**תרגיל 2 בקורס ג'אווה ואינטרנט – סמסטר ב' 2017**

**מילהדיאדהWordiada**

**מגישים:**

**ערן כץ, 200864858**

**לירן גרוסמן, 200899557**

**מייל לתקשורת:** erankatz91@gmail.com **(ערן)**

lirangr@mail.mta.ac.il **(לירן)**

**הפעלת התוכנית**

בעת פתיחת חלון המשחק, יש לבחור קובץ XML ע"י הכפתור **Load XML**.

במידה והקובץ נטען בצורה תקינה, יופיע לוח המשחק עם כלל המידע שנתבקשנו להציג, שנטען מהקובץ ומהמילון. לאחר מכן, יש ללחוץ על כפתור: **Start!** .

בראש המסך מוצג באופן שוטף הצעד הבא שיש לבצע, כגון תורו של איזה שחקן כעת, הוראה לבחירת אריחים מהלוח וכו'.

כאשר יגיע תורו של שחקן אנושי, הכפתור Roll Dice יהיה זמין, ואז צריך לבחור אריחים לפי ההוראה המוצגת בראש המסך.

עבור כל שגיאה של בחירה לא נכונה של אריחים, יקפוץ חלון dialog עם השגיאה ויש ללחוץ על

**Selection Clear Card.**

בכל שלב שחקן יכול לפרוש מהמשחק ע"י כפתור **Quit**.

אם נותר שחקן פעיל אחד וכל השאר פרשו אז הוא יוכרז כמנצח.

בכל שלב במהלך המשחק, ניתן לטעון קובץ XML ולהתחיל משחק חדש.

כאשר משחק מסתיים, יוצגו כפתורים **Next** ו **Prev**, שמאפשרים מעבר בין המהלכים שבוצעו במשחק.על מנת לצפות במהלך, יש ללחוץ על כפתור **Play.**

**מחלקות עיקריות**

הפרוייקט מורכב מ- 2 מודולים: engine ו uijavafx.

**Engine**

**Move -** מחלקה שאחראית על שמירת הסטטיסטיקות והנתונים בכל תור לצורך צפייה חוזרת בסוף המשחק.

**PlayerData -** מחלקה שאחראית על שמירת נתוני שחקן עבור הצגתם במסך

שאר המחלקות הינן בעיקר מתרגיל 1.

**uijavafx**

**BoardButtonController –** מחלקה זו אחראית על התממשקות לוח המשחק מול ה- model.

**CardUI –** מחלקה זו אחראית על התממשקות אריח מול ה- model.

**GameModel –** מחלקה זו אחראית על החלק הלוגי של ה- UI.

כל רכיבי ה- UI מתממשקים עם מחלקה זו.

**GameUIController –** מחלקה זו אחראית על ביצוע המהלכים של שחקן מול ה- Model.

בנוסף, אחראית על עדכון נתוני השחקנים מול ה- Model.

**רשימת בונוסים שמומשו:**

1. **החלפת skin למשחק** – בראש המסך מצד ימין קיים combobox שמאפשר טעינה של קובץ css עבור ה- UI. לאחר מכן, יש ללחוץ על כפתור Apply על מנת לבצע את ההחלפה של ה skin בפועל.
2. **אלגוריתמיקה חכמה עבור שחקן ממוחשב** – השחקן הממוחשב יבחר בכל מהלך מילה חוקית, כאשר מציאת המילים החוקיות בלוח נתון תבוצע ע"י האלגוריתם הבא:

קיים Package בשם wordSearch המטפל בחיפוש מילים חוקיות ע"י קבלה של רשימה של קלפים (האריחים החשופים בלוח המשחק).   
משתנה בשם wordSearcher הנמצא במחלקה Board נוצר בזמן יצירת המילון.  
ה- wordSearcher יוצר טיפוס Trie המכיל את כל המילים במילון –

יעילות יצירת הTrie הינו **O(W\*L)** כאשר W הוא סה"כ מספר המילים במילון ו-L הינו ממוצע אורכי המילים במילון.  
ברגע שמגיע תורו של השחקן הממוחשב, WordSearcher מקבל רשימה של אריחים גלויים ומחרוזת ריקה, ומבצע:

1. מאתחל מערך Boolean כאורך הרשימה של האריחים שהתקבלו כקלט, המשמש לבדיקה שלא השתמשו באותו אריח פעמיים להרכבת אותה מילה.
2. מריץ את הפונקציה הרקורסיבית dfs עם מערך ה-Boolean ומחרוזת ריקה:
   1. מוסיף לסוף המחרוזת שבונה את אחד האריחים מהרשימה (בכל פעם מתחילה מאריח אחר)
   2. הפונקציה dfs עוברת עם המחרוזת על ה- Trie, וכאשר היא מוצאת שקיימת תחילית של מילה במילון שזהה למחרוזת, היא מוסיפה אריח נוסף מהרשימה למחרוזת שהיא בונה, ואז מחפשת שוב תחילית של מילה במילון שתתאים למחרוזת המעודכנת (וכך הלאה), תוך כדי בדיקה שלא משתמשת באותו אריח פעמיים(באמצעות מערך (Boolean.
   3. אם לא קיימת מילה במילון עם תחילית שמתאימה למחרוזת המעודכנת אז החיפוש עבור מחרוזת זו מסתיים.
   4. אם היא מוצאת שהמילה קיימת בTrie- היא מוסיפה אותה לרשימת המילים החוקיות.
3. מוחזרות המילים החוקיות שניתן להרכיב באמצעות אותם אריחים (מה שפונקציית ה-dfs מצאה)

סה"כ יעילות: O(W \* L!), כאשר W הוא סה"כ מספר המילים החוקיות ו-L הינו ממוצע אורכי המילים שנמצאו חוקיות באריחים.