מס' גרסה: **7**

פיתוח תוכנה מבוסס java

ABS – Alternative Banking System 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](#_Toc101611228)

[כללי 4](#_Toc101611229)

[איך להגיש תרגילים באיחור, ולהישאר בחיים 5](#_Toc101611230)

[הנחיות כלליות לכתיבת התרגיל 6](#_Toc101611231)

[תרגיל reflection – תרגיל עצמאי ביחידים (5%) - הגשה: 14.3.22 8](#_Toc101611232)

[כללי 8](#_Toc101611233)

[מבנה התרגיל 8](#_Toc101611234)

[ניקוד 8](#_Toc101611235)

[איך בודקים ? 9](#_Toc101611236)

[מה מגישים ? 9](#_Toc101611237)

[מקרה בדיקה לדוגמא 9](#_Toc101611238)

[A.B.S – Alternative Banking System 11](#_Toc101611239)

[מטרת התרגיל(ים) בקורס 11](#_Toc101611240)

[כללי 11](#_Toc101611241)

[מע' בנקאית אלטנרטיבית 11](#_Toc101611242)

[לקוחות 12](#_Toc101611243)

[ציר הזמן 12](#_Toc101611244)

[הלוואה 12](#_Toc101611245)

[תהליך ההלוואה 13](#_Toc101611246)

[הנחיות ספציפיות למימוש מערכת ABS 13](#_Toc101611247)

[תרגיל 1 – מימוש **A.B.S** כאפליקציית Console (25%) - הגשה: 9.4.22 14](#_Toc101611248)

[פרטים יבשים 14](#_Toc101611249)

[דרישות 14](#_Toc101611250)

[חלוקה למודולים 18](#_Toc101611251)

[איך מתחילים ? (המלצה...) 19](#_Toc101611252)

[בונוסים 19](#_Toc101611253)

[סבבה, סיימתי. מה ואיך להגיש ? 19](#_Toc101611254)

[שאלות ותשובות 19](#_Toc101611255)

[תרגיל 2 – מימוש **ABS** כאפליקציית JavaFX (35%) – הגשה: 4.6.22 20](#_Toc101611256)

[פרטים יבשים 20](#_Toc101611257)

[דרישות 20](#_Toc101611258)

[חלוקה למודולים 22](#_Toc101611259)

[איך מתחילים ? (המלצה...) 22](#_Toc101611260)

[בונוסים 23](#_Toc101611261)

[סבבה, סיימתי. מה ואיך להגיש ? 23](#_Toc101611262)

[שאלות ותשובות 23](#_Toc101611263)

[תרגיל 3 – מימוש **ABS**כאפליקציית Client - Server (35%) – הגשה: 10.7.22 24](#_Toc101611264)

[פרטים יבשים 24](#_Toc101611265)

[דרישות 24](#_Toc101611266)

[אפליקציית לקוח (Customer) 25](#_Toc101611267)

[אפליקציית מנהל (Admin) 26](#_Toc101611268)

[חלוקה למודולים 26](#_Toc101611269)

[איך מתחילים ? 27](#_Toc101611270)

[בונוסים 27](#_Toc101611271)

[סבבה, סיימתי. מה ואיך להגיש ? 28](#_Toc101611272)

[שאלות ותשובות 28](#_Toc101611273)

[נספח א' – דוגמא למתן הלוואות ופריסת תשלומים 29](#_Toc101611274)

[נספח ב' – תיאור מהלך החיים של הלוואה 30](#_Toc101611275)

[נספח ג' – מבנה אפליקציה גרפית 31](#_Toc101611276)

[נספח ד' – תיאור מבנה המערכת באמצעות קובץ XML 35](#_Toc101611277)

[סכמת קובץ XML – גרסה I 35](#_Toc101611278)

[סכמת קובץ XML – גרסה III 36](#_Toc101611279)

[נספח ה' – קבצי סכמה לוידוא קובץ ה XML 37](#_Toc101611280)

דרישות הקורס

## כללי

1. בקורס אין בחינה אך חובה להגיש תרגילים (סה"כ 4).
2. 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: 14.3.22

## כללי

**מועד הגשה: 14.3.2022 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();  
 }  
}

A.B.S – Alternative Banking System

## מטרת התרגיל(ים) בקורס

חלק א' – מימוש ליבת המע' ופיצ'רים בסיסים, תוך תפעול האפלקיציה דרך ממשק משתמש מסוג Console

חלק ב' – הרחבת אפשרויות המע' ופיתוח תצוגה גרפית על בסיס אפליקציית Java FX (Desktop).

חלק ג' – ניהול המע' בתצורה מבוזרת תוך הסתמכות על ארכיטקטורת Client-Server

## כללי

המע' הכלכלית נשלטת מזה עשרות שנים ע"י גורם מרכזי שנקרא הבנק.

אחד מתפקידיו העקריים של הבנק הוא לאפשר את הנעת הכלכלה ע"י מתן אפשרות להזרמת הכספים בין מקור מסויים (בדר"כ הבנק) אל היעד (יחידים/עסקים). האמצעי העיקרי במודל זה הוא מתן הלוואות כספים מסוגים שונים לגורם שדורש אותם: משכנתא / הקמת עסק / סגירת מינוס וכו'.

אבל הבנק לא פראיר.

אימרה ישנה נושנה גורסת כי: "..מי שמתעסק עם כסף – עושה כסף..." וכזה הוא גם הבנק.   
בשבתו כגורם מאפשר מרכזי למשחק הכלכלי – הבנק עומד במרכז כל העסקאות, ועל כל מעבר כספים שכזה הוא גוזר עמלה/ריבית. בהיעדר תחרות (מי אמר מבנה ריכוזי של הבנקים ?) העמלות/ריביות בבנק הולכות וגדלות ולמשתמשי הקצה (אנשים פרטיים/עסקים) אין אלא להיכנע למודל ולהיעשק בלית ברירה – פשוט כי אין אלטרנטיבה.

האמנם ?

נניח כי אדם פרטי היה צריך הלוואה לצרכים מסויימים. תארו לכם כי במקום לפנות אל הבנק (העושק) הוא היה פונה לחברו (החביב), שבמקרה יש לו סכום כסף פנוי.   
החבר היה יכול להלוות לו את הכסף, תוך שהם מגיעים להסכמה על הסדר ופריסת תשלומים המקובלת על שניהם, ובתקווה ומתוך שאיפה כי יכלו היו להסכים על מתן עמלה (ריבית) נמוכה יותר כך שגם הלווה וגם המלווה היו מרווחים:  
הלווה היה מקבל הלוואה בריבית נמוכה יותר ; המלווה היה מייצר לעצמו אפיק השקעה ריווחי יותר על הכספים הפנויים שלו.

זה בוודאי נחמד ואפשרי בין 2 חברים שמכירים זה את זה וסומכים זה על זה לאורך זמן.   
אבל מה אם היינו יכולים למנף את המודל הזה בין אוסף של אנשים פרטיים שיש להם כספים פנויים ורוצים להלוות אותו לאחרים, לבין אוסף (אחר ?) של אנשים פרטיים שצריכים ללוות כספים ? אולם במקרה זה, אם אין היכרות מוקדמת ומקדם בטחון, איך נבטיח כי הלווים יעמדו בפריסת התשלומים שלהם והמלווים לא יסכנו לשווא את כספיהם ?

## מע' בנקאית אלטנרטיבית

בשנים האחרונות פועלות בתחום מס' חברות/ארגונים המציעים בדיוק את השירות הזה: פלטפורמה המאפשרת חיבור בין מלווה ללווה פרטי תוך הבטחת ריבית נמוכה ללווה מחד, והשקעה בעלת תשואה סבירה ובטוחה למלווה מנגד. (למתעניינים, [בלנדר](https://www.blender.co.il/) ו [BTB](https://www.btbisrael.co.il/) הם 2 דוגמאות לחברות ישראליות הפועלות בתחום זה בדיוק).

בתרגיל זה נבנה מע' בנקאית אלטרנטיבית דומה.

המע' תאפשר לקיים הלוואות בתוכה, ותגדיר 2 סוגי לקוחות: לווים ומלווים.   
הלווים יוכלו להגיש בקשה לקבלת הלוואה. כל הלוואה תאופיין במספר פרטים (ראו פירוט בהמשך).   
מן הצד השני המע' תאפשר למלווים להגדיר איזה סכום הם מעוניינים להשקיע ומהן רמות הסיכון שהם מוכנים להן (ראו פירוט בהמשך).  
המע' תפעיל אלגוריתם שיבוץ מיוחד אשר יציע למשקיע להלוות את כספיו למספר הלוואות, כך שהסיכון שלו מפוזר למעשה בין מספר אפיקים. בכך המע' מקטינה את הסיכוי לאובדן הכספים במידה והלוואה לא משולמת בזמן (או בכלל).  
המע' תשקף את מלוא הנתונים על התהליך והתקדמותו הן ללווים והן למלווים.   
כמו כן המע' תאפשר למלווים גם "למכור" הלוואה קיימת, במידה והם רוצים/צריכים את הכספים שלהם מוקדם יותר ממועד סיום ההלוואה.

## לקוחות

לקוח במע' יכול לתפקד הן כלווה והן כמלווה.   
לכל לקוח המע' תנהל חשבון פנימי ותדע להציג את התנועות בחשבון ככל שציר הזמן מתקדם.

הלקוח יכול למשוך או להפקיד כספים מ/אל חשבונו.

לקוח יכול להגיש בקשה להלוואה (בשבתו כ"לווה"), או להקצות סכום מסויים מהיתרה בחשבונו לטובת הלוואות לאחרים (בשבתו כ"מלווה".)

חשבונו של הלקוח לא יכול להיכנס ליתרה שלילית (אין אוברדרפט במע').

ללקוחות יהיו שמות מזהים חד חד ערכים. (אין 2 לקוחות שונים עם שמות זהים)

## ציר הזמן

כאשר עוסקים במתן (והחזר) הלוואות יש משמעות לציר הזמן: מתי הלוואה מתחילה ? מתי היא מסתיימת ? מהו ההחזר החודשי הצפוי בכל פרק זמן נתון ? וכן הלאה.

במסגרת תרגיל זה ננהל ציר זמן עצמאי, דמיוני, ללא קשר לתאריך מציאותי כזה או אחר (כדי לפשט ולהקל על הדברים).   
ציר הזמן נמדד ב**י**חידות **ז**מן (**י"ז**) שמסמלות פרק זמן מסויים (יום/חודש/חצי דקה – מה שבא לכם).   
במהלך העבודה במע' נוכל להזיז את ציר הזמן (בצורה מלאכותית) בכמה י"ז'ים שנרצה קדימה (או אחורה ???) וכך לסמלץ את התקדמות והנעת הכספים ממקום למקום.

בכל רגע נתון המע' נמצאת בי"ז נוכחי. בהפעלת המע' הי"ז הוא 1. הי"ז הוא מספר שלם בלבד. (אי אפשר לבלות בי"ז 3.5...)

## הלוואה

מי שמגיש הלוואה חדשה במע' הוא הלקוח בשבתו כ"לווה".

בהלוואה יש לציין את גובה הסכום הנדרש (נקרא קרן) ואת הריבית שתשולם בגינו בפרק זמן מסויים (במציאות בדר"כ מדברים על מסגרת ריבית שנתית)

תשלום ההחזר של כל הלוואה כולל בתוכו 2 חלקים: מרכיב הקרן, ומרכיב הריבית.  
במציאות יש שיטות שונות לפריסה וחלוקה של 2 המרכיבים הנ"ל בהחזרי ההלוואה (למשל, שפיצר, קרן שווה וכו') אולם כדי לשמור על דברים פשוטים וקלים להבנה (ומימוש) – בכל מועד תשלום ישולם מרכיב זהה של הקרן ועליו הריבית המוצעת (מרכיב זהה גם כן). כמו כן במציאות יש להחזיר תשלום "כל חודש". במע' שלנו ייתכן כי ניתן יהיה לבצע החזרים כל כמות י"ז שתוגדר ע"י מבקש ההלוואה.  
עיקבו אחר [נספח א'](#appendixA) המתאר 2 דוגמאות למתן הלוואות והתנהלותן במע'.

בהגשת הלוואה חדשה למע' יש לכלול את הפרטים הבאים:

* שם ייחודי המזהה חד חד ערכית את ההלוואה הנ"ל.
* שם הלקוח שהגיש אותה
* סכום ההלוואה המקורי
* סה"כ י"ז להלוואה
* קצב התשלומים לי"ז: כל כמה י"זים ישולם החזר ההלוואה ? מספר שלם בין 1 לסה"כ י"ז להלוואה.
* גובה ריבית שתשולם בכל תשלום, מספר שלם באחוזים: (0, 100]
* קטגוריה: מידע המתאר את סוג ההלוואה: עסק, שיפוץ בית, סגירת מינוס, אירוע וכו'. זהו סט סופי ומוגדר של אפשרויות.

מידעים דינמים שנצברים עבור הלוואה:

* רשימה של זיהוי המלווים שהעניקו את ההלוואה
* מס' י"ז לסיום ההלוואה
* סך הריבית ששולמה על ההלוואה עד כה. סך הריבית שעוד עתידה להיות משולמת על ההלוואה.
* סך הקרן ששולמה בהלוואה עד כה. סך הקרן שעוד נותרה להיות משולמת.
* האם ההלוואה פעילה ?

במהלך התנהלות ההלוואה במע' היא עוברת בין כמה תחנות (life cycle). ברגע שהלוואה מוגשת למע' היא בסטטוס New.

## תהליך ההלוואה

כדי להעניק הלוואה ולהשקיע את כספיו, הלקוח המלווה יגדיר מספר פרמטרים שלפיהם המע' תתאים לו את מנעד ההלוואות הנדרש:

* סכום כולל שהוא מוכן להשקיע בסבב זה. מס' שלם החסום בגובה היתרה הפנויה בחשבונו. זהו נתון חובה.
* קטגוריה/ות שהוא מעוניין להשקיע בהן.
* מינימום ריבית להלוואה שהוא מוכן לקבל. מספר שלם גדול מ 0.
* מינימום זמן י"ז להלוואה. מספר שלם, גדול מ 0.
* מקסימום אחוז בעלות על הלוואה. מס' שלם בין 1 ל 100.
* מקסימום הלוואות פתוחות ללווה

להוציא את הנתון הראשון (גובה הסכום), אין חובה לציין את כל הפרטים. פרט שלא מצויין כמוהו כ"הכל מתאים".

כעת המע' תפעיל את אלגוריתם השיבוץ אשר ימצא את כל ההלוואות העומדות בתנאים שהוא הציב.   
ככל שנמצאו כאלה (ולפעמים לא יימצאו כאלה), המע' תחלק את סכום השקעתו להלוואות אלה שווה בשווה, ככל שניתן.

מרגע שכספי מלווה מוקצים להלוואה מסויימת הוא מופיע כמי שמשוייך כמלווה להלוואה זו.   
הכספים שהוא הקצה יורדים מיידית מחשבונו ועוברים לחשבון זמני של ההלוואה עצמה (עדיין לא ללווה !)  
מרגע שיש לפחות מלווה אחד שהעביר כספיו להלוואה היא מוגדרת בסטטוס "Pending".

כאשר מלוא הכספים הנדרשים להלוואה מסויימת ניתנים לה ע"י אוסף של מלווים (ייתכן רק לאחר כמה סבבים של שיבוצים), רק אז היא עוברת לסטטוס "Active" והכספים שנצברו בה מועברים לחשבונו של הלווה.   
יש לתעד במסגרת ההלוואה את הי"ז הנוכחי שבו הושלם התהליך וההלוואה הופכת ל Active.

ככל שציר הזמן מתקדם (ע"י פעולה מלאכותית שהמשתמש מבצע) בהגיע מועד התשלום, על הלווה לשלם את התשלום הנדרש לכל המלווים המקושרים להלוואה. התשלום יירד מיתרת חשבונו של הלווה, ויינתן על פי חלקו היחסי של המלווה. המידע ישוקף הן מצד הלווה והן מצידו של כל מלווה.  
יש להעביר את התשלום במלואו, או בכלל לא. (כלומר לא משלמים רק לחלק מהמלווים)  
במידה ואין מספיק כספים בחשבונו של הלווה התשלום לא יבוצע (ומידע זה כמובן מתועד ומשוקף במסגרת ההלוואה ולמלווים).

במצב זה ההלוואה עוברת לסטטוס "In Risk".  
מכאן ואלך על הלווה לשלם את מלוא החוב שנצבר. כל עוד הוא משלם רק חלק מהחוב שנצבר – ההלוואה עדיין נשארת במצב In Risk. ברגע שישלם את מלוא החוב שנצבר, עד וכולל הי"ז הנוכחי – או אז ההלוואה תחזור למצב של Active.  
כאשר הלוואה נמצאת ב In Risk הלווה יכול לשלם את החוב (ולהחזירה ל Active) כל אימת שירצה, בלי קשר למועד התשלום הפורמלי של הלוואה.  
כאשר הלוואה שולמה במלואה והסתיימה היא עוברת לסטטוס "Finished".

עיקבו אחר [נספח ב'](#appendixB) לתצוגה מסכמת של מחזור החיים של הלוואה במע'.

משתמש בכובע של לווה יכול לבחור "לסגור" הלוואה קיימת בבת אחת.   
הוא עושה זאת ע"י תשלום יתרת הקרן והריבית הנדרשת לכלל המלווים המוגדרים בהלוואה.

משתמש בכובע של מלווה יכול לבחור "למכור" את ההלוואה שהוא מחזיק. בעשותו כן הוא מציע אותה למכירה למשתמשים אחרים שישלמו את ערכה הנותר של הקרן בלבד למלווה המוכר.   
מכאן ואלך המלווה הקונה משוייך להלוואה זו במקומו של המלווה המוכר. (תהליך זה שקוף לחלוטין מצידו של הלווה)

## הנחיות ספציפיות למימוש מערכת ABS

1. **המטרה היא לבנות מנוע מערכת גנרי, כזה שידע לקבל את הפרטים לגבי אופי המע' 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. **בכל מקום שבו יש להציג חישוב שערכו יכול להיות עשרוני יש לעשות זאת עם נקודה עשרונית אחת בלבד.**
3. **המע' כולה תיכתב ותורץ בסביבת העבודה של ג'אווה 8**

תרגיל 1 – מימוש **A.B.S** כאפליקציית Console (25%) - הגשה: 9.4.22

## פרטים יבשים

צפי תחילת עבודה: **13.3.22** תאריך הגשה: **9.4.22**

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

משקל התרגיל: **20%** קושי: **סביר**

**מטרת התרגיל העיקרית**

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

## דרישות

1. בתרגיל זה תממשו את מנוע המערכת, אשר יודע לקבל נתונים על לווים ומלווים ויאפשר למשתמש את תהליך שיבוצי ההלוואות על פי הקריטריונים הנדרשים. המשתמש יוכל גם להזיז את ציר הזמן ולהניע את הכספים בין המשתמשים השונים.

את מנוע המע' תלווה שכבת UI מסוג של console application אשר תאפשר למשתמש "לדבר" עם המע' באמצעות הצגת סדרה של תפריטים ופעולות אפשריות. המשתמש יבחר פעולה לביצוע, יכניס קלט נדרש (במקרים הנדרשים) ויקבל חזרה את הפלט מהמע' וחוזר חלילה.

1. פרטיה הטכנים של המערכת יינתנו מקובץ XML (כמפורט [בנספח](#appendixC) ג').
2. 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.

בראשית כל הצגת תפריט תמיד יוצג הי"ז הנוכחי של המע' (כאמור, מתחיל מ 1).

**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'](#appendixC) 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 אכן קיים, והוא מסוג XML (די לבדוק לשם כך כי הוא נגמר בסיומת xml)
2. אין הפנייה לקטגוריה שאינה מוגדרת בקובץ
3. אין הפנייה מהלוואה ללקוח שלא הוגדר במע'
4. קצב התשלומים מתחלק בצורה מלאה עם סך אורך ההלוואה. (נתון 3 ו 4 כפי שהם מוגדרים [בנספח א'](#appendixA))
5. אין לקוחות שונים עם שמות זהים

במידה והקובץ לא תקין יש לדווח זאת למשתמש בצורה ברורה כך שניתן יהיה להבין מה לא תקין בקובץ.

אין לקרוס על exception במידה וקובץ מתגלה כאינו תקין ; יש לאפשר למע' להמשיך לפעול במצב זה.

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

במידה והקובץ נמצא תקין – יש לטעון את פרטיו למע' ולדווח על כך למשתמש.  
כל טעינת קובץ מוצלחת מאפסת את הי"ז הנוכחי חזרה ל 1. (כלומר כל קובץ מאפס את ציר הזמן).

**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 מידעים על ההלוואות הקיימות וסטטוסן

פקודה זו תציג למשתמש את המידע על כל ההלוואות הקיימות במע' (אשר נטענו במקור מקובץ ה XML).

עבור כל הלוואה יש להציג את הפרטים הבאים:

* מזהה ההלוואה
* בעל ההלוואה (שם הלווה)
* קטגורית ההלוואה
* גובה ההלוואה ; סך הזמן המקורי של ההלוואה
* ריבית ההלוואה ; קצב/מרווח התשלומים
* סטטוס ההלוואה (כפי שמוגדר בחלק הכללי)
  + אם היא pending – רשימה של כל המלווים שכבר רשומים בה.   
    עבור כל מלווה את שמו ואת ההשקעה הפרטית שלו כחלק מסכום ההלוואה.  
    יש להציג את סך הסכום שכבר גויס ע"י המלווים ואיזה סכום עוד נותר כדי להפוך הלוואה זו לפעילה.
  + אם היא active – יש להציג את כל מידעי המלווים (כפי שמוגדרים ב Pending) ובנוסף:  
    מהו הי"ז שבה היא הפכה לפעילה ; מהו הי"ז שבו ישולם התשלום הבא שלה.

מידע על כל התשלומים שבוצעו עד כה. על כל תשלום יש לפרט:

* י"ז התשלום
* רכיב הקרן
* רכיב הריבית
* סך התשלום (קרן + ריבית)

יש להציג סכומים מצטברים של סך הקרן ששולמה עד כה ; סך הריבית ששולמה עד כה ; סך הקרן שעוד נותר לשלם ; סך הריבית שעוד תשולם על ההללוואה.

* + אם היא Risk יש להציג את מידעי ה active ובנוסף עבור כל תשלום שלא בוצע (והיה אמור להתבצע) יש לסמן זאת למשתמש. יש להציג את סך התשלומים (מספר וסך הסכום) שעוקבו.
  + אם היא finished יש להציג את:
  + מידעי ה Pending
  + י"ז התחלה ; י"ז סיום
  + מידע על כל התשלומים ששולמו בפועל (כפי שמוגדר ב active)

**דגשים:**

* + - אין לאפשר פקודה זו במידה ולא נטען קובץ למע'

1. תצוגת מידע על לקוחות המע'

עבור כל לקוח יש להציג מידע על החשבון שברשותו (תנועות בחשבון). על כל תנועה:

* י"ז ביצוע
* סכום
* הכנסה או הוצאה (אפשר גם בדמות סימן + (הכנסה) או – (הוצאה) על הסכום)
* יתרה בחשבון לפני הפעולה
* יתרה בחשבון אחרי הפעולה

לקוח יכול להיות מקושר להלוואות הן כלווה והן כמלווה.   
יש להציג את פירוטי ההלוואות שהוא מקושר אליהן ב 2 רשימות שונות (בשבתו כלווה ובשבתו כמלווה).

עבור כל הלוואה יש להציג את הפרטים הבאים:

* שם ההלוואה
* קטגורית ההלוואה
* סכום ההלוואה המקורי (קרן)
* קצב/מרווח התשלומים
* ריבית עבור כל תשלום
* סכום ההלוואה הסופי (קרן + ריבית)
* סטטוס ההלוואה
  + אם הסטטוס הוא pending יש לציין איזה סכום עוד חסר (קרן) כדי להפוך את ההלוואה לפעילה
  + אם הסטטוס הוא active יש לציין מועד הי"ז הבא לתשלום + סכום התשלום הצפוי הכולל (קרן + ריבית)
  + אם הסטטוס הוא risk יש לציין את סך כל התשלומים שלא שולמו ואת ערכם הכולל (קרן + ריבית)
  + אם הסטטוס הוא finished יש לציין את י"ז ההתחלה והסיום.

**דגשים:**

* אין לאפשר פקודה זו במידה ולא נטען קובץ למע'

1. טעינת כספים לחשבון

יש לאפשר ללקוח להטעין את חשבונו הפנימי במע' בסכום כסף כרצונו. הסכום הוא מספר שלם חיובי.   
ההטענה תוצג כתנועה בחשבון.

יש להציג למשתמש את שמות כל הלקוחות. עליו לבחור לקוח מסוים ואז להכניס את הסכום המדובר ולסיים את הפקודה.

1. משיכת כספים מהחשבון

יש לאפשר ללקוח למשוך כספים מחשבונו הפנימי במע' בסכום כרצונו.   
הסכום הוא מספר חיובי החסום כמובן בגובה היתרה הנוכחית שלו.  
המשיכה תוצג כתנועה בחשבון.

יש להציג למשתמש את שמות כל הלקוחות. עליו לבחור לקוח מסוים ואז להכניס את הסכום המדובר ולסיים את הפקודה.

1. הפעלת שיבוץ

פעולה זו מבצעת את השיבוץ של הלוואות מסויימות ללקוח אחד בודד

הנה סדר הפעולות:

1. הציגו למשתמש את רשימת כל הלקוחות במע'. עבור כל לקוח הציגו את שמו ויתרת החשבון שלו.   
   המשתמש יבחר את זהות הלקוח שהוא היה רוצה לבצע את השיבוץ עבורו
2. אפשרו למשתמש לבחור את הפרמטרים הבאים לשיבוץ:  
   (בתרגיל זה תתמכו ברשימה חלקית מזו המוגדרת בחלק הכללי של התרגיל)

* סכום כולל שהוא מוכן להשקיע בסבב זה. מס' שלם החסום בגובה היתרה הפנויה בחשבונו.   
  **חובה להכניס נתון זה.**
* הציגו רשימת קטגוריות ואפשרו לבחור את כל הקטגוריות הנדרשות לשיבוץ.   
  אין חובה לבחור בנתון זה (כאילו אמר כי מעוניין בכל הקטגוריות)
* מינימום ריבית להלוואה שהוא מוכן לקבל. מספר שלם גדול מ 0.  
  אין חובה לבחור בנתון זה
* מינימום זמן י"ז כולל להלוואה. מספר שלם גדול מ 0.  
  אין חובה לבחור בנתון זה

1. תהליך השיבוץ יעבור על כל הלוואות ה Pending ו New ויבדוק האם יש הלוואות שעומדות בתנאים שהגדיר המשתמש. אין להכליל ברשימת ההלואות המדוברות הלוואות שהמשתמש מקושר אליהם כלווה. (הוא לא יכול להשקיע בעצמו)  
   בסיום התהליך יש להציג למשתמש את רשימת ההלוואות המדוברות.   
   עבור כל הלוואה יש להציג את כלל הפרטים שלה (כפי שהוגדרו בפקודה 3)
2. המשתמש יבחר את ההלוואות שהוא חפץ להשתתף בהם (יכול לבחור ביותר מאחת מהן).   
   המע' תקצה את הכספים בצורה שווה ככל הניתן בין ההלוואות השונות.  
   מרגע שהוא בחר להשתתף בהלוואה המידע צריך להיות משוקף באמצעות פקודות 2 ו 3:  
   הכספים יורדים מחשבונו אוטומטית (גם אם ההלוואה עדיין ב pending). יש להציג תנועה עצמאית עבור כל הלוואה והלוואה.  
   ייתכן שפעולה זו היא הפעולה שהופכת הלוואה מ Pending ל Active. במקרה זה יש להגדיר את הי"ז הנוכחי כי"ז התחלת ההלוואה. הכספים שנצברו להלוואה עוברים לחשבונו של הלווה (תנועה אחת בחשבונו של הלווה)

**דגשים:**

* אין לאפשר פקודה זו במידה ולא נטען קובץ למע'

1. קידום ציר הזמן ומתן תשלומים

פעולה זו תקדם את ציר הזמן בי"ז אחד בלבד.

יש להציג את הי"ז הקודם ואת הי"ז הנוכחי - לאחר הקידום.  
(דוגמא: היינו בי"ז 4 ; עכשיו הי"ז הנוכחי הוא 5.

במסגרת הקידום ייתכן ויגיע מועד התשלום של הלוואות פעילות.  
לווה יכול להיות מקושר למספר הלוואות (פעילות). בתרגיל זה התשלום מתבצע אוטומטית מבלי להיוועץ בלקוח.  
סדר התשלום להלוואות הוא על פי הי"ז שבו הן הפכו לפעילות: הלוואות קדומות יותר ישולמו לפני הלוואות מאוחרות יותר.   
אם יש כמה הלוואות שהחלו באותו הי"ז – הבחירה בסדר התשלום להן היא מהתשלום הנמוך לגבוה.   
אם יש כמה מאותו הי"ז ובאותו גובה תשלום – הבחירה היא שרירותית.  
ההלוואות לתשלום כוללות גם את ההלוואות הפעילות וגם את אלה שהם ב Risk (אין עדיפות יתרה ל Risk במובן זה)

התשלום לכל הלוואה מתבצע במלואו או לא מתבצע בכלל (אי אפשר לשלם רק חלק מהסכום הנדרש).

שימו לב כי בהחלט ייתכן כי הלווה לא יוכל לעמוד בכלל התשלומים שהוא נדרש להם.   
במקרה ויש הלוואה שלא שולמה כראוי יש לעדכן אותה לסטטוס risk.   
התשלום שלא בוצע מתווסף לתשלום הבא בתור.

(בתרגיל זה ניתן לצאת ממצב risk רק עם הגיע מועד התשלום הבא ובו הלקוח משלם את כל הסכום שעוד נותר)

**דגשים:**

* אין לאפשר פקודה זו במידה ולא נטען קובץ למע'

1. יציאה מהמע'

פקודה זו יוצאת מהמערכת.

## 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...)

ככלל, מומלץ להתחיל משלד ראשוני שמאפשר קליטת קלט מהמשתמש, העברתו למנוע המע' והחזרת התשובה למשתמש. ברגע שתעשו זאת מההתחלה במצב שבו אתם כבר עובדים עם 2 מודולים – יקל עליכם להרחיב את המע' עוד ועוד ולתמוך בפקודות הנדרשות.

התחילו בבניית התשתית המתארת את השחקנים העיקריים במע': הלוואה ולקוחות והקשרים השונים בינהם.  
החליטו באיזה מבני נתונים תשתמשו ואיך למדל את התלויות השונות בינהן.

המשיכו בעבודה על פקודות 2,3 אשר יאפשרו לכם הצגה ראשונית של המידע הקיים. המשיכו לפעולת השיבוץ, אותה תוכלו לבדוק באמצעות פקודה 2 ו 3. לבסוף עיבדו על פקודה 5 שתאפשר גם הנעה של ציר הזמן ובדיקת ביצוע התשלומים עצמם (באמצעות פקודות 2 ו 3).  
עבור כל פקודה בצעו את המסלול המלא החל מה UI, העברת המידע למנוע, קבלת התשובה והצגתה למשתמש.

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | מגניב לאללה ! | שמירת מצב מע':  בבונוס זה תוסיפו פקודה נוספת לתפריט שתאפשר למשתמש לשמור את כל מצב המע' לקובץ. פורמט הקובץ נתון לבחירתכם. עליכם לבקש מהמשתמש נתיב מלא לשם קובץ. אתם יכולם לבחור איזו סיומת שתרצו עבור הקובץ.  יש לשמור את כל המצב של המע', ובפרט את נתוני ההלוואות הנוכחיות, התנועות בחשבון המשתמשים והי"ז הנוכחי כפי שהן ברגע השמירה  בנוסף הוסיפו למשתמש פקודה נוספת שתאפשר לו בעליית המע' לבחור לטעון את המצב האחרון שנשמר (במקום לטעון קובץ XML כרגיל).  זו יכולה להיות פקודה נוספת או כתת סעיף בפקודה מס' 1 של טעינת הקובץ.  המשתמש יכניס נתיב מלא לקובץ שנשמר באמצעות הפקודה הקודמת.  לאחר הטעינה המע' חוזרת למצב הקודם בו שהתה בצורה מדוייקת, והמשתמש יוכל להמשיך לפעול על המע' בדיוק כמו מקודם. | 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 |
| 1 | 21.3.22 | האם יכולים להיות לקוחות עם שמות זהים ? מה מבדיל בינהם ? | לא. לקוח מזוהה חד חד ערכית במע' ע"י שמו. עליכם לוודא זאת גם כחלק מבדיקות הקלט שנעשות על הקובץ בפקודה 1, סעיף 5 |
| 2 | 6.4.22 | האם בפקודה 6 מינימום ריבית הוא בשקלים או אחוזים ? | הנתון הוא באחוזים. |

תרגיל 2 – מימוש **ABS** כאפליקציית JavaFX (35%) – הגשה: 4.6.22

## פרטים יבשים

צפי תחילת עבודה: **1.5.22** תאריך הגשה: **4.6.22**

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

משקל התרגיל: **35%** קושי: **מאתגר**

**מטרות התרגיל העיקרית**

1. מימוש האפליקציה כ desktop application באמצעות Java FX
2. הרחבת מנוע האלגוריתם לתמיכה בפיצ'רים נוספים

## דרישות

* 1. בתרגיל זה תממשו את המערכת כ desktop application באמצעות ארכיטקטורת JavaFX כפי שנלמד בכיתה.

בתרגיל זה אין יותר סדרה של "תפריטים" כפי שהיו בתרגיל 1. כל ממשק המשתמש הוא ממשק גרפי בלבד, המבוסס על JavaFX. כל האינטראקציה עם המשתמש מתרחשת דרך פעולותיו על גבי הפקדים השונים שתציגו לו (כפתורים, תיבות טקסט וכו').

1. העבודה במע' תתחלק בין 2 יישויות מרכזיות: Administrator (מנהל המע') ו Custumer (לקוח רגיל במע', כפי שהיה בתרגיל 1). יכולים להיות הרבה לקוחות, אולם רק admin אחד בכל רגע נתון.

2 היישויות יוכלו לבצע פעולות שונות במע'.   
משתמש הקצה (הבודק) יחווה את המע' מצד ה admin או מצד הלקוח כולל כל המגבלות והשינויים בין 2 סוגי המשתמשים.   
עיקבו אחר [נספח ג'](#appendix_graphics) המציג את האפיון הגרפי הכללי של המע'

1. המע' בעיני המנהל (Administrator):
   1. העבודה במע' תחל בטעינת קובץ XML (כשם שהיה בתרגיל 1).

את קובץ ה XML יש לטעון באמצעות FileChooser Dialog בלבד, כפי שהוצג בדוגמא המסכמת.

יש לבצע בדיקת תקינות על תוכנו של קובץ ה XML, כפי שהוגדרו בתרגיל 1.

* 1. הלוואות קיימות במע':

אפשרו לאדמין לקבל מידע על מצבן של כל הלוואות הקיימות במע'.   
המידע שצריך להיות מוצג הוא המידע כפי שמוגדר בפקודה 2 בתרגיל 1.

* 1. משתמשים קיימים במע':

אפשרו לאדמין לקבל מידע על הלקוחות הקיימים במע'. עבור כל לקוח יש להציג את המידע הבא:

* שם
* יתרה נוכחית בחשבון (אין צורך להציג תנועות)
* עבור הלקוח בכובע של לווה (ככל שזה רלבנטי), הציגו את כמות הלוואות שהוא מקושר אליהן, בחלוקה לפי סטטוס ההלוואה.
* עבור הלקוח בכובע של מלווה (ככל שזה רלבנטי), הציגו את כמות ההלוואות שהוא מקושר אליהן בחלוקה לפי סטטוס ההלוואה.
  1. אדמין יכול לקדם את ציר הזמן.  
     האדמין יוכל לקדם את ציר הזמן יחידה אחת כל פעם, קדימה בלבד.

כידוע, כאשר ציר הזמן מקודם קדימה, ייתכן והוא פוגש הלוואות שתשלומן צריך להתבצע.   
בתרגיל זה התשלום **כבר לא מתבצע בצורה אוטומטית**, אלא ניתנת האחריות ללקוחות עצמם לבצע אותם בצורה עצמאית.  
יש לייצר "הודעה" עבור הלקוחות הרלבנטים אשר תיידע אותם בדבר הצורך לבצע תשלום להלוואה הנדרשת (ראו פירוט בצד הלקוח).

האדמין יכול לבצע כמה קידומים שהוא רוצה לציר הזמן. גם בצורה רצופה. בהחלט ייתכן כי זה יגרום ללקוחות "לאבד" את היכולת לשלם בזמן (גם אם רוצים/יכולים) ובכך למעשה ייכנסו חלק מההלוואות ל risk.

1. המע' בעיני הלקוח:

בכל פעם שעובדים במע' מעיני הלקוח יש לאפשר למשתמש לבחור לקוח ספציפי והוא הלקוח שבהקשרו רואים את המידעים ו/או מבצעים את הפעולות. המשתמש יוכל לעבור בין הלקוחות השונים מתי שירצה.

פעולות שהלקוח יכול לעשות:

* 1. הצגת מידעים על ההלוואות שהוא מקושר אליהן כלווה.

הלקוח יראה אך ורק את ההלוואות השייכות לו. הציגו את כל המידע הרלבנטי עבור הלוואה.

* 1. הצגת מידעים על ההלוואות שהוא כרגע מקושר אליהן כמלווה.  
     הלקוח יראה אך ורק את ההלוואות שהוא מקושר אליהן. הציגו את כל המידע הרלבנטי עבור ההלוואה
  2. הצגת תנועות בחשבון של הלקוח, כשם שהוגדר בתרגיל 1. יש להציג גם יתרה סופית נוכחית.

יש להציג באזור זה גם את היכולת לטעון/למשוך כספים מהחשבון.

* 1. בקשה לשיבוץ הלוואות חדשות:

פקודה זו תאפשר ללקוח להפעיל את תהליך השיבוץ.

כמו בתרגיל 1, יש לבקש מהלקוח את הפרמטרים הרלבנטים על פיהם יש להציע לו הלוואות מתאימות.

בתרגיל זה נתמוך בכל הפרמטרים האפשריים (כפי שהוגדרו בחלק הכללי):

* סכום כולל שהוא מוכן להשקיע בסבב זה. מס' שלם החסום בגובה היתרה הפנויה בחשבונו. נתון חובה.
* הציגו רשימת קטגוריות ואפשרו לבחור את כל הקטגוריות הנדרשות לשיבוץ. נתון אופציונלי.
* מינימום ריבית להלוואה שהוא מוכן לקבל. מספר שלם גדול מ 0. נתון אופציונלי.
* מינימום י"ז כולל להלוואה. מס' שלם גדול מ 0. נתון אופציונלי.
* מקסימום הלוואות פתוחות ללווה. מס' שלם. נתון זה בודק כמה הלוואות כבר יש ללווה. במניין ההלוואות יש לספור את כל ההלוואות המקושרות ללווה בסטטוס של new, pending, active, in risk. (כלומר הלוואות שלא נגמרו).  
  יש להציע רק הלוואות המקושרות למלווים שלהם מספר הלוואות פתוחות הנמוך (או שווה) למספר המבוקש.   
  זהו נתון אופציונלי.
* מקסימום אחוז בעלות על הלוואה. מס' שלם בין 1 ל 100 המתפקד כאחוז.  
  נתון זה אומר שהמלווה מעוניין שחלקו בסך ההלוואה לא יעלה על הנתון המסופק. נתון אופציונלי.  
  שימו לב זהו נתון שלא משתתף בסינון ההלוואות המתאימות כי אם רלבנטי רק לאופן פיזור וחלוקת הכספים בין ההלוואות הנבחרות (ככל שיהיו כאלה)

בסיום הכנסת הנתונים המע' תבצע את השיבוץ הנדרש ותציג את רשימת ההלוואות שעומדות בקריטריונים.   
תהליך השיבוץ יתבצע באמצעות task של java fx ברקע. יש להעביר למשתמש חיווי על התקדמות התהליך.   
(**שימו לב**: התהליך עצמו צפוי להיות מהיר ביותר ועל כן הוסיפו השהייה מלאכותית אשר תדמה כאילו הוא לוקח מעט יותר זמן מהנדרש...)

המשתמש ייבחר באילו הלוואות הוא רוצה להשתתף ומרגע שהוא מאשר המע' מעכלת את הנתונים ופועלת כשם שפעלה במהלך תרגיל 1.

* 1. תשלום על הלוואות:

הלקוח יקבל בצורה ברורה ובאזור מוגדר מידע על כלל התשלומים של ההלוואות הנדרשים לביצוע.

במידה והלקוח לקח הלוואה, עליו לקבל הודעות מסודרות לגבי הגעת מועד התשלום.   
ההודעות נשלחות ללקוחות הרלבנטים ברגע שהאדמין מקדם את ציר הזמן.   
על כל הודעה להכיל את:

* שם ההלוואה הרלבנטי
* י"ז התשלום
* גובה התשלום הסופי הנדרש (קרן + ריבית).

הלקוח יכול להחליט האם הוא רוצה לשלם, למי ובאיזה סדר.   
אפשרו לו לבצע את התשלום הנדרש עבור הלוואה מסויימת.  
ברגע שיתקדם ציר הזמן מעבר למועד התשלום (והלקוח לא יודע לצפות מתי זה יקרה) – ההלוואות הרלבנטיות שלא שולמו במועדן נכנסות למצב RISK ויש לשקף מידע זה ללקוח (כחלק מפקודה 1 ו 2).

עבור הלוואות הנמצאות ב in risk, יש לאפשר למשתמש לשלם את מלוא החוב או חלקו, ובכל י"ז שיחפץ בכך.   
כל עוד לא שולם מלוא החוב – ההלוואה עדיין נמצאת ב In risk.  
שימו לב כי ניתן לשלם חלק מהחוב, גם אם הוא לא חופף בהכרח לגובה התשלום המקורי.   
דוגמא: אם יש חוב של 200 ש"ח שנצבר בגלל פיגור ב 2 תשלומים של 100 ש"ח כל אחד, יש לאפשר למשתמש לשלם 150 ש"ח, אם ירצה, ואז ההלוואה עדיין נשארת ב In risk עם שארית חוב של 50 ש"ח.)

* 1. סגירת הלוואה במכה אחת:

אפשרו ללקוח הלווה להחליט כי הוא רוצה לסגור הלוואה פעילה (active\in risk) כאן ועכשיו ובמלואה. במקרה זה הלקוח נדרש לשלם את יתרת חוב ההלוואה (קרן + ריבית) בבת אחת. כמובן שיש לאפשר לו פעולה זו רק אם יש לו יתרה מספקת בחשבון. בעשותו כן ההלוואה מסתיימת (finished) וכל המלווים מקבלים את כלל הכספים הנדרשים בבת אחת לחשבונם.

1. כחלק מבדיקת המערכת 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.

**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 בונוס כזה או אחר אין הכרח לגרום להם לפעול גם בתרגיל זה, אולם אם זה מתאפשר זה יחמם את ליבי. (למען הסר ספק, חימום ליבי אינו מתורגם לחימום ליבו של הבודק ו/או להעלאת נקודות).

## חלוקה למודולים

בתרגיל זה יש לממש את מודול ה Java FX כ jar עצמאי שעובד מול מודול מנוע המערכת (שגם הוא משתכלל כפוקנציה של הדרישות השונות).

מומלץ לפתח את המודול של JavaFX באותו פרוייקט ב intelij, כמודול נוסף ליד מודול מנוע המערכת.

שימו לב: **אין צורך** לגרום למודול הקונסול של תרגיל 1 להמשיך לעבוד גם עם המנוע של תרגיל 2.   
כלומר, אם במהלך תרגיל 2 אתם צריכים לשנות את הקוד של המנוע כך ששכבת ה UI של תרגיל 1 לא תוכל לעבוד יותר – זה בסדר !

## 

## איך מתחילים ? (המלצה...)

קיראו את [נספח ג'](#appendix_graphics) כדי להבין את קווי המתאר והמבנה הכללי של המע'.

התחילו במסך האדמין. דרכו תוכלו לטעון קובץ ולהתחיל להתעסק עם הצגת המידע הכללי של ההלוואות והלקוחות. כך ייקל עליכם להמשיך למסכים הבאים תוך שהרכיבים הבסיסיים כבר קיימים ועובדים.  
עבור הגדרת כל רכיב (שמציג פרטי הלוואות ; שמציג פרטי חשבון ; שמאפשר שיבוץ וכו') – צרו fxml, css ו controller משלו (עדיף

לאחר מסך האדמין התחילו עם מסך הלקוח: מידע. שם תשאפו ותוכלו לבחון את עצמכם כמה קל לייצר אותו באמצעות מרבית הרכיבים שכבר יצרתם כחלק מהעבודה על מסך האדמין.   
לאחר מכן ממשו את תהליך פעולת השיבוץ (בהתחלה בלי TASK. כשהכל יעבוד כמו שצריך – הוסיפו את התמיכה ב TASK), ורק בסוף התרכזו בתהליך ביצוע התשלומים ומכירת ההלוואה.

## 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 | פשוט מגניב ! | אנימציות  ממשו לפחות 2 סוגי אנימציות לבחירתכם, כפי שנלמד בכיתה. תעדו בקובץ ה README איזה אנימציות מימשתן, והיכן צפוי המשתמש להיתקל בהן.  האנימציה צריכה להתבצע לכל היותר במשך 2 שניות  יש לאפשר גם לנטרל את האנימציות הנ"ל, כך שהיא לא תאיט את התקדמות המערכת | כי זה מה-זה קל... | 5 נקודות  (ועד ל 100) |
| 2 | הגדלת ראש נאה | החלפת skin למע'  בבונוס זה תממשו (לפחות) 2 ערכות צבעים נוספות על הערכה הדיפולטית, ותאפשרו למשתמש להחליף את ערכות הצבעים.  אני לא מגדיר במדויק מה אמור להשתנות במסגרת המעבר מ skin ל skin – אבל דאגו לבצע שינוי מהותי שלא יוכל לחמוק מעינו של המשתמש הסביר (הכי טוב: ביחנו זאת על הוריכם) | כי זה בכל זאת תרגיל בממשק גרפי – ומה יותר גרפי מזה ??? | **5 נקודות**  **(ועד ל 100)** |
| 3 | הקדמת תרופה למכה | קנייה/מכירה של הלוואה  ממשו כבר עכשיו את היכולת לקנות/למכור הלוואה. הפרטים מוצגים בתרגיל 3 כחלק מהגדרת הפעולות של אפליקציית הלקוח. | מה-זה בקטנה... וחוסך לכם עבודה מתרגיל 3... | 5 נקודות  (ועד ל 100) |
| 4 | מגניב למדי ! | הצגת ציר זמן עתידי: הציגו את ציר הזמן בצורה גרפית, בצורה אופקית כקו עם שנתות עליו. בכל שנתה שקרתה בעבר ניתן לראות את התשלומים שהועברו עבור ההלוואות הרלבנטיות.  אפשרו למשתמש לראות כמות זמן עתידית והציגו לו גם את צפי התשלומים שהוא עתיד לקבל בהמשך הדרך.  ציר הזמן צריך להופיע עבור כל לקוח (בהקשרו) ואצל האדמין להראות מצב כולל של כלל התשלומים שבוצעו (או עתידים להיות מבוצעים) במע', על פני כלל הלקוחות | זה המקום להיות יצירתיים ולהמציא משהו יש מאין ! | 10 נקודות **(מעל ל 100)** |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. בתרגיל זה ניתן להגיש Jar אחד המהווה את כל הקוד שלכם, בצירוף קובץ אצווה (batch) שהפעלתו תריץ את התוכנית   
   מי שרוצה להמשיך לעבוד עם יותר מ Jar אחד, כשם שהיה בתרגיל 1 יותר ממוזמן לעשות זאת.
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 **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 3 – מימוש **ABS**כאפליקציית Client - Server (35%) – הגשה: 10.7.22

## פרטים יבשים

צפי תחילת עבודה: **5.6.22** תאריך הגשה: **10.7.22**

צפי זמן לביצוע: **5 שבועות** ציון אפשרי מקסימלי: **100**

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

**מטרות התרגיל העיקרית**

1. מימוש המערכת כאפליקציית client-server.
2. הגדרת משתמשים דינמית
3. הגדרת הלוואות דינמית

## דרישות

1. בתרגיל זה נשדרג את ABS לעבוד בצורה מבוזרת בארכיטקטורת שרת-לקוח.  
   השרת יכיל את מנוע המע' עצמה ושני סוגי המשתמשים (administrator ו customer) יתחברו אל המע' בדמות קליינטים עצמאיים.

הפעם מי שמעלה קבצים למע' הם הלקוחות, המתארים את ההלוואות שהם מגישים למע'. האדמין יוכל לקבל את כלל המידע על המתרחש בה, כולל ניהול ציר הזמן. הלקוחות, מנגד, יקבלו את המידעים הפרטיים שלהם ויוכלו להנות ממנעמי המע'.

1. הן האדמין והן הלקוח ("הקליינטים" של השרת) יוצגו בדמות אפליקציות java fx עצמאית (אפלקיציה לכל אחד מהם). התקשורת בינן לבין השרת תתבצע באמצעות http client בדומה לדוגמא המסכמת.   
   ברגע נתון יכול להתחבר למע' (אפליקציית) אדמין אחד בלבד ומספר רב של (אפליקציות) לקוחות.

ABS  
Server  
)Tomcat)

Customer  
(Client)

Customer  
(Client)

**...**

**HTTP**

**HTTP**

Admin  
(Client)

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

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

## אפליקציית לקוח (Customer)

האפליקציה תיפתח במסך לוגין שבו יידרש המשתמש לבחור לעצמו שם. (אין לאפשר רישום כפול תחת שם שכבר קיים במע'.)  
לאחר מסך הלוגין המשתמש יעבור למסך הבא – מסך ה dashboard.

* העלאת קובץ לשרת:  
  בתרגיל זה מי שמעלה את הקובץ לשרת הוא הלקוח עצמו. (ולא האדמין)

הקובץ יכיל את פרטי ההלוואות שהלקוח הזה מעוניין להוסיף למע' כבעליהן.

קבצי ה XML יהיו מוגדרים על פי סכמה חדשה של תרגיל 3   
(ראו [נספח XML](#appendixC_1) כדי להבין מה התוספות/הבדלים יחסית לסכמה של תרגיל 1 ו 2)

בהגיעו לשרת, הקובץ יעבור בדיקת תקינות (כל הבדיקות הרלבנטיות כפי שהוגדרו בתרגיל 1).

במידה והקובץ נמצא תקין, הודעה מתאימה תוחזר ללקוח ומעכשיו האדמין במע' והלקוח הספציפי בלבד (אשר העלה את הקובץ) יכולים לראות את הפרטים הנ"ל. כמו כן ההלוואות האלה נכנסות לסך ההלוואות שיותאמו בתהליך השיבוץ.

במידה ונמצא תקול, הודעה מתאימה תוחזר ללקוח שהעלה אותו בלבד והקובץ (והמידע שבו) כמובן לא נכנס למע'.

כל לקוח יכול להעלות קובץ. לקוח יכול להעלות יותר מקובץ אחד. כל קובץ מכיל גם תיאור של הקטגוריות שנצברות מקובץ לקובץ (כל קטגוריה נבדלת על פי שמה)

* הלקוח יוכל להגיש ולייצר הלוואה חדשה במע' באמצעות ממשק המשתמש. אפשרו ללקוח להכניס את כל המידעים הנדרשים להגדרת הלוואה (כפי שקיבלתם מידעים אלה מקבצי ה XML בתרגילים האחרונים) ומרגע שההלוואה נכנסה היא נחשבת הלוואה במע' כמו כל שאר ההלוואות.
* הלקוח יוכל גם כאן לראות את רשימת כל ההלוואות שהוא מעורב בהן (הן כלווה והן כמלווה), על כלל פרטיהן (כפי שהיה בתרגיל 2).
* הלקוח יוכל גם כאן לראות ולקבל את רשימת התנועות שלו בחשבון, כמו גם את היכולת למשוך/לטעון כספים מ/להחשבון (כמו שהיה בתרגיל 2).   
  שימו לב כי יתרת החשבון של כל לקוח מתחילה מ 0. (בניגוד לתרגילים קודמים שבהם המידע הגיע מקובץ ה XML)
* הלקוח צריך לקבל את רשימת ההודעות (notifications) המתריעות בפניו כי הגיע מועד התשלום להלוואה.   
  כמו בתרגיל 2 אפשרו ללקוח להחליט לבד האם/איזה הלוואות הוא בוחר לשלם.  
  כמו כן, הלקוח עדיין יוכל להחליט שהוא סוגר את הלוואה במלואה (כמו בתרגיל 2)
* הלקוח יוכל לבקש מהמע' לבצע עבורו תהליך שיבוץ הלוואות.   
  כמו בתרגיל 2 איספו את כל הנתונים הרלבנטים, ושילחו אותם לשרת לטובת ביצוע החישוב. כאן החישוב יבוצע ע"י השרת (אין צורך יותר ב task של java fx שמדמה את התהליך) והתשובה החוזרת תהיה רשימה של הלוואות מתאימות העומדות בהגדרות הלקוח.   
  הלקוח יבחר איזה הלוואות הוא מעוניין להצטרף אליהן והמע' תפעל בהתאם (כפי שהיה בתרגיל 2)
* קניית/מכירת הלוואה

בתרגיל זה הלקוח המלווה יכול להחליט כי הוא מעוניין למכור את חלקו הנותר בהלוואה פעילה בלבד (Active).  
המכירה תתבצע כך שמחשבים את יתרת הקרן בלבד שעוד נותרה לתשלום וזהו מחיר ההלוואה.   
הלקוח מעמיד את ההלוואה למכירה ומחכה לראום אם לקוחות אחרים ירצו לקנותה.   
כל עוד לא קנו אותה – ההלוואה עדיין נחשבת שלו לכל עניין ודבר. אם תוך כדי העמדתה למכירה פתאום ההלוואה הופכת ל In Risk – יש להסיר את ההלוואה ממועמדותה למכירה ולספק ללקוח התראה בנושא.

אפשרו ללקוחות לראות את כל ההלוואות העומדות למכירה ולהחליט כי הם רוצים לקנות את מי מהן.   
ברגע שמתבצעת הקנייה, הכסף יורד מחשבון הלקוח הקונה ועובר לחשבון הלקוח המוכר.   
החל מרגע זה הלקוח הקונה הוא המקושר להלוואה זו.

**שימו לב:** הלקוח הלווה לא מודע ואינו מהווה צד בתהליך.

* יש להציג את הי"ז הנוכחי של המע' בראש האפליקציה כך שיהיה חשוף וידוע ללקוח בכל רגע נתון

## אפליקציית מנהל (Admin)

האפליקציה תיפתח במסך לוגין שבו יידרש המשתמש לבחור לעצמו שם.   
אין לאפשר רישום כפול תחת שם שכבר קיים במע'.  
במידה ויש כבר אדמין שמחובר למע' יש למנוע ממופע נוסף של אדמין להירשם גם כן.

לאחר מסך הלוגין המשתמש יעבור למסך הבא – מסך ה dashboard.

יש להציג את רשימת כל הלקוחות במע'. עבור כל לקוח יש להציג את שמו ואת יתרת החשבון הנוכחית שלו.   
(קלאסי ל all the information all the time)

פעולות שהמשתמש (admin) יוכל לבצע:

1. כמו בתרגיל 2, האדמין יוכל לראות את כלל המידעים על ההלוואות הקיימות במע' וסטטוסן השונה
2. כמו בתרגיל 2, האדמין יוכל לשלוט בציר הזמן ולקדם את הי"ז קדימה יחידה אחת.
3. בחינת העבר:  
   בתרגיל זה האדמין יוכל גם להזיז את ציר הזמן לאחור !  
   הזזת ציר הזמן לאחור נועדה לתת לאדמין יכולת לבוא ולבחון מה היה מצב המע' בכל י"ז וי"ז.   
   כדי להזיז את ציר הזמן לאחור, על ה admin להכניס את המע' למצב "REWIND". רק כשהמע' במצב זה ניתן "לשוטט בעבר" ולראות את כל פרטי המע' כפי שהיא נראתה בכל י"ז.   
     
   העובדה שהמע' במצב REWIND מועברת גם לאפליקציות הלקוחות המחוברות כרגע ולמעשה במצב זה הן עוברות למצב READONLY: הלקוח לא יכול לבצע שום פעולה אקטיבית משנה מצב (למשל: העלאת קבצים, ביצוע שיבוץ, תשלום, ביצוע תנועות בחשבון וכו'), אלא רק לראות את המידעים השונים של המע'.   
   כשסיים את נבירתו בעבר, האדמין מסיים את מצב ה REWIND והמע' חוזרת חזרה לי"ז הנוכחי (כולל כמובן כל שיקוף המידעים המתאימים ושידור העובדה שה REWIND נגמר לאפליקציות הלקוחות).  
     
   כאמור, במהלך שהות במצב REWIND לא ניתן לבצע שום פעולה המשנה את המצב של המע', אלא רק לראות איך נראתה המע' בכל אחד מהי"ז הנוכחי. האדמין אם כן יוכל לדפדף אחורה עד לי"ז מס' 1 וקדימה עד לי"ז הנוכחי. בכל מעבר לי"ז מסויים – המע' מציגה את מלוא המידעים כפי שהם היו בי"ז זה במקור.   
   אפליקציות הלקוח מתעדכנות בי"ז שכרגע מוצג במע' וכמובן מושכות ומשקפות את המידע כנדרש. אם מדובר על לקוח שהצטרף בי"ז X והאדמין מניע את המע' לי"ז קטן מ X (י"ז שבו במקור הלקוח בכלל לא היה קיים) – הרי שאז הלקוח פשוט לא רואה שום מידעים. (לא סוגרים לו את האפליקציה או משהו כזה)

## 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 2 מודולים נוספים (חדשים) שמהם ייבנו שתי האפליקציות ה client (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.

לאחר מכן התחילו בעבודה על פי סדר העמודים:

ראשית התחילו במסך הלוגין. הוא יכול להיות זהה בעיקרו (בצד השרת) לסרבלט הלוגין כפי שהוצג בדוגמא המסכמת (הרגישו חופשי "להעתיק" משם את המידע הנדרש לכם. מתנה ממני אליכם. כזה אני. גדוש בטוב.)

תו"כ כתיבת עמוד זה תתרגלו את שלב ה UI ל שרת וחזרה ל UI.

בהמשך התרכזו בבניית אפליקציית הלקוח, אשר תהיה זהה בעיקרה וברכיבים שלה לאופן שבו מומשה המע' מעיני הלקוח בתרגיל 2. ההבדל היחיד הוא העובדה שצריך עכשיו להעביר את כל המידעים ולקבל תשובות חזרה באמצעות עבודה עם http client. השלימו גם את חלקה של אפליקציית האדמין. גם כאן היא תהיה זהה מאוד למה שהיה בתרגיל 2. (שימו לב כי נושא העלאת הקובץ "עובר" מהאדמין ללקוח.)

עכשיו, כשהכל כבר עובד ממשו את היכולות החדשות של מכירת/קניית ההלוואה, יצירת הלוואה חדשה ע"י ממשק המשתמש ורק בסוף התרכזו בסוגיית הזזת ציר הזמן לאחור.

לאורך פיתוח 2 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 2 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 2 האפלקיציות (וייכלל כמובן ב 2 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 (וכמובן יקדם אתכם בדרך לבונוס מס' 2).

## 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 | מתנה, ממני אליכם | Postman Collection:  עליכם לייצר Postman Collection שיאפשר למשתמש העצמאי (זה שלא רוצה לעבוד עם קליינט גרפי) לעבוד גם כן מול השרת שלכם.  על ה collection שלכם להיות מתועד היטיב בקובץ ה readme (כולל דוגמאות הפעלה) ועליו להכיל את היכולות הבאות:   * אפשרו למשתמש לייצר הלוואה חדשה * אפשרו למשתמש לבקש לבצע שיבוץ תוך העברת הפרמטרים הנדרשים התשובה היא אוסף של אפשרויות הלוואה * אפשרו למשתמש לבחור איזו הלוואה הוא רוצה להיכנס אליה. | כדי שתוכלו לדבר על זה בראיון עבודה, כמובן... | 5 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. 2 ספריות עבור הקליינטים השונים: admin ו customer. 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. אם ביצעתם את בונוס מס' 2, יש לספק גם Postman Collection שניתן יהיה לייבא ולהפעיל אל מול השרת שלכם בכתובת localhost:8080
4. קבצי קוד המקור של הפרויקט שלכם (גם צד השרת וגם אפליקציות java fx של ה clients).
5. 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.
6. 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...)
7. 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' – דוגמא למתן הלוואות ופריסת תשלומים

נניח כי במע' מתקיימים 2 הלוואות:

|  |  |  |  |
| --- | --- | --- | --- |
| # | רכיב | הלוואה 1 | הלוואה 2 |
| 1 | קרן | 120 | 300 |
| 2 | ריבית (%) | 10 | 5 |
| 3 | סך משך זמן ההלוואה (בי"ז) | 12 | 15 |
| 4 | מרווח התשלומים (משלמת כל י"ז) | 1 | 5 |
| 5 | תשלום (קרן) עבור י"ז בודד | 10 | 20 |
| 6 | תשלום בודד (קרן) | 10 | 100 |

בהינתן הפרטים על הלוואה (1 - 4) כל הלוואה תחשב:

* מרכיב הקרן עבור י"ז בודד (5): גובה הקרן (1) לחלק לסך משך זמן ההלוואה (3)
* תשלום בודד (6): תשלום הקרן עבור י"ז בודד (5) כפול מרווח התשלומים (4)

הנה טבלת פריסת תשלומי ההלוואה כפונקציה של ציר הזמן.   
על כל הלוואה ניתן לראות את מרכיב קרן, הריבית וסך התשלום.   
כמו כן ניתן לראות את המועדים שבהם משולמת כל הלוואה.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| י"ז | הלוואה 1 | | | הלוואה 2 | | |
| קרן | ריבית | סה"כ | קרן | ריבית | סה"כ |
| 1 | 10 | 1 | 11 | לא משלם | | |
| 2 | 10 | 1 | 11 | לא משלם | | |
| 3 | 10 | 1 | 11 | לא משלם | | |
| 4 | 10 | 1 | 11 | לא משלם | | |
| 5 | 10 | 1 | 11 | 100 | 5 | 105 |
| 6 | 10 | 1 | 11 | לא משלם | | |
| 7 | 10 | 1 | 11 | לא משלם | | |
| 8 | 10 | 1 | 11 | לא משלם | | |
| 9 | 10 | 1 | 11 | לא משלם | | |
| 10 | 10 | 1 | 11 | 100 | 5 | 105 |
| 11 | 10 | 1 | 11 | לא משלם | | |
| 12 | 10 | 1 | 11 | לא משלם | | |
| 13 |  |  |  | לא משלם | | |
| 14 |  |  |  | לא משלם | | |
| 15 |  |  |  | 100 | 5 | 105 |
| סה"כ | 120 | 12 | 132 | 300 | 15 | 315 |

נספח ב' – תיאור מהלך החיים של הלוואה

הנה תרשים מצבים המתאר את מהלך החיים של הלוואה במע':Diagram, schematic

Description automatically generated

נספח ג' – מבנה אפליקציה גרפית

על האפליקציה להכיל מבנה המתאים ל 2 סוגים של משתמשים:  
אדמין ולקוחות. יכול להיות אדמין אחד והרבה לקוחות.

מבנה האפליקציה משתנה בהתאם לסוג המשתמש (אדמין או לקוח) ולכן ראשית יש לבצע בחירה בהקשרו של איזה משתמש מתפעלם את המע'. הבחירה תתבצע באמצעות combo box שבו יוצג בראש האדמין ואחריו שמות כל הלקוחות המוגדרים במע'.

בכל רגע נתון יש להציג לכל סוגי המשתמשים את הי"ז הנוכחי ואת מסלול הקובץ הטעון במע' (ככל שיש כזה)

מידע על הלוואות ותנועות בחשבון הגיוני מאוד להציג באמצעות רכיב טבלאי כזה או אחר. ממליץ ללמוד ולהכיר את tableview ולעבוד איתו. שימו לב כי יש הרבה מקומות שצריך להציג מידע על הלוואות (אולי כל פעם בהקשר אחר או עם פרטים מעט שונים) ולכן יהיה הגיוני ביותר לייצר רכיב שיודע להציג הלוואות כרצונכם, לכתוב אותו פעם אחת ולשלב אותו במקומות השונים באפליקציה.

הנה סקיצה כללית ושל קווי המתאר של האפליקציה.   
שימו לב אתם **מחוייבים** למבנה הנ"ל. (מוזמנים לפנות אלי במייל אם יש הצעות/השגות למבנה אחר)

**מסך ראשי מעיני האדמין:**

**Graphical user interface, text, application, email

Description automatically generated**

דגשים:

* רק לאדמין יש יכולת לטעון קובץ ולקדם את הי"ז במע'.
* אפשר להציג את המידעים של ההלוואות ופרטי הלקוחות גם בצורה אופקית, אם תמצאו את זה מתאים
* אפשר להוסיף עוד פרטים נדרשים למסך האדמין ככל שתצטרכו.

**מסך ראשי מעיני הלקוח: מידע**

**Graphical user interface, application

Description automatically generated**

דגשים:

* האזור של account transactions נועד להציג גם את מידע התנועות בחשבון.
* אפשר לסדר את 3 האזורים גם בצורה אנכית אם תרצו, או לחילופין להשתמש ב splitter כדי להקצות מקום דינמי בין אזורי ההלוואות לאיזור פירוט החשבון

**מסך ראשי מעיני הלקוח: שיבוץ**

**Graphical user interface

Description automatically generated with medium confidence**

הותרתי לכם יד חופשית בניסוח מבנה המסך הזה. הוא צריך להכיל את כל הנדרש כדי לבצע את תהליך השיבוץ במלואו.

**מסך ראשי מעיני הלקוח: תשלומים**

**Application

Description automatically generated**

דגשים:

* Loaner lons table – מציג את כלל הללואות של הלקוח (בכובעו כלווה) אשר נמצאות בסטטוס active או risk. אלו ההלוואות שהלקוח יוכל לשלם בהן תשלום בודד (אם הגיע הי"ז המתאים להן) או לסגור את כולם במכה אחת אם יחפץ
* ב Payment controls צריך לאפשר ללקוח לבצע תשלום עבור מי מההלוואה שנבחרה בטבלת ההלוואות הממתינות לתשלום
* ב payment controls צריך לאפשר ללקוח גם "לסגור" הלוואה בבת אחת
* אזור ה notification צובר את ההודועת השונות שמקבל הלקוח לגבי התשלומים הנדרשים. אין למחוק הודעות !

נספח ד' – תיאור מבנה המערכת באמצעות קובץ XML

תיאור מבנה המערכת מפורט כחלק מקובץ ה XML

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

ביחנו 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)

## סכמת קובץ XML – גרסה I

Text

Description automatically generatedראשית הקובץ מתארת מבנה של ABS

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | abs-descriptor | מתאר את ראשית קובץ ה XML |
| 2 | Element | abs-categories | מבנה המתאר את כל הקטגוריות האפשריות |
| 3 | Element | abs-loans | מבנה המתאר את כל ההלוואות |
| 4 | Element | abs-customers | מבנה המתאר את כל המשתמשים |

**abs-categories**

Shape

Description automatically generated with low confidence

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | abs-category | מכיל את שם הקטגוריה. השם הוא מחרוזת באנגלית. יכולה להכיל רווחים. שם הקטגוריה הוא case insensitive. אפשר להוסיף כמה אלמנטים כאלה שרוצים. |

**abs-loans**

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | abs-loan | מבנה המתאר הלוואה אחת במע'. אפשר לייצר כמה מבנים כאלה שרוצים. |
| 2 | Attribute | id | מתאר את שם ההלוואה. השם הוא ייחודי. טקטס חופשי שיכול להכיל רווחים. השם הוא case insensitive. |
| 3 | Element | abs-owner | אלמנט זה מתאר את שם המשתמש שמעוניין בהלוואה זו |
| 4 | Element | abs-category | אלמנט זה מתאר את הקטגוריה אליה משתייכת ההלוואה |
| 5 | Element | abs-capital | אלמנט זה מתאר את גודל ההלוואה (הקרן). מס' חיובי ושלם. |
| 6 | Element | abs-total-yaz-time | אלמנט זה מתאר את משך הזמן הכולל של ההלוואה. מס' חיובי ושלם. |
| 7 | Element | abs-pays-every-yaz | אלמנט זה מתאר את קצב התשלומים של ההלוואה. כל כמה י"ז היא משולמת. מס' חיובי ושלם. |
| 8 | Element | abs-intrist-per-payment | אלמנט זה מתאר את גובה הריבית שישא כל תשלום. מס' חיובי ושלם. |

**Text

Description automatically generated**

Text

Description automatically generated**abs-customers**

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | abs-customer | מבנה המכיל הגדרה של לקוח אחד במע'. אפשר להגדיר כמה כאלה שרוצים. |
| 2 | Attribute | name | מתאר את שם המשתמש. טקסט חופשי. יכול להכיל רווחים. השם הוא case insensitive. |
| 3 | Element | abs-balance | מתאר את יתרת החשבון שהלקוח מתחיל איתה. מס' חיובי שלם. |

## סכמת קובץ XML – גרסה III

עדכונים מגרסה קודמת:

1. באלמנט abs-loan אין יותר את את האלמנט abs-owner
2. לא מופיע בקובץ כל האלמנט של abs-customers

נספח ה' – קבצי סכמה לוידוא קובץ ה XML

קובץ ה XML מקושר לסכמה שמגדירה את חלקיו השונים ואת חוקיותם.

להלן תרשימים המתארים את מבנה הסכמות בגרסאותיהם השונות:

### סכמה – גרסה 1

Diagram

Description automatically generated

### סכמה – גרסה 2

Diagram

Description automatically generated