תכנות המשחק רברסי - חלק א׳-1

המשחק רברסי הוא משחק אסטרטגיה שמשוחק על לוח בגודל 8x8 המיועד לשני משתתפים. לכל אחד מהמשתתפים דיסקיות בצבע שחור או לבן. המשחק נפתח כשלכל שחקן יש שתי דיסקיות במרכז הלוח המסודרות בהצלבה. כל שחקן בתורו מניח דיסקית משלו על הלוח במקום אשר לוכד לפחות דיסקית אחת של היריב. כתוצאה מכך כל דיסקיות היריב אשר נמצאות בין שתי דיסקיות של השחקן שהניח את הדיסקית משנות את צבען להיות בצבע שלו. המשחק מסתיים כאשר לאף אחד מהשחקנים אין מהלך לשחק. אם לאחד השחקנים אין תור לשחק (חובה ״ללכוד״ לפחות דיסקית אחת של היריב) אז המשחק עובר ליריב. המנצח הוא מי שבסיום המשחק יש יותר דיסקיות שלו על הלוח.

צפו בוידאו הקצר עם ההוראות ונסו לשחק במשחק:

1. הוראות המשחק
2. אתר שבו ניתן לשחק את המשחק
3. ניתן לקרוא עוד על המשחק בויקיפדיה בעברית או אנגלית

מימוש המשחק:

לוח המשחק ייוצג על ידי מערך דו-מימדי של מספרים וערכי התאים ייצגו את מצב התא:

1. הערך 0 - תא ריק
2. הערך 1 - יש בתא דיסקית של שחקן 1 (דיסקית לבנה)
3. הערך 2 - יש בתא דיסקית של שחקן 2 (דיסקית שחורה)

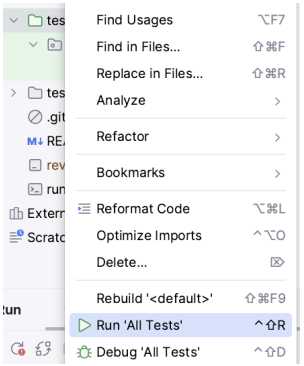
נתונות מחלקות שנשתמש בהן למימוש המשחק רברסי. לאחר שנממש את המשחק נכתוב גם ״בוטים״ שישחקו אותו.
הגישה אל הקוד וכל קבצי המשימה מתבצעת מהמשימה בgithub classroom. לינק ישירות למשימה נמצא במודל והוראות התחלת עבודה עם github נמצאות במודל תחת ״כללי״ וגם כאן.
כל המחלקות שאתם מממשים צריכות להיות בחבילה (package) שנקראת reversi.
שתי המחלקות שבהן נתמקד בחלק א׳:

1. המחלקה ReversiGame - מחלקת המשחק. מייצגת את המשחק רברסי ומכילה את הלוגיקה לניהול המשחק וביצוע המהלכים.
2. ReversiMain - המחלקה הראשית, בה נמצאת הפעולה הראשית אשר מאתחלת את הלוח, קוראת מהמשתמשים מהלכים ומדפיסה את הלוח.

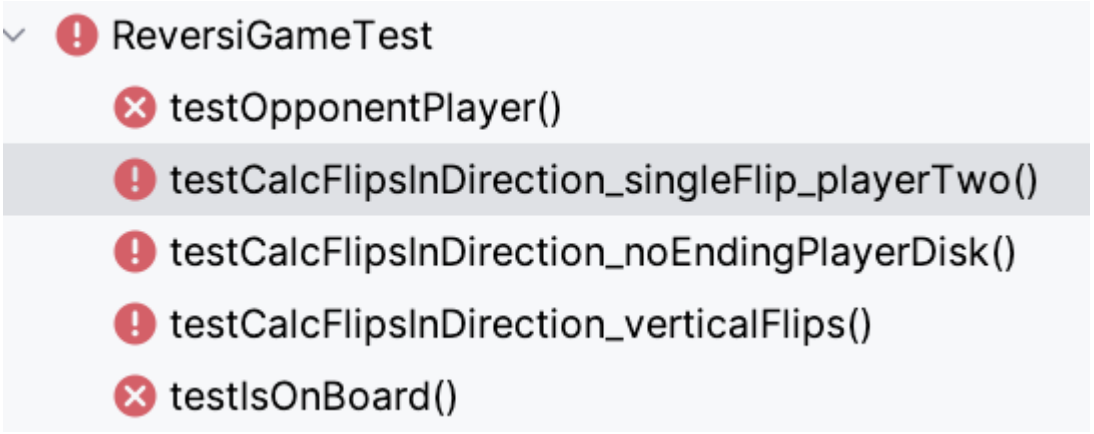
אם אתם עדיין לא מבינים לגמרי מה עושה כל מחלקה זה בסדר גמור, אל תדאגו אתם הולכים לממש אותן ודרך זה גם להבין את התפקיד והשימוש שלהן.

הרצת בדיקות

בשביל להריץ את הבדיקות לוחצים קליק ימני על הpackage שנקראת test ובוחרים Run 'All Tests'



שם הבדיקות תואם את שם הפעולה. למשל שם הבדיקות בעבור הפעולה calcFlipsInDirection מתחילות בtestCalcFlipsInDirection.



יודעים אם הבדיקה עברה על פי הצבע. אדום ← נכשל, ירוק ← עבר.

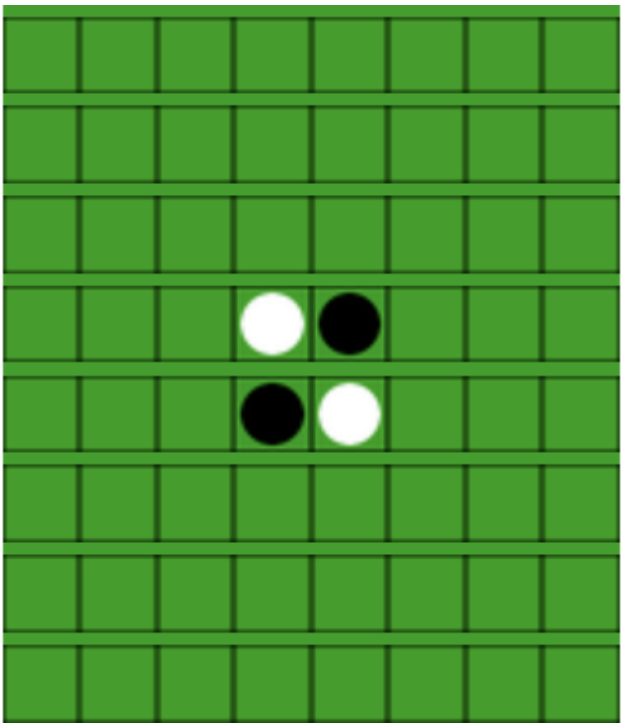
שימו לב: חלקים ב1 וב2 הם גם חלק מההגשה ומהציון אבל הם יבדקו ידנית ולא ע״י בדיקות אוטומטיות (יכול להיות שתקבלו 100 בבדיקות אבל 80 סה״כ בגלל טעויות בחלקים האלה או בגלל שלא פתרתם אותם)

למחלקה ReversiGame שני שדות

1. לוח המשחק: int[][] board
2. השחקן הנוכחי - int curPlayer אשר מקבל ערכים 1 (לבן) או 2 (שחור)

משימה 1: מימוש הבנאי והפעולה המאחזרת של curPlayer

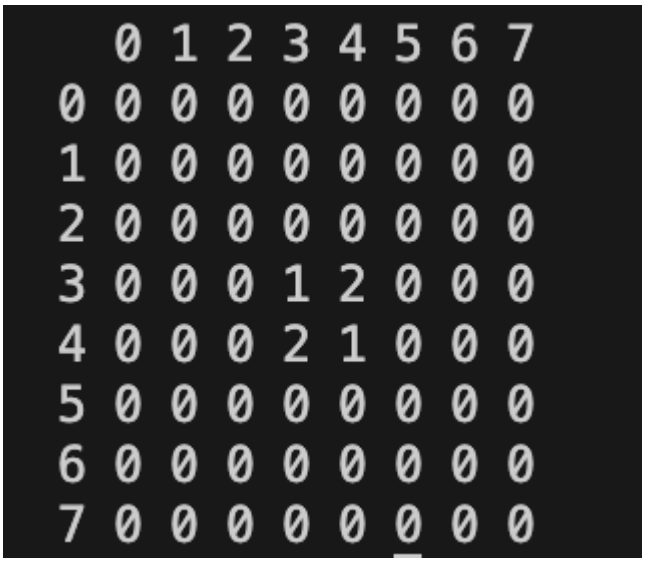
הבנאי מייצר את לוח המשחק ומאתחל את השחקן הלבן להיות השחקן שמשחק את התור הבא.
כמו כן הבנאי יאתחל את הלוח למצב ההתחלתי:



בנוסף למימוש הבנאי עליכם גם לממש את הפעולה המאחזרת בעבור התכונה curPlayer (הפעולה קיימת, יש רק צורך לתקן את הקוד שלה).

משימה 2: הוספת הפעולה printBoard המדפיסה את הלוח לConsole.

נדפיס את הלוח כך שמספרי השורות יודפסו בצד שמאל (מתחילים מ0) ומספרי העמודות למעלה(מתחילים מ0)
למשל לאחר אתחול המחלקה למצב ההתחלתי המופיע במשימה 1 וקריאה לprintBoard יודפס הפלט הבא:



משימה 3: מימוש פעולות העזר isOnBoard וopponentPlayer

במשימה זו נממש שתי פעולות עזר:

1. הפעולה isOnBoard המקבלת שורה ועמודה ומחזירה אמת אם המיקום הינו על לוח המשחק. איו צורך לבדוק ערך בתא, כל מה שמעניין אותנו הוא האם המיקום הזה הוא מיקום חוקי על לוח המשחק. למשל המיקום שורה: 1- ועמודה: 3 הוא לא חוקי וגם שורה 2 ועמודה 8 לא חוקי.
2. הפעולה opponentPlayer המקבלת שחקן (1 או 2) ומחזירה את השחקן היריב (2 בעבור 1 ו1 בעבור 2).

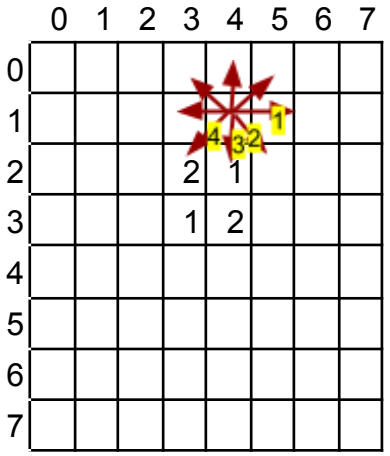
תכנות המשחק רברסי - חלק א׳- 2

בחלק זה נממש את יסודות המשחק רברסי: ביצוע מהלך, סיום המשחק וקביעת המנצח.

משימה 4: מימוש עדכון דיסקיות, אך רק בכיוון יחיד

הפעולות שנממש במשימה הזאת יעזרו לנו לממש הנחת דיסקית במשימה הבאה.

על פי חוקי המשחק, בעקבות הוספה חוקית של דיסקית ללוח המשחק, הופכות את כל הדיסקיות של היריב לדיסקיות של השחקן, במידה והן נמצאות על ישר כלשהו (אפקי אנכי או אלכסוני) בין הדיסקית החדשה לדיסקית אחרת של השחקן.



ישנם שמונה כיוונים שצריך להתייחס אליהם (תרשים למעלה). אנחנו נייצג את שמונת הכיוונים ע״י שימוש בשני מספרים: גדילת ערך השורה וגדילת ערך העמודה בכיוון. למשל:

1. בעבור החץ ימינה (1) השורות לא ישתנו והעמודות יגדלו ב1 ועל כן נייצג אותו כ 0,1
2. בעבור האלכסון מטה (2) השורות יגדלו ב1 והעמודות יגדלו ב1 ועל כן ניתן לקרוא לו 1,1
3. בעבור החץ מטה (3) השורות יגדלו ב1 והעמודות לא ישתנו ועל כן ניתן לקרוא לו 1,0
4. בעבור האלכסון מטה ושמאלה (4) השורות יגדלו ב1 והעמודות יקטנו ב1 ועל כן ניתן לקרוא לו 1,-1

אנחנו נטפל בכל אחד מהכיוונים האלה בנפרד ולאחר מכן במשימה הבאה נכתוב פעולה שתטפל בכל הכיוונים למימוש משחק תור.

מימוש הפעולה: **calcFlipsInDirection**
הפעולה תקבל מספר שחקן ושורה ועמודה שבה השחקן שוקל להניח דיסקית ושני מספרים שמייצגים כיוון אחד מ8 הכיוונים שהזכרנו למעלה. הפעולה תחזיר את כמות דיסקיות היריב שישנו את צבען כתוצאה מהנחת הדיסקית ו0 אם אף דיסקית לא שינתה את צבעה / המהלך לא חוקי.
למשל בעבור הערכים row: 1, col: 4, rowInc: 1, colInc:0 (חץ 3) בתרשים לעיל אם תורו של השחקן השחור (2), היא תחזיר 1 כיוון שכתוצאה מהמהלך בכיוון הזה דיסקית אחת תהפוך להיות שחורה (הדיסקית בשורה 2 עמודה 4).
אם למשל היא תקרא בעבור אותו שחקן (2) ואותה המשבצת עם הערכים rowInc: 1, colInc:-1 (חץ 4) היא תחזיר את הערך 0, כיוון שלא תהפך אף דיסקית בכיוון הזה. שימו לב שבכדי שיהפכו

דיסקיות שצריך לוודא שבכיוון שאנחנו בודקים יש: 1) רצף, לפחות באורך 1, של דיסקיות יריב בצמוד לנקודת הבדיקה ואחריה 2) דיסקית של השחקן שמשחק - זה יבטיח ש״נלכוד״ דיסקיות יריב.

למשל בעבור הלוח הבא, כאשר תור שחקן 1, קריאה עם מיקום שורה 4 ועמודה 2 (מודגש ומסומן על הלוח) וכיוון 0 (שורות) ו 1(עמודות) - כיוון ימינה. יחזיר את הערך 2 כיוון ששני התאים המודגשים יוחלפו ב1. שימו לב שכיוון שאנחנו מסתכלים על כיוון יחיד אנחנו לא סופרים את ההחלפה שתתבצע ב3, 3 בכיוון האלכסון ימינה ולמעלה (כיוון: 1, 1-).

	0	1	2	3	4	5	6	7
0								
1				1		2		
2				2	1	2	1	
3				2	2			
4			*1*	**2**	**2**	1		
5								
6								
7								

אם באמת נשים שם דיסקית הלוח יראה ככה:

	0	1	2	3	4	5	6	7
0								
1				1		2		
2				2	1	2	1	
3				1	2			
4			*1*	1	1	1		
5								
6								

<u>מימוש הפעולה updateMoveDisksInSingleDirection:</u>

בדומה לפעולה הקודמת רק שהפעם גם נעדכן את הלוח ונבצע את ההחלפות **בעבור כיוון אחד**. על כן בעבור הלוח הקודם התאים עם המספרים המודגשים גם יעודכנו לערך 1. אם המהלך בוצע בהצלחה והפך דיסקיות מספר דיסקיות היריב שהשתנו יוחזר. שימו לב: בקריאה הזו אנחנו לא מעדכנים את התא עם הדיסקית שהונחה (4,2 בדוגמה למעלה) אלא רק את דיסקיות היריב שנהפכו. חובה להעזר בפעולה calcFlipsInDirection - אם ישנן דיסקיות בכיוון הזה שעלינו להפוך נעבור עליהן ונעדכן אותן. חישבו כיצד לכתוב לולאה שתעבור על המקומות בכיוון המבוקש ותבדוק שיש

רצף של דיסקיות יריב שמסתיים בדיסקית שלנו (יש להזהר לא לצאת מהלוח במהלך הבדיקה - מומלץ להעזר בפעולה isOnBoard שכתבנו בסעיף קודם).

משימה 5: מימוש הנחת דיסקית ע״ שחקן!

כעת נממש את הפעולה placeDisk אשר מקבלת שורה ועמודה ומשחקת תור ע״י השמת דיסקית בשורה והעמודה ע״ השחקן שתורו לשחק ולאחר מכן, באם המהלך בוצע בהצלחה, עדכון התור לתורו של היריב. חובה לבדוק שהמהלך הינו חוקי (לפחות דיסקית אחת השתנתה כתוצאה מהנחת הדיסקית) לפני ש״מניחים את הדיסקית״ על הלוח.
הפעולה תחזיר אמת אם המהלך בוצע בהצלחה (הוא חוקי) ושקר אחרת.
שימו לב יש להעזר בפעולה updateMoveDisksInSingleDirection בעבור כל אחד מ8 הכיוונים האפשריים (ניתן לקרוא לה בעבור כל כיוון אפשרי. אם אין מה לעדכן, לא יקרה כלום).

משימה 6: מימוש מהלך המשחק

נסיים את החלק הראשון של מימוש רברסי בכתיבת התוכנית הראשית שתאפשר לשני שחקנים לשחק אחד נגד השני. במחלקה ReversiMain ממשו פעולה ראשית אשר תריץ משחק שלם. בכל שלב נבקש מהשחקן הנוכחי להקליד מספר שורה ועמודה (למשל: ״Player 2 please enter your row and column״) אם המהלך אינו חוקי נודיע למשתמש ונבקש להקליד שוב שורה ועמודה - נשתמש במסננת קלט לצורך המימוש (״Illegal move! Player 2 please enter your row and column״). לאחר ביצוע התור נעביר את התור למשתמש השני וחוזר חלילה. שימו לב שבשלב הזה המחשב עדיין לא יודע בעצמו לבצע בדיקה אם המשחק הסתיים ולא מכריז על מנצח. אלה פעולות שנוסיף בחלק הבא. בינתיים מוטל על השחקנים לזהות את סוף המשחק.

בשלב הזה מומלץ לשחק כמה משחקים בשביל לראות שהכל עובד כמצופה

תכנות המשחק רברסי - חלק ב׳1. משחק נגד המחשב!

בחלק הראשון נממש את הפעולות הדרושות לצורך מימוש משחק ע״י המחשב.

משימה 7: חישוב ״היפוכים״ בעבור מהלך

ממשו את הפעולה calcMoveFlips אשר בהנתן מהלך מסויים מחזירה את מספר הדיסקיות יריב שיתהפכו כתוצאה מהמהלך. לדוגמה בעבור הלוח הבא והנחת הדיסקית המסומנת עם קו תחתי, הפעולה תחזיר 3 כיוון ש3 דיסקיות יריב יחליפו את צבען (המשבצות עם המספר 2 המודגש יהפכו ל1). חובה להיעזר בפעולה calcFlipsInDirection שמימשנו במשימה קודמת.
אם המהלך איננו חוקי הפעולה תחזיר 0.

	0	1	2	3	4	5	6	7
0								
1				1		2		
2				2	1	2	1	
3				**2**	2			
4			*1*	**2**	**2**	1		
5								
6								
7								

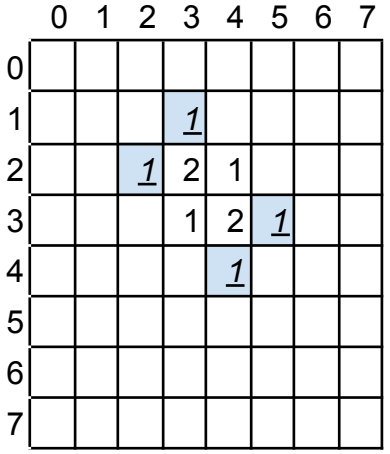
משימה 8: חישוב כל המהלכים האפשריים ו״היפוכי הדיסקיות״ בעבורם

נתונה המחלקה MoveScore המייצגת מהלך ואת הציון של המהלך. למחלקה שלוש תכונות:

1. row - שורה שבה תונח הדיסקית במהלך
2. column - עמודה שבה תונח הדיסקית במהלך
3. Score - ציון של המהלך. בשלב הזה ציון הוא תמיד מספר הדיסקיות שהמהלך גורם להיפוך שלהן. בהמשך כאשר לכתוב בוטים, ניתן ״ציון״ לכל מהלך (נרחיב על כך בהמשך).

למשל בעבור הדוגמה מהסעיף הקודם המהלך יכול להיות מיוצג על ידי מופע של MoveScore עם ערכי השדות: שורה: 4, עמודה: 2 וציון: 3 (מספר ההיפוכים כתוצאה מהמהלך).

נממש את הפעולה **getPossibleMoves** אשר מקבלת מספר שחקן (1 או 2) ומחשבת ומחזירה מערך ובו כל המהלכים האפשריים בעבור הלוח הנוכחי ובעבור השחקן הנתון. כל מהלך ייוצג ע״י מופע של המחלקה MoveScore. ועל כן ערך החזרה של המחלקה יהיה מערך של MoveScore - ניתן לראות את החתימה במחלקה. שימו לב המערך מכיל **רק** מהלכים חוקיים ולא אמור להכיל ערכי null. זאת אומרת, שאם לדוגמה יש שלושה מהלכים חוקיים גודל המערך המחוזר צריך להיות 3.
לדוגמה בעבור מצב הלוח ההתחלתי:



בעבור השחקן הלבן (1) ישנם ארבעה מהלכים אפשריים (מסומנים בתכלת עם קו תחתי בלוח) על כן קריאה לפעולה עם הערך 1 תחזיר **מערך** בגודל 4 ובו **ארבעה מופעים** של המחלקה MoveScore (אחד בעבור כל מהלך אפשרי) והניקוד של כל מהלך הינו 1 כיוון שבעבור כל אחד מהמלכים דיסקית אחת של היריב תשנה את צבעה - פירוט המהלכים (הרישום נועד רק לפרט את טיפוס המחלקה וערכי השדות והוא אינו מהווה קוד חוקי בשפת ג'אווה).

[MoveScore{row: 1, column: 3, score: 1}, MoveScore{row: 2, column: 2, score: 1}, MoveScore{row: 3, column: 5, score: 1}, MoveScore{row: 4, column: 4, score: 1}]

משימה 9: בדיקה אם הסתיים המשחק

כעת, נממש את הפעולה isGameOver אשר מחזירה אמת אם המשחק הסתיים.
תזכורת: משחק מסתיים כאשר אין לאף אחד מהשחקנים מהלך אפשרי. שימו לב שהמצב הזה יכול לקרות כאשר למשל הלוח מלא או כשלא נותרו דיסקיות לאחד מהשחקנים. נוכל לממש אותה בקלות יחסית ע״י שימוש בgetPossibleMoves מהסעיף הקודם בעבור כל אחד מהשחקנים.

משימה 10: מימוש העברות תור בין שחקנים

במשימה קודמת מימשנו את הפעולה placeDisk ובסופה דאגנו להעביר את התור לשחקן השני. זה כמובן עובד טוב לרוב המוחלט של המקרים אך יתכן שלשחקן השני אין תור לשחק. על כן, בטרם נעביר את התור לשחקן השני נרצה לבדוק אם יש לו מהלך חוקי לשחק. לצורך כך נממש את הפעולה switchToNextPlayablePlayer. הפעולה תעביר את התור לשחקן הבא אך אם אין לו מהלך לשחק תחזיר / תשאיר את התור אצל השחקן הנוכחי.
כעת נעדכן את הפעולה placeDisk להשתמש בפעולה switchToNextPlayablePlayer ובעזרתה נעביר את התור לשחקן השני רק אם יש לו מהלך.

משימה 11: קביעת המנצח

כל שנותר לנו לצורך מימוש פעולות המשחק הבסיסיות של המשחק הוא בדיקה מי השחקן שניצח.
לצורך כך נממש את הפעולה getWinner. הפעולה תחזיר: 1 אם שחקן 1 ניצח, 2 אם שחקן 2 ניצח 0 אם המשחק הסתיים בתיקו ו1- אם המשחק לא הסתיים.

משימה 12: יצירת בוט משלכם

נתונה המחלקה RenameThisClassMyReversiBot - המייצגת בוט שיודע לשחק את המשחק רברסי. כמו ששמה רומז, אתם מתבקשים להמציא שם לבוט שלכם ולשנות את שמה של המחלקה בהתאם. שימו לב: השם חייב להסתיים במילה Bot למשל FlipMasterBot או ReversiConquerorBot

ניתן להוסיף אסטרטגיות שונות לבוט שלכם ולממש כל אחת כפעולה נפרדת.
כאשר תשלחו את הבוט שלכם לטורניר, עליכם לקרוא מהפעולה getNexMove לפעולה שאתם מאמינים שתספק את הביצועים הכי טובים. למשל בבוט שנתון אנחנו קוראים לgetRandomMove, כך שאם הבוט הזה ישלח לטורניר הוא ישחק בצורה רנדומלית - אתם כמובן צריכים להוסיף אסטרטגיות טובות מהאסטרטגיה הרנדומלית.

משימה 13: מימוש בוט חמדן

נממש את הפעולה getNextGreedyMove במחלקת הבוט שיצרנו בסעיף הקודם אשר לא תקבל פרמטרים ותחזיר את המהלך האפשרי הבא (MoveScore) שבעבורו מספר היפוכי דיסקיות היריב הוא הגבוה ביותר. אם אין מהלך אפשרי הפעולה תחזיר null. הפעולה "תבקש" מהמשחק את כל המהלכים האפשריים (וכחלק מכך את מספר ההיפוכים של כל מהלך) באמצעות קריאה לפעולה getPossibleMoves שממשנו במשימה קודמת ותחזיר את הפעולה בעלת התמורה הגבוהה ביותר.
שימו לב, יש להעזר בפעולה הפומבית, getPossibleMoves, במחלקה ReversiGame אשר קוראת לפעולה getPossibleMoves הפרטית עם השחקן הנוכחי.

משימה 14: עדכון המחלקה הראשית לצורך משחק כנגד הבוט החמדן

נעדכן את המחלקה הראשית כך שבמקום לקלוט מהלכים משני שחקנים, נקלוט מהלכים משחקן אחד, נדפיס את הלוח ולאחר מכן ניתן לבוט "לשחק מהלך" ע"י קריאה לgetNextGreedyMove.
נסו לשחק כמה משחקים נגד הבוט.

תכנות המשחק רברסי - חלק ב׳2. מימוש בוטים נוספים והכנה לטורניר בוטים

בחלק הזה של הפרויקט אנחנו הולכים לבחון אסטרטגיות נוספות ולממש בוטים שמיישמים אותן. אחת השיטות שניישם תהיה בחירת מהלך שנותן את התמורה הגבוהה ביותר בהסתכלות של כמה מהלכים קדימה. בכדי לממש את האסטרטגיה הזו, נרצה לנסות כמה מהלכים ולבחון את האפשרויות של היריב לשחק. דרך אחת לבחון את האופציות האלה תהיה לייצר העתק של המשחק ולבדוק כמה מהלכים קדימה על ההעתק.
על כן, נתחיל בלממש פעולה בונה מעתיקה שיוצרת ReversiGame חדש שהוא העתק של הלוח הנוכחי.

משימה 14: בוט נגד בוט!

נעדכן את המחלקה הראשית כך שתאפשר לבוט החמדן לשחק נגד הבוט הרנדומלי. בעבור שחקן אחד נקרא לgetNextGreedyMove ובעבור השחקן השני נקרא לgetRandomMove כך שהבוטים ישחקו אחד נגד השני. בצעו כמה הרצות ותנו לכל אחד מהם להיות השחקן הראשון / שני.

משימה 15: הוספת פעולה בונה היוצרת עותק של המשחק

נייצר בנאי חדש למחלקת המשחק

```
public ReversiGame(ReversiGame game)
```

הבנאי מקבל משחק אחר ויוצר משחק חדש שהוא העתק של המשחק הנוכחי. שימו לב שיש לבצע ״העתקה עמוקה״ של לוח המשחק, קרי נייצר לוח חדש ונמלא את כל המשבצות בערכים של הלוח שממנו אנחנו מעתיקים. הסיבה לכך היא שאם לא נבצע העתקה עמוקה ונבצע שינויים בלוח במחלקה החדשה הם יתבצעו גם על הלוח המקורי (למה?).

משימה 16: מימוש בוט חמדני שמסתכל מהלך אחד קדימה

ננסה לשפר קצת את הבוט החמדני שמימשנו. נוסיף פעולה חדשה

```
public MoveScore getGreedyTwoStepsMove()
```

אשר מסתכלת מהלך אחד קדימה. זאת אומרת הפעולה בודקת בעבור כל מהלך אפשרי גם את המהלך שבו היריב יוכל להפוך דיסקיות שלנו. הבוט יבחר את המהלך שבעבורו סה״כ מאזן הדיסקיות ש״נרוויח״ יהיה המספר הגבוה ביותר. למשל אם בעבור מהלך מסוים אנחנו הופכים שלוש דיסקיות ולאחריו היריב יכול לשחק ולהפוך שתי דיסקיות אז מאזן הדיסקיות הכולל יהיה 1. שימו לב, יכול להיות שבעבור מהלך מסוים מאזן הדיסקיות הכולל יהיה שלילי. למשל, אם מספר הדיסקיות שאנחנו הופכים הוא 1 והיריב לאחר מכן יכול להפוך 3 דיסקיות אז סה״כ מספר הדיסקיות בעבור המהלך יהיה 2-. יש להשתמש בפעולה הבונה מהסעיף הקודם ולשכפל את הלוח בעבור כל מהלך

אפשרי שלנו ובאמצעות הלוח המשוכפל נבדוק את כל המהלכים שהיריב יכול לעשות. מתוך כל האפשרויות נבחר את המהלך שבו המאזן (ההפרש) הוא הגבוה ביותר.

משימה 17: עדכון אסטרטגיות הבוטים שלנו כך שיהיו רנדומליות

בטרם נגיע לתחרות הבוטים הרשמית, נרצה לשנות את האסטרטגיות שמימשנו כך שאם ישנם מספר מהלכים ״הכי טובים״ בעלי ציון דומה, הבוט יבחר אחד מהם באופן רנדומלי. המטרה היא שהמשחקים יהיו יותר מעניינים ויהפכו להיות פחות דטרמיניסטיים - קרי, פחות צפויים מראש. אם שני בוטים תמיד פועלים בדיוק באותו אופן המשחקים שלהם תמיד יראו בדיוק אותו הדבר.

משימה 18: ביצוע מספר רב של משחקים ברצף

נעדכן את הפעולה במחלקה Main אשר מריצה תחרות בין שני בוטים, כך שבכל הרצה יהיה מספר רב של משחקים בין שני הבוטים. בסיום הריצה יודפס בעבור כל בוט כמה פעמים הוא ניצח/הפסיד וכמה פעמים המשחק הסתיים בתיקו. המטרה היא לקבל תמונה יותר מדויקת איזה משני הבוטים טוב יותר מהשני.

משימה 19: כתיבת אסטרטגיה משלכם!

כעת תנסו להוסיף אסטרטגיות משלכם לבוט שכתבתם. תוכלו למשל לתעדף מהלכים שמשתלטים על פינות, גם אם המהלך לא הופך את המספר הגבוה ביותר של דיסקיות. לא ניתן להפוך כלי שנמצא בפינה, ועל כן זה נחשב למהלך טוב.

משימה 20: השתתפות בטורניר הבוטים