מס' גרסה:1

פיתוח תוכנה מבוסס java

Code Names אביב 2024 -

מרצה: **אביעד כהן** [aviadco@mta.ac.il](mailto:aviadco@mta.ac.il)

בודק: **איתי כהן**  itaych@mta.ac.il

התרגיל מנוסח בלשון זכר, אך מכוון לכלל המגדרים והתחושות בצורה שווה

תוכן העניינים

[דרישות הקורס 4](#_Toc162699352)

[כללי 4](#_Toc162699353)

[איך להגיש תרגילים באיחור, ולהישאר בחיים 5](#_Toc162699354)

[הנחיות כלליות לכתיבת התרגיל 6](#_Toc162699355)

[תרגיל reflection – תרגיל עצמאי ביחידים (% 10) - הגשה: 25.4.24 8](#_Toc162699356)

[כללי 8](#_Toc162699357)

[מבנה התרגיל 8](#_Toc162699358)

[ניקוד 8](#_Toc162699359)

[איך בודקים ? 9](#_Toc162699360)

[מה מגישים ? 9](#_Toc162699361)

[מקרה בדיקה לדוגמא 9](#_Toc162699362)

[Code Names (שם-קוד) 11](#_Toc162699363)

[חוקי המשחק 11](#_Toc162699364)

[מהלך המשחק 11](#_Toc162699365)

[הרחבות 11](#_Toc162699366)

[הנחיות ספציפיות למימוש תרגיל שם-קוד 12](#_Toc162699367)

[תרגיל 1 – מימוש **שם-קוד** כאפליקציית Console (% 35) - הגשה: 21.5.24 13](#_Toc162699368)

[פרטים יבשים 13](#_Toc162699369)

[דרישות 13](#_Toc162699370)

[חלוקה למודולים 17](#_Toc162699371)

[איך מתחילים ? (המלצה...) 17](#_Toc162699372)

[בונוסים 18](#_Toc162699373)

[סבבה, סיימתי. מה ואיך להגיש ? 18](#_Toc162699374)

[תרגיל 22 – מימוש **שם-קוד** כאפליקציית Client - Server (% 55) – הגשה: 23.7.24 19](#_Toc162699375)

[פרטים יבשים 19](#_Toc162699376)

[דרישות 19](#_Toc162699377)

[התממשקות עם ממשק גרפי 22](#_Toc162699378)

[אפליקציית Administrator 23](#_Toc162699379)

[אפליקציית משתמש 24](#_Toc162699380)

[חלוקה למודולים 24](#_Toc162699381)

[איך מתחילים ? 24](#_Toc162699382)

[בונוסים 25](#_Toc162699383)

[סבבה, סיימתי. מה ואיך להגיש ? 25](#_Toc162699384)

[נספח א' – תיאור מבנה המערכת באמצעות קובץ XML 26](#_Toc162699385)

[סכמה תרגיל 2 28](#_Toc162699386)

[נספח ב' – סכמת קובץ ה XML 29](#_Toc162699387)

[נספח ג' – סקיצות מסכי ה UI 31](#_Toc162699388)

[מסך הלוגין 31](#_Toc162699389)

[מסך הדשבורד 31](#_Toc162699390)

דרישות הקורס

## כללי

1. בקורס אין בחינה אך חובה להגיש תרגילים (סה"כ 4).
2. המלצתי היא להגיש את התרגילים ביחידים. אולם אם אין ברירה - את מרבית התרגילים (3) ניתן להגיש בזוגות, אך לא בשלישיות / רביעיות / חמישיות או יותר

(כן, גם אם מדובר בשלישיה / רביעיה / חמישיה הצועדת יחדיו לאורך שנים מאז גיל הגן והגישה עד עכשיו את כל הפרוייקטים ביחד).

את תרגיל ה - [reflection](#_כללי) חובה להגיש ביחידים.

1. בעבודה משותפת על תרגיל יש להקפיד על מעורבות אקטיבית של כלל המגישים בכל חלקי התרגיל.
2. במידה והוגדר בונוס לתרגיל מסוים, ציון הבונוס יתווסף לציון התרגיל בלבד (ולא לציון הסופי של הקורס כולו).
3. התרגילים יוגשו דרך מערכת Mama. מוגדר רכיב 'מטלה' נפרד לכל תרגיל.
4. לפני שליחת התרגיל יש לבדוק שהוא עובד ומכיל את הקבצים המעודכנים ביותר, על מערכת "נקייה".

בצעו את סט הפעולות שאתם מצפים מן הבודק לבצע וודאו כי הכל מתנהל כראוי וכסדרו.

1. ניתן להחליף את השותפ/ה בכל תרגיל, ללא צורך באישור או הודעה למרצה.

הניקוד על כל תרגיל נזקף לזכות הסטודנטים שבצעו אותו בלבד.

1. טרם הגשת התרגיל המתגלגל הראשון, תידרשו לשלוח מייל לבודק שבו מידע על צוות מגיש/י התרגיל.   
   במידה ויחול שיוי בציוות בתרגילים הבאים – יש לשלוח עדכון מתאים במייל לקראת ההגשה של התרגיל הבא.  
   הבודק ישלח הודעה מתאימה בלוח ההודעות של הקורס לגבי מבנה המייל ותכולתו.
2. יש להעלות את התרגיל רק עבור אחד מבני הזוג ולהוסיף את שם בת/בן הזוג ומספר תעודת הזהות שלה/ו גם באתר וגם בקובץ ה readme המצורף (פרטים בהמשך).

## איך להגיש תרגילים באיחור, ולהישאר בחיים

1. ניתן להגיש תרגיל עד שבוע איחור, כאשר עבור כל 24 שעות איחור – תורד נקודה אחת מציון התרגיל ; תרגיל שיוגש באיחור של יותר משבוע (ללא סיבה מוצדקת) – **פשוט לא יבדק**.
2. במידה והגשתם תרגיל אולם הבודק נתקל בבדיקתו במצב שפשוט לא מאפשר את המשך הבדיקה (למשל כישלון בטעינת קובץ הבדיקה) – הרי שאתם מוגדרים כתקלת level 0. במקרה של תקלה שכזו הבודק יידע אתכם ויאפשר לכם לבדוק, לתקן ולהגיש מחדש את התרגיל כדי שאפשר יהיה לבודקו אחרי הכל.   
   שימו לב **כי בכל במקרה של הגשה חוזרת** בגלל level 0 – הציון לתרגיל יתחיל מ 90, ללא שום קשר לאופי הבעיה ו/או התיקון (גם אם התיקון היה "קטן". גם אם התיקון היה בגלל בלבול בהגשה של גרסה קדומה יותר של הקבצים. גם אם הכלב אכל לכם את שיעורי הבית)
3. בתרגילים השונים ניתן לממש בונוסים (פרטים בהמשך).

המטרה של הבונוס היא לעזור לכם להעלות את הציון ולא להורידו !

רוצה לאמר: אל תגישו באיחור רק בשביל להספיק לפתח בונוס.

בונוס מפתחים **אם ורק אם** סיימתם את כל דרישות הבסיס להגשה, יש לכם עוד מספר ימים, וברצונכם לנסות ולהגדיל את הציון ע"י בונוס.

היות וכך, ולמען הסר כל ספק: **לא ייבדקו** הבונוסים עבור תרגילים שהוגשו באיחור (שאינו מוצדק).

בהתאם לכך, אני שומר לעצמי חירות רבה יותר בשינוי כזה או אחר של מי מסעיפי הבונוס, גם במהלך התרגיל עצמו.

1. עומס בלימודים, בעבודה, בחיים, בגלל הילדים או ההורים, שכנים וחברים (או בכל תחום אחר) אינו נחשב כסיבה לגיטימית לבקשת הארכה.
2. במקרה של בקשה להארכה (מכל סיבה שהיא, לרבות מילואים ומחלה) יש לפנות למרצה מראש על מנת לקבל אישור.  
   הפנייה תתבצע במייל.
3. לאנשי הקבע – הישארות של שבת בבסיס אינה נחשבת כסיבה לגיטימית להארכה (מכיוון שזה חלק מהסדר העבודה בצה"ל); יציאה לאבט"ש כן נחשבת כמילואים ויש להגיש אישור ממפקד הבסיס.
4. ניתן לערער על ציון של תרגיל לכל היותר שבוע מיום פרסום המשוב והציון במע' המאמא.

כדי לערער יש לשלוח מייל לבודק בצירוף כל הסיבות והטענות שלכם.

## הנחיות כלליות לכתיבת התרגיל

1. במהלך הקורס יוצגו דוגמאות והסברים מבוססים על כתיבה בסביבת הפיתוח (IDE) – Intellij IDEA.

אתם מוזמנים (ומעודדים בזאת) לפתח גם כן את התרגיל בסביבת העבודה intellij.

ניתן לקבל רישיון חינם לשימוש בגרסת ultimate, רק בשל היותכם סטודנטים במכללה (כבר שווה !)

יחד עם זאת, כל אחד רשאי לבחור לעבוד בסביבת העבודה הנוחה והמוכרת לו. כך או אחרת הגשת התרגיל אינה כוללת את סביבת הפיתוח אלא אך ורק הרצה ידנית מ cmd (כמו פעם...).

שימו לב: מבחינתכם, לבודק פשוט אין intellij (או כל ide אחר) ולכן זו אפילו לא אופציה.

חיסכו ממני (ומכם) את כתיבת המייל המבקש זאת.

1. יש להגיש את התרגילים בתור קובץ zip/rar (לא 7Z !)

הקובץ יכיל:

1. כל הקבצים הרלבנטים להפעלת התרגיל (jar/war – פרטים בגוף התרגיל).
2. קובץ אצווה ( == batch) שיכיל את הפקודה שמריצה את התרגיל.
3. קובץ readme שיכיל את פרטי המגיש/ים, כמו גם הנחיות כלליות להרצה התרגיל וכל הנחות שלקחתם במהלך התרגיל ואתם סבורים שחשוב כי הבודק יכיר. דמיינו כי בכל שאלה/תקלה שיתקל בהן הבודק, יעמוד לרשותו רק קובץ ה readme שלכם. דאגו להבהיר ולהסביר את כל הדברים שיכולים להשתבש ו/או שבעטיים ייתכנו בעיות/שאלות/תהיות וכיוצב'.

כמו כן, כל הנחה שאתם מניחים בעצמכם לגבי אופן מימוש התרגיל (בין אם בלוגיקת התרגיל ובין אם בהנחה טכנולוגית) צריכה להיות רשומה בקובץ.

על קובץ הreadme להיות בפורמט word או pdf (**לא notepad !**).   
חי נפשי – אם מישהו מגיש readme כקובץ טקסט פשוט -ירד לו ניקוד מהתרגיל...

1. דווקא בגלל שאין זהו קורס שבו יכנסו לנבכי הקוד ויבדקו כל שורה ושורה, יש להקפיד ביתר שאת על קוד נקי, מסודר, קריא ויעיל. בפרט:
2. הימנעו משכפול קוד
3. פונקציות ארוכות מדי (בדר"כ יותר מגודל עמוד)
4. בחירת שמות גרועים למחלקות, לפונקציות ולמשתנים
5. הזחה (אינדנטציה) נכונה
6. imports מיותרים
7. יש להקפיד להשתמש ב-modifiers בצורה נבונה:
8. מחלקה שלא אמורים לבנות אובייקטים שלה אמורה להיות מוגדרת כ-abstract
9. קבועים יש לסמן כ- final
10. משתנים של המחלקה רצוי להגדיר כ-private אלא אם יש סיבה לגיטימית לבחירה אחרת.
11. יש להקפיד על מוסכמות בסגנון הכתיבה – שמות מחלקות יתחילו באות גדולה, שמות חבילות, משתנים ופונקציות באות קטנה, שמות קבועים יהיו מורכבים רק מאותיות גדולות וכו'. ראו מסמך java coding conventions שהועלה למאמא.
12. התמודדות עם קלט שאינו תקין (במקומות הרלבנטים) היא חלק בלתי נפרד מחווית המפתח (לטוב ולרע...).

יש לוודא קלט תקין מהמשתמש בכל שלב ולהחזיר הודאות שגיאה קריאות, אינפורמטיביות במידה והקלט אינו תקין. (למשל: לא להגיד שהקובץ לא תקין – אלא מה לא תקין בקובץ בצורה מפורטת...)

1. כל הקלט והפלט בתרגילים השונים יהיה באנגלית בלבד.

אין להציג או לתמוך בקבלת קלט ו/או הצגת פלט בעברית או בכל שפה אחרת.

כל הקלטים באנגלית יהיו case insensitive, כלומר אין חשיבות ל capital case. דוגמא: MoMo=mOmO

1. הוראות שגויות שייגרמו לאפליקציה שלא לרוץ יורידו נקודות, ולכן רצוי מאוד שתנסו להתקין את האפליקציה בעצמכם לפי ההוראות שתכתבו.

* **זהו תרגיל מתגלגל. המטרה היא לבנות בסיס ראשוני בתרגיל הראשון, ולהמשיך ולהשתמש בו, ככל האפשר (ואפשר !) במהלך התרגילים הבאים. השקיעו חשיבה ותכנון בעיצוב הפתרון תוך מחשבה על איך מה שתעשו היום ישרת אתכם מחר. (זה כלל נכון לחיים, לא רק לתרגיל זה).**
* חלק מהעבודה בתרגילים היא קבלת החלטות בנושאים שאינם מפורטים במדויק. מטרת התרגיל היא לתרגל את הנושאים המרכזיים הנלמדים בקורס. על כן, בכל מקום שלא מופיעה דרישה מדויקת – מוטל עליכם לבחור בדרך ההגיונית ביותר שנראית לכם ולציין את בחירתכם בקובץ ה Readme אשר מוגש עם התרגיל. אם יש ספק לגבי אופן פעולתכם אתם מעודדים לשאול האם הפתרון שאתם חושבים לתת לסוגיה מסוימת הוא קביל ולגיטימי (שאלות בפורום, מייל למרצה וכו)
* התרגיל מתקיים כולו במסגרת ג'אווה גרסה 8. הקפידו להוריד, לעבוד, לקמפל ולהריץ עם הגרסה המתאימה בלבד.
* **ווידוא הגשת התרגיל טרם הגשתו:**

1. **יש לוודא כי ההגשה שלכם רצה היטיב על מע' נקייה, באופן שבו גם הבודק יריץ אותה,** על מערכת נקייה וללא תוצרי לוואי אחרים של הפעלות קודמות שלכם.
   * **הבודק יבצע את הבדיקה על מע' windows 10. כל מי שמפתח על גבי mac/linux – זכותכם – אבל גם חובתכם לוודא כי אתם רצים היטיב על windows 10. למען הסר ספק, לא תתבצע בדיקה על מע' הפעלה אחרת. כמו כן לא תהיה התחשבות בתקלות שמקורם רק בשל עבודה על מע' הפעלה שונות (ולא שאמורות להיות תקלות כאלה..)**
   * **יש לוודא כי כל קבצי הבדיקה השונים שהועלו ל mama נטענים בהצלחה ע"י המע' שלכם טרם ההגשה. בדיקת הבודק תתחיל מבדיקה בסיסית המבוססת בצורה גסה על קבצים אלה. חבל ליפול Level 0 על שטות שיכולתם לעלות עליה בשנייה עוד בשלב הפיתוח.**
   * **כאמור, הגשה חוזרת בשל תקלות level 0 תתחיל מראש מציון של 90. בלי שום יוצא מן הכלל. הקדימו תרופה למכה.**

* **בחלק מהתרגילים ניתנת אפשרות למימוש דרישות בונוס.**
  + **ישנם 2 סוגי בונוסים:**
    1. **בונוס בתוך טווח התרגיל - יכול להביא אתכם לכל היותר לציון 100, ולחפות במקרה והורדו לכם נקודות בשל תקלות.**
    2. **בונוס מחוץ לטווח התרגיל – יכול להעלות את ציונכם אף מעבר ל 100 (וכן, יש כפל מבצעים לטובת הלקוח).**
  + **בכל מקרה יש לבצע את הבונוס אם ורק אם סיימתם את כל דרישות הבסיס ההכרחיות לתרגיל.**
  + **חלק מהבונוסים בתרגילים השונים הם כאלה שנועדו "להקדים תרופה למכה" – מימוש דרישה בתרגיל n אשר בכל מקרה תגיע כדרישה חובה בתרגיל n+1.**

**הדבר נועד לעודד אתכם להוריד את העומס הצפוי בתרגיל n+1, מתוך הנחת יסוד שתרגיל n הוא קל יותר ומרווח יותר.**

**תכננו את עבודתכם בהתאם ושאפו "להקדים תרופה למכה", במידת האפשר. (וגם זה כלל חשוב לחיים, בלי קשר לתרגיל ולקורס).**

* + **פירוט הבונוסים, משקלם ונקודותיהם מפורט בגוף התרגיל הספציפי.**
  + **אם כבר מממשים בונוס, יש לממש את כולו, עפ"י דרישתו כדי לזכות במלוא הניקוד שהוא מקנה. בכל מקרה ההחלטה על ניקוד הבונוס היא בידי הבודק/מרצה בלבד (המגמה היא להיות נדיבים ככל האפשר...)**
  + **כאמור, ולמען הסר כל ספק – ניקוד הבונוס מתווסף לניקוד התרגיל הספציפי שבו הוא מומש ולא לניקוד הסופי של הקורס. לא ניתן לקבל ציון סופי בקורס שהוא מעל ל 100 (גם אם בזכות הבונוסים הגעתם לציון כזה(.**

תרגיל reflection – תרגיל עצמאי ביחידים (% 10) - הגשה: 25.4.24

## כללי

**מועד הגשה: 25.4.2024 צורת הגשה: ביחידים בלבד**

בתרגיל זה תזכו להכיר ולתרגל את מנגנון ה Reflection בג'אווה.

המצגת מכילה ידע בסיסי המהווה נק' פתיחה בלבד לנושא זה, ולכן כחלק מהתרגיל תדרשו גם ללמוד לבד נושאים נוספים הקשורים לעולם ה reflection שייתכן ואינם מכוסים (או אינם מכוסים כהלכה) במצגת.

שימו לב כי במצגת ישנם 3 נושאים. התרגיל הוא על הנושא הראשון בלבד של Reflection.

מהות התרגיל היא לכתוב investigator שיודע לקבל מופע (instance) של איזה שהוא אובייקט, ואז יודע "לחקור" אותו ולענות על מספר שאלות בהקשרו.

במהלך התרגיל תצטרכו להשתמש אך ורק ביכולת ה Reflection של ג'אווה כפי שמוסברות במצגת.

**אין להשתמש (ואין שום צורך) בשום ספריית צד שלישי כדי לבצע את המטלות השונות !**

צפי העבודה על התרגיל, בהינתן שקראתם והבנתם את המצגת, הוא 6-5 שעות.

צפי אורך הקוד שעליכם לכתוב הוא לא יותר מ 250 שורות (אני עשיתי זאת ב 160 שורות מרווחות היטיב..)

הבדיקה לתרגיל תבוצע בצורה אוטומטית, ע"י קוד שיטען את הקובץ שלכם ויריץ את סט הבדיקות שנגזרות ממנו, כלומר יקרא לכל השיטות המוגדרות ב interface שמימשתם, תוך השוואת הערך המוחזר מהקריאה אל ערך מצופה.

## מבנה התרגיל

התרגיל מכיל ממשק בשם Investigator, אותו עליכם לממש. מהות השיטות בממשק היא לחקור instance של class אחר.

כל שיטה בממשק מתועדת היטיב מבחינת מה היא צריכה לעשות, מה ההנחות שנלקחות במסגרת תפעולה, מה הפרמטרים שהיא מקבלת ומה היא צריכה להחזיר.

חלק מהתרגיל כולל גם התמודדות עם התיעוד והבנה בעזרתו בלבד מה עליכם לעשות בכל שיטה ושיטה.

**הערות חשובות**:

1. שימו לב כי ה class שאתם מממשים חייב להכיל default public constructor !
2. שימו לב כי עליכם למקם את הממשק שקיבלתם (Investigator.java) בדיוק תחת ה package שנקרא **reflection.api** (ובהתאם לכך גם היררכיית הספריות כמובן). המחלקה שאתם מממשים, מאידך, יכולה להיות ממוקמת בכל package אחר.
3. במידה ויש מטודות הדורשות תפיסת exception, יש לתפוס אותו אולם אין להדפיסו ! (זה יוצר אי-סדר בהדפסות הפלט). במידה ומתרחשת תקלה או שתזרקו הלאה את ה exception ותוכנית הבדיקה תתמודד איתו בדרכה (הורדת ניקוד על הסעיף המדובר) או שתחזירו ערך כלשהוא כתוצאת המטודה (גם אם הוא לא נכון).
4. התוכנית מתחילה מקריאה לפונקיה load (אחת הפונקציות המתוארות בממשק). בפונקציה זו תקבלו את ה instance אותו עליכם לחקור. אפשר להניח כי זו הפונקציה הראשונה שתקרא, וכי היא תיקרא בדיוק פעם אחת.
5. כל מטודה בתרגיל עובדת בפני עצמה ופועלת על ה instance שקיבלתם במטודה load. אין להניח או להסתמך על סדר בקריאת המטודות (למען האמת בכל בדיקה הסדר הוא רנדומלי)
6. אין שום צורך (מבחינת התרגיל) לכתוב פונקציית main בקובץ התוכנית שלכם. אני לא אפעיל את התוכנית שלכם, אלא אפעיל את תוכנית הבדיקה שלי (שם יש main) והיא, בתורה, תטען את התוכנית שלכם.

## ניקוד

התרגיל שווה עד 5 נקודות **מהציון הסופי** (!!)

(לא רע ל 6 - 5 שעות עבודה ו 250 שורות קוד...)

## 

## איך בודקים ?

קיבלתם את תוכנית הבדיקה עצמה, אותה הבודק הולך להפעיל לטובת בדיקת התרגיל.

התוכנית מקבלת כפרמטר חיצוני את שם הקובץ המקומפל שלכם (.class) ובודקת אותו על מספר מקרי בדיקה.

בעותק התוכנית שקיבלתם כרגע, יש דוגמא למקרה בדיקה פשוט עליו תוכלו לנסות, להתנסות ולוודא אם אתם בכיוון הנכון או אם לאו.

כדי להפעיל את תוכנית הבדיקה עליכם לנווט לספרייה בה נמצאת תוכנית הבדיקה, ולהקליד ב CMD:

…\> RunTester <your .class file name (including the .class suffix)>

אם הכל עובד כשורה, תקבלו את הפלט הבא:

Test name: Rectangle Class

Testing Rectangle Class basics...

Testing [ getTotalNumberOfMethods ]: expecting answer [6] and got [6]

…

Test Score: 100

## מה מגישים ?

עליכם להגיש קובץ zip, הכולל **בדיוק** 2 קבצים (ו 2 קבצים בלבד !):

1. קובץ התוכנית שלכם בלבד, בגרסתו המקומפלת (.class).
2. קובץ קוד המקור (למקרה של בעיות חמורות בלבד)

שם קובץ הזיפ צריך להכיל את שמכם ואת הת.ז. (אין להגיש קובץ readme בתרגיל זה...)

**מה לא מגישים ?**

1. את קובץ הממשק שקיבלתם
2. תיקיית פרויקט...
3. כל דבר אחר שבמקרה יושב לכם ליד הקוד...

## 

## מקרה בדיקה לדוגמא

במקרה זה מתואר האובייקט Rectangle היורש מאובייקט שנקרא Polygon.

מימוש זה כבר מוטמע בתוכנית הבדיקה שקיבלתם ומופע שלו יינתן כקלט לתוכנית שלכם.

המופע יאותחל בצורה הבאה:

**rectangle** = **new** Rectangle(4,6);

(המימוש הוא חלקי ולא תמיד הגיוני – אז אל תתפסו לקטנות...)

**public class** Polygon {  
  
 **private** Set<Point> **points**;  
  
 **public** Polygon() {  
 **points** = **new** HashSet<>();  
 }  
  
 **public int** getTotalPoints() {  
 **return points**.size();  
 }  
  
 **protected void** addPoint(**int** x, **int** y) {  
 **points**.add(**new** Point(x, y));  
 }  
}

**public class** Rectangle **extends** Polygon **implements** Comparable, Serializable {  
  
 **private int x**;  
 **private int y**;  
 **private final int SCALE** = 2;  
  
 **public static void** PRINT\_SOMETHING() {  
 System.***out***.println(**"this is a static method"**);  
 }  
  
 **public** Rectangle() {  
 **x** = -1;  
 **y** = -1;  
 }  
  
 **public** Rectangle(**int** x, **int** y) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 updateParent();  
 }  
  
 **private void** updateParent() {  
 addPoint(0, 0);  
 addPoint(**x**, 0);  
 addPoint(0, **y**);  
 addPoint(**x**, **y**);  
 }  
  
 **public int** calcArea() {  
 **return x** \* **y**;  
 }  
  
 **public int** calcPerimeter() {  
 **return** twice(**x**) + twice(**y**);  
 }  
  
 **private int** twice(**int** num) {  
 **return** 2 \* num;  
 }  
  
 @Override  
 **public int** compareTo(Object o) {  
  
 **return this**.calcArea() - ((Rectangle)o).calcArea();  
 }  
}

Code Names (שם-קוד)

[שם-קוד](https://he.wikipedia.org/wiki/%D7%A9%D7%9D-%D7%A7%D7%95%D7%93_(%D7%9E%D7%A9%D7%97%D7%A7)) הוא משחק חברה מוכר שיצא לאור בשנת 2015.  
במסגרת התרגילים במהלך הקורס נפתח גרסה מורחבת של המשחק אשר תכיל כמה שדרוגים והרחבות על גבי המשחק הבסיסי.

## חוקי המשחק

מטרת המשחק היא לנחש מילים על פי הגדרות שונות

לוח המשחק (הקלאסי) מורכב מ 25 מילים שונות (המשתנות בכל משחק ומשחק).   
במשחק משחקים 2 צוותים. לכל צוות מוקצים מספר מילים משלו.   
מטרתו של כל צוות היא לנחש ראשון את כלל המילים המשוייכות אליו.  
הניחוש מתבצע בעקבות הגדרה הקשורה בדרך כזו או אחרת לחלק מהמילים על הלוח.   
באמצעות ההגדרה על שאר חברי הצוות לנסות ולהבין אילו מהמילים על גבי לוח המשחק אכן "קשורות" להגדרה זו ומשוייכות לצוות שלהן.

כל צוות מורכב ממספר שחקנים הנחלקים לתפקידים הבאים:

1. "המגדיר" (נקרא גם spy master, רב-המרגלים): שחקן אחד בלבד בצוות נבחר להיות זה שנותן את ההגדרות.   
   המגדיר רואה את כלל הפרטים של לוח המשחק. הוא יודע להגיד אילו קלפים שייכים לכל צוות (כולל אילו קלפים שייכים לצוות האחר).   
   עליו לחשוב על הגדרה אשר תרמוז לחבריו לצוות לגבי מילות הצוות הנקשרות להגדרה זו. בנוסף עליו לספק מספר שאומר כמה מהמילים אכן מקושרות להגדרה המדוברת.
2. "המנחשים": שאר השחקנים החברים בצוות הם אלה המנחשים אילו מילים קשורות להגדרה שניתנה.   
   המנחשים יכולים לדבר בינם לבין עצמם ולהתייעץ, אולם אסור בתכלית האיסור "למגדיר" להיות ער לשיחתם או לרמוז להם בדרך זו או אחרת אם הם "בכיוון" או אם לאו. חברי הצוות המתחרה גם הם שומעים את הדיונים של הצוות שזהו תורו וחשופים להגדרה שנתן המגדיר של הצוות.  
   המטרה של המנחשים היא לנחש אילו מהמילים על גבי הלוח משוייכות לצוות שלהם על פי ההגדרה שניתנה ע"י "המגדיר". הם יכולים לנחש לכל היותר את כמות המילים שהמגדיר מסר להם, אולם יכולים גם להחליט כי הם לא רוצים לנחש את כל המילים האפשריות בתור מסויים.

אחת מהמילים על גבי הלוח היא "מילה שחורה".   
"המגדירים" יודעים את זהותה. שאר השחקנים במשחק לא.  
במידה ואחד הצוותים מנחש את ה"מילה השחורה" ובוחר בה – המשחק נגמר והצוות השני מנצח מיידית.

## מהלך המשחק

לצוות הראשון מוקצות 9 מילים מתוך הלוח. לצוות השני מוקצות 8 מילים מתוך הלוח. יש מילה שחורה אחת ו 7 מילים נטרליות.  
בכל תור ישחק רק צוות אחד. ה"מגדיר" של הצוות ייתן הגדרה ואת מספר המילים האמורות להתאים להגדרה זו.   
שאר חברי הצוות צריכים לנחש את המילים המתאימות להגדרה (ולכל היותר את כמות המילים המקסימלית על פי המגדיר)  
עבור כל מילה שניחשו:  
אם ניחשו מילה השייכת לצוות שלהם – הצוות קיבל נקודה.   
אם ניחשו מילה השייכת לצוות המתחרה – הצוות המתחרה קיבל נקודה.  
אם ניחשו את ה"מילה השחורה" – המשחק נגמר והם מפסידים.  
אם ניחשו מילה רגילה שאינה משוייכת לאף צוות – לא קרה כלום.

## הרחבות

במסגרת התרגילים בקורס נרחיב את המשחק ונאפשר כמה גרסאות שונות:

* לשחק ביותר מ 2 צוותים
* לשחק עם יותר מ 25 מילים
* לשחק עם יותר מ"מילה שחורה" אחת
* הקמת שרת המאפשר ריכוז של סוגי משחקים ומנהל שחקנים המשחקים בינהם באמצעות צ'ט.

## הנחיות ספציפיות למימוש תרגיל שם-קוד

1. **המטרה היא לבנות מנוע מערכת גנרי, כזה שידע לקבל את הפרטים לגבי המשחק (הגדרת המילים, שיוכם לצוותים, מילה שחורה וכו') מתוך קובץ נתונים בפורמט XML (עבודה עם XML'ים תילמד במהלך הקורס כמובן).**

**מנוע המערכת הגנרי ילך וישתכלל במהלך התרגילים, בהתאם לפיצ'רים השונים. כך תוכלו לחוות מהלך שלם של מוצר החל מרעיון קטן במימוש בסיסי וכלה במנוע מע' המניע אפליקציית ווב שלמה.**

1. **כחלק מהמע' תצטרכו לחשוב ולבחור לבד את מבני הנתונים השונים שישרתו את הצרכים של דרישות המע'. זהו לא קורס במבני נתונים או באלגוריתמים, ומבני הנתונים/אלגוריתמים שתבחרו לממש לא חייבים להיות היעילים ביותר או האופטימליים. מספיק שהם יעבדו בצורה נכונה (ללא טעויות) ובזמן סביר.**
2. **ממשקי המשתמש השונים יפעלו מול מנוע המע' אשר יפותח מתרגיל לתרגיל בהתאם לדרישות.   
   המנוע יכיל, בין היתר, את הגדרות המשחקים**
3. **כל הממשקים הפונים אל המשתמש החיצוני ומציגים לו מידע ו/או מבקשים ממנו מידע מספרי, חייבים להיות מבוססי ספירה המתחילה מ 1. גם אם פנימית אתם מממשים את מי מהרכיבים כמערך או רשימה (אז בסיס הספירה מתחיל מ 0) – עליכם להקפיד ולודא כי כלפי חוץ "תדברו" אך ורק במונחים של בסיס 1.**
4. **המע' כולה תתואר בשפה האנגלית בלבד, עם ממשק משתמש המתנהל משמאל לימין (במקומות הרלבנטים)**
5. **המע' כולה תיכתב ותורץ בסביבת העבודה של ג'אווה 8.**

תרגיל 1 – מימוש **שם-קוד** כאפליקציית Console (% 35) - הגשה: 21.5.24

## פרטים יבשים

צפי תחילת עבודה: **21.4.24** תאריך הגשה: **21.5.24**

צפי זמן לביצוע: 4 **שבועות** ציון אפשרי מקסימלי: 105

משקל התרגיל: **35%** קושי: **סביר**

**מטרת התרגיל העיקרית**

1. הקמת מנוע המע' הבסיסי
2. יצירת ממשק console לתפעול המשחק

## דרישות

1. בתרגיל זה תקימו את תשתית העבודה הראשונית והבסיסית של שם-קוד.   
   את התשתית "תתפעלו" באמצעות ממשק console פשוט המציג תפריט פקודות אשר דרכו מפעילים את המע'.  
   בתרגיל תממשו את היכולת לקלוט תיאור של משחק מתוך קובץ XML, כמו גם לתפעל ריצה של משחק.   
   כל אלה יישבו בתוך "מנוע" המערכת, אשר ידע להגיב לכל פנייה המגיעה משכבת ממשק המשתמש, לעבד את הקלט ולהחזיר פלט רלבנטי.
2. יש לוודא תקינות קלט כחלק מכל אינטרקציה עם המשתמש, ובכל מקום שבו זה רלוונטי:

אם אתם מצפים לקבל מספר – לא לקרוס כי הכניסו לכם בטעות (או בכוונה) טקסט וכו'.

בכל מקרה של תקלה יש להיות מאוד ברורים במסר שמעבירים חזרה למשתמש: מה קרה ? מה הייתה מהות התקלה ? היכן שזה רלבנטי, איך לתקנה וכו'.

חישבו איך להיות ידידותיים למשתמש ולעולם אל תניחו כי מי שמשתמש באפליקציה שלכם הוא מתכנת בעצמו או מישהו שמגיע מהתחום ו"מכיר" איך דברים עובדים לבד.

1. **אין צורך** להשתמש בצבעים שונים במהלך תרגיל זה בעת ההדפסה ל console.

יתרה מזאת, ישנו צפי רב (ניסיון מהסמסטרים הקודמים) כי ניסיון לעשות כן תוך שימוש בספריות צד שלישי קורס אצל הבודק,

מעוות את כל תצוגת המסך וגורם לחוסר יכולת לבדוק את ההגשה.

גם אם בדקתם את זה אצלכם וזה עבד.

גם אם בדקתם במחשב של השכנה וזה עבד.

כמו כן **אין** לנקות את המסך בין פקודה לפקודה.

1. עליכם לכתוב ממשק משתמש בתצורת console.

ממשק המשתמש יכיל סט סופי של פקודות שדרכן ניתן יהיה להפעיל את המערכת.

אחרי הצגת תפריט הפקודות יש לחכות לקלט מהמשתמש באשר לפעולה אותה הוא רוצה לבצע. לאחר ביצוע הפעולה (שאולי תגרור בקשת קלט נוסף מהמשתמש) יש להציג את הפלט החוזר ממנה (לכל פקודה יש פלט החוזר ממנה) ואז להציג שוב את התפריט וחוזר חלילה.

**שימו לב**:

* ישנן פקודות שאין הגיון לבצע אותן אם לא קדמו להן פקודות אחרות. במידה וזה קורה יש להציג הודעת שגיאה רלוונטית למשתמש ולאפשר את המשך פעילות המע'.
* בכל המקומות שבהם מציגים "רשימות של דברים" וצריך לאפשר למשתמש לבחור פריט(ים) מרשימה – יש לאפשר בחירה זו ע"י הצמדת מספר לכל אחת מהאפשרויות ולאפשר לו לבחור על פי המספר המזהה של האפשרות מהרשימה (או באמצעות כמה מספרים במקומות הרלבנטים). המספרים יתחילו מ 1 (לא מ 0)  
  **אין להניח** שהמשתמש הולך להקליד לכם מלל חופשי של תיאור האפשרות !

להלן רשימת הפקודות שיש לתמוך בהן:

1. קריאת קובץ פרטי המע'

פקודה זו טוענת את פרטי המערכת מתוך קובץ נתונים בפורמט XML.

קבצי דוגמא מתאימים הועלו מבעוד מועד לאתר הקורס ואתם מוזמנים להורידם ולבחון אותם בהתאם.   
(אתם מעודדים לייצר לעצמכם קבצי בדיקה נוספים כדי לבדוק את המע' בצורה יסודית וטובה יותר כאוות נפשכם).

יש לבקש מהמשתמש נתיב מלא לקובץ ה XML אותו הוא רוצה לטעון למע'.   
הנתיב יכול להכיל רווחים בתוכו (למשל "program files") ויש לוודא כי הדבר לא מכשיל אתכם (ולא שהוא אמור).   
מובטח לכם (ואין צורך לוודא זאת) כי הנתיב יכיל רק אותיות באנגלית (לא ג'יבריש של אותיות בעברית וכו')

הקובץ יהיה קובץ XML שפרטיו וחוקיו המפורטים מובאים [בנספח א'](#appendix_C) לתרגיל זה.   
אתם מצופים לעבור על פרטים אלה ולהתייחס אליהם כחלק אינטגרלי מהגדרת התרגיל.

עליכם לוודא בדיקת קלט לקובץ ה XML ולוודא כי הקובץ מכיל מידע תקין ואמין.

(מובטח כי הקובץ יהיה תקין schema-wise אבל לא בהכרח תקין application-wise...)

בפרט יש לוודא את הפרטים הבאים:

1. הקובץ קיים, והוא מסוג XML (די לבדוק לשם כך כי הוא נגמר בסיומת .xml)
2. ה cards-count קטן (>) מכמות המילים האפשריות של המשחק
3. ה black-cards-count קטן (>) מכמות המילים האפשריות של הקלפים השחורים
4. כמות הקלפים של שני הצוותים קטנה (>) מסך קלפי המשחק
5. rows X columns צריך להיות גדול או שווה בדיוק לכמות הקלפים על השולחן (צירוף של כמות קלפי המשחק והקלפים השחורים)
6. שמות הצוותים ייחודיים

במידה והקובץ לא תקין יש לדווח זאת למשתמש בצורה ברורה כך שניתן יהיה להבין מה לא תקין בקובץ.   
אין לקרוס על exception במידה וקובץ מתגלה כאינו תקין ; יש לאפשר למע' להמשיך לפעול במצב זה.

(כחלק מבדיקת התרגיל יטענו למערכת קבצים שאינם חוקיים כדי לבדוק מהי התגובה).

במידה והקובץ נמצא תקין – יש לטעון את פרטיו למע' ולדווח על כך שהקובץ נמצא תקין ונטען במלואו למשתמש.

**דגשים:**

* + - יש לאפשר למשתמש לטעון כמה קבצים אחד אחרי השני (כלומר להפעיל את הפקודה כמה פעמים רצוף).

כל קובץ תקין "דורס" לחלוטין את כל פרטי הקובץ (התקין) שהיה טעון לפניו במע' (ככל שהיה כזה).

כל נסיון טעינה של קובץ תקול לא דורס את פרטי הקובץ (התקין) האחרון שהיה במע' (ככל שהיה כזה)

* + - פקודה זו מוצגת ומאופשרת תמיד. אפשר לבחור בה בכל רגע נתון במע'.

1. הצגת פרטי משחק

פקודה זו תציג את פרטי המשחק כפי שנקראו מקובץ ה XML.  
יש להציג את הפרטים הבאים:

1. כמות המילים האפשריות לבחירה (כמות המילים ב Game-Words)
2. כמות המילים השחורות האפשריות לבחירה (כמות המילים ב Black-Words)
3. כמה מילים רגילות ומילים שחורות אשר ישתתפו במשחק
4. פרטי הצוותים. עבור כל צוות:
   1. שם הצוות
   2. כמות מילים

**דגשים:**

* אין לאפשר פקודה זו אם טרם נטען קובץ משחק בהצלחה

1. התחלת המשחק  
   פקודה זו מתחילה את המשחק.

בבטן המנוע עליכם לייצר מופע של המשחק באמצעות הנתונים וההוראות שנקראו מקובץ ה XML:

1. מתוך מילון המילים האפשריות למשחק יש להגריל מילים ככמות ה cards-count.  
   יש לוודא כי כל מילה תופיע לכל היותר פעם אחת. (שיעול Set שיעול)
2. מתוך מילון המילים האפשריות למילים השחורות, יש להגריל מילים שחורות ככמות ה black-cards-count.  
   יש לוודא כי כל מילה תופיע לכל היותר פעם אחת, כמו גם אסור לבחור מילה שכבר הוגרלה כמילת המשחק. (שיעול שיעול שיעול ...)
3. מתוך המילים שהוגרלו למשחק (שלב 1), יש להגריל אילו מהמילים תהיינה שייכות לצוות הראשון על פי כמות המילים הנדרשת לו על פי המוגדר ב cards-count הצוותי (וכך גם עבור הצוות השני)

לוח המשחק יוצג כטבלה. כמות השורות והעמודות של הטבלה מתוארת בקובץ תחת ECN-Layout.  
יש להגריל את מיקום המילים על הלוח. מרגע שהמיקום נבחר – הוא ישאר המיקום הקבוע של המילה על הלוח לכל אורך המשחק. (לדוגמא: אם המילה book מופיעה בשורה שניה עמודה שלישית – שם היא תישאר לכל משך המשחק הנ"ל)

לכל מילה יינתן מספר סידורי (1,2,3,...) רץ המתחיל מהתא השמאלי העליון בצורה אופקית ומתקדם אל התא הימני התחתון.  
  
הצגת הלוח תתאפשר בשני מצבים:

בכל מצב תוצג המילה והמספר הסידורי שלה מתחתיה בתוך [ ]

**מצב מוסתר**: מתאים לשחקנים "המנחשים"  
במצב זה רואים רק את המילים ללא השיוך שלהם לצוותים השונים, וללא זהות המילים השחורות.   
במידה והמילה התגלתה יש לציין V תחתה.  
במידה ומילה כבר התגלתה והיא שייכת לצוות – יש לכתוב את שם הצוות אליו היא שייכת בסוגריים מתחתיה.  
במידה והמילה כבר התגלתה והיא מילה שחורה – יש לכתוב (BLACK) מתחתיה.

**מצב גלוי**: מתאים לשחקנים "המגדירים"

במצב זה השחקן רואה את כל הלוח במלואו:

מתחת לכל מילה יצויין V או X כדי לדעת אם כבר התגלתה או לא.  
לידו יצויין בסוגריים שם הצוות אליו היא שייכת. אם זו מילה שחורה יצויין בסוגריים (BLACK)

דוגמא לנראות של מילה:   
יש להציג את לוח המילים כשהוא "מוסתר".  
מרגע הפעלת פקודה זו המשחק הופך ל"פעיל".

kombat  
[23] V

encapsulation

[1]V (T1)

morning  
[2] X (BLACK)

**דגשים:**

* אין לאפשר פקודה זו אם טרם נטען קובץ משחק בהצלחה
* ניתן להפעיל פקודה זו שוב ושוב והיא למעשה מאפסת את המשחק הפעיל המתקיים בבטן המע' (ככל שמתקיים כזה) ומייצרת משחק חדש (על פי ההוראות מעלה)

1. ביצוע תור

פקודה זו אחראית על ביצוע תור אחד במשחק.  
  
בראשית פקודה זו יש לציין את שם הצוות שזהו תורו ואת הניקוד הנוכחי שלו.   
את הניקוד יש להציג בתצורה של Y / X: כמה מילים התגלו מתוך כמה מילים סה"כ מוקצים לצוות. (דוגמא: 4/9: התגלו 4 מילים מתוך 9)  
  
לאחריו יש להציג את לוח המשחק כשהוא "גלוי", כדי שה"מגדיר" ידע לנסח את הגדרתו.  
יש לבקש מהמשתמש את ההגדרה של התור הנוכחי. הגדרה זו תנתן באמצעות מחרוזת תווים רגילה באנגלית.

לאחר מכן יש לבקש את מספר המילים שהגדרה זו רלוונטית עבורן

כעת מגיע תורם של המנחשים:  
יש להדפיס מחדש את הלוח, הפעם במצב "נסתר".  
יש לבקש מהמשתמש להכניס את ניחושי המילים. בחירת המילה מתבצעת אך ורק באמצעות מספרה הסידורי  
עבור כל ניחוש תתבצע הבדיקה בבטן המנוע לגביו, ובסיומה יודפס מחדש הלוח (במצב "נסתר") כשהוא מכיל את התוצאה עבור המילה שזה עתה נוחשה.  
  
כמו כן יש להחזיר שורת הסבר לגבי תוצאת הניחוש:

* האם היא אכן שייכת לצוות שזו תורו (ולכן זיכתה בנקודה את הצוות)
* האם היא שייכת לצוות המתחרה (ולכן זיכתה בנקודה את הצוות המתחרה)
* האם זו מילה שחורה (ולכן הצוות הפסיד והמשחק נגמר)
* האם זו מילה ללא שום שייכות (ואז לא קרה כלום)

יש לציין את הניקוד של הצוות לאחר סיום ביצוע התור (באותו פורמט שהצגתם את הניקוד בהתחלה: Y / X)  
חוזרים על תהליך זה כל עוד המשתמש לא סיים לנחש את כל המילים שנדרש להן על פי החלטת "המגדיר".

המשתמש יכול לבחור לסיים את התור גם אם לא ניחש את כל כמות המילים כפי שהוגדרו לו לתור זה.  
יש לאפשר לו להחליט לסיים את התור אם ירצה.

בסיום כל תור ייתכן והמשחק יסתיים, אם בגלל פגיעה במילה שחורה, או סיום זיהוי כלל המילים של הצוות וכו'.

במידה וזה קורה יש לציין זאת במפורש ולהכריז על זהות הצוות המנצח והסיבה לכך.   
כמו כן במקרה זה המשחק הסתיים ואיננו "פעיל" יותר.  
  
כך או אחרת בסיום פקודה זו חוזרים חזרה לתפריט הראשי.

**דגשים:**

* אין לאפשר פקודה זו אם המע' לא נמצאת במצב של משחק פעיל.
* הפעלה חוזרת של פקודה זו למעשה מעבירה את התור לצוות הבא ובכך מקדמת את המשחק

1. סטטוס משחק פעיל

פקודה זו תאפשר להציץ ולקבל את מצב המשחק הפעיל המתקיים במע'.

במסגרתה יש להציג:

* לוח במצב גלוי
* מידע על הצוותים. עבור כל צוות:
  + שם
  + מצב הניקוד שלו (בפורמט של Y / X)
  + כמה תורות ביצע כבר
* שם הצוות שהתור הבא הוא שלו

**דגשים:**

* אין לאפשר פקודה זו אם המע' לא נמצאת במצב של משחק פעיל.

1. יציאה מהמערכת

פקודה זו מסיימת את פעולת התוכנית.

## חלוקה למודולים

בתרגיל זה **חובה** לייצר (לפחות) 2 מודולים (מהם תפיקו בהמשך 2 jar'ים):

* + 1. ממשק ה ui, המציג את התפריטים השונים, אחראי על קליטת קלט מהמשתמש והחזרת הפלט למשתמש.

שימו לב זהו המודול "האקטיבי", המניע את כל המע'. הוא זה האחראי על פנייה ותפעול מנוע המערכת.

כפועל יוצא, כל ההדפסות של מידעים למשתמש (System.out.println) מתבצעות **אך ורק** מתוך מודול זה ; במודול זה יושבת מטודת ה main ; מודול זה אחראי על לולאת תפעול המע' העיקרית, הצגת התפריטים, איסוף הקלט מהמשתמש, הצגת הפלטים למשתמש וכו'.

* + 1. מנוע המשחק, האחראי על קבלת הפקודות (ממודול ה ui), ביצועם והחזרת פלטים מתאימים.

שימו לב שמודול זה "פסיבי", והוא **רק** מגיב לבקשות ולפקודות המתקבלות ממקורות בלתי ידועים לו (בתרגיל זה מודול #1). בתרגילים הבאים מקורות נוספים יפנו אליו לקבלת מידע וחשוב מאוד להקפיד על כך **שמודול זה אינו מכיר/מודע למי פונה אליו.**בתוך מנוע המשחק למעשה מתקיימת כל הלוגיקה וניהול המשחק. מנוע המשחק הוא שיודע את המצב של הצוותים והלוח, של מי התור הבא, איך מידע על מבנה המשחק שהתקבל מקובץ ה XML וכו'.

## איך מתחילים ? (המלצה...)

אם אינכם מכירים את המשחק מבעוד מועד – אני ממליץ לשחק אותו. אפשר לקנות את גרסת משחק הקופסה או לשחק אותו גם [online](https://codenames.game/) עם חברים. היכירות עם המשחק והתנהלותו וחוויתו על בשרכם תעזור לכם להבין לעומק את המבנה, הקצב ואת אופן ההתנהלות.

צרו פרוייקט חדש ב Intellij שישמש כפרוייקט האב לכלל התרגילים.   
בתוך הפרוייקט צרו מודול נפרד עבור מנוע המע'. המנוע יכיל את כלל החלקים הנדרשים לתפעול המע' (הגדרת לוח המשחק, מימוש לולאת המשחק, היכולת לטעון קובץ XML וכו').  
המנוע יחשוף סט של יכולות (החופפות לפקודות השונות בתפריט), יחזיק מופע של המשחק הפעיל ויתווך את הפקודות השונות הלוך וחזור. התחילו בבניית תשתית המנוע (מומלץ לעטוף את המנוע בממשק (interface) ולהתחיל לתרגל (ולהתרגל) לחווית העבודה הנ"ל.)  
  
התחילו בבניית ייצוג של מבנה המשחק (המידע שמצוי בקובץ ה XML). הוא ישמש אתכם כבסיס ליצירת המשחק עצמו ולאחר מכן להתנהלותו. תוכלו להתחיל להרים את המע' גם לפני קריאת הנתונים מקובץ ולוודא כי היא מתחילה לעבוד ע"י זה שתטענו מידע קבוע דמיוני לאובייקט זה.  
ממשו את פקודה 2 שתאפשר לכם לראות כי בידכם המידע הנדרש ואתם מציגים אותו כראוי.  
  
לאחר מכן התחילו לעבוד על בניית והגדרת משחק פעיל. הגדירו את המילה ושאר הפרטים הרלוונטים לה, נסחו את מבנה הנתונים שמתאר את הלוח ואת הצוותים ואת הקשרים בינהם. עכשיו ממשו את פקודה 3 שתאפשר למעשה יצירת משחק פעיל.  
עיבור ליצירת לולאת המשחק הראשית ובהתאם לה את פקודות 4 ו 5.

לסיום, צרו מודול נוסף שהוא מודול שכבת ה UI (console). זה המודול שבו תשב בסופו של דבר מטודת ה main הראשית שתתפעל את כלל המע'. זה המקום היחיד שבו מוצג פלט (System.out.pritnln) ונאסף קלט (scanner) מהמשתמש. זה המקום המכיל את לולאת התפריט הראשית המניעה את כלל המע'. שכבת ה UI תכיל הפנייה (reference) למופע המנוע (שבתוכו מכיל הפנייה למשחק פעיל, הגדרת המשחק וכו') וכך תוכל להעביר ולתרגם לו את הפקודות הנאספות מהמשתמש ולהציג חזרה את הפלטים החוזרים מהמנוע.

## בונוסים

|  |  |  |  |
| --- | --- | --- | --- |
| # | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | **שמירה וטעינה של המע'**  בבונוס זה המצב הנוכחי של המע' (כולל המצב של המשחק הפעיל שמתקיים בו, אם כזה קיים) נשמר לקובץ חיצוני (באיזה פורמט וטכניקה שתבחרו).   יש להוסיף פקודה שמאפשרת לשמור את מצב המע' וגם פקודה המאפשרת לטעון מצב קיים (מקובץ שנשמר זה לא מכבר), וזאת בניגוד לטעינה רגילה מקובץ ה XML של התרגיל.  יש לאפשר למשתמש לבחור את הנתיב המלא כולל שם הקובץ (בלי הסיומת) שהוא היה מעוניין לשמור את המע' אליו (ולטעון אותה ממנו מאוחר יותר) | עם תכנון נכון זה אמור להיות משהו כמו 4-5 שורות... | 5 נקודות **(מעל ל 100)** |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. 2 jar'ים (לפחות) שהם כל הקוד שלכם, בצירוף קובץ אצווה (batch) שהפעלתו תריץ את התוכנית

(כלומר תבצע java -jar <class name> וכו').

1. קבצי קוד המקור של הפרויקט שלכם.
2. קובץ ההגשה יכיל גם קובץ **readme** שיכיל הסבר על המערכת, בחירותיכם השונות במקומות שבהם היו לכם בחירה, או כל דבר נוסף העולה על דעתכם שחשוב שהבודק ידע.
3. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות ותפקידם.
4. יש לכלול בקובץ ה **readme** גם פירוט של המגישים שם, ת.ז. ואי מייל זמין ורלבנטי (!!) – במידה ויהיה צורך ליצור קשר.
5. במידה ומימשתם את מי מהבונוס(ים) לעיל, ציינו את שם הבונוס שמימשתם בראשית קובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

הגשה באיחור, שאינה באישור, תבטל כל מימוש בונוס. אין להגיש באיחור בשביל להספיק לעשות בונוסים.   
תכננו את הזמן בהתאם.

תרגיל 22 – מימוש **שם-קוד** כאפליקציית Client - Server (% 55) – הגשה: 23.7.24

## פרטים יבשים

צפי תחילת עבודה: **23.6.24** תאריך הגשה: **23.7.24**

צפי זמן לביצוע: **4 שבועות +** ציון אפשרי מקסימלי: **110**

משקל התרגיל: 55**%** קושי: **ווא'עליה אלל'בחטי**...

**מטרות התרגיל העיקרית**

1. מימוש המשחק כאפליקציית client-server.
2. הוספת מנגנון של חדרי משחקים

## דרישות

1. בתרגיל זה נממש את שם-קוד כארכיטקטורת שרת-לקוח.  
   משתמש מסויים יוכל להעלות מספר קבצי משחק שונים (בפורמט של תרגיל 2) לבטן השרת. כל קובץ מגדיר סוג אחר של משחק הנבדל בכמות הקלפים, כמות הצוותים, הרכבם וכו'.  
   משתמשים יכולים להירשם למע' ולמעשה לבחון באיזה משחק הם רוצים להשתתף, ולהירשם כמשתתפים במשחקים על פי הגדרתם. כשנרשמו כל המשתתפים הנדרשים למשחק הוא מתחיל ומנוהל בחדר משחק. שם השחקנים השונים (המגדירים כמו גם המנחשים) יוכלו לתקשר אחד עם השני באמצעות צ'ט ולהתנהל בחווית משחק עצמאית לחלוטין.
2. השרת (tomcat) יכיל את מנוע המשחק ויחשוף את היכולות בדמות endpoints שונים.  
   מול השרת יעבדו 2 סוגים של לקוחות/יישויות:

**Administrator**ה Administrator (אדמין, בקיצור, בישראלית) הוא מנהל המע'. יש בדיוק administrator אחד בלבד.   
בין יכולותיו של האדמין:

* העלאת קבצים (קובץ בפורמט של תרגיל 2) המכילים הגדרות של משחקים.   
  הפעם כל קובץ המכיל הגדרת משחק מתווסף למנוע המע' (ולא מחליף את הקובץ הקיים, כפי שהיה בתרגיל 1).   
  כך יכול האדמין להעלות כמה קבצים ולהעשיר את המע' במגוון של משחקים.
* לקבל סקירה לגבי איזה משחק משוחק עכשיו ומה מצב המשחקים השונים במע'
* לקפוץ לחדר משחק שמתנהל "כצופה מן הצד".

**משתמש רגיל**

המשתמשים הרגילים הם השחקנים עצמם שרוצים להשתתף במשחקים.  
ברגע נתון יכולים להתחבר למע' ולהשתמש בה מספר רב של משתמשים.  
בין יכולותיו של המשתמש:

* סקירת המשחקים האפשריים במע'. קבלת מידע על מצב הרישום של כל אחד מהם.
* הירשמות למשחק בתפקיד "מגדיר" או "מנחש".
* השתתפות פעילה במשחק תוך החלפת מידעים באמצעות צ'ט עם שאר המשתתפים.

**Code-Names  
Server  
  
(Tomcat)**

**HTTP**

**HTTP**

Administrator  
(Client)

User  
(Client)

**…**

User  
(Client)

User  
(Client)

מילון מילים:  
במסגרת תרגיל זה רשימת המילים האפשריות למשחק ולמילים השחורות לא תתקבל עוד בגוף הקובץ, כי אם באמצעות קובץ מילון ייעודי אשר יכיל אוסף מגוון של מילים.  
במסגרת טעינת הגדרת משחק (קובץ ה XML) יש לטעון עכשיו קובץ טקסט נוסף (txt). המילים שבקובץ הן המילים שישמשו כבנק המילים הן עבור מילות המשחק והן עבור המילים השחורות.  
קובץ זה הוא קובץ המכיל בליל של מילים. רובן הגדול חוזר על עצמו. יש לקלוט ולעבד את הקובץ ולהפיק ממנו את רשימת המילים הייחודיות המתקיימות במסגרתו.   
עליכם לקרוא קובץ זה ולעבד את תוכנו. קחו בחשבון את הדברים הבאים:

* התו המפריד בין המילים שבקובץ הוא תו הרווח.
* יש לבצע הסרה של כל הרווחים המובילים/מסיימים של כל מילה (trim)
* יש לבצע הסרה של התווים הבאים מתוך תווי המילה: !@#$%^&\*)(-\_,.?

1. ריבוי צוותים ומגדירים:

בתרגיל זה תתמכו ביכולת לשחק שם-קוד עם יותר משני צוותים.

בנוסף הפעם יוכל צוות להכיל יותר מ"מגדיר" אחד, כך שלמעשה כמה חברי צוות יוכלו לחבור יחדיו ולחשוב על הגדרות קולעות לצוות המנחשים שלהם.  
המידע על כמות הצוותים, כמות המגדירים וכמות המנחשים בכל צוות מוגדר בקובץ ה XML.  
ינתנו פרטים לגבי כמות "המגדירים" וכמות "המנחשים" שנדרשים לו.  
כמו כן כל צוות יקבל צבע ייחודי שיאפיין אותו. המע' תתמוך בדיוק בשישה סוגי צבעים ולכן אפקטיבית יכולים להתקיים לכל היותר שישה צוותי משחק. (רשימת הצבעים האפשרית מוגדרת [בנספח א](#ex2_scheme)')  
במידה ואחד הצוותים סיים למצוא את כל המילים שלו – הוא מנצח במקום הראשון, אולם המשחק ממשיך להתנהל כל עוד יש עוד צוותים פעילים בו.

במידה ואחד הצוותים גילה מילה שחורה במסגרת התור שלו, הוא מפסיד ויוצא מהמשחק מיידית, אולם המשחק ממשיך כל עוד יש עוד צוותים אחרים.   
המשחק ייגמר כשיישאר צוות אחרון (אם כולם יצאו בגלל מילים שחורות) או שהוא האחרון שטרם מצא את המילים השייכות לו.

1. התנהלות המשחק:  
   ברגע שמשחק הועלה לשרת הוא מופיע כזמין לכלל המשתמשים הרגילים במע'.   
   על כל משחק יינתנו הפרטים המסבירים כמה צוותים נדרשים לו ומהו הרכבם (כמות המגדירים/מנחשים)  
   לכל משחק ינוהל סטטוס שבו הוא נמצא. כשמשחק מועלה למע' הוא בסטטוס PENDING – ממתין לרישום שחקנים.  
     
   משתמש רגיל יוכל לבחור להירשם להשתתף במשחק כ"מגדיר" או כ"מנחש".  
   עליו לבחור את אחד מהצוותים שהוא רוצה להצטרף אליהם ולהירשם למי מהתפקידים הנ"ל שהוא חפץ בהם ואשר עדיין נדרשים בצוות. ברגע שביצע רישום למשחק הוא עובר "לחדר המשחק" וממתין עד שהוא יתחיל.  
   ברגע שנרשמו כלל השחקנים הנדרשים לכלל הצוותים המוגדרים במשחק, המשחק עובר לסטטוס ACTIVE (פעיל).  
   בשלב זה כל השחקנים כבר נמצאים בחדר המשחק.

מכאן ואלך המשחק מנוהל "אוטומטית":  
כלל השחקנים יקבלו הודעה על זהות הצוות שזהו תורו. המגדיר(ים) של הצוות יראו את הלוח במצבו המלא ויוכלו לדבר בינהם בצ'ט שרק הם חשופים לו לגבי זהות ההגדרה שהם רוצים לתת. ברגע שמתקבלת ההגדרה וכמות המילים מאחד מהם שאר המשתתפים במשחק (כולל מצוותים אחרים רואים את ההגדרה ואת הלוח במצבו "החסוי".  
המנחשים של אותו הצוות צריכים להכניס את המילים שהם מנחשים. הם יכולים גם כן לצו'טט בינם לבין עצמם כדי להחליט מה הן המילים הנדרשות. המגדירים של אותו הצוות כמו גם שאר השחקנים של הצוותים האחרים יכולים לראות את התנהלות הצ'ט הנ"ל, אולם אינם יכולים להגיב שם.

לאחר שניתנו המילים שנבחרו (ע"י אחד מהמנחשים) ישוקפו תוצאות בחירותיהם לכלל המשתתפים במשחק מכל הצוותים.  
התור עובר אוטומטית לצוות הבא וכל המשתתפים מקבלים על כך מידע וחוזר חלילה.  
צוות שבטעות בחר מילה שחורה הפסיד ויוצא מהמשחק. הודעה מתאימה תועבר לכלל המשתתפים במשחק (כולל לצוות המפסיד, כמובן). כלל השחקנים שלו חוזרים למסך ריכוז המשחקים ויכולים להירשם למשחקים אחרים ככל שירצו.  
כל עוד יש צוותים פעילים במשחק – הוא ממשיך להתקיים עד להשגת ניצחון בדרך זו או אחרת.  
כאשר המשחק מסתיים יש להודיע על זהות המנצח לכלל המשתתפים ואו אז כולם חוזרים למסך ריכוז המשחקים.

1. חלק מפרטי המידע במע' צריכים להתעדכן בצורה אוטומטית עבור כל משתמש בצורה של Pull כפי שלמדנו בשיעור והודגם בדוגמא המסכמת (אם בשיטה של all או של delta fetching – כל מקרה לגופו).   
   אפשר לבצע את ה Pull בטווח זמן של עד 2 שניות, אולם זמן סביר ל pull הוא סדר גודל של חצי שנייה.
2. אין צורך לבצע שמירה של נתונים מעבר למופע הנוכחי של השרת. במילים אחרות כשהשרת יורד - כל הנתונים נעלמים.

לא שומרים את המשתמשים שנרשמו, המשחקים שהועלו וכו'.

## התממשקות עם ממשק גרפי

על מנת לאפשר התנהלות מסודרת (ונעימה) לאפליקציה בסדר גודל שכזה, כבר לא נוח לעבוד באמצעות ממשק console ותפריט טקסטואלי.  
על כן, סופק לכם ממשק גרפי desktop'י המכיל שכבת UI גרפית שתאפשר ניהול משחק מסודר.   
(עקבו אחר [נספח ג'](#appendix_A) לסקיצה מפורטת יותר של מסכי הממשק)  
שכבת ה UI שניתנה לכם אחראית אך ורק על תצוגת המידעים (היא מחליפה את שכבת ה console) אולם התהליך הלוגי המתרחש באפליקציית ה UI הינו באחריותכם ועליכם לכתוב אותו בהתאם.  
  
שכבת ה UI מגיעה בדמות קובץ Jar שניתן להפעלה עצמאית (java -jar CodeNamesApp.jar).  
אינכם מצופים או נדרשים להיכנס לקוד של שכבת ה UI ו/או להבין אותה בשום צורה ואופן.  
  
על מנת להתממשק ולהיות מסוגלים להתחבר אל תוך שכבת ה UI הוגדרו מספר ממשקים (interface) המכילים את כל הפעולות ששכבת ה UI צריכה שיספקו להן מענה כדי לתפעל את עצמה בצורה מלאה.  
קבצי ה Interface הנ"ל מסופקים לכם בדמות jar נוסף: CodeNamesUIApi.jar.  
הממשקים מכילים את הפעולות השונות שתדרשו לממש. הפעולות הנ"ל למעשה יהוו את הקשר בין שכבת ה UI לשכבת השרת. בתוכן למעשה תייצרו את בקשות ה http ו/או תעבדו את המידעים החוזרים מהן כדי למסור אותן חזרה לשכבת ה UI.  
המטודות מתועדות היטיב וחלק מהתרגיל הוא ההתמודדות עם קריאת והבנת תיעוד המטודות ולעבוד איתו כדי להבין אותו כהלכה.  
(קיבלתם jar נוסף שנקרא CodeNamesUIApi-sources.jar המכיל את קוד המקור עם כל התיעוד של ה Interface'ים השונים. יש לטעון אותו ב Intellij וכך תוכלו לראות את כל התיעוד בצורה מסודרות)  
  
חלק מהמטודות המוגדרות ב Interface'ים מצפות לקבל ו/או מחזירות מבני נתונים מוגדרים היטיב, אותם ואיתם שכבת ה UI מצפה לקבל ויודעת לעבוד. אלה הם למעשה DTO'ים קטנים המאפשרים לקבל ולהחזיר את שלל המידעים הנדרשים כדי להציגם ב UI בצורה מסודרת.   
ייתכן וה DTO'ים האלה זהים לאלה שכבר יש לכם, אך סביר יותר שהם שונים בעליל.  
כך או אחרת זה לא משנה: כדי לעבוד עם שכבת ה UI עליכם להתאים עצמכם לדרישותיה, הווה אומר כי אתם מקבלים ממנה את המידע באמצעות ה DTO'ים הנ"ל (ומשם יכולים להמיר אותם ל DTO'ים שלכם) ו/או בונים ומחזירים לה את ה DTO'ים שהיא מצפה לקבל מהתשובות שאתם מחזירים מהשרת, אשר סביר שיכילו מבני נתונים המסוג שאתם כבר הגדרתם.  
ה DTO'ים הנ"ל מוגדרים גם הם בתוך CodeNamesUIApi.jar.

ככלל, אפשר למדל את הפעולות בשכבת ה UI לשני סוגים:  
  
**פעולות אקטיביות**:

אלו הן פעולות שה UI יוזם בעקבות פעולת משתמש אקטיבית, למשל לחיצה על כפתור או בחירה של נתון.

בשל מגבלות הנוגעות לתפעול של שכבת ה UI, כלל הפעולות בממשק נחוות מבחינתה כפעולות אסינכרוניות. כלומר גם אם אתם מבצעים את הפעולה מיידית – עדיין מבחינת ה UI הוא לא יכול "להמתין" לתשובתכם ויצטרך לקבל אותה בדרך אסינכרונית.   
שימו לב כי רוב הסיכויים שמרבית הפעולות יתורגמו גם כך לקריאות http כאלה ואחרות מול השרת ואלה צפויות לקחת פרק זמן מסויים עד שיחזרו מהן תשובות ועל אתם כבר נמצאים במצב "אסינכרוני" שבו ייקח זמן מרגע הגשת הבקשה עד לקבלת תשובתה.  
  
הדרך למדל חוויה זו בהתנהלות מול שכבת ה UI היא באמצעות callback'ים:  
אוסף של פונקציות קטנות (lambda, מישהו ?) שעליכם לקרוא להן ברגע שהתשובה הנדרשת לפעולה מתקבלת.   
כל פעולה יכולה פוטנציאלית גם להיכשל, מכל סיבה שהיא: אם כשלון לוגי (למשל טעינת קובץ תקול) ואם כשלון טכני (למשל לא מצליחים ליצור קשר עם השרת). על מנת למדל זאת תקבלו גם error callback ובמידה ונתקלתם במשהו שאתם חווים אותו ככשלון – תוכלו להעביר את המידע המילולי הנדרש לשם כך דרך ה callback ושכבת ה UI תציג אותו כתיבת Alert כדי לידע את המשתמש (והבודק) על סוג התקלה שקרתה.

היות וכך, כל מטודה ב Interface'ים מקבלת (לפחות) 2 ארגומנטים:  
 **success**: למבדה מסוג של runnable או consumer אשר עליכם לקרוא לה במידה והפעולה הסתיימה בהצלחה.  
במידה ומשתמשים ב Consumer הוא יוגדר לקבל מבנה נתונים כלשהוא המכיל את התשובה של הפעולה ונתונים שה UI צריך לקבל חזרה (אחד ה dto'ים שה UI עובד איתם)  
חובה לקרוא למטודת ה success בכל מצב שבו הבקשה נגמרה בהצלחה, שכן היא חיונית לפעולתה התקינה של שכבת ה UI.  
  
  
  
**failure**: למבדה מסוג של Consumer<String> אשר עליכם לקרוא לה במקרה של תקלה. יש להעביר לה ארגומנט עם מידע מילולי על מהות וסיבת התקלה. ככל שהמידע יהיה מפורט יותר כך יקל עליכם להבין מה התקלה שקרתה. שכבת ה UI מבחינתה לוקחת את המידע AS IS ומציגה אותו על המסך בדמות תיבת דיאלוג שהמשתמש יהיה חייב ללחוץ על OK כדי להשתחרר ממנה ולהמשיך את פעולת המע'.  
  
**פעולות ריאקטיביות (תגובתיות)**:  
אלו הן פעולות המתרחשות בבטן השרת אותם הוא רוצה "לדחוף" לקליינטים השונים ולהעביר להם את המידע. בפעולות שכאלה המשתמש לא יוזם כלום, אולם עדיין צריך לקבל את הנתונים תוך כדי תנועה (לדוגמא: רשימת המשחקים במע', מידע על ההגדרה שנתן המגדיר, הניחוש שביצעו המנחשים, זהות הצוות המנצח במשחק וכו').  
  
במקרים שכאלה זו תהיה אחריותכם לבצע את הלוגיקה שאחראית לייבא את המידע בצורה אוטומטית מהשרת (בדרכים של Pull שנלמדו בכיתה) ורק לעדכן את ה UI בהם.  
על מנת למדל פעולות שכאלה, חלק מהפעולות ב Interface'ים השונים הן כאלה המקבלות רק פונקציות עצמאיות (למבדות) שעליכם לשמור ולקרוא להן כל אימת שקיבלתם מידע אוטומטי שכזה.

עליכם לממש את ה Interface'ים השונים שקיבלתם על מנת שהממשק הגרפי יעבוד בהצלחה.  
הפלט שלכם יהיה Jar עצמאי והוא יכיל את כלל מימושי הממשק. אתם רשאים, כמובן, להשתמש בכל ספריית צד שלישי נוספת שתדרשו לה במסגרת מימושים אלה (okhttp, gson וכו')

שכבת ה UI מכילה את מימוש שתי סוגי היישויות הנדרשות (אדמין ומשתמש רגיל).   
היא מכילה סה"כ 3 מסכים:

* מסך לוגין שבו יש לבחור את סוג הקליינט (אדמין או משתמש רגיל).
* מסך המכיל ריכוז המידע לגבי המשחקים המנוהלים במע' ופרטי הרישום שלהם.
* מסך המאפשר לנהל משחק אחד אקטיבי

## אפליקציית Administrator

אפליקציה זו מותאמת לקליינט מסוג Admin. ברגע נתון ניתן להפעיל בדיוק מופע אחד שלה.  
בניסיון להפעיל את האפליקציה ולבחור תפקיד אדמין בפעם השנייה יש למנוע זאת ולהעביר הודעת שגיאה מתאימה שתצא למשתמש.   
אם מורידים את אפליקציית האדמין ומעלים אותה מחדש – המידע שהוגדר באמצעותה נשמר בשרת וכמובן מוצג כמות שהוא לאדמין החדש.

המשתמש (האדמין) יוכל לבחור להעלות קובץ מהפורמט של תרגיל 2 (עיקבו אחר [נספח א'](#ex2_scheme) כדי לראות את השינויים יחסית לסכמה של תרגיל 1). הקובץ נבחר ממחשב המשתמש ומועלה אל השרת. העלאת הקובץ יכולה (וצריכה) להתבצע בהתאם לדוגמא שנלמדה בכיתה.   
**אין שום צורך** להשתמש בספריות צד שלישי (apache commons וכיוצב') בשביל להעלות את הקובץ.   
**אין שום צורך** לשמור את תוכן הקובץ בשרת (ויש להימנע מכך בכל מחיר !). לשרת (של הבודק) אין הרשאות לכך ואתם תקרסו !

בנוסף לקובץ ה XML, אפליקציית האדמין תאפשר לבחור גם קובץ שישמש כמילון מילים.  
טעינה של משחק לשרת עוברת דרך העלאת 2 הקבצים בלבד. לא ניתן להעלות רק קובץ משחק (xml) או רק קובץ מילון (txt).

לכל משחק מתווסף גם שם ייחודי. אין לאפשר להעלות משחקים בעלי אותו השם.

כשתוכן קובץ המשחק מגיע לשרת, יבוצעו בו בדיקות הקלט כפי שבוצעו במהלך תרגיל 1.

1. לכל משחק יש שם ייחודי משלו
2. לכל צוות יש צבע ייחודי משלו
3. כמות ה Definers וה guessers של כל צוות גדולה או שווה ל 1 (>= 1)

באם הקובץ תקול – הודעה מתאימה הכוללת את סיבת התקלה צריכה להופיע לאדמין לאדמין (ולאדמין ולאדמין בלבד), והקובץ **אינו** מוכר כתקין בצד השרת.   
אם הקובץ תקין – הודעה מתאימה צריכה להגיע לאדמיןלאדמין.  
הגדרות המשחק התקינות יועלו ויתווספו לרשימת המשחקים המוגדרת בשרת.   
רשימת המשחקים האפשרית מוצגת במסך הראשון עם כלל הפרטים על כל משחק ומשחק.   
  
  
האדמין לא יכול להשתתף במשחקים, אולם הוא יכול לבחור להיכנס למשחק שמתנהל כ"צופה מן הצד".

## אפליקציית משתמש

אפליקציה זו מותאמת לקליינט מסוג משתמש רגיל – שחקן.

בעמוד הלוגין כל שחקן יידרש להכניס את שמו. השם הוא ייחודי ואין לאפשר רישום של משתמש בשם שכבר קיים.  
  
לאחר רישום מוצלח השחקן יגיע גם הוא למסך ריכוז המשחקים, אשר בו הוא יראה את כל המשחקים המתקיימים במע'.   
עמוד זה צריך להתעדכן בלייב ולשקף את המצב ואת מלאי המשחקים במע' (בשרת) כך שאם האדמין מוסיף עוד משחק, או אם פרטים של משחק מסויים משתנים תו"כ תנועה – כלל המשתמשים הרגילים יקבלו עדכונים אלה (יש לעבוד ב Pull כפי שלמדנו בכיתה).

בחירת משחק מסויים תגלה את הפרטים השונים לגביו ותאפשר גם להירשם להשתתף במשחקים שבמצב PENDING.  
ניתן יהיה לראות את כל המידע על הצוותים ובתוך כל צוות לבחור להירשם לתפקיד ה DEFINER (אם עוד יש מקום פנוי) או ה GUESSER (אם עוד יש מקום פנוי).

מרגע שבוצעה בחירה להירשם למשחק, המשתמש יועבר אוטומטית למסך האחרון: מסך משחק פעיל.

כל עוד המשחק נמצא במצב PENDING – המשתמש ממתין במסך זה עד להגעת כל השחקנים הנדרשים.  
מרגע שנרשמו כלל השחקנים הנדרשים המשחק מתחיל להתנהל באופן אוטומטי כפי המתואר [בנספח ג'](#appendix_A)

בסיום המשחק המשתמש מועבר חזרה למסך ריכוז המשחקים להמשך הפעילות.

## חלוקה למודולים

עבור השרת, יש לייצר מודול נוסף שממנו ייבנה war המכיל את כל ה jar'ים הנדרשים (Jar של מנוע המע' ; gson.jar ; אולי אחרים ?).

בנוסף, יש לייצר מודול חדש שבו יהיה עליכם לממש את ה Interface'ים הרלוונטים כדי שתוכלו לשלבם בממשק המשתמש.  
במודול זה תממשו גם את הממשקים הרלוונטים לאדמין וגם את אלה הרלוונטים למשתמש הרגיל.  
בהחלט סביר והגיוני שידרשו תלויות שילוו את המימושים שלכם (למשל http client; gson ועוד).  
ממודול זה יופק jar שיתלווה ל Jar של ממשק ה UI וביחד יהוו את אפליקציית הקליינט המלאה.

## איך מתחילים ?

אני ממליץ להתחיל במעבר יסודי על הדוגמא המסכמת של הקורס.   
היא תקנה לכם שליטה בסיסית בדרך העבודה עם http client ובעבודה מול השרת ומהלך החיים והתפקידים השונים של כל רכיב ושחקן בסיפור.

התחילו במימוש ה interface'ים של האדמין. עיבדו במקביל עם ממשק המשתמש וודאו כי אתם מצליחים להעלות קבצים ולראות את המשחקים הקיימים במע'. ממשו את החלק בצד השרת שמתעסק עם עיבוד המילון ופירוקו למילים השונות וכו'.  
המשיכו במימוש ה Interface'ים של המשתמש הרגיל. גם כאן עיבדו במקביל עם ממשק המשתמש והפעם יחד עם אפליקציית האדמין וודאו כי אתם יכולים לקבל את המידע שנאגר בבטן השרת וחושפים אותו בהצלחה באפליקציית המשתמש במסך ריכוז המשחקים.

עכשיו המשיכו לפתח את היכולת לבצע רישום של שחקנים לצוותים השונים, ורק אחרי כן את כל הקשור להתנהלות משחק פעיל.

ממליץ מאוד לאורך כל העבודה להקפיד לעבוד ולבחון את עצמכם דרך postman כאמצעי וידוא כי הסרבלטים עובדים כראוי ללא תלות בקליינט כזה או אחר. כך תוכלו לוודא כי צד השרת עובד טוב (up to a degree) טרם כתיבת החלק הרלבנטי בקליינט

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
|  |  |  |  |  |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. WAR אחד בדיוק אשר יושם בספריית tomcat\webapp ויעבור deployment אוטומטי.   
   על WAR זה לכלול את כל התלויות שלכם. אין להניח כי יסופקו לכם תלויות מבחוץ (למשל Gson וכו').
2. ספריה אחת עבור קוד הממשק הגרפי.   
   בספרייה יישבו כל ה jar'ים הנדרשים כדי לתפעל את האפליקציה, כולל ה Jar של ה UI וה jar של ה Interface'ים. יש לספק קובץ batch שמפעיל אותה (כמו שהיה בתרגיל 1).   
   שימו לב כי האפליקציה שלכם צריכה להכיר אוטומטית את השרת ואת ה context path הרלוונטי לה   
   אפשר כמובן להניח את localhost:8080 כדומיין של השרת; אפשר להניח כי לא נשנה את שם קובץ ה WAR שהגשתם.
3. קבצי קוד המקור של הפרויקט שלכם (גם צד השרת וגם מימושי הקליינטים).
4. קובץ ההגשה יכיל גם קובץ readme שיכיל הסבר על המערכת, בחירותיכם השונות במקומות שבהם היו לכם בחירה, כמו גם כל דבר נוסף העולה על דעתכם שחשוב שהבודקת תדע.
5. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות החדשות ותפקידם. (חישבו מה יסייע לבודק להיכנס ביתר קלות לקוד שלכם ולהבין מי נגד מי...)
6. במידה ומימשתם את מי מהבונוסים לעיל, ציינו את שם הבונוס **בראשית** קובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

הגשה באיחור, שאינה באישור, תבטל כל מימוש בונוס. אין להגיש באיחור בשביל להספיק לעשות בונוסים.   
תכננו את הזמן בהתאם.

נספח א' – תיאור מבנה המערכת באמצעות קובץ XML

תיאור המע' ומרכיביה בתרגילים השונים נתון באמצעות קובץ XML.

במהלך הבדיקה (של שלל התרגילים), תיבדק המערכת באמצעות מספר קבצים שונים, חלקם חוקיים וחלקם תקולים, במטרה לראות האם וכיצד המערכת מגיבה לשגיאות.

בחנו היטיב את קבצי הדוגמא שהועלו למע' ה Mama וודאו כי אתם מבינים את פרטיהם ומבניהם.

היכן שמצויין במפורש case **in**sensitive הכוונה היא שאין חשיבות ל case של האותיות באנגלית.   
במקרה זה הערך milk זהה לערך MiLk  
בכל מקרה אחר הנחת היסוד היא כי יש חשיבות ל case של האותיות (כלומר case sensitive). במקרים אלה הערך milk שונה מהערך MiLk.

היכן שמצויין שהמחרוזת יכולה להכיל רווחים – המדובר הוא רק על רווחים בתוך המחרוזת.   
אם מופיעים רווחים בתחילתה/סופה יש להתעלם מהם (רמז: המטודה trim() על המחלקה String)

כברירת מחדל כל ה element'ים הם mandatory, אלא אם מצויין במפורש כי הם אופציונלים.

A group of blue and red text

Description automatically generated מבנה המע' מאוגד תחת האלמנט **ECN-Game**:

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ECN-Game | אלמנט זה מכיל את כלל פרטי המשחק ומתאר את מאפייניו השונים |
| 2 | Element | ECN-Words | מתאר את בנק המילים האפשריות במשחק |
| 3 | Element | ECN-Board | מתאר את מבנה וגודל לוח המשחק |
| 4 | Element | ECN-Team1\2 | מתאר את הצוותים השונים המשתתפים במשחק |

האלמנט **ECN-Words** מתאר את בנקי המילים המוקצים למשחקA computer code with text

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ECN-Game-Words | מתאר את בנק המילים האפשריות לבחירה במהלך המשחק. המילים מופרדות באמצעות תו הרווח ( ). הן אינן חוזרות על עצמן ואפשר להתייחס אליהן AS IS (אינן דורשות עיבוד נוסף). |
| 2 | Element | ECN-Black-Words | מתאר את בנק המילים האפשריות לבחירה עבור המילים השחורות בלבד. המילים מופרדות באמצעות תו הרווח ( ). הן אינן חוזרות על עצמן ואפשר להתייחס אליהן AS IS (אינן דורשות עיבוד נוסף). |

A close up of a text

Description automatically generatedהאלמנט ECN-Board מתאר את מבנה ומתאר לוח המשחק

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ECN-Board | מתאר את מבנה לוח המשחק |
| 2 | Attribute | black-cards-count | מתאר את כמות המילים השחורות שיופיעו במשחק. מספר שלם חיובי. יכול להיות גם 0. |
| 3 | Attribute | cards-count | מתאר את כמות המילים הרגילות שיופיעו במשחק. מספר שלם חיובי. יכול להיות גם 0. |
| 4 | Element | ECN-Layout | מתאר את הדרך שבה יש למקם את הקלפים של לוח המשחק במרחב. את הקלפים מסדרים על פי השורות (בסיום חלוקת קלפים בשורה אחת (ככמות העמודות) פותחים שורה חדשה. כתוצאה מכך ייתכן והשורה האחרונה לא תהיה מלאה. |
| 5 | Attribute | rows | מתאר את מספר השורות. מספר חיובי שלם גדול או שווה ל 1 (>= 1) |
| 6 | Attribute | column | מתאר את מספר העמודות. מספר חיובי שלם גדול או שווה ל 1 (>= 1) |

האלמנט **ECN-Team1(2)** מתאר את הפרטים של צוות במשחק.



|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ECN-Team1 | מתאר את פרטי הצוות. יכולים וצריכים להיות בדיוק 2 צוותים והאלמנט ייקרא בהתאם (Team1 ו Team2) |
| 3 | Attribute | name | מתאר את שם הצוות. השם יכיל מחרוזת תווים רגילה. יכול להכיל רווחים |
| 4 | Attribute | cards-count | מתאר את כמות הקלפים שצוות זה מקבל. מספר חיובי שלם הגדול או שווה ל 1 (>= 1) |

## סכמה תרגיל 2

סכמה זו מכילה מספר שינויים מרכזיים:

1. **שם המשחק  
   ה**אלמנט הראשי (**ECN-Game**) מכיל גם attribute חדש בשם name המהווה את שם המשחק.  
   השם יכיל מחרוזת תווים רגילה. יכול להכיל רווחים.  
   
2. **מספר צוותים במשחק**

A math equations with red text

Description automatically generated with medium confidenceהאלמנט ECN-Teams יכיל את המידע על כלל הצוותים המשתתפים במשחק.  
המשחק יכול להכיל עד 6 צוותים

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ECN-Teams | מתאר את סך הצוותים שישתתפו במשחק |
| 2 | Element | ECN-Team | מתאר מידע על צוות אחד |
| 3 | Attribute | name | מתאר את שם הצוות. השם יכיל מחרוזת תווים רגילה. יכול להכיל רווחים |
| 4 | Attribute | cards-count | מתאר את כמות הקלפים שצוות זה מקבל. מספר חיובי שלם הגדול או שווה ל 1 (>= 1) |
| 5 | Attribute | definers | מתאר את כמות המשתתפים בתפקיד "המגדיר" שבצוות.  מספר חיובי שלם הגדול או שווה ל 1 (>= 1) |
| 6 | Attribute | guessers | מתאר את כמות המשתתפים בתפקיד "המנחש" שבצוות.  מספר חיובי שלם הגדול או שווה ל 1 (>= 1) |
| 7 | Attribute | color | מתאר את הצבע שיוענק לקלפי הצוות.  סט סופי של 6 אפשרויות (כולן באותיות קטנות; case sensitive):   * blue * green * orange * yellow * red * purple |

נספח ב' – סכמת קובץ ה XML

**סכמה תרגיל 1**

A diagram of a computer

Description automatically generated

**סכמה תרגיל 2**

A screenshot of a computer

Description automatically generated

נספח ג' – סקיצות מסכי ה UI

נספח זה מציג את מסכי ה UI השונים של שכבת ממשק המשתמש והחיבור שלהם למטודות המתוארת ב interface ומבני הנתונים השונים.

## מסך הלוגין

זהו מסך ההתחברות לאפליקציה.

A screenshot of a computer screen

Description automatically generatedה interface המלווה מסך זה, ואותו עליכם לממש הוא: **mta.jad.codenames.ui.api.login.Login**

בחירת מצב Admin ולחיצה על כפתור ה Login תתורגם לקריאה למטודה האקטיבית:

בחירת מצב Player תאפשר גם את מילוי שם השחקן. לחיצה על כפתור ה Login תתורגם לקריאה למטודה האקטיבית:

## מסך הדשבורד

זהו מסך המציג את כלל המשחקים שבמע' ומאפשר צפייה ורישום לגבי מצבם.  
ה Interface הראשי שיש לממש כאן הוא **mta.jad.codenames.ui.api.dashboard.GamesDashboard.** הוא מכיל פעולות המשותפות הן ל admin והן ל player.   
בנוסף הוא מבצע הרחבה (extends) לשני interface'ים ייעודיים המייצגים את הפעולות הפרטניות של admin ו player: **mta.jad.codenames.ui.api.dashboard.admin.AdminGamesDashboard** ו **mta.jad.codenames.ui.api.dashboard.player.PlayerGamesDashboard**

זוהי התצורה הכללית של מסך הדשבורד:

A screenshot of a computer

Description automatically generatedמבנה המסך:  
בצד שמאל ניתן לראות רשימה של כלל המשחקים הקיימים במע'.   
לחיצה ובחירה של משחק מסויים מעלה ומציגה את פרטיו בצד ימין. עבור כל משחק ניתן לראות פרטים כלליים כמו גם את מבנה והרכב הצוותים הנמצאים בו

**פעולות כלליות המשותפות ל Admin ול Player:**

1. מילוי טבלת המשחקים בצד שמאל יתבצע בצורה ריאקטיבית: עליכם להתריע ל UI בכל פעם שיש משחק חדש שרוצים להוסיף לטבלה. לטובת זה שכבת ה UI תבצע רישום באמצעות קריאה למטודה:



מבנה הנתונים שהיא מצפה לקבל נקרא LightweightGameDetails והוא מכיל מידע על כל השדות שבטבלה:

A screenshot of a computer screen

Description automatically generated

1. בבחירת משחק מסויים בטבלה שנמצאת בצד שמאל, שכבת ה UI תבצע קריאה אקטיבית למטודה הבאה:



תפקידה להחזיר את הפרטים המלאים של המשחק הנבחר דרך מבנה הנתונים FullGameDetails

**פעולות הרלוונטיות ל Administrator בלבד:**

1. האדמין הוא היחיד שיכול לטעון קבצי משחק חדשים למע'.  
   לאחר בחירת הקבצים, לחיצה על כפתור Load תבצע קריאה אקטיבית למטודה הבאה:
2. לחיצה על כפתור ה watch תבצע מעבר בתוך שכבת ה UI אל מסך משחק פעיל ואינה דורשת התערבות שלכם בהקשר זה.  
   (שאר הפעולות יתרחשו במסגרת מסך משחק פעיל אליו תועברו)

**פעולות הרוונטיות ל Player בלבד:**

1. רישום שחקן כ definer (spy master).  
   לחיצה על כפתור ה join תוביל לקריאה אקטיבית למטודה הבאה:
2. רישום שחקן כ guesser  
   לחיצה על כפתור ה Join תוביל לקריאה אקטיבית למטודה הבאה:

גכע