תרגיל 7

יש להגיש את התרגיל עד תאריך 22.12.21.

שימו לב ל<u>הערות</u> ול<u>הנחיות ההגשה</u> המפורטות בסוף המסמך.

חומר הלימוד לתרגיל: מערכים חד-ממדיים (אין להשתמש במערכים רב-מימדיים)

בתרגיל זה עליכם לממש מחלקה בשם Backgammon. יש לרשום ת"ז בשורה הראשונה של הקובץ **Backgammon.java** (ב-comment)

אנא קראו את כל ההנחיות בקפידה לפני שאתם שואלים שאלות!

בתרגיל זה תתבקשו לקודד את המשחק שש בש. אנא קראו את חוקי המשחק גם אם אתם מכירים אותם:

- <u>כאן בעברית</u>
- <u>כאן באנגלית</u>
- בגרסה שלנו:

1. המשחק מתחיל על ידי קביעת שחקן שישחק ראשון באקראי (במשחק שש בש פיזי הרבה פעמים נקבע לפי זריקת קוביה אך במקרה שלנו - אקראי בוליאני - רמז: אפשר להשתמש בפונקציה rd.nextBoolean() - עוד על התכונה rd בהמשך ההוראות).
2. אפשר להוציא אבנים רק מרגע שבו כל האבנים של השחקן נמצאות ברביע האחרון.
3. אין קוביית הכפלה.
4. אתם מתבקשים לממש רק לולאת משחק אחת (עוד על "לולאת המשחק" בהמשך) בלי התחשבות בכמות הניצחונות לכל משחק.
5. אם אותם המספרים מוגרלים בשתי הקוביות (סיטואציה שמכונה "דאבל") - יש שני מהלכים לכל קוביה. לדוגמא - אם יצא 2 ו-2 בקוביות אזי השחקן יזיז 2 פעמיים (שתי פעמיים) ועוד 2 פעמיים. אם יצא 3 ו-3 בקוביות אז השחקן יזיז 3 פעמיים ועוד 3 פעמיים.

לתרגיל זה מצורף קובץ עם שלד של הקוד - הקובץ Backgammon.java. עליכם לממש את כל הפונקציות בשלד ולהגישו בסוף. אחרי שסיימתם, אתם צריכים להיות מסוגלים להריץ את המשחק ולשחק בו מתחילתו ועד סופו בצורה חוקית. **באחריותכם לבדוק מקרי קיצון שונים בקוד.**

טיפ: חוקיות המשחק היא זהה לכל גודל של לוח (שמתחלק ב4).

תוכלו לבדוק את עצמכם יותר בקלות על ידי מימוש של לוח קטן יותר בן 8 משולשים בלבד (2 משולשים בכל רביע) עם קוביות בהן המספרים 1 ו-2 בלבד ומספר קטן של אבנים לכל שחקן.

המטרה כאן היא לבדוק את תקינות הקוד שלכם בתנאים מקלים - בדיקה של תזוזת אבנים, בדיקת חוקיות מהלכים (שלא ניתן לבצע מהלך לא חוקי), הוצאת אבנים מהלוח רק כשמותר ולנצח כשכל האבנים הוצאו.

תכונות

המחלקה תומכת בתכונות הפרטיות הבאות (כפי שניתן לראות בקובץ שקיבלתם. בקובץ תוכלו לראות גם מה הטיפוס של כל תכונה):

- הלוח (board) מיוצג על ידי מערך חד-מימדי בגודל 24 של מספרים שלמים. כדי לייצג את השחקנים נשתמש במספרים שלמים. המספר אפס מייצג מיקום שבו אין אבנים. מספר חיובי מייצג את האבנים הלבנות ומספר שלילי מייצג את האבנים השחורות. השחקן הלבן יזוז מהחלק העליון השמאלי ימינה ואז למטה ושמאלה. כלומר, מתחילת המערך ועד סופו. השחקן השחור יתחיל מהחלק השמאלי התחתון, יזוז ימינה ואז למעלה ושמאלה. כלומר, מסוף המערך ועד תחילתו.
- מערך בגודל 2 המתאר את האבנים ה"אכולות" (התכונה eaten) של כל אחד מן השחקים. האיבר הראשון במערך מכיל את כמות האבנים ה"אכולות" של הלבן והאיבר השני הוא כמות האבנים ה"אכולות" של השחור.
- המערך cubesUsages מתאר כמה מהלכים נשארו לכל אחת מהקוביות (נשים לב ש במקרה של הכפלה כל קוביה משוחקת פעמיים).
- המשתנה הבוליאני whitesTurn המתאר אם כרגע תור הלבן (אם whitesTurn סימן שכעת הלבן משחק).
- משתנה rd המחזיק אובייקט שמייצר משתנים רנדומליים.
- משתנה sc המאפשר קריאה לקלט מן המשתמש.

בנאים

ממשו את הבאים הבאים:

`Backgammon()`

בנאי ריק. צריך לקרוא לשיטה initBoard.

`Backgammon(int boardSize)`

בנאי שמקבל גודל של לוח. צריך לתמוך בהשמת אבנים נכונה ללוחות בגודל 24 בלבד. המלצה: בזמן דיבאגינג אתחלו לוח בגודל 8 עם מעט אבנים לכל שחקן והגרילו קוביות בטווח מ1 ועד 2 בלבד.

שיטות

ממשו את השיטות הבאות:

1. שיטה שבה יאותחלו כל התכונות של המחלקה. board, eaten, cubesUsages ו-whitesTurn. בנוסף יש לאתחל את board עם ערכים ספציפיים כפי שמתואר בסעיף 2.

חתימת השיטה:

```
public void initBoard(int boardSize)
```

2. שיטה שתחזיר מחרוזת אשר מייצגת את תכולת הלוח ואת מספר האבנים האכולות לכל שחקן בפורמט הבא:
רבע לוח ראשון משמאל עם סוגריים (מומלץ להשתמש ב-Arrays - יש רמזים איך לעשות את זה בשלד הקוד) ועוד רבע לוח שני עם סוגריים מימינו. ירידת שורה, ואז שוב אותו הדבר עם הרבעים השלישי והרביעי. ירידת שורה ואז Whites eaten - X , blacks eaten - Y.
לדוגמא:

```
[2, 0, 0, 0, 0, -5][0, -3, 0, 0, 0, 5]
[-2, 0, 0, 0, 0, 5][0, 3, 0, 0, 0,-5]

Whites eaten - 0, blacks eaten - 0
```

זהו מבנה הלוח ומספר אבני המשחק האכולות בהתחלה. כלומר, אם תדפיסו את המשתנה של הקלאס שלכם מיד לאחר יצירתו, אתם צריכים לקבל את ההודעה הזאת. חתימת השיטה:

```
public String toString()
```

שימו לב!

המערך שמייצג את הלוח הינו רציף, בעוד המערכים שמוצגים בהדפסה מייצגים את הלוח כפי שהוא נראה "באמת". בשורה השניה מודפסים קודם הרביע הרביעי (**בסדר הפוך**) ואז הרביע השלישי (**בסדר הפוך**) בעוד בשורה הראשונה מודפסים הרביע הראשון והשני בסדר הרגיל.

```
[ first quarter  ,   second quarter  ,    third quarter    ,    fourth quarter   ]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
[2, 0, 0, 0, 0,-5, 0,-3, 0, 0,  0,  5, -5,  0,  0,  0,  3,  0,  5,  0,  0,  0,  0, -2]
```

3. שיטה שמחזירה את התכונה המייצגת את הלוח. חתימת השיטה:

```
public int[] getBoard()
```

4. שיטה המחזירה את התכונה whitesTurn. חתימת השיטה:

```
public boolean getWhitesTurn()
```

5. שיטה המחזירה אמת או שקר בהתאם להאם השחקן הלבן מתחיל את המשחק. שימו לב, מדובר בשיטה שמחזירה כל אחד מהערכים בהסתברות של חצי: יש להחזיר true בהסתברות של חצי ו-false בהסתברות של חצי. חתימה השיטה:

```
public boolean whiteStarts()
```

6. שיטה שמגרילה שתי קוביות משחק חוקיות. כל קוביה יכולה להחזיר מספר שלם בין 1 (כולל) ל-6 (כולל). השיטה מחזירה מערך של שני המספרים שיצאו בהטלת קוביות (כלומר, כל תא מכיל מספר בין 1 ל-6, כולל). המלצה: בזמן דיבאגינג צרו לוח קטן יותר ותקטינו בהתאם את טווח המספרים שיצאו מהקוביות (למשל עבור לוח בגודל 8 הגרילו מספרים בין 1 ל-2). חתימת השיטה:

```
public int[] roll2Cubes()
```

7. שיטה שמחזירה האם המשחק נגמר (כלומר מישהו ניצח). חתימת השיטה:

```
public boolean gameOver()
```

8. שיטה המבצעת מהלך בהינתן מיקום וגודל צעד (הפרמטרים שהיא מקבלת, בהתאם). השיטה הזאת תעזר בשיטות המפורטות בסעיפים הבאים, בין היתר כדי לוודא מהלך חוקי ונכון. מהלך לדוגמא - לזוז מהמיקום של אינדקס 3 (כלומר הרביעי בלוח), שני צעדים. התזוזה ימינה או שמאלה תעשה לפי תורו של מי כרגע (לבן זז ימינה [כלומר זז מספר חיובי של צעדים] ושחור זז שמאלה [כלומר זז מספר שלילי של צעדים]). אם יש ללבן אבנים אכולות, צריך להתייחס למיקום הנוכחי שלהן כמינוס 1 כדי להזיז אותן כיוון שהן מגיעות מחוץ ללוח. אם יש לשחור אבנים אכולות צריך להזיז את האבנים האלו מהמיקום <גודל הלוח> (24 במקרה הדיפולטיבי) כי זה הצד של הלוח שממנו השחור משחק והוא בדיוק תזוזה אחת מחוץ ללוח.

```
public boolean move(int position, int moveSize)
```

9. שיטה המחזירה אמת אם נשארו לשחקן מהלכים חוקיים, ואחרת השיטה מחזירה שקר.

הערה: שיטה זו שימושית כדי להעביר את התור לשחקן היריב כשאין לשחקן מהלכים חוקיים לשחק בהם אבל המשחק עדיין לא הסתיים (השחקן "חסום"). אפשר ורצוי להשתמש בשיטה haveLegalMoves המפורטת מיד אחרי הפסקה הזו.

חתימת השיטה:

```
public boolean haveLegalMoves(int[] cubes)
```

10. שיטה המקבלת שני פרמטרים: מיקום וגודל צעד.
השיטה מחזירה האם המהלך הספציפי הוא מהלך חוקי. אם כן, השיטה מחזירה אמת, אחרת שקר. חתימת השיטה:

```
public boolean legalMove(int startPosition, int moveSize)
```

11. שיטה שמחזירה מה היא האבן הכי רחוקה מסוף הלוח. שימו לב שהתוצאה הזו תלויה בשחקן: מבחינת השחקן הלבן, האבן השמאלית ביותר השייכת שלו (הכי קרובה לתחילת המערך) ומבחינת השחקן השחור, מדובר באבן הימנית ביותר השייכת לו (הכי קרובה לסוף המערך). חתימת השיטה:

```
public int farthestStone()
```

12. שיטה המקבלת אינדקס המייצג את המיקום של האבן הכי רחוקה מסוף הלוח.
השיטה מחזירה האם האבן הרחוקה ביותר (כמו שמצאנו בעזרת השיטה הקודמת) נמצאת בתוך הרביע האחרון לשחקן. שיטה זו עוזרת להחליט האם אפשר להתחיל להוציא אבנים מהלוח. חתימת השיטה:

```
public boolean farthestStoneInLastQuadrant(int farthestStone)
```

13. שיטה המקבלת כפרמטר מיקום ומחזירה אמת אם הוא מחוץ לגבולות הלוח ואחרת שקר.
חתימת השיטה:

```
public boolean outOfBoard(int position)
```

14. שיטה שמחליפה את ערך המשתנה whitesTurn וכך בעצם מחליפה את התור (כלומר, אם היה אמת תהפוך אותו לשקר ולהפך). חתימת השיטה:

```
public void nextTurn()
```

לולאת המשחק אשר מיוצגת בפונקציה runGame כבר נכתבה לשימושכם.

מותר להוסיף שיטות פרטיות בהתאם לצורך . הקפידו לתעד את הקוד.
ביחד עם השלד של התרגיל, הקובץ Backgammon.java, אתם תקבלו גם טסטר, Tester.java עם מספר בדיקות. מומלץ להוסיף בדיקות נוספות משלכם שכן הטסטר בודק מקרים בסיסיים בלבד.

הערות

1. יש לרשום ת"ז בשורה הראשונה של הקובץ **Backgammon.java**.
2. לפני ההגשה חובה לקרוא את הקובץ "הגשת תרגילים - נהלים". על התרגיל לעמוד בחוקי כתיבת הקוד כפי שמפורט בקובץ. בפרט, שימו לב לתעד את הקוד, להקפיד על אינדנטציה ועל magic numbers. **שימו לב שירדו נקודות על הפרות של ההוראות בקובץ הנ"ל.**
3. לתרגיל זה מצורף הקובץ **Backgammon.java** המכיל את השלד של התרגיל - עליכם להוסיף מימוש לפונקציות בקובץ זה.
4. לתרגיל זה מצורף קובץ Tester.java אשר מכיל פונקציית main. קובץ הטסטר וקובץ הפתרון שלכם צריכים להתקמפל שניהם. שימו לב לשים את שניהם ב **default package**. תרגיל שלא עומד בתנאים (למשל לא מתקמפל עקב שם מחלקה לא נכון או שם package לא נכון, או Tester שלא מתקמפל עם התרגיל) יקבל 0 ללא אפשרות להגיש מחדש.

הנחיות הגשה

1. יש לרשום ת"ז בשורה הראשונה של הקובץ **Backgammon.java** (בהערה).
2. ארזו את הקובץ **Backgammon.java** בקובץ zip בשם **ex7.zip**.
3. הגישו את הקובץ **ex7.zip** במערכת המטלות של המכללה. **אין להגיש בזוגות ואין לשלוח קוד. העתקות נשלחות באופן אוטומטי לועדת משמעת.**
4. יש להגיש את התרגיל עד **22.12.21**. ניתן להגיש באיחור ללא קבלת אישור מיוחד אך עבור כל יום איחור ירדו 4 נקודות. ניתן להגיש עד 4 ימי איחור בלבד, כלומר יום אחרון להגשה הוא 26.12.21.