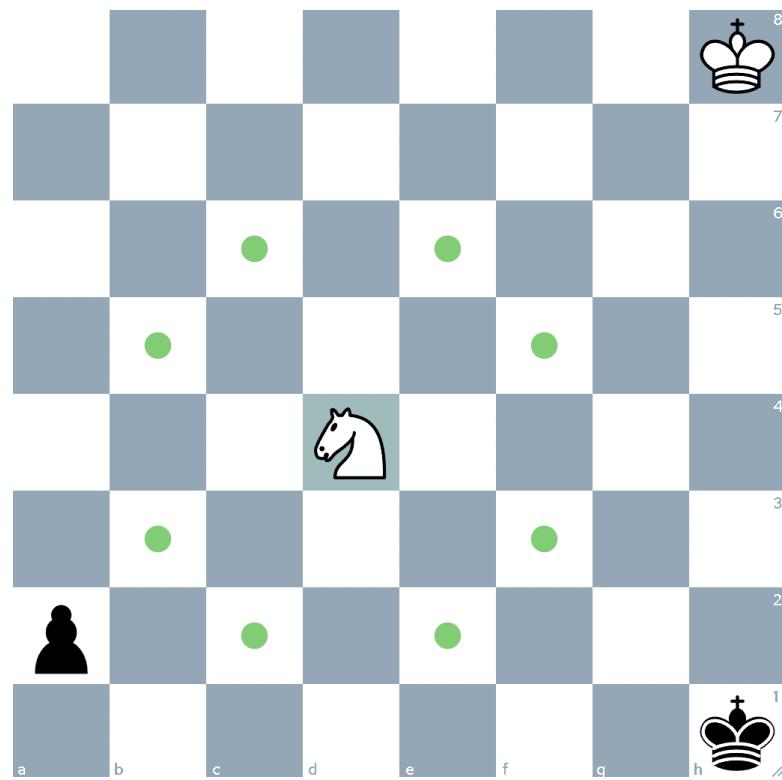
מלך: המלכה יכולה לנוע מספר בלתי מוגבל של משבצות לכל כיוון.



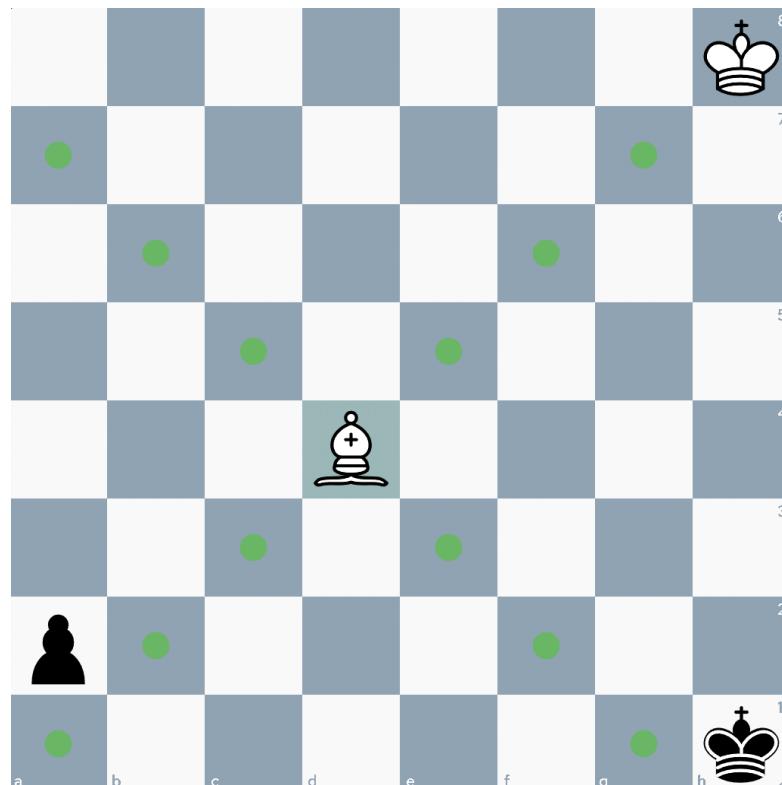
צריח: הצריח יכול לנוע מספר בלתי מוגבל של משבצות בתורים ובסורות.



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.

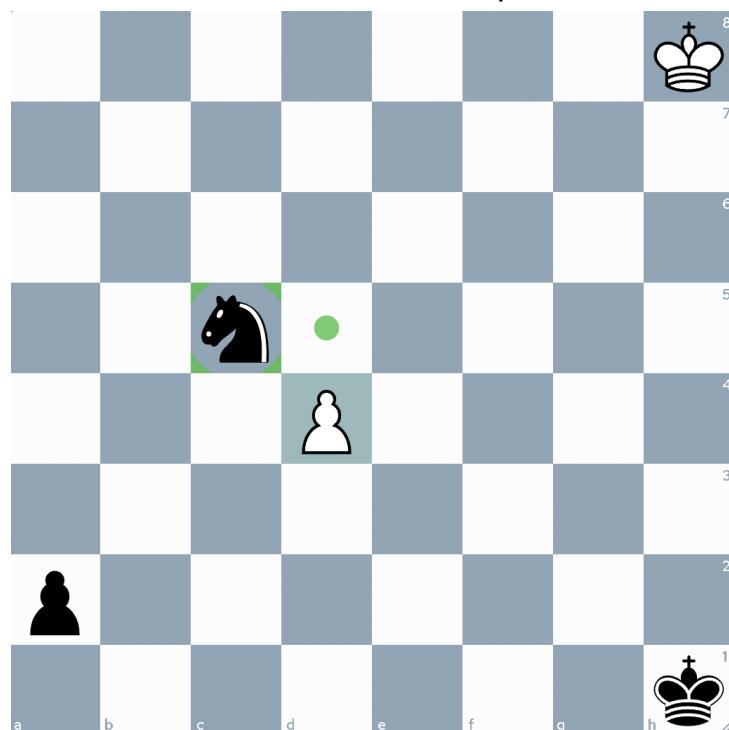


רץ: הרץ יכול לנوع מסוים בלתי מוגבל של משבצות באלאנסונים.



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 היילוקו": 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.



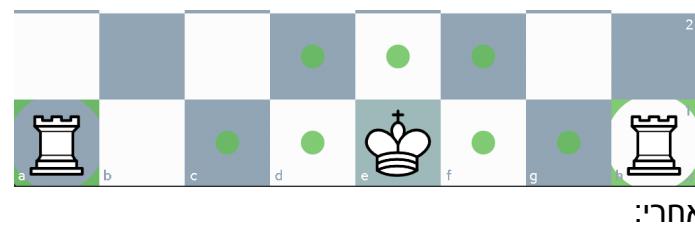
תנועות מיוחדות

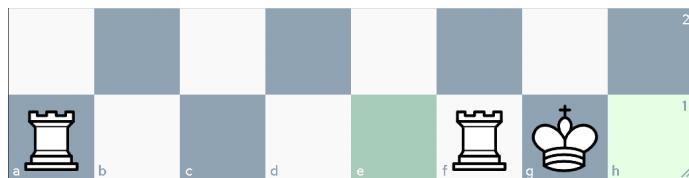
ההצרכה

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 הצירה הקטנה. הצרכה שבה משתמש הצריח השני מגף המלכה נקראת הצרכה גדולה.

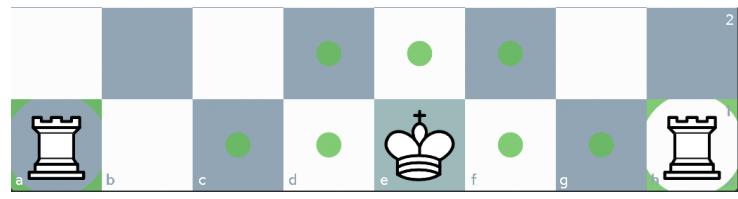
הצרכה קטנה

לפני:

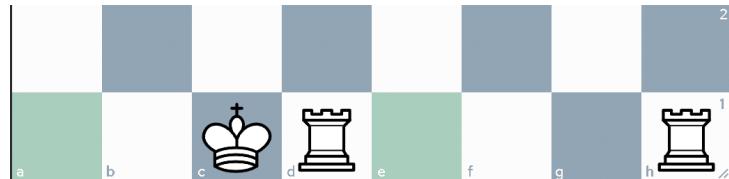




הצרכה ארוכה:
לפניכם:



אחרי:



המצבים בהם זכות הצרכה מבוטלת:

- אם כלי האויב שולטים באחת העורוגות אותן המלך צריך לחצות כדי להגיע לעמדת הצרכה.
- כשר 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:

- שחmate: 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 השחקן שנכנע.
- נגמר הזמן: מצב שבו נגמר הזמן לשחקן שתורו לשחק, וליריבו יש קומבינציית כלים שיכולים להנחתת מט.

פט (תיקו)

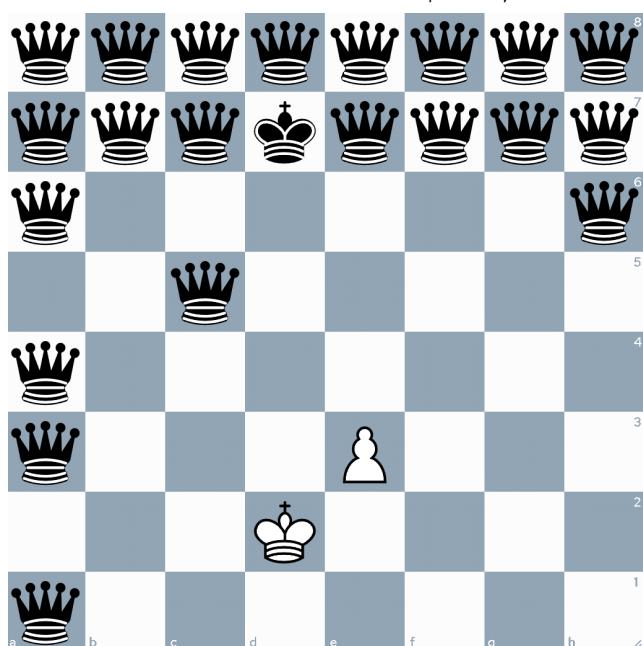
פט יקרה כאשר אחד מהדברים הבאים קוראים:

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

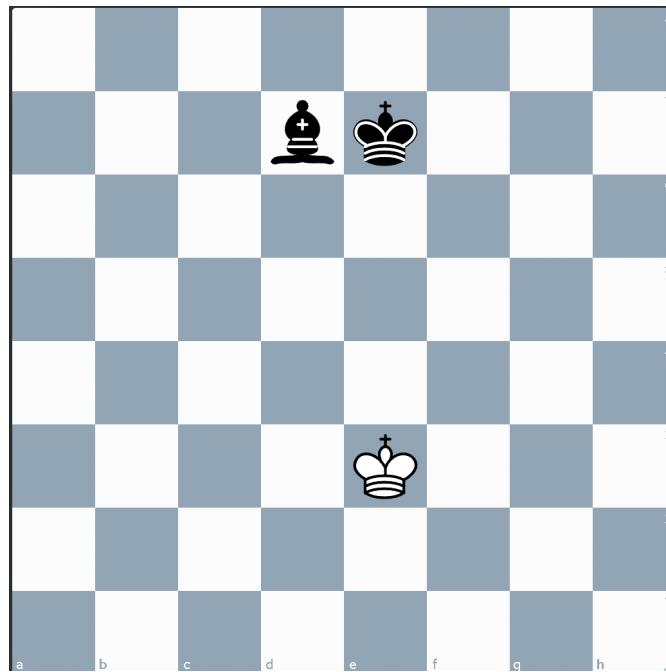
- כאשר אין אפשרות לבצע מהלך חוקי אבל המלך של הצד שטורו לשחק לא מואים בשח.
- כאשר אין לשני הצדדים שום דרך חוקית לחתת מט גם אם היריב טועה(למשל מלך מול מלך או מלך ופרש מול מלך).
- חזרה משולשת: כאשר העמدة חוזרת על עצמה שלוש פעמים או עומדת לחזור על עצמה 3 פעמים(המנוח עמדה אומר הצד שטורו לשחק זהה, וشعמדות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

כמה דוגמאות לסופי משחק:

תورو של הלבן לשחק

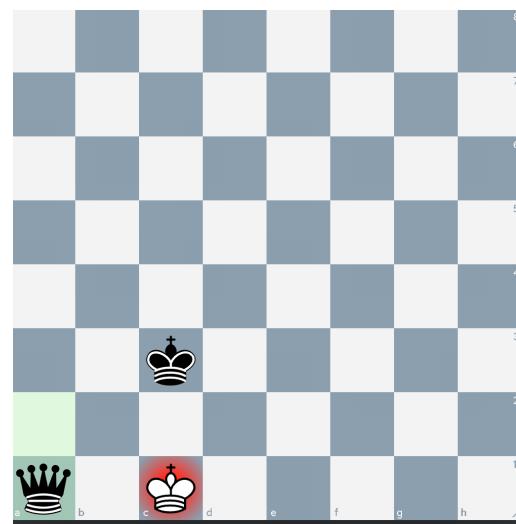


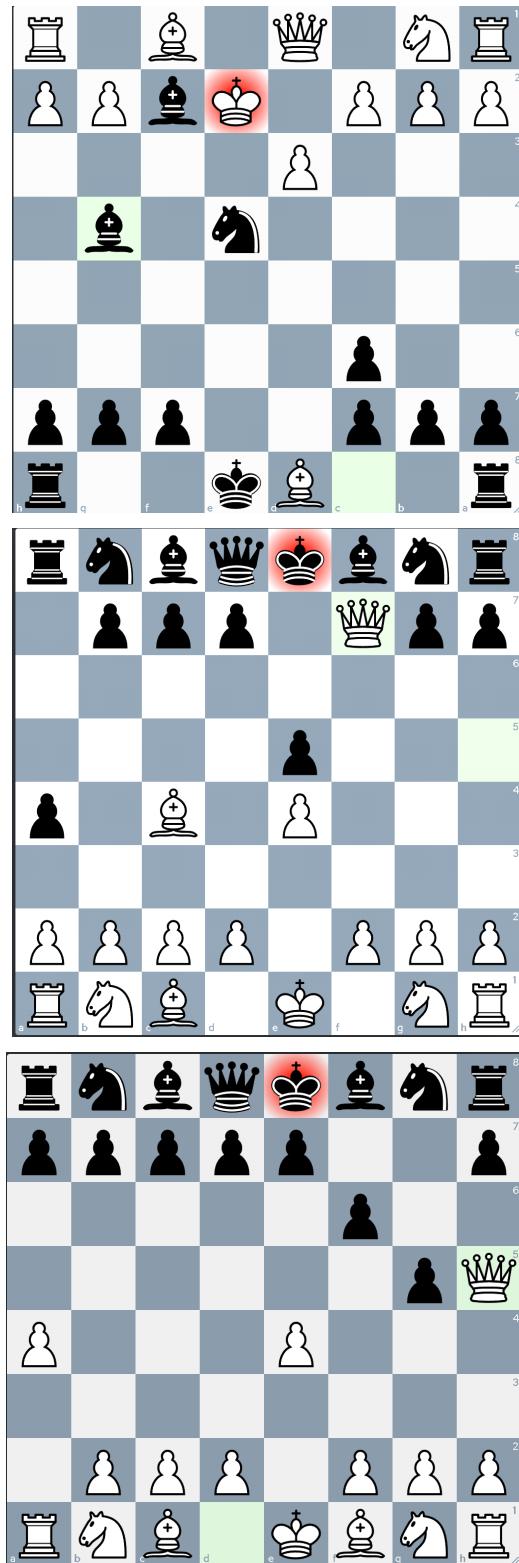
פט. לאחר מכן לשחקן הלבן מהלכים חוקיים.



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

כמה דוגמאות למט-ים:





3.2 דרישות טכניות

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

- ✓ מחשב עם מערכת הפעלה 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 הפרויקט המצורף יש 3 תת-תיקיות:

- ▼ **תיקיית הגשה**
- > **תת תיקייה לספר הפרויקט**
- > **תת תיקייה לקבץ ההרצה של הפרויקט**
- > **תת תיקייה לקוד המקור של הפרויקט**

יש להכנס לחת-התיקייה **Run Files** שבתוכה הקבצים הבאים:

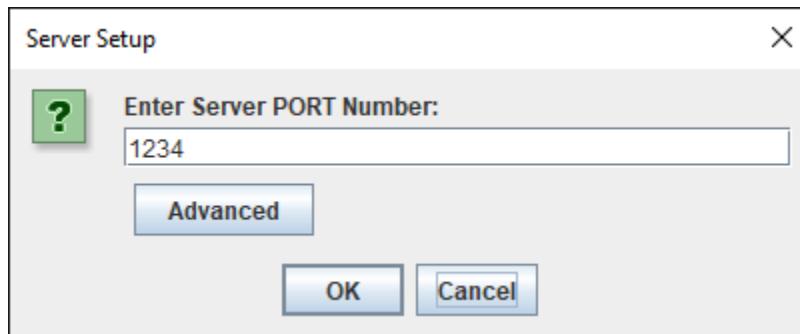
- ▼ **Run Files**
- > **תת תיקייה לספריות עדר וככיסים ללקוח**
- > **תת תיקייה לספריות עדר וככיסים לשרת**
- קובץ הרצה של הלוקו **ChessClient.jar**
- קובץ הרצה של השרת **ChessServer.jar**
- בסיס הנתונים **db.accdb**

הרצת המשחק מצריכה הריצה של שני מודולים נפרדים – מתחילה בהפעלת **השרת ע"י** הרצת הקובץ **ChessServer.jar**, שמיד ממתין לחיבור לקוחות (שחקני-רשות אונשיים). הריצה זו **חד-פעמיות**.
כעת מפעילים את הלוקו ע"י הרצת הקובץ **ChessServer.jar** שמייצג שחון-רשות יחיד. ניתן להרין מספר קווחות כדי ליצור מספר שחוקנים. עבור כל שני שחוקנים, הרשת "משדר" ויוצר עborם משחק.

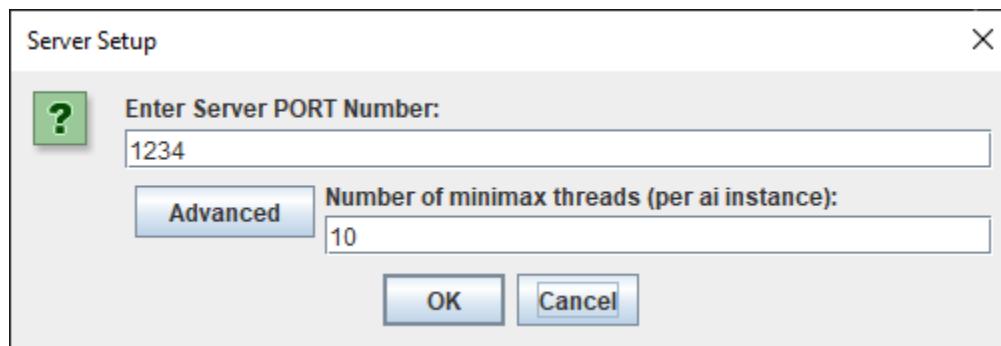
בנסיבות הבאים נסביר בצורה מפורטת, באמצעות צלומי מסך, איך להריץ ולהשתמש בשרת ובלוקות, איך להתחילה משחק בין שני שחקנים וכמוהן איך לשבח.

3.2.1 הרצת השרת והשימוש בו

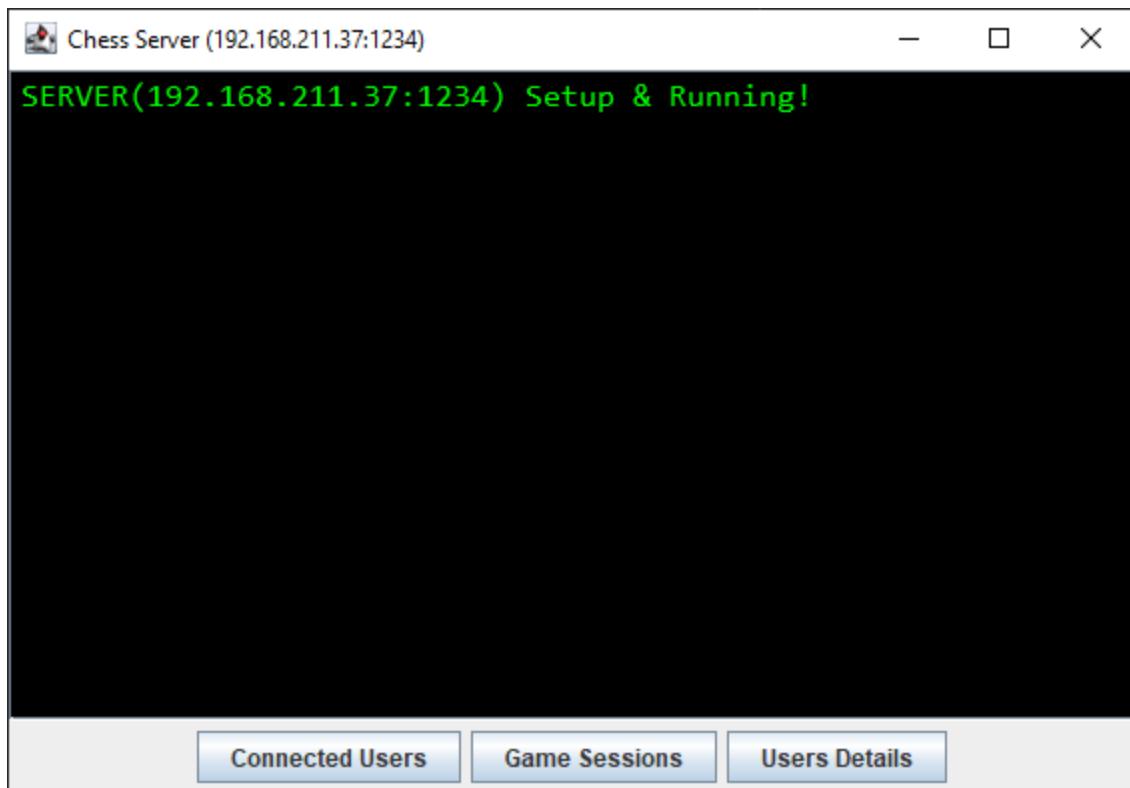
בהרצת השרת קופץ חלון את חולן לשרת, שבו תבקש לבחור פורט עלייו ירוז' השרת.



הפורט צריך להיות לא בשימוש. אם יוכנס פורט שנמצא בשימוש, יקפוץ חלון שגיאה. בלחיצה על Advanced תוכל לבחור את מספר התהיליכים שהמינימום ייצור עכור כל מופע של הבינה המלאכותית.



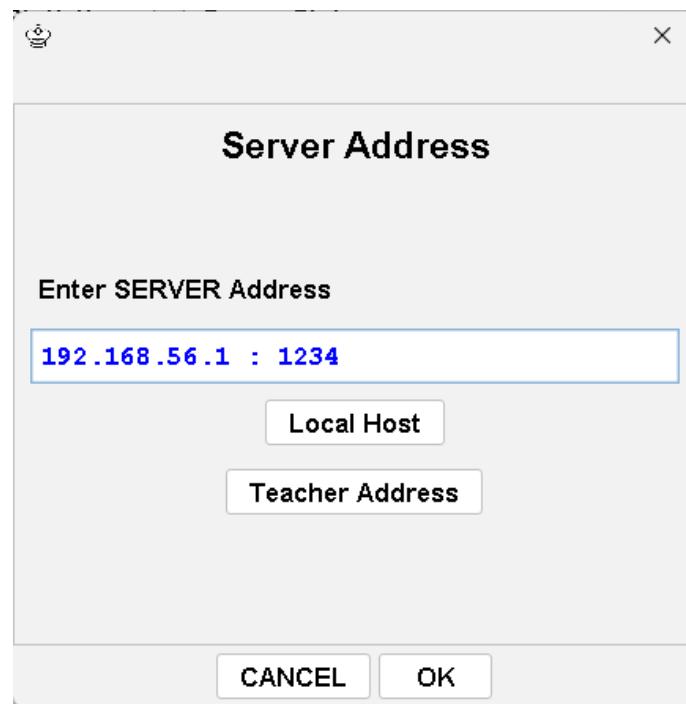
אחרי שהשרה אוחת בצלחה, השרת מוכן לקבלת ליקויות בפורט שצווין.



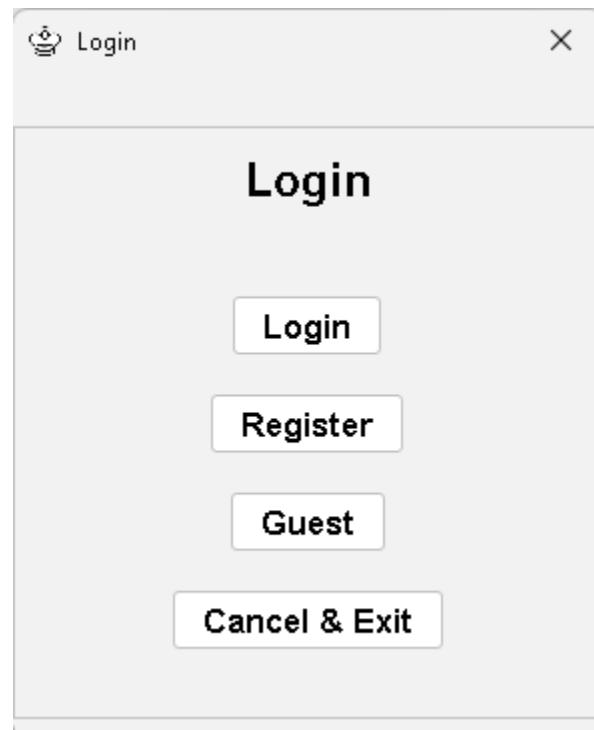
בלחיצה על הכפתור "Connected Users" תוצג רשימה של כל המשתמשים המוחברים.
בלחיצה על הכפתור "Game Sessions" תוצג רשימה של כל המשחקים שרצים בשרת כרגע.
בלחיצה על הכפתור "Users Details" תוצג רשימה של כל שמות המשתמשים והסיסמאות של המשתמשים הרשומים.

3.2.2 הרצת הלוקה והשימוש בו

בעת הרצת הלוקה יוצג חלון להנחת כתובת השרת. הכתובת תהיה בפורמט: IP:PORT ישנים גם ערכיים דיפולטיים: localhost, במקרה שבו השרת מורץ על המחשב הנוכחי על פורט 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 character counter 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 _.-

Buttons at the bottom include "BACK" and "OK".

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:

The screenshot shows a mobile application window titled "Login". At the top left is a small user icon, and at the top right is a close button (X). The main title "Login" is centered above two input fields. The first field is labeled "Username" and contains a placeholder text box. Below it is a red instruction: "5-10 Characters A-z 0-9 _.-". The second field is labeled "Password" and also contains a placeholder text box. To the right of the password field is a small eye icon for password visibility. Below the password field is another red instruction: "5-10 Characters A-z 0-9 _.-". At the bottom are two buttons: "BACK" on the left and "OK" on the right.

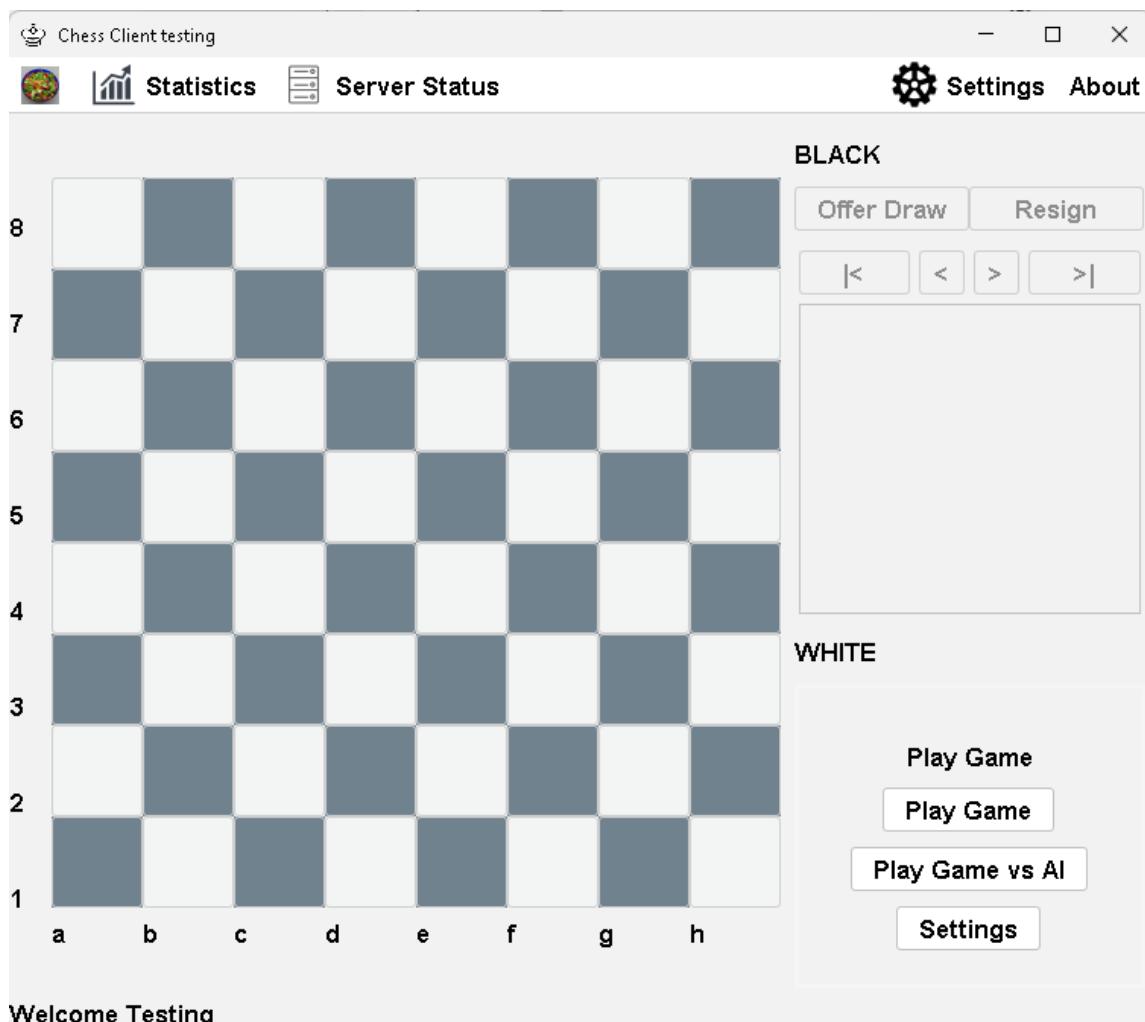
ב"Username" צריך להכניס את שם המשתמש.

ב"Password" צריך להכניס את הסיסמה.

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

3.2.3 הפעלת המשחק וממשק משתמש גרפי (GUI)

לאחר כניסה של משתמש יוצג החלון הבא:



רכיבים גרפיים

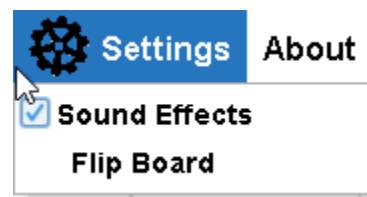
בתחתית החלון בצד שמאל נמצאת תווית הסטטוס. בה מוצג מידע על המצב הנוכחי. בצד ימין של החלון נמצאים התיעוד של המשחק, פעולות בקשר למשחק, והטיימרים של המשחק. ארכיב עליהם בהמשך.

התפריט העליון

בלחיצה על About יוצג:



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



ניתן לשולוט במצב האפקטים הקוליים. כשהחלונית Sound Effects מסומנת בV האפקטים דלוקים וההפק. בלחיצה על הלוח יתאפשר לנקודת 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:

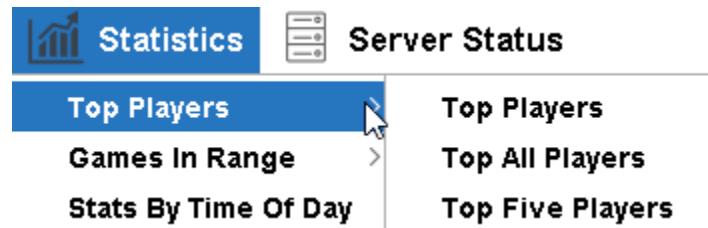


עבור שחון שמחובר בתור אורה, Stats By Time Of Day ו-Games In Range לא יהיה לחיצ. לאחר ואלו סטטיסטיקות שנבנו באמצעות משחקים שמורים בסיס הנתונים. ולאורה אין משחקים שמורים.

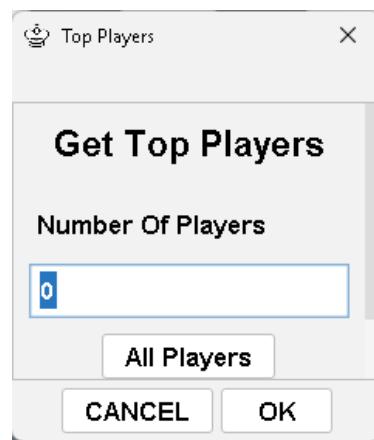
Top Players

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 מטבח בזורה זה: $\frac{ties*0.5+wins}{games}$

לחיצה על Top Players תציג:



לחיצה על Top Players תציג:



הערך הדיפולטי, 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

A screenshot of a dialog box titled 'Top All Players | SUCCESS'. It displays a table of players and their statistics:

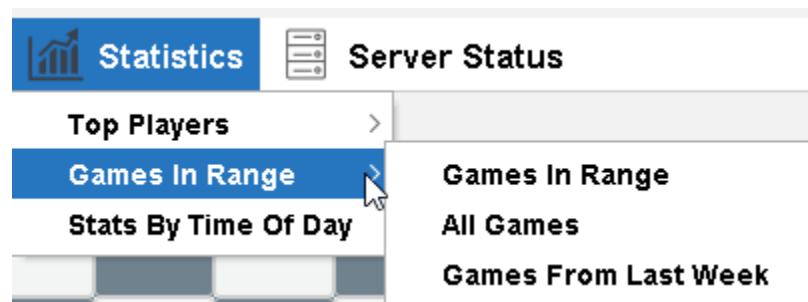
Username	Win-loss-tie Ratio	Num Of Games Played
testing	0.394	19
bezalel6	0.307	13
bezalel0	0.000	0
bezalel1	0.000	0
bezalel_6_	0.000	0

Below the table, there is a section labeled 'Total Games' with the value '31'.

At the bottom is an 'OK' button.

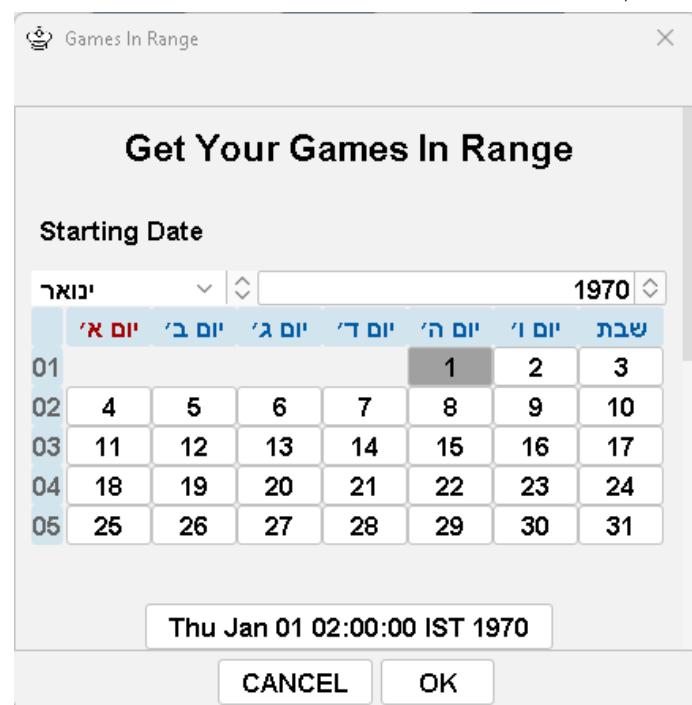
בטבלה העליונה ניתן לראות את רשימת המשתמשים והסתטיטיקות שלהם. בטבלה התחתונה ניתן לראות את הסכום הכלול של השחקנים השמורים בסיס הנתונים.

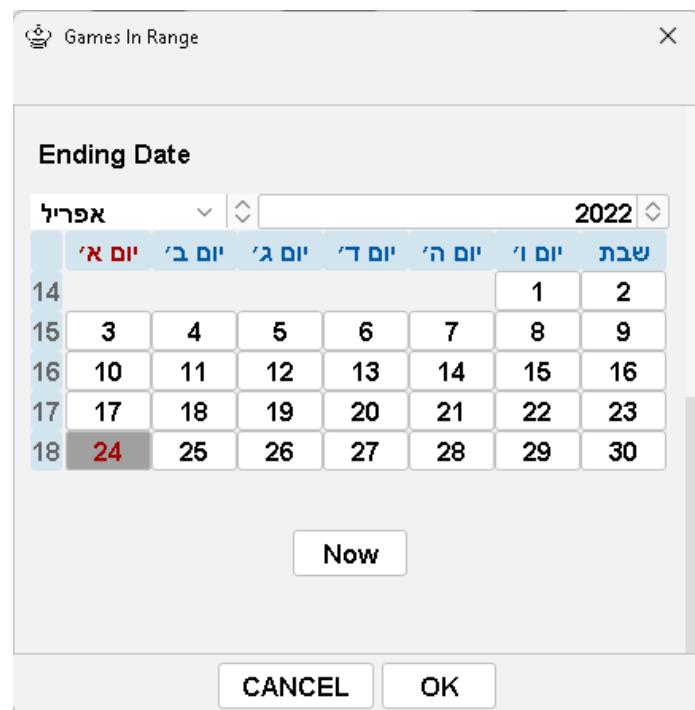
[Games In Range](#)



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 | SUCCESS

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
MyAi	MyAi	2022-04-17 09:18:14.000000
MyAi	testing	2022-04-17 09:14:05.000000
GUEST#0	testing	2022-04-17 07:05:35.000000
MyAi	MyAi	2022-04-17 04:19:59.000000
MvAi	MvAi	2022-04-17 04:19:44.000000

OK

All Games | SUCCESS

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
MyAi	MyAi	2022-04-17 09:18:14.000000
MyAi	testing	2022-04-17 09:14:05.000000
GUEST#0	testing	2022-04-17 07:05:35.000000
MyAi	MyAi	2022-04-17 04:19:59.000000
MyAi	MyAi	2022-04-17 04:19:44.000000
MyAi	MyAi	2022-04-17 00:15:15.000000

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 הסטטיסטיקות 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

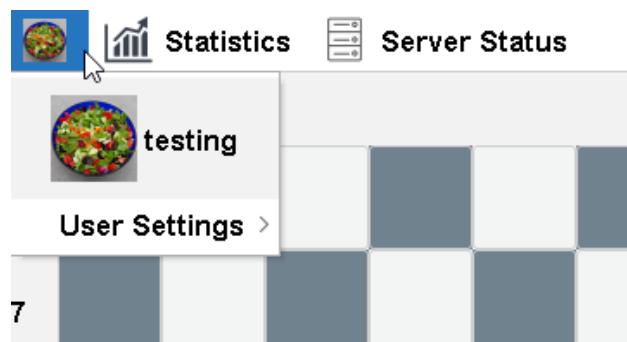
19:00 - 20:00

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

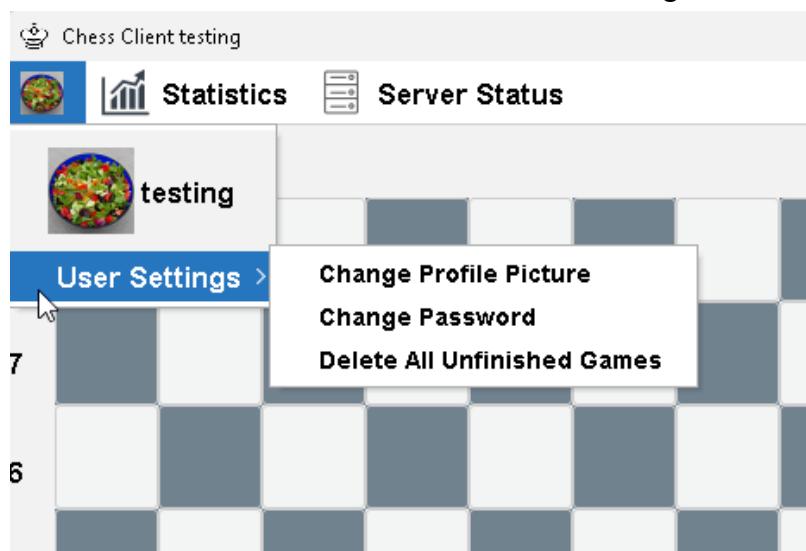
17:00 - 18:00

OK

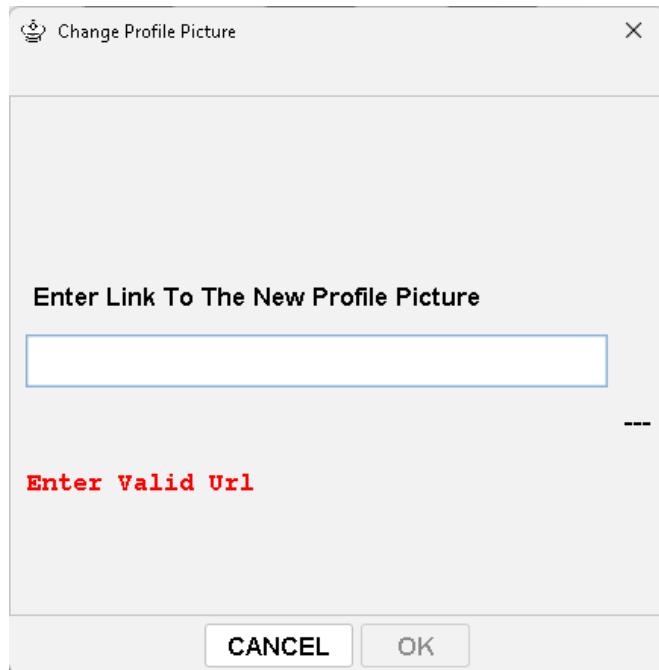
הגדרות משתמש



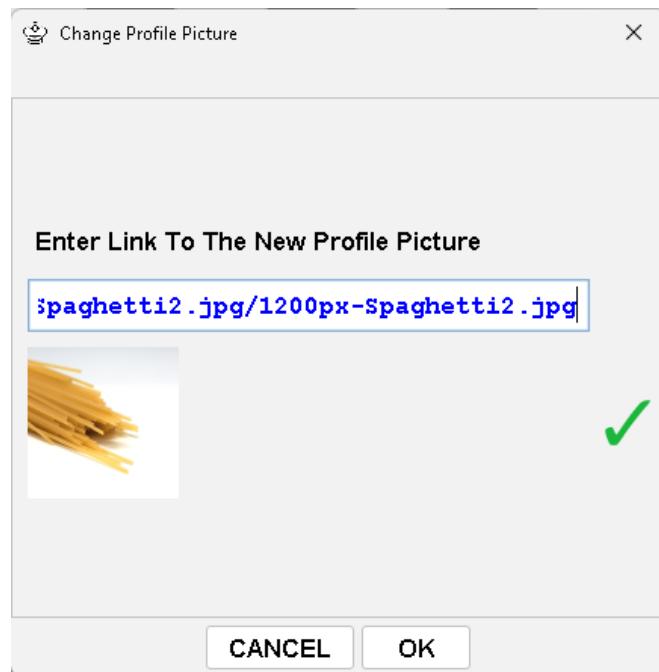
בעה לחיצה על תמונה ה פרופיל בצד שמאל למעלה, יוצג החלון עם שם המשתמש ותמונה הפרופיל של המשתמש (או תמונה אונומית במקורה של אורח או משתמש שלא הגדר את תמונה הפרופיל שלו).
הכפתור User Settings לא יהיה זמין עבור אורח.
בלחיצה על User Settings של משתמש רשום:



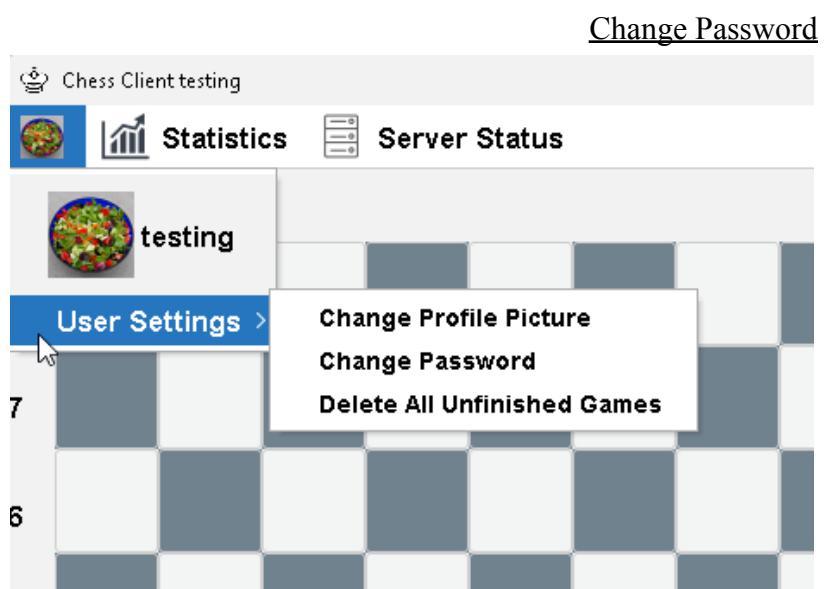
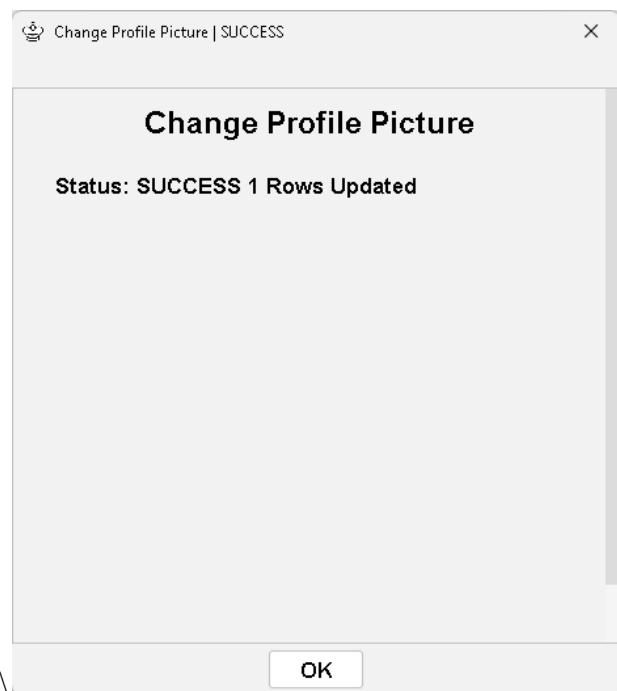
בלחיצה על Change Profile Picture יוצג החלון הבא:



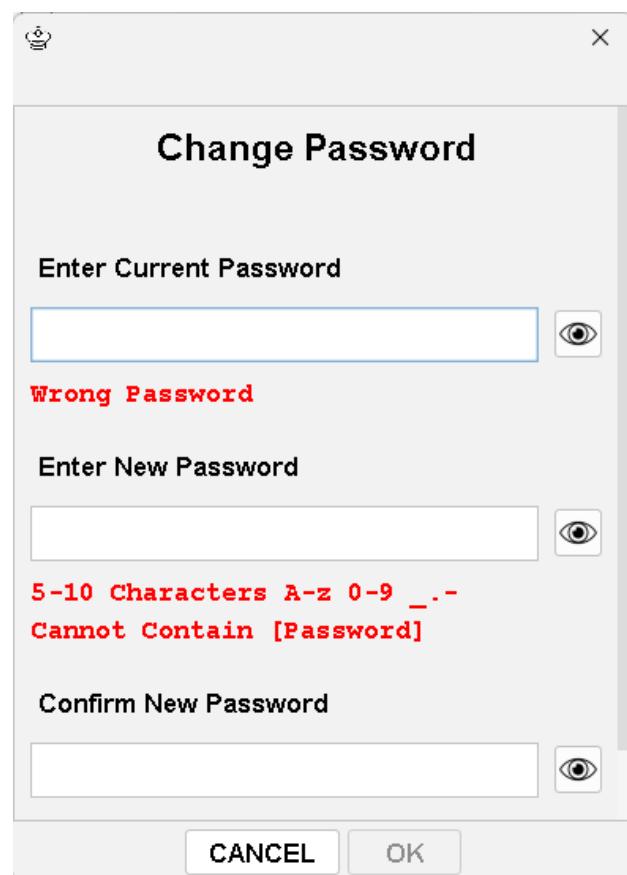
לאחר הכנסה של כתובת חוקית לתמונה:



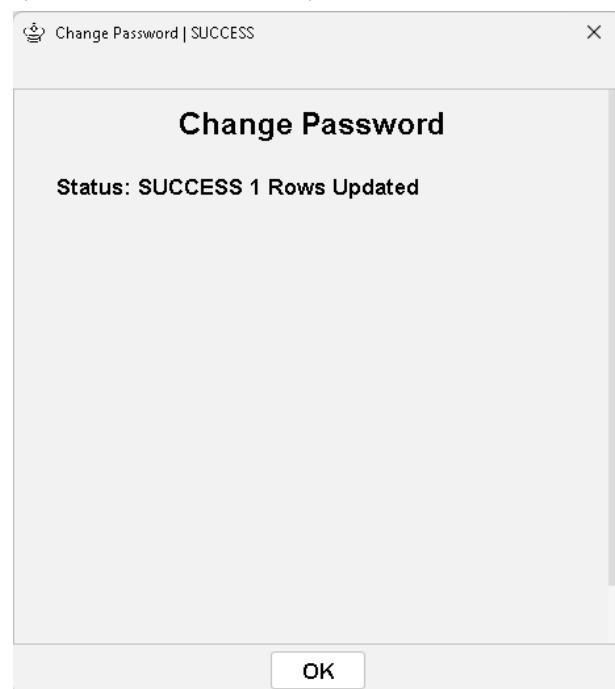
(התמונה תהיה התמונה שנמצאת בכתובת שהוכנסה)
לאחר לחיצה על OK הכתובת החדשה תשלוח לשרת. לאחר שהשרת יעדכן את התמונה מסדר הנתונים
בהצלחה, תוצג הודעה:

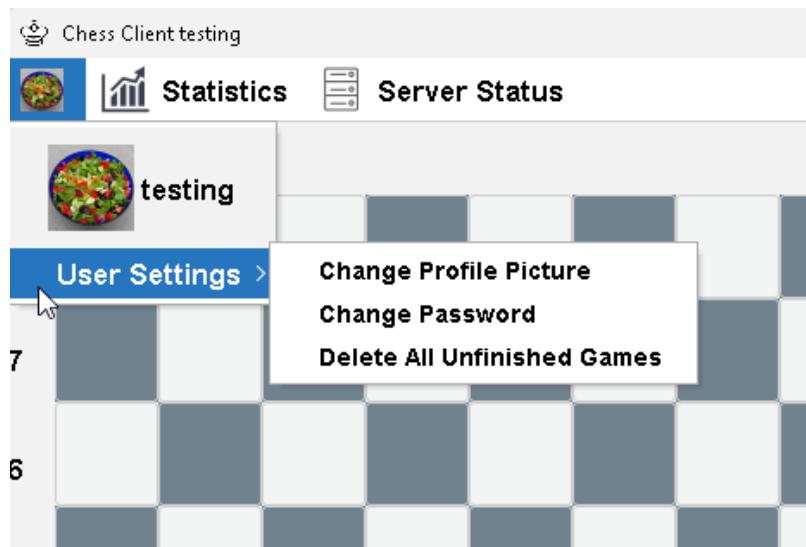


:Change Password לאחר לחיצה על



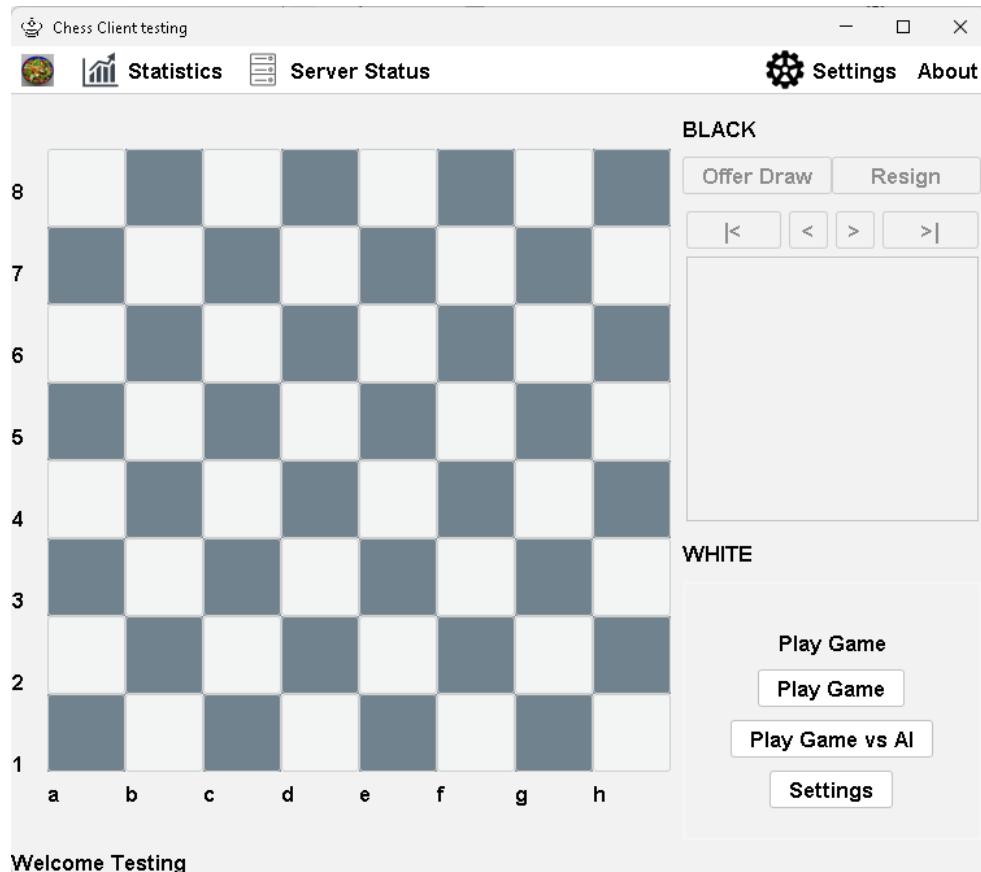
לאחר מילוי הסיסמה הנוכחית, הסיסמא החדש והאימות, ולהזיכת על OK:



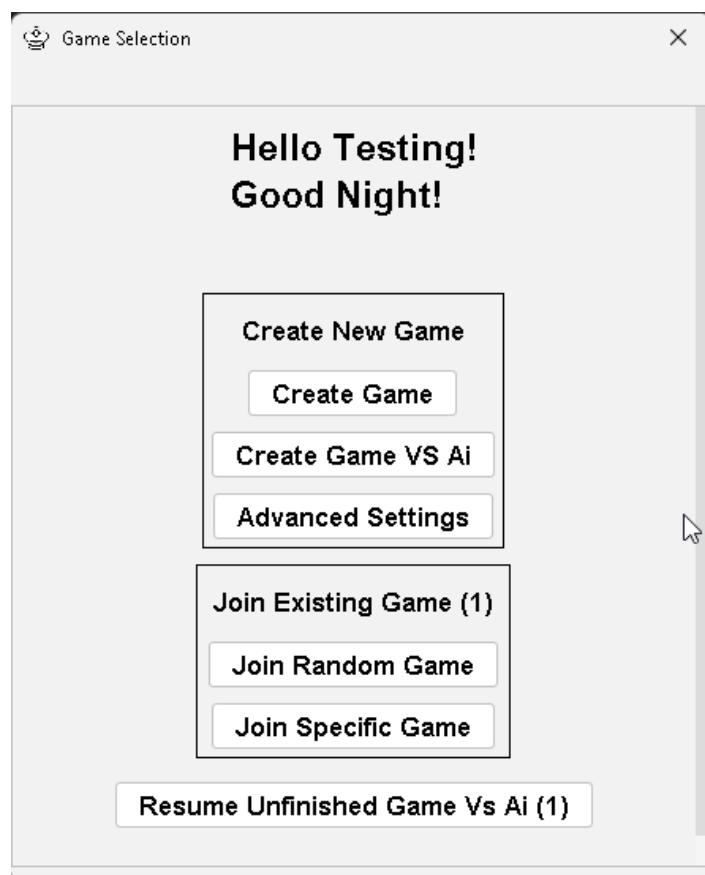


לאחר לחיצה על Delete All Unfinished Games, ימחקו כל המשחקים של השחקן שעדיין לא נגמרו מול AI, ווונגן חלון שמציג את תגובת השרת לבקשת המהיקה.

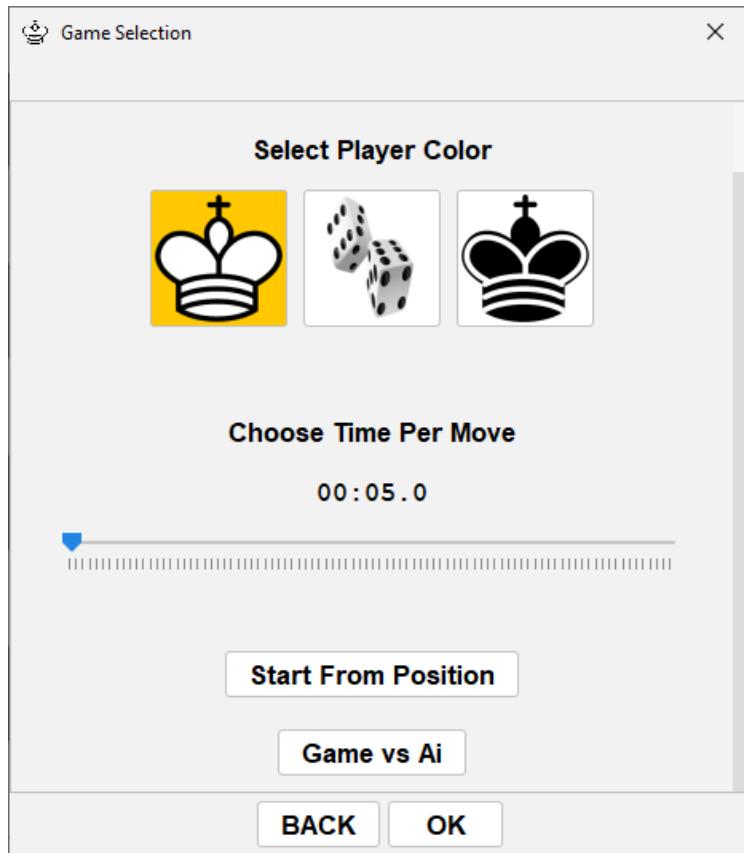
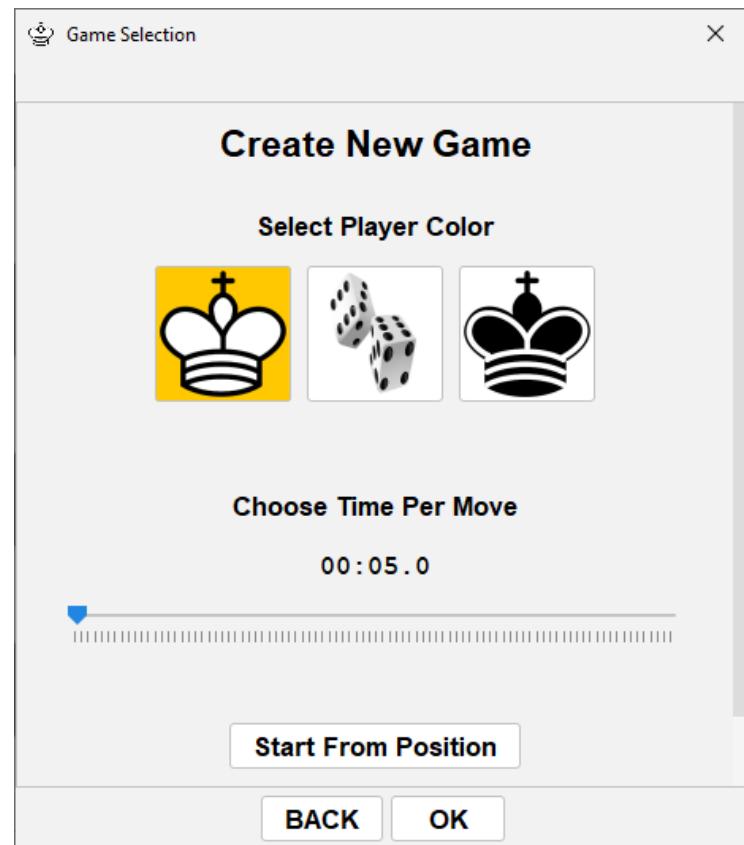
אתחול משחק



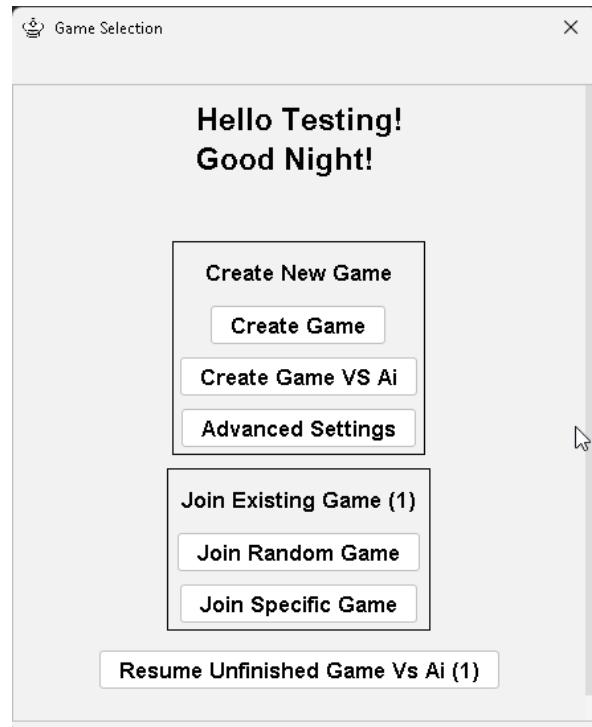
כדי להתחיל לשחק, אפשר לבחור בAI Play Game כדי להתחיל משחק מול הבינה המלאכותית מיד. או Play Game כדי לשחק עם שחפן רשות אחר. במידה וקיים משחק של שחפן אחר שממתין ליריב, המשחק יתחל מיד. אחרת, יש להמתין עד שחפן אחר ייצור משחק ואז יתחל המשחק. בלחיצה על Play Game vs AI יפתח הדיאלוג הבא:



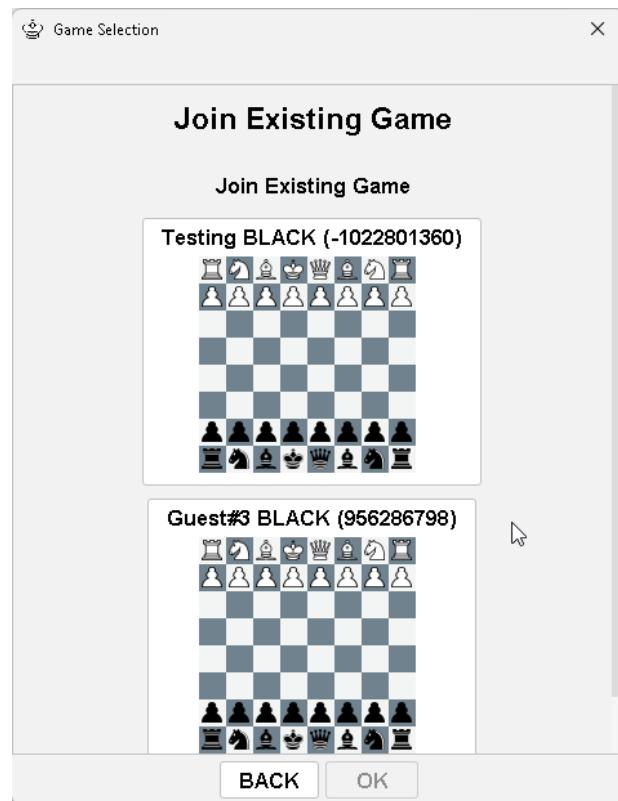
ייצור משחק סטנדרטי וימתין לשחקן אחר שיתחבר למשחק שנוצר. Create Game
ייצור משחק מול הבינה המלאכותית ויתחיל מיד. Create Game VS AI
יציג את החלון הבא: Advanced Settings



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.



ניתןJoin Random Game
לראות את כמות המשחקים שיש בשרת שממתינים לשחקן שני.
יציג את החלון הבא:Join Specific Game

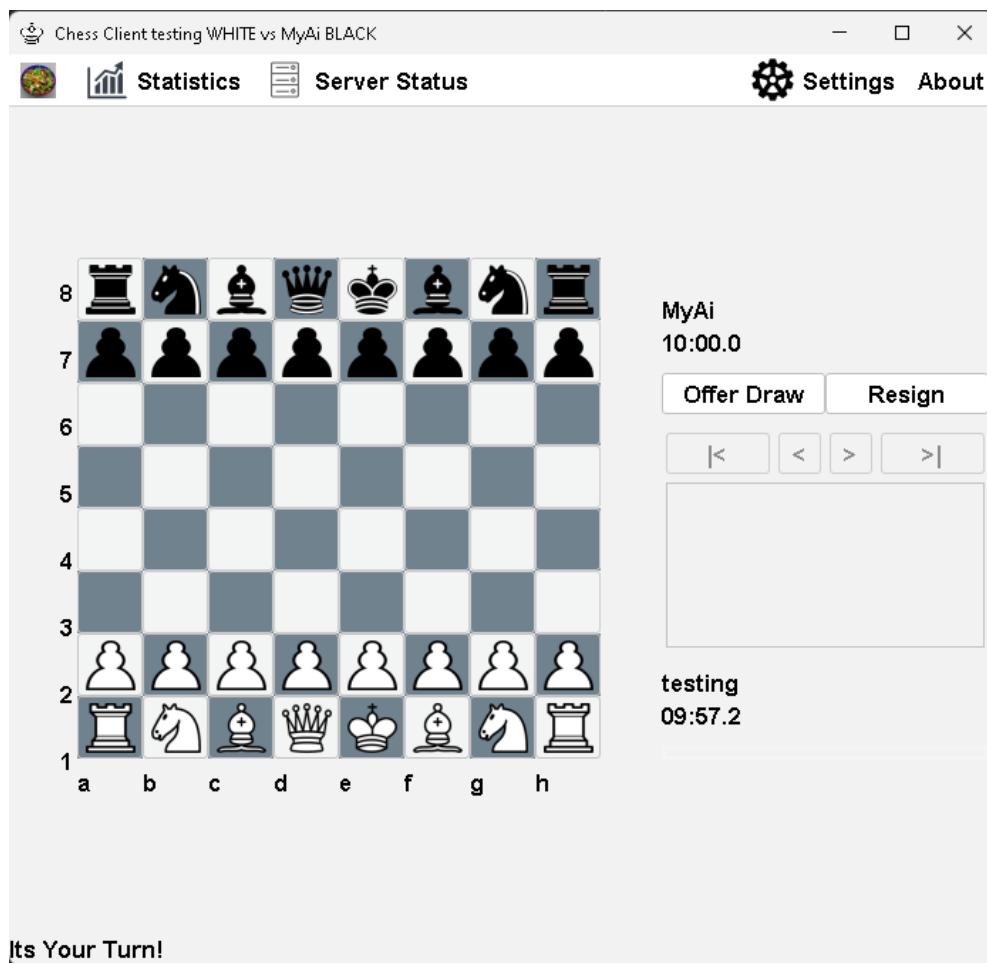


(רישימה תהיה של המשחקים שזמינים בשירות בעת ההרצה) עבור כל משחק, ניתן לראות את שם המשתמש של היוצר של המשחק, והעמדה שמננה השחקן יתחיל לשחק. בבחירה של אחד המשחקים ולחיצה על OK המשחק יתחל.

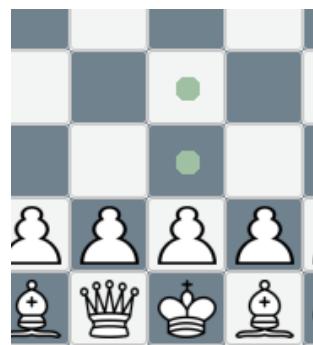
Resume Unfinished Game VS ai
יציג רשימת משחקים שהופסקו מול הבינה המלאכותית. האופציה תהיה זמינה רק לשחקן שמור. לאחר ולאורחים לא נשמרים משחקים שהופסקו.

מהלך המשחק

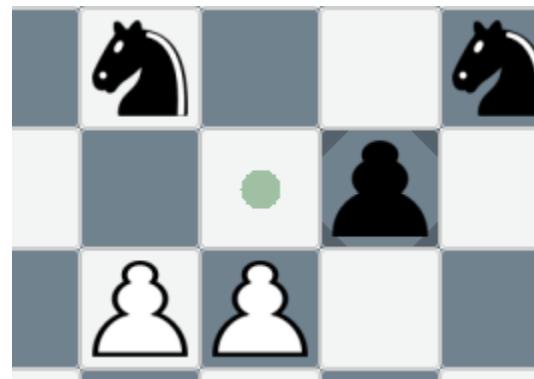
לאחר אתחול המשחק, ייטען הלוח עם העמדה שנבחרה(בכ"כ עמדת הפתיחה) והלוח יהיה מכובן בהתאם לצבע של שחקן הרשת. (צילומי המסך יהיו עבור השחקן הלבן מעמדת הפתיחה)



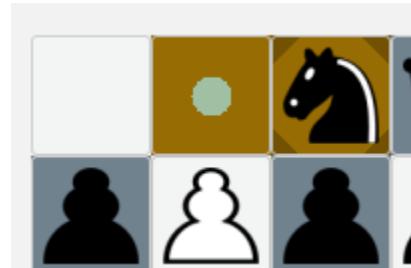
לפי תווית הסטטוס או הטימר שרצ ניתן לראות שתורנו, ולכן עליינו לבצע מהלך. בלחיצה על אחד מהכליים שלך, יסומנו כל המשבצות שאליים ניתן לכת באופן חוקי עם אותו כל. לדוגמה, בעת לחיצה על הרגלי בe2 בעמדת הנוכחתה:



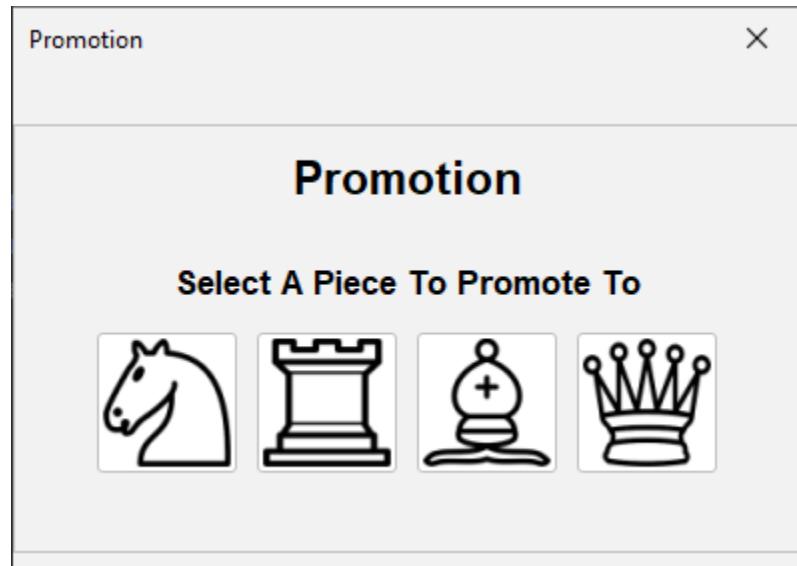
ניתן להחליף את הכלי שברצונך להזיז ע"י לחיצה שנייה על אותו הכלי (מה ש לבטל את הלחיצה הראשונה) או לחיצה על כלិ אחר. לאחר שבחירה כלិ ויעד, הלוח יתעדכן ומהלך שלך נשלח לשרת. במקרה שבו נבחר כלិ שעבורו יש אפשרות אפשרית, היא מסומן בצורה זו:



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



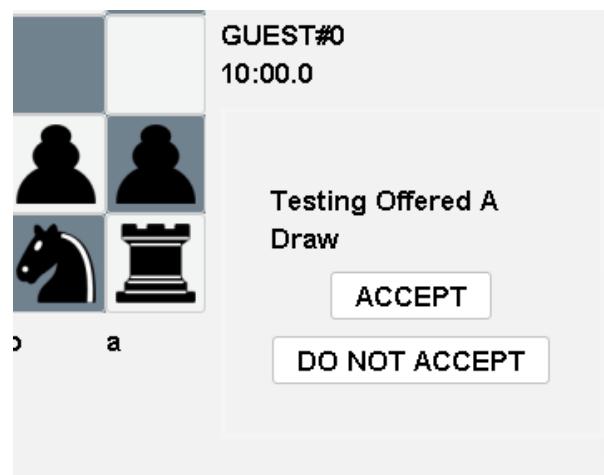
כששחקן מבצע מהלך קידום, תוצג החלונית:



שנה תתבקש לבחור סוג כלי לקדם אליו.

הצעת תיקו

כל עוד לא נגמר המשחק, כל שחקן יכול להציע תיקו לשחקן השני. לא משנה אם תורו. כדי להציע תיקו לוחצים על הכפתור Offer Draw. כל שחקן יכול להציע תיקו פעמיים אחד בלבד במהלך כל משחק. לאחר שהיריב הצעה תיקו, מוצגת הודעה בצד 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 לרוуз. כשהוא מגיע ל-10 השניות האחרונות הטימר הופך לאדום. אם נגמר לך הזמן לפני שביצעת מהלך, הפסדת.

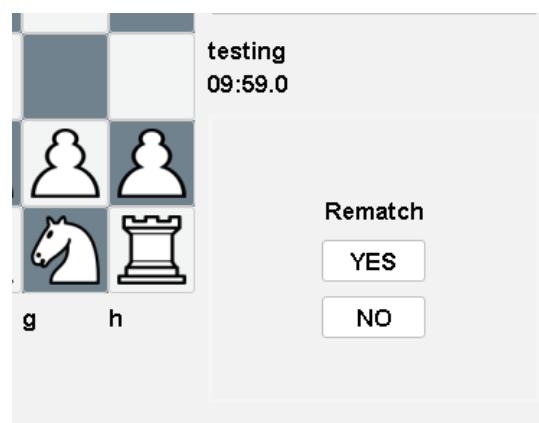
התנטקות

במקרה של התנטקות, המשחק נגמר מיד. אם המשחק היה בין 2 שחקנים אנושיים, המשחק מוכרע בתור הפסד של השחקן שהתנטק. 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 התנטק, 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 מנוקודת 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 לקוחות ובמקביל (שימוש-ב-Threads)
- בעת חיבור לוקוח, על המערכת להציג מסך LOGIN שמאפשר ללקוח להתחבר משתמש רשום (עם שם משתמש וסיסמה) או כאורח או להירשם כמשתמש חדש. אם הוא מתחבר כמשתמש רשום, נתנוינו בדקו 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 המשחק 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 ללקוח נוספת שיהיה בין הזוג שלו למשחק. במקרה זה המשחק לא מתחילה עד אשר מתחבר עוד ללקוח. 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 – לקוחות) ושניהם ביצעו ההתחברות 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 היישובים, חוץ שליחת וקבלת הודעות. ורוב העומס מוטל על השרת. מה שמאפשר חיבור של מחשבים לא עצמאיים 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 `ucanaccess`.

טבלה: Users

פקוד הטבלה הינו לאחסן את פרטיים של המשתמשים. הוא משתמש לאיומות פרטיים 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	-2-500x500.jpg
testing	123456	1650018895	tos/414768.jpg

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 שחקן 1 במשחק
- Player2: מחרוזת קצרה, שמכילה את שם המשתמש של שחקן 2 במשחק
- ArchivedGameInfo: מחרוזת ארוכה, שמכילה סיריליזציה של אובייקט מסווג SavedGame
- עוד פרטים, ראה את דיאגרמת 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 השניות מאז 12 בלילה ב-1/1/1970 [לעתה](#)
- מידע שמייצג את תאריך הייצירה של המשחק

טבלה UnfinishedGames

תפקיד הטבלה הינו לאחסן את פרטי המשחקים שהופסקו בין משתמשים שמוררים לבינה מלאכותית, כדי שהשחקנים יוכלו המשיכם בפעם הבאה שיינסו.

עמודות:

- GameID: מפתח, מחרוזת קצרה, שמכילה את מזהה 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
- עוד פרטים, ראה את דיאגרמת 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-1/1/1970 [לעתה](#)
- מידע שמייצג את תאריך הייצירה של המשחק

Threads 4.2.5

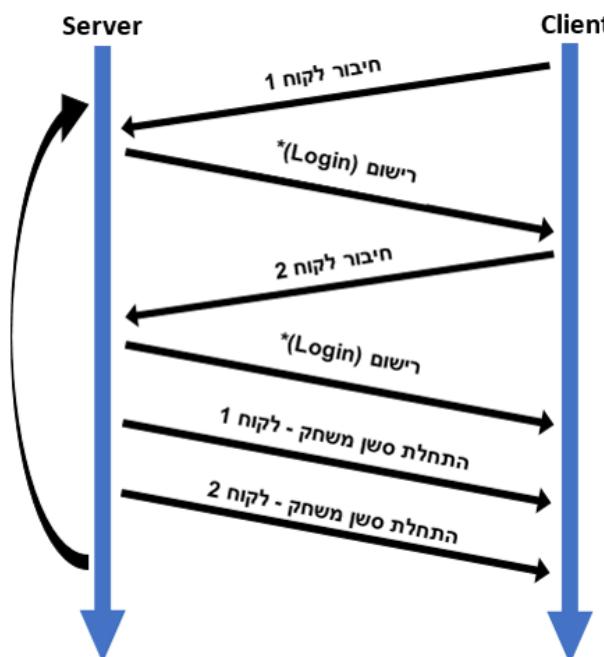
תהליכיון הינו אובייקט שמאפשר שזירה של קוד להרצה במקביל לתחליק הראשי או תהליכיים אחרים. מה שמאפשר ביצוע מספר פעולות במקביל. תכונה שהשרת שלנו חייב. כדי שיווכל לניהל מספר משחקים ושחקנים במקביל. השתמשנו בו בפרויקט כדי לניהל את כל המשחקים שרצים בשרת במקביל, ולקבל ל Kohoות דרך שקע השירות. פעולה חוסמת, שהייתה מוגדרת לנו לגמרי את השרת ללא שימוש בתהליכיים. עוד שימוש היה במינימקס. שבו החיפוש מפוץ בין מספר תהליכיים, וכך מגיע לתוצאות טובות משמעותית. עוד שימוש היה גם אצל השרת וגם אצל הלוקה במחלקה AppSocket. שבו יש תחליך שככל הזמן מזין להודעות מהשרת או מהлокה ומנתב אותן לעדן.

4.3 תהליכיים עיקריים וזרימת המידע

בסייף זה נסביר את התהליכיים העיקריים מרכיבת התוכנה ואת זרימת המידע ביניהם. את התהליכיים נציג بصورة גרפית באמצעות **תרשיימי רצץ** (Sequence Diagrams) שיתארו את זרימת המידע (פרוטוקולי השיח) בין השרת לлокה הנקודות, שידוך בין שני שחקנים להתחלה משחק, התנהלות ומחזור המשחק ביניהם, סיום משחק ועוד.

4.3.1 תחליך הראשי

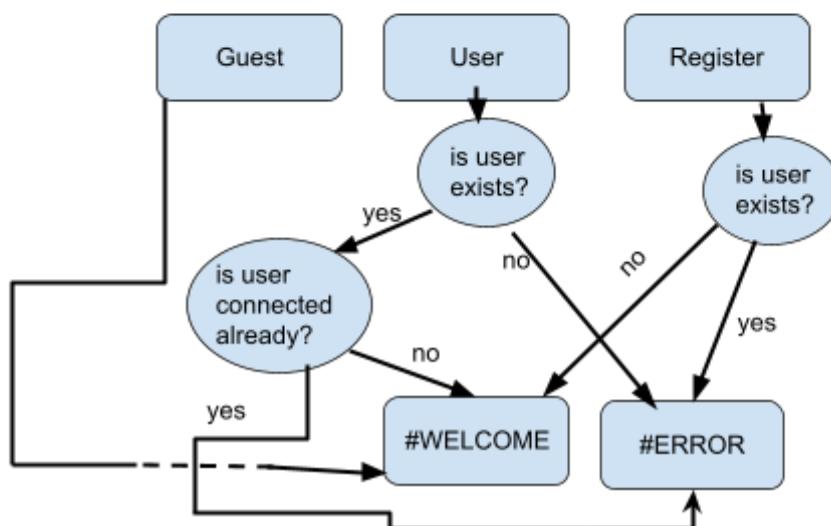
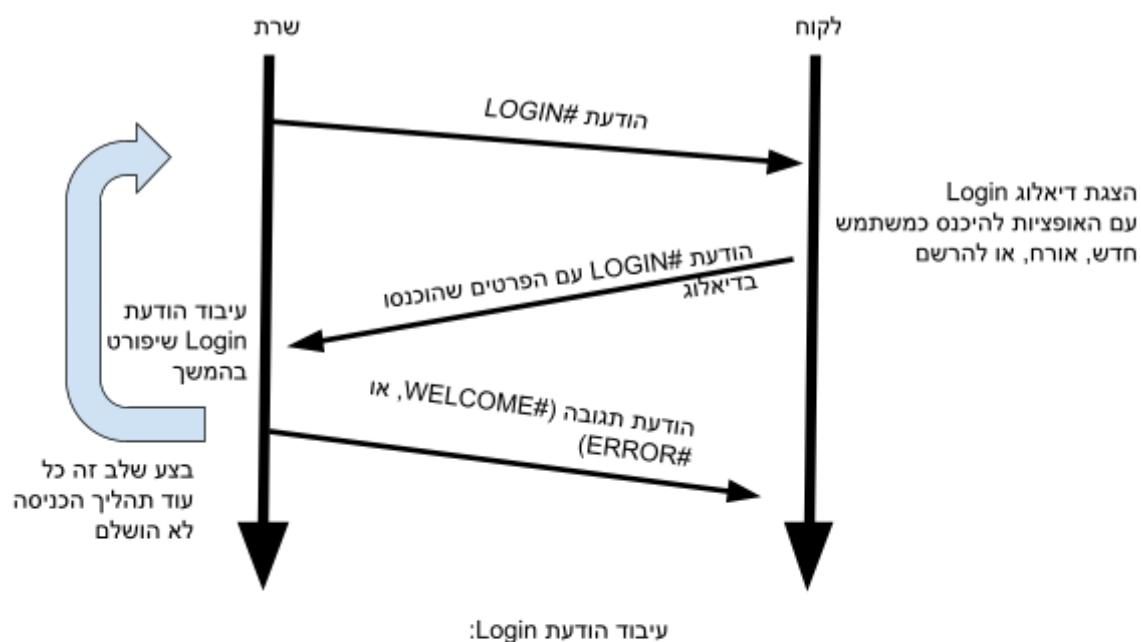
להלן תרשימים רצף עבור **התחליך הראשי** "משחק רשות – מרובה שחקנים ומשחקים":



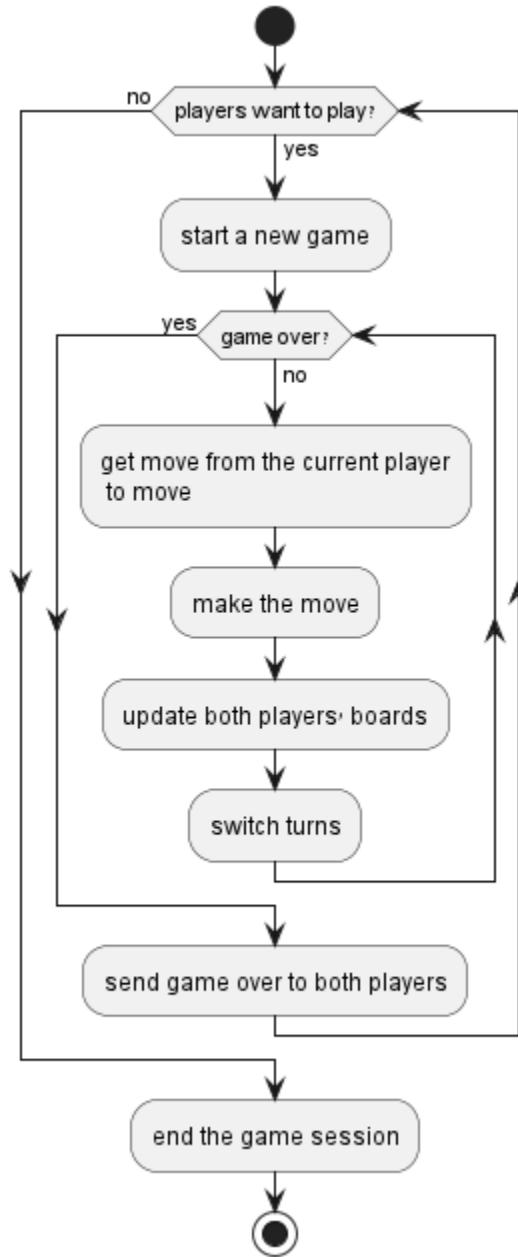
תרשים הרצף הנ"ל מתאר את פרוטוקול השיח בין הרשות ללקוחות, עברו התהלייר הראשי בו הרשות ממתין להיבור לckoות, בלולה אינסופית, ובמעבר כל זוגckoות שהתחברו בהצלחה, הוא משדק ביניהם וווצר עבורם משחק.

שני החצים האנכיים, מתארים שני הצדדים בשיח - הרשות (משמאלה) והckoות (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 שמועבר עם ההודעה.

להלן תרשימים רצף עבור תת-טהלייר "התחברות לckoות - Login :

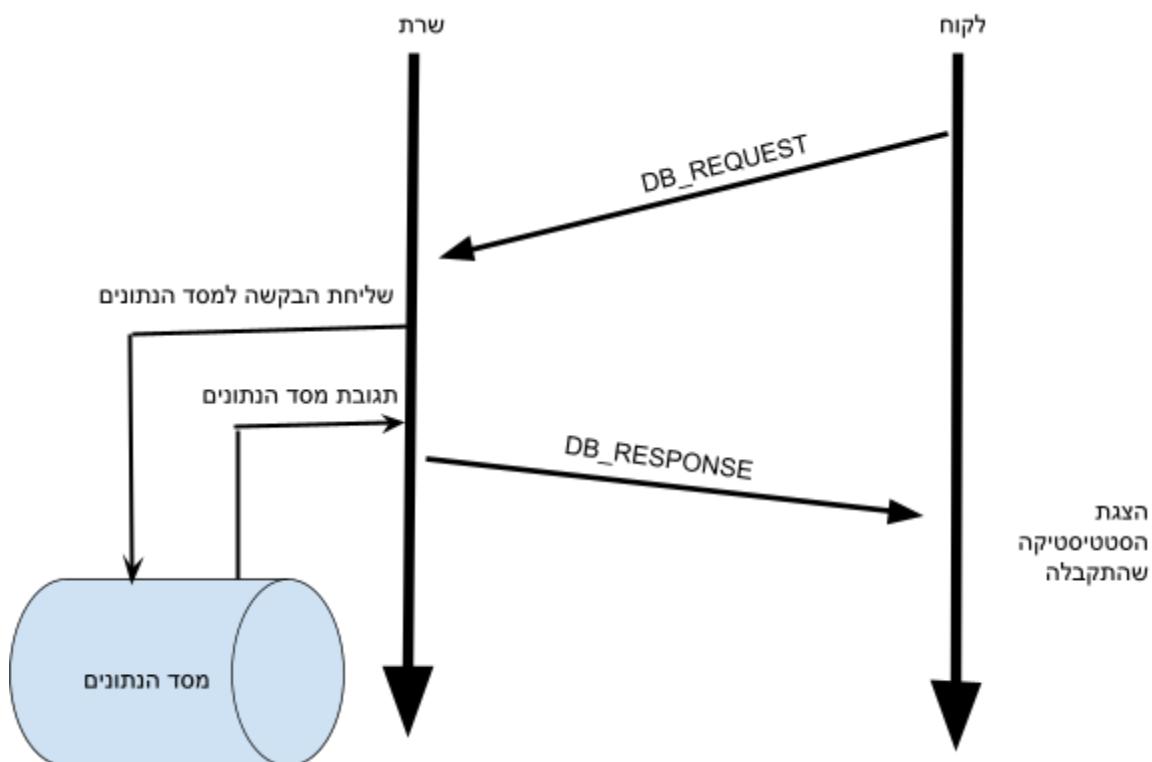


תהליך הGame Session



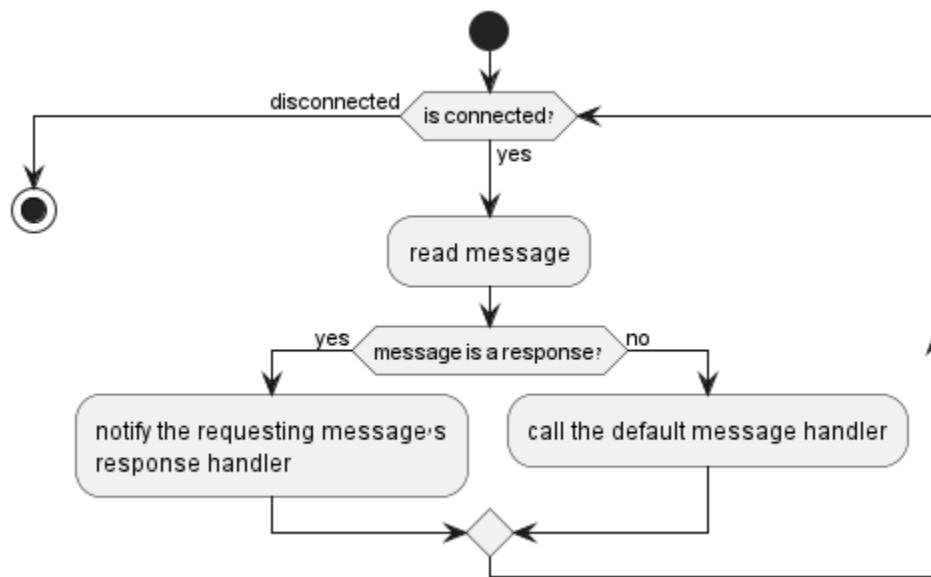
כששני שחקנים מתחביבים לשחק זה עם זה, נוצר ביניהם סשן משחק. שבו מתחביב משחק חדש. בתוך לולאה: כל עוד לא נגמר המשחק, הרשת מבקשת מהשחקן שתוורו לשחק את המהלך, מעדכן את הלוחות של 2 השחקנים, מחליף תור וחוזר חלילה. לאחר שנגמר המשחק, נשלחת הודעה סוף משחק ל-2 השחקנים, והם נשאלים האם הם רוצחים לשחק שוב. במידה וכן, נוצר ביניהם משחק חדש. אחרת, הסשן נגמר.

בקשת סטטיטיקת:



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

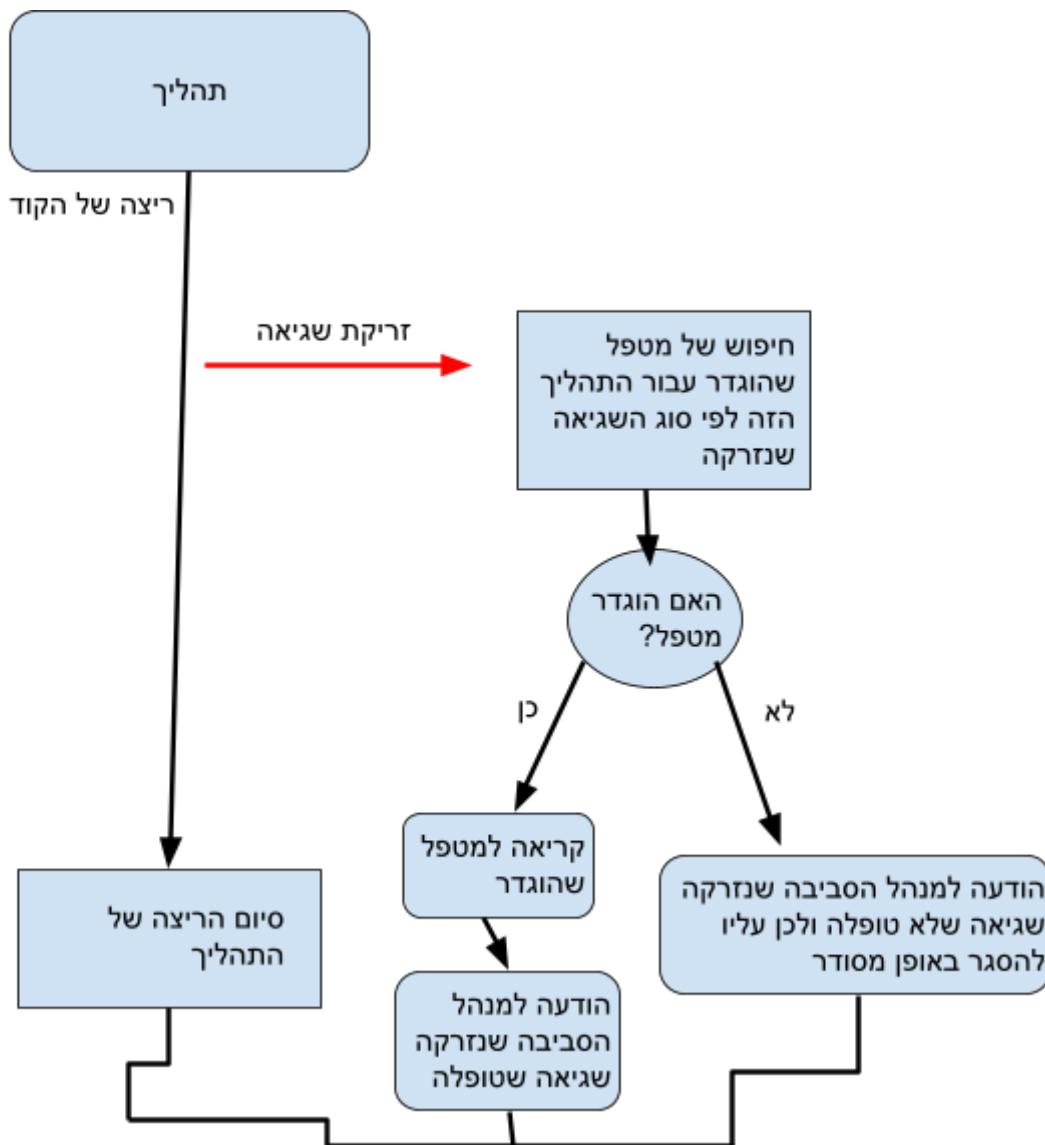
תהליך שליחת וקבלת הودעות



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

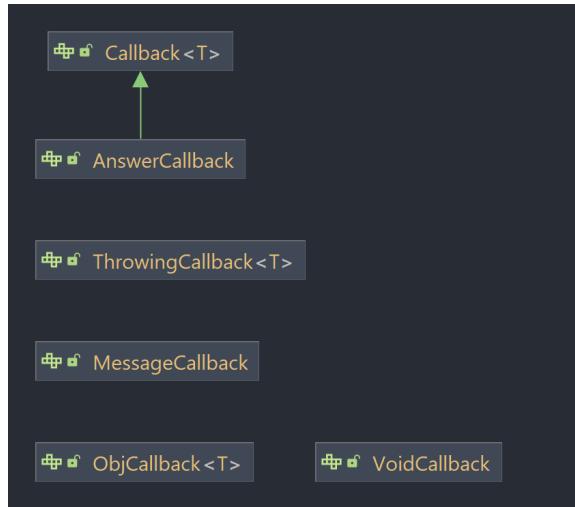
בסעיף זה נציג את כל המחלקות של הפרויקט.... יש להוסיף עוד כמה משפטים (3-2 שורות) המתארים באופן כללי את המחלקות והחלוקת לשני המודולים, כפתיחה לקריאת הסעיפים הבאים.

תרשיימי UML של המחלקות הקשורות ללקוח ולשרת

להלן תרשימי UML של המחלקות הקשורות ללקוח ולשרת, הבנוי מ90 מחלקות שעלייהן אכתוב פירוט מעמיק....

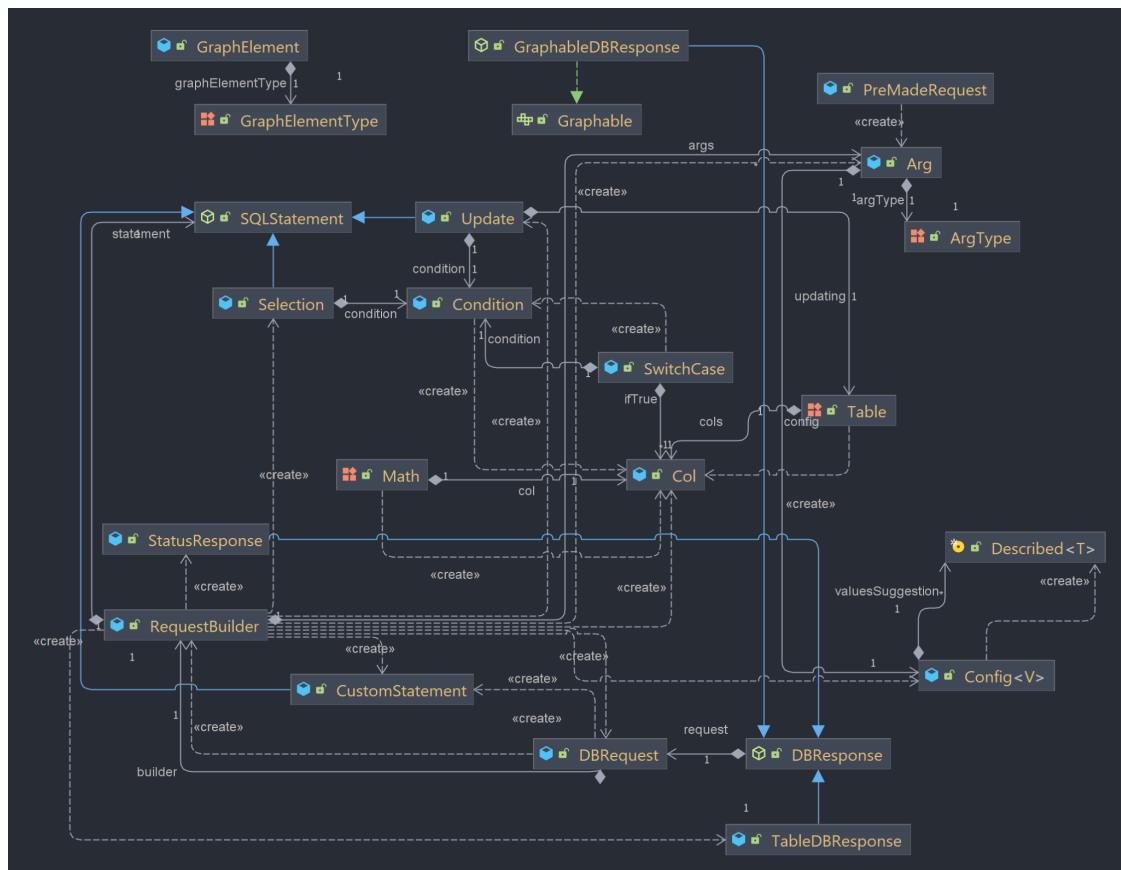
Callbacks

משמשים לקרוא א-סינכרונית של פעולות.



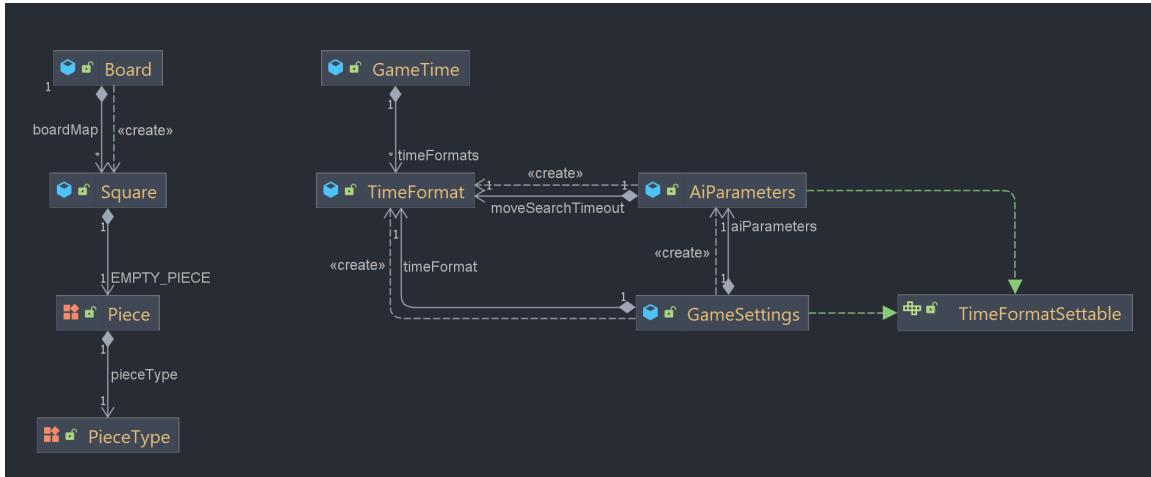
DBActions

מחלקות הקשורות למסד הנתונים וביצוע פעולות עליון



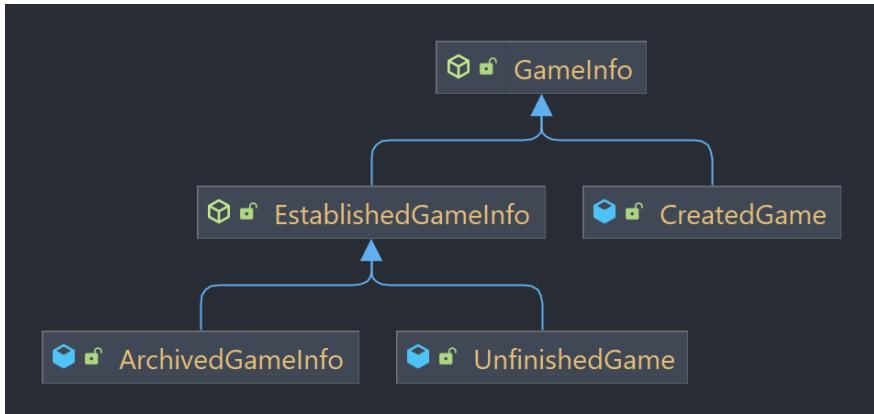
GameSetup

מחלקות הקשורות לאתחול וניהול המשחק



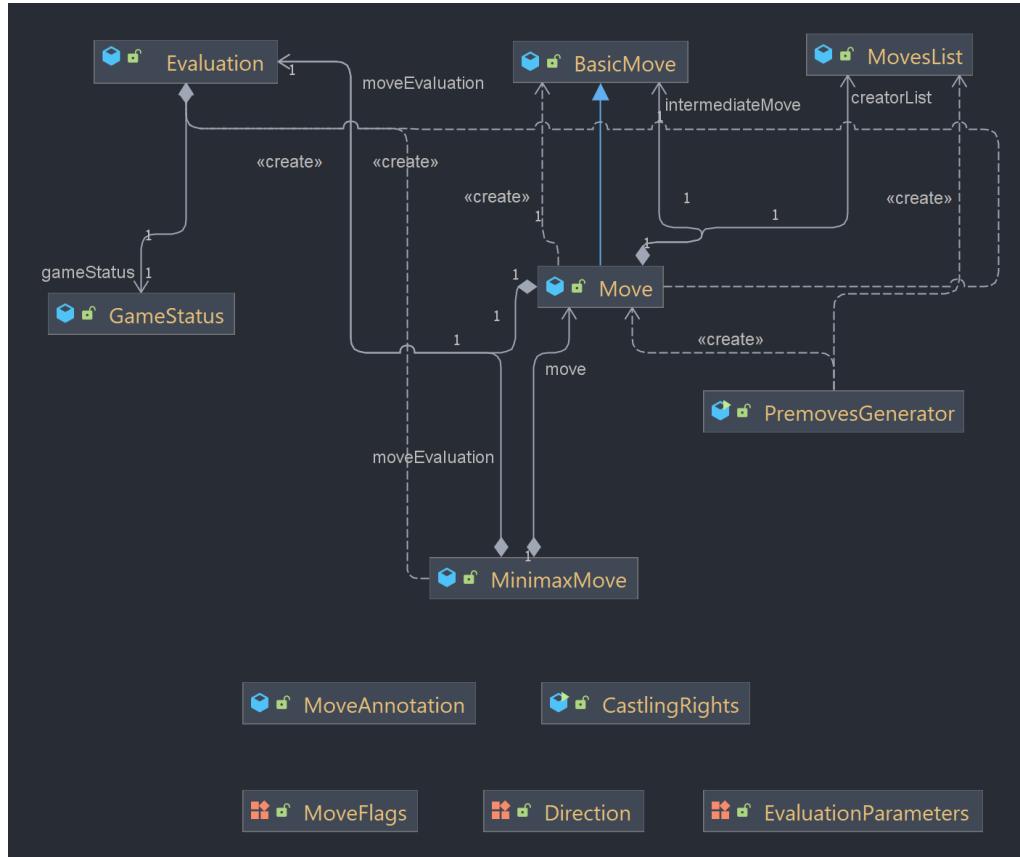
SavedGames

ייצוג משחקים שמורים

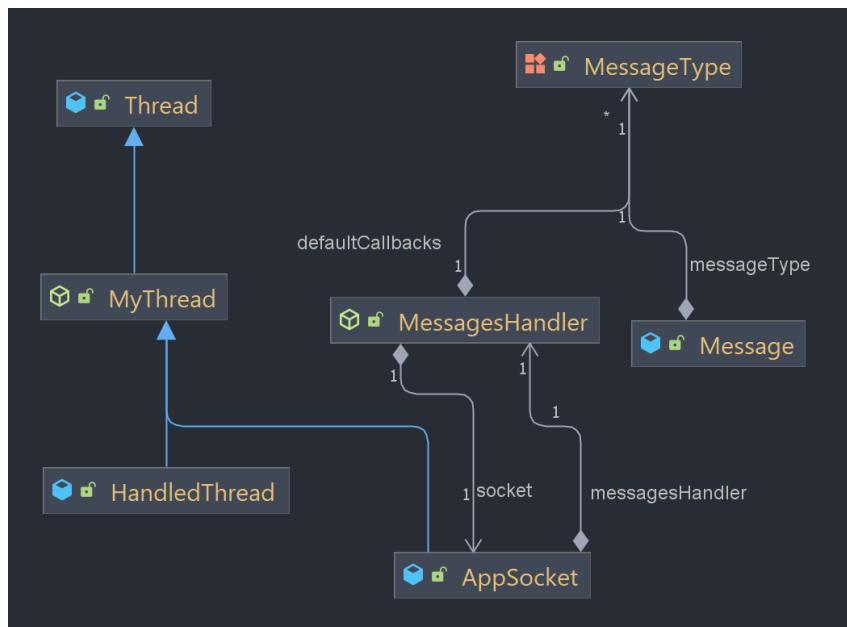


Moves

ייצוג, ייצור, וניהול מיליכים.

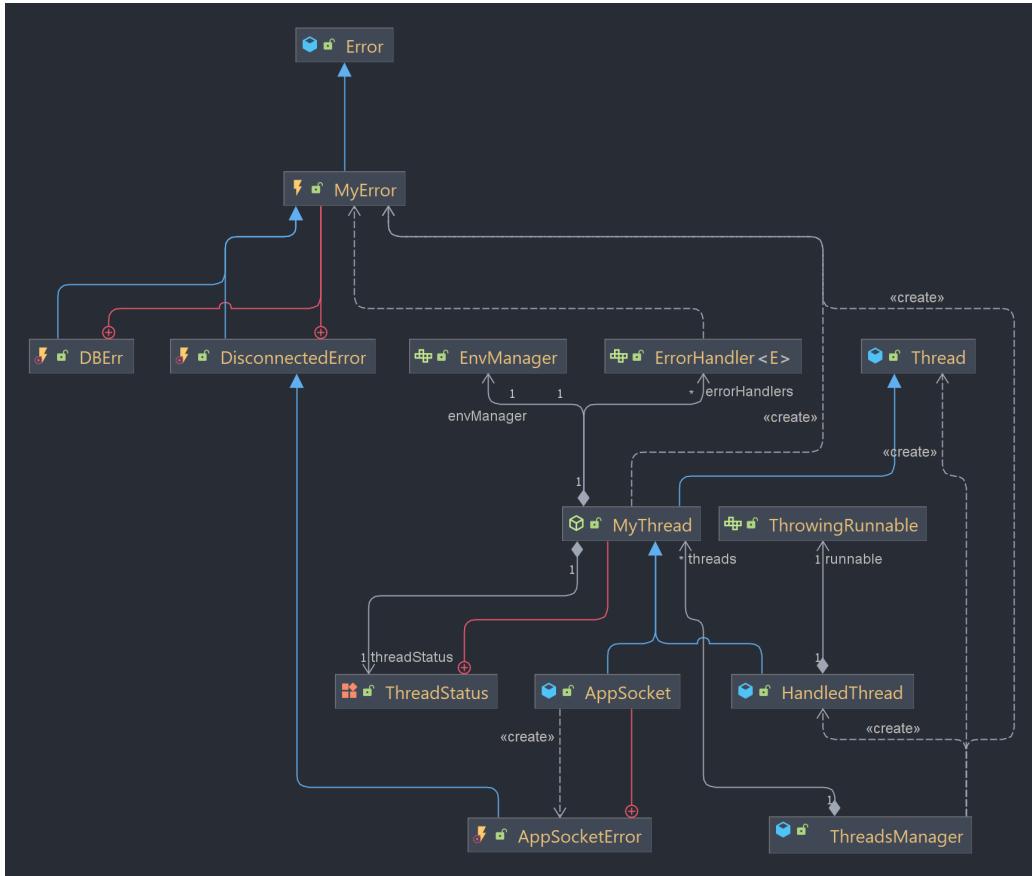


Networking



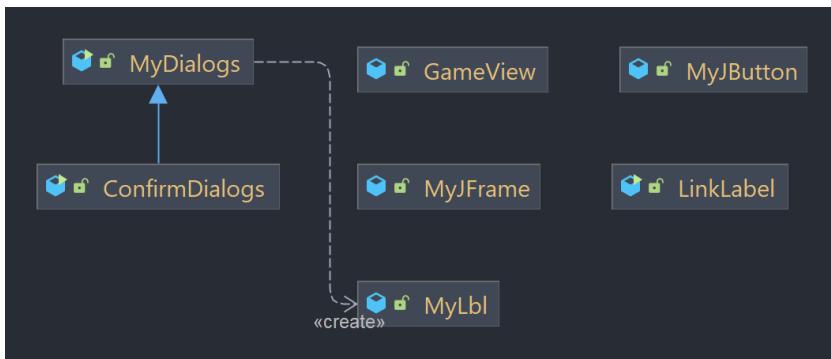
Threads

תהליכיונים



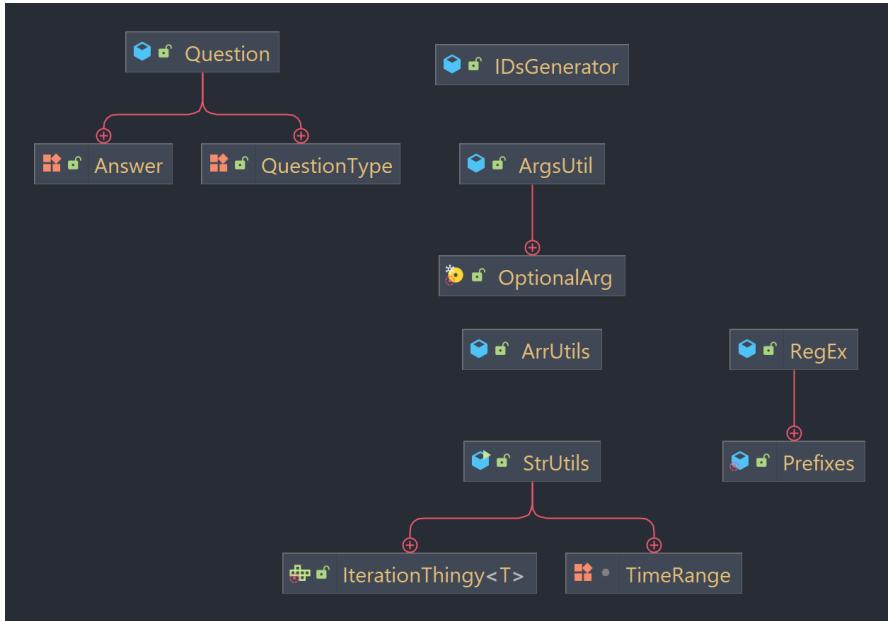
UI

רכיבים גרפיים שימושתיים ללקוח ולשרה



Utils&Misc

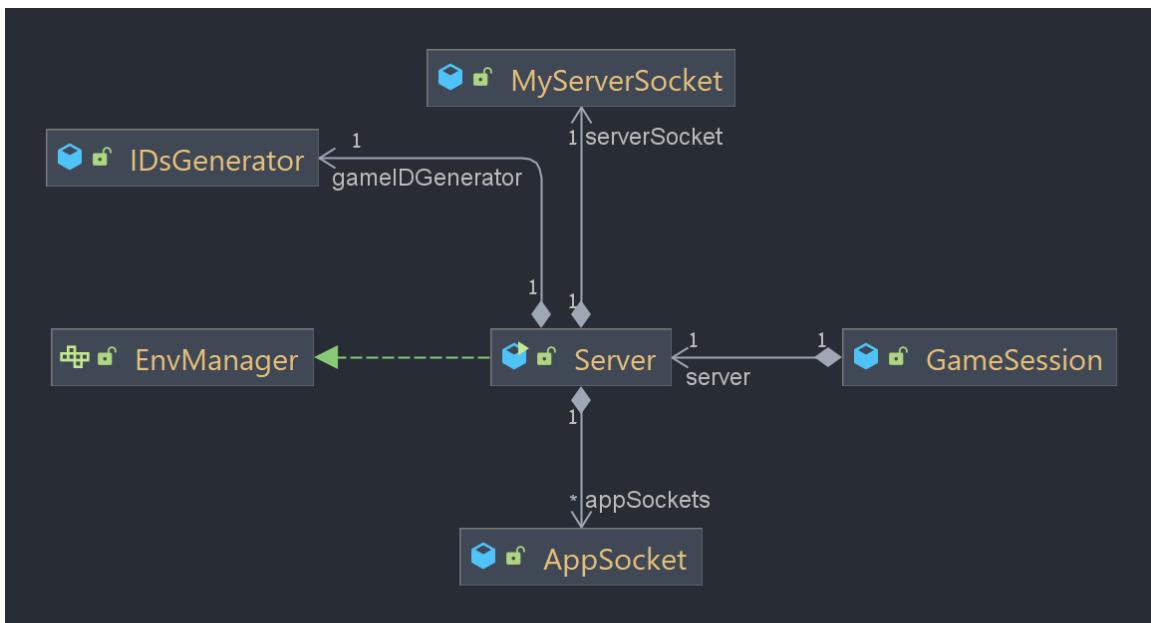
מחלקות שירות ושותנות*



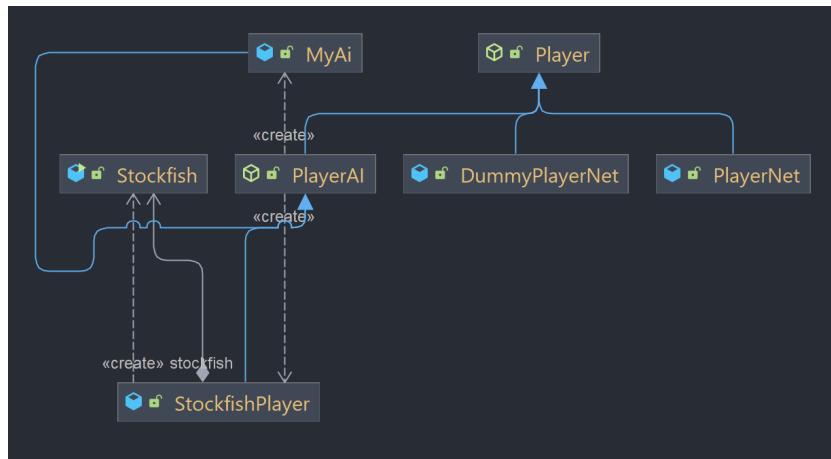
4.4.1 תרשימי UML של המחלקות הצד השרת

להלן תרשימי UML של הצד השרת, הבניי 48 מחלוקת שעלייהן נכתב פירוט מעמיק...

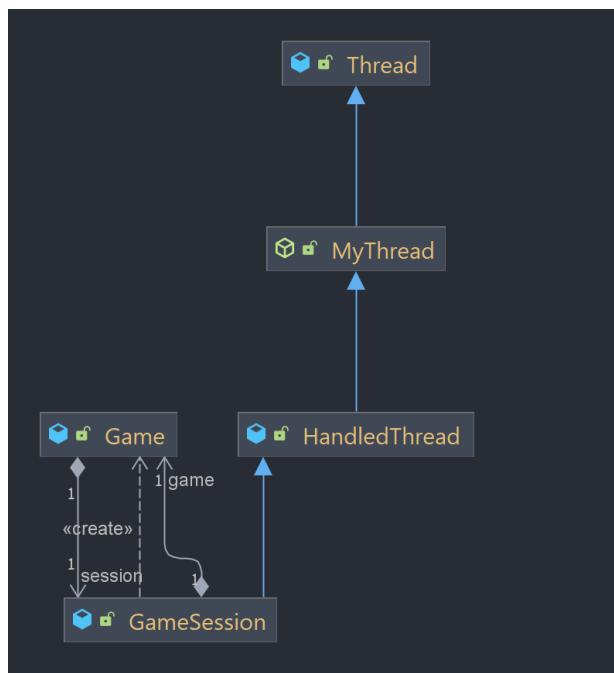
Server



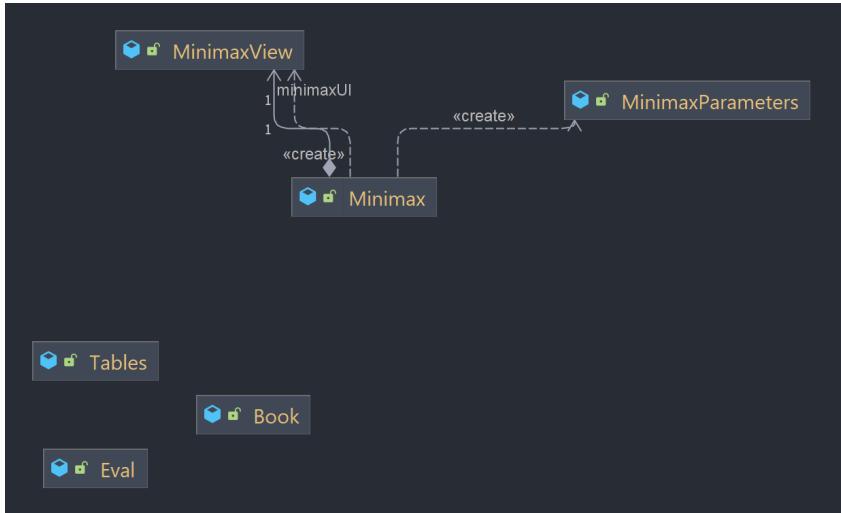
Players



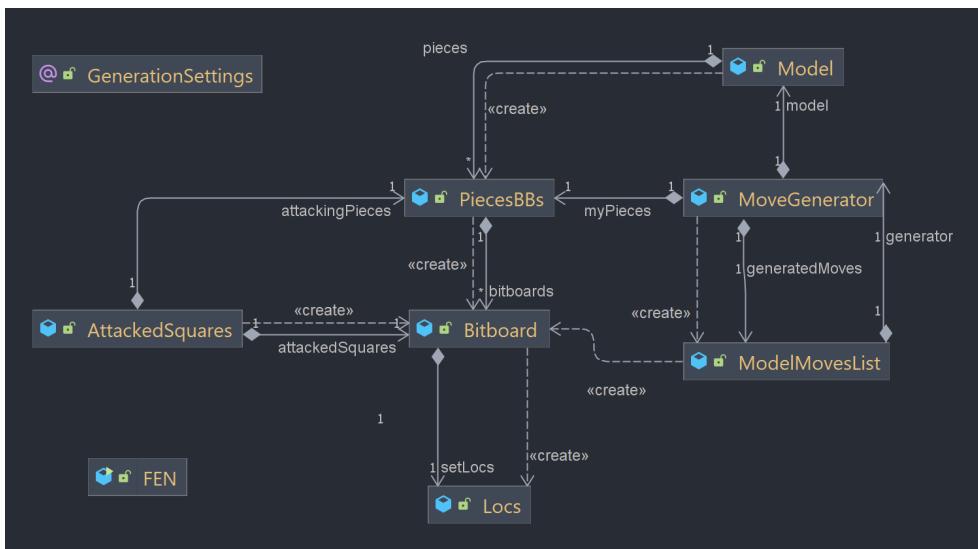
Game



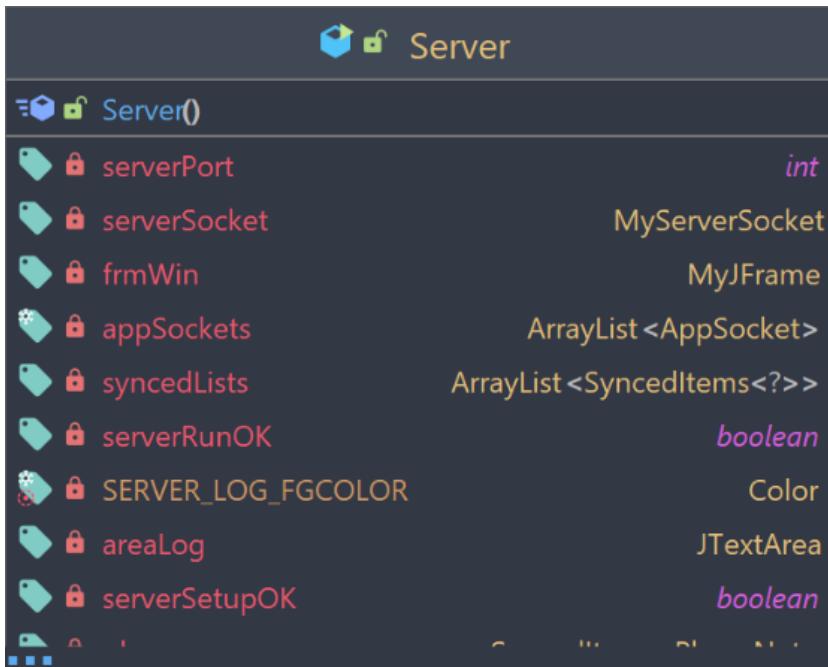
Minimax



Model



4.4.2 תייעוד המחלקות מצד השירות



Server - represents a chess server

Server()

Constructor for ChessServer.

runServer public 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.

handleClient private void **handleClient**(AppSocket playerSocket)
Handle client in a separate thread to allow for concurrent client handling.

Parameters: playerSocket - the socket to the player

login public ver14.Players.PlayerNet.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

responseToLogin private 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

gameSetup public 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

playerDisconnected public 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 בקובץ עצמו

ServerMessagesHandler	
ServerMessagesHandler(Server, AppSocket)	
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 בקובץ עצמו

UsernameSuggestions	
UsernameSuggestions()	
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()
createSuggestions public static ArrayList<String> createSuggestions(String
username)
Create suggestions array list.
```

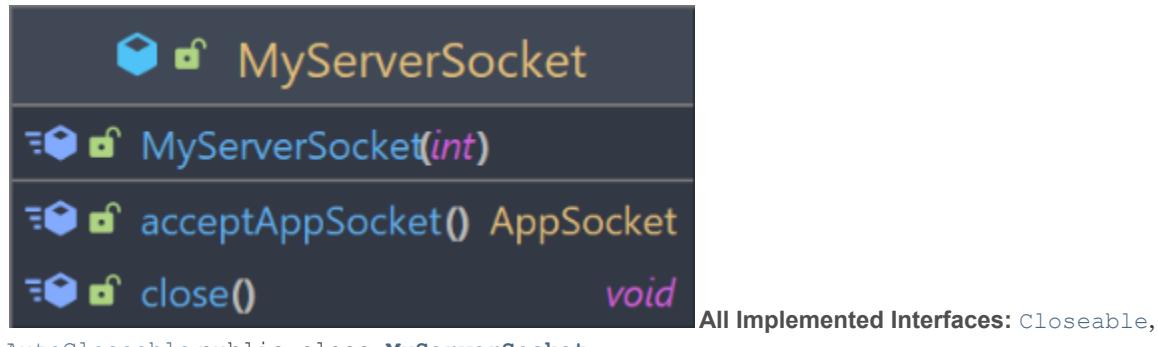
Parameters: username - the username **Returns:** the array list
את שאר הפעולות ניתן לראות בקובץ עצמו

Enclosing class: UsernameSuggestions private static interface
UsernameSuggestions.MatchIterations

Match iterations.

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

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



AutoCloseable public class **MyServerSocket**
extends ServerSocket

My server socket - represents a server sockets accepting AppSockets.

```
MyServerSocket(int port)
Instantiates a new My server socket.
```

```
acceptAppSocket public ver14.SharedClasses.Networking.AppSocket acceptAppSocket()
Accept app socket.
```

Returns: the accepted app socket
את שאר הפעולות ניתן לראות בקובץ עצמו



DB - utility class for communicating with the database.

DB ()

runUpdate public static 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

getConnection public static Connection **getConnection()**

create a connection to the db.

Returns: the created connection to the db.

runQuery public static 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

request public static 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 בקובץ עצמו Game

GameSession	
<code>GameSession(UnfinishedGame, Player, Player, Server)</code>	
<code>GameSession(String, Player, Player, GameSettings, Server)</code>	
<code>GameSession(EstablishedGameInfo, Player, Player, Server)</code>	
<code>GameSession(GameInfo, Player, Player, Server)</code>	
<code>game</code>	Game
<code>gameID</code>	String
<code>server</code>	Server
<code>creator</code>	Player
<code>p2</code>	Player
<code>getSyncableItem()</code>	SyncableItem
<code>...</code>	

All Implemented Interfaces: Runnable, 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)`

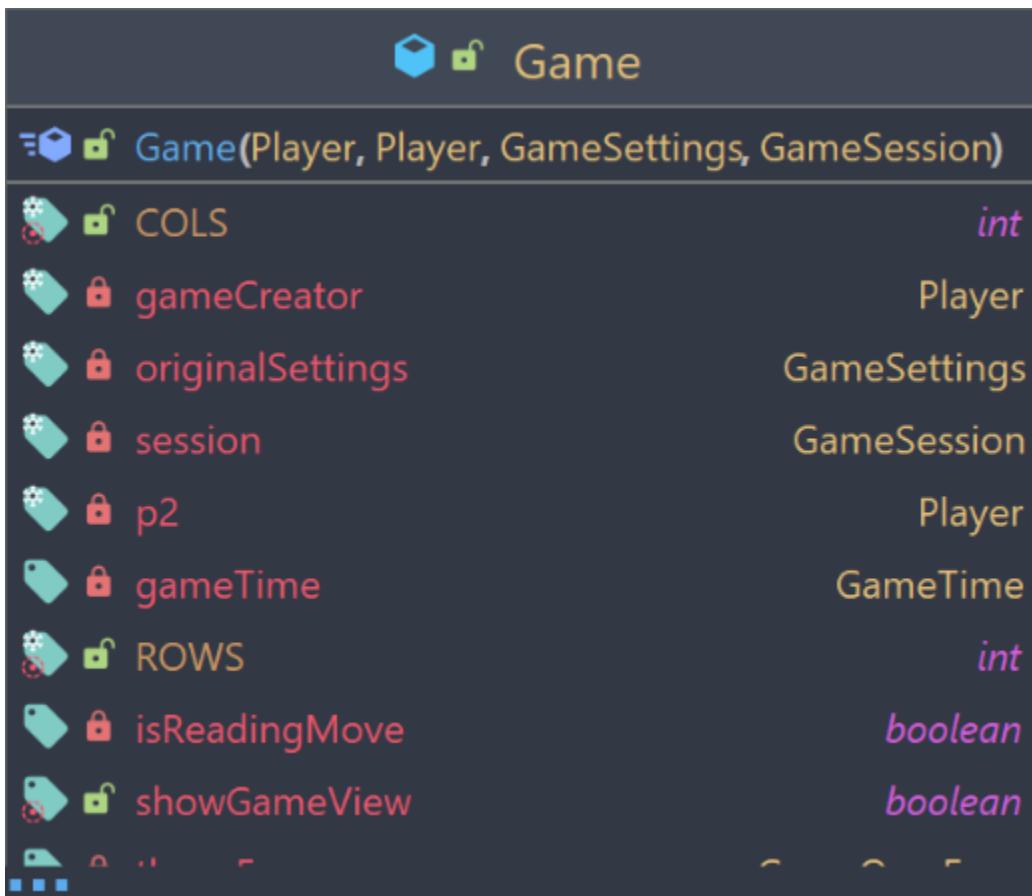
Instantiates a new Game session. resuming an unfinished game.

`handledRun` public 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



```
public class Game
```

Game - represents a game between two Players.

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

Instantiates a new Game.

```
runNewGame public GameStatus runNewGame()
```

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

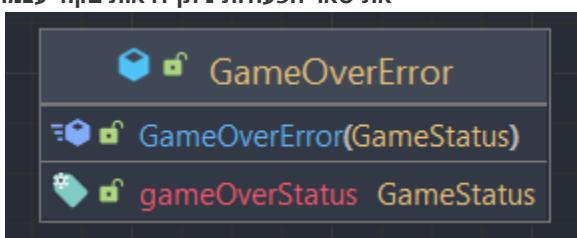
Returns: the game over status

```
playTurn private 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 עצמן



All Implemented Interfaces: Serializable 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.

	PlayerDisconnectedError
	PlayerDisconnectedError (Player)
	disconnectedPlayer Player
	createGameStatus () GameStatus
	getDisconnectedPlayer () Player

All Implemented Interfaces: Serializable 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 [עצמן Model](#)

Eval	
	Eval(Model, PlayerColor, boolean)
	Eval(Model, PlayerColor)
	Model
	PlayerColor
	Evaluation
	double
	PlayerColor
	PlayerColor
	double
	boolean
...	

All Implemented Interfaces: Serializable 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.
```

checkGameOver private 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.
calcEvaluation private void **calcEvaluation ()**
 if the game isn't over, an evaluation is calculated.

materialSum private int **materialSum (PlayerColor playerColor)**

the sum of all the pieces' values of a player

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

pieceTables private 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

Enclosing class: Tables

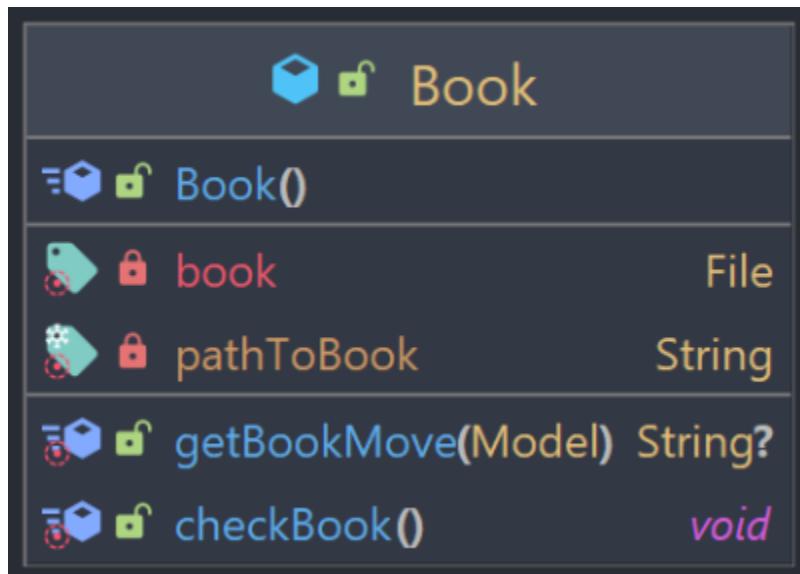
public static class **Tables.PieceTable**

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.

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



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

Book()
getBookMove public static 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 בקוד עצמו



public class **Minimax**
Minimax - represents my implementation of a multithreaded minimax algorithm.

getBestMove public Move **getBestMove**(PlayerColor player)
Gets best move using minimax.

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

```
minimaxRoot private 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
startMultithreaded private 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	
MinimaxParameters (Model, boolean, int, int, PlayerColor, int)	
MinimaxParameters (Model, boolean, int, PlayerColor)	
model	Model
b	int
isMax	boolean
a	int
maxDepth	int
currentDepth	int
minimaxPlayerColor	PlayerColor
isMax()	boolean
...	

```
public class MinimaxParameters
parameters used by the Minimax.
```

את שאר הפעולות ניתן לראות בקוד עצמו



```
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.
```

@ GenerationSettings	
➡️ 🔒 EVAL	<i>int</i>
➡️ 🔒 QUIESCE	<i>int</i>
➡️ 🔒 ANNOTATE	<i>int</i>
➡️ 🔒 ANY_LEGAL	<i>int</i>
➡️ 🔒 LEGALIZE	<i>int</i>

```
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.

MoveGenerator	
≡🔒 MoveGenerator(Model, int)	
➡️ 🔒 generationSettings	<i>int</i>
➡️ 🔒 movingPlayerColor	PlayerColor
➡️ 🔒 numSquaresToEdge	<i>int</i> [][]
➡️ 🔒 kingMoves	ArrayList<Location>[]
➡️ 🔒 logicBoard	Board
➡️ 🔒 knightMoves	ArrayList<Location>[]
➡️ 🔒 myPieces	PiecesBBs
➡️ 🔒 model	Model
➡️ 🔒 generatedMoves	ModelMovesList
...	

```
public class MoveGenerator
```

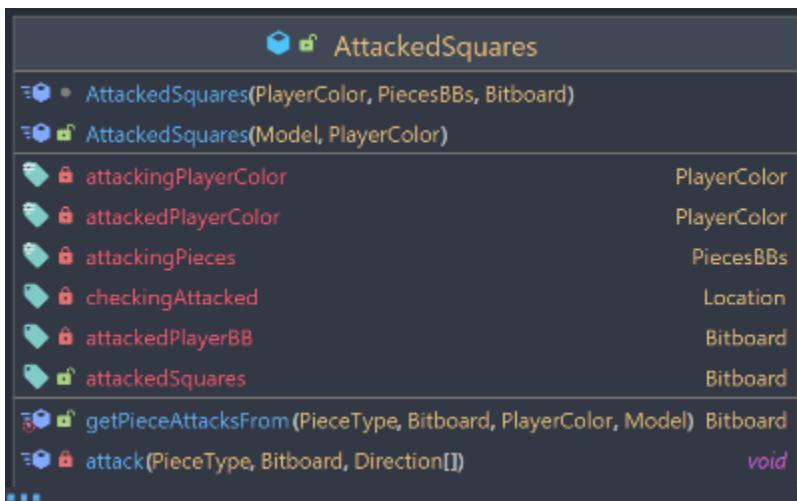
Generates moves from a given position and GenerationSettings

generateMoves public static 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



```
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

getAttackedSquares public static 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`

attack private void **attack**(PieceType pieceType, Bitboard attackingPiecesBB, ver14.SharedClasses.Game.Moves.Direction... 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

isAttacked public static boolean **isAttacked**(Model model, Location loc, PlayerColor attackingPlayerColor)

Is the given loc threatened by the `attackingPlayerColor`.

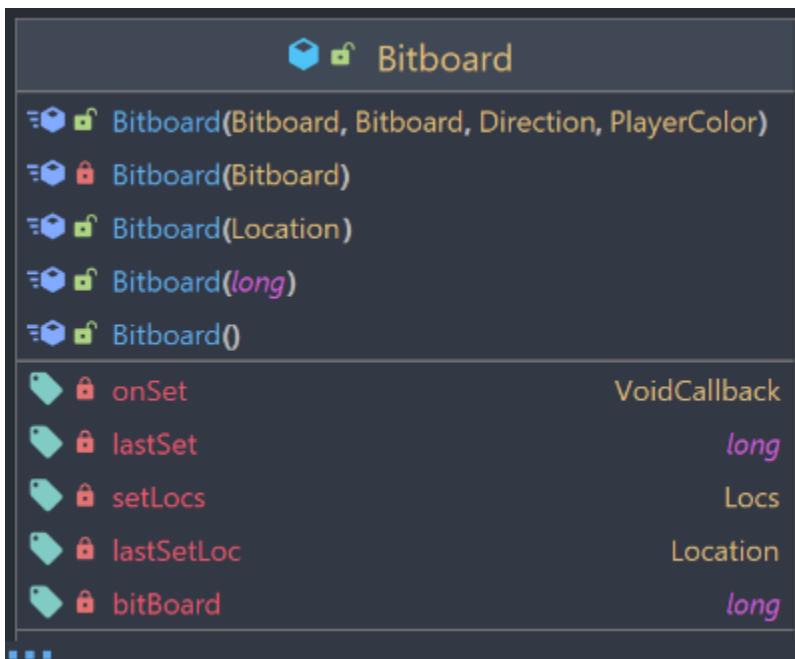
Parameters: model - the model representing the current position loc - the loc `attackingPlayerColor` - the attacking player color **Returns:** true if the loc is attacked by the attacking player, false otherwise.

getPieceAttacksFrom public static 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

**All Implemented Interfaces:** Serializable

```
public class Bitboard
implements Serializable
```

Bitboard - a bitboard representation of the chess board. every bit set is a chess piece on that square.

set public 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.

shiftMe public Bitboard **shiftMe**(PlayerColor playerColor, Direction direction)
performs a bitwise shift on this bitboard instance.

Parameters: playerColor - the player color. used for perspective direction - the direction to shift in

Returns: the changed bitboard. used to chain actions

orEqual public 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

isEmpty public boolean **isEmpty**()

Is this board empty

Returns: is this board empty

exclude public 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

and public 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 בקובץ עצמן



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

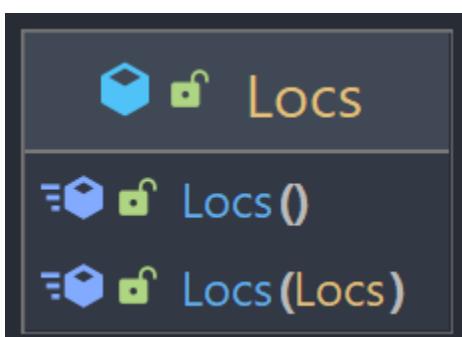
See Also: ...

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

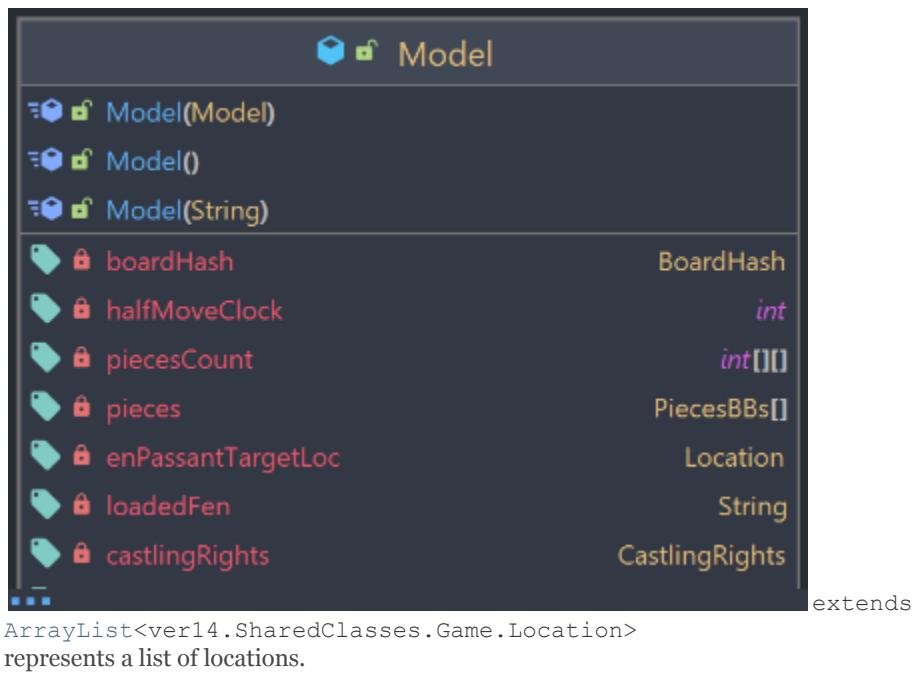
Parameters: model - the model **Returns:** the fen for the given model's position
isValidFen public static boolean **isValidFen**(String fen)
Is valid fen.

Parameters: fen - the fen **Returns:** true if the fen is valid and false otherwise
loadFEN public static void **loadFEN**(String fen, Model model)
Loads a fen.

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



All Implemented Interfaces: Serializable, Cloneable,
 Iterable<ver14.SharedClasses.Game.Location>,
 Collection<ver14.SharedClasses.Game.Location>,
 List<ver14.SharedClasses.Game.Location>, RandomAccess
 public class Locs

**All Implemented Interfaces:** Serializable

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

Model (String)

Instantiates a new Model, and setting it up from the fen.

setup public void setup(String fen)

initializes the model according to a given position's string representation.

Parameters: fen - the position's fen **See Also:** FEN

generateAllMoves public ModelMovesList generateAllMoves()

Generate all moves for the current player to move.

Returns: a list with all the moves the current player can play

isInCheck public boolean isInCheck(PlayerColor playerColor)

Is playerColor in check.

Parameters: playerColor - the player color **Returns:** true if player is in check, false otherwise.

applyMove public void applyMove(Move move)

applies move to the current position.

Parameters: move - the move

undoMove public 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 בקובץ 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	
...	

All Implemented Interfaces: Serializable, Cloneable,
 Iterable<ver14.SharedClasses.Game.Moves.Move>,
 Collection<ver14.SharedClasses.Game.Moves.Move>,
 List<ver14.SharedClasses.Game.Moves.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.
initAnnotation public void **initAnnotation()**
 Initializes moves notation.

את שאר הפעולות ניתן לראות בקוד עצמו

FoundLegalMove	
FoundLegalMove()	

All Implemented Interfaces: Serializable
 public class FoundLegalMove
 extends Throwable
 Found legal move. used for saving time when looking for any legal move.

PiecesBBs	
≡⊕ ☑ PiecesBBs(int)	
✖ ↗ size	int
✖ ↗ prevAll	Bitboard
✖ ↗ bitboards	Bitboard[]
≡⊕ ☑ getAll()	Bitboard
≡⊕ ☑ getBitboards()	Bitboard[]
≡⊕ ☑ getPieceType(Bitboard)	PieceType
≡⊕ ☑ getBB(PieceType)	Bitboard
≡⊕ ☑ toString()	String

```
public class PiecesBBs
```

represents a collection of bitboards of pieces.

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

Parameters: pieceType - the piece type **Returns:** the bb
את שאר הפעולות ניתן לראות בקוד עצמו

Players

Player	
≡⊕ ☑ Player(String)	
✖ ↗ playerColor	PlayerColor
✖ ↗ gameSession	GameSession
✖ ↗ game	Game
✖ ↗ username	String
✖ ↗ partner	Player
✖ ↗ createdGameID	String
≡⊕ ☑ isAi()	boolean
≡⊕ ☑ isSaveWorthy()	boolean
≡⊕ ☑ isGuest()	boolean
...	

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

All Implemented Interfaces: SyncableItem

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	
<code>StockfishPlayer(AiParamers)</code>	
<code>stockfish</code>	Stockfish
<code>initGame()</code>	<code>void</code>
<code>cancelQuestion(Question, String)</code>	<code>void</code>
<code>gameOver(GameStatus)</code>	<code>void</code>
<code>disconnect(String, boolean)</code>	<code>void</code>
<code>waitForMatch()</code>	<code>void</code>
<code>updateByMove(Move)</code>	<code>void</code>
<code>waitTurn()</code>	<code>void</code>
<code>getMove()</code>	<code>Move</code>
<code>...</code>	

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

את שאר הפעולות ניתן לראות בקוד עצמו
```

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

```
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
```

getBestMove `public 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 **Returns:** Best Move in PGN format

getEvalScore `public float getEvalScore(String fen, int waitTime)`
Get the evaluation score of a given board position

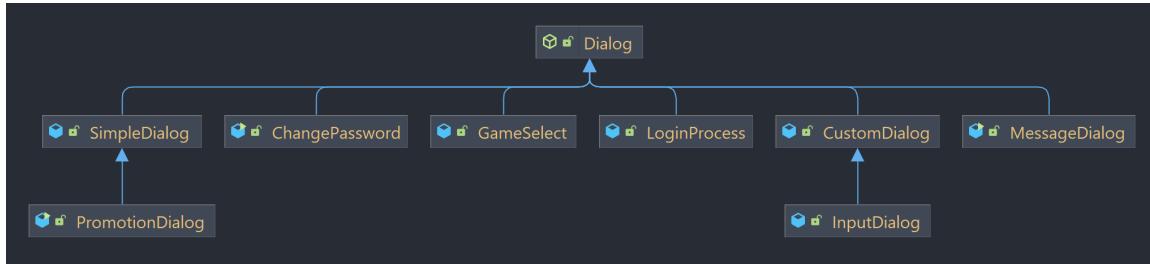
Parameters: fen - Position string **waitTime** - in milliseconds **Returns:** evalScore

את שאר הפעולות ניתן לראות בקוד עצמו

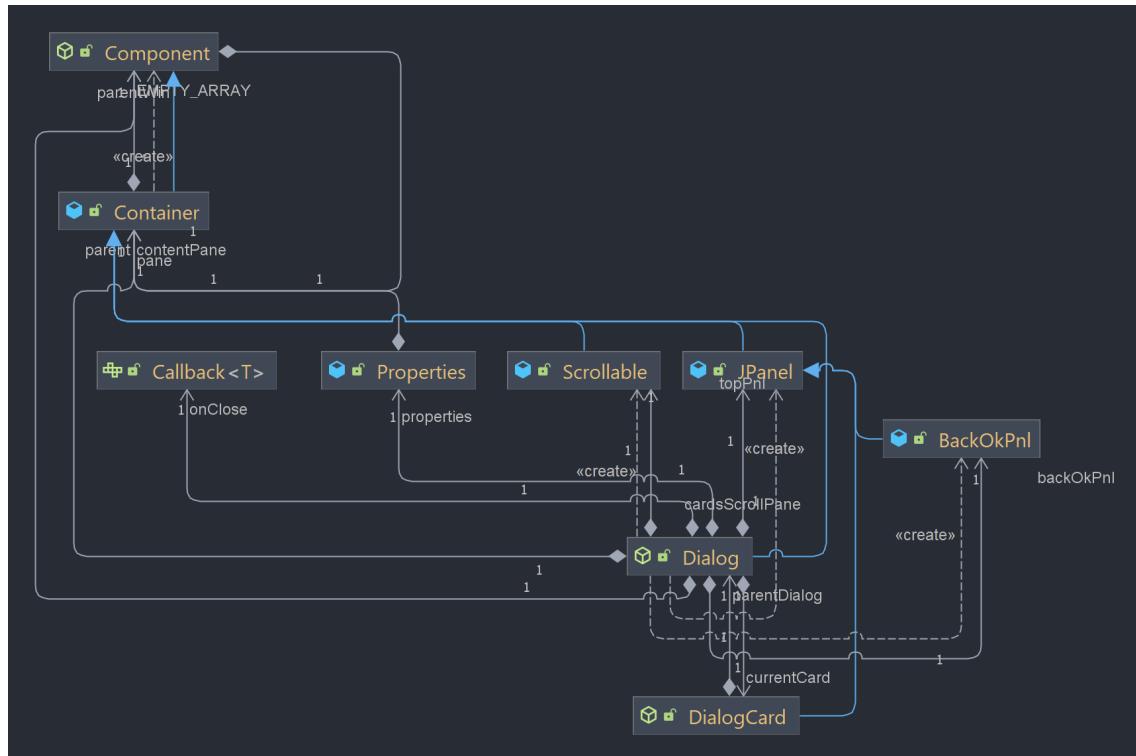
4.4.3 תרשימים UML של המחלקות בצד הלוקוֹת

דיאלוגים:

תצוגה כללית של כל הדיאלוגים:

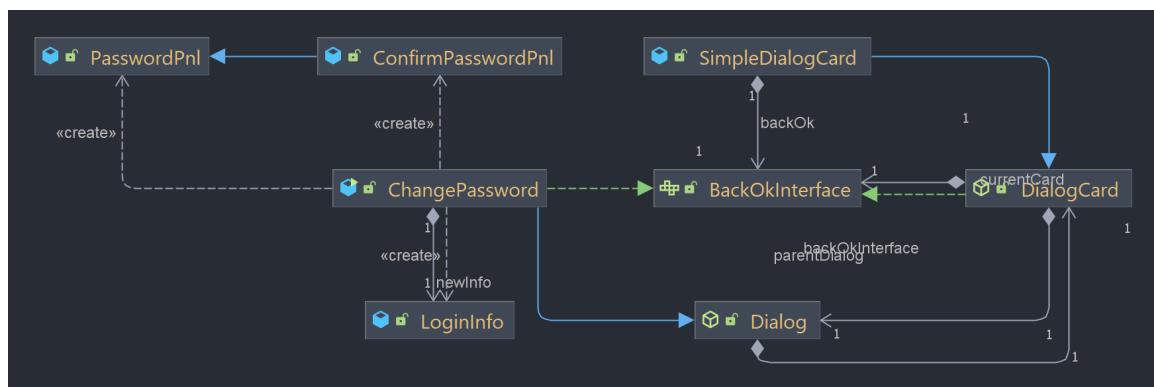


מחליקת דיאלוגים:

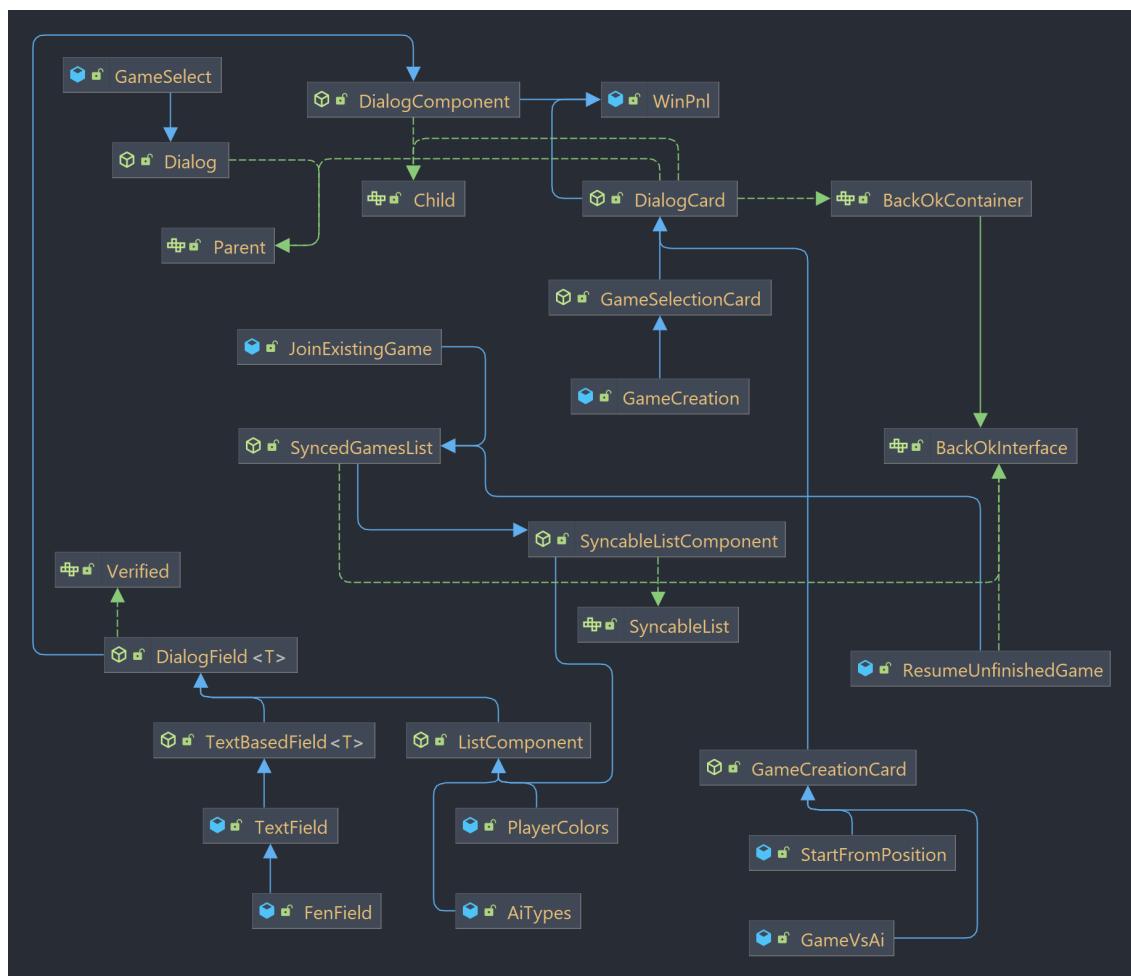


כל אחד מהדיאלוגים:

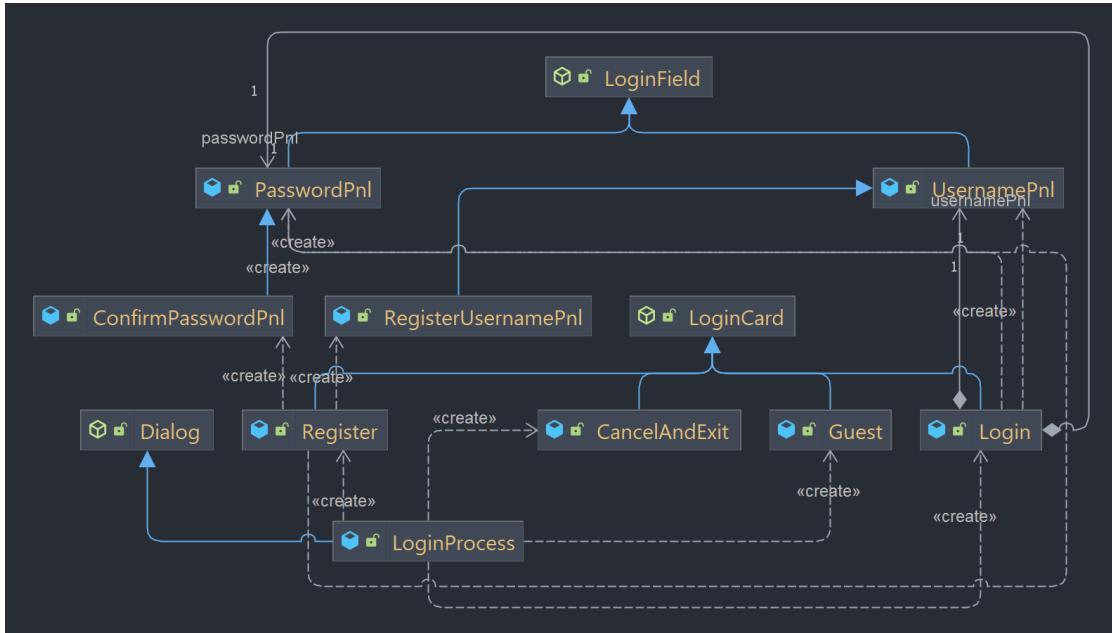
Change Password



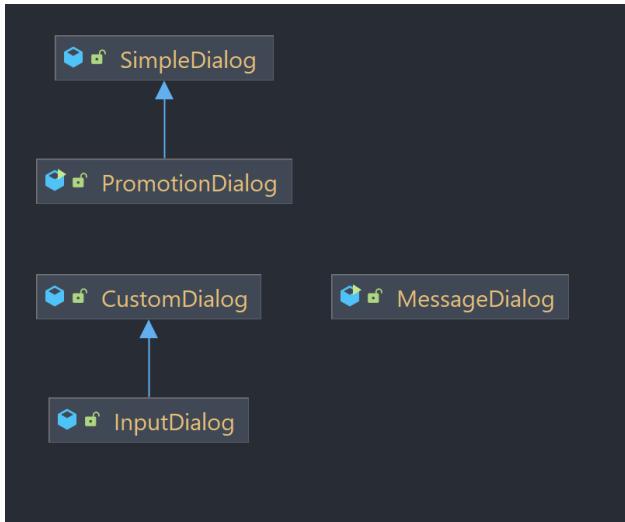
Game Select



Login Process

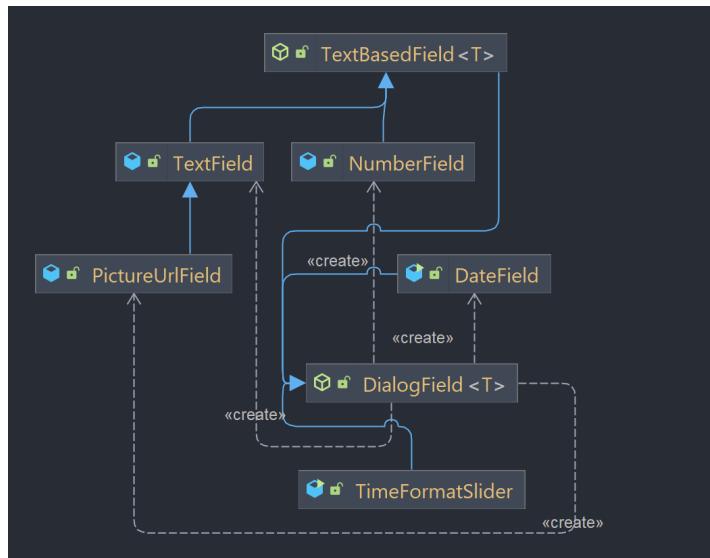


Simple Dialogs

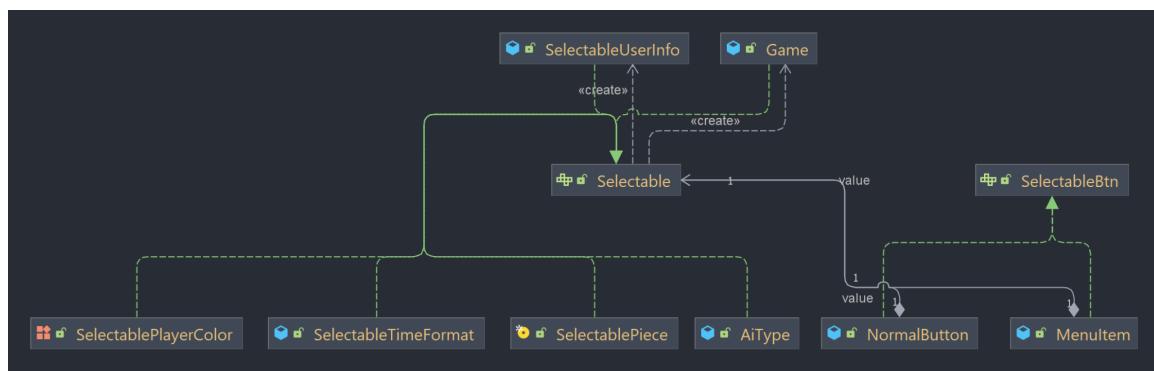


תכונות הדיאלוגים:

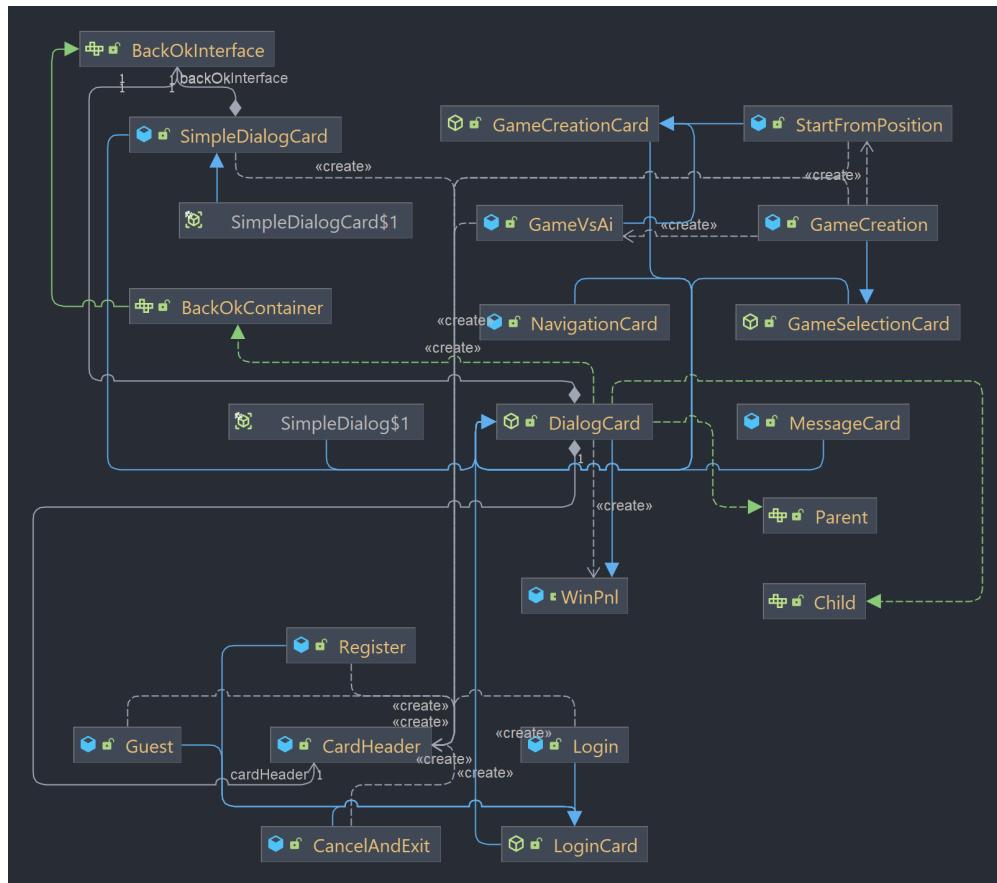
Fields



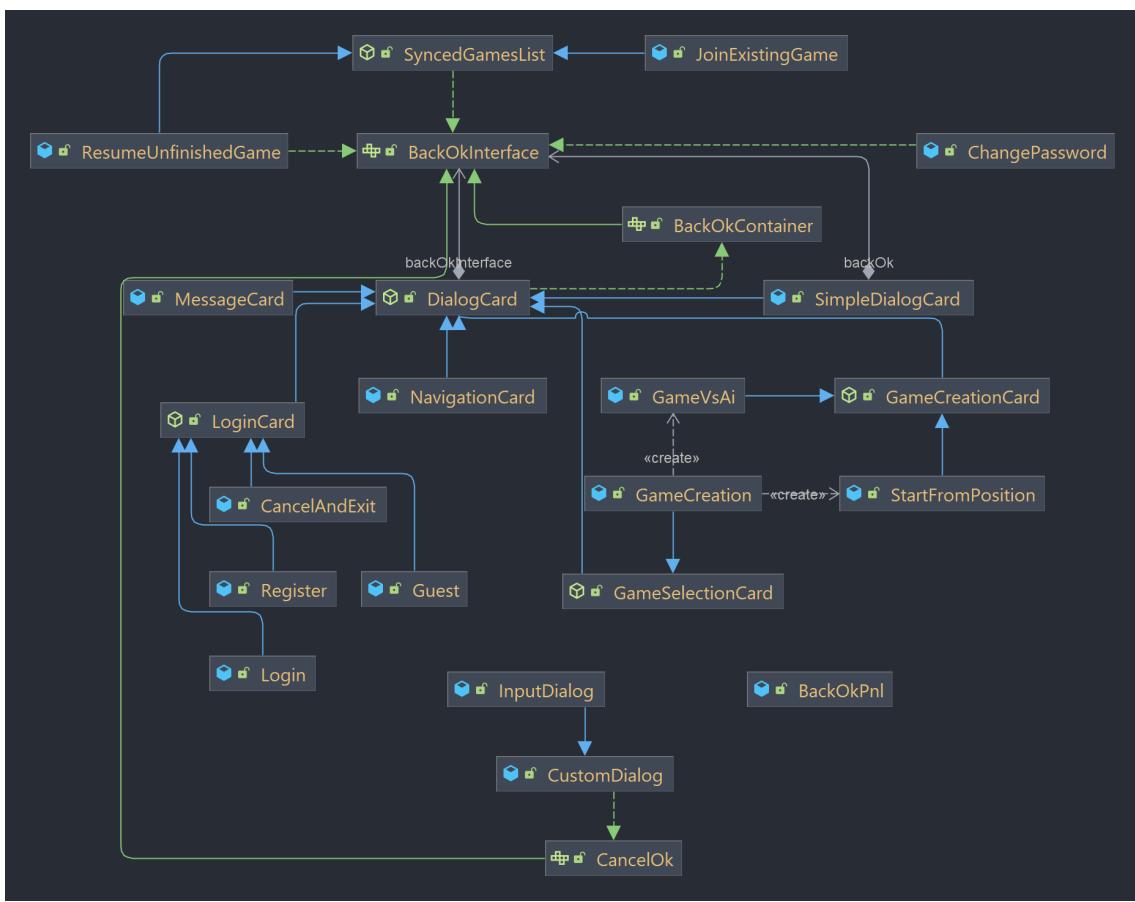
Selectables



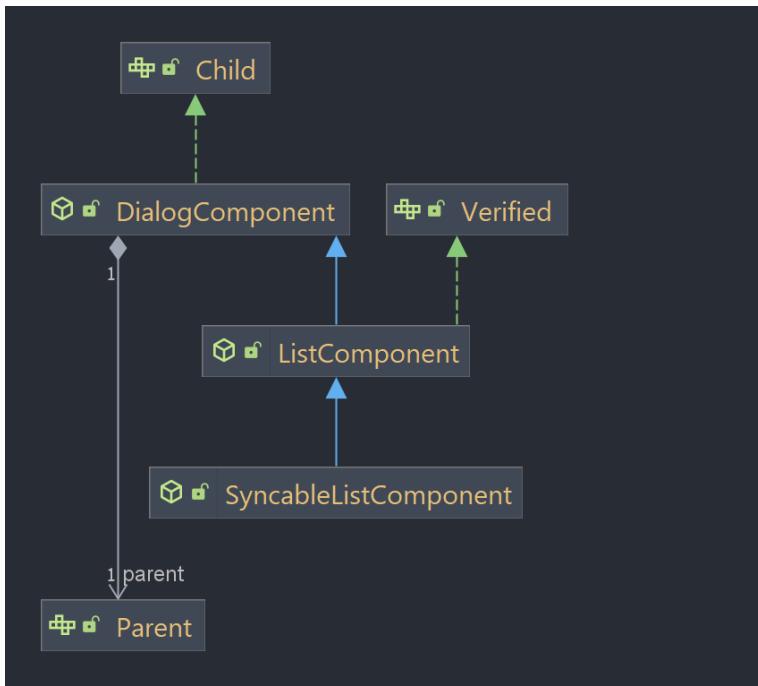
Cards



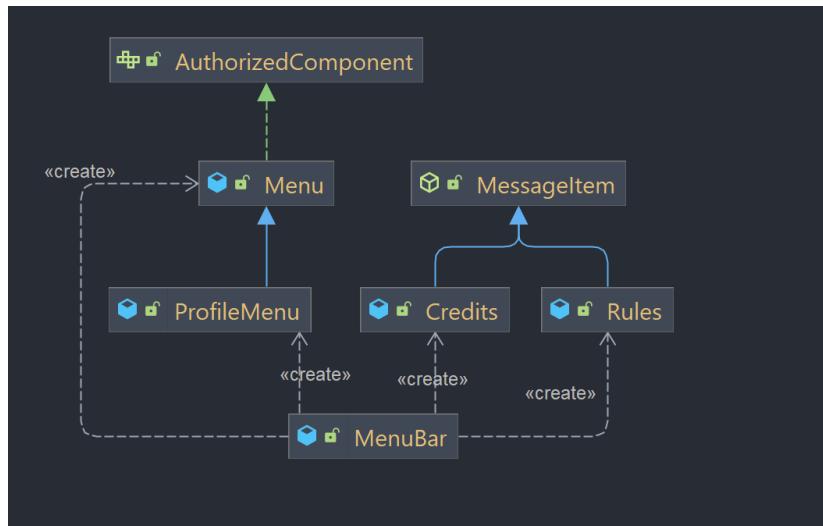
Back Ok



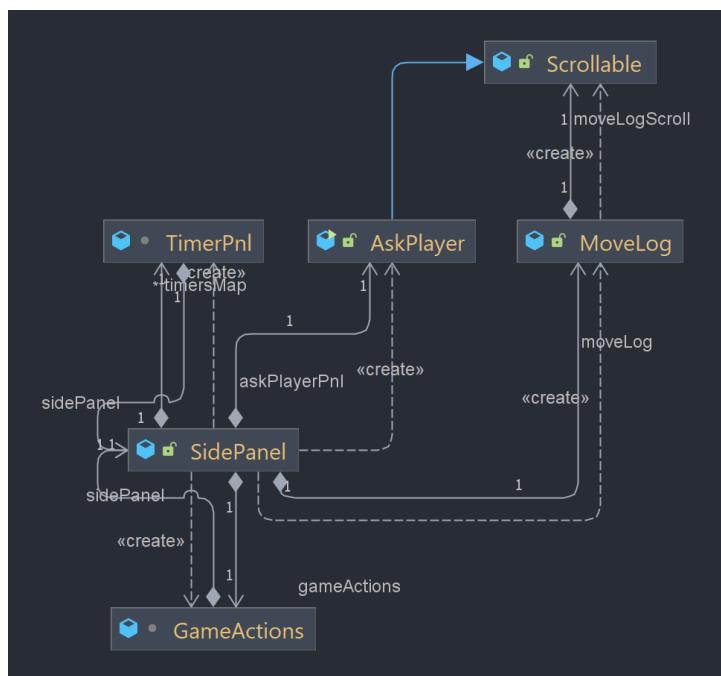
Components



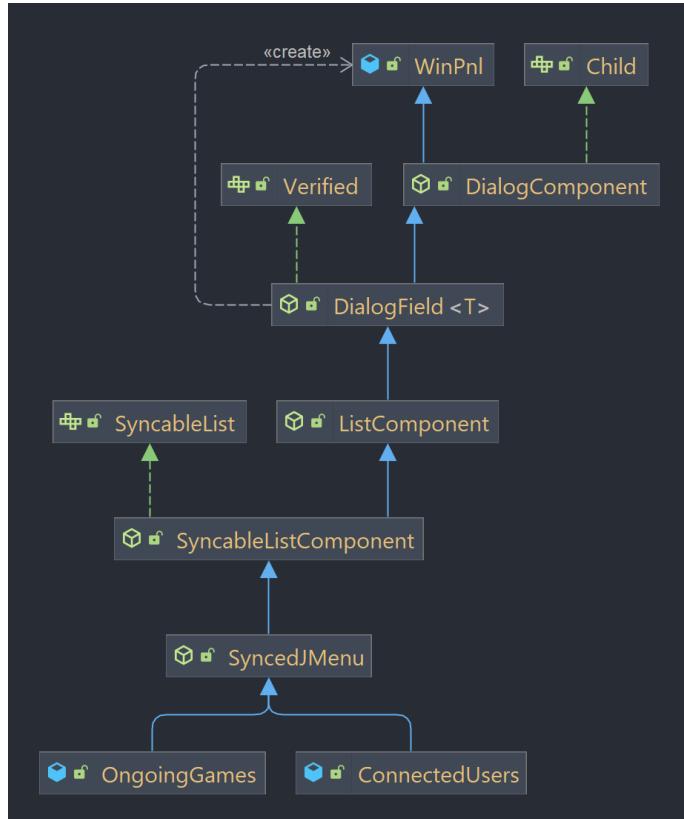
Menu Bar



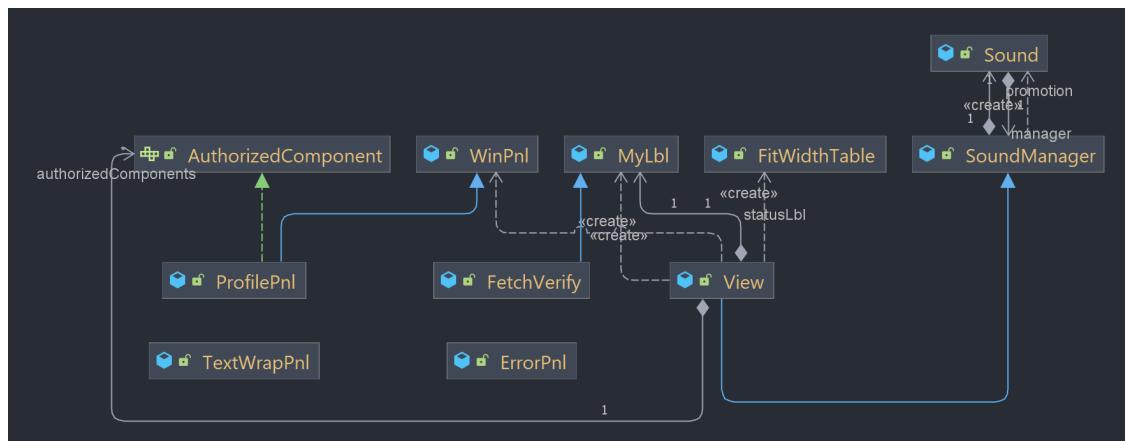
SidePanel



Synced JMenus



Misc



תיעוד המחלקות בצד לקוח

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

All Implemented Interfaces: EnvManager

```
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.

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

boardButtonPressed public void **boardButtonPressed**(ViewLocation clickedLoc)
handle board button pressed.

Parameters: clickedLoc - the button's location

showPromotionDialog public 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

request public void **request**(RequestBuilder builder,
ver14.SharedClasses.Callbacks.Callback<ver14.SharedClasses.DBActions.DBResponse
.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 בקובץ עצמן

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

```
public class ClientMessagesHandler
extends MessagesHandler
```

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

See Also: MessagesHandler

registerSyncableList public 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	
<code>SoundManager()</code>	
<code>tenSeconds</code>	Sound
<code>soundEnabled</code>	boolean
<code>promotion</code>	Sound
<code>selfMove</code>	Sound
<code>gameStart</code>	Sound
<code>gameEnd</code>	Sound
<code>capture</code>	Sound
<code>castle</code>	Sound
<code>check</code>	Sound
<code>...</code>	

```
public class SoundManager
```

represents a Sound manager.

isSoundEnabled public boolean `isSoundEnabled()`

Is sound enabled boolean.

Returns: the boolean

setSoundEnabled public void `setSoundEnabled(boolean soundEnabled)`

Sets sound enabled.

Parameters: soundEnabled - the sound enabled

moved public 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.

playLoop public void **playLoop()**
Play forever (loop).

stop public void **stop()**
Stop playing.

play public 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

All Implemented Interfaces: Iterable<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..
```

View public **View**(Client client)

Instantiates a new View.

Parameters: client - the client

addListToRegister public void **addListToRegister**(SyncableList list)

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

Parameters: list - the list

addAuthorizedComponent public 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

showMessage public void **showMessage**(String message, String title, MessageType messageType)

Show message.

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

authChange public 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

initGame public 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

askQuestion public 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

showDBResponse public void **showDBResponse**(DBResponse response, String respondingTo, String title)

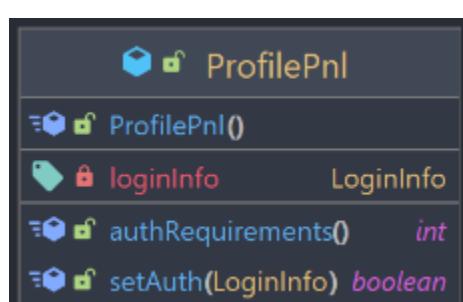
Show a db response from the server.

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

dispose public void **dispose**()

Dispose. closes the main window and all open dialogs

את שאר הפעולות ניתן לראות בקובץ עצמו



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, AuthorizedComponent

public class **ProfilePnl**

extends **WinPnl**

implements **AuthorizedComponent**

represents a Profile panel.

ProfilePnl 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
...	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, CellEditorListener, ListSelectionListener, RowSorterListener, TableColumnModelListener, TableModelListener, Scrollable

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

```
FitWidthTable 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

```
fit public void fit()
calculates the optimal size for the contents of this table
```

את שאר הפעולות ניתן לראות בקוד עצמה

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, SwingConstants

```
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



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
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



```
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

authRequirements int `authRequirements()`
this component's authentication requirements

Returns: the int

את שאר הפעולות ניתן לראות בקוד עצמו

Menu	
<code>Menu(String, int)</code>	
<code>authSettings</code>	<code>int</code>
<code>childrenFont</code>	<code>Font</code>
<code>add(String, VoidCallback)</code>	<code>JMenuItem</code>
<code>authRequirements()</code>	<code>int</code>
<code>setChildrenFont(Font)</code>	<code>void</code>
<code>add(JMenuItem)</code>	<code>JMenuItem</code>

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants, AuthorizedComponent

```
public class Menu
extends JMenu
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 בקוד עצמו

MenuItem	
<code>MenuItem(String)</code>	
<code>MenuItem(String, int)</code>	
<code>authSettings</code>	<code>int</code>
<code>authRequirements()</code>	<code>int</code>

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants, AuthorizedComponent

```
public class MenuItem
extends JMenuItem
implements AuthorizedComponent
Menu item - 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 בקוד עצמו

Board

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

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, SwingConstants

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

is public 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

globalPaint public 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 בקובץ עצמו

@ State	
<code>CHECK</code>	int
<code>PROMOTING</code>	int
<code>MOVING_FROM</code>	int
<code>CAN_MOVE_TO</code>	int
<code>HOVERED</code>	int
<code>SELECTED</code>	int
<code>DRAGGING</code>	int
<code>CAPTURE</code>	int
<code>MOVING_TO</code>	int
<code>CLICKED_ONCE</code>	int
<code>...</code>	

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

ViewLocation
ViewLocation(Location)
viewLocation Location
originalLocation Location
toString() String

```
public class ViewLocation
```

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

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: Serializable,

```
Iterable<ver14.SharedClasses.Game.GameSetup.BoardSetup.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.

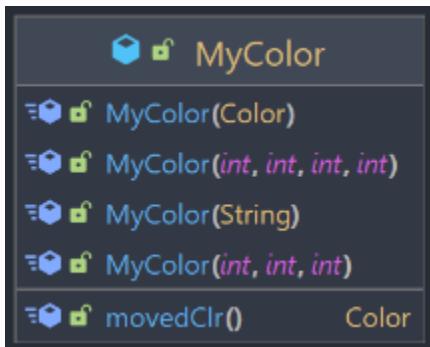
את שאר הפעולות ניתן לראות בקוד עצמו

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

```
public class SavedSquare
```

represents a Saved square on the saved board.

את שאר הפעולות ניתן לראות בקוד עצמו



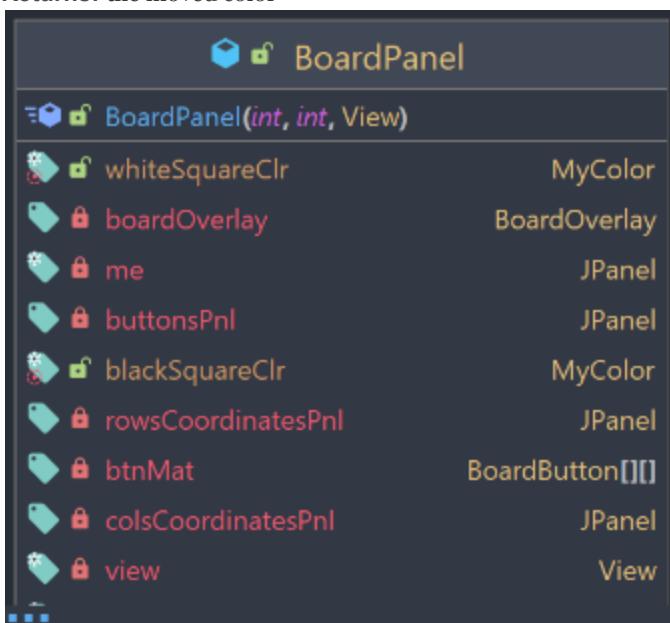
All Implemented Interfaces: Paint, Transparency, Serializable

```
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 .

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

Returns: the moved color



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable,

Iterable<ver14.view.Board.BoardButton[]>, Accessible

```
public class BoardPanel
```

extends JPanel

implements Iterable<ver14.view.Board.BoardButton[]>

represents the Board panel. holding all the BoardButtons.

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

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

Parameters: lock - the lock

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

Parameters: savedBoard - the saved board

getBtnMat public ver14.view.Board.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<?>

All Implemented Interfaces: Serializable

```
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.

currentColor public Color **currentColor()**

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

Returns: the color

centerPoint private Point **centerPoint (Point point)**

Center point in the middle of a button.

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

getBtn private BoardButton **getBtn (Point point)**

Gets a button that the given point is inside.

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

getLoc public Location **getLoc (Point point)**

converts a Point (x,y) to a Location

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

את שאר הפעולות ניתן לראות בקוד עצמו

• Arrow	
Arrow(Point, Point, Color)	
• barb	<i>int</i>
• start	Point
• phi	<i>double</i>
• end	Point
• clr	Color
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.

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

Sets clr.

Parameters: clr - the clr

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

Draws this arrow.

Parameters: g2 - the g 2

את שאר הפעולות ניתן לראות בקובץ עצמה

Dialog

Dialog	
Dialog(DialogProperties)	
• onCloseCallbacks	ArrayList<VoidCallback>
• backOkPnl	BackOkPnl
• onClose	Callback<Dialog>
• MAX_DIALOG_SIZE	Size
• cardsPnl	JPanel
• errorPnl	ErrorPnl
• DEFAULT_DIALOG_SIZE	Size
• cardStack	Stack<DialogCard>
• bottomPnl	JPanel
...	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible,

RootPaneContainer, WindowConstants, Parent

```
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.

done public void **done()**

Description copied from interface: Parent

the dialog is finished and should close.

Specified by: done in interface Parent

back public void **back()**

Description copied from interface: Parent

the Back button has been clicked and the dialog should probably go to the previous card.

Specified by: back in interface Parent

scrollToTop public void **scrollToTop()**

Description copied from interface: Parent

Scroll to the top of the dialog.

Specified by: scrollToTop in interface Parent

verifyCurrentCard private void **verifyCurrentCard()**

Verify the current card.

See Also: Verified

notifyClosed protected void **notifyClosed()**

Notify all close listeners.

start public void **start**(ver14.SharedClasses.Callback<Dialog> onClose)
Start showing the dialog.

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

navigationCardSetup protected NavigationCard **navigationCardSetup**(Size navCardSize,
ver14.view.Dialog.Cards.DialogCard... 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

cardsSetup protected void **cardsSetup**(DialogCard startingCard,
ver14.view.Dialog.Cards.DialogCard... 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 בקובץ עצמו

DialogProperties	
<code>DialogProperties(DialogDetails)</code>	
<code>DialogProperties(LoginInfo, AppSocket, MyJFrame, DialogDetail)</code>	
<code>loginInfo</code>	LoginInfo
<code>dialogDetails</code>	DialogDetails
<code>contentPane</code>	Container
<code>argConfig</code>	Config<?>
<code>socketToServer</code>	AppSocket
<code>parentWin</code>	MyJFrame
<code>toString()</code>	String
<code>setContentPane(Container)</code>	void
<code>...</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)
extends Record
```

represents a dynamic object with a few optional Dialog Details.

את שאר הפעולות ניתן לראות בקוד 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.

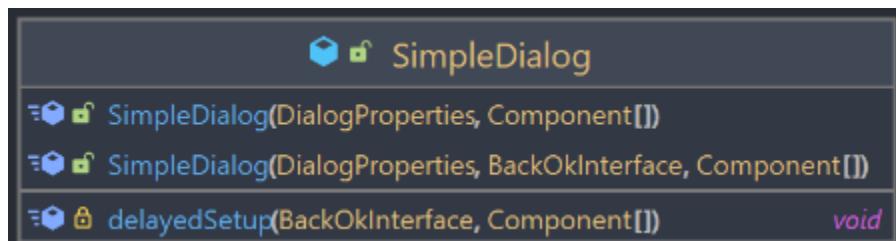
`obj() obj()`

get the value of this object.

Returns: the value of this object

errorDetails `String errorDetails()`
get the Error details.

Returns: the error details



All Implemented Interfaces: `ImageObserver, MenuContainer, Serializable, Accessible,`

`RootPaneContainer, WindowConstants, Parent`

`public class SimpleDialog`

`extends Dialog`

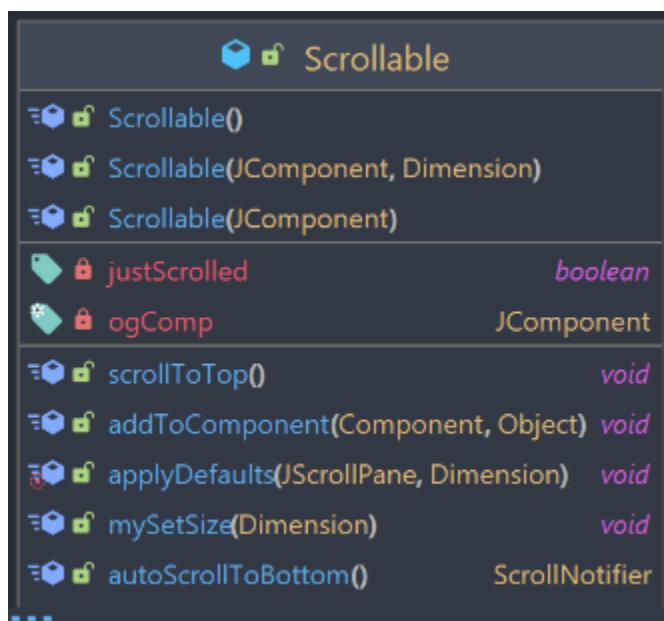
represents a Simple dialog, that can be created with simple components.

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

Instantiates a new Simple dialog.

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

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



All Implemented Interfaces: `ImageObserver, MenuContainer, Serializable, Accessible, ScrollPaneConstants`

`public class Scrollable`

`extends JScrollPane`

my implementation of a Scrollable component.

`scrollToTop public void scrollToTop()`
Scroll to top.

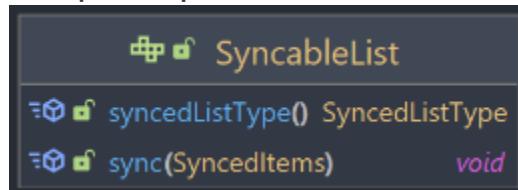
`scrollTo private void scrollTo(int position)`
Scroll to.

Parameters: `position` - the position

`scrollToBottom public void scrollToBottom()`

Scroll to bottom.

את שאר הפעולות ניתן לראות בקוד עצמו



`public interface SyncableList`
represents a Syncable list.

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

Returns: the synced list type

sync `void sync(SyncedItems items)`
Sync this list with items.

Parameters: items - the items



All Implemented Interfaces: `ImageObserver`, `MenuContainer`, `Serializable`, `Accessible`

`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 `GridBagLayout`

setHeader `public void setHeader(Header header)`
Sets header.

Parameters: header - the header

add `public Component add(Component comp)`

Overrides: `add` in class `Container`

`public void add(Component comp, GridBagConstraints gbc)`
Add a component with custom settings.

Parameters: comp - the comp `gbc` - the gbc
את שאר הפעולות ניתן לראות בקוד עצמו

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

BackOkInterface	
<code>noInterface</code>	<code>BackOkInterface</code>
<code>getOkText()</code>	<code>String</code>
<code>onBack()</code>	<code>void</code>
<code>createSimpleInterface(VoidCallback)</code>	<code>BackOkInterface</code>
<code>onOk()</code>	<code>void</code>
<code>getBackText()</code>	<code>String</code>

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.

`onBack` void `onBack()`
called on click of the back button.

`onOk` void `onOk()`
called on click of the ok button.

CancelOk	
<code>onCancel()</code>	<code>void</code>
<code>getBackText()</code>	<code>String</code>

All Superinterfaces: `BackOkInterface`

public interface `CancelOk`
extends `BackOkInterface`
represents a cancel ok options navigation.

`getBackText` default `String getBackText()`

Description copied from interface: `BackOkInterface`

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.

BackOkContainer	
onBack()	void
actualInterface()	BackOkInterface
getBackText()	String
getOkText()	String
onOk()	void

All Superinterfaces: BackOkInterface

```
public interface BackOkContainer
extends BackOkInterface
represents a container for a Back ok interface.
```

actualInterface BackOkInterface **actualInterface()**
Actual interface back ok interface.

Returns: the back ok interface
את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
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 בקוד עצמו
Cards

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
 public abstract class **DialogCard**
 extends WinPnl
 implements ver14.view.Dialog.Components.Child,
 ver14.view.Dialog.Components.Parent, AncestorListener, BackOkContainer
 represents a Dialog card. a panel for displaying and managing DialogComponents.

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

checkVerifiedComponents public String `checkVerifiedComponents()`
 Check verified components.

Returns: Error message if any not verified, null otherwise
displayed public void `displayed()`
 called when this card gets Displayed.

askServer public void `askServer(Message msg, MessageCallback onRes)`
Description copied from interface: Parent

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

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

Parameters: component - the component
 את שאר הפעולות ניתן לראות בקובץ עצמן

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent

```
public class MessageCard
extends DialogCard
```

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

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: Serializable, Comparable<MessageType>, Constable

```
public enum MessageType
extends Enum<MessageType>
```

Message type.

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
public class CardHeader
extends Header
```

represents a header for a dialog card.

את שאר הפעולות ניתן לראות בקוד עצמו

NavigationCard	
<code>NavigationCard(CardHeader, Dialog, DialogCard[])</code>	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent

```
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,
ver14.view.Dialog.Cards.DialogCard... linkTo)
Instantiates a new Navigation card.
```

SimpleDialogCard	
<code>SimpleDialogCard(CardHeader, Dialog, BackOkInterface)</code>	
<code>backOk</code>	BackOkInterface
<code>onBack()</code>	void
<code>create(DialogComponent, Dialog)</code>	SimpleDialogCard
<code>create(CardHeader, Dialog, BackOkInterface, DialogComponent[])</code>	SimpleDialogCard
<code>onOk()</code>	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent

```
public class SimpleDialogCard
extends DialogCard
Simple dialog card - used for simple creation of a dialog card.
```

```
create public static SimpleDialogCard create(CardHeader header, Dialog parent,
BackOkInterface backOk, ver14.view.Dialog.Components.DialogComponent...
components)
Create simple dialog card.
```

Parameters: header - the header **parent** - the parent **backOk** - the back ok **components** - the components
Returns: the simple dialog card

את שאר הפעולות ניתן לראות בקובץ עצמוני [Components](#)

DialogComponent	
<code>DialogComponent(int, Header, Parent)</code>	
<code>DialogComponent(Header, Parent)</code>	
<code>parent</code>	Parent
<code>parent()</code>	Parent
<code>onUpdate()</code>	void
<code>setParent(Parent)</code>	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child

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

את שאר הפעולות ניתן לראות בקוד עצמו

Parent	
<code>currentCard()</code>	DialogCard
<code>tryCancel()</code>	boolean
<code>scrollToTop()</code>	void
<code>back()</code>	void
<code>dialogWideErr(String)</code>	void
<code>onUpdate()</code>	void
<code>registerSyncedList(SyncableList)</code>	void
<code>tryOk(boolean)</code>	boolean
<code>keyAdapter()</code>	MyAdapter
<code>askServer(Message, MessageCallback)</code>	void
<code>...</code>	

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

registerSyncedList void **registerSyncedList**(SyncableList list)

Register a synced list with the server.

Parameters: list - the list to register

askServer 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

onUpdate void **onUpdate()**

On update to the hierarchy.

scrollToTop default void **scrollToTop()**

Scroll to the top of the dialog.

currentCard DialogCard **currentCard()**

get the currently displayed dialog card

Returns: the currently displayed dialog card

את שאר הפעולות ניתן לראות בקוד עצמו

Child	
<code>parent()</code>	Parent

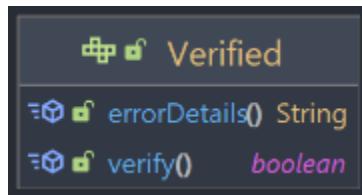
public interface Child

Child - represents an object with a parent.

parent Parent **parent()**

Parent.

Returns: the parent



```
public interface Verified
```

Verified - represents a verified component that according to the situation, might not verify.

verify boolean **verify()**

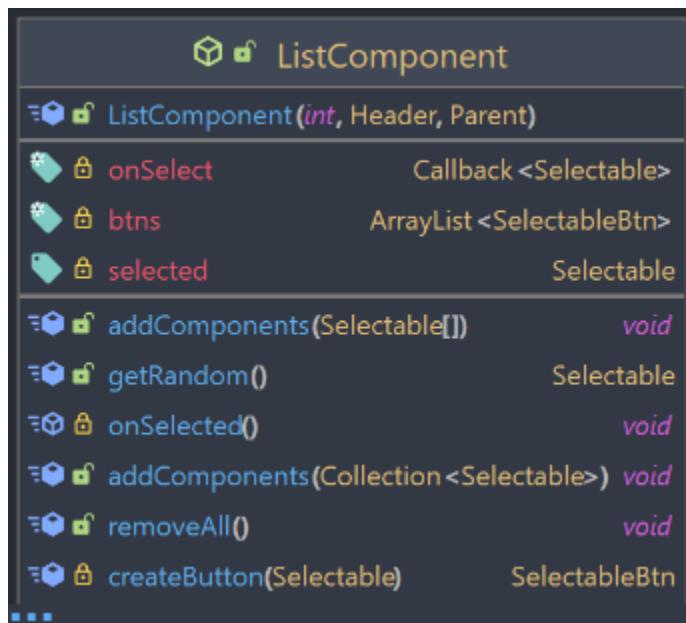
Verify check the current conditions, and try to verify.

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

errorDetails String **errorDetails()**

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

Returns: the error details.



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public abstract class ListComponent
extends
ver14.view.Dialog.DialogFields.DialogField<ver14.view.Dialog.Selectables.Selectable>
```

List component - represents a list component.

addComponents public void

```
addComponents(ver14.view.Dialog.Selectables.Selectable... components)
```

Add components.

Parameters: components - the components

addComponent public void **addComponent**(Selectable item)

Add component.

Parameters: item - the item

createButton protected 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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified, SyncableList

```
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` public SyncedListType `syncedListType()`
Description copied from interface: SyncableList

get this synced list's type.

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

Description copied from interface: SyncableList

Sync this list with items.

Specified by: `sync` in interface SyncableList **Parameters:** items - the items
`canUseIcon` protected 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 בקובץ עצמו

[Dialog Fields](#)

DialogField <T>	
=	DialogField(int, Header, Parent)
=	DialogField(Header, Parent)
*	notEqualsTo ArrayList<CanError<T>>
*	errMsg 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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
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**(ver14.SharedClasses.DBActions.Arg.Config<T> config)
Sets the configuration of this field.

Parameters: config - the config

createValBtn private ver14.SharedClasses.UI.Buttons.ValueBtn<T>
createValBtn(ver14.SharedClasses.DBActions.Arg.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

verify public boolean **verify()**

Description copied from interface: Verified

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

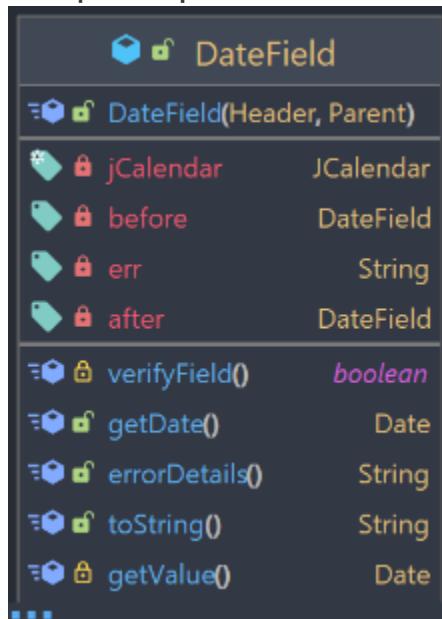
getResult public T **getResult()**
Gets the result of this field.

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

createCard public DialogCard **createCard()**
Create card out of this field.

Returns: the dialog card

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

public class **DateField**
extends ver14.view.Dialog.DialogFields.DialogField<Date>
a Date field.

getValue protected Date **getValue()**

Description copied from class: **DialogField**

Gets the current value of this field.

Specified by: `getValue` in class `ver14.view.Dialog.DialogFields.DialogField<Date>`

Returns: the value

verifyField protected boolean **verifyField()**

Description copied from class: **DialogField**

Verify field.

Specified by: `verifyField` in class `ver14.view.Dialog.DialogFields.DialogField<Date>`

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

setValue public void **setValue**(Date value)

Description copied from class: **DialogField**

Sets value.

Specified by: `setValue` in class `ver14.view.Dialog.DialogFields.DialogField<Date>`

Parameters: value - the value

setBefore public void **setBefore**(DateField before)

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

Parameters: before - the before

setAfter public void **setAfter**(DateField after)

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

Parameters: after - the after

errorDetails public String **errorDetails()**

Description copied from interface: **Verified**

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

Returns: the error details.
את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public class TimeFormatSlider
extends
ver14.view.Dialog.DialogFields.DialogField<ver14.SharedClasses.Game.GameSetup.TimeFormat>
a Time format slider.
```

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

GetValue protected TimeFormat **GetValue()**
Description copied from class: DialogField

Gets value.

Specified by: GetValue in class
ver14.view.Dialog.DialogFields.DialogField<ver14.SharedClasses.Game.GameSetup.TimeFormat> **Returns:** the value

SetValue public void **SetValue**(TimeFormat value)
Description copied from class: DialogField

Sets value.

Specified by: SetValue in class
ver14.view.Dialog.DialogFields.DialogField<ver14.SharedClasses.Game.GameSetup.TimeFormat> **Parameters:** value - the value
את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public abstract class TextBasedField<T>
extends ver14.view.Dialog.DialogFields.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.

getTextFieldText protected String getTextFieldText()
Gets text field text.

Returns: the text field text

verifyRegEx protected boolean verifyRegEx()
Verify this field by its regex limitations.

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

את שאר הפעולות ניתן לראות בקובץ עצמו

TextField		
=	TextField(Header, Parent, RegEx)	
=	TextField(String, Parent, RegEx)	
?	getValue()	String
?	setValue(String)	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public class TextField
extends ver14.view.Dialog.DialogFields.TextBasedFields.TextBasedField<String>
```

See Also: Serialized Form

getValue protected String getValue()

Description copied from class: DialogField

Gets value.

Specified by: getValue in class ver14.view.Dialog.DialogFields.DialogField<String>

Returns: the value

setValue public void **setValue**(String value)
Description copied from class: [DialogField](#)

Sets value.

Specified by: `setValue` in class `ver14.view.Dialog.DialogFields.DialogField<String>`

Parameters: value - the value

[Dialogs](#)

Header	
Header (String, boolean)	
Header (ImageIcon, boolean)	
Header (String)	
Header (String, ImageIcon, boolean)	
maximumSize	Size
center	boolean
insets	Insets
lbl	MyLbl
icon	ImageIcon
text	String
...	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
public class Header
extends JPanel
```

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

את שאר הפעולות ניתן לראות בקוד עצמו

[Change Password](#)

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible,

RootPaneContainer, WindowConstants, BackOkInterface, Parent

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

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

getPassword public String getPassword()
Gets password.

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

GameSelect	
GameSelect(DialogProperties)	
gameSettings	GameSettings
getGameSettings()	GameSettings

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants, Parent
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.

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

Returns: the game settings
Cards

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
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(Parent, GameSettings)	
createCard()	DialogCard

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, BackOkInterface, Child, Verified, SyncableList
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.

createCard public DialogCard **createCard()**
Description copied from class: DialogField

Create card out of this field.

Overrides: `createCard` in class
`ver14.view.Dialog.DialogFields.DialogField<ver14.view.Dialog.Selectables.Selectable>` **Returns:** the dialog card

	ResumeUnfinishedGame
	ResumeUnfinishedGame(Parent, GameSettings)
	getOkText() String

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, BackOkInterface, Child, Verified, SyncableList

```
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 עצמן

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent

```
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 עצמן

		GameVsAi
		GameVsAi(Dialog, GameSettings)
		aiParameters
		setEnabledState(boolean) void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
 public class **GameVsAi**
 extends GameCreationCard
 Game vs ai card.

את שאר הפעולות ניתן לראות בקוד עצמו

		GameCreation
		GameCreation(Dialog, GameSettings)

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
 public class **GameCreation**
 extends GameSelectionCard
 Game creation card.

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

		StartFromPosition
		StartFromPosition(Dialog, GameSettings)
		iconLbl JLabel
		getPreferredSize() Dimension
		setEnabledState(boolean) void
		onUpdate() void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
 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[]	
...		

All Implemented Interfaces: Serializable, Comparable<Position>, Constable

public enum Position

extends Enum<Position>

Position - represents pre-made default positions.

את שאר הפעולות ניתן לראות בקוד עצמו

Components

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

public class FenField

```
extends TextField
a Fen field. A FEN is a string representation of a chess position.

setFen public void setFen(String fen)
Sets fen.

Parameters: fen - the fen
את שאר הפעולות ניתן לראות בקובץ עצמו
```

  PlayerColors
  PlayerColors(Dialog, GameSettings)
  gameSettings GameSettings
  onSelected() void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public class PlayerColors
extends ListComponent

Player colors.
```

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

את שאר הפעולות ניתן לראות בקובץ עצמו

  SyncedGamesList
  SyncedGamesList(Header, SyncedListType, Parent, GameSettings,
  gameSettings GameSettings
  gameType GameType
  onBack() void
  onSelected() void
  onOk() void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, BackOkInterface, Child, Verified, SyncableList

```
public abstract class SyncedGamesList
extends SyncableListComponent
implements BackOkInterface

represents a Synchronized list of games.
```

את שאר הפעולות ניתן לראות בקובץ עצמו
Login Process

  LoginProcess
  LoginProcess(DialogProperties)
  loginInfo LoginInfo
  getLoginInfo() LoginInfo

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants, Parent

```
public class LoginProcess
extends Dialog

represents the dialog for the Login process.
```

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

Returns: the chosen login info
Cards

LoginCard	
LoginCard(CardHeader, Dialog, LoginInfo, LoginType)	
loginInfo	LoginInfo
loginType	LoginType
onOk()	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
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	
CancelAndExit(Dialog, LoginInfo)	
navToMe()	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
public class **CancelAndExit**
extends **LoginCard**
Cancel and exit card.

navToMe public void **navToMe()**
Description copied from class: **DialogCard**

Navigate to this dialog card.

Overrides: navToMe in class **DialogCard**

Guest	
Guest(Dialog, LoginInfo)	
navToMe()	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
public class **Guest**
extends **LoginCard**
Guest login.

navToMe public void **navToMe()**
Description copied from class: **DialogCard**

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
 public class **Login**
 extends LoginCard
 represents the normal Login card. enables a login with a username and a password.

את שאר הפעולות ניתן לראות בקוד עצמו

Register	
Register(LoginProcess, LoginInfo)	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, EventListener, Accessible, AncestorListener, BackOkContainer, BackOkInterface, Child, Parent
 public class **Register**
 extends LoginCard
 represents a Register card.

Components

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified
 public abstract class **LoginField**
 extends TextField
 a Login field.

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified
 public class **UsernamePnl**
 extends LoginField
 represents a Username field.

את שאר הפעולות ניתן לראות בקוד 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
...	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public class RegisterUsernamePnl
extends UsernamePnl
```

Register username field. a field that after verifying the regex requirements for usernames, checks if the username is available.

verifyField public boolean **verifyField()**
Description copied from class: [DialogField](#)

Verify field.

Overrides: [verifyField](#) in class [ver4.view.Dialog.DialogFields.TextBasedFields.TextBasedField<String>](#)
Returns: true if the field has verified successfully, false otherwise.

checkAvailable private boolean **checkAvailable()**
Check if the current username is available on the server.

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

setSuggestions private void **setSuggestions**(ArrayList<String> suggestions)
Sets username suggestions.

Parameters: suggestions - the suggestions

fetch private void **fetch**(String un)
Fetch result from the server.

Parameters: un - the un

את שאר הפעולות ניתן לראות בקובץ עצמו

PasswordPnl	
PasswordPnl (<i>boolean</i> , LoginInfo, Parent)	
PasswordPnl (<i>String</i> , <i>boolean</i> , LoginInfo, Parent)	
showPasswordBtn	MyJButton
iconRatio	<i>double</i>
setForeground(<i>Color</i>)	<i>void</i>
setBtnIcon(<i>boolean</i>)	<i>void</i>
getPasswordField()	JPasswordField
onUpdate()	<i>void</i>
getPassword()	<i>String</i>
showText()	<i>void</i>
...	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public class PasswordPnl
extends LoginField
represents a Password field.
```

showText private void **showText()**
Show text.

hideText private void **hideText()**
Hide text.

getPassword public *String* **getPassword()**
Gets password.

Returns: the password
את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

```
public class ConfirmPasswordPnl
```

```
extends PasswordPnl
```

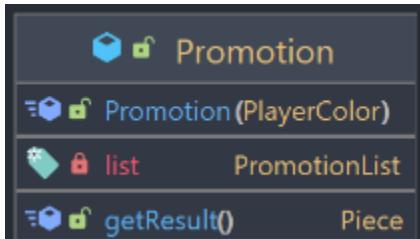
Confirm password field. a field for repeating a password for confirmation.

isMatching public *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 בקובץ עצמו

Promotion



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants, Parent

public class **Promotion**
extends SimpleDialog

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

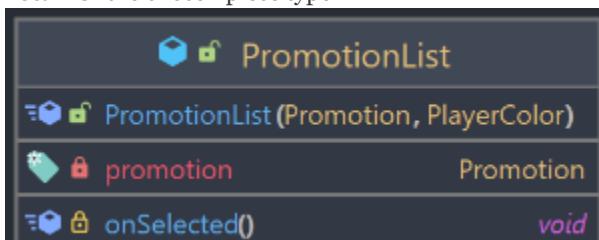
Promotion(PlayerColor playerColor)

Instantiates a new Promotion dialog.

getResult() public Piece **getResult()**

Gets the dialog's result.

Returns: the chosen piece type



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants, BackOkInterface, CancelOk, Parent

```
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, ver14.SharedClasses.DBActions.Arg.Arg... args)`

Instantiates a new Custom dialog with fields to satisfy the given args.

`setup protected void setup(ver14.view.Dialog.DialogFields.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	
<code>InputDialog(DialogProperties)</code>	
<code>InputDialog(DialogProperties, ArgType)</code>	
<code>getInput()</code>	<code>String</code>
<code>getPreferredSize()</code>	<code>Dimension</code>

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants, BackOkInterface, CancelOk, Parent

```
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.

`getInput public String getInput()`

Gets input.

Returns: the input

את שאר הפעולות ניתן לראות בקובץ עצמו

MessageDialog	
=⊕ ☑ MessageDialog(DialogProperties, String, String, MessageType)	
⊕ 🔒 messageType	MessageType
=⊕ 🔒 onXClick()	void
=⊕ ☑ getMessageType()	MessageType

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible,

RootPaneContainer, WindowConstants, Parent

public class **MessageDialog**

extends Dialog

represents a Message dialog, used for displaying messages.

getMessageType public MessageType **getMessageType()**

Gets message type.

Returns: the message type

את שאר הפעולות ניתן לראות בקובץ עצמו

Selectables

Selectable	
=⊕ ☑ getText()	String
=⊕ ☑ getIcon()	ImageIcon
=⊕ ☑ createSelectables(SyncedItems<?>, boolean)	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**(ver14.SharedClasses.Sync.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**getIcon** ImageIcon **getIcon()**

Gets the icon of this selectable.

Returns: null to show no icon, or the icon**getText** String **getText()**

Gets the text of this selectable.

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

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

Record Components: aiType - The Ai type.

All Implemented Interfaces: Selectable

```
public record SelectableAiType(AiType aiType)
    extends Record
    implements Selectable
a selectable Ai type.
```

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: SyncableItem, Selectable

```
public class SelectableGame
    implements ver14.view.Dialog.Selectables.Selectable, SyncableItem
a selectable Game.
```

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: Serializable, Comparable<SelectablePlayerColor>, Constable, Selectable

```
public enum SelectablePlayerColor
extends Enum<SelectablePlayerColor>
implements Selectable
represents a Selectable player color.
```

את שאר הפעולות ניתן לראות בקוד עצמו

SelectableUserInfo	
SelectableUserInfo(UserInfo)	
userInfo	UserInfo
ID()	String
userInfo()	UserInfo
getIcon()	ImageIcon
getText()	String

Record Components: userInfo - The User info.

All Implemented Interfaces: SyncableItem, Selectable

```
public record SelectableUserInfo(UserInfo userInfo)
extends Record
implements ver14.view.Dialog.Selectables.Selectable, SyncableItem
represents a Selectable user info.
```

את שאר הפעולות ניתן לראות בקוד עצמו
Selectable Buttons

SelectableBtn	
comp()	Component
getValue()	Selectable
select(boolean)	void

```
public interface SelectableBtn
```

represents a button with a Selectable value.

getValue Selectable **getValue()**

Gets the selectable value of this button.

Returns: the selectable value of this button.

select 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

NormalButton	
NormalButton(Selectable, Callback<Selectable>)	
	selectedClr Color
	value Selectable
	selected boolean
	normalClr Color
select()	void
getValue()	Selectable
select(boolean)	void
unselect()	void

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, SwingConstants, SelectableBtn

public class **NormalButton**

extends MyJButton

implements SelectableBtn

represents a Normal selectable button.

select public void **select()**
Select this button.

unselect public void **unselect()**
Unselect this button.

את שאר הפעולות ניתן לראות בקוד עצמו

MenuItem	
MenuItem(Selectable, Callback<Selectable>)	
normalClr	Color
value	Selectable
selected	boolean
selectedClr	Color
getValue()	Selectable
select()	void
select(boolean)	void
unselect()	void

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants, SelectableBtn

```
public class MenuItem
extends JMenuItem
implements SelectableBtn
represents a Menu item with a selectable value.

MenuItem(Selectable value,
ver14.SharedClasses.Callback<ver14.view.Dialog.Selectables.Selectable
> onSelect)
Instantiates a new Menu item.

select public void select()
Select.

unselect public void unselect()
Unselect.
```

את שאר הפעולות ניתן לראות בקוד עצמן
[Icon Manager](#)

IconManager	
IconManager()	
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.
```

copyImage public static ImageIcon **copyImage**(ImageIcon og)
Copy image.

Parameters: og - the original image **Returns:** the new copy

loadOnline public static MyImage **loadOnline**(String path,
ver14.view.IconManager.Size... _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

scaleImage public static 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

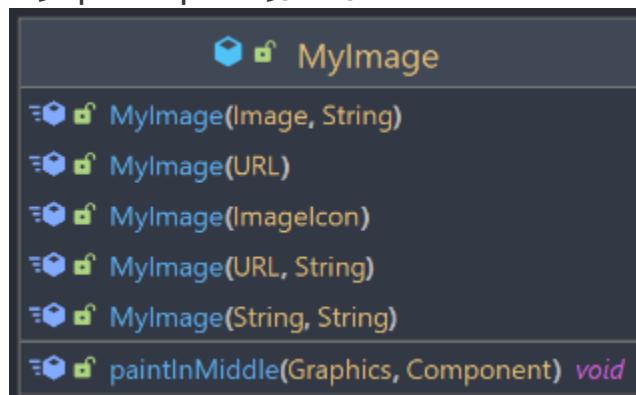
loadImage public static MyImage **loadImage**(String relativePath,
ver14.view.IconManager.Size... _size)
Load an image from a relative path.

Parameters: relativePath - the relative path _size - the optional size **Returns:** the loaded image

getPieceIcon public static ImageIcon **getPieceIcon**(Piece piece)
Gets a piece icon.

Parameters: piece - the piece **Returns:** the piece icon

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: Serializable, Accessible, Icon

public class **MyImage**
extends ImageIcon
My implementation of an icon.

paintInMiddle public 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	
DynamicIcon(ImagenIcon, ImagenIcon)	
DynamicIcon(String, Size)	
defaultHoverFilename	String
normal	ImagenIcon
defaultNormalFilename	String
hover	ImagenIcon
getNormal()	ImagenIcon
set(AbstractButton)	void
getHover()	ImagenIcon

```
public class DynamicIcon
```

Dynamic icon - represents an icon that changes depending on hover.

set public 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
...	

All Implemented Interfaces: Serializable, Cloneable

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

GameliconsGenerator	
GameliconsGenerator()	
maxFenLen	int
generate(String, PlayerColor, Dimension)	ImageIcon

```
public class GameIconsGenerator
Game icons generator - utility class for generating icons of positions.

generate public static 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
Menu Bar
```

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, MenuElement

```
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)	

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants, AuthorizedComponent

```
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

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants, AuthorizedComponent

```
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

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants

```
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

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants

```
public class Credits
extends MessageItem
```

Credits message item.

את שאר הפעולות ניתן לראות בקוד עצמו

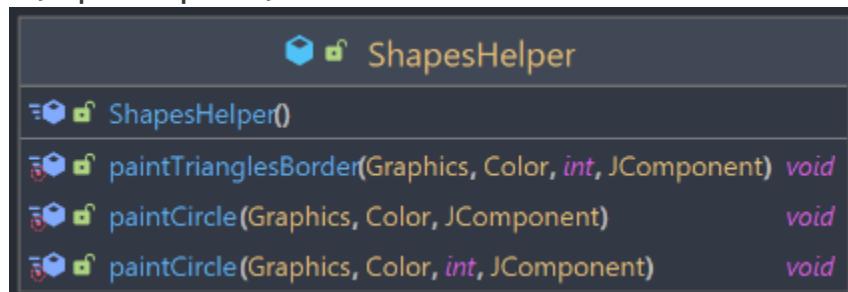
Rules	
Rules(Client)	
title()	String
onClick()	void
msg()	String

All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants

```
public class Rules
```

```
extends MessageItem
Rules message item.
```

את שאר הפעולות ניתן לראות בקובץ עצמן



```
public class ShapesHelper
```

The utility class Shapes helper. can paint some shapes.

paintTrianglesBorder public static 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

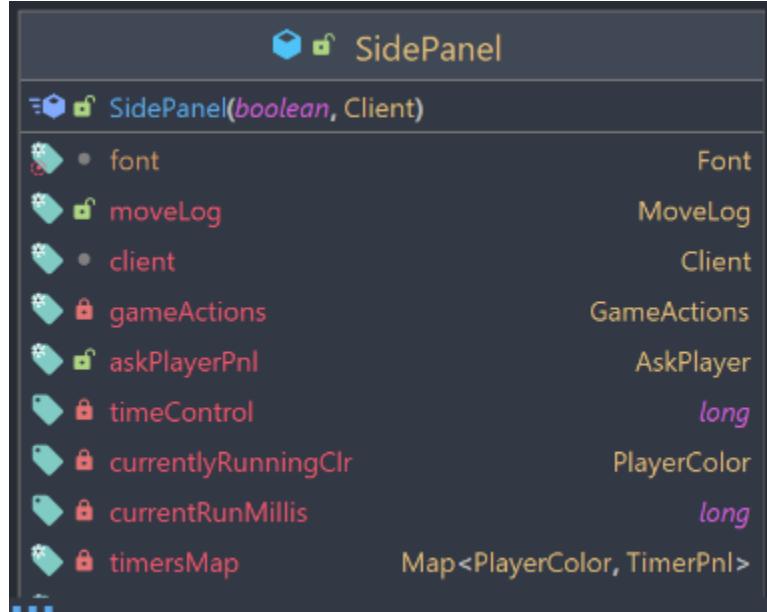
paintCircle public static void **paintCircle**(Graphics g, Color color, int diameterRatio, JComponent component)

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 בקובץ עצמן

Side Panel



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
public class SidePanel
extends JPanel
```

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

```
namesSync private void namesSync()
```

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

```
initGame public 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

```
sync public void sync(GameTime gameTime)
Sync timers.
```

Parameters: gameTime - the game time

```
startRunning public 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	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
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.

```
resignBtnClicked private void resignBtnClicked()
Resign btn clicked.
```

```
offerDrawBtnClicked private void offerDrawBtnClicked()
Offer draw btn c licked.
```

```
enableDrawOfferBtn public 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	
MoveLog()	
boardPanel	BoardPanel
forward	MyJButton
moveLogScroll	Scrollable
currentMoveIndex	int
end	MyJButton
start	MyJButton
blackMoves	JPanel
movesBtns	ArrayList <MyJButton>
back	MyJButton
...	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
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.

forward private void **forward()**

go one move Forward if possible.

back private void **back()**

go one move Backwards if possible.

createNavAdapter public KeyAdapter **createNavAdapter()**

Create nav adapter for navigating moves with the arrow keys.

Returns: the key adapter

switchTo private void **switchTo(int index)**

Switch to a certain move.

Parameters: index - the index

scroll private void **scroll()**

Scroll to the current move index.

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible,

ScrollPaneConstants

```
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.

```
ask public 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

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

Parameters: replacing - the replacing msg - the msg

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

Parameters: questionType - the question type to remove

את שאר הפעולות ניתן לראות בקוד עצמו

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
public class QuestionPnl
extends WinPnl
```

Question pnl - represents a single question panel.

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

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

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

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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible

```
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.
```

TimerPnl public *TimerPnl*(SidePanel sidePanel, PlayerColor color)
 Instantiates a new Timer pnl.

Parameters: sidePanel - the side panel color - the PlayerColor assigned to this timer panel

setName public void *setName*(String name)

Overrides: *setName* in class Component

את שאר הפעולות ניתן לראות בקוד עצמו

SyncedJmenus

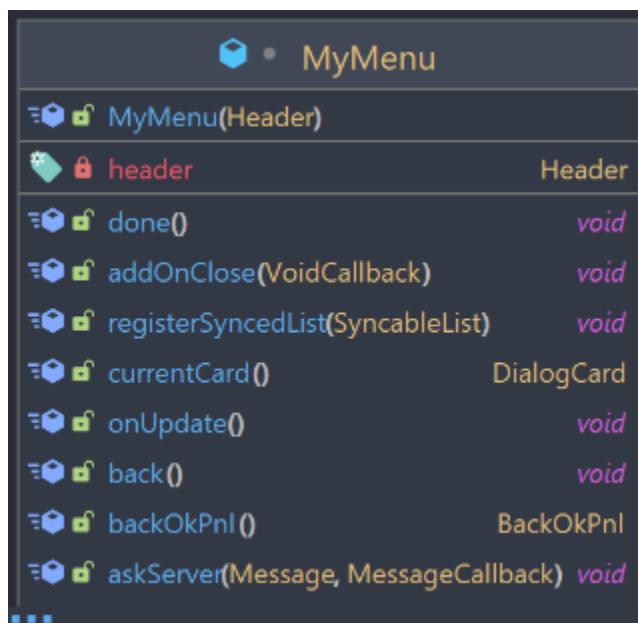
SyncedJMenu	
<i>SyncedJMenu</i> (Header, SyncedListType)	
<i>getJMenu()</i>	JMenu
<i>createButton</i> (Selectable)	SelectableBtn
<i>add</i> (Component)	Component
<i>onSelected()</i>	void
<i>removeContentComponent</i> (Component)	void

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified, SyncableList

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

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

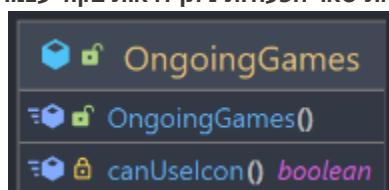
Parameters: header - the header listType - the list type
 את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants, Parent
 class **MyMenu**
 extends JMenu
 implements Parent
 My implementation of a jmenu used for a SyncedJMenu.

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

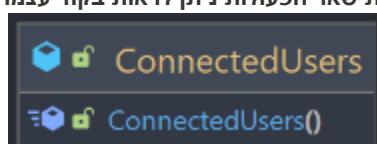
Parameters: header - the menu's header
 את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified, SyncableList
 public class **OngoingGames**
 extends SyncedJMenu
 a synchronized list of all Ongoing games.

OngoingGames public OngoingGames()
 Instantiates a new Ongoing games.

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, Child, Verified, SyncableList

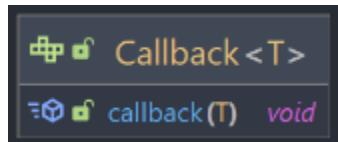
```
public class ConnectedUsers
extends SyncedJMenu
a synchronized list of all Connected users.

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

4.4.5 תיעוד המחלקות הקשורות ללקוח ולשרות

- המחלקה **Connection** מייצגת...
- המחלקה **Location** מייצגת...

Callbacks

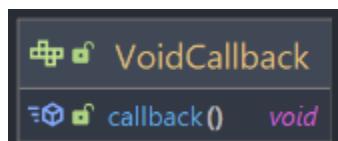


Type Parameters: `T` - the object's type

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.

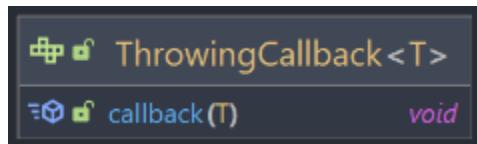
`callback void callback(T obj)` **Callback**.

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



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

את שאר הפעולות ניתן לראות בקובץ עצמו



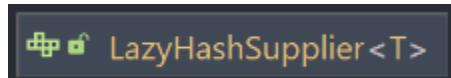
Type Parameters: `T` - the callback type

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

`callback void callback(T obj)`

`throws Exception` Callback.

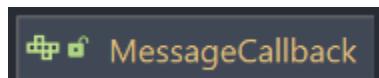
Parameters: `obj` - the obj **Throws:** `Exception` - the exception that might get thrown



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.



All Superinterfaces: `Callback<Message>`

`public interface MessageCallback`

`extends Callback<Message>` represents a message callback.

DB Actions

Arg

Arg	
<code>Arg(ArgType, Config<?>)</code>	
<code>Arg(ArgType)</code>	
<code>Arg(ArgType, boolean, Config<?>)</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>	

All Implemented Interfaces: Serializable

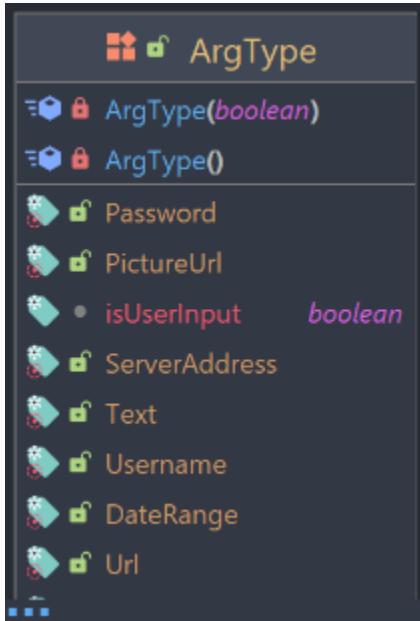
```
public class Arg
```

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

Arg 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

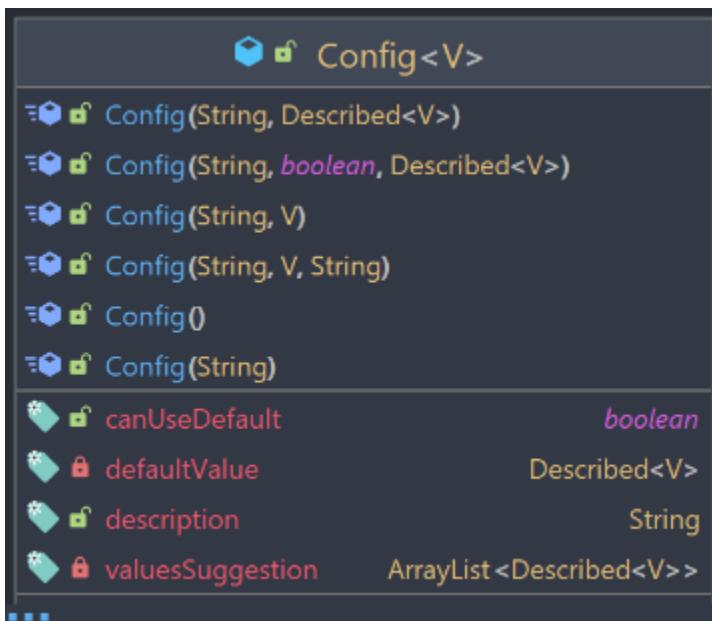


All Implemented Interfaces: Serializable, Comparable<ArgType>, Constable

```
public enum ArgType

extends Enum<ArgType> argument type.
```

את שאר הפעולות ניתן לראות בקובץ עצמו



Type Parameters: v - the type of the argument

All Implemented Interfaces: Serializable

```
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

Config 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



All Implemented Interfaces: Serializable

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

extends Record

implements Serializable represents a described object of type Described.

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

All Implemented Interfaces: Serializable

```
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.

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

Parameters: sqlStatement - the sql statement

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

Returns: the sub request

getRequest public String **getRequest**() Gets request.

Returns: the request

את שאר הפעולות ניתן לראות בקובץ עצמו

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

All Implemented Interfaces: Serializable, Comparable<Type>, Constable

```
public enum Type
```

extends Enum<Type> represents a db request type.

```
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)	PreMadeRequest
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

createBuilder `public RequestBuilder createBuilder() or createBuilder()` Create a new unique request builder

Returns: the request builder

❖ Variation	
❖ Variation(String, Object[], Arg[])	
buildingArgs	Object[]
variationArgs	Arg[]
variationName	String

```
public class Variation
```

Variation - represents a variation of a premade request.

Variation Variation(String variationName, Object[] buildingArgs, ver14.SharedClasses.DBActions.Arg.Arg[] variationArgs) Instantiates a new Variation.

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

❖ VariationCreator	
❖ Variation create(RequestBuilder)	Variation

```
public interface VariationCreator represents a creator of a variation.
```

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

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

DB Response

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

All Implemented Interfaces: Serializable

```
public abstract class DBResponse
```

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

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

Parameters: status - the status request - the request

את שאר הפעולות ניתן לראות בקוד עצמו

Status	
<code>Status()</code>	
<code>ERROR</code>	
<code>SUCCESS</code>	
<code>valueOf(String)</code>	Status
<code>values()</code>	Status[]

All Implemented Interfaces: Serializable, Comparable<Status>, Constable

```
public enum Status
```

extends Enum<Status> Status - represents a response status.

```
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

All Implemented Interfaces: Serializable

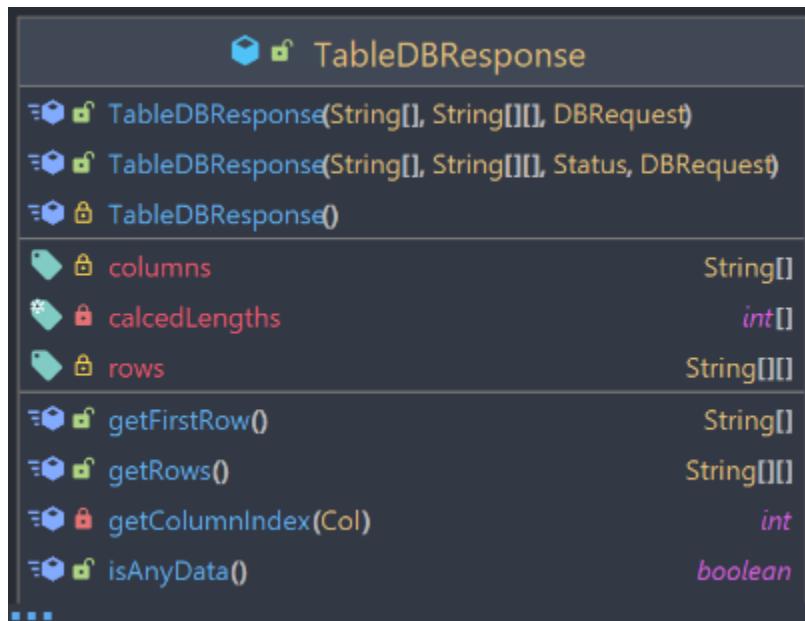
```
public class StatusResponse
```

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

StatusResponse 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 בקוד עצמו



All Implemented Interfaces: Serializable

```
public class TableDBResponse
```

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

TableDBResponse 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 בקוד עצמו

Statements

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

All Implemented Interfaces: Serializable

```
public abstract class SQLStatement
```

implements Serializable represents an sql statement. with a Type.

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

Parameters: type - the sql statement type

replace public 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

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

Returns: the created sql string

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

Returns: the statement

את שאר הפעולות ניתן לראות בקובץ 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>	

All Implemented Interfaces: Serializable

```
public class Selection

extends SQLStatement nt represents a selection sql statement.
```

Selection 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

nestMe public Selection **nestMe**(ver14.SharedClasses.DBActions.Table.Col... 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

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

Parameters: top - the number of results

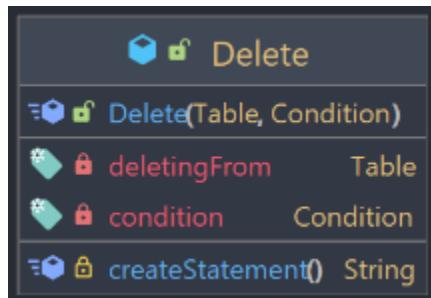
```
join public void join(String joinType, Table joinWith, Condition condition, ver14.SharedClasses.DBActions.Table.Col... groupBy) Join this selection with another Table.
```

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

```
orderBy public 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



All Implemented Interfaces: Serializable

```
public class Delete
extends SQLStatement nt represents a deletion statement.
```

```
Delete 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	
<code>Update(Table, Condition, NewValue[])</code>	
<code>Condition</code>	<code>condition</code>
<code>NewValue[]</code>	<code>newValues</code>
<code>Table</code>	<code>updating</code>
<code>String</code>	<code>createStatement()</code>

All Implemented Interfaces: Serializable

```
public class Update
```

extends SQLStatement
It represents an update statement.

Update 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 בקובץ עצמו

NewValue	
<code>NewValue (Col, Object)</code>	
<code>Col</code>	<code>col</code>
<code>Object</code>	<code>value</code>
<code>Object</code>	<code>value0</code>
<code>Col</code>	<code>col0</code>
<code>String</code>	<code>toString()</code>

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

All Implemented Interfaces: Serializable

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

extends Record

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

See Also: Serialized Form

NewValue 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

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

All Implemented Interfaces: Serializable, Comparable<Table>, Constable

```
public enum Table

extends Enum<Table> represents a table in the db.
```

Table private **Table**(ver14.SharedClasses.DBActions.Table.Col... 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



All Implemented Interfaces: Serializable

```
public class Col
```

implements Serializable represents a column, either existing column in the db (the constant columns GameID, SavedGame ...) or created columns.

count public static 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

countIf public static 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

sum public static 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

switchCase public static Col **switchCase**(String as, ver14.SharedClasses.DBActions.Table.SwitchCase... 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

time public Col **time()** Time col.

Returns: a new col representing time

date public Col **date()** Date col.

Returns: a new col representing datetime

of public 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

All Implemented Interfaces: Serializable

```
public class CustomCol
```

extends Colol represents a Custom column that will keep its name, even on nesting.

CustomCol 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
...

All Implemented Interfaces: Serializable, Comparable<Math>, Constable

```
public enum Math
```

extends Enum<Math> Math - allows for math operations on columns and some math-related utilities for columns.

```
Math 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

formatNum public static String **formatNum**(Object num) Format a num with a default 3 decimal places.

Parameters: num - the num **Returns:** the string

formatNum public static 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

asFloat public static String **asFloat**(Object num) cast value as float.

Parameters: num - the num **Returns:** the formatted value

zeroIfNull protected void **zeroIfNull()** if the col's value is null, it will be replaced with a 0.

execute public Col **execute**(ver14.SharedClasses.DBActions.Table.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)
```

extends Record 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

SwitchCase 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

equals public static 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

defaultCase public static SwitchCase `defaultCase(Col ifTrue)` if none of the previous cases matched, this column will show.

Parameters: ifTrue - the if true **Returns:** the switch case

ifTrue public Col **ifTrue()** ol **ifTrue()** If true col.

Returns: the col

את שאר הפעולות ניתן לראות בקוד עצמו

Condition	
Condition(String, Object[])	
parms	Object[]
str	String
equals(Object, Object)	Condition
add(Condition, Relation, boolean)	Condition
toString()	String
getStr()	String
notEquals(Object, Object)	Condition
between(Object, Object, Object)	Condition
setStr(String)	void
...	

All Implemented Interfaces: Serializable

```
public class Condition
```

implements Serializable represents a condition.

Condition public **Condition**(String str, Object... parms) Instantiates a new Condition.

Parameters: str - the str parms - the parms

equals public static Condition **equals**(Object col, Object value) Equals condition. matches the toString of both objects.

Parameters: col - the col value - the value **Returns:** the condition

noNulls public 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

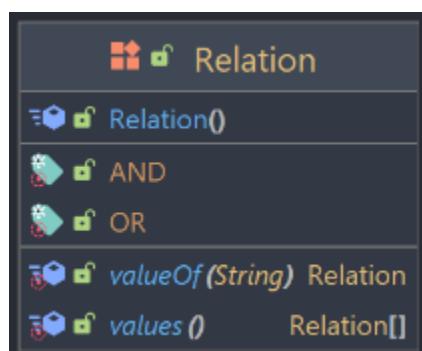
add public Condition **add**(Condition condition, Relation relation, boolean wrap)
Add a condition to this condition.

Parameters: condition - the condition relation - the relation between the conditions wrap - the wrap
Returns: THIS condition

between public static 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



All Implemented Interfaces: Serializable, Comparable<Relation>, Constable

public enum **Relation**

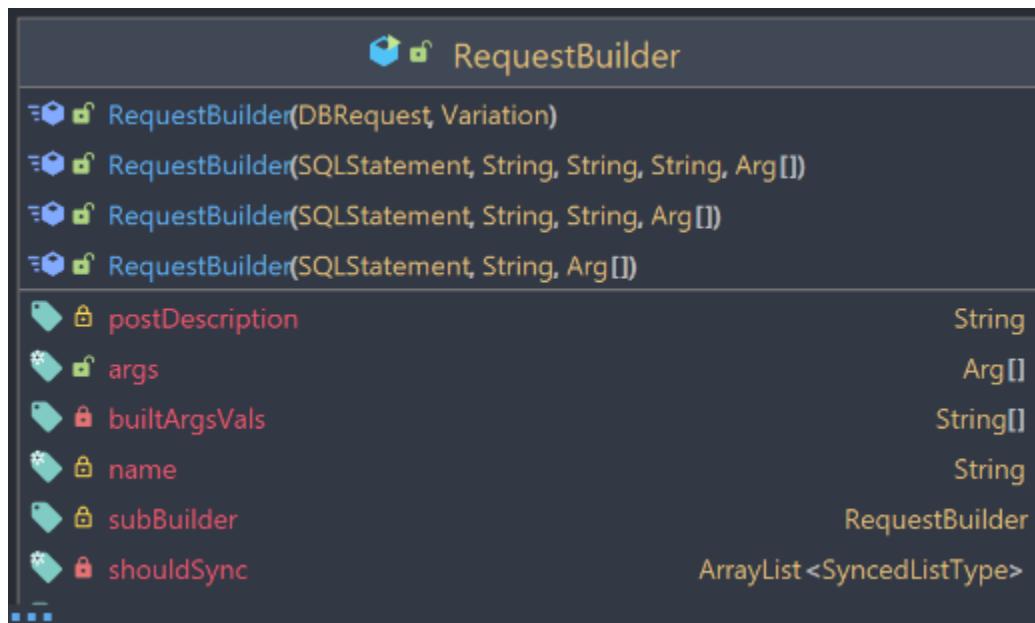
extends Enum<Relation> represents relations between conditions.

Relation `private Relation()`

`Relation AND` And relation.

`Relation OR` Or relation.

את שאר הפעולות ניתן לראות בקובץ עצמו



All Implemented Interfaces: Serializable

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

RequestBuilder public `RequestBuilder`(SQLStatement statement, String name, String

```
postDescription, String preDescription,  
ver14.SharedClasses.DBActions.Arg.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.

```
createVariation public static 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

```
build public 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 עצמו

[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>	<i>boolean</i>	
<code>gameStatus</code>	<i>GameStatus</i>	
<code>perspective</code>	<i>PlayerColor</i>	
<code>detailedEval</code>	<i>Collection<EvaluationDetail></i>	
<code>TIE_EVAL</code>	<i>int</i>	
<code>WIN_EVAL</code>	<i>int</i>	
<code>...</code>		

All Implemented Interfaces: Serializable

```
public class Evaluation
```

implements Serializable represents a position's evaluation relative to a PlayerColor.

book public static Evaluation **book()** Book evaluation.

Returns: the evaluation

addDetail public void **addDetail**(EvaluationParameters parm, int value) Add a detail to the evaluation.

Parameters: parm - the kind of detail value - the value

isCheck public boolean **isCheck()** Is this evaluation a check.

Returns: true if this evaluation is a check

setPerspective public 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

flipEval public void **flipEval()** Flip the evaluation, will multiply the eval by -1.

את שאר הפעולות ניתן לראות בקוד עצמו

EvaluationDetail	
EvaluationDetail (EvaluationParameters, int)	
eval	int
parm	EvaluationParameters
parm0	EvaluationParameters
eval0	int
toString()	String

All Implemented Interfaces: Serializable

```
public record EvaluationDetail(EvaluationParameters parm, int eval)
```

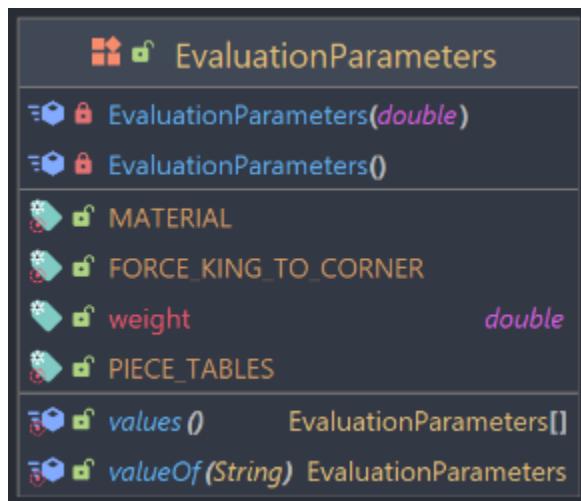
```
extends Record
```

```
implements Serializable represents an Evaluation detail.
```

EvaluationDetail 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 בקוד עצמו



All Implemented Interfaces: Serializable, Comparable<EvaluationParameters>, Constable

```
public enum EvaluationParameters
```

extends `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.

EvaluationParameters 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	<i>int</i>
checkmate(PlayerColor, Location)	GameStatus

...
Serializable

All Implemented Interfaces:

```
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.

checkmate public static GameStatus **checkmate**(PlayerColor winningPlayerColor, Location matedKing) Checkmate game status.

Parameters: winningPlayerColor - the winning player color
matedKing - the mated king **Returns:** the game status

gameGoesOn public static GameStatus **gameGoesOn()** Game goes on game status.

Returns: the game status

tieByAgreement public static GameStatus **tieByAgreement()** Tie by agreement game status.

Returns: the game status

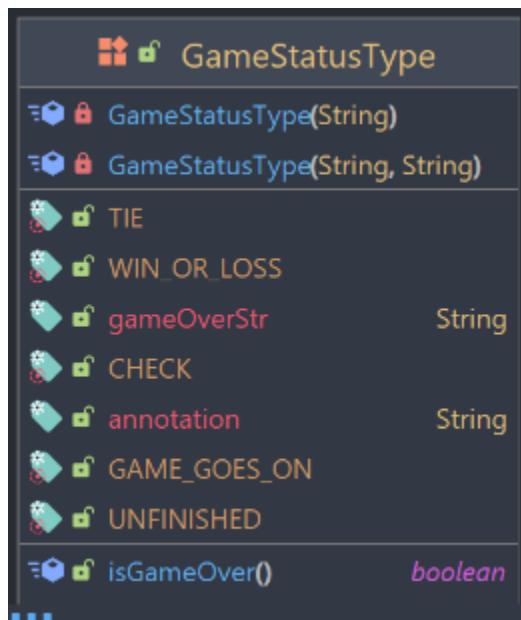
stalemate public static GameStatus **stalemate()** Stalemate game status.

Returns: the game status

fiftyMoveRule public static GameStatus **fiftyMoveRule()** Fifty move rule game status.

Returns: the game status

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: `Serializable`, `Comparable<GameStatusType>`, `Constable`

```
public enum GameStatusType
```

```
extends 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.

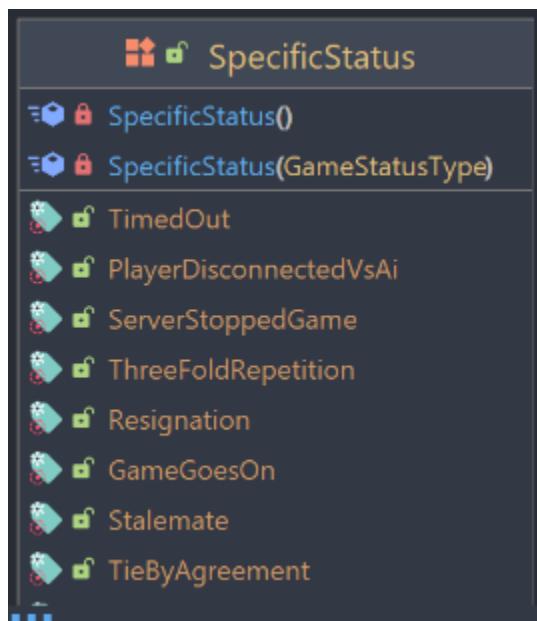
GameStatusType **WIN_OR_LOSS** a player won.

GameStatusType **UNFINISHED** the game is unfinished.

isGameOver public boolean **isGameOver()** Is game over.

Returns: true if is game over. false otherwise

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: Serializable, Comparable<SpecificStatus>, Constable

public enum **SpecificStatus**

extends **Enum<SpecificStatus>** represents a specific game status with all its details.

SpecificStatus 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 .

SpecificStatus **ThreeFoldRepetition** Three fold repetition.

SpecificStatus **Stalemate** Stalemate.

SpecificStatus **InsufficientMaterial** Insufficient material.

SpecificStatus **FiftyMoveRule** Fifty move rule.

SpecificStatus **TieByAgreement** The Tie by agreement.

SpecificStatus **PlayerDisconnectedVsAi** The Player disconnected vs ai.

SpecificStatus **PlayerDisconnectedVsReal** The Player disconnected vs real.

SpecificStatus **ServerStoppedGame** Server stopped game.

את שאר הפעולות ניתן לראות בקובץ עצמו

[Game Setup](#)

[Board Setup](#)

Piece	
<code>=锁定 锁 Piece(PieceType, PlayerColor)</code>	
<code>pieceType</code>	PieceType
<code>playerColor</code>	PlayerColor
<code>W_P</code>	
<code>W_B</code>	
<code>PIECES_ICONS</code>	String[]()
<code>W_Q</code>	
<code>W_N</code>	
<code>B_N</code>	
<code>B_K</code>	
<code>...</code>	

All Implemented Interfaces: Serializable, Comparable<Piece>, Constable

```
public enum Piece
```

extends Enum<Piece> represents a combination of a PieceType and PlayerColor.

Piece 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.

Piece `W_Q` represents a White Queen.

Piece `W_K` represents a White King.

Piece `B_P` represents a Black pawn.

Piece **B_R** represents a Black Rook.

Piece **B_B** represents a Black Bishop.

Piece **B_N** represents a Black Knight.

Piece **B_Q** represents a Black Queen.

Piece **B_K** represents a Black King.

את שאר הפעולות ניתן לראות בקוד עצמו

PieceType	
•	PieceType(String, String, int, boolean)
•	PieceType(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

All Implemented Interfaces: Serializable, Comparable<PieceType>, Constable

```
public enum PieceType
    extends 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.

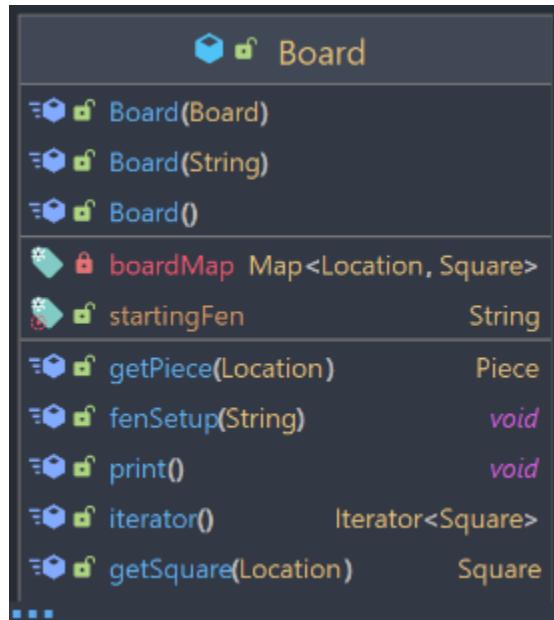
PieceType **QUEEN** Queen piece type.

PieceType **KING** King piece type.

isAttack public 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



All Implemented Interfaces: Serializable, Iterable<Square>

```
public class Board
```

implements `Iterable<Square>`, `Serializable` represents a logic board with 64 Squares

Board public `Board()` Instantiates a new empty Board.

Board public `Board(String fen)` Instantiates a new Board and sets it up with a fen.

Parameters: fen - the fen

setPiece public void `setPiece(Location loc, Piece piece)` Sets a piece on a square.

Parameters: loc - the square's loc **Returns:** piece - the piece

getPiece public Piece `getPiece(ver14.SharedClasses.Game.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

isSquareEmpty public boolean `isSquareEmpty(Location loc)` Is a square empty.

Parameters: loc - the loc **Returns:** true if the square is empty

את שאר הפעולות ניתן לראות בקובץ עצמו

Square		
Constructor		
 ` `	Square(Location)	
 ` `	 ` `	EMPTY_PIECE_STR String
 ` `	 ` `	loc Location
 ` `	 ` `	EMPTY_PIECE Piece
 ` `	 ` `	piece Piece
 ` `	 ` `	getPiece() Piece
 ` `	 ` `	getPieceIcon() String
 ` `	 ` `	getFen() String
 ` `	 ` `	toString() String
 ` `	 ` `	isEmpty() boolean
...		

All Implemented Interfaces: Serializable

```
public class Square

implements Serializable represents a square on the logic board. with a Location and a Piece
```

Constructor: public **Square**(Location loc) Instantiates a new Square.

Parameters: loc - the location of the square

setEmpty public void **setEmpty()** Set this square to be empty.

getPiece public Piece **getPiece()** Gets the piece on this square.

Returns: the piece if this square isn't empty, or **EMPTY_PIECE** if it is.

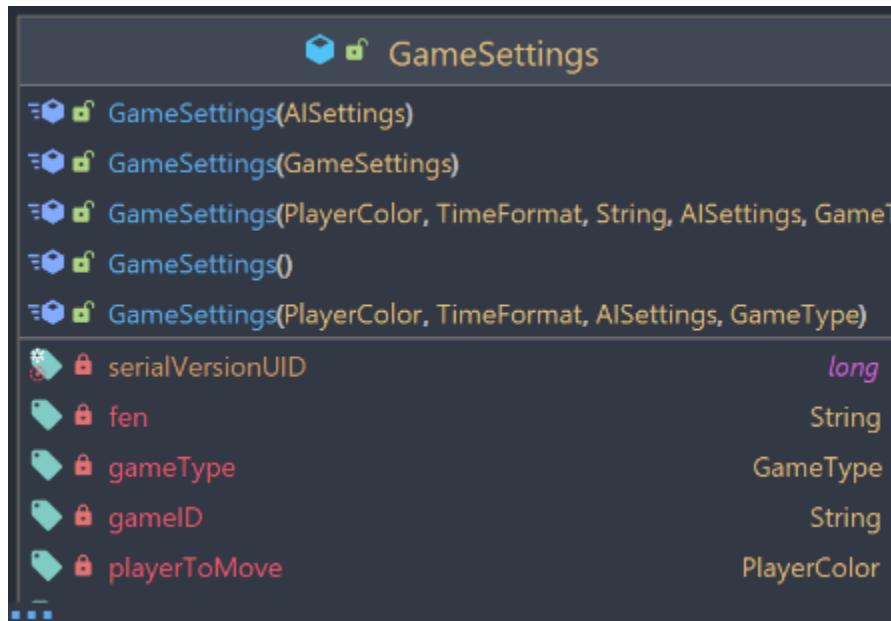
setPiece public void **setPiece(Piece piece)** Sets a piece on this square.

Parameters: piece - the piece

isEmpty public boolean **isEmpty()** Is this square empty.

Returns: true if this square is empty

את שאר הפעולות ניתן לראות בקובץ עצמו



All Implemented Interfaces: Serializable, TimeFormatComponent

```
public class GameSettings
```

implements Serializable, TimeFormatComponent
It represents Game settings. the starting position, which PlayerColor turn is it to move, the TimeFormat, the AISettings and the GameType

GameSettings 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

GameType	
GameType()	
JOIN_EXISTING	
CREATE_NEW	
QUICK_MATCH	
RESUME	
valueOf(String)	GameType
values()	GameType[]

All Implemented Interfaces: Serializable, Comparable<GameType>, Constable

```
public enum GameType
```

extends Enum<GameType> represents a Game's type.

GameType 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 בקוד עצמו



```
public class TimeFormat
```

implements `Serializable` represents a player's Time format. how much time does he have for each move.

TimeFormat public `TimeFormat`(long timeInMillis) Instantiates a new Time format.

Parameters: timeInMillis - the time in millis

את שאר הפעולות ניתן לראות בקובץ עצמו



All Implemented Interfaces: Serializable

```
public class GameTime
```

implements Serializable represents the current state of both player's time.

GameTime public **GameTime** (ver14.SharedClasses.Game.GameSetup.TimeFormat... 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

getTimeFormat public TimeFormat
getTimeFormat (ver14.SharedClasses.Game.PlayerColor clr) at **getTimeFormat**(PlayerColor clr) Gets the time format for a player.

Parameters: clr - the clr **Returns:** the time format

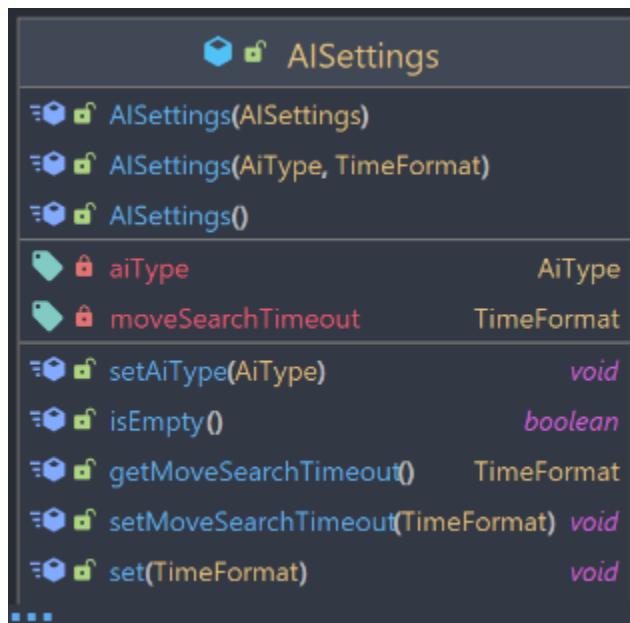
startRunning public void **startRunning**(PlayerColor playerColor) Start running a player's clock.

Parameters: playerColor - the player color

getTimeLeft public 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 בקוד עצמו



All Implemented Interfaces: Serializable, ParentOf<TimeFormat>

```
public class AISettings
```

implements **Serializable**, **ParentOf<TimeFormat>** represents Ai Settings. the **AiType** and how much time can it think.

AISettings public **AISettings**(AiType aiType, TimeFormat moveSearchTimeout)
Instantiates a new Ai parameters.

Parameters: aiType - the ai type moveSearchTimeout - the move search timeout

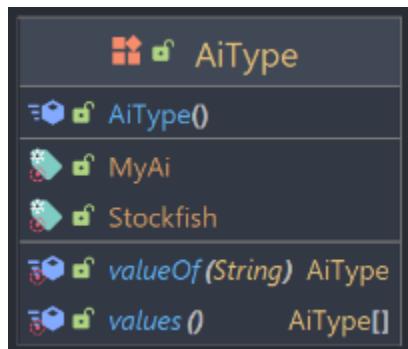
getMoveSearchTimeout public TimeFormat **getMoveSearchTimeout()** at
getMoveSearchTimeout() Gets move search timeout.

Returns: the move search timeout

setMoveSearchTimeout public void **setMoveSearchTimeout**(TimeFormat moveSearchTimeout) Sets move search timeout.

Parameters: moveSearchTimeout - the move search timeout

את שאר הפעולות ניתן לראות בקובץ עצמו



All Implemented Interfaces: Serializable, Comparable<AiType>, Constable

```

public enum AiType
    extends Enum<AiType> Ai type.

```

```
AiType private AiType()
```

```
AiType Stockfish Stockfish ai type.
```

```
AiType MyAi My ai type.
```

את שאר הפעולות ניתן לראות בקובץ עצמו

Location	
<code>Location()</code>	
<code>WHITE_STARTING_ROW</code>	<code>int</code>
<code>D</code>	<code>int</code>
<code>blackSquares</code>	<code>long</code>
<code>asInt</code>	<code>int</code>
<code>F6</code>	
<code>H5</code>	
<code>B8</code>	
<code>F5</code>	
<code>B2</code>	
<code>...</code>	

All Implemented Interfaces: Serializable, Comparable<Location>, Constable

```
public enum Location
```

extends 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.

`Location` private `Location()` Instantiates a new Location.

`Location A8`

`Location B8`

`Location C8`

את השאר ניתן לראות בקוד

`getLoc` public static 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

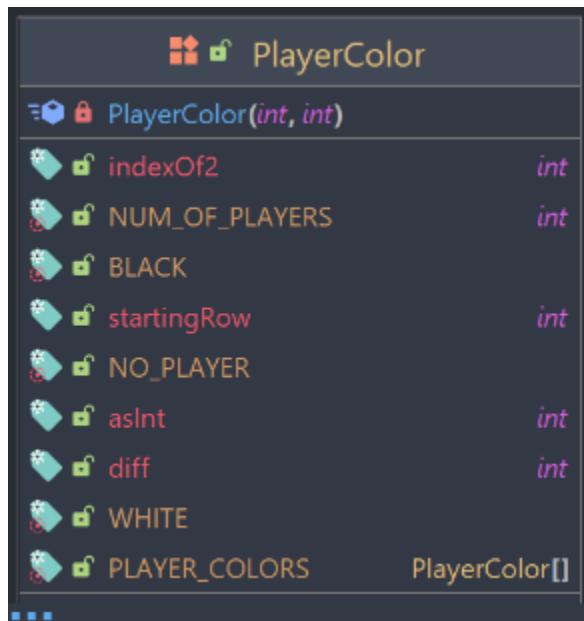
getLoc public static Location **getLoc**(Location loc, int add) Gets the location that is exactly add squares from loc
NOTE: add should be in bitboard format

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

getLoc public static 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 בקוד עצמו



All Implemented Interfaces: Serializable, Comparable<PlayerColor>, Constable

```
public enum PlayerColor
```

extends Enum<PlayerColor> represents a player color.

PlayerColor private **PlayerColor**(int startingRow, int diff) Instantiates a new Player color.

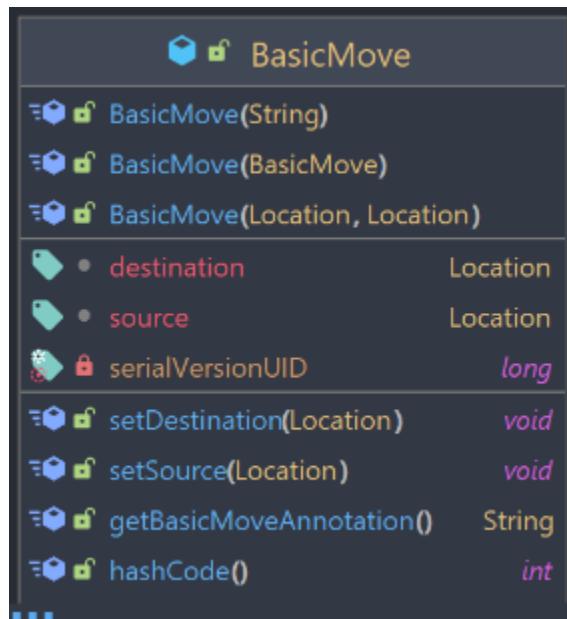
Parameters: startingRow - the starting row diff - the diff

PlayerColor **WHITE** White.

PlayerColor **BLACK** Black.

PlayerColor **NO_PLAYER** No player.

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: Serializable

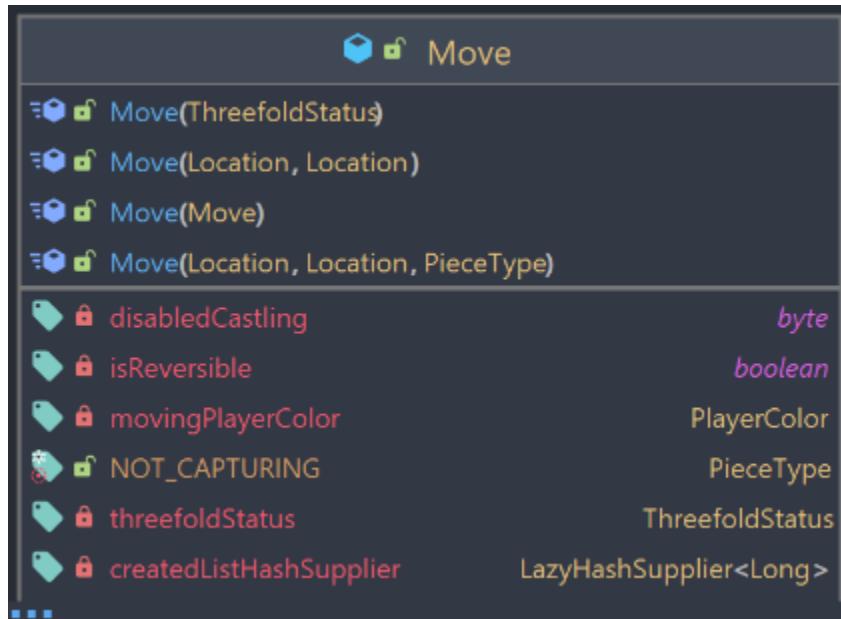
```
public class BasicMove
```

implements Serializable Basic move - represents a basic move. with a source and a destination.

BasicMove 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



All Implemented Interfaces: Serializable, Comparable<Move>

```
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.

Move public **Move**(Location source, Location destination) Instantiates a new Move.

Parameters: source - the source destination - the destination

castling public static 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

isCheck public boolean **isCheck**() Is this move a check.

Returns: true if this move is a check

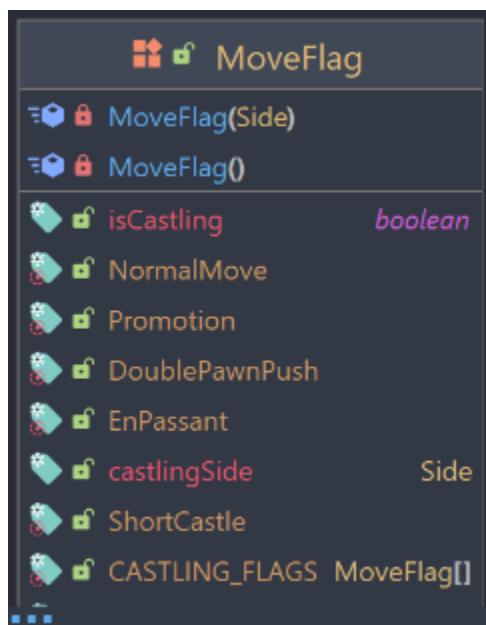
getCapturingPieceType public PieceType **getCapturingPieceType ()** pe
getCapturingPieceType() Gets the type of piece this move captures.

Returns: the piece if one is captured, null otherwise.

isReversible public 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



All Implemented Interfaces: `Serializable`, `Comparable<Move.MoveFlag>`, `Constable`

Enclosing class: `Move`

`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



All Implemented Interfaces: `Serializable, Comparable<Direction>, Constable`

```
public enum Direction
```

extends `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 בקובץ

perspective public 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 בקובץ עצמו

CastlingRights	
<code>CastlingRights(CastlingRights)</code>	
<code>CastlingRights(byte)</code>	
<code>CastlingRights()</code>	
<code>rights</code>	<code>byte</code>
<code>PLAYER_MASKS</code>	<code>byte[]</code>
<code>NO_CASTLING_ABILITY</code>	<code>String</code>
<code>RIGHTS</code>	<code>byte[]</code>
<code>FENS</code>	<code>String[]</code>
<code>toString()</code>	<code>String</code>
<code>disableCastling(PlayerColor, Side)</code>	<code>byte</code>
<code>...</code>	

All Implemented Interfaces: Serializable

```
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

CastlingRights public `CastlingRights()` Instantiates a new Castling rights. with a default value of 0. (no one can castle)

enableCastling public void `enableCastling(PlayerColor playerColor, CastlingRights.Side side)` Enable castling for a player on a side.

Parameters: playerColor - the player color side - the side

isEnabled public 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

disableCastling public 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 בקובץ עצמו

Side	
Side(int, int, int)	
castledKingCol	int
asInt	int
rookStartingCol	int
SIDES	Side[]
kingTravelDistance	int
QUEEN	
KING	
castledRookCol	int
mult	int
...	

All Implemented Interfaces: Serializable, Comparable<CastlingRights.Side>, Constable

Enclosing class: CastlingRights

```
public static enum CastlingRights.Side
```

extends Enum<CastlingRights.Side> Castling side.

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.

kingFinalLoc public Location kingFinalLoc(ver14.SharedClasses.Game.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		
BitData()		
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

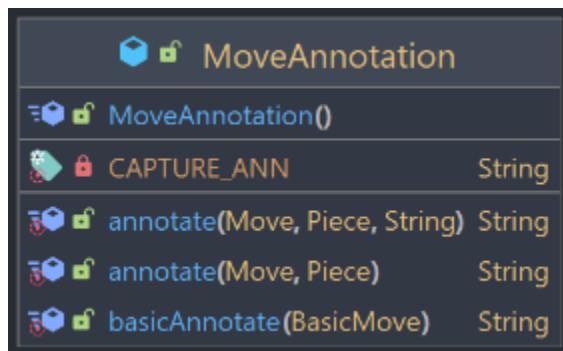
MovesList		
MovesList(MovesList)		
MovesList()		
hash	<i>long</i>	
finalHash	<i>long</i>	
finalizeHash()	<i>void</i>	
findMove(BasicMove, CompareMoves)	Move	
findMove(BasicMove)	Move	
add(Move)	<i>boolean</i>	
getFinalHash()	<i>long</i>	
createSimpleStr()	String	
...		

All Implemented Interfaces: Serializable, Cloneable, Iterable<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 בקובץ עצמו



```
public class MoveAnnotation
```

utility class for annotating moves.

annotate public static String **annotate**(Move move, Piece movingPiece) Annotate move
in the PGN format

Parameters: move - the move movingPiece - the moving piece **Returns:** the annotation

basicAnnotate public static String **basicAnnotate**(BasicMove move) Basic annotate a move.
format:[source:destination]

Parameters: move - the move **Returns:** the string

את שאר הפעולות ניתן לראות בקובץ עצמו

MinimaxMove	
MinimaxMove(Evaluation)	
MinimaxMove(MinimaxMove)	
MinimaxMove(Move, Evaluation)	
moveEvaluation	Evaluation
move	Move
moveDepth	int
isDeeperAndBetterThan(MinimaxMove)	boolean
getShortPrintingStr()	String
setMoveEvaluation(Evaluation)	void
equals(Object)	boolean
...	

All Implemented Interfaces: Serializable, Comparable<MinimaxMove>

```
public class MinimaxMove
```

implements Comparable<MinimaxMove>, Serializable represents a Minimax move with a score and depth.

MinimaxMove public **MinimaxMove** (Move move, Evaluation moveEvaluation) Instantiates a new Minimax move.

Parameters: move - the move moveEvaluation - the move evaluation

isDeeperAndBetterThan public 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	
<code>GameInfo(String, String, GameSettings)</code>	
<code>gameId</code>	<code>String</code>
<code>gameSettings</code>	<code>GameSettings</code>
<code>serialVersionUID</code>	<code>long</code>
<code>creatorUsername</code>	<code>String</code>
<code>getStartingColor()</code>	<code>PlayerColor</code>
<code>getJoiningPlayerColor()</code>	<code>PlayerColor</code>
<code>getGameDesc()</code>	<code>String</code>
<code>toString()</code>	<code>String</code>
<code>isCreator(String)</code>	<code>boolean</code>
<code>...</code>	

All Implemented Interfaces: `Serializable, SyncableItemem`

```
public class GameInfo
```

`implements Serializable, SyncableItemem` represents a Game info.

`GameInfo 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 בקובץ עצמו

EstablishedGameInfo	
EstablishedGameInfo(String, String, String, GameSettings, Stack<Move>)	
moveStack	Stack<Move>
createdAt	Date
opponentUsername	String
setCreatedAt(Date)	void
toString()	String
getMoveStack()	Stack<Move>
getGameDesc()	String
getCreatedAt()	Date
All Implemented	

Interfaces: Serializable, SyncableItemem

```
public abstract class EstablishedGameInfo
```

extends GameInfofo represents a game that was established between two players.

EstablishedGameInfo 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 בקובץ עצמו

UnfinishedGame	
UnfinishedGame(String, String, GameSettings, String, PlayerColor)	
playerColorToMove	PlayerColor
playerToMove	String
isCreatorToMove()	boolean
All Implemented	

Interfaces: Serializable, SyncableItemem

```
public class UnfinishedGame
```

extends EstablishedGameInfo to represents an Unfinished game.

את שאר הפעולות ניתן לראות בקובץ עצמו

ArchivedGameInfo	
String	ArchivedGameInfo(String, String, String, GameSettings, String)
String	winner
String	toString()
String	getWinner()
All Implemented Interfaces: Serializable, SyncableItemem	

```
public class ArchivedGameInfo
```

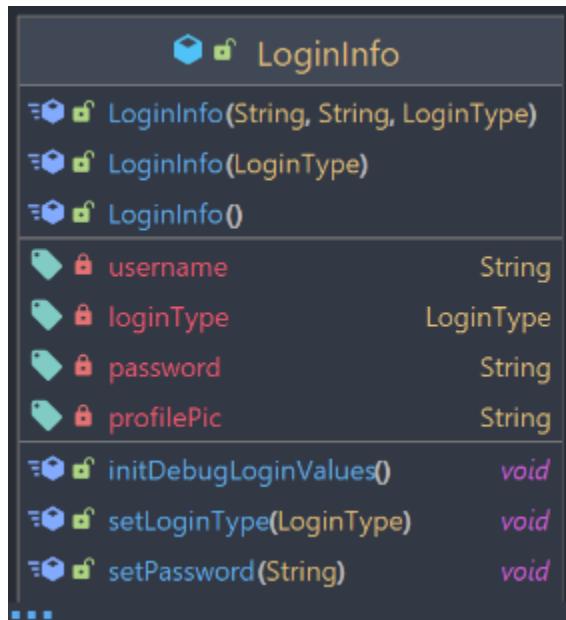
extends EstablishedGameInfo to represents a game that was finished, and is intended to be archived in the db.

getWinner public 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



All Implemented Interfaces: Serializable

```
public class LoginInfo  
  
implements Serializable represents Login info .
```

LoginInfo 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 בקוד עצמו



All Implemented Interfaces: Serializable,
Comparable<LoginType>, Constable

```

public enum LoginType

extends 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.

asUser public 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 בקובץ עצמו

```

@ AuthSettings
  ⚡ GUEST      int
  ⚡ ANY_LOGIN   int
  ⚡ NEVER_AUTH int
  ⚡ USER        int
  ⚡ NO_AUTH    int
  public @interface AuthSettings Auth settings. represents all types
of login authentication.

```

Networking

AppSocket	
⚡ AppSocket(String, int)	
⚡ AppSocket(Socket)	
⚡ messagesHandler	MessagesHandler
⚡ msgIS	ObjectInputStream
⚡ msgSocket	Socket
⚡ msgOS	ObjectOutputStream
⚡ didDisconnect	boolean
⚡ requestMessage(Message)	Message
⚡ stopReading()	void
⚡ close(MyError)	void
...	

All Implemented Interfaces:
Runnable

```
public class AppSocket
```

extends MyThread
This class represents a communications socket able to send and receive Messages to and from another AppSocket.

AppSocket public **AppSocket**(String ip, int port)

throws IOException Instantiates a new App socket to a specified address

Parameters: ip - the ip port - the port **Throws:** IOException - the io exception

close public void **close**(MyError err) Close this socket, and interrupt every method waiting for a message.

Parameters: err - the error to interrupt with

requestMessage public 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

respond public void **respond**(Message msg, Message respondingTo) Respond to a request message.

Parameters: msg - the response respondingTo - the request message

writeMessage public void **writeMessage**(Message msg) Write a message.

Parameters: msg - the message

requestMessage public Message **requestMessage**(ver14.SharedClasses.Networking.Messages.Message requestMsg) get
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 בקוד עצמו

MessagesHandler	
MessagesHandler(AppSocket)	
customCallbacks	Map<String, MessageCallback>
waiting	Vector<CompletableFuture<Message>>
socket	AppSocket
chronologicalSemaphore	Semaphore
didDisconnect	boolean
defaultCallbacks	Map<MessageType, MessageCallback>
isExpectingDisconnect	boolean
onDisconnected()	void
onAnyDisconnection()	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

interruptBlocking public void **interruptBlocking**(MyError err) interrupt every thread waiting for a response with an error.

Parameters: err - the error

blockTilRes public Message
blockTilRes(ver14.SharedClasses.Networking.Messages.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

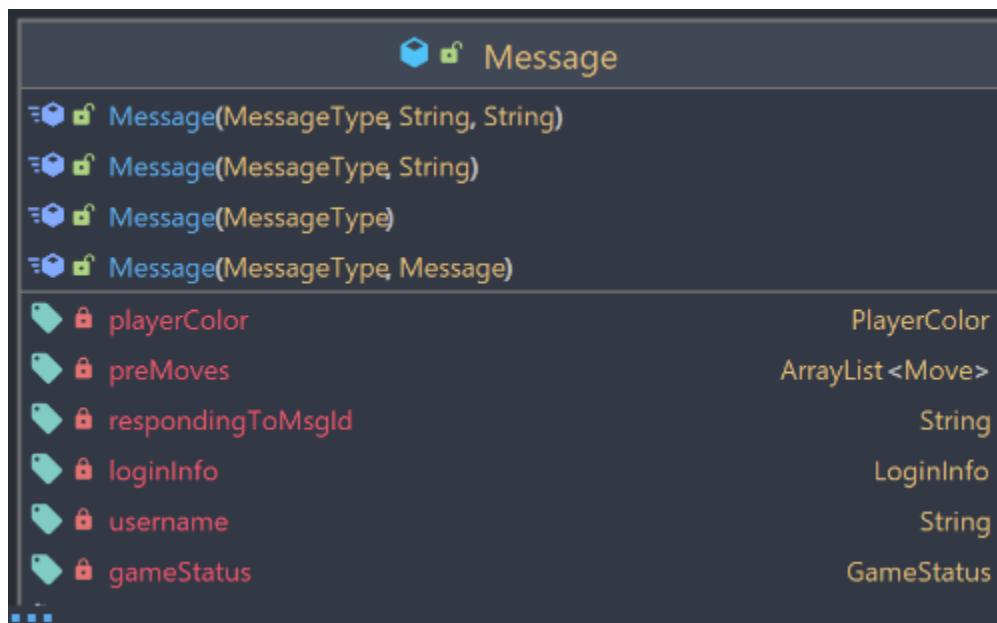
noBlockRequest public void **noBlockRequest**(Message request, MessageCallback onRes) send a non-blocking request.

Parameters: request - the request onRes - the callback to call once a response is received

processMessage private void **processMessage**(Message message) Process a message read from the socket.

Parameters: message - the message

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: Serializable

```
public class Message
```

implements Serializable represents a Message that is used to carry information over the network.

Message 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

Message public **Message**(MessageType messageType, String subject) Instantiates a new Message.

Parameters: messageType - the message type subject - the subject

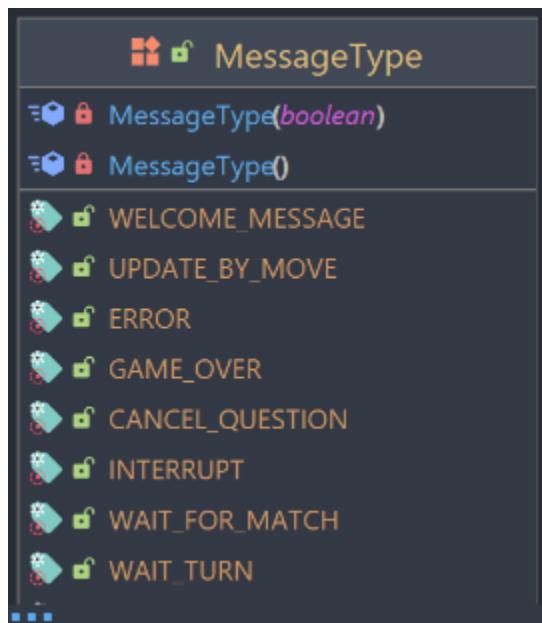
askForLogin public static Message **askForLogin()** create a request for a login message.

Returns: the message

returnLogin public static 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



All Implemented Interfaces: Serializable, Comparable<MessageType>, Constable

public enum **MessageType**

extends **Enum<MessageType>** represents a Message type.

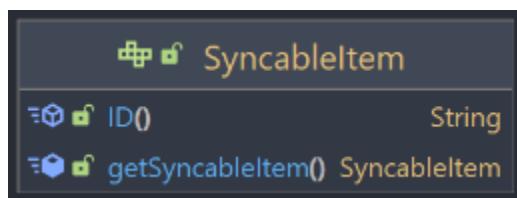
MessageType LOGIN Login message type.

MessageType **RESIGN** Resign message type.

MessageType **WELCOME_MESSAGE** Welcome message type.

את השאר ניתן לראות בקוד

Sync



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

ID String **ID()** a syncable item must have a unique id.

Returns: the id

SyncedItems<E>	
SyncedItems(SyncedListType)	
SyncedItems(SyncedListType, Callback <SyncedItems<E>>)	
serialVersionUID	<i>long</i>
syncedListType	<i>SyncedListType</i>
onUpdate	<i>Callback <SyncedItems<E>></i>
put(E)	<i>SyncableItem</i>
updated()	<i>void</i>
addAll(Collection <SyncableItem>)	<i>void</i>
put(String, E)	<i>E?</i>
forEachItem(Callback <E>)	<i>void</i>
...	

Type Parameters: E - the type of syncable elements in this collection

All Implemented Interfaces: Serializable, ConcurrentMap<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

SyncedItems 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 בקוד עצמו

SyncedListType	
<code>SyncedListType()</code>	
<code>CONNECTED_USERS</code>	
<code>ONGOING_GAMES</code>	
<code>JOINABLE_GAMES</code>	
<code>RESUMABLE_GAMES</code>	
<code>valueOf(String)</code>	SyncedListType
<code>values()</code>	SyncedListType[]

All Implemented Interfaces: Serializable, Comparable<SyncedListType>, Constable

```
public enum SyncedListType
```

extends 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

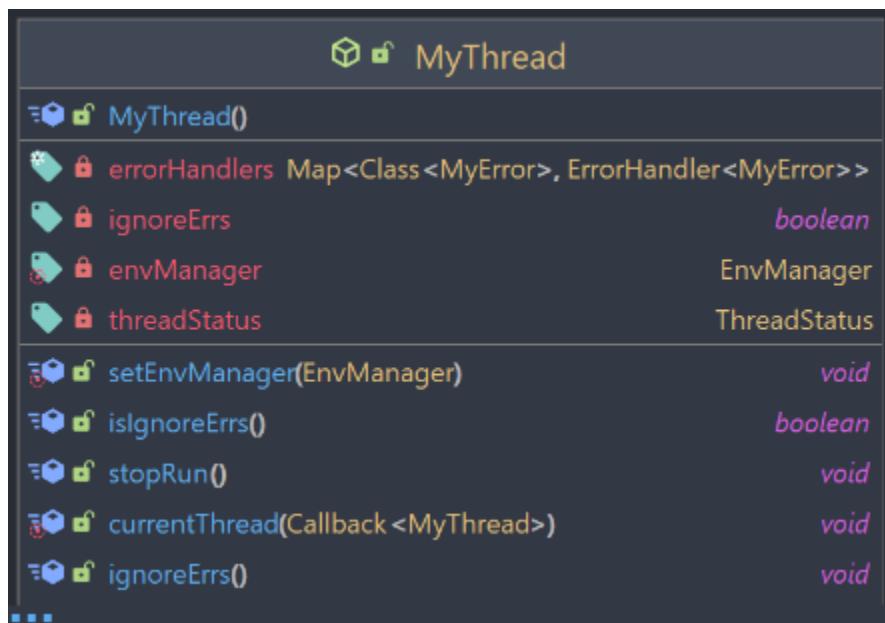
UserInfo	
<code>UserInfo(String)</code>	
<code>UserInfo(String, String)</code>	
<code>profilePic</code>	String
<code>id</code>	String
<code>ID()</code>	String

All Implemented Interfaces: Serializable, SyncableItem

```
public class UserInfo

implements SyncableItem, Serializable represents syncable User information.
```

את שאר הפעולות ניתן לראות בקוד עצמו

Threads**All Implemented Interfaces:** Runnable

```
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.

setEnvManager public static 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

stopRun public 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()**

throws Throwable run this thread in a handled manner.

Throws: Throwable - the error that might get thrown

את שאר הפעולות ניתן לראות בקוד עצמו



All Implemented Interfaces: Serializable, Comparable<MyThread.ThreadStatus>, Constable

Enclosing class: MyThread

```
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	
<code>HandledThread(ThrowingRunnable)</code>	
<code>HandledThread()</code>	
<code>Runnable</code>	ThrowingRunnable
<code>handledRun()</code>	<code>void</code>
<code>setRunnable(ThrowingRunnable)</code>	<code>void</code>
<code>runInHandledThread(ThrowingRunnable)</code>	HandledThread

All Implemented Interfaces: Runnable

```
public class HandledThread
```

extends ver14.SharedClasses.Threads.MyThread represents a thread that can handle errors.

HandledThread public `HandledThread(ThrowingRunnable runnable)` Instantiates a new Handled thread.

Parameters: runnable - the runnable

runInHandledThread public static 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



All Implemented Interfaces: Serializable

```
public class MyError
```

extends Error represents my implementation of an error.

MyError public `MyError(Throwable cause)` Instantiates a new error with a cause.

Parameters: cause - the cause

MyError public `MyError(String message)` Instantiates a new error with an error message.

Parameters: message - the error message

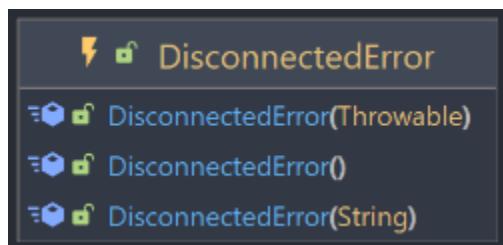
MyError public `MyError(String message, Throwable cause)` Instantiates a new error.

Parameters: message - the error message cause - the cause

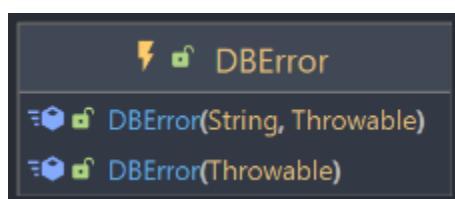
```
getShortDesc public String getShortDesc() Gets a short description of this error.
```

Returns: the short description of this error

את שאר הפעולות ניתן לראות בקוד עצמו

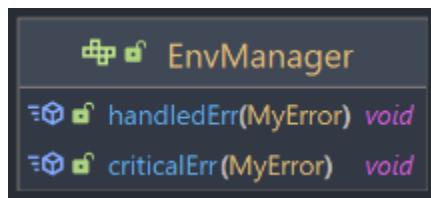


```
public class DisconnectedError
extends MyError represents a disconnection error.
```



All Implemented Interfaces: Serializable

```
public class DBError
extends MyError represents a database error.
```



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.

handledErr void **handledErr**(MyError err) notifies manager of a managed error

Parameters: err - the error thrown

criticalErr void **criticalErr**(MyError err) notifies manager of an un-handleable error. triggering a shutdown

Parameters: err - the error thrown

ErrorHandler	
ignore (ThrowingRunnable)	boolean
handle (MyError)	void

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

handle void **handle**(MyError err) handle an error that was thrown.

Parameters: err - the error thrown

ThrowingRunnable	
run ()	void

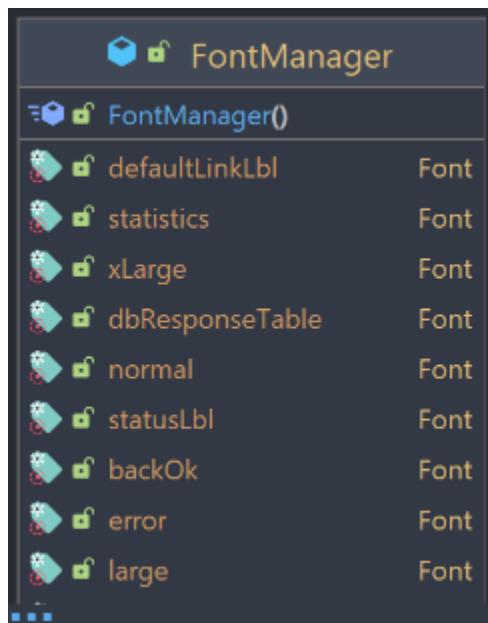
public interface **ThrowingRunnable** represents a runnable that might throw an error.

run void **run**()

```
throws Throwable Run.
```

Throws: Throwable - the throwable that might get thrown

UI



```
public class FontManager
```

represents a Font manager.



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants

```
public class GameView
```

extends JFrame represents a debugging window for viewing a game's status.

update public 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

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, SwingConstants

public class LinkLabel

extends ver14.SharedClasses.UI.MyLbl represents a link label.

את שאר הפעולות ניתן לראות בקובץ עצמו

MyJFrame	
<code>MyJFrame()</code>	
<code>onClose</code>	<code>Closing <?></code>
<code>resizeDelayInMs</code>	<code>int</code>
<code>myAdapter</code>	<code>MyAdapter</code>
<code>doXClick()</code>	<code>void</code>
<code>toggleFullscreen()</code>	<code>void</code>
<code>setOnExit(Closing <?>)</code>	<code>void</code>
<code>getMyAdapter()</code>	<code>MyAdapter</code>
<code>setOnResize(VoidCallback)</code>	<code>void</code>
<code>addBorderRec(Container)</code>	<code>void</code>
<code>...</code>	

All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants

```
public class MyJFrame

extends JFrame represents my implementation of a window.
```

toggleFullscreen public void `toggleFullscreen()` Toggle fullscreen.

setOnExit public void `setOnExit(Closing <?> onClose)` Sets on exit.

Parameters: onClose - the on close

addResizeEvent public static 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 בקובץ עצמו

Closing<T>	
header	String
icon	ImageIcon
title	String
tryClose()	void
show()	T
checkClosingVal(T)	boolean
closing(T)	void

Type Parameters: T - the input type

public interface **Closing<T>** represents a user input confirmation for closing a window.

tryClose default void **tryClose()** show the input dialog and if the input verifies, close the window.

show T show() Show the input dialog.

Returns: the input value

checkClosingVal boolean **checkClosingVal(T val)** check an input value supplied by the user.

Parameters: val - the value **Returns:** true if the window can close

closing 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.

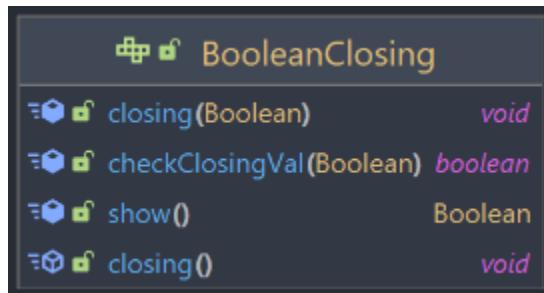
Specified by: show in interface Closing<String> **Returns:** the input

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

initialValue String **initialValue()** the initial value of the input field.

Returns: the initial value of the input field



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

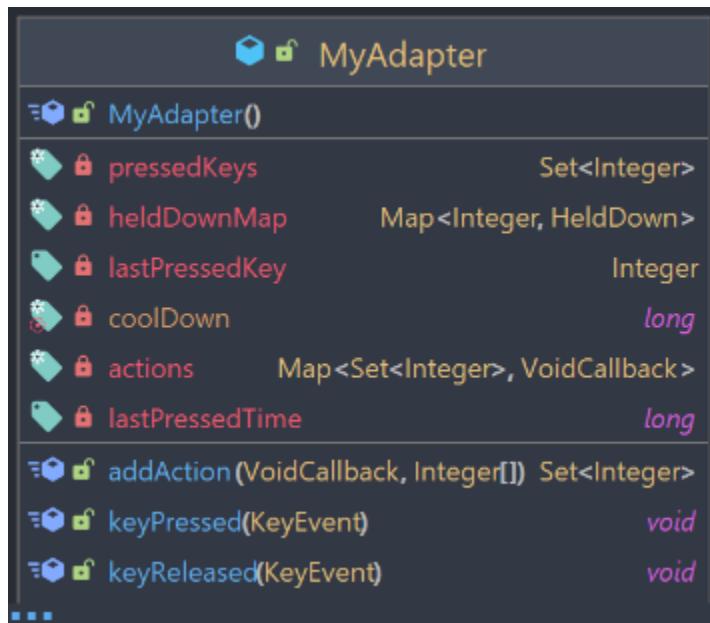
checkClosingVal default boolean `checkClosingVal(Boolean val)` **Description copied from interface:** `ver14.SharedClasses.UI.MyJframe.Closing` check an input value supplied by the user.

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:** `ver14.SharedClasses.UI.MyJframe.Closing` close the window.

Specified by: `closing` in interface `Closing<Boolean>` **Parameters:** `val` - the value the user submitted in the input dialog

closing void `closing()` the user selected 'yes' and the window should close



All Implemented Interfaces: `KeyListener, EventListener`

```
public class MyAdapter
```

extends `KeyAdapter` represents a key adapted

addHeldDown public void **addHeldDown** (MyAdapter.HeldDown heldDown) Add action on held down.

Parameters: heldDown - the action

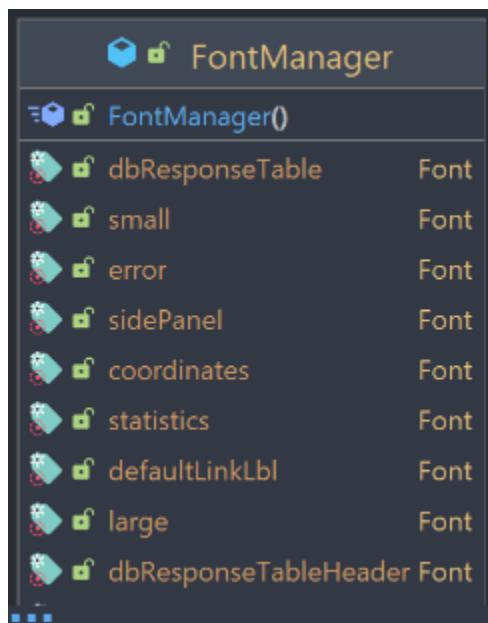
addAction public Set<Integer> **addAction**(VoidCallback action, Integer... keys)
Add action set.

Parameters: action - the action keys - the keys **Returns:** the set

removeAction public void **removeAction**(Set<Integer> action) Remove action.

Parameters: action - the action

את שאר הפעולות ניתן לראות בקובץ עצמו



```
public class FontManager
```

fonts manager



All Implemented Interfaces: ImageObserver, MenuContainer, Serializable, Accessible, SwingConstants

```
public class MyLbl  
  
extends JLabel represents a label.
```

את שאר הפעולות ניתן לראות בקוד עצמו

Utils

StrUtils	
StrUtils()	
htmlNewLines(String)	String
clean(String)	String
formatDate(String)	String
uppercase(String)	String
formatDate(Date)	String
createTimeStr(long)	String
dontCapWord(String)	String
fixHtml(String)	String
countMatches(String, String)	<i>int</i>
...	

```
public class StrUtils
```

utility class for `String` related utilities

isAbsoluteUrl public static 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

countMatches public static int `countMatches(String str, String regex)` Count number of matches in a string.

Parameters: str - the str **regex** - the **Returns:** the number of matches

isEmpty public static boolean `isEmpty(String str)` Is a string empty.

Parameters: str - the string **Returns:** true if the string is empty

strINN public static 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 בקוד עצמו

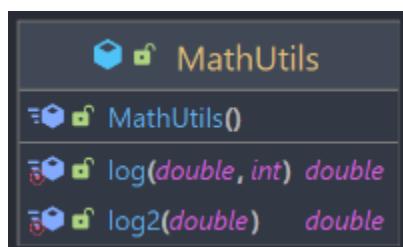


All Implemented Interfaces: Serializable

```
public class RegEx

implements Serializable Regex.
```

את שאר הפעולות ניתן לראות בקוד עצמו



```
public class MathUtils
```

Math utility class.

log2 public static double `log2` (double num) Log in base 2.

Parameters: num - the number **Returns:** the result

```
log public static double log(double num, int base) Log.
```

Parameters: num - the number base - the base **Returns:** the result



```
public class ArrUtils
```

Array Utility Class.

```
concat public static <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

```
createList public static <T> 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

```
exists public static <T> 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

create public static 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	
<code>OptionalArg(String)</code>	
str String	
getBoolean(Boolean) Boolean	
exists() boolean	
checkExists() void	
str0 String	
getString() String	
getInt() int	
getInt(int) int	

Enclosing class: [ArgsUtil](#)

```
public static record ArgsUtil.OptionalArg(String str)
```

extends Record represents an argument that might have been passed.

getString public String **getString()** Gets the string value of this argument.

Returns: the string value of the argument if it was found, null otherwise.

exists public boolean **exists()** was this argument found

Returns: true if the argument was found

את שאר הפעולות ניתן לראות בקוד 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.

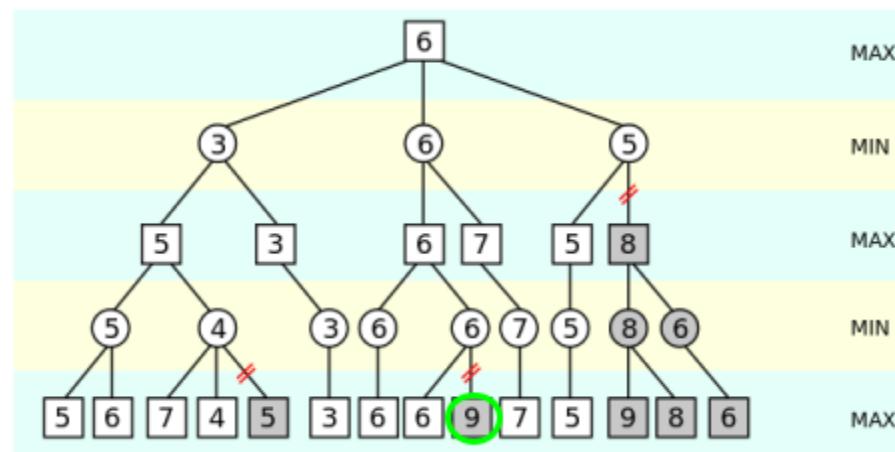
עבור עמדה נתונה, האלגוריתם עובר על כל המהלך האפשריים לשחקן הנוכחי, עושה אותם, ואז עושה את כל המהלך של היריב. בסוף משחק (תיקו, ניצחון או הפסד) הוא מחזיר ערך עבור העמדה הסופית. אותו ערך יהיה מאד גבוה עבור ניצחון, מאד נמוך עבור הפסד, וניטרלי עבור תיקו. ואז חזר בצורה רקורסיבית, ובוחר את המהלך שהשחקנים ישחקו בצורה אידיאלית עבור כל אחד. הרעיון הוא ש כדי למצוא את המהלך הכי טוב, חיברים להנich שהיריב ישחק בצורה מושלמת. וכך ניתן למצוא את המהלך הכי טוב בכל עמדה אפשרית.

גיזום

כדי להעריך את העמדות בצורה הכי מדויקת שאפשר, אנו רוצחים ליעיל את החיפוש ככל האפשר, ולהסוך בחיפושים שלא נצרכים. לשם כך ננסה לגלוות متى חיפוש עמדה מסוימת יהיה מיותר, ול"גוזם" את החיפוש זהה. זו לא לטרוח לחפש בענף זה כלל, וכך להסוך זמן יקר.

אלפה בטא

גיזום אלפה בטא הינו גיזום שמטרתו לחסוך חיפוש שככ"מ ייפסל כשייחסו מהركורסיה. לדוגמה:



כשהען הגיע ל-9 שסמן בירוק, ברור שהוא לא ייבחר. מאחר והוא יותר גדול מהערך המינימלי שנמצא עד עכשו. ובברור שהערך שיוחזר יהיה קטן יותר ל-6. וה-9 נמצא בשכבה מסוימת, ולכן הערך שיוחזר ממנו יהיה גדול יותר מ-9. ואפשר לפסול את החיפוש בעקבות זהה למחר.

באיזה מקום בפרויקט נעשה שימוש באלגוריתם

האלגוריתם ממומש בקוד הפרויקט ע"י המחלקה Minimax.java שנוערת באלגוריתם להערכת מצב (פונקציית הערכה שתוסבר בסעיף הבא) שמנדק לוח 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;
    }
    //the best evaluation found so far in this node
    Evaluation bestEval = null;
    ArrayList<Move> possibleMoves =
        MoveGenerator.generateMoves(parms.model, GenerationSettings.LEGALIZE);

    //at lower depths, sorting the moves is beneficial for the alpha-beta
    pruning
    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 שיווצאים מכל עמדת חוקית (הוא ממוצע 33 (במשחקים כמו שחמט שבהם מספר המהלים החוקיים עבר כל עמדה לא אחד, מחשבים ממוצע). מה שימושי למספר אסטרונומי של עמדות לבחון כבר בעומקם יחסית נמוכים. לדוגמה, אם השתמש ממוצע, בעומק 6 יש כבר יותר מיליארד עמדות לבחון. ועם כה המחשב שלנו היום, להמשיך לרדת בעומק עד שmagics לסופם משחק זה כמעט אף פעם לא אפשרי. וכך חיברים להגביל את החיפוש בצורה מסוימת.

היפosh המינימקס 'ווצר' עץ DFS (הhiposh לא באמת יוצר עץ, אלא שהוא עובד בצורה של צלילה שמאליה, ובחזרה הוא פונה ימינה. בדומה לעץ (DFS) אם הינו מניחים לאלגוריתם לפועל ככה, ומגבילים את החיפוש בזמן מסוים (כדי שהחיפוש גם יעיצר מתיישהו), הינו נתקלים בבעיה. בשל טبع החיפוש, העץ עלול היה להיריד רק עמדות שיצאו ממהלך אחד\כמה בצורה הרבה יותר עמוקה (ומזוקה), ולנוטש לגמרי\כמעט לגמרי עמדות שיצאו מהצד הימני של העץ. כל זה בגלל הנטיה שלו ללכט עד הסוף שמאליה, ורק בחזרה לפנות ימינה. והמצב שעולול להווצר הוא שבعود העץ עמוק שמאליה, נגמר לו הזמן. והוא חייב להחזיר תוצאה. * כדי לפתור בעיה זו, נפעיל את המינימקס בצורה איטרטטיבית. ז"א שקדם כל גריין אותו עד לעומק 1, הוא ירד שמאליה, יחזור, יעבור ימינה, ויחזר תוצאה. כתע, אם נשאר זמן, אפשר לשולח אותו לעומק 2, ירד שמאליה, יחזור, יעבור ימינה וכן הלאה. ככה שהוא לא יוכל להפלות את הענפים הימניים לרעה (המסימום שהוא יכול להפלות אותו לרעה יהיה בעומק 1, במקרה שבו הוא נאלץ לעוזר באמצעות החיפוש).

4.5.2 פונקציית הערך

הציגת האלגוריתם – מטרתו ואופן פעולה

כדי לנויל את המשחק, צריכה להיות דרך לזהות סיום משחק. וכשען המינימקס נבנה, כל מצב סופי מקבל ציון הערך כדי שאלגוריתם המינימקס יוכל לקבל החלטה. אם המצב הסופי הוא סיום משחק - ניצחון/הפסד/תיקו - אז הניקוד ברור וקל. אך אם המצב לא סיום משחק, יש צורך לנகד את המצב הזה המשקף עד כמה הוא טוב עבור השחקן שתורו לשחק. כמעט בשום עמדה לא יהיה ניתן לרדת עד לסוף העץ. בשל כלשהו חיברים לעוזר ולנסות "לשער" מה ניקוד העמדה. לשם כך אנו משתמשים במחלוקת Eval. שמחשבת ניקוד עבור עמדה מסוימת. לכל כל שיש ערך שנמדד ע"פ מידות centipawn (מאיות וגלי). הרגלי שווה 100, וכל שאר ערכי הchossingtons מונוקדים בהתאם למידה זו. ז"א שאם למשל עמדה מסוימת קיבלה ניקוד של 500, ניתן להסיק שהיתרון שיש לשחקן שווה ערך לעמדה שבה יש לו 5 רגליים יותר מיריבו.

לכל אחת מהמטריקות שבهم השתמשתי יש משקל (שעלול להיות שונה ממה שמצוין כאן בשל מיטוב הקוד), שקובע באיזה מידתם הם משפיעים על הציון הסופי. המטריקות:

- **מספר הכלים**
נתחיל במטריקה הבסיסית ביותר. ערך הכלים. גגלי שווה 100, צריח שווה בערך 5 רגליים, ולכון ערכו 500. פרש שווה קצת יותר מ-3 רגליים ולכון ערכו 310. רץ שווה קצת יותר מהפרש, ולכון ערכו 320. והמלכה שווה 900. אפשר לטעון שהמלך שווה לאינסוף, ולכון הוא יקבל ציון 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 הכללה, שלא יכולה להיות נcona בכ-100% מהפעמים, אך חייבים לעשות אותה. ובסך הכול הרעיון הוא שהמטריקות יcssו על ה"טיחים המתים" אחת על השניה). לשם כך חשבנו ציוניים עבור כל משבצת עבור כל כל, וכל הכלים של השחקנים משתמשים לפי ערך המשבצת שבה הם נמצאים. מאחר וכמה טובה משבצת מסוימת עבור כל מסוים משתנה מאוד לךראת, ובמיוחד במהלך, השלב הסופי של המשחק, ישנה טבלת ציוניים ייחודית שמותאמת לסוף המשחק, וכל שהוא מתקרב, המשקל שלה עולה.
משקל טבלאות הכלים הינו 1.1.
- **אילוץ המלך לפינה**
לקראת סוף המשחק, בד"כ, המלך "רוצה" להתקדם למרדף. כדי להיות יותר מועיל, כדי שייהיו לו הרבה מהלכים, ולא יכול להילך בקלות. לכן ככל שהמלך מתקדם לסופו, ישנו יותר ויותר משקל לאילוץ המלך של היריב לפינה. מאחר והוא בד"כ הרבה הרבה פחות פעיל כשהוא בפינה. ובנוסף על כן, בד"כ הרבה יותר קל לעשות מט למך שנמצא בפינה.
משקל אילוץ המלך לפינה הינו 1.5.

ישנן עוד מטריקות שנייסטי, אך כרגע אשר עם השלושה שציינתי.

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.getNumOfPieces(playerColor, PieceType.PAWN) ==
0 &&
        model.getNumOfPieces(playerColor,
PieceType.MINOR_PIECES) <= 1 &&
        model.getNumOfPieces(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));

evaluation.setPerspective(evaluationFor);

}

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);
}

```

ניתוח יעילות האלגוריתם

בדיקה האם נגמר המשחק:

כמויות המהלךים שהשחקן שטורו ל佐ו יכול לבצע = n

O(n)

אם לא נגמר המשחק:

בדיקת השח:

O(n)

ספרת כלים:

O(n)

טבלאות כלים:

O(n)

אילוץ המלך לפינה:

O(1)

4.5.3 אלגוריתם 3 חישוב אלטרנטיבות לשם משתמש תפוס

כשלוקה מנסה להירשם בתור שחון חדש, ובוחר שם משתמש שכבר קיים בשרת, רציתי לחשב כל מיני אלטרנטיבות זמינים לשם המשתמש שהוא ניסה לבחור ולהציג לו אותן.

שם כך נכתבת מחלוקת UsernameSuggestions. שמקבלת שם משתמש, ומנסה ליצור אלטרנטיבות לשם סבירו. תהליך ייצור האלטרנטיבות מתבצע בצורה זו: מהירות RegEx מוצאת תבנית, ובאמצעות מימוש המשק MatchIterations והפעולה createStr שמקבלת כפרמטר את המחרוזות שנמצאה, ואת מספר האיטרציה הנוכחית של ייצור אלטרנטיבות (יצירת האלטרנטיבות מתחכמת בלולאה לפי קבוע שקבע כמה אלטרנטיבות צrüכות להיווצר עבור כל אחד מיוצר האלטרנטיבות. לדוגמה: אם תבנית RegEx מוצאת מספר במחרוזת, ויוצר האלטרנטיבות מעלה אותו באחד. כדי לייצר קוד דינامي יותר, יוצר האלטרנטיבות לא יעלה את המספר באחד, אלא במספר האיטרציה הנוכחית. כדי שנוכל ליצור, לדוגמה עבור 0-test: גם את test-1 וגם את 2-test וכו'. במקומות רק את 1-test), וכן ניתן ליצור הרבה סוגים אלטרנטיבות בקלות.

אחד הקשיים באlgorigithm זהה היה חיפוש לרוחב של שמות משתמשים שלא נמצאים בשימוש, כדי להציג ללקוה רשימת אופציות מגוונת ככל האפשר. הייתה יכול לבדוק את כל האופציות ולקחת מלהט מגוון ככל האפשר מהאופציות הזמיןות, אבל לאחר ואני רוצה לשמר על מספר הקריאה למסד הנתונים למינימום האפשרי, אנסה למצוא את הקומבינציה הכי מגוונת שעונה על מספר האופציות הנדרשות עם מינימום קריאות למסד 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 <= numIterationsPerOptionGroup; 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(match.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 במחירות = n

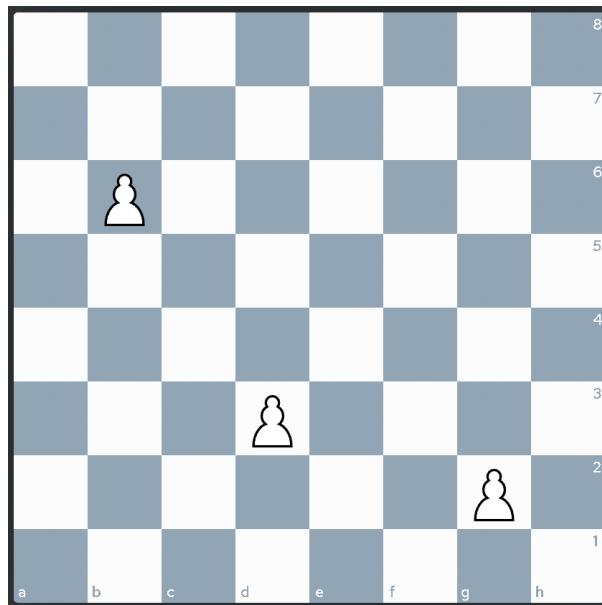
מספר הניסיונות שייצרו

הוא סטטי, ולכן זמן הריצה זהה לינארי, וזמן הריצה של בדיקת הניסיונות מול מסד הנתונים הינו: $O(n^2)$

4.5.4 אלגוריתם/פעולה 4 חישוב איזומים של كلمים

אחד האלגוריתמים החשובים המשחק שחמט, זו הבדיקה האם משਬצת מסוימת מאוימת. כדי לבדוק האם למלך מותר ללכת אליה, או האם שחקן נמצא בשח. כדי ליעיל את החישוב הזה השתמשתי בלוחות ביטים (אובייקט שמייצג את הלוח בצורה בינארית. האם יש או אין כלי עברו כל משਬצת). עברו כל סוג כלי והציבו שלו שמרתி לוח ביטי. ואז באמצעות היסט, שבעצם מזין את כל הלוח לפי אופסת שהוא כיוון תקיפה של אותו כלי, אני יכול לקבל את כל המשבצאות אותו סוג כלי תוקף. ואז בדיקה של & עם ייצוג ביטורדי של מקום של משבצת (מספר ארוך עם בית דלוק באינדקס המשבצת. המשבצת hei שמאלית למעליה (A8) מיצגת ע"י בית 0, והמשבצת hei ימנית למטה (H1) ע"י בית 63) תגיד לי האם אותה משבצת מאוימת ע"י אותו סוג כלי או לא. ככה שאני לא צריך לחשב את המשבצאות עבור כל כלי, אלא בקבוצות של סוגים של كلمים. גם החישוב נעשה בצורה הרבה יותר יעילה.

לדוגמא:



הלוּחַ הַבִּיטֵּי שֶׁל הַרְגָּלִים שֶׁל הַשְׁחָקָן הַלְּבָנָן שֶׁל הַלוּחַ הַזֶּה יְרָאֵה כֹּה:
00000000010000000000100
או בגרסת קצת יותר יידיתית לבני אדם:

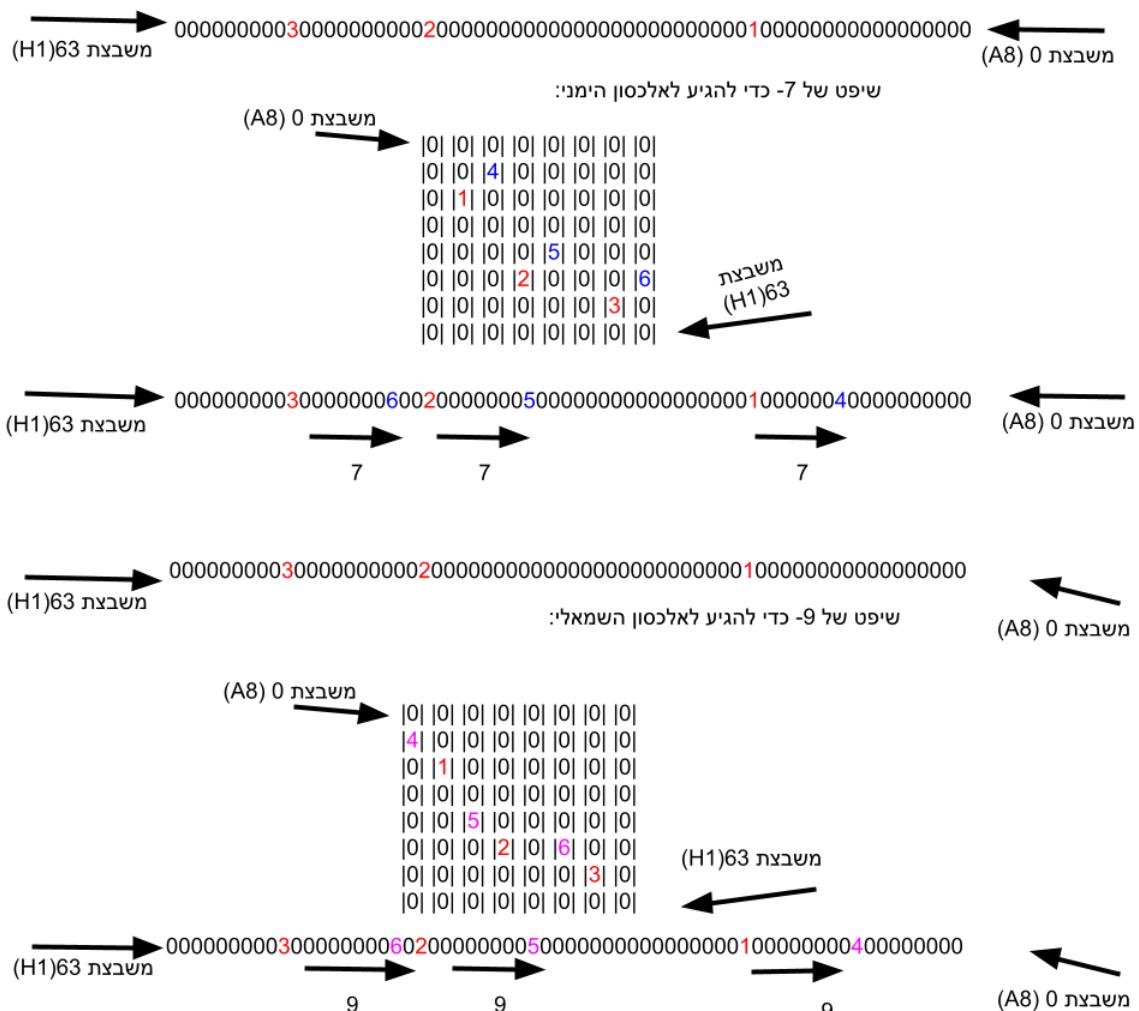
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

כדי לחשב את המשבצות אותן הרגלים תוקפים, צריך לעשות שיפט-ים בכיוונים שבהם רגליים תוקפים. הרעיון הוא שהתזוזה היא איחוד. חוץ ממקרי קצה (ליטרל) שבהם הכללי נמצא בקצתה הלוּחַ ולא יכול לזרום בכיוון ההיסטט, כלי שנמצא ב $3d$ עובד בדיקות כמו אותו סוג כלי שנמצא ב $4h$ או כל משבצת אחרת. היחידים שבהם זה שונה הם הרגלים והמלכינים. שלהם יש מהלכים מיוחדים לפי המיקום שלהם בלוּחַ (הצרחה, קידום

של רגלי לכלי אחר, צעד כפול, (En Passant). אך כל אותן מהלכים לא משנה את צורת האכילה של אותן كلم.

הצורה וצעד כפול: לא יכולה להיות אכילה. קידום וצד En Passant: לא משנה את צורת האכילה של הרגליים. הם עדין תוקפים את המשבצות ישר אלכסון מהם.

במקרה של הדוגמא, רגליים תוקפים ב2 צורות. האלכסון לימין ולשמאל. ולכן, תהליך החישוב יראה כך: (המספרים השונים נועדו לצורך הדוגמה. כמובן שבאמת 4,5 ו-6 הם אחדים. ו-1,2,3 הם נמצאים כלל. מאחר שהם נמצאים רק בלוח המקורי)



כל התוצאות נשמרות בתחום ביטבורד סופי. שבסקרה של הדוגמא הזו ייראה כך:

הציגת 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 החקף = n

מספר הכלionarioים שבהם כלי תוקף = y

עבור כל סוג כלי שיש לשחקן החקף, עבור כל כיוון שבו הוא תוקף, נעשה מקסימום 64 שיפטים כדי לחשב

את המשਬצות שהוא תוקף באותו כיוון.

לכן האלגוריתם פועל ביעילות של:

$O(n * y * 64)$



$O(n * y)$

4.5.5 אלגוריתם/פועלה AppSocket 5

את התקשרות בין השרת ללקוח מימושי בצורה הבא:

המחלקה AppSocket מייצגת תהליך שכל הזמן קורא הודעות. ומאפשרת שליחת בקשות (Request)

ולמקרה שבו מצופה לקבלת תגובה (Response) מהצד השני, מועבר גם Callback שאליו נקרא כשהודעת

תגובה תקרה. או לחלופין, במקרים מסוימים, התהליך ירצה לחסום עד לקבלת תגובה. וכך יש אופציה

שלוחה 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 מהמשמעותים של `MessagesHandler`. לדוגמה: במקרה שבו השרת נתקל בשגיאה לא צפוייה, הוא ישלח הודעה מסוג שגיאיה, ואotta הודעה תנותב לצד הלוקה למטפל של ה הודעות מסוג שגיאיה. שתציג את ה הודעת השגיאה ותשגור את הלוקה. אך במקרים שבהם שגיאיה היא תגובה צפוייה, לדוגמה: כשהЛОקוח מנסה להיכנס עם משתמש רשום, מועבר מטפל偏偏 עם ה הודעת ה כניסה, והטיפול בשגיאות (או בחיבור מוצלח) יבוצע בעורתו.

הצגת 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);
}

```

נתוח יעילות האלגוריתם

האלגוריתם פועל בזמן לינארי.

4.6 מבני נתונים עיקריים

מבנה הנתונים העיקריים שבהם השתמשתי הם:

- **ArrayList**

משמש לאחסון רשימה אובייקטיבים, וגישה אליהם ע"י אינדקס בזמן לינארי.

השתמשתי בו בפרויקט:

לאחסון שחקנים מחוברים, משחקים שרצו, מהלכים ששחקן יכול לבצע, ועוד.

- **HashMap**

משמש לגישה בזמן לינארי לאובייקטיבים שנשמרו ע"פ מפתח מסוים. השתמשתי בו מטעמי נוחות ויעילות.

אחד מהשימושים היה לאחסון AppSocket-callbackים ב-AppSocket callback כדי לנתר כל הודעה למטפל שלא עבר סוגיה הودעות שונות. דבר שהוסך זמן תקורה קרייטי לעיבוד הודעות.

- **מערכיים**

השתמשתי בהם בפרויקט כדי לאחסן אוספים בסדר כלשהו. לדוג' אחסון לוחות Bitboard-ים של הכלים נעשה בתוך מערך שסודר ע"פ סוג הכלים (האינדקס שלהם).

• **מטריצות**

השתמשתי בפרויקט לאחסן טבלאות כלים כמו שתואר ב4.5.2 פונקציית הערכה. או לאחסן מספר כלים לפי כלי ושהקן. השתמשתי בהם כדי לשמר מידע על עמדה (שורות&עמודות) ולגשש אליהם בצורה יעילה.

• **Bitboard**

ייצוג לוח שחמט בצורה בינארית. אם יש כלי על אותה משכצת או לא. השתמשתי בו כדי לייצג את הלוח במקרים סוג הכלי על אותה משכצת ידוע, או לא משנה. תיארתי את השימוש העיקרי בו ב4.5.4 אלגוריתם/פעולה 4 חישוב איזומים של כלים.

5. סיכום אישי

5.1 אתגרים וקשיים איתם התמודדתי

• **ניהול זמן**

בזמן עבודהתי על הפרויקט, דבר שקרה המון היה שהשකעתי המון זמן בפרטים שלולים מאוד, והזנחה חלקים הרבה יותר משמעותיים. כדי לפתור את זה השתדלתי לעזור את מה שאני עושה אחת לכמה זמן, ולשאול את עצמי האם אני מנצל את הזמן שלי בצורה יעילה.

• **אובססיה לגבי לעשות משהו בצורה 'מושלמת'**

כשניגשתי לפתור איזו בעיה או למשתמש באיזה אלגוריתם, המחשבה שאני חייב לעשות את זה בצורה מושלמת כל הזמן חזרה לי בראש. מה שמצד אחד הוא דבר די חביבי, אבל מצד שני יכול לגרום לבגיאות ביכולת שלי בכלל לגשת לעביה. כי אני אהיה עסוק מדי בלבנותו לפתור אותה בדרך "המושלמת", או לנסות לחשב בכלל על מה הוא הפתרון המושלם. מה שבמסופו של דבר גורם לבזבוז זמן ותסכול. כדי להתגבר על זה, אני משתמש לעיתים קודם כל מהצורה הכى פשוטה שלהם, ולהגיע לפתרון קיים. ורק אחר כך לחשב על דרכם אולי למטר את הפתרון הראשוני.

• **KISS**

ראשי תיבות של **keep it simple stupid** (יש שאומרים **keep it super simple**). ביטוי שנועד להזכיר למפתחים לשמור על דברים בצורה פשוטה, בלי לסביר אותם לשוווא. ולבטל כל מרכיבות

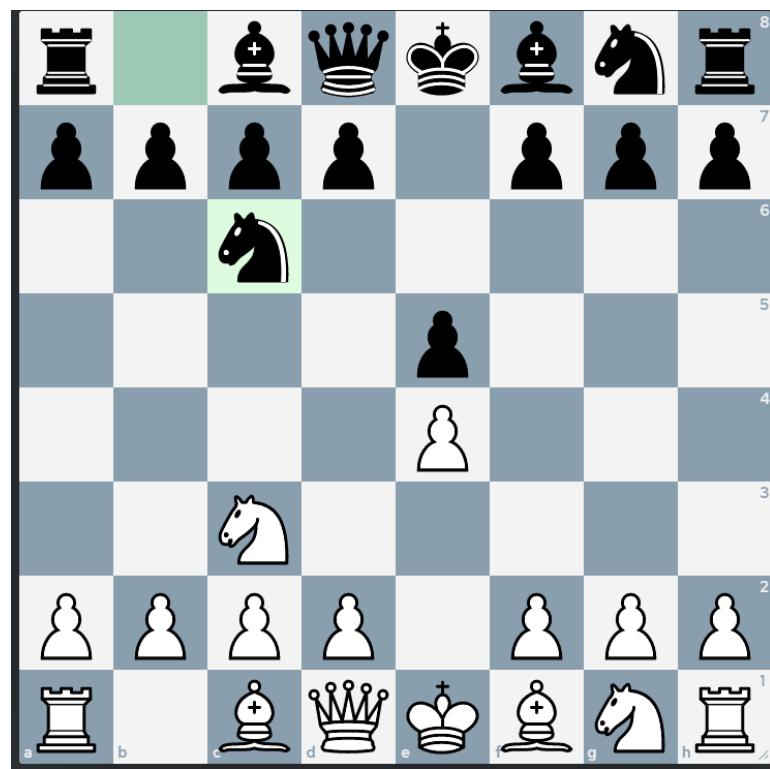
שאינה חיובית. זה דבר שકצת התקשית לעמוד בו במהלך כתיבת הפרויקט (מספר העמודים בספר עלול לرمוז על כך) בהסתכלות אחרת, זה נראה כאילו הנטיה הראשונית שלי היא בדיקת הפה. לבסוף כמה שאפשר. ככה שיצא לפעמים שמגלי לשים לב, יצרתי איזו מערכת מורכבת ומסועפת כדי לפתור בעיה פשוטה. כדי להתגבר על זה, השתדלתי להכricht את עצמי לעזרה לרגע לפני אני מתחילה לעבוד, ולשאול את עצמי האם חשבתי על הדרכך הפוכה שבה אני יכול לעשות את זה.

5.2 מה הפרויקט תרם לי

- כתיבת טסיטים לקוד כדי לוודא את פועלתו, ולוודא שהדרישות נשמרו לאחר כל שינוי.
- כתיבת קוד גנרי נככל האפשר כדי לעבוד בצורה דינמית, ולמנוע חזה על קוד.
- כתיבת תיעוד לקוד בזמן הכתיבה כדי לעזור בקריאות של הקוד, ובמציאת טעויות עיצוב ולתקן אותן בזמן הכתיבה, במקום הרבה מאוחר יותר.
- שימוש בeval כדי לעקוב אחר גרסאות שונות של הקוד, ולהזור בזמן בכל רגע נדרש.

5.3 הצעות לשיפור

- **שיפור Eval:** Eval די בסיסי, וمبוסס על מטריקות יחסית פשוטות. עיקר היתרון של המחשב הוא בחישוב טקטיות שבבוססות על הובלה בכלים (מספר מהלכים שזוכים ברגלי, כלי, או דברים בסגנון), ולא בשחק עמדתי (השגת עמדה טובה). מה שיכول לשים אותו בעמדה שמננה יהיה קשה לו להתואוש. לכן הייתי רוצה להשתמש בעוד מספר מטריקות שישפרו את הערכת העמדות של המחשב.
- **שיפור maxminimax באמצעות טרנספוזיציות:** במהלך החיפוש בעץ המינימקס, ישנן הרבה עמדות שימושיות לאחר סדר מהלכים שונה. לדוגמה: לעמדה הוו



אפשר להגיע לאחר המהלךים: רגלי ל-E4, רגלי ל-E5, פרש ל-C3 ופרש ל-C6. לאוთה עמדה בדיקת אפשר להגיע גם לאחר המהלךים: פרש ל-C3, פרש ל-C6, רגלי ל-E4 ורגלי ל-E5. לאחר שאותה עמדה הושגה בחיפוש בפעם השנייה, המשך החיפוש באותו ענף יהיה בזבוז של זמן. מאחר וכבר חישבנו את תוצאה הענף הזה. כדי לחסוך בחיפוש המיותר הזה, הייתי רוצה להשתמש בטבלת טרנספוזיציות. שומרת מידע על עמדות שבהם חישבנו כבר. דבר שיכל ליעיל את החיפוש משמעותית.

5.4 מסקנות

בשנה וחצי שבהם עבדנו על הפרויקט, בנוסף לשיעורים, השקעתינו בו את רוב הזמן הפנו אליו. כך שיצא פרויקט די גדול (כנאמם.. סליה על זה), והרחבתי את הידע שלי אל מהוון לתהום הנלמן. עוד דבר שיצא לי לעשות ממנו לא מעט בשנה וחצי האחרון, היה לעזור לחבריי לכיתה בהבנה של התהומות הנלמדים. ואני חשב שיש לנו חלק גדול בהבנה שלי בחומר. כי כשאתה צריך להסביר משהו למשהו, נדרש הבנה מסוימת בחומר. ואני רוצה לחשב שעמדתי בדרישה זו רוב הזמן. החלק הכי מספק בפרויקט זהה היא ההרגשה שאני מסוגל להסביר כל פיסת קוד בפרויקט. אני לא אומר שהוא תהיה מושלמת, רחוק מכך, אבל אני יודע בדיקת מה הייתה מטרתה.

6. תודות

ראשית כל, אני רוצה להודות למורה אילן פרץ, 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 שמח לעזור. אפילו שהפרויקט לא היה בתחום אחוריותו. למורה אסף בנימין, שהתמודד עם הנוכחות המציקה שלי בשיעורים, ולא יותר לנו עד שנבין את החומר. לחבריו בחדר ולכיתה, על התמיכה לאורך השנהים.

7.ביבליוגרפיה

- chessprogramming.org שם למדתי הרבה. בעיקר על מימוש המשחק, והערכת עמדות שלילה הרחבותי ב4.5.2. פונקציית הערכה.
- [youtube.com/watch?v=U4ogK0MIzqk](https://www.youtube.com/watch?v=U4ogK0MIzqk) אחת הסיבות העיקריות שבחרתי בשחמט הייתה הסרטון הזה, והרבה חלקים מהפרויקט נבנו בהשראתו.
- [youtube.com/watch?v=l-hh51ncgDI](https://www.youtube.com/watch?v=l-hh51ncgDI) הסרטון שעזר לי להבין איך עובד גיזום אלף בטא.
- cutt.ly/pHiD1Uv נוסחא לחישוב כמה טוב שחקן כדי ליצר סטטיסטיות.
- currentmillis.com ייצוג תאריכים ושעות במסד הנתונים לפי unix timestamp.
- stockfishchess.org בינה מלאכותית בקוד פתוח שמנתה למדתי דרכם לשיפור יעלות הקוד, ובנוסף, לאחר זה קוד פתוח, יכולתי להוריד את כל הבינה המלאכותית, ולהציג אותה בתור אופציה לשחק נגדה.
- כדי לתקשר עם Stockfish השתמשתי בAPI זהה. cutt.ly/aHiDKjb כללי המשחק בעברית.
- cutt.ly/oG26V2Q השתמשתי בגרסה מעט שונה מזו כדי להציג תוצאות שאלתה מסוד הנתונים.