מס' גרסה:9

פיתוח תוכנה מבוסס java

Stepper אביב 2023 -

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](#_Toc127804460)

[כללי 4](#_Toc127804461)

[איך להגיש תרגילים באיחור, ולהישאר בחיים 5](#_Toc127804462)

[הנחיות כלליות לכתיבת התרגיל 6](#_Toc127804463)

[תרגיל reflection – תרגיל עצמאי ביחידים (5%) - הגשה: 28.3.23 8](#_Toc127804464)

[כללי 8](#_Toc127804465)

[מבנה התרגיל 8](#_Toc127804466)

[ניקוד 8](#_Toc127804467)

[איך בודקים ? 9](#_Toc127804468)

[מה מגישים ? 9](#_Toc127804469)

[מקרה בדיקה לדוגמא 9](#_Toc127804470)

[**Stepper** 11](#_Toc127804471)

[מטרת התרגיל(ים) בקורס 11](#_Toc127804472)

[מוטיבציה 11](#_Toc127804473)

[Stepper – תיאור המע' 11](#_Toc127804474)

[הגדרת Flow 13](#_Toc127804475)

[ביצוע Flow 14](#_Toc127804476)

[הנחיות ספציפיות למימוש תרגיל Stepper 15](#_Toc127804477)

[תרגיל 1 – מימוש **STEPPER** כאפליקציית Console (25%) - הגשה: 1.5.23 16](#_Toc127804478)

[פרטים יבשים 16](#_Toc127804479)

[דרישות 16](#_Toc127804480)

[חלוקה למודולים 21](#_Toc127804481)

[איך מתחילים ? (המלצה...) 22](#_Toc127804482)

[בונוסים 22](#_Toc127804483)

[סבבה, סיימתי. מה ואיך להגיש ? 22](#_Toc127804484)

[שאלות ותשובות 23](#_Toc127804485)

[תרגיל 2 – מימוש **Stepper** כאפליקציית JavaFX (35%) – הגשה: 14.6.23 24](#_Toc127804486)

[פרטים יבשים 24](#_Toc127804487)

[דרישות 24](#_Toc127804488)

[איך מתחילים ? 28](#_Toc127804489)

[בונוסים 29](#_Toc127804490)

[סבבה, סיימתי. מה ואיך להגיש ? 29](#_Toc127804491)

[שאלות ותשובות 30](#_Toc127804492)

[תרגיל 3 – מימוש **Stepper**כאפליקציית Client - Server (35%) – הגשה: 23.7.23 31](#_Toc127804493)

[פרטים יבשים 31](#_Toc127804494)

[דרישות 31](#_Toc127804495)

[אפליקציית Administrator 34](#_Toc127804496)

[אפליקציית משתמש 35](#_Toc127804497)

[חלוקה למודולים 35](#_Toc127804498)

[איך מתחילים ? 35](#_Toc127804499)

[בונוסים 36](#_Toc127804500)

[סבבה, סיימתי. מה ואיך להגיש ? 36](#_Toc127804501)

[שאלות ותשובות 36](#_Toc127804502)

[נספח א' – פירוט ה Data Definitions 37](#_Toc127804503)

[נספח ב' – פירוט ה Steps 39](#_Toc127804504)

[מבזבז הזמן 40](#_Toc127804505)

[סורק הקבצים 41](#_Toc127804506)

[מוחק הקבצים 42](#_Toc127804507)

[משנה השמות 43](#_Toc127804508)

[מחלץ המידעים I 45](#_Toc127804509)

[פולט הפורמטים: CSV 46](#_Toc127804510)

[פולט הפורמטים: Properties 47](#_Toc127804511)

[כותב הקובץ 48](#_Toc127804512)

[רוכסן 49](#_Toc127804513)

[שורת הפקודות 50](#_Toc127804514)

[משגר הבקשות 51](#_Toc127804515)

[המג'ייסן 53](#_Toc127804516)

[מחלץ המידעים II 54](#_Toc127804517)

[נספח ג' – דיאגרמות של flow (דוגמא להמחשה) 55](#_Toc127804518)

[נספח ד' – תיאור מבנה המערכת באמצעות קובץ XML 58](#_Toc127804519)

[סכמה תרגיל 2 60](#_Toc127804520)

[נספח ה' – תרשים סכמות XML 62](#_Toc127804521)

[סכמה תרגיל 1 62](#_Toc127804522)

[סכמה תרגיל 2 63](#_Toc127804523)

דרישות הקורס

## כללי

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

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

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

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

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

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

הניקוד על כל תרגיל נזקף לזכות הסטודנטים שבצעו אותו בלבד.

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

## איך להגיש תרגילים באיחור, ולהישאר בחיים

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

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

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

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

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

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

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

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

## הנחיות כלליות לכתיבת התרגיל

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

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

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

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

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

חיסכו ממני (ומכם) את כתיבת המייל המבקש זאת.

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

הקובץ יכיל:

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

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

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

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

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

1. כל הקלט והפלט בתרגילים השונים יהיה באנגלית בלבד.

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

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

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

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

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

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

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

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

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

תרגיל reflection – תרגיל עצמאי ביחידים (5%) - הגשה: 28.3.23

## כללי

**מועד הגשה: 28.3.2023 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();  
 }  
}

Stepper

## מטרת התרגיל(ים) בקורס

חלק א' – מימוש Stepper בסיסי; תפעול באמצעות ממשק console.

חלק ב' – העשרת Stepper ביכולות נוספות כמו גם פיתוח ותפעול באמצעות ממשק גרפי מבוסס Java FX.

חלק ג' – הפיכת Stepper לשרת המאפשר גישה מרחוק למס' סוגי משתמשים. המע' תדע לנהל משתמשים, תפקידים והרשאות ולאפשר לכלל המשתמשים להשתמש ולהגות flow'ים משלהם.

## מוטיבציה

כאשר מייצרים מוצר תוכנה ומשחררים אותו החוצה לשוק הלקוחות יש להקים מערך תמיכה ועזרה שיודע לעזור ולתפעל את המוצר בעת פעולתו. (העבודה לא נגמרת רק במלאכת הקידוד.. ☹ )

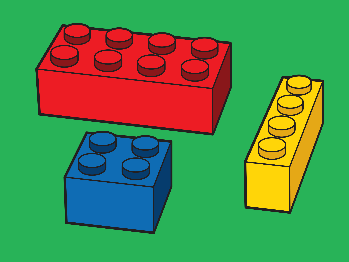
עולם האופרציה והתמיכה של מוצרי תוכנה הוא מורכב ביותר. אנשי התמיכה בדרגים השונים נדרשים לבצע סט של פעולות דינמיות כדי לחקור, לתקן ולסייע ללקוחות בפתרון בעיות המתגלות במוצר.  
אנשי התמיכה והאופרציה אינם בהכרח מתכנתים או בעלי ידע טכני, אולם הפעולות אותן הן נדרשים לבצע נושאות אופי טכני ברובן (לחלץ מידע מקבצים, להריץ בקשות מול שרתים, לשלוף מידע מתהליכים וכו')  
  
הסיטואציה הנ"ל מעלה מספר בעיות:

* **טעויות אנוש**: אנשי התמיכה והאופרציה הם אחרי הכל... אנשים. ככאלה הם נוטים לטעות (בתום לב) פה ושם. ככל שהפעולה מורכבת ומסובכת יותר, דורשת יותר תשומת לב וירידה לפרטים, יותר מידעים שצריך ללקט ולהכניס במקומות הנכונים או לנתח מפלטים מסובכים (ומתוסבכים) – כך גדל הסיכוי לטעות שיכולה לעלות בפתרון המקרה ובמרמור הלקוח.
* **סיבוך**: חלק ניכר מהבעיות דורשות ידע והבנה טכנית. לעיתים נדרשות יכולות תכנות כאלה ואחרות כדי להוציא מידע או להעביר מידע ממקום למקום. הרבה פעמים הפתרון ניתן בדמות כתיבה של סקריפט בשפה כזו או אחרת שאחד מהאנשים הכיר. הסקריפט מכיל לרוב פקודות ברמת מע' ההפעלה והרבה פעמים נופל במהלך הביצוע בשל כתיבה לקויה או מקרי קצה שלא חשבו עליו.
* **חזרתיות**: חלק ניכר מהבעיות דומות "כמעט" לחלוטין לבעיות אחרות. הרבה פעמים יש צורך למחזר ולהשתמש בפתרונות שכבר נכתבו ונהגו בעבר עבור הבעיה החדשה, "עם שינוי" קל לטובת ההתאמה לבעיה החדשה.
* **תסכול**: הרבה פעמים הפעולה שבוצעה עבור לקוח מסויים נדרשת לביצוע עבור מקרים/לקוחות רבים אחרים. ככל שהפעולה מורכבת וארוכה יותר – כך מסובך (ובעיקר מתסכל !) להריץ אותה שוב ושוב ושוב מול כלל הלקוחות/המקרים המטופלים.
* **אחידות**: כל אחד מוצא את הדרך שלו לארגן את המידע, להציג אותו, להעביר אותו לאחר. כשעובדים בהרבה פורמטים שונים, מתחיל להיווצר בלבול בינהם ולא כולם מכירים או בקיאים בפורמטים של אחרים וכו'.
* **העברת הידע**: קשה מאוד להעביר את הידע והנסיון הטכני שנצבר אצל אנשי התמיכה לעובד חדש שמגיע לארגון ו/או לשמר את המידע במידה ועובד וותיק ומנוסה נעלם כדי "לחפש הזדמנויות אחרות" בחיים...

כלל המצוקות הנ"ל (ועוד רבות אחרות אשר קצרה היריעה מלהכיל) היוו את ההשראה והבסיס ליצירת Stepper.

## Stepper – תיאור המע'

Stepper נועדה להיות מע' גנרית שתאפשר הגדרה, הרצה ותפעול של תהליכים שונים ע"י אנשים שאינם טכנים, תוך שימוש בשפה אחידה שכולם יכולים להבין. המע' תאפשר שימוש חוזר ברכיבי התפעול השונים כמו גם להחיל את אותו הפתרון כמה פעמים עבור מקרים רבים.

כדי להבין יותר לעומק את המבנה של Stepper אפשר להשתמש באנלוגיה הבאה:

כולם מכירים (ואוהבים !) לשחק ב Lego.  
LEGO מבוסס על אבני פלסטיק (לרוב) בעלות דרכים שונות לחבר בינהן. הדרך המוכרת ביותר היא באמצעות פינים על חלק אחד (נקראים studs) הנכנסים לשקעים מתאימים על חלק אחר.  
יש הרבה סוגים של אבני לגו. כל אחת משרתת פונקציונליות אחרת, אבל כמעט כל אבן יכולה להתחבר לכמעט כל אבן אחרת.   
בהינתן סט (אנסופי לעניין זה) של אבני לגו, יכול כל אחד לקחת ולהרכיב ולבנות מהן מודל מסויים.   
אוסף של אבנים חוברות יחדיו לכדי הקמה של מודל מסויים, ואפשר להשתמש באותן האבנים (עם/בלי תוספת של אבנים אחרות) כדי לבנות מודלים אחרים.  
כך, מסט סופי של "סוגי" אבנים ומסט ("אנסופי") של מופעים של סוגי האבנים ניתן להרכיב כל מודל ויצירה שרוצים.

Stepper מגדיר מודל עבודה דומה:

בבסיסו עומדת הגדרת ה Step (כמובן..).   
Step מהווה פיסת לוגיקה בודדת ועצמאית, קטנה וממוקדת ככל האפשר (למשל מחיקת קובץ; הפעלת פקודה בודדת; בדיקה אם ספריה קיימת וכו'). Step היא המקבילה לסוג אחד של אבן לגו.  
כדי לבצע את הפעולה הטכנית נדרשים לספק לה קלטים שונים (Inputs) וסביר כי לכל step שכזה יהיה גם פלט אחד או יותר (outputs). ה Inputs וה outputs משולים לפינים ולשקעים השונים באבני הלגו ומהווים את הממשק ודרך החיבור ל Step'ים אחרים.  
(במסגרת המע', ייתכנו גם input'ים שהם אופציונלים ל step – כלומר כאלה שלא חייבים להינתן ויש להם ברירת מחדל במידה ולא ניתנו או קיבלו ערך בדרך כזו או אחרת).

ה input וה Output השונים מהווים למעשה פיסות מידע שהמע' צריכה להניע ולהעביר ממקום למקום. הרכיב הבא שנגדיר אם כן הוא הגדרת מידע (Data Definition).   
כל פיסת מידע מוגדרת באמצעות **סוג** המגדיר את יכולותיה ומאפייניה, ושם ייחודי שמאפשר לפנות ולהתייחס אליה (לקבל ממנה מידע או לעדכן באמצעותה).

בהינתן אוסף של Step'ים, אשר כל אחד מוגדר עם Input ו output משלו, ניתן לבצע "חיבור" בין ה step'ים השונים כך ש output של אחד ישמש כ input של האחר. לאוסף הstep'ים המחוברים יחדיו כך נקרא flow.   
ה flow משול למודל שניתן לבנות מאוסף של אבני הלגו.

אפשר לקחת את אותן הגדרות ה step'ים השונות ולשלב אותן בדרכים שונות כדי לייצר flow'ים (מודלים) שונים של עבודה.   
כל flow שכזה מהווה אוסף של פעולות שמתאימות לביצוע בתרחיש מסויים.  
בהינתן flow, המע' תאפשר "להפעיל" אותו (execution), כלומר להריץ את כל ה step'ים שבמסגרתו. ביצוע הלוגיקה המוגדרת בכל ה step'ים, תוך העברת המידעים בינהם מהווה את הפתרון הפעולה שלשמה הגדרנו את ה flow.

## הגדרת Flow

הגדרת ה flow תינתן באמצעות קובץ xml (עבודה עם xml'ים תילמד במהלך הקורס).  
ההגדרה תכיל את שמות ה step'ים המשתתפים ב flow. יש חשיבות רבה לסדר הופעת ה step'ים. (ראו בהמשך).  
  
כל Flow יכול להגדיר את מי מה output'ים המופקים ע"י ה step'ים שבמסגרתו כ output הפורמלי של המע'.   
הגדרה זו תעזור למקד את המשתמשים השונים ב"תכלס" שהם רוצים להשיג ולהפיק מהרצת ה flow.

הנה כמה מאפיינים של Flow, שיש לתת עליהם את הדעת:  
readonly

כל step יכיל ערך המתאר האם הוא readonly או לא.  
Step נחשב ל readonly אם הוא מבצע רק פעולות של קריאה של מידעים ממקורות שונים ובכל מקרה לא מתיימר לבצע ו/או להחיל שינוי במע'.  
אם flow מכיל לפחות step אחד שאינו readonly – אזי כל ה flow נחשב כ לא-readonly.

automatic mapping  
במסגרת הגדרת ה flow, מע' ה Stepper תדע לבצע התאמה אוטומטית של ה ה output'ים ל Input'ים השונים בצורה אוטומטית. ההתאמה האוטומטית תתבצע על פי הכללים הבאים:

* Output של step x יכול להיות מוצמד ל input של step y אך ורק אם step y מופיע אחרי step x
* Output יכול להיות מוצמד ל input רק אם יש להם שם זהה ואם הם מאותו הסוג (2 התנאים גם יחד)
* Output אחד יכול להיות מוצמד ליותר מ input אחד.
* ל Input אחד יכול להיות מוצמד output אחד לכל היותר
* ייתכנו output'ים שאינם מוצמדים לשום input'ים. הם ייקראו output'ים חופשיים.
* ייתכנו input'ים שאינם מוצמדים לשום output'ים. הם ייקראו input'ים חופשיים.

בסיום ביצוע ההתאמה האוטומטית, כל ה input'ים החופשיים (כלומר כאלה שלא מוצמד להם שום output) נחשבים Mandatory Inputs והם הופכים להיות ה input'ים אותם יידרש המשתמש להכניס בעת הרצת ה flow (יתואר בהמשך). ייתכנו גם optional inputs – כאלה שה step'ים יודעים להסתדר גם אם לא יתקבל עבורם ערך.  
במידה ויש כמה mandatory inputs בעלי אותו השם והסוג – הם מתאחדים לכדי הצהרה אחת של mandatory input (למשל כמה step'ים המגדירים FILE NAME כ input – המשתמש יכניס FILE NAME פעם אחת בלבד.)  
כל ה output'ים החופשיים (כלומר כאלה שלא מוצמדים לשום input) יהוו עוד פלטים שיופקו במסגרת הרצת ה flow.  
  
custom mapping  
לעיתים יש צורך בהתערבות ידנית בכל הקשור לחיבור וטווית הקשרים בין ה output'ים ל input'ים השונים. התערבות זו תיקרא custom mapping ותאפשר למשתמש המגדיר את ה flow להחליט איזה output של איזה step מקושר לאיזה input של איזה step שיבחר. גם כאן יש כמובן לוודא כי יש התאמה של סוג בין ה Output ל input.

### Aliasing

לעיתים ייתכן ונרצה להשתמש במסגרת ה flow באותו step כמה וכמה פעמים. היות וכך יהיה צורך להבדיל בין ה step'ים השונים מבחינת השם שלהם. יכולת זו תיקרא step aliasing.   
  
בעקבות step aliasing (אך לא רק !), ייתכן ובמסגרת ה flow נקבל כמה output'ים "כפולים" בשמם.   
כדי לבצע הבדלה בינהם נצטרך לכנות אותם בשמות אחרים, רק במסגרת ה flow הזה. יכולת זו תיקרא flow level aliasing.  
  
בכל מקרה ברגע שניתן aliasing (אם ל step ואם ל data) – מכאן ואלך ובכל מקום שבו צריך להתייחס ל step\data הרלוונטי במרכיבים השונים יש להשתמש ב alias בלבד.

Initial Values  
לעיתים ייתכנו מצבים שבהם אנו יודעים או נידרש להגדיר ערך של mandatory input כבר במועד הגדרת ה Flow. כלומר מדובר בערך קבוע וידוע מראש. המע' תאפשר להגדיר ולפתור מצבים אלה ע"י בחירת ה data הרלוונטי והצהרת ערכו.

### Continuation

במסגרת הרצת flow (ראו בהמשך) לפעמים נוח וקל להשתמש במידעים ובערכים שהופקו במסגרת הרצת flow קודם. יכולת זו מאפשרת חיבור מיידי ומהיר בין הרצת flow אחד לאחר.

## ביצוע Flow

לאחר ש Flow הוגדר ובהצלחה, המשתמש ירצה להריץ ולבצע אותו – execution.  
כדי לעשות זאת המשתמש יוכל לבחור את ה flow הרלוונטי והמע' תדע להציג לו מיידית מי הם כל ה mandatory inputs המוגדרים במסגרתו אותם הוא מחוייב להכניס כדי להפעיל את ה flow.  
רק בסיום הכנסת כל ה Input'ים הנדרשים יש לאפשר לבצע את הרצת ה flow.  
  
ההרצה מתבצעת תוך מעבר על ה step'ים על פי הסדר שבו הם הוגדרו והפעלת הלוגיקה של כל step ו step. כל ה output'ים שהופקו במסגרת הפעלת הלוגיקה של step נשמרים באזור משותף לכלל ה step'ים ולמעשה זמינים לשימוש step'ים אחרים כמו גם למשתמש הקצה שהריץ את ה flow (ייתכן והוא מתעניין גם ב intermediate values ולא רק ב formal outputs של ה flow).

במסגרת הרצה של כל step ייתכן והוא יצטרך לפלוט תיאור של מעשיו או הסבר על פועלו (log'ים).   
המע' תאפשר לצבור את המידעים הנ"ל ולחשוף אותם למשתמש הקצה.  
לכל step יש שורת סיכום (summary line) – מחרוזת המתארת בשורה אחת את טיב פועלו.

כאשר מריצים כל step ייתכן (ובשאיפה) שהוא יצליח להגשים את פועלו ולהיגמר בהצלחה, אולם ייתכן כי הוא ייכשל.   
אם כן, לכל step נגדיר את אפשרויות הסיום הבאות:

* Success – ה step עבר בהצלחה. שורת הסיכום תציין זאת בגאווה ! (ואולי עוד מידעים רלוונטים על פי ההקשר)
* Warning – ה step הסתיים בהצלחה, אולם קרה משהו במהלכו שמחייב התייחסות של המשתמש. שורת הסיכום תכיל את הסיבה ל warning
* Failure – ה Step נכשל בגלל סיבה מסויימת. שורת הסיכום תכיל את תמצית הסיבה לכשלון

במידה ומריצים flow ואחד ה step'ים מסתיים עם Failure – כל הרצת ה flow נעצרת ונחדלת מיידית**\*\*** וגם ה flow נגמר בכשלון.  
מצבי הסיום של ה flow אם כן הם:

* Finished Success – כל ה flow, על כלל ה step'ים שלו, נגמרו ועברו בהצלחה.
* Finished with Warning – יש step אחד לפחות (ייתכנו אף יותר) שנגמרו ב warning.
* Finished Failure – ה step האחרון שבוצע נגמר ב failure ולכן כל הביצוע נעצר. (ייתכנו step'ים שבכלל לא הגיעו לבצעם)

**\*\***במסגרת הרצה של step'ים, ניתן להגדיר כי במידה ו step מסויים נכשל, בהקשר ה flow הזה בלבד, אין להכשיל ולעצור את כל ה flow אלא להמשיך הלאה.

בגמר ההרצה משתמש הקצה יוכל לקבל את כלל המידעים של ההרצה המדוברת ובכללם:

* איך נגמר ה flow: האם בהצלחה או בכשלון ? ואם בכשלון מה הייתה הסיבה ?
* מידעי זמנים: כמה זמן לקח לבצע את כלל ה flow; כמה זמן לקח לבצע כל step
* כלל המידעים שנוצרו/נצברו במהלך ההפעלה
* כלל הלוגים של כל ה step'ים שבוצעו; שורות הסיכום של כל step

## הנחיות ספציפיות למימוש תרגיל Stepper

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

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

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

תרגיל 1 – מימוש **STEPPER** כאפליקציית Console (25%) - הגשה: 1.5.23

## פרטים יבשים

צפי תחילת עבודה: **2.4.23** תאריך הגשה: **1.5.23**

צפי זמן לביצוע: +**3 שבועות** ציון אפשרי מקסימלי: 105

משקל התרגיל: **25%** קושי: **סביר**

**מטרת התרגיל העיקרית**

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

## דרישות

1. בתרגיל זה תקימו את תשתית העבודה הראשונית והבסיסית של Stepper.   
   את התשתית "תתפעלו" באמצעות ממשק console פשוט המציג תפריט פקודות שדרכו מפעילים את המע'.  
   בתרגיל תממשו מספר step'ים בסיסיים, כמו גם את מנוע הרצת ה flow'ים אשר ידע לקבל נתונים על ה flow'ים המוגדרים ולהגיב לכל פנייה שמגיעה משכבת ממשק המשתמש, לעבד את הקלט ולהחזיר פלט רלבנטי.
2. במסגרת תרגיל זה תצטרכו לפתח את הפיצ'רים הבאים של מערכת ה stepper:

* מימוש step'ים והגדרות המידעים (Data Definitions) כפי שמוגדרים בנספח א' וב' והמשוייכים לתרגיל 1
* טעינת מידע על flow'ים קיימים מקובץ XML (כמפורט [בנספח ד'](#appendix_D_XML)).
* תמיכה ב data & step aliasing, custom & automatic mapping
* תמיכה ביכולת להגדרת step'ים שכשלונם לא מכשיל את הרצת ה flow המלא (מתואר גם במסגרת פקודה 4)

1. 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.

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. במסגרת תרגיל זה עליכם לממש חלק מה data definitions וה step'ים.   
   ההגדרות בקובץ ה XML של מבנה ה Flow'ים השונים יתייחסו לשמות ה step'ים ויצפו לקבל את ה input\output השונים כפי שמוגדר בהגדרות ה step'ים.

עיקבו אחר [נספח א'](#appendix_A_DD) ו[נספח ב'](#appendix_B_STEPS) כדי להבין מהן ההגדרות של ה data definitions וה step'ים הנדרשים למימוש.   
יש לממש (לכל הפחות) את הרכיבים הנדרשים לתרגיל 1 (מוזמנים להגדיל ראש ולהקדים תרופה למכה ולממש גם את ה data וה step'ים הבאים בתור, אלה הרלוונטים לתרגיל 2...)

1. בניית הגדרת Flow:  
   הגדרות ה FLOW היבשות יינתנו מקובץ ה XML (ראו פרטים בהמשך).   
   הגדרות אלה הן הגדרות יבשות המתארות את רצון המשתמש. עליכם לפענח אותם ולבנות מהם מודל אופרטיבי שמתאר את ה Flow על כלל מרכיביו. זה גם יהיה התהליך שבמסגרתו תוכלו לזהות אם הגדרת ה Flow תקינה או אם לאו.  
     
   בתהליך בניית ה Flow, עליכם לשים לב לכללים ולדגשים הבאים:
2. לכל step צריך שיהיה שם סופי שניתן במסגרת השתתפותו ב Flow. כברירת מחדל – זהו פשוט שם ה Step.   
   במידה וניתן Step aliasing – מכאן ואלך השם הסופי יהיה ה alias שקיבל ה step, ובו ישתמשו בכל שאר המבנים שדורשים ציון שם ה step.
3. לכל input\output המשתתף ב Flow (במסגרת הגדרות ה step'ים) צריך שיהיה שם סופי. כברירת מחדל זהו פשוט שם ה input\output כפי שהוגדר במסגרת ה Step.   
   במידה וניתן flow level aliasing עבור Input\output מסויים – מכאן ואלך השם הסופי של ה input\output יהיה ה alias שניתן.  
   במידה וניתן flow level aliasing, כל שאר המבנים שבהם מתייחסים לשמות של מידעים יינתנו בשם ה alias. (למשל: custom output, flowOutput)
4. במסגרת הגדרת ה FLOW יש לבצע את תהליך המיפוי במסגרתו יחוברו output'ים ל input'ים.   
   יש להחיל automatic mapping ו custom mapping (במידה וסופק).  
   **שימו לב**: בכל מקרה של התנגשות – ה Custom mapping הוא הקובע.
5. לאחר ביצוע תהליך המיפוי, כל ה Input'ים החופשיים מוגדרים כ mandatory inputs. ייתכנו כמה Input'ים עם אותו השם (למשל כמה STEP'ים דורשים input בשם folder name) – במקרה זה כל ה input'ים הזהים מאוחדים לכדי input אחד "שיוזרם" לכלל היעדים.  
   ייתכנו בינהם optional inputs – כאלה שאין חובה למלא אותם. בכל Step מוגדר עבור Input'ים אלה מה לעשות במידה והם לא ניתנים.  
   אם יש כמה Input'ים באותו השם והסוג, וחלק מהם מוגדרים Mandatory וחלק מהם מוגדרים optional – זה תקין – זה אומר שבהכרח יתקבל עבורם ערך כחלק מהרצת ה flow והם לא יישארו ריקים
6. לכל input\output מוגדרת מחרוזת (user string) שבה יש להציג את ערכו למשתמש. (השם המקורי של ה Input\output סביר שיהיה סתום ולא ברור למשתמש הקצה).   
   במידה ויש התנגשות בין ה User strings של כמה mandatory inputs שהתאחדו להם – יש לבחור שרירותית את אחד מה user strings.
7. עליכם לכתוב ממשק משתמש בתצורת console.

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

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

**שימו לב**:

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

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 ד'](#appendix_D_XML) לתרגיל זה.   
אתם מצופים לעבור על פרטים אלה ולהתייחס אליהם כחלק מהגדרת התרגיל.

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

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

בפרט יש לוודא את הפרטים הבאים:

1. הקובץ קיים, והוא מסוג XML (די לבדוק לשם כך כי הוא נגמר בסיומת .xml)
2. אין Flow'ים בעלי שמות כפולים (Flow אמור להיות ייחודי על פי שמו)
3. אין הפניה ל step שלא קיים. (מוגדר על פי שמו)
4. יש לוודא כי הגדרת ה flow'ים תקינים.  
   Flow נחשב לא תקין אם:
   1. יש יותר מ output אחד באותו השם הסופי
   2. יש mandatory inputs שלמשתמש אין גישה אליהם. (user friendly = false)
   3. בעיות שונות בהגדרת custom mapping:

* הפנייה ל step/data שלא קיים במסגרת הגדרת ה Flow
* הפניה מ step מאוחר ל step קדום
* נסיון לחבר בין סוגים שונים של מידעים
  1. הגדרת flow aliasing ל step\data שלא קיים במסגרת הגדרת ה Flow
  2. Flow output מכיל פרט מידע שלא קיים
  3. יש כמה free Inputs בעלי אותו שם אך מסוגים שונים

במידה והקובץ לא תקין יש לדווח זאת למשתמש בצורה ברורה כך שניתן יהיה להבין מה לא תקין בקובץ.   
במידה ויש אפילו flow אחד שאינו תקין – כל הקובץ נחשב לא תקין.

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 את כל מופעי ה flow'ים ההנגזרים ממנו.   
יש לדווח על כך שהקובץ נמצא תקין ונטען במלואו למשתמש.

**דגשים:**

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

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

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

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

1. הצגת הגדרת Flow

פקודה זו תאפשר להציג למשתמש מידע מסודר לגבי ה flow'ים המוגדרים בקובץ.  
יש להציג למשתמש רשימה של כל ה flow'ים המוגדרים במע'. לכל flow יש להציג רק את השם שלו.  
המשתמש יבחר לגבי איזה flow הוא רוצה לקבל פרטים מקיפים (ע"י בחירת המספר של האפשרות של ה Flow הרלוונטי). יש לאפשר להקיש 0 כדי לחזור חזרה לתפריט הראשי מבלי לקבל מידע מקיף על flow מסויים.  
  
בבחירת flow מסויים יש להציג את פרטי המידע הבאים:

1. שם ה flow
2. תיאור ה flow
3. רשימת ה formal output של ה flow
4. האם ה flow הוא read only או לא (מספיק שיש step אחד שאינו read only – וכל ה flow מוגדר ככזה שאינו readonly)
5. ה step'ים הנכללים ב flow. יש להציג אותם על פי סדר הופעתם. עבור כל step:
   * + 1. שם ה Step. במידה ויש step aliasing יש להציג את השם המקורי של ה step יחד עם ה aliasing שלו.
       2. יש לציין עבור כל step האם הוא read only או לא.
6. מי הם כל ה input'ים החופשיים   
   עבור כל input יש לציין:
7. שם סופי (אחרי aliasing, במידה וניתן)
8. סוג
9. רשימת שמות ה step'ים (שם ה alias במידה וניתן) שהם קשורים אליהם
10. האם זהו Input שהוא mandatory או optional
11. רשימת כל ה Outputs המופקים במסגרת ה flow.   
    עבור כל output יש לציין:
12. שם סופי (אחרי aliasing, במידה וניתן)
13. סוג
14. שם ה step (שם ה alias במידה וניתן) אשר הפיק אותו.
15. הפעלת flow (Execution)

פקודה זו תאפשר להפעיל flow אחד במע'.

יש להציג למשתמש רשימה של כל ה flow'ים המוגדרים במע'. לכל flow יש להציג רק את השם שלו.  
המשתמש יבחר איזה flow הוא רוצה להפעיל ע"י בחירת המספר של האפשרות של ה Flow הרלוונטי.   
יש לאפשר להקיש 0 כדי לחזור חזרה לתפריט הראשי מבלי לקבל מידע מקיף על flow מסויים.  
  
לאחר בחירת ה flow יש להציג למשתמש את כל ה input'ים החופשיים במע'. לכל Input יש לציין:

* + 1. שם ה Input. השם יוצג באמצעות ה user string המשוייך לו (במסגרת הגדרת ה Input ב step) ולידו, עטוף בסוגריים, את השם הפנימי של ה Input כפי שהוא הוגדר במסגרת ה step/flow בהתאם ל alias.   
       (ראו [כאן](https://mama.mta.ac.il/mod/forum/discuss.php?d=10143#p14422) הסבר מקיף)
    2. סוג ה Input: mandatory או optional

המשתמש ייבחר את ה Input שהוא מעוניין לעדכן את ערכו ויכניס את הערך הנדרש.  
תהליך הצגת ה Input'ים חוזר על עצמו עד אשר המשתמש בוחר להמשיך להפעלת ה flow.   
יש לאפשר לו לבחור אפשרות זו אך ורק כאשר כלל ה mandatory inputs התקבלו.  
ייתכן שהוא ירצה להמשיך ולעדכן Inputs גם בשלב זה: או לעדכן שוב ערך של input שכבר הוכנס ו/או לעדכן optional inputs.  
כמו כן יש להציע למשתמש אפשרות לוותר על כל התהליך ולחזור חזרה לתפריט הראשי.  
  
מהרגע שבחר להפעיל את ה flow ובאם זה מתאפשר – המע' תעבור ותבצע את כל ה step'ים על פי הסדר, עד אשר כולם ייגמרו בהצלחה או עד אשר תיתקל ב step שנגמר בכשלון.  
בזמן הזה המשתמש "ממתין" לתוצאות שיגיעו ואינו רואה כלום (אין צורך לבצע פלט איטרטיבי).

לכל הרצה של flow תוגדר ותיווצר מחרוזת זהות ייחודית אותה עליכם להפיק במהלך ההפעלה. באמצעות מחרוזת ייחודית זו ניתן יהיה לאחזר את תוצאות ההרצה של Flow מהעבר. תוכן המחרוזת נתון לבחירתכם. (אני ממליץ על [UUID](https://www.uuidgenerator.net/dev-corner/java))

כחלק מהפעלת ה flow וכלל ה step'ים שלו יש לתעד את זמן ההתחלה וזמן סיום הביצוע שלו, כמו גם של כל אחד מה step'ים. עבור כל step (ואחרי כן ברמת ה Flow) יש לחשב את משך הזמן שהם פעלו (ב ms). אני ממליץ לקרוא ולעבוד עם Instance ו Duration)  
  
כל step פולט לוגים במסגרת פעולתו. יש לשמר מידע זה במסגרת ההפעלה לטובת תצוגה עתידית (ראו פקודה הבאה).

כעקרון במידה והפעלת step מסויים נכשלת, כל ה flow עוצר. אולם תיתכן הגדרה של חלק מה step'ים ככאלה שאם הם נכשלים אין לעצור את כל ה flow בגינם, אלא להמשיך לעיבוד ה step'ים הבאים.   
הגדרה זו מגיעה במסגרת הגדרת ה flow ויש להתייחס אליה כחלק ממנגנון ההרצה

בגמר הפעלת ה flow יש להציג את המידעים הבאים:

1. מחרוזת זהות ייחודית להפעלה זו של ה flow
2. שם ה flow שהורץ
3. תוצאת הרצת ה Flow הסופית (SUCCESS \ WARNING \ FAILURE)
4. תצוגת המידע של ה flow formal output(s), על פי ה user presentation המוגדר לכל סוג של data, עם כותרת של user string לפני כן.  
   במידה וה flow נגמר בכישלון, אזי לא בהכרח כל ה (formal) output יהיו קיימים. במקרה זה יש להדפיס עבורם את ה user string שלהם, אולם ערכם יהיה השורה הקבועה הזו: Not created due to failure in flow
5. הצגת פרטים מלאים של הפעלת עבר

פקודה זו תאפשר לקבל פרטי הרצה של הרצת-עבר של אחד ה Flow'ים.  
יש להציג למשתמש את רשימת כלל הפעלות ה FLOW'ים שבוצעו.   
על הרשימה להיות ממוינת כך שההרצה האחרונה תופיע ראשונה בסדר.

עבור כל הרצה יש להציג:

1. שם ה FLOW שהורץ
2. מחרוזת ייחודית להפעלה
3. מועד ההפעלה (חותמת זמן בלבד: HH:MM:SS)

המשתמש יבחר את ההרצה עליה הוא רוצה לקבל פרטים מלאים.   
הנה הפרטים שיש להציג:

1. מחרוזת זהות ייחודית להפעלה זו של ה flow
2. שם ה flow שהורץ
3. תוצאת הרצת ה FLOW הסופית ( WARNING\ SUCCESS \FAILURE )
4. סה"כ זמן הרצת ה FLOW. יוצג ב ms. (ייתכן ויהיה 0).
5. הצגת המידע של כל ה input'ים החופשיים שהוזנו ל Flow מלכתחילה

עבור כל פרט מידע:

* 1. שם סופי (אחרי aliasing, במידה ובוצע)
  2. סוג
  3. תוכן
  4. האם הוא mandatory או optional

יש להציג את הרשימה כך שכל ה mandatory יופיעו ראשונים לפני כל ה optional (במידה וקיימים)

1. הצגת המידע של כלל ה output'ים שהופקו במסגרת ה Flow, כלומר כל ה output'ים של כל ה step'ים (לא רק ה formal outputs).   
   עבור כל פרט מידע:
   1. שם סופי (אחרי aliasing, במידה ובוצע)
   2. סוג
   3. תוכן (במידה ולא קיים כי ה flow נכשל, יש להציג Not created due to failure in flow, כשם שהוגדר בפקודה 3)
   4. שם ה step שבמסגרתו הם הופקו (אחרי aliasing במידה וקיים)
2. הצגת המידע של הפעלות ה step'ים השונים:
   1. שם ה STEP (כולל alias, במידה וקיים)
   2. משך עבודת ה STEP. יוצג ב ms. (ייתכן ויהיה 0).
   3. תוצאת סיום ה step (SUCCESS, WARNING, FAILURE)
   4. שורת סיכום ה step
   5. לוגים שנצברו בזמן עבודת ה step.   
      הלוגים יוצגו על פי סדר היווצרותם (כלומר האחרון שנוצר יופיע אחרון ברשימה)  
      עבור כל לוג יש להציג:
   6. חותמת זמן שבה ניתן הלוג (HH:MM:SS.sss)
   7. תוכן שורת הלוג.
3. סטטיסטיקות

פקודה זו תאפשר לחלץ סטטיסטיקות על הפעלות ה flow'ים שקרו במע' עד עכשיו (במסגרת הקובץ התקין האחרון שנטען).

עבור כל FLOW יש להציג:

1. כמות פעמים שהורץ
2. ממוצע זמני ההפעלה שלו. יינתן ב ms. (יכול להיות 0)

עבור כל STEP יש להציג:

1. כמות הפעמים שהורץ
2. ממוצע זמני ההפעלה שלו. יינתן ב ms. (יכול להיות 0)
3. יציאה מהמערכת

פקודה זו מסיימת את פעולת התוכנית.

## חלוקה למודולים

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

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

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

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

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

שימו לב שמודול זה "פסיבי", והוא **רק** מגיב לבקשות ולפקודות המתקבלות ממקורות בלתי ידועים לו (בתרגיל זה מודול #1). בתרגילים הבאים מקורות נוספים יפנו אליו לקבלת מידע וחשוב מאוד להקפיד על כך **שמודול זה אינו מכיר/מודע למי פונה אליו.**

## איך מתחילים ? (המלצה...)

התחילו מהבנת מבנה ופעולת המע' בצורה מוחלטת. שחקו את זה על הנייר והגדירו כמה flow'ים משלכם כדי להבין איך הם מתחברים ומוגדרים עם כל מיני מקרים ותגובות.

צרו פרוייקט חדש ב Intellij שישמש כפרוייקט האב לכלל התרגילים.   
בתוך הפרוייקט צרו מודול נפרד עבור מנוע המע'. המנוע יכיל את כלל החלקים הנדרשים לתפעול המע' (הגדרה של flow ושל step, הגדרה של אובייקט ההפעלה של flow וכו').  
המנוע יחשוף סט של יכולות (הפקודות השונות בתפריט), יחזיק מופע של המע' ויתווך את הפקודות השונות מולה, הלוך וחזור. התחילו בבניית תשתית המנוע (מומלץ לעטוף את המנוע בממשק ולהתחיל לתרגל (ולהתרגל) לחווית העבודה הנ"ל.)  
  
התחילו במימוש פקודה 1 (על כלל היביטי הטעינה), ופקודה 2 שתאפשר לכם לראות כי קיבלתם את המידע כנדרש.   
עיברו לממש את פקודות הפעלת ה flow (3 ו 4) ורק בסוף התעסקו עם סוגיית ההסטוריה והסטטיסטיקה.

לסיום, צרו מודול נוסף שהוא מודול שכבת ה UI (console). זה המודול שבו תשב בסופו של דבר מטודת ה main הראשית שתתפעל את כלל המע'. זה המקום היחיד שבו מוצג פלט (System.out.pritnln) ונאסף קלט (scanner) מהמשתמש. זה המקום המכיל את לולאת התפריט הראשית המניעה את כלל המע'. שכבת ה UI תכיל הפנייה (reference) למופע המנוע (שבתוכו מכיל הפנייה למופע המע') 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 המע'.  בבונוס זה המצב הנוכחי של המע' (כולל כל ההרצות שכבר בוצעו וה flow'ים המוגדרים) נשמר לקובץ חיצוני (באיזה פורמט וטכניקה שתבחרו).   יש להוסיף פקודה שמאפשרת לשמור את מצב המע' וגם פקודה המאפשרת לטעון מצב קיים (מקובץ שנשמר זה לא מכבר), וזאת בניגוד לטעינה רגילה מקובץ ה XML של התרגיל.  יש לאפשר למשתמש לבחור את הנתיב המלא כולל שם הקובץ (בלי הסיומת) שהוא היה מעוניין לשמור את המע' אליו (ולטעון אותה ממנו) | 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 | 8.4.23 | במידה ויש כמה mandatory inputs בעלי אותו שם ו user string שונה – איזה מה user string לבחור לטובת הצגה למשתמש ? | במצב זה יש לבחור אחד מהם שרירותית. (מתואר עכשיו גם בסעיף 6.5) |
| 2 | 8.4.23 | במידה ויש כמה Input'ים חופשיים בעלי אותו שם והם מתאחדים לכדי input אחד אולם חלקם Mandatory וחלקם optional. האם זה תקין ? | כן. במצב שכזה ה Input כמובן יתקבל ע"י המשתמש (בגלל שהוא mandatory עבור חלק מה step'ים) וכך יהנו ממנו גם ה step'ים שהגדירו אותו optional. (מתואר עכשיו גם בסעיף 6.4) |

תרגיל 2 – מימוש **Stepper** כאפליקציית JavaFX (35%) – הגשה: 14.6.23

## פרטים יבשים

צפי תחילת עבודה: **14.5.23** תאריך הגשה: **14.6.23**

צפי זמן לביצוע: 4 **שבועות +** ציון אפשרי מקסימלי: **140**

משקל התרגיל: **35%** קושי: **מאתגר**

**מטרות התרגיל העיקרית**

1. מימוש תצוגת ותפעול המע' כאפליצקיה גרפית JavaFX
2. הוספת יכולת להרצת ה Flowים בצורה אסינכרונית
3. תמיכה בהמשכיות (continuity) בין flow'ים
4. תמיכה במתן ערכים התחלתיים

## דרישות

1. בתרגיל זה תממשו שכבת משתמש גרפית דרכה המשתמש יוכל לתפעל את המע' ולצרוך את המידע.  
   בנוסף תרחיבו את מנוע המע' במימוש step'ים נוספים, כמו גם תפתחו יכולות חדשות לחלוטין: הרצת ה flow'ים תתרחש מעכשיו בצורה אסינכרונית "במקביל" (בין flow ל flow; לא בתוך ה flow) כמו גם אפשר יהיה להמשיך מהרצת flow אחד להרצת flow אחר בלחיצת כפתור.
2. הרצה אסינכרונית:  
     
   הרצת ה Flow'ים יכולה לקחת זמן (תאורטית, גם אם לא בהכרח במקרה שלנו). במידה וכמה משתמשים ירצו להריץ Flow, וכדי להימנע מממצב שבו אחד חייב לחכות שהרצת ה Flow הקודם תסתיים, נבצע את הרצת ה Flow'ים באמצעות thread'ים עצמאיים תוך שימוש ב thread pool. (כמות ה thread'ים היא נתון המגיע מקובץ ה xml).  
     
   ברגע שהמשתמש בוחר להפעיל flow, לאחר שסיפק לו את כל המידעים הנדרשים – הרצת ה flow לא תתבצע מיידית על ה thread המפעיל (שהוא כמובן ה JAT) אלא בסה"כ תוסיף "משימה" ל threadpool פנימי בתוך מנוע המע' שאמון על הפעלת ה flow'ים. המשימה תכלול את אקט ההפעלה עצמו באותו אופן כפי שהיה במהלך תרגיל 1 ואפקטיבית היא תורץ ע"י thread אחר.   
     
   היות שלא "נתקעים" עד שהרצת flow מסתיימת (כך או אחרת), פירושו של דבר שיש לפתח מנגנון שיאפשר לתשאל את המע' מהו סטטוס הביצוע של הרצת flow שבוצעה זה לא מכבר. בדיוק לשם כך הכינותם מבעוד מועד את המזהה הייחודי לכל הפעלה של flow שבאמצעותו ניתן לבחור ולתשאל את סטטוס העבודה וההתקדמות של כל הרצה והרצה.
3. המשכיות בין flow'ים (Continuation)

הרבה פעמים המשתמש ירצה להריץ flow מסוים ואז לעבור להריץ flow אחר אחריו. ייתכן ויש קשרים בין ההרצות השונות כך שערכים שהוא כבר הזין בהרצה המקורית, הוא יצטרך להזין אותם שוב בהרצה השנייה, ו/או ייתכן שפלטים שהופקו במהלך ההרצה המקורית נדרשים כקלטים עבור ההרצה השנייה.  
  
כדי לחסוך למשתמש את כאב הראש של הכנסה חוזרת של מידעים, תוך פוטנציאל התבלבלות של העתק-הדבק בין הפלטים, הוא יוכל להיעזר ב continuation.  
  
Continuation מאפשר להגדיר קשרים בין Flow'ים שיקלו על המשתמש להפעיל את ה flow הבא בתור.  
ברגע ש Flow מקור יגדיר continuation ל flow יעד, מאחורי הקלעים המע' תדע להציע אפשרות להמשיך ל flow היעד בגמר הרצת ה flow המקורי.   
במידה והמשתמש יבחר בזאת, המע' תבצע העתקה אוטומטית של כל הקלטים המצופים ב flow היעד (mandatory ו/או optional) אשר ניתן למצוא להם התאמה על פי שמם וסוגם במסגרת המידעים שהופקו בהרצה של flow המקור.  
  
בנוסף, במידה ומתעורר הצורך, הגדרת ה continuation יכולה לכלול הכוונה מפורשת של העתקת מידע מ flow המקור אל flow היעד, גם כאשר אין להם את אותו השם (אבל חייב להיות כמובן את אותו הסוג).

שימו לב כי בהחלט ייתכן כי ב flow היעד יוותרו גם input'ים ריקים. אלה עדיין ידרשו כמובן את התערבות המשתמש. (כלומר ה continuation רק "חוסך" למשתמש את הכנסת המידעים האפשריים. לא את כולם).  
  
ייתכן כי יתקיים continuation בין flow x ל flow y בזמן שיש גם continuation בין flow y ל flow x.  
אפשר לבצע continuation גם במידה והרצת ה flow נכשלה. במקרה זה מבצעים העתקה של המידעים הזמינים מהרצת ה flow שנכשל, ככל שכאלה קיימים.  
הגדרת ה continuation מופיעה כחלק מהגדרת flow ב XML. עיקבו אחר [נספח ד'](#appendix_D_XML_2) כדי להבין את פרטיה.

1. ערכים התחלתיים:

לפעמים במסגרת הגדרת flow רוצים להשתמש ב step מסויים שמצפה ל Input כלשהוא. במסגרת ה flow המדובר, ערכו של ה Input המסויים ידוע כבר מבעוד מועד ונכונות ערכו חיונית וקריטית להתנהלותו התקינה של ה flow.   
לא רק זאת, אם חלילה המשתמש בטעות (או בכוונה) ינסה לתת ערך שאינו הערך המצופה – הרי שאז דברים יכולים להשתבש.

כדי להתמודד עם תופעה (אזוטרית) זו, אפשר להשתמש ביכולת הערכים ההתחלתיים.

הגדרה זו המופיעה במסגרת הגדרת flow (מופיעה כחלק מה XML, ראו [נספח ד'](#appendix_D_XML_2)) מגדירה שם של Input וערך התחלתי שהוא מקבל.  
במסגרת הפעלת ה flow הערך הנ"ל מוגדר וקיים וזמין לשימוש כלל ה step'ים במע' לכל עניין ודבר.  
הגדרת Initial input היא סופית ובמידה וקיימת היא זו שמכריעה מה יהיה ערכו של ה input. Input שניתן לו ערך התחלתי כבר אינו נחשב free input. הוא לא מוצג למשתמש. לא ניתן לבצע לו automatic\custom mapping וכו'.

1. מהלך העבודה:

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

* 1. **מסך 1 – Flows Dashboard**

מסך זה יציג את המידע על כל ה Flow'ים הקיימים במע', ויאפשר למשתמש לבחור אחד מהם לביצוע (execution).

בראשית המע' ריקה מכל הגדרות. על כן במסך זה תתבצע טעינה של קובץ xml (על פי סכמה חדשה המתאימה לקבצים בפורמט של תרגיל 2 בלבד). בסיומה יוצגו פרטי כל ה flow'ים הקיימים במע' (בהנחה שהקובץ נטען בהצלחה, כמובן).

שימו לב כי בכל רגע נתון במע', בכל המסכים שלה, יש להציג את הנתיב לקובץ שטעון כרגע. (כפי שעולה מהסקיצה)  
  
קובץ ה XML יכיל הפעם גם מידע נוסף הנוגע לאפשרות של continuation ו initial values

עליכם לבצע בדיקות קלט תקין על קובץ המע'. כל הבדיקות שבצעתם בתרגיל הקודם תקפות.   
בנוסף יש להוסיף את הבדיקות הבאות:

1. לוודא כי אין הכוונה ב continuation ל flow שלא קיים
2. לוודא כי אין הכוונה ב continuation למידע מ flow המקור או היעד שלא קיים ומוגדר כהלכה.
3. לוודא כי אין הכוונה לשם של data שלא קיים ומוגדר במסגרת ה flow כחלק מהגדרת ה Initial values

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

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

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

המשתמש יוכל לראות את כל ה flow'ים הקיימים במע'.   
היות ול flow יש הרבה פרטים הרי שבראייה ראשונית יש להציג רק את הפרטים הבאים:

1. שם ה flow
2. תיאור ה flow.
3. כמות step'ים ב flow
4. כמות input'ים חופשיים (Mandatory & optional) המוגדרים בו
5. כמה continuations מוגדרים ממנו

אם בחר המשתמש flow מסויים אזי יש להציג לו פרטים מעמיקים לגבי ה flow.  
יש להציג את כלל הפרטים כפי שהוצגו בתרגיל 1 פקודה 2.  
  
בנוסף, יתאפשר לבצע ירידה מעמיקה יותר לפרטים של מבנה ה flow והקשרים בין פרטי המידע השונים.  
במסגרת הפרטים יינתנו מידע על ה step'ים המשתתפים ב flow.  
יש לאפשר למשתמש לבחור את כל אחד מה step'ים ועבור כל אחד מהם להציג:

1. שם ה step
2. רשימת כל ה Input'ים שלו, כולל flow level alias במידה והוגדר. לכל input יש לציין אם הוא mandatory או אופציונלי או מחובר ל Output אחר. במידה ומחובר ל output אחר יש להציג את שם ה step ושם ה Output שממנו הוא מגיע
3. רשימת כל ה output'ים שלו, כולל flow level alias במידה והוגדר. אם ה output מחובר ל Input של step אחר יש להציג את שם ה step ושם ה Input אליו הוא מחובר

הציעו למשתמש כפתור שבאמצעותו הוא יכול לבחור את ה flow הזה ל Execution.   
לחיצה על הכפתור תעביר אותו למסך 2...

* 1. **מסך 2 – flow execution**

במסך זה יתאפשר למשתמש לבצע הפעלה של flow.  
  
ראשית יש להציג לו בצורה מסודרת את כל ה Input'ים שיש למלא תוך הפרדה ל mandatory ו optional.  
רק לאחר שמילא את כל ה Mandatory יש לאפשר לו (אך לא לחייב !) ללחוץ על כפתור ייעודי: Start Execution.  
  
בשלב זה ה flow מתחיל להתבצע במרתפי המע' ע"י thread עצמאי מה threadpool.  
יש להציג את פרטי הרצת ה flow:  
ברמה הכללית יש להציג את המידעים הכלליים עבור כל ההרצה, כפי שהוגדר במסגרת פקודה 4 בתרגיל 1 (מסגרות זמנים, תוצאה סופית וכו').  
לאחר מכן יש להציג את רשימת כל ה step'ים שהורצו, תוך מתן מידע על האופן שבו הסתיים כל step (success\warning\failure).   
בחירה ב step מסוים תאפשר לקבל מבט מעמיק על הרצת ה step הספציפי. יש להציג (עבור ה step הנבחר):

1. שם ה step
2. תוצאת הרצת ה step (success\warning\failure)
3. מסגרת הזמנים של הרצת ה step (התחלה;סיום;משך)
4. רשימת כל ה Input'ים (כולל flow level alias) במידה והוגדר. עבור כל input יש לציין את ערכו בהתאם ל user presentation המותאם לסוג ה Input.
5. רשימת כל ה Output'ים (כולל flow level alias) במידה והוגדר. עבור כל output יש לציין את ערכו בהתאם ל user presentation המותאם לסוג ה Input.
6. יש להציג את הלוגים של ה step שנצברו במהלך פועלו

מהרגע שהמשתמש לחץ על Start Execution המע' דוגמת את פועלו של ה flow בצורה קבועה כל 200 ms. בכל פעם היא תעדכן Live את פרטי ההתקדמות (כמה step'ים בוצעו וכו') ותאפשר למשתמש לנבור לעומקה של ההרצה שכבר זמינה לו. אפשר לבצע עבודה זו באמצעות Task של JFX (הפתרון האידיאלי, כפי שנלמד בכיתה), או ללמוד על רכיב שנקרא Timer...

בסיום ההרצה, ובסיומה בלבד, יש להציג למשתמש את פרטי ה continuation מ flow זה ככל שקיימים כאלה.  
יש להציג את שמות ה flow'ים אליהם אפשר להמשיך. לחיצה ובחירה ב flow שכזה מאפסת את המסך הנוכחי לפרטי ה flow החדש שאותו בחרנו. כל ה Input'ים הרלוונטים יגיעו כשהם כבר מוכנים ומלאים במידעים שנצברו בהרצה הקודמת והמשתמש יוכל להמשיך ולהפעיל גם את ה flow הזה.

המשתמש יכול לבחור לעזוב את מסך בזמן שה FLOW מתבצע. (הרצת ה FLOW יכולה כמובן לקחת פרק זמן ניכר).  
בכל פעם שהוא בוחר לחזור למסך זה, על המערכת להציג את מצב ההתקדמות של ה FLOW האחרון שהופעל, ככל שכזה עדיין רץ מאחורי הקלעים, או מסך ריק מתכנים אם כרגע לא מורץ שום FLOW ברקע. (שימו לב: לא ניתן לבחור FLOW לביצוע ממסך 2; יש לעשות זאת רק ממסך 1 או ממסך 3, כפי שמוגדר).  
במידה ו FLOW רץ ברקע אפשר לחזור למסך 1 (או 3) ולבחור FLOW נוסף להרצה. פעולה זו תחזיר אותנו למסך 2, והמסך יתנקה לקראת הפעלת ה FLOW החדש. ברגע שנפעיל את ה FLOW החדש – המסך יציג את סטטוסו בלבד, שכן זהו ה FLOW האחרון שהוזנק לביצוע כרגע. כל זאת כאשר ברקע ייתכן ועדיין רץ FLOW נוסף (ע"י thread אחר בבטן המנוע). את פרטיו של ה FLOW הנוסף, "הקודם", לא ניתן יהיה לראות יותר, אלא אך ורק כשהוא ייגמר, וזאת באמצעות מסך 3.

* 1. **מסך 3 – execution history**

במסך זה יש להציג את היסטוריית ההרצות של כל ה flow'ים שהרצתם הסתיימה זה לא מכבר.  
יש להציג טבלה המתארת את כל ההרצות הקודמות. עבור כל flow יש להציג בטבלה:

* + 1. שם ה flow
    2. מועד ההרצה
    3. תוצאת ההרצה

יש לאפשר למשתמש למיין את הטבלה על פי כל אחת מהעמודות, כמו גם לסנן על פי תוצאת הריצה בלבד (בקשה לראות רק הרצות שנגמרו בהצלחה או כשלון וכו')

בבחירת אחת מההרצות מהרשימה יש להציג למשתמש את פרטי ההרצה כפי שזו מוגדרת במסגרת מסך 2 (מי אמר REUSE ולא קיבל !)  
  
יש להציג כפתור ייעודי המאפשר לבצע הרצה חוזרת של ההרצה שכרגע מציגים למשתמש. הרצה חוזרת תחזיר את המשתמש למסך 2, כשכל ה Input'ים (mandatory ו/או optional) מלאים בערכים שהוזנו בהרצה המקורית.   
  
שימו לב: ה flow לא מורץ אוטומטית. אם המשתמש ירצה הוא יצטרך ללחוץ בעצמו על הכפתור של Start Execution.

1. **מסך 4 – statistics**

במסך זה יש להציג את הסטטיסטיקות הנצברות במהלך הפעלת ה flow'ים השונים (כשם שהיה בתרגיל 1).  
יש להציג את המידע ב 2 טבלאות: אחת עבור flows והשנייה עבור steps.

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 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 ?

התחילו בהקמת השלד הראשי של המע' ובהתאם לקווי המתאר של האפליקציה.   
  
התחילו בהקמת המסך הראשון – שאמור להתכתב ולתפעל את היכולות הקיימות שלכם במע'. טענו קובץ והציגו את פרטי ה flow'ים הרלוונטים.

עיברו לממש בצד המנוע את היכולת להריץ flow'ים במקביל. לאחר/בזמן שזה קורה תוכלו להמשיך במימוש המסך השני שבו תתאפשר הרצה של flow, תוך מילוי הפרטים הרלוונטים.   
חישבו כיצד אפשר להציג את סוגי ה data השונים כך שתוכלו להשתמש ברכיבים ייעודיים לטובת כל סוג של data במהלך הצגתם במסכים השונים.

לאחר מכן עיברו לממש את מסך 3 של ההסטוריה. שימו לב ושאפו להשתמש בכמה שיותר רכיבים שכבר קיימים לכם במסגרת מסך 2 כדי להציג את תוכנה של הרצה מסויימת.

רק בסוף עיברו לממש את יכולת ה continuity, ראשית בצד המנוע ואז השלימו זאת במסך 2 מבחינת חווית ממשק המשתמש.

## בונוסים

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | הגדלת ראש נאה | אפשרות החלפת skin למשחק. בבונוס זה תממשו (לפחות) 2 ערכות צבעים נוספות על הערכה הדיפולטית, ותאפשרו למשתמש להחליף את ערכות הצבעים. שימו לב ערכת הצבעים כוללת (לכל הפחות):   1. החלפת רקע של כל המסך 2. החלפת המראה של הכפתורים על המסך 3. החלפת הפונט והגודל של כל ה label על המסך | כי זה בכל זאת תרגיל בממשק גרפי – ומה יותר גרפי מזה ??? | **5 נקודות**  **(ועד ל 100)** |
| 2 | הגדלת ראש קלילה | ממשו 2-3 אנימציות המלוות את תהליך ההפעלה  האנימציה צריכה להתבצע לכל היותר במשך 2 שניות  יש לאפשר גם לנטרל את האנימציות הנ"ל, כך שהיא לא תאיט את התקדמות המשחק. | כל כך קל... | 5 נקודות  (ועד ל 100) |
| 3 | פיסת עוגה | במסך הסטטיסטיקה הציגו את המידעים השונים בנוסף גם באמצעות גרפים מתמטים של עמודות/קווים. יש ב java fx יכולת מובנת להצגת גרפים... | כשמגלים כמה זה קל – זה מרגיש מדהים ! | 5 נקודות **(מעל ל 100)** |
| 4 | מדהים ! | הציגו דיאגרמה גרפית של ה flow במסגרת מסך מס' 1. אני ממליץ להשתמש בכלי חיצוני ונפוץ שמייצר דיאגרמות בשבילכם תוך התייחסות לארגון הרכיבים על המסך: GraphViz. יש ללמוד עליו לבד, אבל הוא די שווה וסביר שתיתקלו בו במהלך חייכם המקצועיים... אפשר לעקוב אחר הדוגמאות [בנספח ג'](#appendix_C_DIAGRAMS) כדי להתרשם איך ואילו מידעים יש להציג בדיאגרמה שכזו... | אחלה פיצ'ר להשוויץ בו בראיונות.. | 10 נקודות **(מעל ל 100)** |
| 5 | בלתי ניתן להכלה ! | אפשרו למשתמש ליצור flow'ים בעצמו, באמצעות כלי גרפי על בסיס drag & drop. ה flow יווצר ויסורטט על גבי קנבס "אינסופי". המשתמש יוכל "לגרור" לקנבס step'ים שהוא רוצה שישתתפו ב flow.  כל חיבור אוטומטי בין output ל input יווצר ויוצג אוטומטית ע"י המע'.  המשתמש יקבל את האפשרות גם "למתוח" קו מקשר בין Output ל Input כאמצעי להצגת ה custom mapping. יש לאפשר כמובן להגדיר alias (עבור step וברמת ה flow). בסוף יש לאפשר לשמור את ה flow במע' ולהשתמש בו כאילו היה flow רגיל שנטען מתוך קובץ מידע. | אם אתם בעסקי FE – זו אחת ההתמודדויות המורכבת ביותר לביצוע ויזניק אתכם השמימה ! | 25 נקודות **(מעל ל 100)** |
| 6 | הבונוס על שם יגאל ומיכאל | הפעלה מחדש של FLOW במסגרת מסך ה execution  אפשרו למשתמש להפעיל מחדש את ה FLOW שהרצתו זה הרגע הסתיימה כבר ומיידית תוך שאתם נמצאים במסך 2 – מסך ה execution. שימו לב שה input'ים מקבלים אוטומטית את הערך שהוכנס להם בהרצה שהסתיימה, אולם המשתמש עדיין יכול לערוך אותם ולשנות אותם אם ירצה. בכל מקרה המשתמש חייב ללחוץ אקטיבית על start execute כדי להפעיל מחדש את ה FLOW (הוא לא מורץ אוטומטית או משהו כזה).. | כי מיכאל ויגאל כבר מימשו את זה, וזה מה-זה קל... | 5 נקודות (עד ל 100) |
| 7 | בונוס על שם עמית | הפעלת continuation מעמוד ה history אפשרו למשתמש בראותו execution שכבר נגמר דרך מסך ה history לבחור את ה continuation המתאים, ככל שכזה מוגדר. ברגע שבחר הוא מועבר חזרה למסך 2, כאשר ה input'ים מלאים על פי כל חוקי ה continuation כפי שכבר מימשתם בהגדרת הפיצ'ר מלכתחילה | כי זה סופר קל | 7 נקודות (עד ל 100) |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. jar (אחד או יותר) שהוא כל הקוד שלכם, בצירוף קובץ אצווה (batch) שהפעלתו תריץ את התוכנית
2. קבצי קוד המקור של הפרויקט שלכם.
3. קובץ ההגשה יכיל גם קובץ readme שיכיל הסבר על המערכת, בחירותיכם השונות במקומות שבהם היו לכם בחירה, כמו גם כל דבר נוסף העולה על דעתכם שחשוב שהבודקת תדע.
4. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות החדשות ותפקידם. (חישבו מה יסייע לבודק להיכנס ביתר קלות לקוד שלכם ולהבין מי נגד מי...)
5. במידה ומימשתם את מי מהבונוסים לעיל, ציינו זאת בקובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

## שאלות ותשובות

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

תרגיל 3 – מימוש **Stepper**כאפליקציית Client - Server (35%) – הגשה: 23.7.23

## פרטים יבשים

צפי תחילת עבודה: **15.6.23** תאריך הגשה: **23.7.23**

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. הוספת מנגנון הרשאות ומשתמשים

## דרישות

1. בתרגיל זה נממש את היכולת לתפעל את stepper כשרת, המציע את יכולות הרצת ה flow'ים עבור משתמשים שונים ומגוונים.   
   משתמשים מסוימים יוכלו לטעון קבצים המכילים הגדרות flow'ים למע', ומנגד משתמשים אחרים יוכלו לבחור ולהריץ את ה flow'ים השונים המוגדרים במערכת (במקביל, כמובן), לקבל את התוצאות השונות, להחליף ביניהם מידעים, לראות היסטוריה וכו'.
2. השרת (tomcat) יכיל את מנוע ה stepper ויחשוף את היכולות בדמות endpoints שונים.  
   מול השרת יעבדו 2 סוגים של לקוחות/יישויות:

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

* העלאת קבצים (אותו הפורמט של תרגיל 2) המכילים הגדרות של flow'ים.   
  הפעם כל קובץ המכיל הגדרות flow'ים מתווסף למנוע המע' (ולא מחליף את הקובץ הקיים, כפי שהיה בתרגיל 1 ו 2).   
  כך יכול האדמין להעלות כמה קבצים ולהעשיר את המע' ב flow'ים רבים.
* לראות ולנהל את הרשאות הגישה של המשתמשים השונים (ראו פירוט בהמשך).
* חשיפה לכל ההרצות שבוצעו במע' ויכולת לראות אותן ואת תוצאותיהן בכל רגע נתון.
* קבלת מידע על מצבה הפנימי של המע': מי מריץ כמה ולמה.

**Client – משתמש**

קליינטים הם משתמשים "רגילים" במע', שרוצים להנות מיכולותיה.   
ברגע נתון יכולים להתחבר למע' ולהשתמש בה מספר רב של קליינטים.  
בין יכולותיו של הקליינט:

* בחירת flow, הפעלתו וצפייה בתוצאותיו, כשם שהיה בתרגיל 2 (reuse !).   
  הקליינטים יוגבלו ב flow'ים שהם יכולים להפעיל ולהיחשף אליהם ע"י מנגנון ההרשאות.
* צפייה בתוצאות הרצות העבר של הקליינט עצמו בלבד (כלומר לא של הרצות שבוצעו ע"י משתמשים אחרים)

**Stepper  
Server  
(Tomcat)**

**HTTP**

**HTTP**

Administrator  
(Client)

Client

Client

Client

**…**

1. מודל ההרשאות:

בתרגיל זה תתמכו במודל הרשאות עבור הקליינטים השונים.   
המוטיבציה היא כי לא כל המשתמשים זהים ולכן לא כולם יכולים להיחשף ולהפעיל את כל ה Flow'ים.  
  
המע' תגדיר יישות חדשה הנקראת Role (תפקיד).   
כל Role יגדיר אוסף של flow'ים "המורשים" להרצה על ידו.  
הגדרת ה Role תכיל:

* 1. שם
  2. תיאור קצר
  3. רשימה של שמות flow'ים המותרים ל Role זה.

שימו לב:

* ה role'ים השונים נבדלים זה מזה בשמם. (כלומר השם ייחודי)
* Flow מסויים יכול להיות משויך ומוגדר במסגרת כמה role'ים שונים.

המע' תנהל הגדרה של יישות משתמש, אשר תתאר משתמש במע'.   
לכל משתמש יש שם ייחודי המתקבל במסגרת הרשמותו למע'.

לכל יישות משתמש יוצמד/ישוייך role אחד או יותר.   
המשתמש ייחשף עכשיו אך ורק לflow'ים המותרים במסגרת ה role'ים המשוייכים אליו.   
**שימו לב**: ייתכן כי אותו flow יוגדר בכמה role'ים שונים. סך ה flow'ים החשופים למשתמש הם כמובן קבוצת האיחוד הייחודית (ביטוי שהמצאתי) של סך ה flow'ים השונים הנגזרת מאוסף ה role'ים המשוייכים למשתמש.

בנוסף לכך, ניתן להגדיר משתמש (קליינט רגיל במע') כ Manager.   
במידה והוא מוגדר ככזה – הוא מקבל אוטומטית גישה לכלל ה flow'ים במע', כמו גם לצפייה בהרצות עבר של כלל המשתמשים.

רק האדמין יכול לנהל את המשתמשים, ה role'ים השונים ולבצע השמות ושיוכים שלהם למשתמשים השונים.

המע' תעלה עם 2 role'ים קבועים ומוגדרים מראש:

1. קריאה בלבד:  
   תפקיד זה מתאר את כל ה flow'ים שהם read-only (תזכורת: flow נחשב read-only אם כל ה step'ים המוגדרים במסגרתו הם read-only).  
   שם התפקיד הוא **Read Only Flows**.   
   בעליית המע' ה role הזה יהיה מוגדר אוטומטית, יתעדכן אוטומטית ויכיל תמיד את כל ה flow'ים שהם read-only.
2. כולם-כולל-כולם:

תפקיד זה מתאר את כל ה flow'ים המוגדרים במע' בלי שום יוצא מן הכלל.   
שם התפקיד הוא **All Flows**.   
בעליית המע' ה role הזה יהיה מוגדר אוטומטית, יתעדכן אוטומטית ויכיל תמיד את כל ה flow'ים הקיימים במע' בלי שום יוצא מן הכלל.  
  
(שימו לב כי אפשר לממש את המשתמש מסוג manager באמצעות role זה...)

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 ה flow'ים שנטענו ההרצות שבוצעו וכו'.

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

## אפליקציית Administrator

אפליקציה זו מותאמת לקליינט מסוג Admin. עיקבו אחר הסקיצה כדי לקבל הבנה כללית של מבנה האפליקציה.  
ברגע נתון ניתן להפעיל בדיוק מופע אחד שלה.  
(בניסיון להרים את האדמין השני האפליקציה פשוט תיכשל לעלות ותוציא הודעה מתאימה למשתמש)  
אם מורידים אדמין ומעלים אותו מחדש – המידע שהוגדר באמצעותו נשמר בשרת וכמובן מוצג כמות שהוא לאדמין החדש.

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

כשתוכן הקובץ מגיע לשרת, יבוצעו בו בדיקות הקלט כפי שבוצעו במהלך תרגיל 1 ו 2.

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

כל הגדרות ה flow'ים במסגרת הקובץ התקין יועלו ויתווספו לרשימת ה flow'ים המוגדרת בשרת.  
במידה וקובץ מכיל הגדרת flow שכבר מוגדרת וקיימת במע' (על פי השם) אין לטעון אותה מהקובץ החדש ונשארים עם ההגדרה הישנה (שימו לב זה עדיין נחשב קובץ תקין !)  
  
גם כאן ינוהל threadpool פנימי בתוך השרת אשר ישמש להרצת ה flow'ים. המידע לגבי גודל ה pool יגיע מקובץ ה xml הראשון שייטען ויישאר קבוע בהתאם ! (גם אם בקבצים הבאים שנטענים יש שינוי של מספר זה).

האדמין לא יכול להפעיל flow'ים, אולם הוא יכול לנווט למסך של הסטטיסטיקות והיסטוריית ההרצות כשם שהיה בתרגיל 2.

בנוסף יש לו 2 מסכים חדשים, במסגרתם הוא יכול לנהל את המשתמשים וההרשאות.

**מסך ניהול role'ים:**יש לאפשר להציג את כל ה role'ים המוגדרים במע', על כלל פרטיהם (כפי שמפורט בהמשך).  
עבור כל role יש להציג את רשימת המשתמשים שהוא מוקצה להם. (יכול להיות גם רשימה ריקה).

יש לאפשר להגדיר role חדש.  
במסגרת הגדרת role יש לקבל את שמו, תיאורו ולאפשר לבחור אוסף של flow'ים המוקצים ל role זה.

יש לאפשר לחזור ולעדכן את ה flow'ים המשויכים ל role בכל רגע נתון.

**מסך ניהול משתמשים:**

יש להציג במסך זה את כל המשתמשים המחוברים למע'.  
עבור כל משתמש, יש להציג:

* שם (כפי שנרשם איתו במסך הלוגין)
* רשימת ה role'ים המשוייכת אליו
* סך כל ה flow'ים המאופשרים לכל משתמש (כפונקציה של ה role'ים המשויכים לו)
* סך כל הרצות ה flow'ים שביצע המשתמש הנ"ל

יש לאפשר לבחור ולשייך כמה role'ים שרוצים למשתמש מסויים.   
role מסוים יכול להיות בשימוש ע"י כמה משתמשים.   
ייתכנו משתמשים שאין להם role בכלל (במקרה זה הם מנוונים, אלא אם הם מוגדרים כ Manager)

יש לאפשר לאדמין להגדיר עבור כל משתמש אם הוא Manager או לא (ברירת המחדל היא לאו).

## אפליקציית משתמש

משתמש מתחבר למע' באמצעות עמוד לוגין מסודר. במסגרת עמוד זה עליו לספק את שמו. לא ניתן להירשם למע' עם שם שכבר קיים (במידה וזה קורה יש לתת התראה מתאימה).  
**אין לחייב** או להגדיר שימוש בסיסמה או כל רכיב authentication כזה או אחר.

לאחר רישום מוצלח, המשתמש מגיע לעמוד הצגת ה flow'ים כפי שהיה בתרגיל 2.  
המשתמש יוכל לראות אך ורק את סך ה flow'ים המוגדרים ב role'ים שהוא קיבל (ע"י האדמין). ייתכן משתמש שאין לו role'ים בכלל ואז למעשה הוא לא רואה שום flow שהוא יכול להריץ.  
היות והגדרת ה role'ים, כמו גם שיוכם למשתמשים השונים, יכולה להשתנות תדיר ע"י האדמין, הרי שאפליקציית המשתמש צריכה לבצע משיכה של מידע המשתמש וה role'ים השונים כל פרק זמן קצוב (לכל היותר 2 שניות) כדי להתעדכן במצב ההרשאות של המשתמש. (ממליץ על delta fetching כדי להימנע מעדכוני יתר וריצודים במסך)  
  
יש להציג למשתמש בכל רגע נתון, בצורה מסודרת את הפרטים הבאים:

* שם המשתמש
* שמות ה role'ים המוקצים לו
* האם הוא מוגדר כ Manager או לא.

משתמש יכול להריץ רק flow'ים שהוא מורשה לראות.   
במסגרת הפעלה והרצה של כל flow יש להוסיף עוד פרט מידע חדש המתאר את זהות המשתמש שהריץ את ה flow.  
יש להגדיר את הפרטים הבאים:

* שם המשתמש
* תפקיד (רגיל/מנהל)

משתמש יכול לצרוך ולראות אך ורק הרצות שהוא ביצע בעבר. הוא אינו חשוף ואינו יכול לראות הרצות שבוצעו ע"י אחרים.  
במידה ונלקח למשתמש role שהיה לו בעבר, הוא עדיין יכול לראות הרצות עבר שהוא ביצע על flow'ים שכרגע הוא אינו מורשה להם, אולם הוא לא יוכל להפעיל הרצות חדשות שכאלה.

במידה וקליינט מוגדר כ Manager (ע"י האדמין), הוא יכול לראות את כל ה flow'ים הקיימים במע' ויכול לראות הרצות עבר של כלל המשתמשים במע'.  
במידה והרשאות ה Manager יילקחו לו – הוא יחזור להיות משתמש רגיל שחשוף למידע אך ורק על פי הגדרות ה role'ים שלו.  
  
במידה ומוגדר continuity ל flow שאין למשתמש הרשאות לבצעו – הוא לא יופיע לו כאפשרות הפעלה.

## 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 אפליקציות ה client השונות (על שלל Jar'הן).

## איך מתחילים ?

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

התחילו בפיתוח אפליקציית ה admin והמשתמש ללא מודל ההרשאות. זה אמור להיות זהה כמעט לחלוטין למצב הפתיחה איתו אתם מגיעים לאחר תרגיל 2.

המשיכו בפיתוח מודל ההרשאות בצד המנוע שבשרת. לאחר מכן פתחו את 2 המסכים החדשים בצד האדמין המאפשרים להגדיר ולשחק את משחק ההרשאות.  
המשיכו בעדכון אפליקציית המשתמש שתדע למשוך את המידע באופן תדיר מצד השרת ולפעול בהתאם לו.

לאורך פיתוח 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 סוגי הקליינטים יצטרכו להשתמש בקוד משותף (למשל כל תפעול וגישה ל http client). מאוד הגיוני לייצר מודול נוסף (== jar) שיחזיק את סך הקוד המשותף לאפלקיציות (וייכלל כמובן בכולן).

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

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | הגדלת ראש מגניבה ! | Chat:  אפשרו למשתמשים לקיים chat בינהם תו"כ התחברותם למע'.  ה chat צריך להתנהל כולו אל מול השרת בצורה שבה כולם רואים את מה שכולם כותבים. (מאוווווווווווד דומה לדוגמא הסופית)  הצ'ט מתבצע בין המשתמשים הרשומים והמחוברים למערכת בלבד. וצריך להיות פעיל הן בעמוד 2 והן בעמוד 3. | היש מדהים מזה ??? | 5 נקודות  (ועד ל 100) |
| 2 | די מתבקש | LOGOUT:  אפשרו למשתמש (לא האדמין) לעזוב את המע'.  במקרה זה כל הרצות העבר שלו לא נמחקות, אולם שמו נמחק מהמע' ואפשר יהיה להירשם עם שם זה מחדש. יישות המשתמש עצמה נמחקת מהשרת ולמעשה זולת הרצות העבר שלו – אין שום עדות עבר לקיומו במע'. | עוד קצת משחקי Servlet'ים.. מה קרה ?? (אבל פיצ'ר מדהים לספר עליו במהלך ראיונות..) | 10 נקודות (ועד ל 100) |
| 3 | בקטנה... | מחיקת ROLE: יש לאפשר לאדמין למחוק role.  במידה והוא כרגע משויך למשתמש אחד לפחות יש להתריע על כך בפני האדמין ולמנוע את המחיקה של ה role עד אשר הוא לא בשימוש ע"י אף משתמש אחר. | אמור להיות פיסת עוגה.. | 5 נקודות (ועד ל 100) |
| 4 | בונוס על שם עומרי וגל | הורדת אדמין והעלתו מחדש אפשרו להוריד את אפליקציית האדמין ולהעלות אותה מחדש. שימו לב: היות ויש אדמין אחד ויחיד – אזי לא מוחקים או מאפסים את המידע שהאדמין הקודם טען; האדמין החדש שעולה רואה את המידע שכבר נטען ע"י האדמין הקודם שירד. | זה היה דרישת חובה, שהפכה לבונוס.  כלומר זה בונוס מתנה... | 5 נקודות (ועד ל 100) |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. WAR אחד בדיוק אשר יושם בספריית tomcat\webapp ויעבור deployment אוטומטי.   
   על WAR זה לכלול את כל התלויות שלכם. אין להניח כי יסופקו לכם תלויות מבחוץ (למשל Gson וכו').
2. 2 ספריות עבור הקליינטים השוניםץ על הספריות להיקרא Admin, User.   
   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 הרלוונטי לה   
   אפשר כמובן להניח את localhost:8080 כדומיין של השרת; אפשר להניח כי לא נשנה את שם קובץ ה WAR שהגשתם.
3. קבצי קוד המקור של הפרויקט שלכם (גם צד השרת וגם אפליקציות java fx של ה clients).
4. קובץ ההגשה יכיל גם קובץ readme שיכיל הסבר על המערכת, בחירותיכם השונות במקומות שבהם היו לכם בחירה, כמו גם כל דבר נוסף העולה על דעתכם שחשוב שהבודקת תדע.
5. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות החדשות ותפקידם. (חישבו מה יסייע לבודק להיכנס ביתר קלות לקוד שלכם ולהבין מי נגד מי...)
6. במידה ומימשתם את מי מהבונוסים לעיל, ציינו זאת **בראשית** קובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

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

## שאלות ותשובות

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

נספח א' – פירוט ה Data Definitions

המע' תגדיר כמה סוגים של מידעים (Data Definition) אפשריים שיזרמו בין ה step'ים השונים.

הנה כמה מאפיינים של Data Definitions שיש לתת עליהם את הדעת:

* המידעים מסודרים על פי סדר נחיצותם בהתאם לסדר התרגילים. רובם המוחלט נדרשים כבר מתרגיל 1.
* יש בהם מידעים פשוטים ופרימיטיבים שקלים להבנה ולמימוש (למשל String, Number) ויש כאלה "מסובכים" יותר הדורשים מימוש מורכב כדי להכיל את כלל המידע המוצג בהם (למשל Relation, List)
* לכל פרט מידע מצויין האם הוא user friendly או לא. במידה והוא user friendly פרט מידע זה יכול לשמש כ mandatory input אותו יצטרך להכניס המשתמש. פרטי מידע שאינם user friendly יכולים להופיע רק בשבתם כפרטי מידע המופקים בתוך/במסגרת flow ולא כ input שהמשתמש יצטרך להכניס
* לכל פרט מידע מצויין user presentation המגדיר כיצד עליו להיות מוצג למשתמש.   
  natural פירושו שיש להציג באופן הטבעי (למשל מספר, מחרוזת).   
  אחרת מתואר בדיוק מה יש להציג במסגרתו.

אתם חופשיים לממש את הגדרת המידע כרצונכם.

|  |  |  |
| --- | --- | --- |
| # | שם | נדרש החל מתרגיל |
| 1 | String | 1 |
| 2 | Number | 1 |
| 3 | Double | 1 |
| 4 | Relation | 1 |
| 5 | List | 1 |
| 6 | File | 1 |
| 7 | Mapping | 1 |
| 8 | Enumerator | 2 |
| 9 | Json | 3 |

1. **מחרוזת תווים**

שם הסוג: String  
user friendly: true

User presentation: natural

תיאור: מחרוזת תווים כלשהיא. המחרוזת יכולה להכיל כל תו, כולל ספרות, תווים מיוחדים וכו'.

1. **מספר שלם**

שם הסוג: Number

user friendly: true

User presentation: natural

תיאור: מספר שלם במע' (בגבולות Integer), כולל מספרים שלילים.

1. **מספר עשרוני**

שם הסוג: Double

user friendly: true

User presentation: natural

תיאור: מספר עשרוני במע' (בגבולות ה Double). כולל מספרים שלילים

1. **טבלה**

שם הסוג: Relation

user friendly: false

User presentation: יש להציג את שמות העמודות ואת כמות השורות בטבלה. מתרגיל 2 ואלך יש להציג כטבלה מסודרת.

תיאור: מידע זה מכיל מידע "טבלאי".   
כחלק מהגדרתו יש לקבל את שמות כל העמודות שבטבלה. כל עמודה היא מסוג של מחרוזת תווים בלבד. (כלומר הטבלה מכילה מחרוזות בלבד)  
יש חשיבות לסדר העמודות בטבלה.  
המידע עצמו יכיל 0 או יותר "שורות" שבהם יש ערך לכל עמודה בטבלה.  
ייתכן שלא בכל העמודות יהיה ערך. במקרה זה הערך הדיפולטי שיש להציג הוא N/A (אלא אם הוגדר אחרת)

1. **רשימה**

שם הסוג: List

user friendly: false

User presentation: יש להציג את האברים השונים (בהתאם ל User presentation שלהם) תוך מתן ספרור רץ המתחיל מ 1 לפני כל אחד מהם.

תיאור: מידע זה מתאר רשימה של "דברים".   
הרשימה יכולה להיות מכל סוג שלdata definition , למשל: רשימה של מספרים, רשימה של מחרוזות, רשימה של טבלאות...  
**אין צורך** לתמוך ברשימה של רשימות.

1. **קובץ**

שם הסוג: File

user friendly: false

User presentation: נתיב מלא של הקובץ

תיאור: מידע זה מתאר קובץ במע' (לא ספריה. רק קובץ). יש לשמור במסגרתו את הנתיב המלא של הקובץ חוץ מכל נתון אחר שתמצאו לנכון (למשל input stream ?)

1. **צמד**

שם הסוג: Mapping

user friendly: false

User presentation: יש להציג את האברים השונים (בהתאם ל User presentation שלהם).

הפורמט יהיה כזה:

car: <car user presentation>  
cdr: <cdr user presentation>

תיאור: מידע זה מתאר צמד ערכים ויכול לשמש גם כמיפוי בין ערך אחד למשנהו.   
בתוך צמד הערכים הראשון ייקרא car והשני ייקרא cdr.   
סוגי הערכים יכולים להיות כל אחד מה data definitions המוגדרים במע'.   
**אין צורך** לתמוך בצמד המכיל צמד וכו'.

1. **סט סגור של ערכים**

שם הסוג: Enumerator

user friendly: כן

User presentation: natural

תיאור: מידע זה מתאר סט סופי מוגדר של ערכים מסוג של String בלבד.   
הערכים עצמם יוגדרו במסגרת ובהקשר ה step הרלוונטי, ומבנה זה יוכל לקבל או להכיל אך ורק את אחד מהערכים המדוברים.

1. **JSON**

שם הסוג: Json

user friendly: כן

User presentation: כמחרוזת json

תיאור: מידע זה מתאר JSON בפורמט של String.

נספח ב' – פירוט ה Steps

הרשימה מטה מתארת את כלל ה step'ים המוגדרים במע'.

הנה כמה מאפיינים שיש לקחת בחשבון בבואכם לקרוא ולממש את הגדרת ה step:

* כל step מוגדר במסגרת תרגיל מסוים ואין ציפייה שתממשו כבר בהתחלה את כולם.
* לכל step מוגדרים ה input'ים וה output'ים שלו. עבור כל אחד מהם יוגדר סוג ה data definition והשם שלו במסגרת ה step עצמו. (השם של ה data definition יכול ואף סביר שיהיה שונה משם הסוג).
* עבור כל input יוגדר אם הוא נחוץ או אם לאו.   
  עבור Input'ים שאינם נחוצים יוגדר ערך ברירת מחדל במידה והם לא סופקו.
* שם ה input/output במסגרת ה step יורכב מאותיות גדולות בלבד תוך שימוש ב \_ (underscore) כדי להפריד בין מילים (למשל: FOLDER\_NAME).   
  היות והצגת מידע שכזה למשתמש אינו תמיד נעים בעין, לכל סוג של input\output יוצמד גם מחרוזת משתמש ובכל המקרים בהם יש להציג את המידע יש להשתמש במחרוזת המשתמש הרלוונטית.
* לכל step יצוין האם הוא read only או לא (ערך בוליאני; true פירושו שהוא read only)
* לכל step יוגדרו רשימות הלוגים שהוא צריך לפלוט במסגרת פועלו.
* לכל step יוגדרו "מקרים ותגובות" כיצד הוא צריך להיגמר בהתאם לפועלו.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | שם | Read only | סך הקלטים | סך הפלטים | נדרש החל מתרגיל |
| 1 | [Spend some Time](#_מבזבז_הזמן) | **V** | 1 | 0 | 1 |
| 2 | [Collect Files In Folder](#_סורק_הקבצים) | **V** | 2 | 2 | 1 |
| 3 | [Files Deleter](#_מוחק_הקבצים) | **X** | 1 | 2 | 1 |
| 4 | [Files Renamer](#_משנה_השמות) | **X** | 3 | 1 | 1 |
| 5 | [Files Content Extractor](#_מחלץ_המידעים) | **V** | 2 | 1 | 1 |
| 6 | [CSV Exporter](#_פולט_הפורמטים:_CSV) | **V** | 1 | 1 | 1 |
| 7 | [Properties Exporter](#_פולט_הפורמטים:_Properties) | **V** | 1 | 1 | 1 |
| 8 | [File Dumper](#_כותב_הקובץ) | **V** | 2 | 1 | 1 |
| 9 | [Zipper](#_רוכסן) | **X** | 2 | 1 | 2 |
| 10 | [Command Line](#_שורת_הפקודות) | **X** | 2 | 1 | 2 |
| 11 | [HTTP Call](#_משגר_הבקשות) | **X** | 4 | 2 | 3 |
| 12 | [To Json](#_המג'ייסן) | **V** | 1 | 1 | 3 |
| 13 | [Json Data Extractor](#_מחלץ_המידעים_1) | **V** | 1 | 1 | 3 |

## מבזבז הזמן

**שם**: Spend Some Time

**תיאור**: מבצע sleep למשך זמן מגודר בשניות.

**נדרש החל מתרגיל**: 1

**Read only**: true

**Inputs**:

* שם: TIME\_TO\_SPEND

סוג: Number

תיאור: כמות של שניות שיש "לבזבז" (בקיצור, sleep).

מחרוזת משתמש: Total sleeping time (sec)

נחיצות: חובה

**Outputs**: אין

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to sleep for <TIME\_TO\_SPEND> seconds… | רגע לפני שמתחיל בזבוז הזמן |
| 2 | Done sleeping… | לאחר שמסיים את השינה |

**מקרים ותגובות**:

* במידה ומתקבל מספר שלילי או 0 – נכשל. יש לתאר הסבר על המצב בשורת הסיכום, כמו גם בשורת לוג ייעודית

## סורק הקבצים

**שם**: Collect Files In Folder

**תיאור**: בהינתן נתיב לספריה במע', יעבור ויסרוק את כל הקבצים בלבד הנמצאים בה ויחזיר רשימה (List) של ערכים מסוג של קובץ (File).

**נדרש החל מתרגיל**: 1

**Read only**: true

**Inputs**:

* שם: FOLDER\_NAME

סוג: String

תיאור: נתיב מלא לספריה אותה רוצים לסרוק

מחרוזת משתמש: Folder name to scan

נחיצות: חובה

* שם: FILTER

סוג: String

תיאור: מחרוזת על פיה יש לסנן את סוגי הקבצים לסריקה על פי הסיומות שלהם בלבד (מחרוזת מהסוג של \*.xml וכו'). אפשר להניח כי המחרוזת תהיה תקינה, ואין צורך לבצע בה וידוא קלט.

מחרוזת משתמש: Filter only these files

נחיצות: אופציונלי. במידה ולא מתקבל – סורקים את כל הקבצים.

**Outputs**:

* שם: FILES\_LIST

סוג: List המכילה אובייקטים מסוג File

תיאור: רשימת הקבצים שנמצאו במסגרת סריקת הספרייה ובהתאם ל FILTER

מחרוזת משתמש: Files list

* שם: TOTAL\_FOUND

סוג: Number

תיאור: כמות הקבצים שנמצאה בסריקת הספרייה.

מחרוזת משתמש: Total files found

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | Reading folder <folder full path> content with filter <filter content> | לפני שמתחילה קריאת הקבצים |
| 2 | Found <total files found> files in folder matching the filter | לאחר קבלת הקבצים |

**מקרים ותגובות**:

* במידה וה folder לא קיים, או שהנתיב שהוכנס אינו folder – ה step ייגמר בכשלון.   
  יש לספק בשורת הסיכום הסבר על מהות הסיבה לכשלון, כמו גם להוסיף לוג (עם תוכן כרצונכם) במקרה זה.
* במידה וה folder קיים אך אין בו קבצים – ה step ייגמר ב warning.   
  במקרה זה ה FILES\_LIST יהיה ריק ו TOTAL\_FOUND יהיה 0.   
  יש לספק בשורת הסיכום הסבר על המצב, כמו גם להוסיף לוג (עם תוכן כרצונכם) במקרה זה.

## מוחק הקבצים

**שם**: Files Deleter

**תיאור**: בהינתן רשימה של אובייקטים המתארים קבצים – עובר על כולם ומוחק אותם.  
הכוונה לקבצים בלבד. לא לספריות. אפשר להניח זאת; אין צורך לוודא זאת

**נדרש החל מתרגיל**: 1

**Read only**: false

**Inputs**:

* שם: FILES\_LIST

סוג: List המכילה אובייקטים מסוג File

תיאור: רשימת הקבצים למחיקה

מחרוזת משתמש: Files to delete

נחיצות: חובה

**Outputs**:

* שם: DELETED\_LIST

סוג: List המכילה אובייקטים מסוג String.

תיאור: רשימת כל הנתיבים המלאים של הקבצים שלא הצליחו להימחק. במידה והכל נמחק בהצלחה – רשימה ריקה.

מחרוזת משתמש: Files failed to be deleted

* שם: DELETION\_STATS

סוג: Mapping המכיל 2 Number'ים

תיאור: car: כמות הקבצים שנמחקו בהצלחה. Cdr: כמות הקבצים שנכשלו במחיקתם

מחרוזת משתמש: Deletion summary results

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to start delete <total files> files | לפני שמתחילה מחיקת הקבצים |
| 2 | Failed to delete file <file name> | עבור כל קובץ שלא מצליח להימחק |

**מקרים ותגובות**:

* במידה ו DELETED\_LIST הוא ריק (כלומר כל הקבצים נמחקו בהצלחה) ה step נגמר בהצלחה. DELETION\_STATS יכיל את סך הקבצים שנמחקו (car) ו 0 עבור סך הקבצים שנכשל במחיקתם (cdr).   
  שורת הסיכום של ה step צריכה להכיל תיאור הולם לסיטאציה.
* במידה וחלק מהקבצים לא הצליחו להימחק – ה step ייגמר ב Warning ושורת הסיכום תתאר את הסיטואציה. יש להוסיף שורת לוג מתאימה כרצונכם.
* במידה וכל הקבצים לא הצליחו להימחק – ה step ייגמר ב Failure ושורת הסיכום תתאר את הסיטואציה. יש להוסיף שורת לוג מתאימה כרצונכם.

## משנה השמות

**שם**: Files Renamer

**תיאור**: בהינתן רשימה של שמות קבצים משנה את השם של כולם על ידי הוספת prefix (תחילית) או suffix (סופית) כפי שמוגדר בקלט.  
את ה PREFIX יש להוסיף לתחילת שם הקובץ המקורי; את ה SUFFIX יש להוסיף לסוף השם של הקובץ המקורי (לא ל extension !).  
במידה ו PREFIX ו/או SUFFIX לא ניתנו – פשוט לא מבצעים את התפקיד שלהם.

מחזיר פירוט בתצורת טבלה המתאר את כל הקבצים עם שמם המקורי ואת שמם הסופי (שם בלבד, לא נתיב מלא)

**נדרש החל מתרגיל**: 1

**Read** **only**: false

**Inputs**:

* שם: FILES\_TO\_RENAME

סוג: List המכילה אובייקטים מסוג File

תיאור: רשימת הקבצים לשינוי שמם

מחרוזת משתמש: Files to rename

נחיצות: חובה

* שם: PREFIX

סוג: String

תיאור: מכיל את המחרוזת להוסיף לתחילת שם הקובץ.   
אפשר להניח כי יכיל רק תווים חוקיים לשימוש במסגרת שם קובץ

מחרוזת למשתמש: Add this prefix

נחיצות: אופציונלי. במידה ולא קיים פשוט מתעלמים ולא מבצעים את השינוי המוצע

* שם: SUFFIX

סוג: String

תיאור: מכיל את המחרוזת להוסיף לסופו של שם הקובץ.   
אפשר להניח כי יכיל רק תווים חוקיים לשימוש במסגרת שם קובץ

מחרוזת למשתמש: Append this suffix

נחיצות: אופציונלי. במידה ולא קיים פשוט מתעלמים ולא מבצעים את השינוי המוצע

**Outputs**:

* שם: RENAME\_RESULT

סוג: Relation

תיאור: טבלה המכילה את העמודות הבאות:

* מס' סידורי (ספרור רץ; מתחיל מ 1)
* שם הקובץ המקורי (שם בלבד. לא נתיב מלא)
* שם הקובץ לאחר השינוי (שם בלבד. לא נתיב מלא)

מחרוזת משתמש: Rename operation summary

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to start rename <total files> files. Adding prefix: <prefix>; adding suffix: <suffix> | לפני שמתחילה מלאכת שינוי הקבצים |
| 2 | Problem renaming file <file name> | עבור כל קובץ שלא מצליח לשנות את שמו |

**מקרים ותגובות**:

* במידה ו FILES\_TO\_RENAME הוא ריק ה step נגמר בהצלחה ו RENAME\_RESULT יהיה ריק.   
  שורת הסיכום של ה step צריכה להכיל תיאור הולם לסיטאציה.
* במידה והיה כשלון בהמרה של קובץ אחד או יותר (למשל אם הקובץ לא נמצא), ה step ייגמר ב Warning ושורת הסיכום תתאר את הסיטואציה בצירוף שמות כל הקבצים שנכשלו בהמרת שמם.  
  בטבלת הסיכום, "שם הקובץ לאחר השינוי" יהיה זהה ל"שם הקובץ המקורי" (אחרי הכל הוא לא שונה..)

## מחלץ המידעים I

**שם**: Files Content Extractor

**תיאור**: בהינתן רשימה של קבצי טקסט ומספר שורה רלוונטי, מחזיר חזרה את התוכן של השורה הרלוונטית.

**נדרש החל מתרגיל**: 1

**Read only**: true

**Inputs**:

* שם: FILES\_LIST

סוג: List המכילה אובייקטים מסוג File

תיאור: רשימת הקבצים להוציא מהם מידע

מחרוזת משתמש: Files to extract

נחיצות: חובה

* שם: LINE

סוג: Number

תיאור: מספר השורה לחילוץ המידע שלה. הספרור מתחיל מ 1.

מחרוזת למשתמש: Line number to extract

נחיצות: חובה.

**Outputs**:

* שם: DATA

סוג: Relation

תיאור: טבלה המכילה את העמודות הבאות:

* מס' סידורי (ספרור רץ; מתחיל מ 1)
* שם הקובץ המקורי (שם בלבד. לא נתיב מלא)
* המידה הטקסטואלי שאוחזר מתוך הקובץ בשורה הרלוונטית.   
  במידה והשורה לא נמצאה יוצג המידע: Not such line  
  במידה והקובץ לא קיים יוצג המידע: File not found

מחרוזת משתמש: Data extraction

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to start work on file <file name> | לפני שמתחילים לעבוד על כל קובץ ברשימה |
| 2 | Problem extracting line number <line number> from file <file name> | במידה ונתקלים בבעיה בשליפת המידע מהקובץ |

**מקרים ותגובות**:

* במידה ו FILES\_LIST הוא ריק ה step נגמר בהצלחה ו DATA יהיה ריק.   
  שורת הסיכום של ה step צריכה להכיל תיאור הולם לסיטואציה. הוסיפו שורת לוג כרצונכם.

## פולט הפורמטים: CSV

**שם**: CSV Exporter

**תיאור**: בהינתן מידע טבלאי כלשהוא, ממיר אותו לפורמט CSV (בתצורה של מחרוזת)  
קובץ CSV הוא קובץ טקסטואלי שבו השורה הראשונה מכילה שמות של עמודות המופרדות ב , (פסיק) ואחרי כן כל שורה מכילה אוסף של ערכים מופרדים ב , (פסיק) בהתאם לכמות העמודות.

**נדרש החל מתרגיל**: 1

**Read only**: true

**Inputs**:

* שם: SOURCE

סוג: Relation

תיאור: מידע טבלאי.

מחרוזת משתמש: Source data

נחיצות: חובה

**Outputs**:

* שם: RESULT

סוג: String

תיאור: תוצאת ההפעלה. התוכן של כל הטבלה כשהוא מומר לגרסה של CSV

מחרוזת משתמש: CSV export result

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to process <total lines> lines of data | לפני שמתחילים לעבוד על הטבלה |

**מקרים ותגובות**:

* במידה והטבלה ריקה מתוכן ה step ייגמר ב Warning ויש לתאר את המצב בשורת הסיכום, כמו גם שורת לוג מתאימה כרצונכם.  
  התוכן יכיל במקרה זה שורה אחת בלבד של שמות העמודות.

## פולט הפורמטים: Properties

**שם**: Properties Exporter

**תיאור**: בהינתן מידע טבלאי כלשהוא, ממיר אותו לפורמט של קובץ properties (בתצורה של מחרוזת).  
קובץ properties הוא קובץ טקסטואלי מהמבנה של key=value. עבור כל שורה i ועמודה j נגדיר את ה Key במבנה הבא: Row-i.<column-j-name> וה value יהיה למעשה ערך התא של row i ו column j.  
דוגמא: row-1.name=menash  
יש להציג את כל המידעים לפי סדר השורות (1,2,3 וכו') ובתוך כל שורה על פי סדר העמודות המוגדר בטבלה

**נדרש החל מתרגיל**: 1

**Read only**: true

**Inputs**:

* שם: SOURCE

סוג: Relation

תיאור: מידע טבלאי.

מחרוזת משתמש: Source data

נחיצות: חובה

**Outputs**:

* שם: RESULT

סוג: String

תיאור: תוצאת ההפעלה. התוכן של כל הטבלה כשהוא מומר לגרסה של properties

מחרוזת משתמש: Properties export result

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to process <total lines> lines of data | לפני שמתחילים לעבוד על הטבלה |
| 2 | Extracted total of <total properties> | אחרי שהסתיימה הפקת התוכן |

**מקרים ותגובות**:

* במידה והטבלה ריקה מתוכן ה step ייגמר ב Warning ויש לתאר את המצב בשורת הסיכום, כמו גם בשורת לוג כרצונכם.  
  במקרה זה התוכן יהיה string ריק.

## כותב הקובץ

**שם**: File Dumper

**תיאור**: מייצר קובץ טקסטואלי עם תוכן שניתן מבחוץ.

**נדרש החל מתרגיל**: 1

**Read only**: true

**Inputs**:

* שם: CONTENT

סוג: String

תיאור: מידע טקסטואלי שיש לכתוב אותו לקובץ

מחרוזת משתמש: Content

נחיצות: חובה

* שם: FILE\_NAME

סוג: String

תיאור: נתיב מלא לקובץ היעד שצריך להיווצר, כולל הסיומת הנדרשת (למשל .txt)

מחרוזת למשתמש: Target file path

נחיצות: חובה.

**Outputs**:

* שם: RESULT

סוג: String

תיאור: תוצאת ההפעלה. במידה ונכשל – סיבת הכשלון. במידה והצליח – SUCCESS

מחרוזת משתמש: File Creation Result

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to create file named <file name> | לפני שמתחילים לעבוד על הקובץ |

**מקרים ותגובות**:

* במידה והתוכן הוא string ריק, ה Step ייגמר ב warning. הקובץ ייווצר ריק מתוכן.   
  יש לציין זאת בשורת הסיכום של ה step, כמו גם בשורת לוג (עם תוכן כרצונכם).
* במידה ויש כשלון כלשהוא ביצירת הקובץ – ייגמר ב Failure.   
  יש לתאר את סיבת הכשלון בשורת הסיכום, כמו גם להוסיף לוג (עם תוכן כרצונכם) במקרה זה.

## רוכסן

**שם**: Zipper

**תיאור**: מבצע פעולה של un\zip לקובץ/ספרייה שמקבל.   
הפעולה תתבצע באמצעות ZipStream פנימי של ה JDK בלבד (אין להניח או להשתמש בכלים חיצוניים כמו winrar או tar לטובת הפעולה). הנה מספר מקורות מידע מועילים לסוגייה:

* [איך לעשות zip לקובץ בודד](https://makeinjava.com/compress-zip-files-using-zipoutputstream-java-example/)
* [איך לעשות zip לספריה](https://stackoverflow.com/a/6472872/1869409)
* [איך לעשות unzip לקובץ המכיל בפנים קבצים, ספריות וכו'](https://zetcode.com/java/zipinputstream/)

אם נבחרת פעולת ה UNZIP, אזי שאז ה source מצופה להיות נתיב לקובץ עם סיומת zip בלבד.   
הפתיחה תתבצע בו במקום, כלומר בספריה שבה ממקום קובץ ה zip.

אם נבחרת פעולת ה ZIP, אזי שאז ה source מכיל יעד לכיווץ שיכול להיות ספריה או קובץ פרטני.   
הקובץ המכווץ ייקרא באותו שם הספריה/קובץ שקיבלתם (בתוספת סיומת .zip) וימוקם גם הוא בספריה שבה נמצא ה SOURCE.

**נדרש החל מתרגיל**: 2

**Read only**: false

**Inputs**:

* שם: SOURCE

סוג: String

תיאור: נתיב מלא שעליו צריך לעבוד

מחרוזת משתמש: Source

נחיצות: חובה

* שם: OPERATION

סוג: Enumeration

תיאור: מתאר את סוג הפעולה הנדרשת לביצוע. האפשרויות הן ZIP או UNZIP.

מחרוזת למשתמש: Operation type

נחיצות: חובה.

**Outputs**:

* שם: RESULT

סוג: String

תיאור: תוצאת ההפעלה. במידה ונכשל – סיבת הכשלון. במידה והצליח – SUCCESS

מחרוזת משתמש: Zip operation result

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to perform operation <operation name> on source <source content> | לפני תחילת הפעולה |

**מקרים ותגובות**:

* במידה וה source לא קיים, או לא ניתן לעבוד עליו (נניח התקבלה הפניה לספרייה כשצריך לבצע unzip לקובץ) או שהפעולה עצמה נכשלת מכל סיבה - יש לסיים בכישלון, ולתעד זאת בשורת הסיכום כמו גם להוסיף לוג (עם תוכן כרצונכם) במקרה זה.

## שורת הפקודות

**שם**: Command Line

**תיאור**: מבצע הפעלה של פקודת מע' הפעלה פשוטה (command line). אוסף את הפלט של הפקודה ומפיק אותו כמחרוזת.

מומלץ להפעיל את הפקודה ולאסוף את הפלט באמצעות [ProcessBuilder](https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html).

**נדרש החל מתרגיל**: 2

**Read only**: false

**Inputs**:

* שם: COMMAND

סוג: String

תיאור: הפקודה לביצוע

מחרוזת משתמש: Command

נחיצות: חובה

* שם: ARGUMENTS

סוג: String

תיאור: מתאר אוסף של ארגומנטים שייתכן ויהיו נחוצים לטובת הפעלת הפקודה שהתקבלה

מחרוזת למשתמש: Command arguments

נחיצות: אופציונלי. אם לא ניתן פשוט לא מעבירים ארגומנטים לפקודה.

**Outputs**:

* שם: RESULT

סוג: String

תיאור: הפלט שניתן מהרצת הפקודה.

מחרוזת משתמש: Command output

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to invoke <command name> <arguments> | לפני תחילת הפעולה |

**מקרים ותגובות**:

* Step זה תמיד מסתיים בהצלחה, גם אם תוכן הפקודה נכשל, וגם אם טכנית הפקודה נכשלה מאחר ולא הייתה קיימת במע'.

## משגר הבקשות

**שם**: HTTP Call

**תיאור**: שיגור בקשת HTTP אל יעד מסוים והחזרת תוצאתה.

**נדרש החל מתרגיל**: 3

**Read only**: false

**Inputs**:

* שם: RESOURCE

סוג: String

תיאור: שם המשאב מולו עובדים, כולל query parameters במידה וישנם.

מחרוזת משתמש: Resource Name (include query parameters)

נחיצות: חובה

* שם: ADDRESS

סוג: String

תיאור: שם מלא של ה domain + Port שמולו עובדים. המחרוזת תתקבל בתצורה של domain:port

מחרוזת משתמש: Domain:Port

נחיצות: חובה

* שם: PROTOCOL

סוג: Enumeration

תיאור: פרוטוקול לשימוש. סט סופי של ערכים. יכול להכיל: http או https בלבד (אותיות קטנות)

מחרוזת משתמש: protocol

נחיצות: חובה

* שם: METHOD

סוג: Enumeration. יכול להכיל: GET, PUT, POST, DELETE

תיאור: ה http verb שיש להשתמש

מחרוזת משתמש: Method

נחיצות: אופציונלי. במידה ולא מתקבל יש להתייחס אליו עם הערך GET כברירת מחדל

* שם: BODY

סוג: Json

תיאור: אובייקט בתצורת Json שניתן במתכונת של String

מחרוזת למשתמש: Request Body

נחיצות: אופציונלי. במידה ולא מתקבל פשוט לא מעבירים שום body

**Outputs**:

* שם: CODE

סוג: Number

תיאור: סטטוס קוד של תשובת הבקשה

מחרוזת משתמש: Response code

* שם: RESPONSE\_BODY

סוג: String

תיאור: תוצאת הבקשה

מחרוזת משתמש: Response body

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | About to invoke http request <request details:  PROTOCOL | VERB | DOMAIN | RESOURCE> | לפני שמשגרים את הבקשה |
| 2 | Received Response. Status code: <status code> | עם קבלת ה response |

**מקרים ותגובות**:

* במידה ולא ניתן בכלל להגיש את הבקשה מכל סיבה שהיא – ייגמר בכשלון.   
  שורת הסיכום תכיל את המידע הרלוונטי, כמו גם שורת לוג כרצונכם.
* במידה והבקשה הוגשה, גם אם תוצאתה היא "כשלון" (XX4 או XX5) – עדיין ייגמר בהצלחה (והמידע על הכשלון הלוגי יגיע כחלק מה RESPONSE).

## המג'ייסן

**שם**: To Json

**תיאור**: אחראי לבדוק ולהפוך תוכן טקסטואלי לאובייקט מסוג JSON

**נדרש החל מתרגיל**: 3

**Read only**: true

**Inputs**:

* שם: CONTENT

סוג: String

תיאור: תוכן המידע הטקסטואלי שיש לבדוק ולהפוך ל JSON במידת האפשר

מחרוזת משתמש: Content

נחיצות: חובה

**Outputs**:

* שם: JSON

סוג: Json

תיאור: אובייקט המתאר json

מחרוזת משתמש: Json representation

**לוגים**:

|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | Content is JSON string. Converting it to json… | במידה ומסתבר שהתוכן ניתן להפיכה ל json |
| 2 | Content is not a valid JSON representation | במידה ומסתבר שהתוכן אינו ניתן להפרה ל json |

**מקרים ותגובות**:

* במידה והתוכן אינו ניתן להמרה ל json ה step ייכשל, ושורת הסיכום תתעד את המצב. כמו כן הודעת לוג מתאימה תינתן בהקשר זה.

## מחלץ המידעים II

**שם**: Json Data Extractor

**תיאור**: מאפשר לחלץ פיסות מידע מתוך מבנה של json.   
פיסות המידע המחולצות יהיו תמיד בתצורה של String (גם אם טכנית מתארות json בעצמן).

חילוץ המידע יעשה באמצעות עבודה עם [json path](https://github.com/json-path/JsonPath). הורידו מהמאמא את כלל ה jar'ים המשמשים לעבודה איתו

**נדרש החל מתרגיל**: 3

**Read only**: true

**Inputs**:

* שם: JSON

סוג: Json

תיאור: אובייקט המתאר מידע בפורמט של Json חוקי ותקין

מחרוזת משתמש: Json source

נחיצות: חובה

* שם: JSON\_PATH

סוג: String

תיאור: מחרוזת json path המתארת את המידע שצריך לחלץ. המחרוזת יכולה להכיל סדרה של ביטויים מופרדים בסימן ה "|". כל מחרוזת מתחילה בסימן "$" (זהו חלק מה json path).

מחרוזת משתמש: data

נחיצות: חובה

**Outputs**:

* שם: VALUE

סוג: String

תיאור: מחרוזת המתארת את המידע שחולץ מאובייקט המקור. ייתכן ויחולצו מספר פרטי מידע (כולם מסוג של string). במקרה זה יש להציג אותם משורשרים עם , באמצע

מחרוזת משתמש: Data value

**לוגים**:

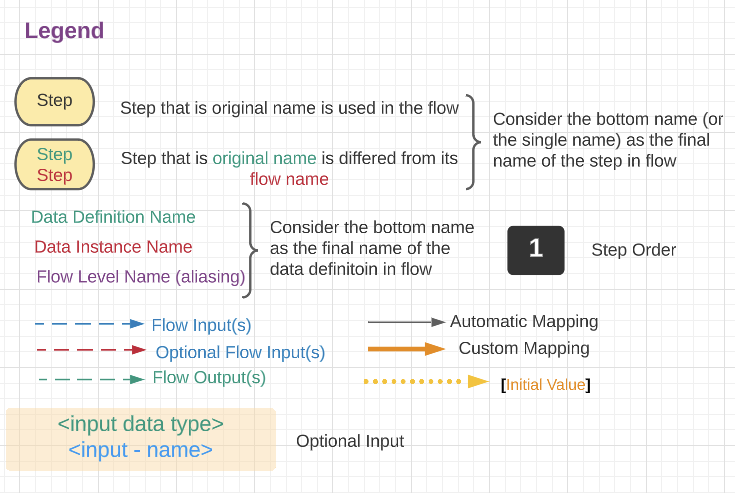
|  |  |  |
| --- | --- | --- |
| # | שורת לוג | מתי |
| 1 | Extracting data <json path>. Value: <extracted value | לאחר שמחלצים מידע מתוך ה json. |
| 2 | No value found for json path <json path> | במידה וה Json path לא הניב שום מידע |

**מקרים ותגובות**:

* במידה וקרתה תקלה כלשהיא – ייגמר בכשלון. שורת הסיכום תכיל את המידע כמו גם שורת לוג מתאימה.
* במידה ולא נמצא שום מידע מתאים לפי ה json path שהתקבל – עדיין נגמר בהצלחה.

נספח ג' – דיאגרמות של flow (דוגמא להמחשה)

הדיאגרמות הבאות מציגות את מבנה ה flow'ים המתואר בקובץ ex1.xml  
הנה המקרא לביאור הסימונים בדיאגרמות:



Flow: **Deleted Matched Files**

Diagram

Description automatically generated

Flow: **Rename Files**

Diagram

Description automatically generated

נספח ד' – תיאור מבנה המערכת באמצעות קובץ XML

תיאור המע' ומרכיביה בתרגילים השונים נתון באמצעות קובץ XML.

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

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

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

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

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

Text

Description automatically generatedמבנה המע' מאוגד תחת האלמנט **ST-Stepper**:

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-Stepper | אלמנט ראשי זה מכיל את כלל פרטי ה Flow'ים המוגדרים במע' |
| 2 | Element | ST-Flows | מכיל רשימה של אובייקטים המתארים Flow'ים במע'. |

Text

Description automatically generatedהאלמנט **ST-Flow** מתאר ייצוג של Flow אחד במע':

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-Flow | מתאר Flow אחד במע' |
| 2 | Attribute | name | מתאר את שם ה Flow. לכל flow שם ייחודי משלו.  השמות הם case insensitive. |
| 3 | Element | ST-FlowDescription | מתאר תיאור מילולי של flow |
| 4 | Element | ST-FlowOutput | מתאר את שמות המידעים (data names) המוגדרים כתוצרים הפורמלים של flow זה. השמות הם השמות הסופיים של המידעים, אחרי flow level aliasing במידה והוגדרו כאלה. |
| 5 | Element | ST-StepsInFlow | מתאר את רשימת כל ה step'ים המשתייכים ל flow זה |
| 6 | Element | ST-FlowLevelAliasing | מתאר את אוסף ה flow level aliasing המוגדרים במסגרת flow זה.  זהו אלמנט אופציונלי וייתכן ולא יופיע כלל אם אין בו צורך |
| 7 | Element | ST-CustomMappings | מתאר את אוסף המיפויים הידניים המוגדרים במסגרת flow זה. זהו אלמנט אופציונלי וייתכן ולא יופיע כלל אם אין בו צורך |

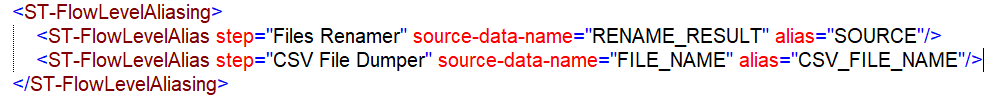
האלמנט **ST-StepsInFlow** מכיל אוסף של אלמנטים מסוג של **ST-StepInFlow**:

A picture containing logo

Description automatically generated

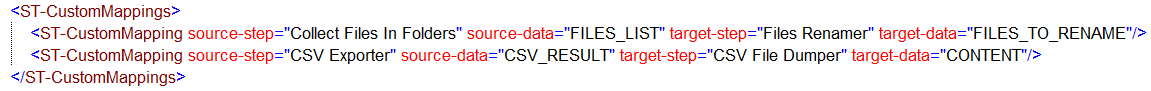
|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-StepInFlow | מתאר מופע אחד של step במסגרת ה flow. סדר הופעת האלמנטים האלה קובע את סדר הרצת ה step'ים ב flow. |
| 2 | Attribute | name | מתאר את השם הפורמלי של ה step, כפי שהוגדר במסגרת התרגיל.  במידה ולא ניתן alias, זה יהיה שמו הסופי של ה step במסגרת flow זה בכל המקומות שבהם יש צורך להתייחס ל step. |
| 3 | Attribute | alias | מתאר שם/כינוי שניתן ל step זה במסגרת הגדרת ה flow. זהו השם הסופי של ה step במסגרת flow זה בכל המקומות שבהם יש צורך להתייחס ל step זהו attribute אופציונלי. במידה ולא קיים שמו הסופי של ה step יוכרע לפי ה name |
| 4 | Attribute | continue-if-failing | מתאר האם יש צורך להמשיך את ביצוע ה flow במידה ו step זה מסיים עם כשלון (FAILURE). ערך בוליאני.  זהו attribute אופציונלי. ברירת המחדל היא false – כלומר ביצוע ה flow צריך להיעצר במידה ו step זה נכשל. |

האלמנט **ST-FlowLevelAliasing** מתאר אוסף של הגדרות מתן שמות ספציפים למידעים השונים שה step'ים מקבלים/מוסרים (input\output) במסגרת flow זה בלבד.   
הגדרת שיום בודד מוגדרת תחת האלמנט **ST-FlowLevelAlias**. אין משמעות לסדר ההגדרות.



|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-FlowLevelAlias | מתאר מופע אחד של הגדרת שיום במסגרת ה flow |
| 2 | Attribute | step | שם ה Step אשר לאחד המידעים שלו (Input\output) יש לשנות את השם במסגרת flow זה.  שם ה step יינתן במתכונתו הסופית במסגרת flow זה (אחרי alias במידה והוגדר לו) |
| 3 | Attribute | source-data-name | מתאר את השם המקורי של פרט המידע (input\output) של ה step המדובר, כפי שהוגדר פורמלית במסגרת הגדרת ה step, אשר רוצים לשנות את שמו במסגרת הגדרת flow זה. |
| 4 | Attribute | alias | מתאר את השם החדש שיינתן לפרט המידע המדובר. |

האלמנט **ST-CustomMappings** מתאר אוסף של מיפויים ידניים המוגדרים במסגרת flow זה.  
מיפוי בודד יוגדר תחת האלמנט **ST-CustomMapping**. אין משמעות לסדר ההגדרות.



|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-CustomMapping | מתאר מופע אחד של מיפוי ידני במסגרת ה flow. |
| 2 | Attribute | source-step | שם Step המקור אשר רוצים למפות את אחד התוצרים שלו (output)  שם ה step יינתן במתכונתו הסופית במסגרת flow זה (אחרי alias במידה והוגדר לו) |
| 3 | Attribute | source-data | מתאר את השם הסופי (אחרי alias, אם הוגדר) של פרט המידע (output) של step המקור, אשר רוצים למפות אותו ל input של step אחר במסגרת הגדרת flow זה. |
| 4 | Attribute | target-step | שם Step היעד אשר רוצים למפות אל אחד הקלטים שלו (input). שם ה step יינתן במתכונתו הסופית במסגרת flow זה (אחרי alias במידה והוגדר לו) |
| 5 | Attribute | target-data | מתאר את השם הסופי (אחרי alias, אם הוגדר) של פרט המידע (input) של step היעד, אשר רוצים למפות אותו ל output של step המקור במסגרת הגדרת flow זה. |

## סכמה תרגיל 2

סכמה זו מכילה 3 שינויים מרכזיים:

**גודל ה threadpool:**אלמנט המכיל מידע על גודל ה threadpool לשימוש במסגרת הרצת ה flow'ים.  
  
נוספו 2 אלמנטים חדשים להגדרת flow בודד (במסגרת האלמנט **ST-Flow**)

**A picture containing text

Description automatically generatedST-Continuations**מתאר את אוסף ה continuations שניתן לבצע לאחר ביצוע flow זה.  
זהו אלמנט אופציונלי ואינו חייב להופיע אם אין בו צורך.  
הוא יכיל אוסף של **ST-Continuation**:

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-Continuation | מתאר מופע אחד של continuation ל flow אחר המוגדר במסגרת ה flow. |
| 2 | Attribute | target-flow | שם flow היעד אליו אפשר להמשיך לאחר ביצוע flow זה |
| 3 | Element | ST-Continuation-Mapping | מתאר מיפוי ידני מפורש שיש לבצע בין אחד המידעים הקיימים במסגרת הרצת flow זה, אל אחד הקלטים הנדרשים ב flow היעד. זהו אלמנט אופציונלי ואינו חייב להופיע |
| 4 | Attribute | source-data | מתאר את השם הסופי (אחרי alias, אם הוגדר) של פרט המידע ב flow המקור אשר את תוכנו יש להעתיק אל פרט מידע אחר ב flow היעד |
| 5 | Attribute | target-data | מתאר את השם הסופי (אחרי alias, אם הוגדר) של פרט המידע (input) של flow היעד, אשר אליו צריך להעתיק את תוכנו של המידע ב flow המקור |

**ST-InitialInputValues**   
מתאר את אוסף המידעים ההתחלתיים שיש להגדיר עבור הקלטים השונים של ה flow במסגרת הרצתו.  
זהו אלמנט אופציונלי ואינו חייב להופיע אם אין בו צורך.  
הוא יכיל אוסף של **ST-InitialInputValue**:

A picture containing logo

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ST-InitialInputValue | מתאר מופע אחד של ערך התחלתי שיינתן במסגרת הרצת flow זה |
| 2 | Attribute | input-name | מתאר את השם הסופי (אחרי alias, אם הוגדר) של פרט מידע (input) ב flow, אשר צריך להגדיר את ערכו אוטומטית במסגרת הרצתו |
| 3 | Attribute | initial-value | ערכו הסופי של פרט המידע אשר יוגדר במסגרת הרצתו |

נספח ה' – תרשים סכמות XML

## סכמה תרגיל 1

Diagram

Description automatically generated

## סכמה תרגיל 2

Diagram, schematic

Description automatically generated