מס' גרסה: **6**

פיתוח תוכנה מבוסס java

C.T.E – Cracking The Enigma אביב 2022 -

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](#_Toc111755164)

[כללי 4](#_Toc111755165)

[איך להגיש תרגילים באיחור, ולהישאר בחיים 5](#_Toc111755166)

[הנחיות כלליות לכתיבת התרגיל 6](#_Toc111755167)

[תרגיל reflection – תרגיל עצמאי ביחידים (5%) - הגשה: 27.7.22 8](#_Toc111755168)

[כללי 8](#_Toc111755169)

[מבנה התרגיל 8](#_Toc111755170)

[ניקוד 8](#_Toc111755171)

[איך בודקים ? 9](#_Toc111755172)

[מה מגישים ? 9](#_Toc111755173)

[מקרה בדיקה לדוגמא 9](#_Toc111755174)

[Cracking the Enigma Machine 11](#_Toc111755175)

[מטרת התרגיל(ים) בקורס 11](#_Toc111755176)

[מכונת האניגמה 11](#_Toc111755177)

[מגבלות ומאפיינים של מכונת האניגמה בתרגיל זה: 14](#_Toc111755178)

[הנחיות ספציפיות למימוש תרגיל "פענוח מכונת האניגמה" 15](#_Toc111755179)

[תרגיל 1 – מימוש **מכונת האניגמה** כאפליקציית Console (25%) - הגשה: 20.8.22 16](#_Toc111755180)

[פרטים יבשים 16](#_Toc111755181)

[דרישות 16](#_Toc111755182)

[חלוקה למודולים 20](#_Toc111755183)

[איך מתחילים ? (המלצה...) 20](#_Toc111755184)

[בונוסים 21](#_Toc111755185)

[סבבה, סיימתי. מה ואיך להגיש ? 21](#_Toc111755186)

[שאלות ותשובות 21](#_Toc111755187)

[תרגיל 2 – **Cracking The Enigma** - כאפליקציית JavaFX (35%) – הגשה: 19.9.22 22](#_Toc111755188)

[פרטים יבשים 22](#_Toc111755189)

[דרישות 22](#_Toc111755190)

[חלוקה למודולים 27](#_Toc111755191)

[איך מתחילים ? 27](#_Toc111755192)

[בונוסים 28](#_Toc111755193)

[סבבה, סיימתי. מה ואיך להגיש ? 28](#_Toc111755194)

[שאלות ותשובות 28](#_Toc111755195)

[תרגיל 3 – מימוש **Cracking The Enigm-** כאפליקציית Client - Server (35%) – הגשה: 29.10.22 29](#_Toc111755196)

[פרטים יבשים 29](#_Toc111755197)

[דרישות 29](#_Toc111755198)

[אפליקציית UBoat 32](#_Toc111755199)

[אפליקציית Allies (DM) 33](#_Toc111755200)

[אפליקציית Agent 34](#_Toc111755201)

[חלוקה למודולים 34](#_Toc111755202)

[איך מתחילים ? 35](#_Toc111755203)

[בונוסים 35](#_Toc111755204)

[סבבה, סיימתי. מה ואיך להגיש ? 36](#_Toc111755205)

[שאלות ותשובות 36](#_Toc111755206)

[נספח א' – פרטים לגבי מימוש מכונה 37](#_Toc111755207)

[נספח ב' – מכונת אניגמה לגזירה ותרגול על נייר 41](#_Toc111755208)

[נספח ג' – תיאור מבנה המערכת באמצעות קובץ XML 42](#_Toc111755209)

[סכמה תרגיל 2 43](#_Toc111755210)

[סכמה תרגיל 3 44](#_Toc111755211)

[נספח ד' – תרשים סכמות XML 45](#_Toc111755212)

[סכמה תרגיל 1 45](#_Toc111755213)

[סכמה תרגיל 2 45](#_Toc111755214)

[סכמה תרגיל 3 46](#_Toc111755215)

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

* 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.

* 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...

* דווקא בגלל שאין זהו קורס שבו יכנסו לנבכי הקוד ויבדקו כל שורה ושורה, יש להקפיד ביתר שאת על קוד נקי, מסודר, קריא ויעיל. בפרט:
* הימנעו משכפול קוד
* פונקציות ארוכות מדי (בדר"כ יותר מגודל עמוד)
* בחירת שמות גרועים למחלקות, לפונקציות ולמשתנים
* הזחה (אינדנטציה) נכונה
* imports מיותרים
* יש להקפיד להשתמש ב-modifiers בצורה נבונה:
* מחלקה שלא אמורים לבנות אובייקטים שלה אמורה להיות מוגדרת כ-abstract
* קבועים יש לסמן כ- final
* משתנים של המחלקה רצוי להגדיר כ-private אלא אם יש סיבה לגיטימית לבחירה אחרת.
* 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.
* 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...)

* 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

* 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:**
* **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 (5%) - הגשה: 27.7.22

## כללי

**מועד הגשה: 27.7.2022 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();  
 }  
}

Cracking the Enigma Machine

## מטרת התרגיל(ים) בקורס

חלק א' – מימוש מכונת האניגמה בקוד. הפעלתה ותפעולה באמצעות ממשק console.

חלק ב' – מימוש תהליך הפענוח האוטומטי של הודעות מוצפנות באמצעות "סוכנים" תוך הצגת התהליך כממשק גרפי באמצעות JavaFx.

חלק ג' – מימוש תחרות שדה קרב, שבו מוצבים מחד כוחות האויב (צוללות גרמניות) המשדרות צפנים באמצעות מכונת האניגמה ומנגד מוצבים כוחות בעלות הברית, הפרוסים בצוותים שונים, המנסים ליירט ולפענח את התשדורות. תחרות זו תמומש באמצעות תצורת שרת-לקוח ותיכלול הסתמכות על הממשק הגרפי שנוצר זה לא מכבר בחלק ב', לצד הרמת מספר תהליכים (processes) נוספים.

## מכונת האניגמה

במהלך מלחמת העולם השנייה, השתמשו הכוחות הגרמנים [במכונת האניגמה](https://he.wikipedia.org/wiki/%D7%90%D7%A0%D7%99%D7%92%D7%9E%D7%94) כאמצעי להצפנה ולהעברת תשדורות בין הצוללות למטה הקרב לתיאום התקפות על כוחות בעלות הברית.

מכונת האניגמה הינה מכונה אלקטרו-מכנית המאפשרת להצפין הודעות טקסט. באמצעות המכונה ניתן להצפין (Encrypt) ולפענח (Decrypt) את הטקסט וכך יכלו הצוללות הגרמניות להעביר בינהן מסרים, לתאם את ההתקפות השונות ולקבל הוראות ביצוע מהדרג הפיקודי ביבשה, כל זאת מבלי לחשוף את תוכניותיהן לבעלות הברית. בעלות הברית מצידן ביצעו מאמצים רבים וניכרים (שנשאו פרי לבסוף.. go Alan !) לפענח ולפצח את מכונת האניגמה ולהבין את תוכניות הגרמנים מבעוד מועד כדי לסכל ולמנוע אותן לעיתים, וכדי להזין את הגרמנים במידע שווא לעיתים אחרות (בתחבולות תעשה לך מלחמה...)

במהלך תרגיל זה נדמה את השלבים השונים בעבודה עם מכונת האניגמה. ראשית נבין את מבנה המכונה ונבנה מכונה כזו בעצמינו (בקוד כמובן...). לאחר מכן נממש טכניקה מסוימת לפיצוח צפני המכונה – brute force (כוח מתפרץ). לסיום נגיע אל עיקר מהלך המלחמה: מחד נשגר מידע מוצפן באמצעות האניגמה ומנגד נשתמש בצי של "סוכנים" כדי לפצחו ולהגיע למידע המקורי.

**מבנה מכונת האניגמה:**

מכונת האניגמה הומצאה (בגרסתה הראשונית) כבר בשנות ה 20, אולם היא שוכללה עוד ועוד, עד להגיעה לשיא הפעילות במהלך מלחמת העולם השנייה. המכונה נראית כ"סוג" של מכונת כתיבה של פעם, רק שיש בה "מקלדת" נוספת שבה יש נורה לכל אות (מקלדת הנורות). בתהליך ההצפנה המפעיל לוחץ על אות במקלדת הראשית, וכהבזק רגע נדלקת נורה של האות המקודדת במקלדת הנורות. המפעיל רושם את האות שנורתה דלקה, ומתקדם לאות הבאה בקלט. כך, אות אחר אות, מתקבלת מחרוזת מוצפנת. כדי לפענח את המחרוזת המוצפנת ולקבל חזרה את המחרוזת המקורית, המפעיל (בצד השני) חוזר על אותו תהליך בדיוק רק הפעם המחרוזת המוצפנת מהווה קלט (למקלדת הראשית) והמחרוזת המפעונחת תופיע במקלדת הנורות. במובן זה אין משמעות ל"פענוח" או "הצפנה" של טקסט במכונה – מדובר על עיבוד הקלט שנכנס והפקת פלט מתאים לו. "הצפנה" או "פענוח" נקבעים עפ"י ההקשר של מחרוזת הקלט (אם היא טקסט מקורי או מוצפן שמוכנס למכונה).

An old typewriter on a table

Description automatically generated with medium confidenceמאפיין חשוב של מכונת האניגמה הוא העובדה שלא די להחזיק את המכונה עצמה ולדעת את המבנה שלה כדי לפענח את הצופן, אלא יש לדעת את ה"קוד" הסודי והייחודי שבאמצעותו הוצפנה ההודעה כדי לפענחה חזרה. במציאות, אכן תפסו הבריטים והפולנים מספר מכונות אניגמה – אולם לא היה די בכך כדי לפענח את ההודעות והיו צריכים לעבור עוד כברת דרך ניכרת עד שהגיעו אל היעד הנכסף.

מכונת האניגמה מורכבת ממספר חלקים מכנים ואלקטרונים. בליבת המכונה ניצבים מספר גלגלי המרה ("רוטורים"). גלגל ההמרה הוא גלגל בעל X מוצאים ו X יציאות. המוצאים והיציאות מסומנים כאותיות ה ABC – ובמכונה הקלאסית יש 26 כאלה (אצלנו יהיו יותר). גלגל ההמרה למעשה ממיר כניסה X ליציאה Y: לדוגמא הכניסה המזוהה עם האות A יכולה להיות מחוברת למוצא המזוהה עם האות K. ההמרה היא דו כיוונית, כלומר A->K ו K->A. המכונה בנויה על שילוב של מספר גלגלי המרה הניצבים אחד אחרי השני (בדר"כ 3 גלגלים שכאלה, אצלנו יהיו הרבה יותר), כך שהיציאות של אחד מהם מחוברות לכניסות של הבא אחריו. זרם חשמלי עובר בין הכניסות והיציאות של הגלגלים השונים בכיוון "הלוך" (מהגלגל הימני ביותר אל השמאלי ביותר, דרך כל הגלגלים שבדרך) ואז גם בכיוון חזור (מהגלגל השמאלי ביותר אל הימני ביותר דרך כל הגלגלים שבדרך), כך שבסוף התהליך עבור כניסה ''E ניתן לקבל מוצא 'Z'.

את הגלגלים השונים ניתן לסובב במקומם ולקבוע מיקום ראשוני שונה בכל פעם. כניסות הזרם החשמלי הממופות למקשי המקלדת נותרות קבועות. כלומר אם במצב רוטור מסוים, הקלדה על אות E ששוגרה בכניסה 5 אל הרוטור הימני ולכן נפלה על מיפוי m מסוים, לאחר סיבוב הרוטור לפוזיציה אחרת, אותה הקלדה על האות E, עדיין מגיעה לכניסה 5 אל הרוטור הימני – רק שהפעם היא תיפול על מיפוי אחר (n) וכו'. מכונת האניגמה הכילה לכל רוטור מעין "חלונית" קטנה שאפשרה לראות מהו מיקומו ולסובב את הרוטור למיקום מסויים ע"י הזזתו עד שמתקבלת האות המבוקשת בחלונית (הפוזיציה הנדרשת).

היות וכל רוטור מחווט אחרת (כלומר מיפויים שונים בין האותיות), קל להבין כי לקביעת סדר הרוטורים (מי הימני, מי באמצע, ומי בשמאלי), כמו גם למיקום הראשוני של כל רוטור (אילו אותיות ניבטות מהחלונית) יש חשיבות מכרעת, קריטית, בהצפנה ובפענוח ההודעה. אם זה לא מספיק, מכונת האניגמה מגיעה עם סט של מספר רוטורים, אבל משתמשת בפועל רק **בחלקם**, בסדר מסוים, ובמיקום התחלתי מסוים.

Chart, radar chart

Description automatically generatedהמכונה מורכבת מרכיב נוסף הנקרא רפלקטור ("משקף"). הרפלקטור ניצב אחרי סדרה הגלגלים ותפקידו לקבל זרם חשמלי בכניסה ולהחזיר אותו ביציאה חזרה לסדרת הגלגלים (reflect - לשקף). הוא האחראי לתהליך ה"חזור" בהצפנה. גם כאן המכונה הגיעה עם מספר רפלקטורים, ויש צורך להכריע במי מהרפלקטורים משתמשים.   
**שימו לב**: בניגוד לרוטור, אין משמעות ל"מיקום" הרפלקטור – הוא לא רוטור שניתן לסובב ולשנות את מיקומו.

אני ממליץ בחום לראות את [הסרטון](https://www.youtube.com/watch?v=ybkkiGtJmkM) הזה (אנימציה. 20 דקות) המסביר לפרוטרוט את המבנה המכני/אלקטרוני של מכונת האניגמה. זה יעזור לכם "להיכנס לעסק" 😊

אם כן, ממה מורכב "הקוד הסודי" של מכונת האניגמה ?

המכונה מגיעה עם מספר רוטורים אפשריים לשימוש, ממוספרים בספרות 1..n יש לבחור C רוטורים לשימוש מהם, ולקבוע את סדר הרוטורים. לאחר מכן יש לקבוע את המיקום ההתחלתי של כל רוטור. על כל רוטור יש את 26 אותיות ה ABC כך שפשוט קובעים מי האות המהווה את המיקום הראשוני של כל רוטור. לסיום, יש לבחור באיזה רפלקטור משתמשים (יש כמה מהם. מסומנים באותיות יווניות בכתיב רומי)

כדי להקשות על פענוח הצופן עוד יותר, מכונת האניגמה השתמשה בטריק פשוט (אך זדוני): בכל הקשת אות כקלט, רגע לפני שילוח הזרם דרך הרוטורים בכיוון "הלוך"-> משקף -> "חזור", המכונה הייתה מקדמת את הגלגל הימני ביותר צעד אחד קדימה ("פסיעה"). כשהגלגל הימני היה מגיע לפוזיציה מסויימת, זיז קטן (notch) המקובע על הרוטור הימני היה גורם לגלגל הבא אחריו להתקדם גם כן בפסיעה אחת. בדומה לכך, כשהגלגל המרכזי היה מגיע לפוזיציה מסויימת, זיז קטן המקובע על הרוטור המרכזי היה גורם לגלגל הבא אחריו להתקדם גם כן בפסיעה אחת. למעשה כל הקלדת אות קידמה את אחד (או יותר) מהגלגלים שלב אחד קדימה ובכך למעשה שינתה את המיקום של הגלגלים, וכפועל יוצא את כל החיווט עבור כל אות שהוצפנה (!)   
כלומר מכונת האניגמה מצפינה בצופן שאינו 'צופן החלפה' פשוט: אותה אות שתוקלד כמה פעמים רצוף תוצפן בכל פעם לאות אחרת (!!) הדבר מוסיף מימד קושי נוסף בנסיונות לפענח את הצפנים בטכניקות שהיו ידועות עד אז, המבוססות על ניתוח תדירויות של אותיות וכו'.

מיקומו של הזיז בכל גלגל הוא שרירותי (כלומר **לא כל** הזיזים ממוקמים בפסיעה ה 26, כפי שאולי הייתם מצפים...), ועם כל הזזה של כל רוטור – הזיז שלו "זז" יחד איתו (נסו להגיד את זה כמה פעמים רצוף ותראו מה יוצא לכם... )

פרט נוסף לשים לב אליו הוא שבשל קיומו של המשקף, אות לעולם לא יכלה להיות מוצפנת לעצמה (חישבו מדוע ?).   
(בדיעבד, פרט זה דווקא יצא בעוכריה של המכונה שכן הוא סייע (במעט) לכוחות הבריטים בפענוח הצופן אחרי הכל...)

הספיק לכם ? אה עוד לא ? אז...

הרכיב האחרון במכונה היה לוח התקעים:

לוח התקעים הכיל כניסה לכל אות מהאב' (26 אותיות ה ABC) ואיפשר למעשה להגדיר צמדים של אותיות מתחלפות. המפעיל היה מחבר כבל עם 2 פינים לתוך התקעים של צמד אותיות (נגיד K ו D או למשל Z ו O) וזה אומר שכל פעם שאותיות אלה מגיעות אל לוח התקעים – הן מתחלפות.   
אות הגיעה אל לוח התקעים פעמיים:

ב"הלוך", מייד אחרי שהמפעיל הקיש על אות (נגיד K), היא זרמה אל לוח התקעים. במידה והיא הייתה חלק מצמד מתחלף, הרי שעכשיו היא הייתה מתחלפת בהתאם לצמד שלה (במקרה זה D). האות D הייתה זו שמגיעה עכשיו אל החלק הארי של המכונה (רוטורים -> משקף -> רוטורים) והיא זו שהייתה מוצפנת (נגיד ל Z).

בדרכה "חזור" אל מקלדת הנורות, היא הייתה עוברת שוב בלוח התקעים, ואולי פוטנציאלית מוחלפת שוב (למשל במקרה שלנו מוחלפת ל O) ורק אז התוצאה הייתה זורמת למקלדת הנורות.  
אות שמגיעה ללוח התקעים ואיננה חלק ממיפוי – נותרת בעיני ויוצאת ממנו כשם שבאה.

מאפייני שימוש בלוח התקעים:

* במכונת האניגמה נהוג היה להשתמש בעד 10 צמדים (מתוך 13 אפשריים סה"כ). אצלנו נשתמש בכמה שרק נרצה.
* בניגוד לשאר הרכיבים – אין חובה להשתמש בלוח התקעים.
* אות לא יכולה להופיע בצמד עם עצמה.
* אות יכולה להופיע לכל היותר בצמד אחד (או בכלל לא)

פשוט ונאיבי ככל שזה נשמע – לוח התקעים הוסיף מימד שהכפיל בכמה סדרי גודל (!) את הקושי לפצח את מכונת האניגמה (ראו הפנייה לסרטון בהמשך המפרט את הסוגייה).

אם כן, מבנה הקוד במכונה צריך להכיל את הפרטים הבאים (בתצורה הבאה):  
בהינתן מכונה עם 5 רוטורים אפשריים, אך כזו המשתמשת רק בשלושה בפועל, ו 3 משקפים אפשריים, אזי הינה מספר דוגמאות לפענוח הקודים הבאים:

|  |  |  |
| --- | --- | --- |
| # | מבנה קוד | משמעות |
| 1 | **<2,3,1><A(1),I(5),C(15)><II><A|D,K|E>** | בחרי את רוטורים המסומנים 1,2,3 הצב אותם כך ש 1 בימין, 3 במרכז ו 2 בשמאל. קבעי את המיקום ההתחלתי של הרוטורים כך ש 1 ממוקם ב C (והזיז שלו מרוחק 15 פסיעות מחלונית ההצצה), 3 ממוקם ב I (והזיז שלו מרוחק 5 פסיעות מחלונית ההצצה) ו 2 ממוקם ב A (והזיז שלו מרוחק פסיעה אחת מחלונית ההצצה). השתמשי במשקף מספר 2 (II). השתמשי ב 2 תקעים: בין A ו D ובין K ו E |
| 2 | **<5,1,4><D(5),O(2),W(0)><I>** | בחר את רוטורים המסומנים 5,1,4 הצב אותם כך ש 4 בימין, 1 במרכז ו 5 בשמאל. קבע את המיקום ההתחלתי של הרוטורים כך ש 4 ממוקם ב W (והזיז שלו נמצא ממש בחלונית ההצצה), 1 ממוקם ב O (והזיז שלו מרוחק 2 פסיעות מחלונית ההצצה) ו 5 ממוקם ב D (והזיז שלו מרוחק 5 פסיעות מחלונית ההצצה). השתמש במשקף מספר 1 (I). אין שימוש בתקעים. |

בתרגיל זה, כל הגדרות המכונה יוגדרו בקובץ בתצורת XML שמבנהו ופרטיו מסופקים בהמשך. אנו נבנה מכונת אניגמה "גנרית" – כזו שאינה מצייתת למגבלות המכונה הפיזית ושניתן להרחיבה ככל העולה על רוחינו (ואכן יעלה על רוחינו להרחיבה...)

על מנת לשמור על אחידות ותאימות, כניסות המקלדת הקבועות ייקבעו על פי סדר הופעת האב' (ממוספרות מ 1 עד n).  
הגדרת רוטור נעשית באמצעות הגדרת 2 "טורים" של אותיות האב' (המפוזרים רנדומלית בינם לבין עצמם). הטורים נקראים right ו left. "המיפויים" של כל רוטור ורוטור נקבעים ע"י מציאת האות הזהה ב right וב left.   
חלונית ההצצה נקבעת להיות במיקום מס' 1. כאשר הזיז מגיע למיקום זה או אז הגיעה העת לבצע את מהלך הפסיעה של הרוטור הבא בתור. (מהלכי הפסיעות מתקדמים אך ורק מימין לשמאל ; רוטור פוסע רק בעקבות הגעת הזיז של הרוטור מימין לו לחלונית ההצצה)   
כאשר מקבעים רוטור מסויים לפוזיציה מסויימת – המדובר הוא על הופעת האות/מיקום הרלוונטי על פי עמודת ה right. יש להביא את הרוטור כך שהאות בעמודה right הנדרשת תופיע בחלונית ההצצה מול מיקום מס' 1.   
עיקבו אחר [נספח א'](#Appendix_A_machine_implementation) ו[נספח XML](#Appendix_B_XML) כדי לקבל את מלוא הפרטים בהקשר זה.

כדי לפענח את ההודעות המוצפנות ע"י מאזין חיצוני (בעלות הברית), כזה שאין ברשותו את הקוד הראשוני הנחוץ להפעלה, נשתמש בטכניקה מוכרת הנקראת brute force (כוח מתפרץ בתרגום חופשי). בטכניקה זו, רצים וסורקים את כל האפשרויות השונות של הקודים, ומנסים לפענח את התשדורת המוצפנת. הנחת היסוד, כאמור, היא כי לשני הצדדים יש את מכונת האניגמה על שלל מרכיביה השונים (רוטורים, משקפים וכו') וזה יהיה גם המקרה אצלנו (2 הצדדים יקבלו את פרטי המכונה בדמות קובץ ה XML).

כמה אפשרויות מתקיימות ?

לדוגמא: במכונת אניגמה סטנדרטית, המגיעה עם 5 רוטורים, אבל פועלת רק עם 3, על כל רוטור יש בדיוק (אך ורק) את 26 אותיות ה ABC, וברשותינו 2 משקפים לבחירה, וללא תקעים (ובהנחה שאנו יודעים זאת), ניתן לחשב את כמות האפשרויות בצורה הבאה:

כמות האפשרויות לבחירת 3 גלגלים מתוך 5 גלגלים אפשריים, בלי חזרות (הבינום של ניוטון - n מעל k) X

כמות האפשרויות לסידור 3 גלגלים בינם לבין עצמם (בלי חזרות) X

כמות המשקפים שברשותינו X

כמות האפשרויות לבחירת קוד התחלתי על פני 3 גלגלים

ובנוסחה:

5!/[3!(5-3)!] \* 3! \* 2 \* 26^3 = 10 \* 6 \* 2 \* 17576 = 2,109,120

([בסרטון הזה](https://www.youtube.com/watch?v=G2_Q9FoD-oQ) ניתן לראות את החישוב המלא הכולל את רכיב לוח התקעים. אם חשבתם שגוגל זה מספר גדול...)

קל לראות שהרכיב המרכזי הוא כמות האפשרויות לבחירת קוד התחלתי, ואלו תלויים בכמות האותיות האפשריות לקידוד. סט האותיות האפשריות לקידוד במכונה ייקרא אלף-בית של המכונה (Alphabet). אצלנו בתרגיל, האלף-בית של המכונה יכיל גם ספרות ותווים מיוחדים ויהיה גדול הרבה יותר מ 26, כך שמספר האפשרויות מגיע בקלות לעשרות מליונים אם לא ליותר.

היות ומדובר בכמות אפשרויות אדירה, נשתמש באמצעים המחשוביים העומדים לרשותינו כדי לבזר ולחלק את מלאכת הפענוח על פני כמות משאבים מוגדרת. על מנת לפשט במעט את התהליך, נניח כי תשדורת יכולה להכיל אך ורק מילים מתוך בנק מילים סופי. גם נתונים אלה יגיעו מקובץ ה XML המתאר את התרגיל.

בקיצור,

**משימה קשה במיוחד !**

## מגבלות ומאפיינים של מכונת האניגמה בתרגיל זה:

1. סט התווים הסופי שניתן לקודד במכונה יכול להשתנות ולהכיל תווים שונים ככל שנגדיר זאת.

התווים יכולים להיות אותיות ה ABC, ספרות, תווים מיוחדים (!@#$%^& וכו'), תו הרווח ועוד.

מובטח כי **לא ייכללו** התווים הבאים:

טאב (\t), enter (\r\n), esc (027)

1. גודל האב' של המכונה חייב להיות זוגי (בעיקר בשל מגבלות המשקף ולוח התקעים)
2. המכונה תוגדר לכל היותר עם 5 משקפים לבחירה (מהם צריכים לבחור **בדיוק** אחד)
3. המכונה תוגדר לכל היותר עם 99 רוטורים **בשימוש** (שימו לב: יכולים להיות מוגדרים יותר מ 99 רוטורים, אולם לכל היותר 99 יהיו בשימוש)
4. משקף **לעולם לא** יוכל לבצע מיפוי בין אות לעצמה.
5. **בהחלט ייתכנו** מיפויים ברוטורים בין אות לעצמה.
6. נוכל להשתמש במקסימום התקעים האפשרי, או בכלל לא. ככל שנרצה

## הנחיות ספציפיות למימוש תרגיל "פענוח מכונת האניגמה"

1. **המטרה היא לבנות מנוע מערכת גנרי, כזה שידע לקבל את הפרטים לגבי אופי המע' (מכונת האניגמה) מתוך קובץ נתונים בפורמט XML (עבודה עם XML'ים תילמד במהלך הקורס כמובן).**

**מנוע המערכת הגנרי ילך וישתכלל מתרגיל לתרגיל, 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' שיפותח מתרגיל לתרגיל בהתאם לדרישות. המנוע יכיל, בין היתר, את מימוש המכונה עצמה, את היכולת לפענח את ההודעות המוצפנות ובהמשך את יכולת "המשחק" בין יריבים שונים (צוללות מול בעלות הברית).**
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. **מיפוי כל רוטור הוא מיפוי דו כיווני ("הלוך": right to left ו"חזור": left to right).**
5. **המחרוזות והתווים שנצפין/נפענח יכולים להתקבל גם ב lower case, אולם תוצאת העיבוד (הצפנה או פענוח) יוחזרו תמיד כ Uppercase. יש לדעת להתמודד עם 2 המצבים.**
6. **על מנת ליישר קו, כיוון עיבוד התווים יהיה מימין לשמאל – כלומר הרוטור הראשון שנבחר הוא הימני והאחרון שנבחר הוא השמאלי. מקלדת הקלט ומקלדת הפלט (הנורות) נמצאות מימין לרוטור הראשון והמשקף נמצא משמאל לרוטור האחרון:**

**מקלד(ו)ת -> רוטור 1 -> רוטור 2 -> ... -> רוטור n -> משקף**

**כך גם במתן והגדרת קוד המכונה (ראו פרטים בהמשך).**

1. **המע' כולה תיכתב ותורץ בסביבת העבודה של ג'אווה 8.**
2. **המע' כולה תתואר בשפה האנגלית בלבד, עם ממשק משתמש המתנהל משמאל לימין (במקומות הרלבנטים)**

תרגיל 1 – מימוש **מכונת האניגמה** כאפליקציית Console (25%) - הגשה: 20.8.22

## פרטים יבשים

צפי תחילת עבודה: **24.7.22** תאריך הגשה: **20.8.22**

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

משקל התרגיל: **25%** קושי: **סביר**

**מטרת התרגיל העיקרית**

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

## דרישות

1. בתרגיל זה תקימו את מכונת האניגמה ותעמלו על יכולת התפעול התקינה שלה, על כלל ושלל מרכיביה.   
   את המכונה "תתפעלו" באמצעות ממשק console פשוט המציג תפריט פקודות שדרכו מפעילים את המכונה.  
   בתרגיל תממשו את מנוע המערכת, אשר יודע לקבל נתונים על המכונה ויודע להגיב לכל פנייה שמגיעה משכבת ממשק המשתמש, לעבד את הקליט ולהחזיר פלט רלבנטי.
2. פרטיה הטכנים של המערכת יינתנו מקובץ XML (כמפורט [בנספח](#Appendix_B_XML) ג').
3. יש לוודא תקינות קלט כחלק מכל אינטרקציה עם המשתמש, ובכל מקום שבו זה רלבנטי:

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. (זה הזמן לחשוב על...)

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 רלבנטית למשתמש ולאפשר את המשך מהלך 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).   
  כלומר, **אין להניח** שהמשתמש הולך להקליד לכם מלל חופשי של תיאור האפשרות !

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).   
הנתיב יכיל רק אותיות באנגלית (לא ג'יבריש של אותיות בעברית וכו')

הקובץ יהיה קובץ XML שפרטיו וחוקיו המפורטים מובאים [בנספח ג'](#Appendix_B_XML) לתרגיל זה.   
אתם מצופים לעבור על פרטים אלה ולהתייחס אליהם כחלק 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. גודל ה ABC הוא זוגי
3. כמות הגלגלים שהמכונה צריכה (rotors-count) קטנה או שווה לכמות הגלגלים הקיימת.
4. rotors-count >= 2
5. לכל רוטור יש id ייחודי. כל ה id'ים של הרוטורים השונים צריכים בסוף להוות ספרור רץ המתחיל מ 1. (הם אינם חייבים להופיע על פי הסדר)
6. אין מיפויים כפולים בתוך רוטור
7. זיז הפסיעה (notch) בכל רוטור מוגדר בטווח גודל הרוטור (פונקציה של גודל ה ABC)
8. לכל משקף יש מספר סידורי בשפה הרומית. כל ה id'ים השונים צריכים בסוף להוות ספרור רץ של הספרות הרומיות

(1-5: I, II, III IV, V). הם אינם חייבים להופיע על פי סדר הספירה הטבעי.

1. אין מיפוי בין אות לעצמה באף לא אחד מן המשקפים המוגדרים

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' ולייצר מופע של המכונה הנגזר ממנו. יש לדווח על כך 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 מפרט המכונה

פקודה זו מציגה את מפרט המכונה המוגדרת בהתאם לקובץ התקין האחרון שנטען בהצלחה.

יש להציג את הפרטים הבאים:

1. כמות גלגלים אפשרית, כמות גלגלים בשימוש (למשל 3/5)
2. כמות משקפים
3. כמה הודעות עובדו במכונה עד כה, סה"כ, מאז שנטען הקובץ, עבור דרך כלל הקודים שהוגדרו בה (פק' 3 או 4)
4. תיאור תצורת קוד מקורית (במידה וקיימת ; האחרונה שנקבעה ע"י פקודה 3 או 4)
5. תיאור תצורת קוד נוכחי (במידה וקיימת ; ייתכן ששונה מתצורת הקוד המקורית בעקבות עיבוד קלט – פקודה 5)

את תצורת קוד המכונה (סעיפים 4 ו 5) יש להציג בפורמט הבא (זהו הפורמט והוא מחייב):

<בחירת וסידור גלגלים בחריצים><מיקום גלגלים ומרחקי זיזים מהחלונית><מספר משקף בספרות רומיות><צמדי תקעים>

דוגמא (ראו הסבר צבעוני בהמשך):  
(הצבעים נועדו להבהרה בלבד. אין צורך להשתמש בצבעים במסגרת תרגיל console!):

<45,27,94><A(2),O(5),!(20)><III><A|Z,D|E>

<198,346,123,959><I(4),=(10),3(52),K(3)><V>

זיכרו כי מיקומי הרוטורים נקבעו מימין לשמאל, על אף שמדפיסים את הנתונים הנ"ל משמאל לימין...

(בדוגמא הראשונה, הרוטור הימני ביותר הוא 94 והוא מכוון לתו '!'. הרוטור השמאלי ביותר הוא 45 והוא מכוון לתו 'A')

הסבר על הרכיבים השונים:

* 1. מספרי גלגלים שנבחרו + סדר בינהם – סדרת מספרים מופרדת בפסיקים (,). רוטור לא יכול להופיע פעמיים.
  2. מיקומו של כל רוטור – נתון כסדרת תווים חוקיים מהאלף-בית של המכונה, מופרדים בפסיקים.   
     ליד כל אות, בתוך סוגריים עגולים, מופיע מרחק הזיז שלו מחלונית ההצצה. 0 אומר שהוא ממש בחלונית ההצצה.
  3. משקף נבחר – נתון בספירה רומית: I, II, III, IV, V
  4. תקעים בשימוש (במידה וקיימים) – צמדי תווים מופרדים באמצעות התו pipe (|). הצמדים מופרדים באמצעות פסיק (,)

ממליץ ללמוד לעבוד עם [StringBuilder](https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuilder.html) כאמצעי לבניית המחרוזת (המתוסבכת משהו) צעד אחרי צעד בצורה יעילה (ובעיקר חביבה)

1. בחירת תצורת קוד ראשונית (בצורה ידנית)

פקודה זו תאפשר למשתמש להגדיר את תצורת הקוד שישמש להצפנה ופענוח במכונת האניגמה.

תצורה זו תורכב מ 4 חלקים, כפי שמוגדר בפקודה 2 סעיף 5

יש לבקש מהמשתמש את הקלטים בכמה מנות, בסדר הזה (אחרי כל קלט, מתקדמים לקלט הבא):

1. הרוטורים ממוספרים בספרות עשרוניות. יתקבלו כרשימה שבה הם מופרדים באמצעות התו פסיק (,).  
   דוגמא: 23,542,231,545  
   על אף שהקלט מוכנס משמאל לימין (מתחילים מ 23) הרוטור הימני ביותר הוא 545 וכו'
2. מיקומי הגלגלים הראשוניים הם חלק מה-אב' של המע'. יש להכניסם כרצף תווים צמודים.  
   דוגמא: A8D4   
   בקלט הנ"ל, על אף שהוכנס משמאל לימין (4 הוכנס ראשון) – האות A היא המיקום של הרוטור הימני ביותר (545)
3. מספר המשקף יצוין בספירה רומית (I, II, III, IV, V).  
   יש להציג למשתמש רשימה מספרית ולקבל ממנו מספר עשרוני, המציין את המשקף הרלוונטי בספירה הרומית. אין לצפות מהמשתמש להכניס לכם ספרות רומיות !  
   דוגמא: 3 פירושו III ו 5 פירושו V וכו'.
4. פלגים: יש להכניס מחרוזת רציפה של תווים המהווים את כל הצמדים (הפלגים) הנדרשים במע'.  
   הצמדים יופיעו בצורה צמודה, ללא שום מפריד כזה או אחר. הקשה על enter תסיים את הכנסת המחרוזת.  
   עליכם לוודא כי המחרוזת היא באורך זוגי (כי עסקינן בצמדים) ולחלק אותה כצמדים על פי הסדר.  
   **שימו לב**: מחרוזת ריקה היא קלט חוקי - פירושו שאין פלגים בכלל במע'.  
   כמו כן יש לוודא כי לא משתמשים באותה האות ביותר מצמד מיפוי אחד, וכן אין מיפוי מאות לעצמה.  
   דוגמא: [**dk49 !**]  
   בקלט הנ"ל הוגדרו 3 פלגים:
   1. D | K
   2. 4 | 9
   3. | ! (שימו לב תו הרווח הוא תו חוקי באב' והוא חולק את הפלג שלו עם התו !)

יש לבצע בדיקת קלט למשתמש ולהתמודד עם מצבים שבהם הוא מקליד מספרים היכן שמצופים אותיות או להיפך, או כל ניסיון אחר שלו להזין קלט שאינו תקין ואינו עומד בציפייתכם. במידה וזה יקרה (וזה יקרה לבטח !) יש להודיע על כך בצורה מסודרת למשתמש – תוך הסבר מפורש מה הבעיה שנתקלתם בה וכיצד הוא יכול לתקנה.   
במידה ואיתרתם תקלה ודיווחתם אותה למשתמש, תוכלו לבחור איך להתקדם מכאן:   
להחזיר את המשתמש לתפריט הראשי (שיתחיל מההתחלה) או לאפשר לו להישאר "בתוך הפקודה" ולתקן רק את הרכיב(ים) הלא תקינים. כך או אחרת אין "לכלוא" את המשתמש בתוך בפקודה (עד שיכניס קוד תקין) ויש לאפשר לו לחזור חזרה לתפריט הראשי אם ירצה.

לאחר קביעת הקוד הסודי יש לכוון את המכונה אליו, כך שעיבוד הקלט שיגיע בהמשך יתרחש בהקשרו. שימו לב כי יש למקם את הרוטורים בהתאם למיקומיהם כמו גם את הפוזיציות הראשונות של הרוטורים יש למקם על פי מה שהוגדר בעמודת ה right.

1. בחירת תצורת קוד ראשונית (בצורה אוטומטית)

פקודה זו תבצע בחירת קוד בצורה אוטומטית עבור המשתמש. כל פרטי הקוד יוגרלו בצורה רנדומלית לחלוטין. בסיום הפקודה תוצג התצורה הנבחרת למשתמש והיא תיקבע כתצורה האקטיבית במכונה.

1. עיבוד קלט

בפקודה זו המשתמש יזין קלט ויקבל בסופו את המחרוזת שהזין כשהיא מעובדת (מוצפנת או מפוענחת – הכל תלוי בהקשר).

העיבוד יתרחש עם תצורת הקוד שהוכנסה אחרונה למערכת. אין לאפשר הפעלת פקודה זו באם טרם בוצעה פקודה 3 או 4 (אפשרי גם ע"י אי הצגת פקודה זו למשתמש במצבים אלה).

יש לוודא כי המשתמש לא מזין אותיות שאינן חלק מה-אב של המע'. (להודיע לו בצורה מסודרת כמובן אם זה קורה)  
בסיום פקודה זו הרוטורים נשארים במקומם הנוכחי ואינם מוחזרים אוטומטית לקוד הראשוני שאותחל במכונה (פק' 3 או 4).

1. ביצוע איפוס קוד נוכחי

זיכרו כי בכל הקלדת אות, הרוטורים השונים מבצעים פסיעות (גלגל ימני תמיד מבצע פסיעה, אחרים לעיתים). על מנת לקודד מספר הודעות ברצף שכולן מתחילות מאותה תצורה ראשונית – יש לאפסה לאחר כל הצפנה/פענוח.

כאמור, בעקבות כל הצפנת אות מיקומי הרוטורים משתנים.

1. הסטוריה וסטטיסטיקה

פקודה זו תחזיר את הסטוריית הפעילות במכונה הנוכחית כמו גם מספר נתונים סטטיסטים לגביה:

1. יש להציג את כל תצורות הקוד השונות המקוריות (כלומר אלה שנקבעו באמצעות פקודה 3 ו 4) אשר בוצעו במע' עד כה (על פי הפורמט המוגדר בפקודה 2.4)
2. עבור כל תצורת קוד שכזו יש להציג את כל המחרוזות שעברו עיבוד (פקודה 5), כמו גם את משך הזמן שלקח לבצע את התהליך.

הפלט יוצג בפורמט הבא:

#. <מחרוזת מקור> --> <מחרוזת מעובדת> (n nano-seconds)

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 המערכת, 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.**

## איך מתחילים ? (המלצה...)

התחילו מהבנת מבנה ופעולת המכונה בצורה מוחלטת. שחקו את זה על הנייר, חפשו סרטונים ב YouTube, ריכשו מכונת אניגמה מאספן מציאות - העיקר וודאו כי התהליך מובן והגיוני לכם לאשורו.

צרו פרוייקט חדש ב Intellij שישמש כפרוייקט האב לכלל התרגילים. בתוך הפרוייקט צרו מודול נפרד שיכיל את מימוש המכונה עצמה. פרקו את מבנה המכונה לחלקים המרכיבים אותה (רוטורים, משקף, מקלדת וכו') והתחילו לממש אותם ואת הקשרים בינהם.  
בידקו את תפעול המכונה בצורה מסודרת דרך מטודת main קטנה כדי לוודא שהמכונה לכשעצמה עובדת כהלכה.

צרו מודול נוסף עבור מנוע המע'. המנוע יחשוף סט של יכולות (הפקודות השונות בתפריט), יחזיק מופע של המכונה ויתווך את הפקודות השונות מול המכונה, הלוך וחזור. התחילו בבניית תשתית המנוע (מומלץ לעטוף את המנוע בממשק ולהתחיל לתרגל (ולהתרגל) לחווית העבודה הנ"ל.  
התחילו במימוש פקודה 1 (על כלל היביטי הטעינה), ופקודה 2 שתאפשר לכם לראות כי קיבלתם את המידע כנדרש. עיברו לממש את פקודות תפעול המכונה עצמה (3-6) ורק בסוף התעסקו עם סוגיית ההסטוריה והסטטיסטיקה.

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 למופע המכונה) וכך תוכל להעביר ולתרגם לו את הפקודות הנאספות מהמשתמש ולהציג חזרה את הפלטים החוזרים מהמנוע.

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | מגניב לאללה ! | שמירה וטעינה של מצב נתון של מכונה. במצב זה המכונה הנוכחית תשמר לקובץ חיצוני (באיזה פורמט וטכניקה שתבחרו), כולל תצורת הקוד הנוכחית, כל ההסטוריה והסטטיסטיקה שנאספה עד כה וכו'. יש להוסיף פקודה שמאפשרת לשמור את מצב המכונה וגם פקודה המאפשרת לטעון מכונה קיימת (מקובץ שנשמר זה לא מכבר), 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 המכונה אליו (ולטעון אותה ממנו) | כי 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 זאת 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.

## שאלות ותשובות

|  |  |  |  |
| --- | --- | --- | --- |
| # | תאריך | שאלה | תשובה |
|  |  |  |  |

תרגיל 2 – **Cracking The Enigma** - כאפליקציית JavaFX (35%) – הגשה: 19.9.22

## פרטים יבשים

צפי תחילת עבודה: **22.8.22** תאריך הגשה: **19.9.22**

צפי זמן לביצוע: 4 **שבועות** ציון אפשרי מקסימלי: **120**

משקל התרגיל: **35%** קושי: **מאתגר**

**מטרות התרגיל העיקרית**

1. מימוש תצוגת ותפעול המכונה כאפליצקיה גרפית JavaFX
2. מימוש יכולת פענוח אוטומטית של הודעות

## דרישות

1. בתרגיל זה תממשו את היכולת לפענח בצורה אוטומטית את הצפנת מכונת האניגמה באמצעות טכניקת brute force.

כדי לבצע זאת תממשו רכיב חדש שנקרא (DM) Decryption Manager, אשר מנהל מספר "סוכנים" (agents), האחראים על ביצוע המשימה המסובכת. ה DM יחלק את עומס העבודה ל"משימות" קטנות יותר בין הסוכנים השונים העומדים לרשותו (נתון המגיע מקובץ ה XML). כל סוכן יבצע את הבדיקות לפי המשימות שקיבל ויישלח את תוצאות הבדיקה שלו חזרה ל DM. התוצר של ה DM הוא אסופה של מחרוזות המהוות מבחינתו מועמדות מתאימות לפענוח של הודעה מסוימת.

1. על מנת להקל במעט את אופן הפענוח, נכניס גורם חדש למשוואה שהוא המילון. כל הודעה יכולה להיות מורכבת אך ורק ממילים המוגדרות במילון. נתוני המילים שבמילון מגיעים כחלק מקובץ ה XML.

להלן מספר מאפיינים של המילון שיש לתת את הדעת עליהם:

* המילים במילון יכולות לחזור על עצמן. אין לכך משמעות ויש "לסנן" כפילויות. (מי אמר SET ולא קיבל...)
* בין כל מילה למילה יפריד תו הרווח.
* ייתכנו תווים "מיוחדים" שיופיעו בצמוד למילים השונות במילון. המילון יגיע עם הגדרה של מהם אותם תווים מיוחדים והמילים "החוקיות" לשימוש הן רק המילים במילון אחרי שעברו "הפשטה" של כל התווים המיוחדים המופיעים בהם.
* ייתכנו מילים במילון שיכילו תווים שאינם מופיעים באב' של המכונה. זה בסדר. זה רק אומר כי אלו מילים שלא ניתן יהיה להקליד אותם כחלק מההודעה (על אף שהם מופיעים במילון) מאחר והם מכילים תווים שהמכונה אינה מכירה.
* ייתכן כי האב' של המכונה מכיל תווים אשר נפסלו לשימוש ע"י המילון (למשל האב' מכיל את השימוב בסימן ! ואילו המילון אוסר זאת).   
  גם כאן, במידה ומשתמשים במילון (בהקשר תהליך הפענוח האוטומטי בלבד), רגע לפני שנצפין את מחרוזת הקלט נפשיט אותה מהתווים המיוחדים במילון (ככל שאלה הופיעו בה) ורק אז נפרקה למילים בודדות (הפרדה לפי תו הרווח) ונבדוק אם היא חוקית ונמצאת במילון.

1. ה DM מחזיק ברשותו עותק מלא של מכונת האניגמה, כולל כל הרוטורים/משקפים המוגדרים בה. ה DM מקבל את ההודעה המוצפנת (מיירטה בדרך מסתורית...) אולם כדי לפענחה הוא צריך לדעת את הקוד הראשוני של מכונת האניגמה. היות והוא לא יודע זאת, אין לו ברירה אלא "לרוץ" על כל ההאפשרויות השונות של הקוד, לכייל את המכונה לכל אחת מהן ולפענח את ההודעה. הפלט שיוצא מפענוח זה מבחינתו מהווה מחרוזת אותיות וסימנים חסרי הקשר. כדי להתמודד עם הקושי, ה DM לוקח את המחרוזת ומנסה לפרקה (תו הרווח הוא המפריד בין מילה למילה) למילים אפשריות, הנמצאות במילון. במידה ומצא מחרוזת שניתן לפרקה לסט מילים שכולן מופיעות במילון – הרי שמחרוזת זו מוגדרת כ"מועמדת" לפענוח והיא תהווה את תוצר הפענוח.

שימו לב:

* 1. ייתכן (ואף סביר) כי למחרוזת מוצפנת אחת יהיה יותר ממועמד אחד לפענוח. לכן אין לעצור אחרי שמצאנו את המועמד הראשון.
  2. זהו לא קורס עיבוד שפה טבעית ולכן נתעלם לחלוטין מן המשמעות הסמנטית של המחרוזת, כלומר היא יכולה להרכיב בליל של מילים – העיקר שכולן מגיעות מן המילון.

1. לטובת ביצוע מלאכת הפענוח בצורה יעילה ה DM יהיה אחראי על חלוקת התהליך לתתי משימות.

"משימה" כוללת מקטע של קודים שאותם צריך לבדוק הסוכן. המקטע מגיע לידי ביטוי אך ורק בשינוי הפוזיציה הפנימית של הרוטורים השונים. (כלומר לא הסדר בין הרוטורים, אלא איזה אות של הרוטור מופיעה בחלונית ההצצה...)

זיכרו שכל רוטור מכיל כמות כניסות ויציאות (מיפויים) הזהים בגודלם לגודל האלף-בית של המכונה (L). אם כן, אפשר להסתכל על סט כל הרוטורים כאילו היו שעון דיגיטלי או ספידומטר של מכונית שבו החל ממצב נתון מתקדמים כמה פסיעות קדימה (שניות או מטרים).

לדוגמא:

אם הקוד הראשוני של מיקומי הרוטורים בלבד הוא AAA וגודל המשימה הוא 1000, אזי זה אומר שהסוכן בודק את כל הקופיגורציות החל מ AAA ואז AAB ואז AAC וכו' - כך 1000 פעמים. הדבר מקביל לספירה בבסיס L...

ה DM מחליט על גודל המשימה (נתון שחשוף למשתמש והוא יכול לכוון אותו כרצונו).  
מרגע שגודל המשימה ידוע - אפשר לחשב את סך המשימות הנדרשות לביצוע.

כל מחרוזת שתפוענח תעבור תהליך של פירוק והשוואה מול מילון המילים בנסיון למצוא האם היא מורכבת ממילים חוקיות.

במידה ונמצאה מחרוזת פענוח המכילה **רק** מילים שכולן נמצאות במילון אזי היא נחשבת למועמדת אפשרית לפענוח מחרוזת ההצפנה.

פענוח הקוד בטכניקת brute force מושפע כמובן ממורכבות הקוד והאפשרויות השונות בו. היות והמשימה יכולה להיות קשה עד בלתי אפשרית באמצעים (הנוכחיים) העומדים לרשותינו, נאפשר מספר רמות קושי של תהליך הפענוח.  
  
נניח כי מחרוזת הוצפנה עם הקוד <3,7,4><C,B,A><V>, הנה תיאור רמות הקושי האפשריות:

1. קל – נתוני הגלגלים הנבחרים (4,3,7), מיקומם בחריצים (3,7,4), זהות המשקף (V) ידועים מראש.

תהליך הפענוח כולל רק את מציאת הפוזיציה הראשונית שלהם (A,B,C).

1. בינוני - נתוני הגלגלים הנבחרים (4,3,7), מיקומם בחריצים (3,7,4), ידועים מראש.

תהליך הפענוח כולל את מציאת הפוזיציה הראשונית שלהם (A,B,C) ואת זהות המשקף (V)

1. קשה – נתוני הגלגלים הנבחרים (4,3,7), ידועים מראש.

תהליך הפענוח כולל את מציאת הפוזיציה הראשונית שלהם (A,B,C), זהות המשקף (V), מיקומם בחריצים השונים (3,7,4).

1. ב"א – אף אחד ממרכיבי הקוד אינו ידוע ויש למצוא את כולם באמצעות brute force.

**שימו לב**: לטובת תהליך הפענוח האוטומטי נניח כי אין שימוש בלוח התקעים (בשל אינפלציית האפשרויות).

1. תפקידו של ה DM הוא בעצם לנהל את תפעול הסוכנים, להקצות להם משימות, לקבל מהם תוצאות ולהציגם חזרה למשתמש. נתון קריטי בתהליך זה הוא כמות הסוכנים העומדת לרשות ה DM (נתון קבוע המגיע מקובץ ה XML), וגודל המשימה הניתנת לכל אחד מהם (נתון דינמי הניתן לעדכון ושינוי על ידי המשתמש).

בתרגיל זה, כדי למקסם את יכולת המחשוב העומדות בפנינו, כל סוכן יהווה thread עצמאי (בתרגיל 3 נגדיל לעשות אף מעבר לכך). לכל סוכן יש מספר מזהה ייחודי לו (כרצונכם ; אפשר ספרור רץ). כל סוכן יודע לקבל משימה ולעבוד עליה. התוצר של כל משימה יכול להיות אוסף של מחרוזות הנחשבות מועמדות לפענוח נכון של המחרוזת המוצפנת. (ייתכן גם שמשימה תסתיים ללא שום מועמדים לפענוח)

ה DM יתחזק תור של משימות שממתינות לביצוע. כל הסוכנים (ה thread'ים) ממתינים על התור וברגע שמשימה נשלפת על ידי מי מהם – היא מבוצעת על ידו. בגמר העיבוד הסוכן חוזר וממתין לקבל משימה נוספת (ככל שכזו קיימת).  
  
כפונקציה של קושי המשימה, כמות המשימות תגדל בצורה ניכרת. לא מן הנמנע כי ברמות קושי הגבוהות (קשה וב.א.) תגיעו לכמויות משימות כאלה שרק הנסיון לייצר את כולן (כמופעי אובייקטים) עלול לגרום ל Out Of Memory Error (נקרא בקיצור גם OOM) שימנע מהתוכנית לעבוד.   
כדי להימנע ממצב (מביך) זה, ה DM לא יוכל לייצר את כל המשימות על ההתחלה ובהתאם לכך גודל תור המשימות (כלומר סך המשימות הממתינות לביצוע) צריך להיות מוגבל. ה DM יחזיק thread שתפקידו לייצר את המשימות הבאות על פי הסדר ולדחוף אותן לסוף התור. אם התור מלא הוא יצטרך להמתין עד שהוא יתרוקן מעט (ע"י הסוכנים שמשכו משימות) ורק אז יוכל לדחוף את המשימות הבאות בתור.

בסיום ביצוע של כל משימה בודדת (ע"י הסוכן), ולפני המעבר למשימה הבאה (ככל שזו קיימת), הסוכן מדווח חזרה ל DM על כל המועמדים שנמצאו במהלך המשימה, ועל הזמן הכולל שלקח לו לבצע את המשימה כולה.   
ה DM ידווח חזרה למשתמש על כל המועמדים שנמצאו, כמו גם ידאג לתחזק סטטיסטיקה לגבי ממוצע הזמנים שלקחו כל המשימות.   
היות וסוכנים רבים יכולים להגיש תוצאות גם כן יחדיו - הרי שגם כאן הסוכנים יגישו את התוצאות לתור ייעודי עליו מאזין ה DM והוא יקרא את הנתונים מהם ויציג אותם למשתמש. (נו... עוד thread...).  
כמות המועמדים לפענוח לא צפוייה להיות אסטרונומית ולכן תור זה לא צריך להיות מוגבל בגודלו.

בכל מי מאפשרויות הקושי – פועלו של הסוכן אינו משתנה והוא אחראי אך ורק על מציאת הפויזיציה הנכונה של הגלגלים השונים. כל שאר המרכיבים השונים בקוד הראשוני נקבעים אך ורק ע"י ה DM (זהות הגלגלים, הסדר בינהם, זהות המשקף).

פועל יוצא מכך הוא כי ייתכן ויהיו משימות קטנות מהגודל הרצוי (בקצוות)

שימו לב כי על כל סוכן להחזיק בעותק משלו של המכונה, וזאת כדי לא להפריע ולדרוס את פעולותם של סוכנים אחרים העובדים במקביל לו.

עבור כל מחרוזת מועמדת לפענוח יש לציין את כל הגורמים שהביאו למציאתה:

* + 1. קונפיגורציה (הקוד) שבה היא נמצאה
    2. מי הסוכן שמצא אותה (מספר מזהה)

1. הנה דוגמא לתהליך ההצפנה-פענוח כולו:

במכונה עם אלף-בית של 26 אותיות ABC, תו הרווח ותו הנקודה (סה"כ 28 תווים), העובדת עם 3 רוטורים, והמוטענת בקוד הבא:

<2,3,1><ABC><III>

המשתמש הקליד את המחרוזת:

[[**HELLO. WORLD**

(סה"כ 12 תווים)

נניח כי תו ה . אינו חוקי על פי המילון, ולכן ראשית נפשיט אותו מהמחרוזת המתקבלת:

[[**HELLO WORLD**

(סה"כ 11 תווים)

נפרק אותו למילים הבודדות שבו לפי תו הרווח (**HELLO** ו **WORLD**) ונוודא כי כולן נמצאות במילון. במידה וכן נמשיך לתהליך ההצפנה (שימו לב כי תהליך ההצפנה **כולל** את תו הרווח שהוכנס. הפירוק לפי תו הרווח היה רק כדי לוודא תקינות מול המילון)

לאחר ההצפנה קיבלנו את בליל התווים הבא (מוקף ב [ ] וצבוע **בכחול** לשם הבהרה):

[**DKR RI JQJE**]

נניח כי ל DM יש 3 סוכנים, וגודל המשימה הוא 10. דרגת הקושי היא הקלה ביותר – כלומר ידוע לו כי מדובר ברוטורים 1,2,3 המופיעים בסדר 2,3,1, וכי נבחר משקף III. כל שנותר לו לגלות הוא את הפוזיציה הראשונית של הרוטורים.

היות ומדובר באלף-בית בגודל 28 תווים, אזי סך כמות האפשרויות עומד על 3^28 (= 21,952 אפשרויות).

ה DM יחלק את טווח האפשרויות למשימות בנות 10 קודים כל אחת. בכל משימה נתוני הרוטורים, זהותם, הסדר בינהם והמשקף ידועים וקבועים. נותר לכל סוכן רק לסרוק את הקודים האפשריים:

N

Z . - …

2

AAK - AAT

1

AAA - AAJ

**…**

נניח כי אחד הסוכנים קיבל את משימה מס' 2, וכחלק ממנה הגיע לשלב שבו הוא בוחן את הקוד:

<2,3,1><AAM><III>

ויצא לו בעקבות זאת המחרוזת המפוענחת הבאה:

[**IDU DER SSQ**]

עכשיו המחרוזת תפורק ל 3 מילים על פי תו הרווח (**IDU**, **DER**, **SSQ**), ותתבצע בדיקה האם **כל** שלושת המילים נמצאות במילון.

אם כן הרי זו מחרוזת מועמדת לפענוח. אם לא – אז לא.

שימו לב כי ברמות הקושי מעבר לרמה הקלה (שבהן יש למצוא את זהות המשקף ו/או זהות הרוטורים ו/או מיקומם בחריצים בנוסף למיקומיהם ההתחלתיים), אזי ה DM הוא זה שיכין רשימת חלוקה למשימות עבור המשקף ובחירות הרוטורים השונה וכו' והסוכנים מבחינתם עדיין יצטרכו לרוץ רק על מיקומי הרוטורים השונים.

1. חוץ מיכולת חדשה (ומגניבה !) זו, יש לאפשר את ההתנהלות והעבודה מול המכונה כשם שהיה בתרגיל 1 – כלומר לתת יכולת לראות את פרטי המכונה, לקבוע (ולאפס) תצורת הפעלה ראשונית, להצפין ולפענח צפנים, לצפות ולקבל בסטטיסטיקה והסטוריית פעולות על המכונה – אולם הפעם כחלק מהממשק הגרפי.   
    **שימו לב**:

* פענוח והצפנה אלה אינם עובדים עם המילון. הם עובדים כשם שהיה נהוג במהלך תרגיל 1.
* במסגרת הנגשת היכולת הבסיסית של תפעול המכונה **יש לאפשר** את השימוש בלוח התקעים, כפי שהיה בתרגיל 1

1. מהלך העבודה:

יכולות האפליקציה בשלב זה ירוכזו בשלושה אפיקים (מסכים) מרכזיים.   
עיקבו אחר קובץ מתאר המע' כדי להכיר את המבנה המצופה. המבנה מגדיר את קוי המתאר הכלליים של האפליקציה.   
**חובה להיצמד אליו**.

* 1. מסך 1 – הגדרות ונתוני מכונה:

במסך זה תתבצע טעינה של קובץ המכונה מקובץ ה xml. בסיומה יוצגו פרטי המכונה (כפי שהוצגו בתרגיל 1).

קובץ ה XML יכיל הפעם גם את המילים המותרות לשימוש לטובת הצפנה/פענוח. שימו לב כי המילים במילון יכולות להיות בכל case אפשרי ועליכם להתמודד עם זאת כהלכה.

המטרה היא להציג למשתמש את פרטי המכונה כדי שיחליט אם הוא רוצה להמשיך לעבוד איתה או לא (על פי המפרט המתואר בפקודה 2, תרגיל 1)

עליכם לבצע בדיקות קלט תקין על קובץ המשחק. כל הבדיקות שבצעתם בתרגיל הקודם תקפות. בנוסף יש לוודא כי **כמות הסוכנים לא תעלה על 50, ולא תפחת מ 2 (50 => agents =>2)**

יש לאפשר לטעון כמה קבצים אחד אחרי השני. כל קובץ שנטען בהצלחה מחליף את קודמו. ברגע נתון יש (לכל היותר) רק קובץ אחד (כלומר תיאור מכונה אחת) הטעון במע'.

טעינת קובץ ה xml תתבצע באמצעות file chooser dialog בלבד ! (אין להניח שהקובץ יחכה לכם בספרייה ייעודית, או שהבודק יקליד את תוכנו לתוך תיבת טקסט וכו'.).

הקובץ יכול להימצא בכל ספרייה חוקית על המחשב (כולל ספריה עם רווחים). **נא לוודא כי הדבר אינו מכשיל אתכם.**

המשתמש מגדיר ומכייל את הקוד במכונה. יש לאפשר למשתמש לבחור בקלות את כל מרכיבי הקוד כפי שהוגדרו בתרגיל 1, פקודה 3 (כולל לוח התקעים). בסיום קביעת הקוד הנדרש המשתמש ילחץ על כפתור ייעודי (set !) שיקבע ויכניס את הקוד למכונה.  
קוד זה ישמש את המכונה במהלך תפעולה באפיקים הנוספים (ראו פירוט בהמשך). בכל תפעול של המכונה יש לדאוג ולעדכן בכל רגע נתון על איזו קונפיגורציה מצויה המכונה כרגע (בעקבות הפסיעות השונות).

**שימו לב**: יש להפריד בין האזור שבו קובעים את הקוד, לבין האזור שמראה את הפוזיציה הנוכחית של רכיבי המכונה.

כמו כן הציעו למשתמש כפתור נוסף שמאפשר לו לבקש שייבחר עבורו קוד אוטומטי לתפעול המע' במקום ההכנסה הידנית (פקודה 4, תרגיל 1)

* 1. מסך 2 – תפעול המכונה: הצפנה / פענוח ידני:

במסך זה יתאפשר למשתמש לבצע הצפנה/פענוח של מחרוזות כשם שהיה נהוג בתרגיל 1.

לאחר שקוד המכונה נקבע במסך 1 (אם בצורה אוטומטית ואם בצורה ידנית) המשתמש יקליד את המחרוזת והיא תוצפן.

יש לאפשר למשתמש לבחור באחד משני סוגי תפעול המכונה:

* + 1. מצב אוטומט/רציף (מחרוזת שלמה)  
       באפשרות זו המשתמש יקליד את כל המחרוזת וכשיסיים ילחץ על כפתור ייעודי (process) ואז תתבצע ההצפנה כפי שהתרחשה במסגרת תרגיל 1. בסיום התהליך תוצג המחרוזת המוצפנת כמו גם מצב הקוד הנוכחי (בעקבות הפסיעות). מחרוזת זו נכנסת כמובן לסטטיסטיקה המוצגת. הציגו כפתור Clear המאפשר לנקות את אזור מחרוזת הקלט והפלט ולעבד מחרוזת חדשה.  
       **שימו לב**: לחיצה על Process או Clear לא מאפסות את הקוד חזרה למצבו המקורי !
    2. מצב ידני/בודדת (תו אחרי תו)

ההצפנה במסך זה תתבצע תו אחרי תו. (ראו בונוס בהקשר זה). המשתמש יקליד תו ומייד יראה את תוצאת הצפנתו הנצברת באזור המראה את מחרוזת הפלט. (ראו בונוס המקלדות בהקשר זה)  
לאורך תהליך ההצפנה/פענוח תוצג הפוזיציה הנוכחית של המכונה כפונצקיה של השינוי במיקומי הרוטורים עם עיבוד כל תו.   
הציגו למשתמש כפתור Done המאפשר לחתום את מלאכת ההצפנה הרציפה, ולנקות את מחרוזת הקלט לקראת הכנסת קלט חדש. זה השלב שבו המחרוזת שנצברה נחתמת ונכנסת גם כן לסטטיסטיקה המוצגת.

**שימו לב**: לחיצה על Done לא מאפסת את הקוד חזרה למצבו המקורי !

בלי קשר לאופן תפעול הקלט, הציגו למשתמש כפתור reset המאפשר להחזיר את הקוד למצבו ההתחלתי (כפי שנקבע במסך הראשון) כדי להמשיך להצפין/לפענח מחרוזות נוספות.  
שימו לב: לחיצה על reset "באמצע" ההקלדה של התווים (בלי קשר למצב התפעול) למעשה מבטלת את הפעולה, מנקה את השדות והמחרוזות שכבר הוקלדה **לא** נכנסת לסטטיסטקה. (והקוד כמובן מתרסט חזרה לקוד המקורי)

במסך זה יש להציג גם את ההסטוריה והסטטיסטיקה שמתחילה להצטבר במכונה בכל רגע נתון, כפי שהוגדר בתרגיל 1,פקודה 7.

* 1. מסך 3 – הצפנה / פענוח אוטומטי:

במסך זה תתבצע הצפנה ופענוח אוטומטי באמצעות DM.   
אפשר להניח כי קוד המכונה **לא יכיל** תקעים בשלב זה (אין צורך לוודא או לבדוק זאת)  
  
המשתמש מקליד הודעה בתיבת ההצפנה ומבצע הצפנה שלה. הפלט (המחרוזת המוצפנת) מוצג למשתמש.

עליכם לוודא כי המחרוזת להצפנה מכילה אך ורק מילים הקיימות במילון. במידה ואינה כזו – אין להצפין אותה ויש להתריע על כך בפני המשתמש.

לשם הקלה על המשתמש – יש לאפשר לו לראות את המילים הקיימות במילון. יש מספר דרכים להראות זאת ואתם חופשיים להחליט כרצונכם איך לממש יכולת זו.

[**המלצה**: חלונית שמראה את כל המילים ומעליה תיבת חיפוש. כל הקלדה בתיבת החיפוש מציגה בחלונית המילים אך ורק את המילים המתחילות במחרוזת שכרגע מצויה בתיבת החיפוש. כך יקל על המשתמש להבין אילו מילים הוא יכול להשתמש ואילו לא. ממליץ לקרוא ולהשתמש במבנה נתונים שנקרא Trie. כמו כן שימו לב לבונוס 3 בהקשר זה.]

גם כאן יש להציג את פוזיצית המכונה לאחר תהליך ההצפנה כמו גם את כפתור ה reset שיאפשר למשתמש להחזיר את המכונה למצבה המקורי ישר ממסך זה.

לאחר ביצוע ההצפנה, יש להפעיל את ה DM. כדי לעשות זאת יש לקבוע לו 3 פרמטרים מרכזיים:

1. כמה סוכנים להפעיל במלאכת הפענוח הנוכחית – עד למקסימום המגיע מקובץ ה XML. (קלאסי לסליידר...)
2. דרגת הקושי, כפי שמוגדרת בסעיף 5. (קלאסי לקומבו בוקס)
3. גודל המשימה – מספר חיובי שלם גדול מ 0.

לאחר קביעת הפרמטרים הנ"ל יש להציג את כמות המשימות הנגזרת מהם.

או אז לאפשר יהיה להפעיל את ה DM באמצעות כפתור ייעודי. (start)

מלאכת הפענוח עתידה לקחת זמן. על כן יש לאפשר לעצור, להשהות ולהמשיך (stop, pause, resume) את פעולת ה DM באמצעות כפתורים ייעודיים.

שימו לב על ה UI להישאר חי ומגיב בכל רגע נתון (כשם שראינו בדוגמא המסכמת עם ה task).

מלאכת הפענוח (הארוכה והכבדה) תבוצע (כמובן !) באמצעות TASK אחד שבמסגרתו יורץ כל התהליך מהצד של ה DM. יש להציג את ההתקדמות הכללית של התהליך באמצעות Progress bar שיתקדם ברגע שמשימות מבוצעות ע"י הסוכנים, מתוך סך המשימות הידוע ל DM.

היות וכל סוכן מחזיק בעותק משלו של המכונה, אזי כל פעולת הפענוח האוטומטי לא מתועדת בהסטוריה של המכונה כפי שמוגדר במסך 1..

ה DM ידווח בכל רגע נתון את המועמדים שנמצאו ע"י סוכניו, יחד עם כל הנתונים הנדרשים לכל מועמד (כפי שמוגדר בסעיף 5)

בסיום מלאכת הפענוח (גמר ביצוע כלל המשימות – או לחיצה על stop ע"י המשתמש) יש להציג את הזמן הכולל שלקחה מלאכת הפענוח כולה (כלומר עד לסיום כל המשימות).

בסיום מלאכת הפענוח יש לאפשר לכוונן את DM מחדש ולהתחיל את כל הפענוח מהתחלה.

1. כחלק מבדיקת המשחק ישונה גודל המסך (resize) ותיבדק המע' שלכם במסך בגודל שונה. עליכם לדאוג לסידור נכון של רכיבי ה ui ולוודא את תקינותם גם בגודל קטן. מומלץ מאוד להשתמש ב scroll pane (יש ללמוד עליו לבד) כדי להציג תוכן גדול בתוך שטח מסך קטן. בפרט עליכם לדאוג להראות את לוח המשחק (שיכול להגיע בגדלים שונים ומשונים) בצורה נאותה כך שאף אחד מרכיביו לא יוסתר בגלל גודל מסך קטן מדי. כמו כן חישבו איך לא להסתיר את טבלת השחקנים במשחק כמו גם את היכולת לבצע מהלך ולהתקדם במשחק.

**למען הסר כל ספק ומניעת כל התחכמות שהיא, אין "לטפל" ב resize ע"י כך שפשוט תמנעו מהמסך להיות resizable.**

**במידה ומימשתם בונוסים בתרגיל הקודם אין הכרח לגרום להם לפעול גם בתרגיל זה, אולם אם זה מתאפשר זה יחמם את ליבי (למען הסר ספק, חימום ליבי אינו מתורגם להעלאת נקודות).**

## חלוקה למודולים

יש לייצר מודול נוסף שממנו ייבנה ה Jar של שכבת ה UI החדשה ב Java FX. מודול זה יעבוד מול מודול המנוע (ושאר המודולים, במידה וקיימים) מתרגיל 1.  
בהחלט סביר שתצטרכו לשנות במעט את המנוע לטובת העבודה על תרגיל 2. **אין שום צורך או ציפייה** לדאוג לכך שמודול ה UI של תרגיל 1 ימשיך לעבוד כהלכה מול המנוע.

## איך מתחילים ?

התחילו בהקמת השלד הראשי של המע' בהתאם לקווי המתאר של האפליקציה.   
התחילו בהקמת המסך הראשון – שאמור להתכתב ולתפעל את היכולות הקיימות שלכם במע'. טענו קובץ והציגו את פרטי המכונה.  
חישבו איך לייצר רכיבים גנרים (להצגת קוד מכונה למשל) שיאפשרו לכם להשתמש בהם גם במסכים השונים.  
עיברו לממש את היכולת לקבוע את קוד המכונה בצורה נאותה.  
התחילו לממש את המסך השני שבו מתאפשר תפעול המכונה בהתאם למה שהיה בתרגיל 1.

רק בסוף התחילו את המימוש של חווית הפענוח/הצפנה באמצעות ה DM והסוכן. ראשית התמקדו ביצירת היכולת הנ"ל במנוע. נסו להפעיל אותה אפילו מהממשק של תרגיל 1, או מתוך קוד, ולו רק כדי לוודא שהיא עובדת בבסיסה בצורה נכונה. לאחר מכן התמקדו בבניית מסך מספר 3 שמאפשר את התפעול הבסיסי של מלאכת הפענוח. השתדלו להשתמש בכמה שיותר רכיבים אפשריים ממסך 1 ו 2 (למשל הרכיב המציג את הקוד המתעדכן כל רגע שהמכונה זזה וכו')  
כשהכל עובד התעסקו עם יצירת כפתורי התפעול הדינמים של המע' (pause, resume, stop) ורק בסוף עם סוגיית הפעלת הסוכנים באמצעות task עצמאי וכו'.

## בונוסים

היכן שזה רלבנטי (בונוס 1 ו 2) על הבונוסים להגיע "מכובים" בתור התחלה כך שהבודק "יפתח" אותם רק לטובת הבדיקה שלהם.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | הגדלת ראש נאה | אפשרות החלפת skin למשחק. בבונוס זה תממשו (לפחות) 2 ערכות צבעים נוספות על הערכה הדיפולטית, ותאפשרו למשתמש להחליף את ערכות הצבעים. שימו לב ערכת הצבעים כוללת (לכל הפחות):   1. החלפת רקע של כל המסך 2. החלפת המראה של הכפתורים על המסך 3. החלפת הפונט והגודל של כל ה label על המסך | כי זה בכל זאת תרגיל בממשק גרפי – ומה יותר גרפי מזה ??? | **5 נקודות**  **(ועד ל 100)** |
| 2 | הגדלת ראש קלילה | ממשו 2-3 אנימציות המלוות את תהליך הפענוח/הצפנה. למשל: הזזת גלגלי המכונה תוך ביצוע ההצפנה, אות אחרי אות..  האנימציה צריכה להתבצע לכל היותר במשך 2 שניות  יש לאפשר גם לנטרל את האנימציות הנ"ל, כך שהיא לא תאיט את התקדמות המשחק. | כל כך קל... | 5 נקודות  (ועד ל 100) |
| 3 | הגדלת ראש חביבה | אפשרו למשתמש לבחור מילים ישירות מתוך המילון ע"י לחיצה על המילה הנבחרת (חסכו לו את ההקלדה). | מגניב לאללא | 3 נקודות  (ועד ל 100) |
| 4 | אותנטי | מקלדות:  במסך השני (במצב התפעול הידני/בודד, בו מתבצעת הצפנה אות אחרי אות) הציגו למשתמש 2 מקלדות: מקלדת הקלט ומקלדת הנורות.  אפשרו למשתמש להכניס תו להצפנה (מקלדת רגילה או ממקלדת הקלט). תוצאת הצפנת התו תוצג/תודגש בצורה כלשהיא (אנימציה !) במקלדת הפלט.  שימו לב כי גודל המקלדות הוא דינמי (שכן הוא פונצקיה של האלףבית של המכונה והוא דינמי). חישבו על דרך נאותה לסדר את המקלדות בהתאם (רמז FlowLayout...) אם רוצים אפשר גם להשתמש באותה המקלדת (במקום שתי מקלדות נפרדות) כך שלחיצה על מקש אחד תאיר/תדגיש מקש אחר שהוא האות המוצפנת. | כמה קל ופשוט – וכמה מדהים מבחינת החוויה ! | 8 נקודות **(מעל ל 100)** |
| 5 | תדהימו אותי ! | הציגו את המסלול שעובר האות במסגרת העיבוד הידני על כל תו ותו. דוגמא לרעיון אפשרי ניתן לראות [כאן](https://www.youtube.com/watch?v=sbm2dmkmqgQ&list=WL&index=2&t=694s) (בין דקות 3,4,5) זה המקום להיות קריאטיבים ומקוריים באופן שבו הדברים מוצגים | כי זה יעשה אותי הכי גאה בעולם... | 12 נקודות **(מעל ל 100)** |
| 6 | אנטומיה | הציגו את המבנה והחיווטים הפנימיים של הרוטורים והמשקף במסגרת הצגת פרטי הקובץ במסך 1. חישבו על דרך מקורית ומעניינת להציג את המיפויים הפנימיים שבתוך הרוטורים/משקפים.  זה יכול להיות בדמות 2 טורים של אותיות כפי שמוגדר בקובץ ה XML ואז hover על אות בעמודה אחת מדגיש/מחבר את חברתה בעמודה אחרת או כל טכניקה אחרת שתמצאו לנכון | אושר צרוף של הבנה ותפעול נבכי המע'. | **7 נקודות**  **(ועד ל 100)** |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. jar (אחד או יותר) שהוא כל הקוד שלכם, בצירוף קובץ אצווה (batch) שהפעלתו תריץ את התוכנית
2. קבצי קוד המקור של הפרויקט שלכם.
3. 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.
4. 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...)
5. במידה ומימשתם את מי מהבונוסים לעיל, ציינו זאת בקובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

## שאלות ותשובות

|  |  |  |  |
| --- | --- | --- | --- |
| # | תאריך | שאלה | תשובה |
|  |  |  |  |

תרגיל 3 – מימוש **Cracking The Enigm-** כאפליקציית Client - Server (35%) – הגשה: 29.10.22

## פרטים יבשים

צפי תחילת עבודה: **20.9.22** תאריך הגשה: **29.10.22**

צפי זמן לביצוע: **5 שבועות +** ציון אפשרי מקסימלי: **110**

משקל התרגיל: 35**%** קושי: **ווא'עליה אלל'בחטי**...

**מטרות התרגיל העיקרית**

1. מימוש המערכת כאפליקציית client-server.
2. מימוש תחרות בין צוותים שונים המנסים לפענח הודעות מוצפנות

## דרישות

1. בתרגיל זה נממש את היכולת לקיים תחרות בין צוותי פענוח אוטומטים שונים.   
   מחד נוכל לטעון הגדרות ותצורות שונות של המכונות ולהפיץ הודעות מוצפנות.   
   מנגד עומדים מספר צוותי פענוח אוטומטים הנבדלים בינם לבין עצמם בפרמטרים השונים (כמות סוכנים, גודל משימה וכו') ומתחרים בינם לבין עצמם מי יצליח לפענח את ההודעה המוצפנת ראשון (ובאם בכלל...)
2. בתרגיל יתקיימו 4 יישויות מרכזיות:

* Uboat – צוללת גרמנית האחראית על קביעת קוד ראשוני למכונה, הפצת הודעה מוצפנת והכרעה מי מהצוותים השונים הצליח לפענח את ההודעה בצורה מדוייקת (מנצח בתחרות)
* Allies – צוותי פענוח של בנות הברית המתחרים בינם לבין עצמם מי הראשון שיצליח לפענח את ההודעה המוצפנת ששידרה ה UBoat.
* Agent – סוכנים החברים בצוותי הפענוח השונים ואחראים על ביצוע משימות הפענוח עצמן. כל סוכן חבר בצוות אחד בדיוק.
* Battlefield – שדה הקרב שבו מתרחשת התחרות. הגדרות שדה הקרב כוללות את השם, כמה צוותי פענוח משתתפים בכל תחרות ואת רמת הקושי של הפענוח (כפי שהוגדרה בתרגיל 2, סעיף 5). בכל battle משתתפת בדיוק צוללת גרמנית אחת. וצוות פענוח אחד או יותר.

1. השרת (tomcat) יכיל את הגדרות התחרויות השונות המתקיימות במע'.

כל סוג משתמש המתחבר לשרת מתחיל את מסעו בעמוד רישום למע' ובו הוא צריך לבחור אם הוא יישות מסוג uboat, alies או agent.

יישות מסוג Uboat יכולה להעלות קובץ משחק (XML) המגדיר את התחרות ופרטיה השונים.   
רק Uboat יכול להגדיר תחרות ע"י העלאת קובץ.  
Uboat יכול רק ליצור תחרות חדשה משלו. הוא אינו יכול להשתתף בתחרות שיש בה כבר Uboat אחר. (במילים אחרות בכל תחרות יש בדיוק Uboat אחד בלבד).

יישות מסוג Allies יכולות להתחבר לשרת ולבחור את התחרות אליה היא רוצה להירשם (רק תחרות אחת בכל רגע נתון). כל שחקן מסוג Allies הוא למעשה DM (מתרגיל 2) המתנהל בצד השרת. היות ויש כמה Allies בתחרות – הרי שמתקיימים גם בצד השרת כמה DM'ים שונים. במקביל.

יישות מסוג Agent יכולה להתחבר לשרת ולבחור את צוות ה Allies שהיא חברה בה. כל Agent חבר בדיוק בצוות אחד. פרט זה נקבע במהלך הרישום ואינו יכול להשתנות לאורך חיי האפליקציה.

כשמתקיימים התנאים הנדרשים, מתכנסים כל השחקנים שנרשמו עבור תחרות מסויימת והיא מתחילה

Agent

AgenEnigma-Ex3.xsdt

Agent

Allies  
(Client)

Enigma  
BattleField  
Server

UBoat  
(Client)

UBoat  
(Client)

**HTTP**

**HTTP**

**...**

UBoat  
(Client)

Agent

AgenEnigma-Ex3.xsdt

Agent

Allies  
(Client)

1. מהלך התחרות:

התחרות מתחילה כשכל השחקנים (DM, Uboat) מודיעים כי הם מוכנים. ה UBoat יבחר קוד ויצפין הודעה מסויימת (הודעה תקינה, ממילים שבמילון). ההודעה עוברת ומופצת לכלל צוותי ה Allies השותפים בתחרות. כל צוות מקצה את המשימות השונות לסוכנים איתם הוא עובד. הסוכנים בתורם מתחילים את מלאכת הפענוח ושולחים מועמדים אפשריים חזרה ל DM והוא בתורו שולח אותם חזרה ל UBoat (זה התהליך מבחינת התחרות. לאו דווקא מבחינת התנהלות המידעים במע.).   
ה UBoat מקבל אוסף של מועמדים מצוותי ה Allies השונים ורק הוא יהיה זה שיקבע מי המועמד המנצח (היות ורק הוא יודע מי הוא המועמד המתאים למחרוזת המקור).

**שימו לב**: היות ועוסקים בפענוח אוטומטי, אפשר להניח כי קוד המכונה לא יכיל תקעים.

כל צוות נבדל זה מזה בכמות הסוכנים (הדינמית) שהוא יכול להקצות וכמובן בגודל המשימה.

הסוכנים בתרגיל זה יממושו כתהליך נפרד (process), שיפעל (לכאורה) ממחשב(ים) אחר(ים).

ה DM עצמו רץ בתוך השרת (כחלק ממנוע המע'). הסוכנים (המשוייכים לכל DM) עובדים גם עצמאית מול השרת (תוך הזדהות לאיזה צוות DM הם שייכים).

כאשר DM (=Allies ) נרשם למשחק הוא קובע לעצמו שם ייחודי. שם זה ישמש אותו לתקשורת מול הסוכנים שלו. כאשר תהליכי הסוכנים יוקמו, הם יוגדרו לעבוד עם/מול DM מסויים ובכל פניותיהם לשרת ישאו את שם ה DMשהם מתקשרים איתו על גבם (רמז רמז).

תהליך הסוכן עצמו יהיה אפליקציית Java FX עצמאית. במהלך הרישום לשרת (מסך הלוגין) על הסוכן לספק את שם ה DM שהוא חבר בצוותו.

בתרגיל זה אין מגבלה על כמות הסוכנים האפשרית עבור DM מסויים. ניתן להקים כמה סוכנים שרוצים, בצורה דינמית, וכך יכולים להתקיים מספר צוותי פענוח שונים – כל אחד בהתאם למשאבים (כוח המחשוב) שברשותו.

תהליכי הסוכנים הינם תהליכים אוטומטים אשר **אינם דורשים כל התערבות יד אדם בתפעולם** (למעט הפעם הראשונה שמרימים אותם לאוויר). מרגע שתהליך סוכן עולה – הוא אוטומטית מתקשר ל DM הרלבנטי (באמצעות השם הייעודי) ומשם והלאה מקבל ממנו משימות ויודע להתנהל עצמאית. הסוכנים משוייכים ל DM אחד בלבד לכל אורך חייהם. הם לא יכולים להחליף DM וכו'  
  
כל מה שרואים באפליקציית הסוכן הוא מידעים על פועלו ומצבו.   
יש להציג:

* 1. שם צוות ה Allies שהוא חבר בו
  2. מידע על התחרות הנוכחית שהוא משתייך אליה
  3. מידע על המשימה שהוא עובד עליה כרגע (אם הוא עובד על משימה)
  4. מועמדים לפענוח שנמצאו על ידו ומועברים חזרה ל DM.

התקשורת בין ה DM לסוכנים שלו עוברת דרך השלבים הבאים (לפחות, ייתכן ויהיו לכם יותר):

1. ה DM צריך להעביר לסוכן את פרטי המכונה איתה הוא יעבוד.   
   שימו לב **לא תוכלו** להעביר את הנתיב לקובץ ה XML המקורי – מאחר ומדובר (פוטנציאלית) על מחשבים שונים פיזית.

תצטרכו למצוא דרך להעביר את המידע עצמו מעל גלי האתר... (רמז json ?)

פרטי המכונה כוללים גם את מבנה הגלגלים, המשקפים והמילון.

1. הסוכן צריך להודיע ל DM מתי הוא מוכן לקבל משימות
2. ה DM צריך לשלוח לסוכן משימות לביצוע
3. הסוכן צריך להחזיר ל DM חזרה מועמדים מתאימים (כפונקציה של ביצוע המשימה).
4. 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.
5. אין צורך לבצע שמירה של נתונים מעבר למופע הנוכחי של השרת. במילים אחרות כשהשרת יורד - כל הנתונים נעלמים.

לא שומרים את המשתמשים שנרשמו, הללוואות, הטרנזקציות וכו'.

1. יש לדאוג ולוודא כי מסכי המשחק לסוגם מתנהגים יפה בresize ולא מחסירים שום פרט
2. יש הרבה רכיבים זהים בין האפליקציות השונות. יש לשאוף להשתמש באותם הרכיבים כמה שרק אפשר כדי לחסוך עומס תכנותי ולשמר התנהגות זהה בין האפליקציות השונות. כמו כן חלק ניכר מהרכיבים כבר קיימים לכם במסגרת תרגיל 2. יש לשאוף למחזר ולהשתמש בהם כמה שרק אפשר.

## אפליקציית UBoat

בפתיחת האפליקציה יש לבצע רישום למע'.  
לכל UBoat יש שם משלה. השם צריך להיות ייחודי ויש לאכוף זאת ולהתריע במידה ומנסים להירשם עם שם שכבר קיים.

אחרי רישום מוצלח UBoat יכול להעלות קובץ תחרות למע'.  
המשתמש בוחר קובץ משחק (ממחשב הקליינט) ומעלה אותו אל השרת. ההעלאת 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 !). ייתכן ולשרת (של הבודק) אין הרשאות לכך ואתם תקרסו !

כשתוכן הקובץ מגיע לשרת, יבוצעו בו בדיקות הקלט כפי שמפורט מטה.

באם הקובץ תקול – הודעה מתאימה הכוללת את סיבת התקלה צריכה להופיע למשתמש (ולמשתמש הזה בלבד), והקובץ **אינו** מוכר כתקין בצד השרת. אם הקובץ תקין – הודעה מתאימה צריכה להגיע למשתמש.

קבצי ה XML יהיו מוגדרים על פי סכמה חדשה של תרגיל 3. הקובץ יכיל מבנה הנתונים הנקרא Battlefield, המגדיר את:

* שם שדה הקרב (game-title)
* כמות צוותי בנות הברית (מפענחים אוטומטים) השותפים לתחרות.
* קושי התחרות (בהתאם לתאור בתרגיל 2, סעיף 5)

יש לבצע בדיקות קלט על הקובץ, ובמידה והוא לא תקין להתריע בפני המשתמש ולא לטעון את המשחק.

בפרט יש לבדוק את הדברים הבאים:

1. אין לאפשר לטעון משחק בשם שכבר קיים במע'.
2. כל בדיקות התקינות הרלבנטיות שביצעתם בתרגיל 1 ו 2 צריכות לבוא לידי ביטוי גם כן.

ברגע שהעליתם קובץ למע', ולאחר וידוא חוקיותו, תועברו למסך התחרות.  
המשתמש יכול לקבוע את קוד המכונה הראשוני (עפ"י הכללים הנהוגים בתרגיל 1 ו 2) ומצפין הודעה חוקית (כזו המכילה רק מילים מקובץ המילון). כשסיים הוא לוחץ על כפתור ייעודי שמודיע כי הוא מוכן להתחיל בתחרות (כפתור ready).

יש להציג במסך זה מידע על הצוותים ויכולותיהם השונות המשתתפים בתחרות. עבור כל צוות:

1. שם הצוות
2. כמות הסוכנים
3. גודל המשימה

ברגע שהתחרות תתחיל (כלומר התכנסו כל צוותי הפיתוח המתחרים הנדרשים), הוא יקבל על כך חיווי מתאים ועכשיו ממתין לקבל זרם של הודעות המועמדות לפיצוח נכון של ההודעה שהקליד. ההודעות ממשיכות להגיע עד אשר נמצאת מועמדת מתאימה. כשזה קורה המערכת תזהה זאת אוטומטית ותסיים את התחרות.   
  
עבור כל מחרוזת מועמדת יש להציג את המידע הבא:

1. המחרוזת עצמה
2. שם צוות ה Allies שפענח אותה
3. הקוד לפיו היא פוצחה

יש להציג את כל המחרוזות בחלונית סבירה עם scroller – שכן צפויות להגיע הרבה מחרוזות. (ממליץ ב table view)

כשמשחק נגמר שחקן ה Uboat נשאר במסך התחרות שמתרוקן משחקניו הקודמים והוא ממתין להגיעם של שחקנים נוספים לטובת התחלת משחק נוסף. אם/ברגע שיגיעו כאלה – הוא יקבל על כך כמובן חיווי והכל מתחיל מהתחלה.

בגמר משחק בלבד, יש להציג כפתור Logout אשר יאפשר לשחקן uboat לפרוש מן המע'. בעשותו כך הוא נמחק מרישומי המע'. קובץ התחרות שהעלה נמחק גם הוא ואינו זמין יותר להשתתפות ע"י שחקנים. באם כבר רשומים שחקנים לקובץ זה – הם מועברים חזרה למסך הרישום כדי לבחור תחרות אחרת להירשם אליה (כולל הודעה וחיווי נאותים כמובן).

שימו לב: **אין צורך להציג את כפתור ה Logout במהלך קיומה של תחרות !**

## אפליקציית Allies (DM)

בפתיחת האפליקציה יש לבצע רישום למע'.  
לכל Allies יש שם ייחודי לו ויש לאכוף זאת ולהתריע במידה ומנסים להירשם עם שם שכבר קיים.  
לאחר רישום מוצלח עוברים ל**מסך מידע התחרויות**.

במסך זה יוצגו רשימת הסוכנים הפעילים שהועלו ומוכנים לקבלת העבודה.   
בכל פעם שמעלים תהליך חדש של סוכן (ראו פירוט בהמשך), הוא נרשם בשרת ואז ה DM מקבל על כך חיווי ויוכל לעדכן את רשימותיו.   
עבור כל סוכן יש להציג את שמו.  
  
כמו כן במסך זה המשתמש יוכל לראות את המידע על כלל התחרויות המתקיימות במע' ברגע נתון.  
עבור כל תחרות יש להציג:

* שם ה ה battle field
* שם המשתמש שהעלה את המשחק (שם ה UBoat)
* סטטוס המשחק (ממתין/פעיל)
* רמת הקושי של המשחק
* כמה צוותי פענוח נדרשים לתחילת המשחק + כמה כבר "רשומים" לו.  
  (לדוגמא: 2/3 – צריכים 3 צוותים רשומים כרגע שניים...)

המשתמש ייבחר באיזו תחרות הוא רוצה להשתתף. ברגע שנבחרה התחרות הרלבנטית, כשהשחקן מוכן לכך, יש ללחוץ על כפתור ייעודי (ready) ואז תועברו **למסך התחרות**. אין לאפשר רישום לתחרות שכבר "מלאה" (כל הצוותים הנדרשים לה כבר נרשמו)

ב**מסך התחרות** יוצגו מס' מידעים:

* פרטי התחרות (כפי שהם מוגדרים למעלה).
* פרטי הצוותים המתחרים. עבור כל צוות:
  + שם הצוות
  + כמות הסוכנים
  + גודל המשימה
* פרטי הסוכנים הפעילים והספק עבודתם. עבור כל סוכן:
  + שם הסוכן
  + סך משימות שקיבל ; כמה משימות עוד עומדות לפתחו וממתינות לביצוע (אפשר כ progress bar ! אבל עם מספרים...)
  + כמה מועמדים ראויים הפיק כל סוכן (סה"כ CROSS כל המשימות)

המשתמש יכניס את גודל המשימה שהוא רוצה להשתמש בו (כפונקציה של קושי התחרות וכמות הסוכנים הזמינה לרשותו).

כאשר ה DM סבור שהוא מוכן (מבחינתו כל הסוכנים הוקמו ואפשר לצאת לתחרות) הוא צריך לדווח זאת ע"י לחיצה על כפתור ייעודי (ready).   
שימו לב: אפשר להקים עוד סוכנים גם בשלב זה, כל עוד לא נלחץ כפתור ה READY והתחרות עוד טרם התחילה. במהלך התחרות עצמה ניתן להקים עוד סוכנים אבל הם ממתינים בצד ולא מתווספים פתאום למהלך התחרות עצמה.

ברגע שכל השחקנים (UBoat ושאר צוותי ה Allies) סימנו READY – התחרות תתחיל.   
גם כאן ה DM'ים יפעלו כשם שפעלו בתרגיל 2: יחלקו את המשימות השונות לסוכניהם באחת מ 2 השיטות המדוברות. שימו לב כי כאן השיטה הראשונה (תור לכל סוכן) הגיונית יותר מאחר והיא מאפשרת לסוכן (שעכשיו מגיע בדמות Process עצמאי) לעבוד כיחידה עצמאית בשטח שרק מדווחת תוצאות חזרה לעמדת הפיקוד (ה DM) ולא להטריל את ה DM בכל פעם כשהוא צריך משימה חדשה לעבוד עליה...

יש לעדכן את המידעים הדינמים השונים (בעיקר המידע של התקדמות הסוכנים) תו"כ תנועה.  
יש להציג את המחרוזת המוצפנת שכרגע הצוות עומל על פיצוחה.

יש להציג טבלה המכילה את המועמדים השונים שמצאו הסוכנים של הצוות הנ"ל (המועמדים שנשלחו ל UBoat).   
עבור כל מחרוזת מועמדת יש להציג את המידע הבא (בדומה ל UBoat. רמז רמז):

1. המחרוזת עצמה
2. שם צוות ה Allies שפענח אותה
3. הקוד לפיו היא פוצחה

יש להציג את כל המחרוזות בחלונית סבירה עם scroller – שכן צפויות להגיע הרבה מחרוזות. (המלצה: tableview)

ברגע שה UBoat מזהה מחרוזת מנצחת, התחרות נפסקת והודעה מתאימה תועבר לכלל הצוותים והיא תוצג בצורה ברורה למשתמש לגבי זהות הצוות המנצח. אם בשלב זה יש DM'ים וסוכנים שעוד נמצאים במהלך עבודה – צריך לחדול אותם עם הודעה נאותה. אין להעביר יותר מועמדים ל UBoat (שכן התחרות נגמרה)

בגמר התחרות, אחרי קבלת ואישור ההודעה הרלבנטית מועברים שחקני ה Allies חזרה **למסך המידע על התחרויות** (המסך הקודם), שם הם יכולים להירשם מחדש לתחרות נוספת (אפילו לאותה התחרות אם ירצו, שכן עכשיו היא כבר לא "פעילה" אלא פתוחה לרישום חוזר).

## אפליקציית Agent

בפתיחת האפליקציה יש לבצע רישום למע'.  
לכל Agent יש שם ייחודי לו ויש לאכוף זאת ולהתריע במידה ומנסים להירשם עם שם שכבר קיים.  
כל Agent צריך גם לציין לאיזה צוות Allies הוא משתייך. אפשרו למשתמש לבחור צוות מתוך רשימה של הצוותים הקיימים במע' (אין לאלץ אותו להקליד את שם הצוות !).  
**שימו לב:** ניתן להעלות סוכנים המשוייכים ל DM גם לפני שה DM המסויים בחר תחרות מסויימת להשתתף בו. זהו למעשה תהליך 'בניית הצוותים' (הקמת והכפפת סוכנים ל DM מסויים) והוא קורה בלי קשר לתחרות כזו או אחרת. הסוכנים מסונפים ומשוייכים ל DM שהם בחרו לכל אורך חייהם וחיי השרת. הם ממשיכים עם ה DM הזה גם לתחרויות הבאות שהוא ישתתף בהן.  
בנוסף ה Agent יבחר כמות טרדים שהוא מקצה לטובת העבודה על המשימות. מספר חיובי שלם בין 1 ל 4, כולל.  
לאחר רישום מוצלח עוברים למסך תפעול ה Agent.

ה Agent היא אפליקציה פסיבית. אין לה כפתורים לתפעול – אלא רק לבצע את המשימות ולהציג מידע לגבי המתרחש.  
בהתאם לנוהל חלוקת המשימות של ה DM, ה Agent יקבל את המשימות ויצטרך לבצע אותן באמצעות מי מהטרדים הפנויים שלו לטובת העבודה.   
אם ה DM מרכז את כל המשימות לפתחו של Agent (הגישה הראשונה, והמומלצת) – אזי הוא יקבל את כל המשימות לתוך תור, והטרדים שלו יגמעו את המשימות מהתור ויבצעו אותן.  
אם ה DM מרכז את כל המשימות בתור אחד ויחיד אצלו – אזי ה Agent צריך להתחבר ולמשוך משימות כל פרק זמן כדי לבצען ע"י הטרדים שלו.

כל Agent יציג את המידעים הבאים:

1. שם צוות ה Allies שהוא חבר בו
2. פרטי התחרות שהוא כרגע משתתף בה ועובד עליה
3. רשימה של כל המשימות שהוא קיבל לביצוע ; רשימה של כל המשימות שהוא כבר ביצע (במסגרת התחרות הנ"ל).

Progress bar המציג את התקדמותו כפונצקיה של גמיעת המשימות השונות

1. עבור כל משימה שבוצעה יש להציג את כמות המועמדים שיצאה ממנה.
2. יש להציג בצורה מורכזת את סך המועמדים שהופקו במהלך העבודה על משימות.   
   עבור כל מחרוזת מועמדת:
   1. המשימה שבמסגרתה היא הופקה
   2. הקוד שהיא הופקה בה

בגמר התחרות וההכרזה על המנצח יש לעצור את העבודה (גם אם יש עוד משימות לביצוע). ה Agent צריך לשאול כל פרק זמן את השרת (ואת צוות ה Allies שלו) לגבי סטטוס התחרות ולהגיב בהתאם.  
כשתחרות נגמרת מוצגת הודעה מתאימה באפליקציית ה Allies. רק כשהמשתמש מאשר את סיום התחרות (באפליקציית ה Allies) יימחק המידע המוצג ב Agent'ים המתאימים עד שהם יקבלו מידע על התחרות החדשה שהם שותפים לה. (זה נועד לאפשר למשתמש להביט במצבם של ה Agent'ים השונים טרם סגירתם).

## חלוקה למודולים

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 3 מודולים נוספים (חדשים) שמהם ייבנו אפליקציות ה client השונות (על שלל Jar'הן).

## 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 בפיתוח אפליקציית ה UBoat. היא זהה במהותה ליכולות שכבר קיימות במסגרת תרגיל 2 (טעינת קובץ ; הצגת מידע ; בחירת קוד ; הצפנת מחרוזת ; הצגת מועמדים לפענוח וכו')

המשיכו בפיתוח אפליקציית ה Allies. גם היא מכילה מרכיבים רבים שמימשתם זה לא מכבר במסגרת תרגיל 2 (הצגת מידעים על הסוכנים השותפים וכו'). ממשו את היכולת לראות את כלל התחרויות במע' והרשמות להן. במסך התחרות עצמה ממשו את היכולת לראות את המידעים השונים.

עברו לפיתוח אפליקציית ה Agent. גם כאן רוב הרכיבים כבר קיימים מתרגיל 2 (ורק עוברים מקום לאפליקציה הנדרשת).  
עכשיו שהכל עובד ומוכן, השלימו את התפרים הנדרשים כדי להשלים את מהלך התחרות (זיהוי שכולם READY ; זיהוי המנצח והפצתו לכולם ; עצירת הצוותים שעוד עובדים כשיש כבר מנצח ; איפוס המע' לתחילתה (אצל הסוכנים) וכו')

לאורך פיתוח 3 סוגי הקליינטים יש לשאוף להשתמש כמה שאפשר ברכיבים הקיימים בתרגיל 2, גם כדי להקל משמעותית על עומס הפיתוח ובעיקר כדי להתנסות (ולהוכיח לעצמכם !) איך אפשר לפתח רכיב פעם אחת ולהשתמש בו בהקשרים נרחבים אחרים.

לא מן הנמנע כי 3 סוגי הקליינטים יצטרכו להשתמש בקוד משותף (למשל כל תפעול וגישה ל http client). מאוד הגיוני לייצר מודול נוסף (== jar) שיחזיק את סך הקוד המשותף ל 3 האפלקיציות (וייכלל כמובן ב 3 האפליקציות).

ממליץ מאוד לאורך כל העבודה להקפיד לעבוד ולבחון את עצמכם דרך postman כאמצעי וידוא כי הסרבלטים עובדים כראוי ללא תלות בקליינט כזה או אחר. 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 ? |
| 1 | הגדלת ראש מגניבה ! | Chat:  אפשרו למשתמשים לקיים chat בינהם תו"כ התחברותם למע'.  ה chat צריך להתנהל כולו אל מול השרת בצורה שבה כולם רואים את מה שכולם כותבים. (מאוווווווווווד דומה לדוגמא הסופית)  הצ'ט מתבצע בין המשתמשים הרשומים והמחוברים למערכת בלבד. וצריך להיות פעיל הן בעמוד 2 והן בעמוד 3. | היש מדהים מזה ??? | 5 נקודות  (ועד ל 100) |
| 2 | די מתבקש | LOGOUT  אפשרו לשחקנים השונים במע' להחליט שהם נוטשים אותה.   * Agent יכול לנטוש צוות Allies כל עוד הוא לא במהלך תחרות פעילה. הציגו כפתור LOGOUT ב Agent. שקפו את המידע בצד ה Allies השונים (לגבי סוכן שנפל ואינו קיים יותר) * Allies יכול לנטוש. הציגו כפתור LOGOUT ואפשרו ללחוץ עליו גם במהלך תחרות פעילה. במידה וזה קורה כל הסוכנים שלו מקבלים את המידע וסוגרים את עצמם. שאר הצוותים במע' (וה UBoat) מקבלים את הידע הזה בדמות זה שיש פחות צוותים בתחרות * Uboat יכול לנטוש. אם זה קורה במהלך תחרות פעילה, היא נעצרת. כל הצוותים מקבלים על כך חיווי ומועברים למסך ריכוז המידע על התחרויות. ה UBoat עצמו נסגר ופרטי התחרות שהוא העלה למע' נמחקים. | עוד קצת משחקי Servlet'ים.. מה קרה ?? (אבל פיצ'ר מדהים לספר עליו במהלך ראיונות..) | 10 נקודות **(מעל ל 100)** |
| 3 | בואו נקל על המשתמש | העלאת סוכן בלחיצת כפתור  אפשרו ל Allies להקים אפליקציה Agent בלחיצת כפתור במסך השני (ריכוז המידע על תחרויות), במקום להכריח את המשתמש לעשות זאת ידנית.  בשביל בונוס זה יש לאפשר בהעלאת אפליקציית ה Agent לקבל מבחוץ את שם צוות ה Allies שהיא שייכת לו.  ממליץ לקרוא על ProcessBuilder לבד ולהשתמש בו לטובת הפעלת תהליך בצורה תכנותית... | אחד הדברים המגניבים ! | 10 נקודות (ועד ל 100) |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. WAR אחד בדיוק שיושם 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. 3 ספריות עבור הקליינטים השונים: Uboat, Allies, Agent בכל ספרייה יישבו כל ה jar'ים הנדרשים כדי לתפעל את האפליקציה הרלבנטית. יש לספק לכל אפליקציה קובץ batch שמפעיל אותה (כמו שהיה בתרגיל 1 ו 2).   
   שימו לב כי האפליקציה שלכם צריכה להכיר אוטומטית את השרת ואת ה context path הרלבנטי לה (אפשר כמובן להניח את localhost:8080 כדומיין של השרת).
3. קבצי קוד המקור של הפרויקט שלכם (גם צד השרת וגם אפליקציות java fx של ה clients).
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 זאת **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.

## שאלות ותשובות

|  |  |  |  |
| --- | --- | --- | --- |
| # | תאריך | שאלה | תשובה |
| 1 |  |  |  |

נספח א' – פרטים לגבי מימוש מכונה

המלצות וחידודים הנוגעים למימוש מכונת האניגמה:

* + 1. הגדירו למכונה שלכם "מצב debug" שבו היא תפלוט מידע רב ככל האפשר המתעד עבור עיבוד כל תו את כל הדרך שהוא עבר, כולל פסיעת הרוטור(ים), היכן הוא נכנס מהיכן הוא יצא, מה היה סטטוסו במשקף וכל הדרך חזרה וכו'. רק בדרך זו תוכלו לעקוב מבחוץ על הנעשה בקרביים של המכונה ולהבין אם היא עובדת כהלכה.
    2. תוכלו לבדוק את פעילותה הנכונה של המכונה באמצעות תיאור מפורט של מהלך הריצה שלה על קובץ קטן (sanity-small), המכיל אלף-בית של 6 אותיות בדיוק, עם 2 רוטורים. בהמשך נספח זה תוכלו למצוא תיאור מפורט של עיבוד 3 תווים במכונה מדגם זה שיעזרו לכם לפענח ולהבין האם אתם עבודים כראוי.

להלן טבלה של מספר עיבודי מחרוזות במכונה הנבנית מ sanity-small, עם קוד ראשוני של **<F|A><I><CC><2,1>**

המחרוזות הוכנסו משמאל לימין, תוך איפוס המכונה חזרה לראשית הקוד (CC) לפני עיבוד כל מחרוזת:

|  |  |  |
| --- | --- | --- |
| # | מחרוזת קלט | מחרוזת פלט |
| 1 | AABBCCDDEEFF | CEEFBDFCDAAB |
| 2 | FEDCBADDEF | BACACFFCDD |
| 3 | FEDCBAABCDEF | BACACFEEAFBB |
| 4 | AFBFCFDFEFFF | CDEBBAFADDAB |
| 5 | AAAEEEBBBDDDCCCFFF | CEFDABCEFFCEDABABD |

* + 1. תוכלו לבדוק עצמכם גם ע"י בדיקה שאתם באמת יכולים להצפין ולפענח מחרוזת תווים הלוך וחזור באמצעות המכונה שלכם לחלוטין – כלומר התהליך הפיך לחלוטין.
    2. תוכלו לבדוק את המכונה עם נתוני מכונה גדולה יותר המוגדרת באמצעות ה paper enigma המובאת בנספח ב'.

להלן טבלה של מספר עיבודי מחרוזות במכונה הנבנית מ sanity-paper-enigma, עם קוד ראשוני של **<I><ODX><1,2,3>**

המחרוזות הוכנסו משמאל לימין, תוך איפוס המכונה חזרה לראשית הקוד (ODX) לפני עיבוד כל מחרוזת.

שימו לב: ב papernigma המקורי הצבת מיקומי הרוטורים נקבעת על פי העמודה השמאלית, בעוד שאצלנו במע' (ובדוגמאות מטה) ההצבה נקבעת דווקא על פי העמודה "הימנית" (ומבחינת ההגדרות בקובץ ה XML – על פי ה right).

|  |  |  |
| --- | --- | --- |
| # | מחרוזת קלט | מחרוזת פלט |
| 1 | THERAINISDROPPING | APZTICDXRVMWQHBHU |
| 2 | HELLOWORLD | DLTBBQVPQV |
| 3 | ENIGMAMACHINEROCKS | QMJIDORMMYQBJDVSBR |
| 4 | WOWCANTBELIEVEITACTUALLYWORKS | CVRDIZWDAWQKUKBVHJILPKRNDXWIY |
| 5 | JAVARULES | MRUHFRZZR |

פירוט המכונה כפי שהיא נבנית מקובץ sanity-small ובאמצעות הגדרת קוד ראשוני של **<B|C><I><CC><2,1>**

|  |  |  |
| --- | --- | --- |
| # | תיאור המצב | דיאגרמה |
| 1 | מצב התחלתי בו נבחרו לשימוש רוטורים 1 (ראשון מימין) ו 2 (שני משמאל). כמו כן נבחר משקף מספר I. |  |
| 2 | לאחר קביעת קוד סודי הממקם את שני הרוטורים במיקום 3 (או ב 'C') |  |
| 3 | לפני כל הקלדת תו, המע' מבצעת פסיעה אחת קדימה החל מהרוטור הימני (1). במקרה הנ"ל הזיז של רוטור 1 הגיע לחלונית ההצצה ולכן הוא גורם גם לקידום רוטור מס' 2 צעד אחד קדימה |  |
| 4 | **הצפנת האות C 🡨 A:**  הקלדת האות C פוגשת קודם כל את לוח התקעים. יש תקע המחליף בין B ו C ולכן הלכה למעשה הוגשה לכניסת הרוטורים האות B.   האות B קודדה לאות A. תרשים החצים הבא מציג את כיוון הזרם בחיווטי הרוטורים השונים. חץ כחול הוא כיוון ההלוך וחץ אדום הוא כיוון החזור.  בחזור האות A ללוח התקעים היא יוצאת משם AS IS מאחר ואין תקע מחליף לאות זו. |  |
| 5 | לקראת הקלדת אות נוספת, המע' מבצעת פסיעה אחת קדימה. הפעם רק רוטור 1 פוסע |  |
| 6 | **הצפנת האות D 🡨 A:**  האות D אינה מוחלפת בלוח התקעים. הקדלת האות D קודדה לאות A. האות A אינה מוחלפת בלוח התקעים.  שימו לב כי בעקבות תזוזת הרוטורים ייתכן כי 2 תווים שונים יקודדו לאותו תו |  |
| 7 | לקראת הקלדת אות נוספת, המע' מבצעת פסיעה אחת קדימה. הפעם רק רוטור 1 פוסע |  |
| 8 | **הצפנת האות A 🡨 F:**  האות A אינה מוחלפת בלוח התקעים.  הקלדת האות A קודדה לאות F  האות F אינה מוחלפת בלוח התקעים. |  |

A picture containing diagram

Description automatically generatedנספח ב' – מכונת אניגמה לגזירה ותרגול על נייר

נספח ג' – תיאור מבנה המערכת באמצעות קובץ XML

תיאור המכונה והתנהלותה בתרגילים השונים נתון באמצעות קובץ XML.

במהלך הבדיקה (של שלל התרגילים), תבדק 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 case **in**sensitive הכוונה היא שאין חשיבות ל case של האותיות באנגלית. במקרה זה הערך milk זהה לערך 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)

Text, calendar

Description automatically generatedמבנה המכונה מאוגד תחת האלמנט Machine ומכיל בתוכו מספר אלמנטים/מאפיינים נוספים:

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | CTE-Machine | אלמנט זה מכיל את כל פרטי ההגדרה של המכונה |
| 2 | Attribute | rotors-count | מכיל את מספר הרוטורים שהמכונה המסויימת הנ"ל משתמשת בהם. שימו לב זהו מספר הרוטורים שחייבים להיות בשימוש בכל רגע. המכונה עצמה יכולה להגיע עם מספר רוטורים גדול מזה. |
| 3 | Element | ABC | מתאר את האב' של המכונה. אוסף אותיות אפשריות לשימוש במהלך העבודה עם מכונה זו. Case insensitive.  שימו לב כי ישנם מספר תווים "לא חוקיים" ב XML ולכל אחד מהם יש חלופות.  בפרט: '&' = &amp; ו '>' = &lt;  יש לבצע trim למחרוזת ה ABC כדי לנקותה מתווים מיותרים בראשיתה ובסופה. |
| 4 | Element | CTE-Rotors | מכיל את הגדרות הרוטורים השונים |
| 5 | Element | CTE-Rotor | מתאר מבנה של רוטור אחד במע'. |
| 6 | Attribute | id | מספר הזהות של הרוטור. צריך להיות יחיד בין כל הרוטורים השונים |
| 7 | Attribute | notch | מיקומו של זיז הדחיפה על גבי הרוטור המסויים. המספר ניתן בבסיס 1 |
| 8 | Element | CTE-Positioning | מגדיר מיפוי אחד בתוך רוטור.  כל רוטור יורכב מסדרה של מיפויים, כגודל ה ABC של המכונה  סדרת המיפויים מתארת את המבנה של הרוטור כפי שהוא מוגדר ב Paper enigma |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Attribute | left | right | מתארים את מיקומי האותיות בסדרת המיפויים.  כל מיפוי אינו נחשב בפני עצמו אלא אך ורק בשבתו כחלק מסדרת המיפויים המרכיבה את הרוטור.  האותיות יכולות להופיע ב case אחר יחסית לזה המתואר ב ABC. |
| 10 | Element | CTE-Reflectors | מכיל את הגדרות המשקפים השונים בשימוש המכונה |
| 11 | Element | CTE-Reflector | מתאר מבנה של משקף בודד |
| 12 | Attribute | Id | מספר מזהה של המשקף הנ"ל. המספר מוגבל ל 5 תווים בספירה רומית: I, II, III, IV, V |
| 13 | Element | CTE-Reflect | שיקוף (מיפוי) בודד של כניסה אחת ליציאה אחת. |
| 14 | Attribute | input\output | מתארים את השיקוף שיש לבצע בין כניסה מספרית מסויימת (input) ליציאה מספרית מסויימת (output). המספרים ניתנים בבסיס 1. שיקוף אינו יכול להיות מכניסה מסויימת לעצמה. כמות המיפויים צפויה להיות חצי מאורכו של ה ABC |

## סכמה תרגיל 2

בתרגיל 2 מצטרף אלמנט נוסף לקובץ – Decipher.

אלמנט זה מתאר את תהליך הפענוח האוטומטי של המכונה:

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | CTE-Decipher | אלמנט זה מכיל את כל המידע בנוגע לתהליך הפענוח האוטומטי. הוא מתאר את "מנהל הפענוח" – DM |
| 2 | Attribute | agents | מספרם המקסימלי של הסוכנים שיכולים להיות מועסקים ברגע נתון ע"י DM זה |
| 3 | Element | CTE-Dictionary | אלמנט המתאר מידע על מילון המילים האפשריות המשוייך למכונה זו |
| 4 | Attribute | exclude-chars | רשימה של תווים מהאלף-בית של המכונה שלא נכנסים למילון. במידה והם נמצאים במי ממילות המילון יש להוריד תווים אלה מן המילה ורק אז להכיס אותה. במידה והם יתגלו כחלק מהפענוח – יש להוריד אותם מהמחרוזת המעובדת ורק אז להשוות אותה מול המילון |
| 5 | Element | Words | מכיל את כל המילים החוקיות האפשריות להקלדה במכונה זה. המילים מופרדות בתו הרווח ' '. ייתכנו מילים שחוזרות על עצמן – אין לכך משמעות |

Graphical user interface, text

Description automatically generated

## סכמה תרגיל 3

בתרגיל 3 מצטרף אלמנט נוסף המתאר שדה קרב במערכה: Battlefield.



|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | CTE-Battlefield | אלמנט זה מכיל מידע שדה קרב שבו תיערך המערכה לפיצוח אוטומטי ע"י מספר צוותים של הודעות אניגמה |
| 2 | Attribute | battle-name | שם המערכה. אמור להיות ייחודי לכל Battlefield |
| 3 | Attribute | Allies | כמות צוותי הפיתוח המשתתפים במשחק זה |
| 4 | Attribute | level | רמת הקושי של המשחק. יכול לקבל Easy, Medium, Hard, Insane |

בנוסף אלמנט ה CTE-Decipher לא יכיל יותר attribute בשם agents.

נספח ד' – תרשים סכמות XML

## סכמה תרגיל 1

Diagram

Description automatically generated

## סכמה תרגיל 2

Diagram

Description automatically generated

## סכמה תרגיל 3

Diagram

Description automatically generated