



פיתוח תוכנה מקצה לקצה

חורף 2025 - Enigma

מס' גרסה: 3

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

מרצה: **אביעד כהן** aviadco@mta.ac.il

בודק: **איתי כהן** itaych@mta.ac.il

תוכן העניינים

4	דרישות הקורס
4	כללי
5	איך להגיש תרגילים באיחור, ולהישאר בחיים
6	הנחיות כלליות לכתובת התרגיל
8	Building the Enigma Machine
8	מטרת התרגיל(ים) בקורס
8	מכונת האניגמה
11	מגבלות ומאפיינים של מכונת האניגמה בתרגיל זה:
12	הנחיות ספציפיות למימוש תרגיל "בניית מכונת האניגמה"
13	תרגיל 1 – מימוש מכונת האניגמה כאפליקציית Console (30%) - הגשה: 14.12.25
13	פרטים יבשים
13	דרישות
17	חלוקה למודולים
17	איך מתחילים? (המלצה...)
18	בונסים
18	סבבה, סיימתי. מה ואיך להגיש?
19	תרגיל 2 – מימוש מכונת האניגמה בתצורת Maven (30%) - הגשה: 13.1.25
19	פרטים יבשים
19	דרישות
21	איך מתחילים?
21	סבבה, סיימתי. מה ואיך להגיש?
22	תרגיל 3 – מימוש מכונת האניגמה כאפליקציית spring boot (35%) - הגשה: 26.2.26
22	פרטים יבשים
22	דרישות
24	חלוקה למודולים
24	איך מתחילים?
25	סבבה, סיימתי. מה ואיך להגיש?
26	תרגיל 4 – ממשק וובי לביצוע שליפת מידע דינמית ממע' אניגמה (תרגיל רשות)
26	פרטים יבשים
26	דרישות
29	נספח א' – פרטים לגבי מימוש מכונה
30	נספח ב' – פירוט תנועת המכונה (כולל לוח תקעים)
33	נספח ג' – מכונת אניגמה לגזירה ותרגול על נייר
34	נספח ד' – עבודה עם DB (תרגיל 3)
35	נספח ה' – תיאור מבנה המערכת באמצעות קובץ XML
36	סכמה תרגיל 2
36	סכמה תרגיל 3

37	נספח ו' – תרשימים סכמות XML
37	סכמה תרגיל 1
38	סכמה תרגיל 2
39	סכמה תרגיל 3

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 חובה להגיש ביחידים.
3. בעבודה משותפת על תרגיל יש להקפיד על מעורבות אקטיבית של כלל המגישים בכל חלקי התרגיל.
4. במידה והוגדר בנוס לתרגיל מסוים, ציון הבונוס יתווסף לציון התרגיל בלבד (ולא לציון הסופי של הקורס כולו).
5. התרגילים יוגשו דרך מערכת Mama. מוגדר רכיב 'מטלה' נפרד לכל תרגיל.
6. 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.
7. ניתן להחליף את השותפ/ה בכל תרגיל, ללא צורך באישור או הודעה למרצה. הניקוד על כל תרגיל נזקף לזכות הסטודנטים שבצעו אותו בלבד.
8. יש להעלות את התרגיל רק עבור אחד מבני הזוג ולהוסיף את שם בת/בן הזוג ומספר תעודת הזהות שלה/ו בקובץ ה readme המצורף (פרטים בהמשך).

1. ניתן להגיש תרגיל עד חמישה ימים איחור, כאשר עבור כל 24 שעות איחור – תורד נקודה אחת מציון התרגיל; תרגיל שיוגש באיחור של יותר מחמישה ימים (ללא סיבה מוצדקת) – **פשוט לא יבדק**.
בכל מקרה עיקבו אחר ההוראות המדוייקות שיינתנו בלוח ההודעות של הקורס. בכל מקרה של כפילות – הן הקובעות !
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 לבדוקו 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.
4. עומס בלימודים, בעבודה, בחיים, בגלל הילדים או ההורים, שכנים וחברים (או בכל תחום אחר) **אינו נחשב** כסיבה לגיטימית לבקשת הארכה.
5. במקרה של בקשה להארכה (מכל סיבה שהיא, לרבות מילואים ומחלה) יש לפנות למרצה מראש על מנת לקבל אישור. הפנייה תתבצע במייל.
6. 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.
7. ניתן לערער על ציון של תרגיל על פי הוראות הבודק בעת פרסום הציונים במע' המאמא.
כדי לערער יש לשלוח מייל לבודק בצירוף כל הסיבות והטענות שלכם.

- במהלך הקורס יוצגו דוגמאות והסברים מבוססים על כתיבה בסביבת הפיתוח IntelliJ IDEA – (IDE). 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 rar/zip (לא 7Z!) הקובץ יכיל:
 1. כל הקבצים הרלבנטיים להפעלת התרגיל (war/jar – 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 שבעיניכם ייתכנו בעיות/שאלות/תהיות וכיוצא בזה.

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

קובץ ה readme צריך לפתוח בעמוד השער המדויק כפי שניתן לכם במסגרת חומרי התרגיל. אין לחרוג משער זה.
- 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 נכונה:
 - 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 הרלבנטיים) 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 ההגייונית 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'אוה גרסה 21. הקפידו להוריד, לעבוד, לקמפל ולהריץ עם הגרסה המתאימה בלבד.
- ווידאו הגשת התרגיל טרם הגשתו:
 - יש לוודא כי ההגשה שלכם רצה היטב 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 linux/mac – זכותכם – אבל גם חובתכם לוודא כי אתם רצים היטב 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 0 Level על שטות שיכולתם לעלות עליה בשנייה עוד בשלב הפיתוח.
 - כאמור, הגשה חוזרת בשל תקלות 0 level תתחיל מראש מציין 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 לציין 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 ושאלו "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 לציין כזה).

מטרת התרגיל(ים) בקורס

חלק א' – מימוש מכונת האניגמה בקוד. הפעלתה ותפעולה באמצעות ממשק console. הכרות והתמקצעות עם אבני הבניין הבסיסיות של הטכנולוגיה.

חלק ב' – הכללת והרחבת יכולות המכונה, והמרת המימוש של חלק א' למבנה עבודה מתאים עם maven.

חלק ג' – הטמעת וחשיפת אניגמה במסגרת שרת תוך עבודה עם postman client כנגדו.

מכונת האניגמה

במהלך מלחמת העולם השנייה, השתמשו הכוחות הגרמנים [במכונת האניגמה](#) 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 ביניהן 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 A.T. !) 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 נרחיב את מבנה המכונה למכונה אוניברסלית שאיננה כבולה למגבלות המכונה הפיזית.

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 שהפעם המחרוזת המוצפנת מהווה קלט (למקלדת הראשית) והמחרוזת המופענחת 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).

מאפיין 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 לפענח חזרה. (במציאות, אכן תפסו הבריטים והפולנים מספר מכונות אניגמה – אולם לא היה די בכך כדי לפענח את ההודעות והיו צריכים לעבור עוד כבדת 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 כניסות מחוברות ל X יציאות. הכניסות והיציאות מסומנים כאותיות ה ABC – ובמכונה הקלאסית יש 26 כאלה (אצלנו, בהמשך, יהיו הרבה יותר). גלגל ההמרה למעשה ממיר כניסה X ליציאה Y: לדוגמה הכניסה המזוהה עם האות A יכולה להיות מחוברת ליציאה 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 (בדרך"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 יציאה 'Z'.



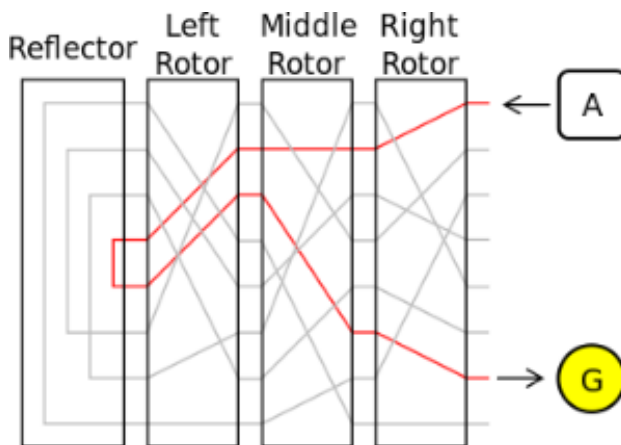
את הרוטורים השונים ניתן לסובב במקומם ולקבוע מיקום ראשוני שונה בכל פעם. כניסות הזרם החשמלי הממופות למקשי המקלדת נותרות קבועות. כלומר אם במצב רוטור מסוים, הקלדה על אות E ששוגרה בכניסה 5 אל הרוטור הימני ולכן נפלה על מיפוי m מסוים, לאחר סיבוב הרוטור לפוזיציה אחרת, אותה הקלדה על אות E, עדיין מגיעה לכניסה 5 אל הרוטור הימני – רק

שהפעם היא תיפול על מיפוי אחר (n) וכו'. מכונת האניגמה הכילה לכל רוטור מעין "חלופית" קטנה שאפשרה לראות מהו מיקומו ולסובב את הרוטור למיקום מסוים ע"י הזזתו עד שמתקבלת האות המבוקשת בחלופית (הפוזיציה הנדרשת).

היות וכל רוטור מחווט אחרת (כלומר מיפויים שונים בין "האותיות", כלומר בין הכניסות ליציאות), קל להבין כי לקביעת סדר הרוטורים (מי הימני, מי באמצע, ומי בשמאלי), כמו גם למיקום הראשוני של כל רוטור (אילו אותיות ניבטות מהחלופית) 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.

המכונה מורכבת מרכיב נוסף הנקרא רפלקטור ("משקף"). הרפלקטור ניצב אחרי סדרה הגלגלים (בצד השמאלי של המכונה) 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](#) 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 (הבסיסית) מגיעה עם מספר רוטורים אפשריים לשימוש, ממספרים בספרות 1..n יש לבחור 3 רוטורים לשימוש מהם, ולקבוע את סדר הרוטורים (ימני;אמצעי;שמאלי). 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 בכתב רומי).

אולי זה נשמע מסובך, אבל בעולם הקריפטוגרפיה – אין כאן אלא צופן החלפה 'פשוט' (כל אות מוחלפת באות אחרת), ואת זה קל מאוד לפענח גם אם לא יודעים את קוד המקור (והיה גם קל מאוד במהלך ימי מלחמת WWII).

כדי להקשות על פענוח הצופן עוד יותר, מכונת האניגמה השתמשה בטריק פשוט (אך זדוני): בכל הקשת את 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. (חישוב על מבנה של ספידומטר). למעשה כל הקלדת אות קדימה 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...).

כל הגלגלים מגיעים עם הגדרת הזיז שלהם, ולאורך חיי המע' הוא מקובע במקומו הרלוונטי בכל רוטור ורוטור.

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? 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 תקעים המסומנים באותיות ובכך יוצר צמדי אותיות להחלפה (נגיד D I K או למשל O I Z) וזה אומר שכל פעם שאותיות אלה מגיעות אל לוח התקעים – הן מתחלפות.

אות הגיעה אל לוח התקעים פעמיים:

ב"הלך", 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 (הבנתם? בדיחה!) 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 בעיניה ויוצאת ממנו כשם שבאה. (נגיד בדוגמא שלנו האות A נשארת A)

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	$\langle A D,K E \rangle \langle II \rangle \langle A(1),I(5),C(15) \rangle \langle 2,3,1 \rangle$	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 5 פסיעות אחת מחלונית ההצצה). השתמשי במשקף מספר 2 (II). השתמשי ב 2 תקעים: בין A ו D ובין K ו E
2	$\langle I \rangle \langle D(5),O(2),W(0) \rangle \langle 5,1,4 \rangle$	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. לאורך התרגיל 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 והם מגדירים את "המיפויים" של כל רוטור ורוטור הנקבעים ע"י מציאת האות הזוהה 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 מסוימת – 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'](#) ו**נספח XML** כדי לקבל את מלוא הפרטים בהקשר זה.

כמה אפשרויות מתקיימות לקוד במכונת האניגמה ?

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](#) 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 יכול להכיל גם ספרות ותווים מיוחדים ויהיה גדול הרבה יותר מ 26, כך שמספר האפשרויות מגיע בקלות לעשרות מליונים אם לא ליותר.

בקיצור, פענוח ההצפנה הוא:

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 (&^%\$#@! וכ'), תו הרווח ועוד. מובטח כי **לא יכללו** התווים הבאים:
טאב (\t), enter (\r\n), esc (027)
2. גודל האב' של המכונה חייב להיות זוגי (בעיקר בשל מגבלות המשקף ולוח התקעים)
3. המכונה תוגדר לכל היותר עם 5 משקפים לבחירה (מהם צריכים לבחור **בדיוק** אחד)
4. המכונה תוגדר לכל היותר עם 99 רוטורים **בשימוש** (שימו לב: יכולים להיות מוגדרים יותר מ 99 רוטורים, אולם לכל היותר 99 יהיו בשימוש)
5. משקף **לעולם לא** יוכל לבצע מיפוי בין אות לעצמה.
6. **בהחלט ייתכנו** מיפויים ברוטורים בין אות לעצמה.
7. מיפוי כל רוטור הוא מיפוי דו כיווני ("הלך": right to left ו"חזר": left to right).
8. 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: מקלדת(ו)ת -> רוטור 1 -> רוטור 2 -> ... -> רוטור n -> משקף
כך גם במתן והגדרת קוד המכונה (ראו פרטים בהמשך).
9. נוכל להשתמש במקסימום התקעים האפשרי, או בכלל לא. ככל שנרצה.

הנחיות ספציפיות למימוש תרגיל "בניית 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 XMLים 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.
2. 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.
3. 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.
4. 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.
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. המע' כולה תתואר בשפה האנגלית בלבד, עם ממשק משתמש המתנהל משמאל לימין (במקומות הרלבנטיים)
7. המע' כולה תיכתב ותורץ בסביבת העבודה של ג'אוה 21.

פרטים יבשים

תאריך הגשה: **14.12.25**

ציון אפשרי מקסימלי: 105

צפי תחילת עבודה: **17.11.25**

צפי זמן לביצוע: **4 שבועות**

משקל התרגיל: **30%**

מטרת התרגיל העיקרית

- הקמת מנוע המע' הבסיסי
- יצירת ממשק console לתפעול המערכת

דרישות

- בתרגיל זה תקימו את מכונת האניגמה ותעמלו על יכולת התפעול התקינה שלה. 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 הקלט ולהחזיר פלט רלבנטי.
- פרטיה הטכנים של המערכת יינתנו מקובץ XML (כמפורט [בנספח ג'](#)). מגבלות טכניות למכונה:
 - עבודה עם 3 רוטורים בשימוש ברגע נתון (אך יכולים להיות מוגדרים יותר מ-3)
 - עבודה ללא לוח תקעים
- יש לוודא תקינות קלט כחלק מכל אינטרקציה עם המשתמש, ובכל מקום שבו זה רלוונטי: אם אתם מצפים לקבל מספר – לא לקרוס כי הכניסו לכם בטעות (או בכוונה) טקסט וכו'. בכל מקרה של תקלה יש להיות מאוד ברורים במסר שמעבירים חזרה למשתמש: מה קרה? מה הייתה מהות התקלה? היכן שזה רלבנטי, איך לתקנה וכו'. חישבו איך להיות ידידותיים למשתמש ולעולם אל תניחו כי מי שמשתמש באפליקציה שלכם הוא מתכנת בעצמו או מישהו שמגיע מהתחום ו"מכיר" איך דברים עובדים לבד. (זה הזמן לחשוב על...)
- 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.

5. עליכם לכתוב ממשק משתמש בתצורת console.

ממשק המשתמש יכול 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 תגרוור 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 הרלבנטיים). כלומר, אין להניח שהמשתמש הולך להקליד לכם מלל חופשי של תיאור האפשרות! זיכרו כי הספרור כלפי חוץ (המשתמש) תמיד מתחיל מ 1 (גם אם פנימית אתם סופרים מ 0)

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 יכול רק אותיות באנגלית (לא ג'יריש של אותיות בעברית וכו')

הקובץ יהיה קובץ XML שפרטיו וחוקיו המפורטים מובאים בנספח ג' 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:

- הקובץ קיים, והוא מסוג XML (די לבדוק לשם כך כי הוא נגמר בסימנת xml).
- גודל ה ABC הוא זוגי
- מוגדרים לפחות 3 גלגלים (רוטורים) במסגרת הקובץ.
- לכל רוטור יש id ייחודי. כל ה id'ים של הרוטורים השונים צריכים בסוף להוות ספרור רץ המתחיל מ 1, וללא חורים. הרוטורים לא חייבים להופיע על פי הסדר.
- אין מיפויים כפולים בתוך רוטור (כלומר אותה אות מופיעה פעמיים בעמודת ה left או ה right)
- זיז הפסיעה (notch) בכל רוטור מוגדר בטווח גודל הרוטור (פונקציה של גודל ה ABC)
- 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.
- אין מיפוי בין אותה הכניסה לעצמה ביציאה 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 ומאפשרת תמיד. אפשר לבחור בה בכל רגע נתון במע'.

2. הצגת מפרט המכונה

פקודה זו מציגה את מפרט המכונה המוגדרת בהתאם לקובץ התקין האחרון שנטען בהצלחה. יש להציג את הפרטים הבאים:

1. כמות רוטורים במע' (סה"כ הכמות האפשרית לבחירה)
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):

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

דוגמא (ראו הסבר צבעוני בהמשך):

(הצבעים נועדו להבהרה בלבד. אין צורך להשתמש בצבעים במסגרת תרגיל console !):

<III><A(2),O(5),I(20)><45,27,94>
<V><I(4),=(10),3(52),K(3)><198,346,123,959>

זיכרו כי מיקומי הרוטורים נקבעו מימין לשמאל, על אף שמדפיסים את הנתונים הנ"ל משמאל לימין... (בדוגמא הראשונה, הרוטור הימני ביותר הוא 94 והוא מכון לתו '!'. הרוטור השמאלי ביותר הוא 45 והוא מכון לתו 'A')

הסבר על הרכיבים השונים:

1. מספרי גלגלים שנבחרו + סדר ביניהם – 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. כל אחד מהמקטעים השונים (1-3) עטוף בתוך סוגריים משלושיים <>

ממליץ ללמוד לעבוד עם [StringBuilder](#) כאמצעי לבניית המחרוזת (המתוסבכת משהו) צעד אחרי צעד בצורה יעילה (ובעיקר חביבה)

דגשים:

- פקודה זו מאפשרת רק לאחר שטעון קובץ תקין במע'

3. בחירת תצורת קוד נוכחית (בצורה ידנית)

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

תצורה זו תורכב מ 3 חלקים, כפי שמוגדר בפקודה 2 סעיף 5

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

א. הרוטורים ממספרים בספרות עשרוניות. יתקבלו כרשימה שבה הם מופרדים באמצעות התו פסיק (,).

דוגמא: 23,542,231,545

על אף שהקלט מוכנס משמאל לימין (מתחילים מ 23) הכוונה היא כי הרוטור הימני ביותר הוא 545 וכו'

ב. מיקומי הגלגלים הראשוניים הם חלק מה-אב' של המע'. יש להכניסם כרצף תווים צמודים.

דוגמא: 4D8A

בקלט הנ"ל, על אף שהוכנס משמאל לימין (4 הוכנס ראשון) – האות A היא המיקום של הרוטור הימני ביותר (545)

ג. 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'.

יש לבצע בדיקת קלט למשתמש ולהתמודד עם מצבים שבהם הוא מקליד אותיות היכן שמצופים מספרים או להיפך, או כל

ניסיון אחר שלו להזין קלט שאינו תקין ואינו עומד בציפיתכם. 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.

דגשים:

- פקודה זו מאפשרת רק לאחר שטעון קובץ תקין במע'

4. בחירת תצורת קוד נוכחי 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.

ממליץ לקרוא על המחלקה [Random](#) אשר מאפשרת בדיוק את זה...

דגשים:

- פקודה זו מאפשרת רק לאחר שטעון קובץ תקין במע'

5. 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.

בסיום פקודה זו הרוטורים נשארים במקומם הנוכחי ואינם מוחזרים אוטומטית לקוד הראשוני שאותחל במכונה (פק' 3 או

4).

דגשים:

- פקודה זו מאפשרת רק לאחר שטעון קובץ תקין במע'
- אין לאפשר הפעלת פקודה זו באם טרם בוצעה פקודה 3 או 4
- יש לוודא כי המשתמש לא מזין אותיות שאינן חלק מה-אב של המע'. (להודיע לו בצורה מסודרת כמובן אם זה קורה)

6. 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 אשר כולן מתחילות מאותה תצורה ראשונית – יש לאפסה לאחר כל הצפנה/פענוח. פקודה זו תאתחל את הרוטורים לקוד החוקי האחרון שנבחר ע"י פקודה 3 או 4 (הוא "הקוד המקורי")

דגשים:

- פקודה זו מאפשרת רק לאחר שטעון קובץ תקין במע'
- אין לאפשר הפעלת פקודה זו באם טרם בוצעה פקודה 3 או 4

7. הסטוריה וסטטיסטיקה

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

- א. 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)
- ב. 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:

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

כאשר # הוא מספר סידורי רץ המתחיל מ 1, והמחרוזות מוקפות בסוגריים משולשיים < >

דגשים:

- פקודה זו מאפשרת רק לאחר שטעון קובץ תקין במע'

8. 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 לזה 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 **אברהם** 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'.
2. 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 ביניהם.

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 במימוש פקודה 1 (על כלל היבטי הטעינה), ופקודה 2 שתאפשר לכם לראות כי קיבלתם את המידע כנדרש. עיברו לממש את פקודות תפעול המכונה עצמה (3,6,5,4) ורק בסוף התעסקו עם סוגיית ההסטוריה והסטטיסטיקה (פקודה 7)

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.println) 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 הסיימת) 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. jar 2 (לפחות) שהם כל הקוד שלכם, בצירוף קובץ אצווה (batch) שהפעלתו תריץ את התוכנית (כלומר תבצע <class name> java -jar וכו').
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. הקובץ צריך להיפתח "בשער" שקיבלתם במסגרת חומרי התרגיל.
3. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות ותפקידם.
4. יש לכלול בקובץ ה **readme** קישור ל Github שלכם המכיל את קוד הפרויקט.

בונוס שימושי אבל לא יתועד – לא ייבדק !

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

פרטים יבשים

תאריך הגשה: **13.1.25**

ציון אפשרי מקסימלי: **100**

צפי תחילת עבודה: **15.12.25**

צפי זמן לביצוע: 4 שבועות

משקל התרגיל: **30%**

מטרות התרגיל העיקרית

1. מידול והעברת המע' לתצורת maven
2. הוספת יכולת לעבוד עם יותר מ 3 רוטורים
3. הוספת תמיכה בלוח התקעים

דרישות

1. במסגרת תרגיל זה נרחיב את יכולת המכונה לעבוד עם מספר דינמי של רוטורים, (ולא רק 3 כפי שהיה בתרגיל הראשון). כמות הרוטורים בשימוש במכונה תופיע בדמות נתון חדש בקובץ ה XML (עיקבו אחר שינוי [בסכמה](#)). שימו לב כי שינוי קובץ ה XML גורר שינוי בסכמה ומכאן גם הצורך לג'נרט מחדש את המחלקות באמצעות הסכמה ו XJC. עליכם לבצע בקבצי הקלט את כל הבדיקות שביצעתם במסגרת תרגיל 1 ובנוסף יש לוודא כי כמות הרוטורים בשימוש אינה גדולה מכמות הרוטורים המוגדרת במכונה.

2. תמיכה בעבודה עם לוח התקעים:
עליכם לממש את התמיכה בעבודה עם לוח התקעים.
לתמיכה זו יש כמה השלכות על התנהלות המכונה:
בדרך הלוח הקלט מהמשתמש עובר ראשית אל לוח התקעים, פוטנציאלית מוחלף באם יש תקע מתאים ורק אז מגיע אל הרוטורים.
בדרך חזור הפלט מהגלגל הימני עובר שוב אל לוח התקעים, פוטנציאלית מוחלף באם יש תקע מתאים ורק אז מוחזר אל המשתמש כפלט הקידוד.
התמיכה בלוח התקעים באה לידי ביטוי בכמה נקודות לאורך המע':

1. בפקודה 1 (טעינת קובץ) יש לבצע בנוסף לכל הבדיקות גם וידוא ובדיקה כי גודל האב' הוא זוגי.

2. בפקודה 2 (הצגת מצב המע') יש להציג גם את התקעים שנעשה בהם שימוש (סעיף 2.4 ו 2.5).
תצורת הקוד תראה מעכשיו כך:

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

הנה דוגמא:

```
<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>
```

כל צמד תקעים יוגדר באמצעות זוג אותיות מופרדות בסימן ה pipe (|). אם יש יותר מתקע אחד הוא יופרד בפסיק.
(בדוגמא הראשונה יש תקע בין A ל Z ובין D ל E. אם A מגיע אל לוח התקעים הוא מוחלף ל Z ולהיפך)
זיכרו כי אין חובה להשתמש בתקעים במסגרת הקוד (כפי שנראה בדוגמא השנייה)

3. בפקודה 3 יש לדרוש מהמשתמש להכניס גם את הפלגים (לאחר שלב הכנסת המשקף).
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:

a. D | K

b. 4 | 9

c. | ! (שימו לב תו הרווח הוא תו חוקי באב' והוא חולק את הפלג שלו עם התו !)

4. פקודה 4 יכולה גם כן לבחור רנדומלית את כמות התקעים, את שילובי הצמדים וכו'.

5. בפקודה 5 יש כמובן להתייחס לתקעים במסגרת עיבוד הקלט במכונה. עיקבו אחר [נספח ב'](#) לתיאור מפורט זה

3. מעבר לארכיטקטורת פיתוח maven:

יש לעדכן ולשנות את המימוש כך שהוא יתאים לתצורת maven כפי שנלמד והוסבר בכיתה.
ה Group הראשי של המע' ייקרא [patmal.course.enigma](#) וכל המודולים יחלקו group זה.

עיקבו אחר הדרישות הבאות בהקשר זה:

1. חלוקה למודולים

על המע' לתמוך בחלוקה (לכל הפחות) למודולים הבאים:

a. מימוש מכונת האניגמה. שם הארטיפקט: [enigma-machine](#)

b. מימוש מנוע המע' על כלל יכולתיו. שם הארטיפקט: [enigma-engine](#)

c. מימוש יחידת טעינת הקבצים בצורה עצמאית. שם הארטיפקט: [enigma-loader](#)

d. מימוש ה UI. שם הארטיפקט: [enigma-console](#)

ייתכנו מודולים נוספים כרצונכם (נניח ל DTO ?) – אבל חובה להשתמש לפחות בארבעת המודולים הנ"ל תחת ה GAV שהוגדר כאן.

2. עליכם לעבוד עם aggregator pom שירכז את (לפחות) ארבעת המודולים הנ"ל וניתן יהיה לבנות את כל הפרויקט דרכו ובאמצעותו.

שם הארטיפקט: [enigma-aggregator](#)

3. עליכם להגדיר plugin שיבנה את כל המע' בתצורה של uber jar המכיל את כלל התלויות הנדרשות.
(אפשר ומומלץ בהחלט להשתמש ב assembly plugin שהוצג בכיתה, אם כי שווה לדעת כי יש עוד אפשרויות)
השם של ה jar הסופי הוא [enigma-machine-ex2.jar](#) (שימו לב אותיות קטנות בלבד !)

במסגרת בדיקת המע' תספיקו לבדוק קובץ batch פשוט שיבצע clone ל repository שלכם ויריץ mvn clean install בראשיתה, תוך ציפיה לקבל uber jar.
את ה jar הזה ניתן יהיה להפעיל באמצעות java -jar בצורה פשוטה.

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).

איך מתחילים ?

אני ממליץ לפתוח פרויקט חדש ב IntelliJ (ואם תרצו גם repository חדש ב github, אם כי לא חובה), כזה המבוסס על עבודה עם maven ולהגר את כל הקוד אליו.

ראשית הייתי מתחיל בלקחת ולהכשיר את האפליקציה הקיימת (תרגיל 1) במסגרת תצורת maven. התהליך יכלול העתקה פיזית של קבצים ועבודה (סיזיפית משהו) של עדכון ה package'ים וה Import'ים הרלוונטים כדי שיתאימו למבנה החדש.

אני ממליץ מאוד לנסות ולהיעזר ביכולות הצ'ט ובעיקר בשבתו כ agent כדי לבצע את המטלות הנ"ל וכמובן לוודא את הביצוע בעצמכם

רק לאחר שכל הפרויקט עובד לכם בתצורת maven התעסקו עם הרחבת היכולות הלוגיות החדשות (כמות הרוטורים ותמיכה בלוח התקעים) על השינויים השונים שנכרכים בהם.

לסיום הוסיפו את הפלגין אשר יודע לייצר uber jar ל pom הראשי וודאו כי ניתן לבנות את הפרויקט מראשיתו באמצעות mvn clean install על גבי ה pom הראשי בלבד והתוצר הסופי הוא uber jar.

סבבה, סיימתי. מה ואיך להגיש ?

עליכם להגיש zip אשר כולל 2 קבצים:

1. קובץ readme:

הקובץ צריך להיפתח "בשער" שקיבלתם במסגרת חומרי התרגיל.

2. קובץ ה run.bat אשר ישמש את הבודק בהרצת והרמת הפרויקט.

קובץ דוגמא הועלה לאתר המאמא. **ממליץ בחום** להוריד אותו ורק להחליף בו את ה git repository שלכם.

קובץ ה **readme** יכיל את הדברים הבאים:

1. הפניה ל git repository שלכם, אשר ממנו יבוצע ה clone. וודאו כי ה repository אינו private !

2. תיעוד והסבר כללי (וממצה) של המחלקות העיקריות ותפקידם.

3. הסבר על המערכת, בחירותיכם השונות במקומות שבהם היו לכם בחירה, או כל דבר נוסף העולה על דעתכם שחשוב שהבודק ידע.

פרטים יבשים

צפי תחילת עבודה: 1.2.26

צפי זמן לביצוע: 5 שבועות

משקל התרגיל: 35%

תאריך הגשה: 7.3.26

ציון אפשרי מקסימלי: 100

מטרות התרגיל העיקרית

1. מימוש המערכת כאפליקציית client-server (באמצעות spring boot)
2. טעינת מספר מכונות ועבודה איתן במקביל
3. הוספת שכבת persistence המבוססת על עבודה עם spring-data מול Postgres

דרישות

1. בתרגיל זה נטמיע את מכונת האניגמה בתוך שרת בתצורת spring boot.
דבר זה יחייב המרה של כלל רכיבי המע' לעבודה עם spring.
כלל המע' תשב תחת ה URL הבא: <http://localhost:8080/enigma>
ועליו יתווספו תתי שמות המשאבים כפי שמוגדרים בסעיפים הבאים.
תצורת המע' תהיה ב spring boot uber jar (כלומר ב java -jar).
2. צד הקליינט:
המע' תיבדק באמצעות Postman כקליינט אשר יגיש בקשות HTTP כדי להנות מיכולותיה.
הבקשות השונות יפעילו את המע' על כלל היבטיה השונים תוך העברת מידעים וקבלה חזרה של מידעים בתצורת Json.
עליכם להיצמד להגדרות הבקשות ולממשם במלואם.
הבודק הולך להשתמש בדיוק בבקשות אלה כדי לבדוק את האפליקציה שלכם.
במסגרת חומרי התרגיל קיבלתם Postman collection, בו תוכלו להשתמש כדי לבדוק את עצמכם.
כלל ה EndPoint'ים השונים מתוארים באמצעות סטנדרט שנקרא Open API. זהו סטנדרט המאפשר לתאר בקובץ טקסט (למען האמת yml <אימוג'י מקיא>) את הפרטים השונים של API של מע'. כלומר ליצר סוג של Spec עבור המע' שלכם.
יש הרבה כלים אי שם בחוץ, שאם תיתנו להם את קובץ ה Spec, יוכלו לפרמל אותו עבורכם בצורה מסודרת, וכך למעשה תקבלו בצורה מסודרת ונגישה תיעוד של כל ה API.
אחד הכלים האלה הוא swagger:
היכנסו ל <https://editor.swagger.io> וטענו את קובץ ה Open API בתוכו.
תוכלו לראות מיידית את כל פירוט ה API, מה הם מקבלים, מה הם מחזירים, איך הדברים נראים וכו'.
בנוסף, אפשר להשתמש גם ב swagger כ http client (כמו postman) ולהריץ את ה API'ים ישירות ממנו (!@%\$).
(למגדילי הראש, אפשר לחשוב על קובץ ה API כעל סוג של "סכמה" המתארת את ה API'ים השונים... רוצים לנחש מה אפשר לעשות עם סכמה בעולם של spring boot ????)

3. בניגוד לתרגילים קודמים, בהם ברגע נתון הייתה טעונה במע' לכל היותר מכונה אחת, כאן למעשה נתמוך ביכולת לטעון מספר מכונות ולהפעיל אותן במקביל.
במסגרת הגדרת קבצי התרגיל התווסף מידע חדש לכל מכונה: שם המכונה (name).
כפועל יוצא השתנתה גם הסכמה (בקטנה) וזה אומר כי עליכם לג'נט מחדש את מחלקות ה JAXB כדי שהן תשקפנה את המבנה החדש של המחלקות. בשלב זה, זה כבר בטח שקוף לכם כי זה חלק מה build לא ? 😊
עקבו אחר [העדכונים בסכמה 3](#) כדי להבין את המבנה החדש. (שוב: בקטנה).
- בנוסף, עליכם לוודא כי שמות המכונה הם ייחודיים. נסיון להוסיף מכונה תחת אותו השם ייכשל עם הודעה מתאימה למשתמש. (זאת כמובן בנוסף לכלל בדיקות הקלט שביצעתם בתרגיל 1 ו 2)
4. בתרגיל זה תבצעו שמירה של נתוני המע' ל DB מוג Postgres.
עליכם לעבוד עם spring-data ולתקשר עם ה DB על מנת לשמור את המידעים השונים (כפי שנלמד בכיתה).
ה DB מגיע מוכן עם מבנה טבלאות המשקפות את המידע הנדרש לשמור במסגרת המע'.
עליכם לעבוד עם ה DB זה בלבד ולתאים את עצמכם לעבוד עם טבלאותיו.
עיקבו אחר נספח ד' כדי להבין את פרטי ההתקשרות וההתחברות אל ה DB.
5. אופן תפעול המע':
המשתמש יוכל להעלות מספר קבצי XML של מכונות למע'. במידה והקבצים מתקבלים הם מופיעים ברשימת המכונות הנתמכות במע' ונשמרים ב DB.
על מנת לעבוד עם מכונה מסויימת, על המשתמש לפתוח Session.
Session מהווה יחידת עבודה המקושרת למכונה, אשר בהקשרה מבוצעות הפקודות השונות של המע'.
במעמד פתיחת ה session, המשתמש בוחר מכונה (על פי שמה) והיא המכונה של ה session הזה (לא ניתן להחליף מכונה לסשן פתוח).
ברגע שהמשתמש פתח session, הוא יקבל ID מסויים.
ה ID הוא מחרוזת לניהולכם; אני ממליץ על עבודה עם GUID – פורמט מוכר של מחרוזות שלא חוזרות על עצמן. (קיראו על המחלקה UUID בג'אווה).
פקודות המע' השונות אשר עובדות על מכונה יקבלו לידן את ה session ID ויעבדו על/בהקשר המכונה המשווייכת אליו.
בגמר העבודה המשתמש סוגר/מוחק את ה session ומכאן ואילך לא ניתן להשתמש בו שנית.
- שימו לב:** ברגע נתון יכולים להתקיים session'ים שונים עם אותה המכונה (חישבו על היבט ה concurrency)
פקודת ההסטוריה יכולה לעבוד בהקשר session מסויים (גם אם הוא סגור) או לעבוד בהקשר של שם מכונה (כלומר עבור כל ה session'ים שבוצעו איתה).
6. מבנה המע' ברמה הטכנית:
השרת ייבנה באמצעות Spring Boot כפרוייקט maven.
הנה מבנה המודולים ב maven שאתם מצופים לעבוד איתם:
1. שלושת המודולים מתרגיל 2 ([enigma-machine](#), [enigma-engine](#), [enigma-loader](#))
2. מודול עבור ניהול הסשנים – [enigma-sessions](#)
3. מודול עבור ניהול שכבת ה DAL – [enigma-dal](#)
4. מודול עבור ניהול ה controller'ים וכל הקשור לתפעול הגישה למע' – [enigma-api](#)
5. מודול נוסף עבור האפליקציה עצמה – מטבע הדברים הוא מכיל רק את ה Main ואת קבצי הקונפיגורציה – [enigma-app](#)
6. כל מודול אחר שעוד משתתף אצלכם במשחק.
את קבוצה 1,2,6 יש לקבץ ל aggregator עצמאי שנקרא [enigma-logic](#).
וכך ה aggregator הראשי () יכיל "כילדים" את [enigma-dal](#), [enigma-app](#), [enigma-api](#) ו [enigma-logic](#).

Controllers:

המע' תכיל מספר controller'ים שדרכם ייחשפו יכולותיה השונות:

1. sessionController – אחראי על תפעול ה session'ים במע'. /session
2. LoaderController – אחראי על מלאכת הטעינה של מכונה. (פקודה 1) נתיב: /load
3. ConfigurationController – אחראי על קבלת וקביעת תצורת מכונה קיימת. (פקודה 2-4,6) נתיב: /config
4. ProcessController – אחראי על הפעלה בפועל של מכונה קיימת. (פקודה 5) נתיב: /process
5. HistoryController – אחראי על כל היבטי ניהול ותפעול ההסטוריה של המכונה. (פקודה 7) נתיב: /history

Services:

עליכם לייצר service (@Service) של spring עבור כל אחד מה controller'ים שתוארו לעיל (ייתכן אפילו יותר מאחד היכן שתבחרו)
כפי שהודגם בשיעור, Service זה אמור לבצע את העבודה הלוגית של המע', כלומר אין לכתוב לוגיקה ישירות ב controller, גם אם היא "פשוטה" ורק קוראת למטודה אחת.

Repository:

עליכם לייצר repository (@Repository) עבור כל אחת מהטבלאות והיישויות שב DB.
ה repository'ים השונים אמורים על הקשר עם שליפת/עדכון המידע ב DB.
כלומר אין לגשת ישירות ל DB משכבת ה service (שלא לדבר על משכבת ה controller, לכל ה NodeJS'ים אי שם בחוץ!) ולהקפיד על הפרדה בין השכבות השונות.

ארטיפקט סופי:

עליכם לייצר ארטיפקט סופי של spring boot שהוא למעשה השרת המכיל את כל הקוד הנדרש להרצת האפליקציה.
שם הארטיפקט הסופי: [enigma-machine-server-ex3.jar](#)

חלוקה למודולים

כפי שהוסבר בחלק הקודם, המודולים של intelliJ תואמים את מבנה המודולים של Maven.
בסוף תרגיל 2 כבר יש לכם aggregator המכיל את כל המע'. אפשר להישאר במסגרת אותו הפרויקט ולהרחיב אותו עם עוד מודולים כפי שתוארו מעלה ולשנות את המבנה שלהם כדי שיתאים למבנה של תרגיל 3 (אני ממליץ על מבנה זה; שימו tag ב git על המצב הנוכחי של הפרויקט שלכם בסוף תרגיל 2 ותמיד תוכלו לחזור אליו במידת הצורך), או לחילופין לפתוח פרויקט חדש לגמרי ב IntelliJ וב Github ולהעתיק את הקוד הקיים אליו ובו לפתח את מסגרת תרגיל 3 (גם אפשרי, לדעתי יותר "מעיק").

איך מתחילים?

אני ממליץ ראשית לעבור על המע' ולבנות אותה כ spring boot application, תוך התעלמות מסוגיית ה DB וה repository, כמו גם מהיכולת להעלות כמה מכונות.

זה יקל עליכם מאוד להתחיל להניע את העסק ולראות דברים עובדים. תוכלו כבר לשחק עם postman ולראות שהכל מתחבר.

אם אין לכם כאלה עדיין, אני ממליץ ליצור מחלקות (כלומר beans) שהן סוג של managers עבור החלקים השונים של המע' (למשל: loaderManager שאחראי על טעינת המע'; HistoryManager וכו').

לאחר שהדברים הבסיסיים עובדים, התחילו בתמיכה בהוספת הקונספט של session ומימוש לאורך כל ה endpoint'ים והלוגיקות השונות.

לאחר מכן התעסקו עם התמיכה בריבוי מכונות. זה כנראה אומר שתצטרכו לייצר עוד איזה manager שיוודע להכיל אוסף של מופעי מכונות (אולי יהיה זה ה sessionManager?), תוך הקפדה על כך שהמכונות וכלל פרטיהן מופרדות אחת מהשנייה כדי שלא יצא ערבוב בין מופעי מכונות שונות.

לבסוף התעסקו עם כל סוגיית ה DB. הגדירו את היישויות הנכונות, את היכולת להתחבר ל DB, ודאו כי אתם מצליחים לשמור ולקרוא מידע וכו'.

סבבה, סיימתי. מה ואיך להגיש ?

עליכם להגיש zip אשר כולל 2 קבצים:

1. קובץ `readme`:

הקובץ צריך להיפתח "בשער" שקיבלתם במסגרת חומרי התרגיל.

2. קובץ `run.bat` אשר ישמש את הבודק בהרצת והרמת הפרויקט.

קובץ דוגמא הועלה לאתר המאמא. **ממליץ בחום** להוריד אותו ורק להחליף בו את ה `git repository` שלכם.

קובץ ה **`readme`** יכיל את הדברים הבאים:

1. הפניה ל `git repository` שלכם, אותו הבודק הולך לעשות `clone`. וודאו כי ה `repository` אינו `private` !

2. תיעוד והסבר כללי (וממצה) של המחלקות העיקריות ותפקידם.

3. הסבר על המערכת, בחירותיכם השונות במקומות שבהם היו לכם בחירה, או כל דבר נוסף העולה על דעתכם שחשוב שהבודק ידע.

פרטים יבשים

צפי תחילת עבודה: 19.2.26

צפי זמן לביצוע: שבועיים

משקל התרגיל: 10 נקודות בונוס

תאריך הגשה: 7.3.26

ציון אפשרי מקסימלי: 100

מטרות התרגיל העיקרית

1. בניית ממשק עבודה WEB'י דרכו אפשר לשאול שאלות על מצב המע' בטקסט חופשי
2. עבודה מול כלי AI לבניית שאילתות SQL בצורה דינמית מול ה DB לשליפת מידע

דרישות

1. בתרגיל זה תלמדו כיצד לעבוד מול כלי AI שונים (Open AI, Gemini וכו') בצורה תכנותית. מהות התרגיל היא לאתגר ולבחון את יכולתכם ללמוד לבד ולהתמודד עם אתגרי היום יום בכל הקשור לעבודה עם כלי ה AI השונים. החומרים והדרישות לתרגיל זה לא נלמדו בקורס. המטרה שלכם היא לעבוד בעיקר ואך ורק עם כלי ה AI השונים ולחקור את אופן פעולתם. (ייתכן ותצטרכו גם לכתוב קוד בעצמכם, ליטרלי; אבל זו לא הציפיה).
אין ערובה לכך שהתרגיל הזה אפשרי או יעבוד בצורה מעולה.
לא ניסיתי אותו בעצמי וייתכן וכולכם תיווכחו לגלות שהדברים פשוט לא עובדים או עובדים חלקית (במגבלות הנתונות).
זה חלק מהתרגיל, וגם אם תספקו תוצאות חלקיות יהיה לזה ערך.
כחלק מהתרגיל תצטרכו גם לתת משוב לגבי אופן העבודה שלכם במסגרתו עם כלי ה AI השונים. המשוב הוא חלק מהתרגיל. (המטרה היא גם ללמוד את אופי העבודה שלכם בתרגילים השונים ולהסיק מזה מסקנות לעתיד).
2. מטרת תרגיל זה היא לאפשר למשתמש לתאר איזה מידע הוא רוצה לקבל בשפה חופשית (אנגלית בלבד), ולהשתמש במודלי ה AI לתרגם את הדרישה לשאילתת SQL המתאימה למבנה הטבלאות. את השאילתה הזו תבצעו מאחורי הקלעים ותחזירו חזרה לכלי ה AI כדי שזה יעבד את התוצאות הגולמיות לכדי תשובה טקסטואלית פשוטה שהמשתמש ידע להבין בקלות.

לשם כך, עליכם ליצור endpoint חדש בשרת:

Request:

url: </enigma/ai>

method: **POST**

body: json with one attribute: "query" which holds the user's prompt

Response:

body: json with this structure:

- **answer**: free text of the final response from the AI
- **sql**: the sql query the AI generated for the user's request

1. המשתמש מקליד שאלה בטקסט חופשי באנגלית. דוגמאות לשאלות:

- How many machines are currently declared in the system?
- List all different session ids that machine "sanity" had gone through
- What is the minimum, maximum and average time it took for processing all the inputs in session id "34929d"?
- How many reflectors are defined in the system?
- What is the machine name that has the longest alphabet?

2. עליכם ליצור Prompt מתאים מול פלטפורמת ה AI הרלוונטית איתה תבחרו לעבוד, אשר ידע להמיר את הבקשה לשאליתת SQL מתאימה אותה תוכלו להריץ מול ה DB.

כחלק מה Prompt עליכם להסביר וללמד את הפלטפורמה על מבנה המע' והטבלאות שבה, כדי שיהיה לה את הידע איך לתרגם את הבקשה המילולית לשאליתת SQL.

3. עליכם לבצע הרצה דינמית של שאליתת ה SQL ולקבל את תוצאותיה.

4. עכשיו עליכם להחזיר לפלטפורמת ה AI את התוצאות הגולמיות של השאליתת ולבקש ממנו לייצר את התשובה הסופית למשתמש.

5. את התשובה הסופית מחזירים למשתמש הקצה.

האתגרים המרכזים הם להבין כיצד לגרום לצ'ט לייצר את השאליתת המתאימות תוך שהוא מבין את מבנה הטבלאות (אולי להעלות לו את ה SQL שיצר אותן?) כמו גם להבין איך מריצים שאליתת דינמית ולוקחים את התוצאות שלה (אשר אותם אתם לא יודעים לצפות) בצורה גנרית כדי להזין אותן חזרה לפלטפורמה כדי שזו תעבד אותן (אולי זה המקום לחזור לשורשים של JDBC?)

אני מציע לבצע את הנסיונות השונים ראשית מול כלי הצ'ט של הפלטפורמה בה אתם רוצים להשתמש (ובפרט לעבוד רק עם המודל המסוים שאתם מורשים לו במידה וזה רלוונטי), כדי לעשות את כל הניסויים ולראות אם הדברים עובדים כהלכה. רק אחרי שיש לכם הוכחת נכונות ורואים שהתהליך ישים – לעבור ולפתח את כל זה בקוד תוך קריאה לפלטפורמה.

4. עליכם לבנות ממשק וובי שבו המשתמש יקבל חווית עבודה רגילה ומוכרת עם מסך צ'ט פשוט.

המשתמש יכניס שאלה והתשובה תוגש לו אחרי עיבוד במע'.

ממשק זה יפנה אל ה endpoint החדש בפלטפורמה שלכם ([/enigma/ai](#)) ומשם יתחיל כל התהליך.

את התשובה יש להציג בשני חלונות נפרדים:

- חלון ראשי שבו תוצג התשובה הטקסטואלית
- חלון משני שבו תוצג שאליתת ה SQL הרלוונטית שבוצעה מאחורי הקלעים.

אין צורך להעשיר את הממשק עם פיצ'רים כמו הסטוריה או כתיבת התשובה בצורה איטרטיבית (אות אחרי אות כמו שהצ'ט עושה). הממשק יכול (וצריך) להיות פשוט ביותר.

אתם יכולים לבנות את הממשק באיזו דרך שרק תבחרו, כמובן תוך שימוש בכלי ה AI.

במסגרת הפיתוח אתם רשאים להשתמש בכל framework של FE שאתם מכירים/נתקלים (react, angular, vue וכו').

כך או אחרת לקליינט זה אסור שיהיו השלכות על אופן תפעול והקמת השרת: כלומר השרת הוא אותו השרת spring boot שפיתחתם במסגרת תרגיל 3.

כפועל יוצא עליכם לספק קובץ readme עצמאי במסגרת הגשה זו שבמסגרתו תסבירו לבודק כיצד עליו להפעיל את האפליקציה.

במידה ואתם מצפים ממנו שירץ משהו – כיתבו לו קובץ bat שאותו הוא מפעיל ואמור להרים את מה שצריך כדי לתמוך בתפעול הקליינט.

אפשר להניח שאצל הבודק מותקן Node JS.

אפשר לבצע פקודות npm install במסגרת קובץ ה bat.

לא ניתן לבקש מהבודק להוריד או להתקין כלים אחרים לטובת תפעול הקליינט.

כדי להימנע מהנחות ופוטנציאל תקלות – אני כמובן ממליץ שהקליינט שלכם יהיה ארוז כ docker image וכל מה שהבודק יצטרך לעשות זה להריץ אותו...

בקובץ ה readme כיתבו בצורה מפורשת לאיזה URL על הבודק להיכנס כדי לעבוד עם הקליינט שלכם.

5. על מנת לעבוד עם כלי ה AI במצב הפיתוח (כלומר לכתוב קוד שקורא להם) יש לעבוד עם פלטפורמות ה API של הכלים השונים (למשל ל Open AI יש את [Open AI Platform](#) ול Gemini יש את [Google AI Studio](#) וכו'). כדי לגשת אל פלטפורמות אלה יש לקבל API KEY. זהו למעשה TOKEN (מחרוזת תווים באנגלית) שאותו צריך להוסיף במסגרת הגשת הבקשה לפלטפורמת ה AI הרלוונטית, וכך הפלטפורמה יודעת "את מי לחייב".

שימו לב: עבודה עם פלטפורמת AI שונה מבחינת מודל החיוב לעבודה עם הצ'ט של אותו הספק. למשל: אם יש לכם חשבון חינמי לעבודה עם chat GPT של Open AI אין זה אומר שיש לכם יכולת חינמית לעבוד מול Open AI Platform. מדובר בשני חשבונות שונים, בעלי התנהלות ואופי חיוב שונה.

היות וכך, אם ברשתכם חשבון פרטי למי מהפלטפורמות השונות וברצונכם להשתמש בו (כלומר לייצר API KEY עבור התרגיל) - עשו זאת. באפשרות זו תוכלו לעבוד עם כל מודל שעולה על רוחכם.

אם אין ברשותכם חשבון, תוכלו לבקש ממני לייצר עבורכם API KEY ב Open AI Platform ותוכלו לעבוד עם ה KEY הזה, אך שימו לב שבמקרה זה אני מנטר את הפעילות על ה KEY ותהיו מוגבלים לעבוד איתו רק עבור פרויקט זה, ולא תהיה לכם גישה לכל מודל (יש מודלים יקרים יותר וכאלה זולים יותר).

כך או אחרת צפי התשלום על העבודה מול הפלטפורמה במסגרת תרגיל זה צפוי להיות זעום (ברמת הסנטים הבודדים).

6. בקובץ ה readme כיתבו על תהליך העבודה עם כלי ה AI.

יש להתייחס לנקודות הבאות (ומוזמנים להרחיב על נקודות נוספות ככל העולה על רוחכם):

1. עם איזה כלי בחרתם לעבוד ולמה? האם ניסיתם כלים אחרים?
2. האם היו אזורים שבהם הרגשתם שכלי ה AI לא "מספק את הסחורה" ונדרשה התערבותכם כמתכנת/ת?
3. האם נתקלתם בבאג או דרישה כלשהיא שה AI פשוט לא הצליח לפתח?
4. נסו להעריך ולתאר כמה הפיתוח כולו (של הקליינט וצד הלוגיקה בשרת) בוצע במלואו ע"י ה AI בלבד (כלומר אתם רק מנסחים פרומפטים; לא מתערבים בקוד) או כמה הוא דרש התערבות שלכם והבנה בנעשה.
5. כמה זמן לקח לפתח ממשק זה? נסו להעריך אם הייתם צריכים לפתח אותו לבד ללא תמיכת AI – כמה זמן זה היה לוקח לכם?
6. האם יש לכם היכרות מוקדמת עם עולם פיתוח ה FE? אם כן, האם/כמה היא סייעה לכם (לדעתכם) במימוש התרגיל?
7. האם בהשוואה לפיתוח של כלל התרגילים בקורס – סגנון עבודה מונחה-AI זה היה לכם כיף?

7. ההגשה של תרגיל זה היא למועד ההגשה של תרגיל 3 הרגיל, אך בתיבת הגשה נפרדת.

המלצות וחיידים 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 מבחון 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 3 רוטורים.
להלן טבלה של מספר עיבודי מחרוזות במכונה הנבנית מ sanity-small, עם קוד ראשוני של $\langle 3,2,1 \rangle \langle CCC \rangle$ המחרוזות הוכנסו משמאל לימין, תוך איפוס המכונה חזרה לראשית הקוד (CCC) לפני עיבוד כל מחרוזת:

#	מחרוזת קלט	מחרוזת פלט
1	AABBCCDDEEFF	FFCCBBEEDDAA
2	FEDCBADDEF	ADEBCFEEDA
3	FEDCBAABCDEF	ADEBCFFCBEDA
4	AFBFCFDFEFFF	FACABAEADAAA
5	AAAEIEBBDDCCCF	FFFDDDDCCCEEEAFEDCB

3. תוכלו לבדוק עצמכם גם ע"י בדיקה שאתם באמת יכולים להצפין ולפענח מחרוזות תווים הלוך וחזור באמצעות המכונה שלכם לחלוטין – כלומר התהליך הפוך לחלוטין.
4. 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 $\langle 1,2,3 \rangle \langle ODX \rangle$ 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	QMJDORMMYQBJDVSBR
4	WOWCANTBELIEVEITACTUALLYWORKS	CVRDIZWDAWQKUKBVHJILPKRNDXWIY
5	JAVARULES	MRUHFRRZR

פירוט המכונה כפי שהיא נבנית מקובץ sanity-small אך תוך שימוש בשני רוטורים בלבד (מפאת מקום) ובאמצעות הגדרת קוד ראשוני של $\langle B|C \rangle \langle O \rangle \langle CC \rangle \langle 2,1 \rangle$

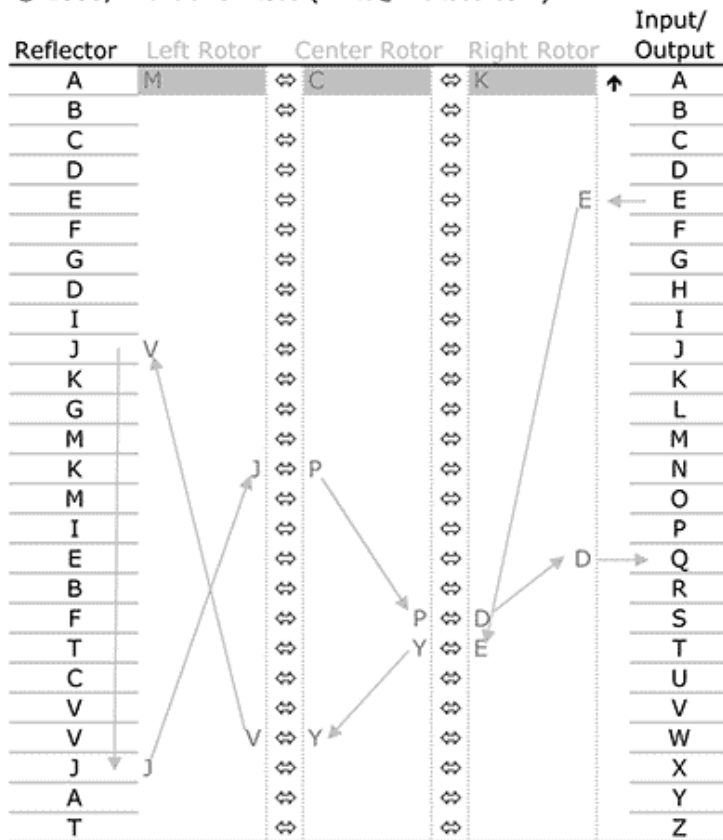
#	תיאור המצב	דיאגרמה
1	מצב התחלתי בו נבחרו לשימוש רוטורים 1 (ראשון מימין) ו 2 (שני משמאל). כמו כן נבחר משקף מספר 1.	
2	לאחר קביעת קוד סודי הממקם את שני הרוטורים במיקום 3 (או ב 'C')	

<div> <div> <div>1</div> <div>1</div> <div>2</div> <div>3</div> <div>1</div> <div>2</div> <div>3</div> </div> <div> <div>2</div> <div>F</div> <div>D</div> <div>C</div> <div>E</div> <div>A</div> <div>F</div> <div>E</div> <div>A</div> <div>B</div> <div>B</div> <div>D</div> <div>C</div> </div> <div> <div>1</div> <div>C</div> <div>D</div> <div>B</div> <div>E</div> <div>A</div> <div>F</div> <div>A</div> <div>F</div> <div>B</div> <div>B</div> <div>D</div> <div>C</div> </div> <div> <div>1</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> <div> <div>A</div> <div>B</div> <div>C</div> <div>D</div> <div>E</div> <div>F</div> </div> </div>	<div>3</div> <p>לפני כל הקלדת תו, המע' מבצעת פסיעה אחת קדימה החל מהרוטור הימני (1). במקרה הנ"ל הזיז של רוטור 1 הגיע לחלופית ההצצה ולכן הוא גורם גם לקידום רוטור מס' 2 צעד אחד קדימה</p>	
<div> <div> <div>1</div> <div>1</div> <div>2</div> <div>3</div> <div>1</div> <div>2</div> <div>3</div> </div> <div> <div>2</div> <div>F</div> <div>D</div> <div>C</div> <div>E</div> <div>A</div> <div>F</div> <div>E</div> <div>A</div> <div>B</div> <div>B</div> <div>D</div> <div>C</div> </div> <div> <div>1</div> <div>C</div> <div>D</div> <div>B</div> <div>E</div> <div>A</div> <div>F</div> <div>A</div> <div>F</div> <div>B</div> <div>B</div> <div>D</div> <div>C</div> </div> <div> <div>1</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> <div> <div>A</div> <div>B</div> <div>C</div> <div>D</div> <div>E</div> <div>F</div> </div> </div>	<div>4</div> <p>הצפנת האות C ← A:</p> <p>הקלדת האות C פוגשת קודם כל את לוח התקעים. יש תקע המחליף בין C ו B ולכן הלכה למעשה הוגשה לכניסת הרוטורים האות B.</p> <p>האות B קודדה לאות A. תרשים החצים הבא מציג את כיוון הזרם בחייוטי הרוטורים השונים. חץ כחול הוא כיוון ההלוך וחץ אדום הוא כיוון החזור.</p> <p>בחזור האות A ללוח התקעים היא יוצאת משם AS IS מאחר ואין תקע מחליף לאות זו.</p>	
<div> <div> <div>1</div> <div>1</div> <div>2</div> <div>3</div> <div>1</div> <div>2</div> <div>3</div> </div> <div> <div>2</div> <div>F</div> <div>D</div> <div>C</div> <div>E</div> <div>A</div> <div>F</div> <div>E</div> <div>A</div> <div>B</div> <div>B</div> <div>D</div> <div>C</div> </div> <div> <div>1</div> <div>B</div> <div>E</div> <div>A</div> <div>F</div> <div>F</div> <div>A</div> <div>E</div> <div>B</div> <div>D</div> <div>C</div> <div>C</div> <div>D</div> </div> <div> <div>1</div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> <div> <div>A</div> <div>B</div> <div>C</div> <div>D</div> <div>E</div> <div>F</div> </div> </div>	<div>5</div> <p>לקראת הקלדת אות נוספת, המע' מבצעת פסיעה אחת קדימה. הפעם רק רוטור 1 פוסע</p>	

	<p>הצפנת האות D ← A:</p> <p>האות D אינה מוחלפת בלוח התקעים. הקדלת האות D קודדה לאות A.</p> <p>האות A אינה מוחלפת בלוח התקעים.</p> <p>שימו לב כי בעקבות תזוזת הרוטורים ייתכן כי 2 תווים שונים יקודדו לאותו תו</p>	6
	<p>לקראת הקלדת אות נוספת, המע' מבצעת פסיעה אחת קדימה. הפעם רק רוטור 1 פוסע</p>	7
	<p>הצפנת האות A ← F:</p> <p>האות A אינה מוחלפת בלוח התקעים. הקדלת האות A קודדה לאות F</p> <p>האות F אינה מוחלפת בלוח התקעים.</p>	8

Paper Enigma Machine

© 2003, Michael C. Koss (mike@mckoss.com)



Setup

1. Select left/center/right rotors.
2. Position initial wheel positions by sliding the indicated window letter up to the first row.

Operation

[Start at the input column at right, then work left to reflector, and then back to the right to the output column.]

1. If the ↑ notch appears in the window row, shift that rotor and the rotor to the left up one row (the Right Rotor is always shifted up one row before each letter is encoded/decoded).
2. Select letter to encode/decode in the Input column.
3. Read adjacent letter, X, in right hand column of the Right Rotor; select the letter X in the left hand column of the Rotor.
4. Repeat for Center Rotor.
5. Repeat for Left Rotor.
6. Read the adjacent letter, R, in the Reflector; select the other letter R in the Reflector.
7. Read adjacent letter, Y, in left hand column of the Left Rotor; select the letter Y in the right hand column of the Rotor.
8. Repeat for Center Rotor.
9. Repeat for Right Rotor.
10. Write down the adjacent letter, Z, in the output column. Repeat for each letter of the message.

Example: Initial setting: I-II-III: MCK, Letter E encodes to Q.
Sample Message: QMJIDO MZWZJFJR

Rotor I	Rotor II	Rotor III
A E	A A	A B
B K	B J	B D
C M	C D	C F
D F	D K	D H
E L	E S	E J
F G	F I	F L
G D	G R	G C
H Q	H U	H P
I V	I X	I R
J Z	J B	J T
K N	K L	K X
L T	L H	L V
M O	M W	M Z
N W	N T	N N
O Y	O M	O Y
P H	P C	P E
Q X	Q Q	Q I
R U	R G	R W
S S	S Z	S G
T P	T N	T A
U A	U P	U K
V I	V Y	V M
W B	W F	W U
X R	X V	X S
Y C	Y O	Y Q
Z J	Z E	Z O
A E	A A	A B
B K	B J	B D
C M	C D	C F
D F	D K	D H
E L	E S	E J
F G	F I	F L
G D	G R	G C
H Q	H U	H P
I V	I X	I R
J Z	J B	J T
K N	K L	K X
L T	L H	L V
M O	M W	M Z
N W	N T	N N
O Y	O M	O Y
P H	P C	P E
Q X	Q Q	Q I
R U	R G	R W
S S	S Z	S G
T P	T N	T A
U A	U P	U K
V I	V Y	V M
W B	W F	W U
X R	X V	X S
Y C	Y O	Y Q
Z J	Z E	Z O

Rev: 2003-03-11-1047

במסגרת התרגיל תעבדו עם Postgres DB.

ה DB ניתן לכם בדמות docker container.

כדי להוריד ולהרים את ה DB לאוויר יש להפעיל אותו באמצעות docker:

docker run -d -p 5432:5432 theshultz/patmal-enigma-postgres:1.0

פרטי ההתחברות ל DB:

User: **postgres**

Password: **enigma**

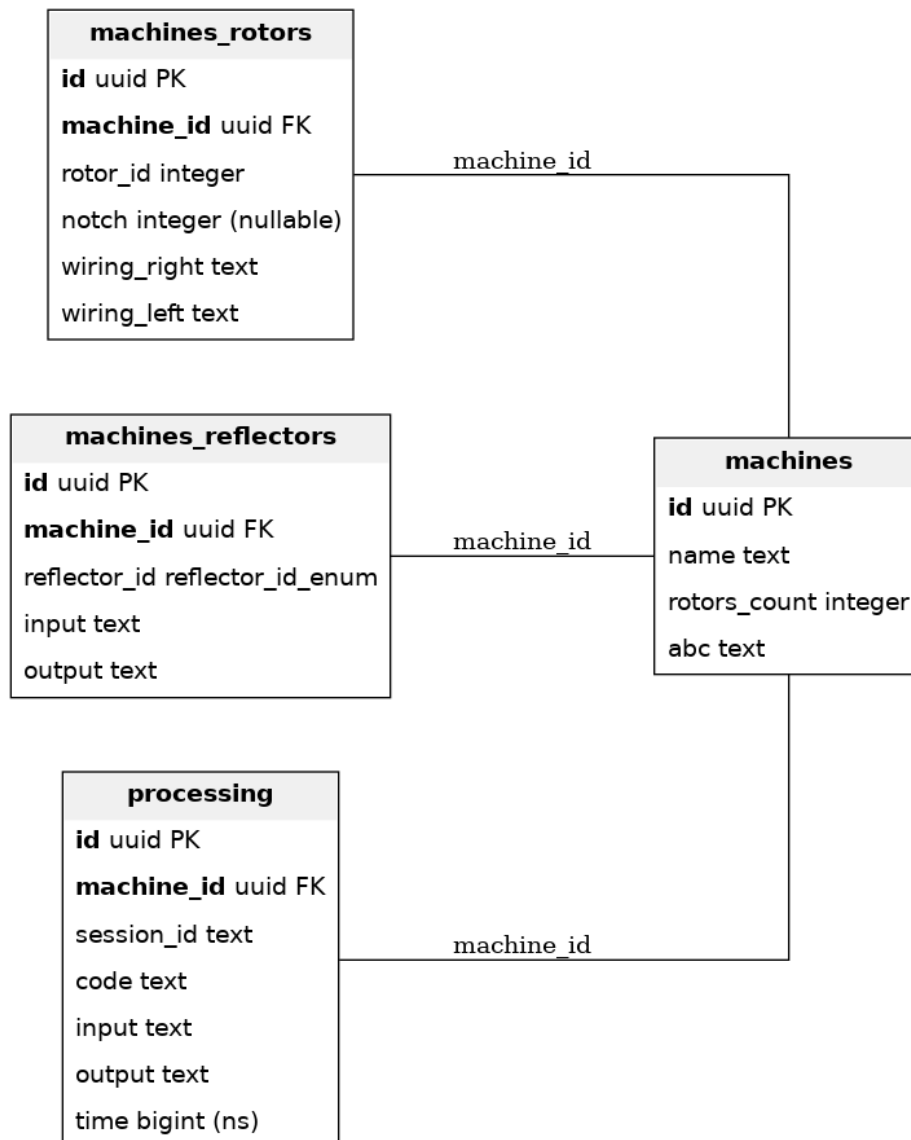
Database: **enigma**

Domain:port: **localhost:5432**

Full connection string: **jdbc:postgresql://localhost:5432/enigma**

לנוחיותכם, העלתי גם את קובץ ה SQL ששימש לבניית הטבלאות (אפשר לחשוב עליו כעל... סכמה ? 😊)

מובטח לכם, ואפשר להניח, כי בעת הרמת והפעלת המע' שלכם ה DB CONTAINER כבר יהיה באוויר, ויפעל ריק ממידע.
הנה תרשים הטבלאות והיחסים ביניהן (ERD):



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 insensitive 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)

מבנה המכונה מאוגד תחת האלמנט BTE-Enigma ומכיל בתוכו מספר אלמנטים/מאפיינים נוספים:

```
<BTE-Enigma xmlns:xsi="http://www.w3.org/2001/
  <ABC>
    ABCDEF
  </ABC>
  <BTE-Rotors>
    <BTE-Rotor id="1" notch="4">
      <BTE-Positioning right="A" left="F"/>
      <BTE-Positioning right="B" left="E"/>
      <BTE-Positioning right="C" left="D"/>
      <BTE-Positioning right="D" left="C"/>
      <BTE-Positioning right="E" left="B"/>
      <BTE-Positioning right="F" left="A"/>
    </BTE-Rotor>
    <BTE-Rotor id="3" notch="6">
    <BTE-Rotor id="2" notch="1">
    </BTE-Rotors>
  <BTE-Reflectors>
    <BTE-Reflector id="I">
      <BTE-Reflect input="1" output="4"/>
      <BTE-Reflect input="2" output="5"/>
      <BTE-Reflect input="3" output="6"/>
    </BTE-Reflector>
    <BTE-Reflector id="II">
    </BTE-Reflectors>
</BTE-Enigma>
```

#	id	שם	מהות
1	Element	BTE-Enigma	אלמנט זה מכיל את כל פרטי ההגדרה של המכונה
2	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: '&' = '&' ו '<' = '<'. יש לבצע trim למחרוזת ה ABC כדי לנקותה מתווים מיותרים בראשיתה ובסופה.
3	Element	BTE-Rotors	מכיל את הגדרות הרוטורים השונים
4	Element	BTE-Rotor	מתאר מבנה של רוטור אחד במע'.
5	Attribute	id	מספר הזהות של הרוטור. צריך להיות יחיד בין כל הרוטורים השונים
6	Attribute	notch	מיקומו של זיז הדחיפה על גבי הרוטור המסויים. המספר ניתן בבסיס 1
7	Element	BTE-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

8	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.
9	Element	BTE-Reflectors	מכיל את הגדרות המשקפים השונים בשימוש המכונה
10	Element	BTE-Reflector	מתאר מבנה של משקף בודד
11	Attribute	Id	מספר מזהה של המשקף הנ"ל. המספר מוגבל ל 5 תווים בספירה רומית: I, II, III, IV, V
12	Element	BTE-Reflect	שיקוף (מיפוי) בודד של כניסה אחת ליציאה אחת.
13	Attribute	input\output	מתארים את השיקוף שיש לבצע בין כניסה מספרית מסויימת (input) ליציאה מספרית מסויימת (output). המספרים ניתנים בבסיס 1. כמות המיפויים צפויה להיות חצי מאורכו של ה ABC

סכמה תרגיל 2

בתרגיל 2 מצטרף אטריביוט נוסף לקובץ – rotors-count המגדיר את כמות הרוטורים בשימוש בפועל. זהו מספר שלם.
הוא יושב על גבי האלמנט הראשי BTE-Enigma:

```
<BTE-Enigma rotors-count="2"
```

סכמה תרגיל 3

בתרגיל 3 מצטרף attribute נוסף לקובץ – name המגדיר את שם המכונה. זוהי מחרוזת תווים.
הוא יושב על גבי האלמנט הראשי BTE-Enigma

```
<BTE-Enigma name="sanity"
```

סכמה תרגיל 1

