# החלטות על עיצוב המערכת

עורכות: חביבה מושבי (ת.ז- 322082892), דנה יפה (ת.ז- 312129240)

- מימוש getPossibleMoves-
  - **הבדלים:** חביבה עשתה שהלוח יודע מי השכנים של כל תא ויודעי ללכת לכיוונים מסוימים שהוא מקבל (הכיוונים זה enum). הלוגיקה עוברת על כל תא ומבקשת מהלוח לבדוק שכנים בכיוונים מסוימים. הלוגיקה קובעת לפי הפלט של הלוח האם השכנים רלוונטים או לא.
    אצל דנה הלוגיקה עוברת על הלוח וגם עושה את הבדיקה המקיפה של השכנים והכיוונים. היא נכנסת למימוש של הלוח ולכן <u>החלטנו לא לקחת את האופציה הזו</u>. הלוגיקה יודעת יותר מדי על הלוח.

- Square / Cell
  - האם הוא צריך להחזיק אובייקט דיסק שמחזיק צבע? או פשוט להחזיק צבע?
    - להחזיק אובייקט דיסק:
      - **יתרונות**: לכל דיסק יש 2 צבעים ויש לו state מסוים איזה צבע הוא הוא יכול להיות ב-2 הצבעים. כך לפעולה flip יש משמעות ממשית. באמת הופכים את הדיסק. (דיסק עובר עם enum)
      - **חסרונות**: הקצאת זיכרון נוספת. רק Cell מחזיק בו.
      - <u>בחרנו: לעבוד עם דיסק אבל להחביא אותו יותר טוב</u>. גישה תתאפשר ישירות דרך cell.
    - להחזיק רק צבע ללא אובייקט:
      - **יתרונות:** פחות הקצאות זיכרון.
      - **חסרונות:** זה לא משמעות המשחק, פחות לוגי מבחינת רברסי.

  - האם cell צריך להחזיק point שבו הוא נמצא? או לתת ללוח לשמור קואורדינאטות בשבילו?
    - להחזיק point:
      - **יתרונות:** גישה מהירה יותר למיקום של תא. לא צריך לעקוב אחרי אינדקסים בלולאות.
      - **חסרונות:** יותר זיכרון. (אפשר בלי הקצאות)
    - <u>החלטנו שכל תא ידע מה המיקום שלו</u> כדי להקל על בירור מיקום זה וגישה מהירה אליו.

  - Find possible moves – האם להחזיר תא שמחזיק את הלוקיישן שלו, או להחזיר רק לוקיישן?
    - להחזיר תא עם לוקיישן:
      - **יתרונות:** עובדים עם האובייקט שנצטרך יותר מאוחר, וכך לא נצטרך למצוא אותו.
      - **חסרונות:** ההשוואה היא לפי כתובות. ניתן לשינוי- <u>נדרוס אופרטור == עבור Cell</u>
    - להחזיר לוקיישן:
      - **חסרונות:** יוצר אצלנו מעברים מיותרים בין תא לבין Point, כי בסופו של דבר השינוי יתבצע ברמת התא, וה-Point זה רק אינדיקציה היכן התא נמצא.

- מי צריך לבצע את פעולת הכנסת הדיסק? המנהל או השחקן?
  - השחקן:
    - **יתרונות:** יודע מה הצבע שלו. במשחק הפיזי השחקן מבצע את הפעולה.

- **חסרונות:** עלול להכניס כשאסור לו או למקום שאסור לו אם לא נבצע בקרה ראויה.
- נשים לב ש זו מתודה שתישאר זהה בין כל מימוש של Player, ולכן נהפוך את Player להיות לא אינטרפייס, אלא מחלקה אבסטרקטית שמממשת חלק מהתודות עבור היורשים שלה. בין היתר- מתודת ה flip.

- המנהל:
  - **יתרונות:** הפרדה מלאה בין הלוח לשחקן- השחקן לא יכול לפגוע בלוח.
  - **חסרונות:** הפרדה מלאה בין הלוח לשחקן- השחקן פסיבי, יש מתווך בינו ובין הלוח. זה לא עובד ככה במשחק המקורי.
- החלטה: נבצע פיקוח על השחקן ונאמר לו מה הפעולות שהוא יכול לבצע- והוא יבצע אותן.

## - Printer או narrator?

- מדפסת:
  - **יתרונות:** יודע להדפיס הודעות סטרינג שמקבל
- דובר:
  - **יתרונות:** יש לו הודעות מוכנות שחוזרות על עצמן במשחק. שינוי יתבצע פעם אחת בקוד. מאוד יעיל
- החלטה: שילוב בין דובר למדפסת: נעשה קלאס חדש בשם **Message** שישמור את היתרונות של הדובר מבחינת פלטים מוכנים מראש, וישמור על הפרדת הנראות של המשחק מהמימוש עצמו (מה שהמדפסת עושה).

כמה משינויי הקוד שביצענו בהתאם להחלטות הנ"ל ולדרישות התרגיל:

1. Messages – קלאס חדש. מכיל מתודות סטטיות להחזרת סטרינגים. הוספנו מתודה ל printer שתקבל Message.
2. Player – הוספנו מתודה ממומשת של flip ו- insert ב- base class.
3. Point – החליף את Location
4. Cell – יחזיק Point שבו הוא נמצא. ב getPossibleMoves נרוץ על קבוצה של Cells.
5. GameLogic – התוספה מתודה getAmounts of Cells to flip
6. AIplayer – מקבל גם את הלוגיקה של המשחק.
7. הוספנו ל GameSetUp מתודה שמקבלת מהיוזר את ההחלטה איזה שחקנים הוא רוצה.