פרויקט משחק Book Scrabble (צד שרת)

הקדמה

סטודנטים יקרים שלום רב,

בימנו הרבה פרויקטים מכילים אלגוריתמים שונים בצד השרת ואפליקציות שונות בצד הלקוח. גם קיימים כלים מאד מתקדמים שמאפשרים לנו לעבוד ברמת אבסטרקציה מאד גבוהה. גבוהה עד כדי כך שאנו עלולים שלא להבין כיצד הדברים עובדים מאחורי הקלעים, וכתוצאה להפעיל אותם לא נכון. לכן, כדי ליצור הבנה עמוקה, בקורס זה אנו נממש דוגמאות פשוטות כמעט מאפס ונבנה אותן שכבה מעל שכבה בפרויקט מתגלגל עד שנגיע לסיום מכובד של מערכת שלמה.

למעשה פרויקט זה יכול לשמש כחלון הראווה שלכם כשתרצו להציג את הניסיון התכנותי שצברתם.

פרויקט זה מכיל את האלמנטים הבאים:

- שימוש בתבניות עיצוב וארכיטקטורה
- תקשורת וארכיטקטורת שרת-לקוח
- שימוש במבני נתונים \ במסד נתונים
- הזרמת נתונים (קבצים ותקשורת)
- הטמעה של אלגוריתמים
- תכנות מקבילי באמצעות ת'רדים
- תכנות מוכוון אירועים, אפליקציית desktop עם GUI

בכל סמסטר תגישו מספר אבני דרך, בקורס פת"מ1 נבנה את צד השרת (בעיקר), ואילו בפת"מ2 נרחיב את צד השרת לטיפול במספר לקוחות במקביל ונבנה גם את צד הלקוח כאפליקציית דסקטופ.

באתר המודול של הקורס יופיעו כל פרטי ההגשה – מה מגישים, לאן, כיצד תתבצע הבדיקה וכדומה.

בהצלחה!

תיאור כללי:

בפרויקט זה נבנה את המשחק Book Scrabble – בדומה למשחק Scrabble ("שבץ נא" בגרסה העברית) השחקנים יצטרכו להרכיב מילים המצטלבות זו עם זו כמו בתשבץ ולצבור נקודות. אולם, המילים החוקיות הן לא כל המילים במילון האנגלי, אלא רק מילים שמופיעות בספרים שנבחרו למשחק.

בחלק זה נבנה בשלבים שרת גנרי שיאפשר למשתמש לשחק מול השרת.

הגדרות:

אריח – Tile

- לוח קטן המכיל אות (באנגלית) ואת ערכה במשחק - כמות הנקודות שהאות שווה.
- בתרשים הבא ניתן לראות כמה כל אות שווה במשחק

- הניקוד מבוסס על יחס הפוך לשכיחות האות בשפה האנגלית. כלל שהאות נדירה יותר כך תקבל ניקוד יותר גבוה.

שק – Bag

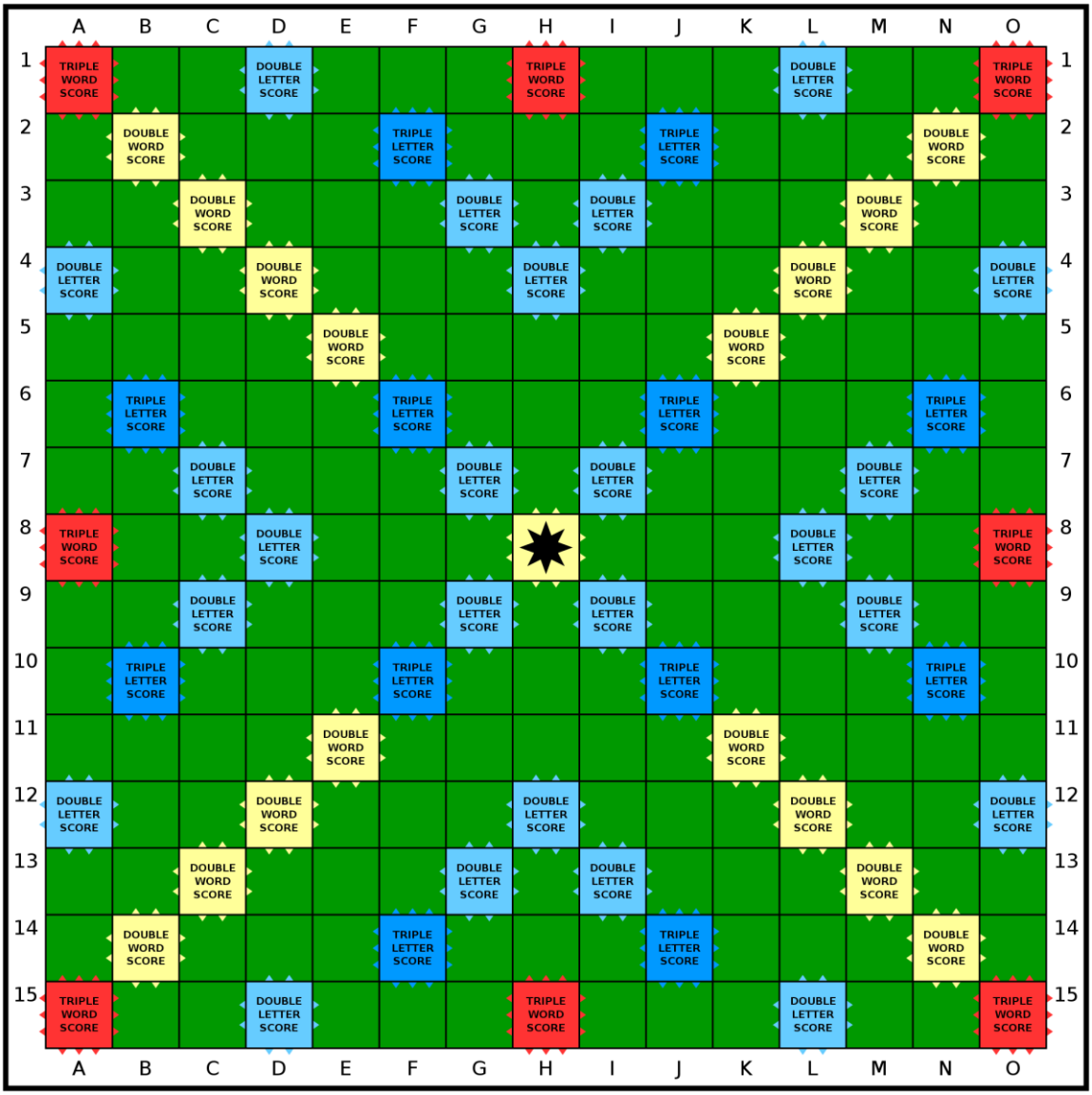
- שק המכיל 98 אריחים *
- מאפשר לשחקנים להוציא באקראי אריחים (כלומר ללא שיראו מה הם מוציאים)
- כמות האריחים בשק לכל אות בתחילת משחק:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
9	2	2	4	12	2	3	2	9	1	1	4	2	6	8	2	1	6	4	6	4	2	2	1	2	1

* במשחק המקורי ישנם גם שני אריחים ריקים אך במשחק שלנו נתעלם מהם

לוח המשחק – Board

- לוח דו-ממדי בגודל 15×15
- ללוח כמה משבצות בונוס:
  - המשבצת המרכזית (מסומנת בכוכב) מכפילה את ערך המילה שכתובה עליה
  - משבצות שמכפילות את ערך האות שנמצאת עליהן (תכלת)
  - משבצות שמשלשות את ערך האות שנמצאת עליהן (כחול)
  - משבצות שמכפילות את ערך המילה כולה (צהוב)
  - משבצות שמשלשות את ערך המילה כולה (אדום)
- משבצות הבונוס מפוזרות כבתרשים הבא:



חוקי ומהלך המשחק

לצורך הפרויקט נגדיר מערכת חוקים מעט פשוטה יותר מהמשחק המקורי:

1. כל שחקן שולף באקראי אריח מתוך השק
2. סדר השחקנים נקבע ע"פ סדר האותיות שנשלפו (מהקטן לגדול)
   a. אם נשלף אריח ריק נחזיר אותו לשק ונשלוף אחר.
3. כל האריחים חוזרים לשק
4. כל שחקן שולף באקראי 7 אריחים
5. השחקן הראשון (זה שהוציא את האות הקטנה ביותר בהגרלה) צריך להרכיב מילה חוקית שעוברת דרך המשבצת המרכזית (הכוכב) בלוח.
   a. אך ורק הוא מקבל עבורה ניקוד כפול.
   b. הוא משלים מהשק כך שיהיו לו שוב 7 אריחים.
6. בהדרגה, כל שחקן בתורו, מרכיב מילה חוקית מהאריחים שברשותו
   a. כאשר כמו בתשבץ, כל מילה חייבת להישען על אחד האריחים שקיימים על הלוח.
   b. לאחר כתיבת המילה השחקן משלים מהשק ל 7 אריחים
   c. הניקוד שלו מצטבר בהתאם לכל המילים שנוצרו על הלוח בעקבות השמת האריחים
      i. אריחים שמונחים על משבצות כפל או שילוש **אות**, יוכפל או ישולש ערכם בהתאמה
      ii. לאחר מכן המילה מקבלת את סכום ערך האריחים שלה
      iii. סכום זה יוכפל או ישולש עבור כל משבצת כפל או שילוש **מילה** שאחד האריחים מונחים עליה (כלומר, תיתכן למשל הכפלה ב 4 או 9 אם המילה תפסה שתי משבצות כפל מילה או שילוש מילה בהתאמה)
      iv. החישוב לעיל נכון לכל **מילה חדשה** שנוצרה על הלוח בעקבות ההשמה בתור
7. שחקן שאינו יכול להרכיב מילה חוקית מוותר על התור שלו.
8. המשחק יסתיים לאחר *N* סבבים.

מילה חוקית חייבת לעמוד בכל התנאים הבאים:

- כתובה משמאל לימין או מלמעלה למטה (ולא באף צורה אחרת)
- מילה שמופיעה באחד הספרים שנבחרו למשחק
- נשענת על אחד האריחים הקיימים על הלוח
- לא מייצרת על הלוח מילים אחרות שאינן חוקיות

דוגמת משחק:

נניח לצורך הדוגמה שיש 2 שחקנים והאות *R* יושבת על הכוכב.

- שחקן א כתב את המילה *Horn* השווה 7 נק', אך קיבל ניקוד כפול (בונוס כוכב) והשלים מהשק 4
- שחקן ב כתב *Farm* השווה 9 נקודות והשלים מהשק 3
- שחקן א כתב את המילה *Paste* השווה לבדה 7 נק', אולם
  - האותיות *P* ו *E* נפלו על משבצת "אות משולשת" ולכן המילה שווה 15 נק'

- בנוסף נוצרה מילה *Farms* השווה 10 נק' ולכן הוא קיבל בסך הכל 25 נקודות
- שחקן ב' כתב את המילה *Mob* (והשלים מהשק 2)
  - חוץ ממנה נוצרו על הלוח גם המילים *Not* ו *Be*
  - ויחד עם משבצות הבונוס בלוח הוא קיבל 18 נק' בסך הכל
- שחקן א' כתב *Bit* ובאופן דומה קיבל ניקוד מצטבר של 22 נק' עובר *Bit, Pi* ו *At*.

אבן דרך 1 – היכרות עם Java

ממשו את הטיפוסים הבאים

המחלקה Tile (אריח)

- נרצה שאובייקטים מסוג המחלקה יהיו immutable – כלומר לא ניתנים לשינוי.
  - נשיג תוצאה זו ע"י כך שהשדות שלה יוגדרו כ final.
  - הבנאי יצטרך לאתחל משתנים אלו
- תגדירו את השדות char letter עבור אות, ואת ה int score עבור הניקוד.
  - מכיוון שהם final, אין לנו בעיה שיוגדרו כ public
- הוסיפו אוטומטית באמצעות ה IDE שלכם
  - בנאי שמאתחל את השדות הללו, equals, ו hashCode

לא נרצה שכל מי שירצה יוכל לייצר אריחים. אנו רוצים לשלוט בכמויות שלהם לטובות המשחק. לכן ההרשאה של הבנאי תהיה private!

אולם, נממש מחלקה פומבית וסטטית בשם Bag (שק) **בתוך** המחלקה Tile, וכך תהיה מחלקה זו היחידה עם האפשרות ליצור אריחים.

המחלקה Bag (שק)

- תחזיק מערך של 26 int-ים המייצגים את הכמות של כל אות ע"פ הגדרות המשחק.
  - למשל תא 0 מייצג A ובו הערך 9 שמייצג שקיימים 9 אריחים מסוג A
  - בתא 1 שמייצג B יהיה 2 וכך הלאה... בתא 25 המייצג Z יהיה 1.
- תחזיק מערך של 26 אריחים, מסודרים לפי ה ABC
  - כל אריח עם האות והערך שלה ע"פ הגדרות המשחק (כל האותיות ב capital)
  - למעשה, אין לנו צורך בעוד אובייקטים מסוג Tile מלבד אלה המוגדרים במערך.
- המתודה ()getRand תחזיר אריח אקראי מתוך השק
  - היא מחזירה למעשה (by value) reference לאחד התאים במערך האריחים.
  - היא מורידה את הכמות המתאימה ממערך הכמויות
  - כמובן לא ניתן לקבל אריח כלשהו אם הכמות שלו ירדה ל 0.
  - אם השק ריק היא פשוט תחזיר null
- המתודה ()getTile תפעל באופן דומה ל getRand פרט לכך שהיא תקבל char ותוציא אריח שזו האות שלו מהשק אם ניתן, אחרת תחזיר null.
- המתודה ()put בהינתן אריח היא "תחזיר אותו לשק"
  - למעשה רק צריך לעדכן את הכמות.
  - בכל מקרה, מתודה זו לא תאפשר הכנסה מעבר לכמות המוגדרת בחוקי המשחק
- המתודה size תחזיר כ int את כמות האריחים שבתוך השק.
- לצורך הבדיקה, המתודה getQuantities תחזיר **העתק** של מערך הכמויות

וכדי להבטיח שיש רק שק אחד בתוכנית, גם כאן הבנאי של Bag יהיה private, ובנוסף ניצור מתודה פומבית וסטטית getBag() אשר תחזיר לנו מופע של Bag ע"פ לוגיקה של Singleton:

ניצור משתנה סטטי ופרטי מסוג Bag המאותחל ל null. במתודה getBag נבדוק האם המשתנה הזה null, אם כן, נאתחל אותו. ובכל מקרה נחזיר את הרפרנס אליו. כך הראשון שמפעיל את המתודה מייצר את האובייקט וכל השאר יקבלו רפרנס לאותו האובייקט. לתבנית זו קוראים Singleton.

המחלקה Word

מחלקה זו מייצגת השמה אפשרית של מילה על לוח המשחק. נגדיר את השדות הבאים:

- tiles – מערך של האריחים המרכיבים את המילה
- row, col – המגדירים את מיקום (שורה, עמודה) האריח הראשון במילה על לוח המשחק
- vertical – בוליאני המייצג האם המילה כתובה בצורה אנכית (מלמעלה למטה). אם הוא 'שקר' אז המילה כתובה בצורה אופקית (משמאל לימין)

בנאי המחלקה יאתחל את כל השדות ע"פ הסדר לעיל. לכל שדה יהיה getter משלו. בנוסף, נצטרך את מתודת ה equals. תוכלו לכתוב את הכל בצורה אוטומטית באמצעות ה IDE שלכם.

המחלקה Board

- לצורך התרגול, גם מחלקה זו תממש את תבנית Singleton שראינו לעיל, כאשר המתודה הסטטית getBoard() תחזיר לנו את הרפרנס למופע היחיד של לוח המשחק.
- מחלקה זו מחזיקה את לוח המשחק (בחרו לבדכם כיצד)
- המתודה getTiles() תחזיר מערך דו-ממדי של אריחים בהתאם למצב הלוח.
  - היכן שאין אריח על הלוח יהיה פשוט null.
  - שימו לב! האריחים הם immutable אך המערך לא. מישהו יוכל להוסיף לו אריחים שלא דרך Board ולכן גם כאן נרצה להחזיר העתק של המערך.
    - וזה לא נורא כי בסוף מדובר רק במצביעים.

במתודות הבאות מתייחסות להשמה של מילה אפשרית על הלוח. תשימו לב איך במקום מתודה אחת של placeWord שהיתה צריכה לבצע את ההשמה בפועל ולבדוק שהמילה חוקית על הלוח וע"פ המילון ולחשב את הניקוד לכל מילה שנוצרה וכו', אנו מפרקים את זה לכמה מתודות שונות ע"פ עיקרון ה **Single Responsibility**.

- המתודה ()boardLegal תקבל מופע של Word ותחזיר 'אמת' אם:
  - כל המילה נמצאת בתוך הלוח
  - נשענה על אחד האריחים הקיימים על הלוח כבתשבץ (אריח צמוד או חופף)
    - ההשמה הראשונה כזכור נשענת על משבצת הכוכב
  - לא דרשה החלפה של אריחים קיימים.

אחרת היא תחזיר 'שקר'.

למשל מתוך הדוגמה לעיל, בתור הראשון (HORN) ראינו שכל המילה נכנסה בתוך הלוח, ושאכן אחד האריחים נשען על הכוכב.

עבור ההשמה של FARM נוודא בנוסף שאחד האריחים צמוד או חופף לאחד האריחים הקיימים על הלוח. האריח R מספק דרישה זו. כמו כן, נצטרך לוודא שה R זהה ל R שהיתה קיימת כבר על הלוח בהשמה של HORN כך שהמילה HORN לא הוחלפה.

- המתודה ()dictionaryLegal תבדוק האם המילה חוקית מבחינת מילון המשחק (מילים המופיעות בספרים שנבחרו). לעת עתה היא תמיד תחזיר true.
- המתודה ()getWords – בהינתן Word היא תחזיר לנו מערך של כל המילים החדשות שיווצרו על הלוח כולל אותה המילה, לו היתה השמה כזו על הלוח. דוגמאות:
  - עבור PASTE בתור 3 לעיל, יוחזר מערך שמכיל את PASTE ואת FARMS.
  - עבור MOB תור 4 לעיל, יוחזר מערך שמכיל את MOB, NOT, BE.
  - סדר המילים במערך לא משנה
  - במקום מערך פרמיטיבי []Word תחזירו הפעם אובייקט מסוג ArrayList<Word>
    - אובייקט זה מאפשר למערך שהוא מחזיק לגדול באופן דינמי.
- המתודה ()getScore בהינתן Word היא תחשב את הניקוד הכולל של המילה, כולל כל משבצות הבונוס שעליהן היא מונחת.

שימו לב! עד כה אף מתודה לא מבצעת השמה על הלוח בפועל. אלו היו מתודות עזר.

כעת בהינתן מילה אפשרית להשמה, נוכל לבדוק באמצעות המתודות לעיל, האם היא חוקית מבחינת הלוח, אם כן אז לדרוש את כל המילים החדשות שהיו נוצרות מההשמה האפשרית של המילה, ועבור כל מילה כזו לבדוק האם היא חוקית מבחינת מילון המשחק. אם כל המילים אכן חוקיות אז נוכל סוף סוף לבצע את ההשמה בפועל על הלוח ולכן נחזיר את הניקוד המצטבר לכל מילה חדשה שנוצרה. בכל מקרה אחר, לא תבוצע השמה ונחזיר ניקוד 0.

בדיוק את זה עליכם לממש במתודה ()tryPlaceWord אשר בהינתן Word היא תחזיר ניקוד מתאים.

<u>שימו לב</u> שההשמה מכילה רק את האריחים **החדשים** שיש להניח על הלוח, ואילו הבדיקות השונות מכילות את כל המילה. לדוגמה כאשר ביצענו השמה ל FARM בתור השני, הנחנו רק FA_M על הלוח (במקום ה R יש אריח null) אך כל הבדיקות השונות לפני ההשמה על הלוח בדקו את המילה FARM במלואה.

כעת, תארו לכם לרגע, שאת כל החוקים האלה היינו מממשים במתודה אחת. האם היא היתה קריאה? האם היא בכלל היתה טסטבילית (ניתנת לבדיקה)? כיצד הייתם מדבגים אותה? ואיך הקוד היה נראה ללא המחלקה Word? אז גם עבור המתודות לעיל, מאד רצוי להשתמש במתודות עזר פרטיות!

בהצלחה!