פיתוח תוכנה מבוסס JAVA

קיץ 2023 - Predictions

מס' גרסה:2

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

מרצה: **אביעד כהן** aviadco@mta.ac.il
בודק: **איתי כהן** itaych@mta.ac.il

תוכן העניינים

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 חובה להגיש ביחידים.

3. בעבודה משותפת על תרגיל יש להקפיד על מעורבות אקטיבית של כלל המגישים בכל חלקי התרגיל.

4. במידה והוגדר בונוס לתרגיל מסוים, ציון הבונוס יתווסף לציון התרגיל בלבד (ולא לציון הסופי של הקורס כולו).

5. התרגילים יוגשו דרך מערכת Mama. מוגדר רכיב 'מטלה' נפרד לכל תרגיל.

6. לפני שליחת התרגיל יש לבדוק שהוא עובד ומכיל את הקבצים המעודכנים ביותר, על מערכת "נקייה".
בצעו את סט הפעולות שאתם מצפים מן הבודק לבצע וודאו כי הכל מתנהל כראוי וכסדרו.

7. ניתן להחליף את השותפ/ה בכל תרגיל, ללא צורך באישור או הודעה למרצה.
הניקוד על כל תרגיל נזקף לזכות הסטודנטים שבצעו אותו בלבד.

8. 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.

9. 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.

4. עומס בלימודים, בעבודה, בחיים, בגלל הילדים או ההורים, שכנים וחברים (או בכל תחום אחר) אינו נחשב כסיבה לגיטימית לבקשת הארכה.

5. במקרה של בקשה להארכה (מכל סיבה שהיא, לרבות מילואים ומחלה) יש לפנות למרצה מראש על מנת לקבל אישור.
הפנייה תתבצע במייל.

6. לאנשי הקבע – הישארות של שבת בבסיס אינה נחשבת כסיבה לגיטימית להארכה (מכיוון שזה חלק מהסדר העבודה בצה"ל);
יציאה לאבט"ש כן נחשבת כמילואים ויש להגיש אישור ממפקד הבסיס.

7. ניתן לערער על ציון של תרגיל 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 <u>אינה כוללת</u> את סביבת הפיתוח אלא אך ורק <u>הרצה ידנית מ cmd</u> (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.

2. יש להגיש את התרגילים בתור קובץ rar/zip (לא 7Z !)
הקובץ יכיל:
   1. כל הקבצים הרלבנטים להפעלת התרגיל (war/jar – פרטים בגוף התרגיל).
   2. קובץ אצווה (== batch) שיכיל את הפקודה שמריצה את התרגיל.
   3. קובץ readme שיכיל את פרטי המגיש/ים, כמו גם הנחיות כלליות להרצה התרגיל וכל הנחות שלקחתם במהלך התרגיל ואתם סבורים שחשוב כי הבודק יכיר. דמיינו כי בכל שאלה/תקלה שיתקל בהן הבודק, יעמוד לרשותו רק קובץ ה readme שלכם. דאגו להבהיר ולהסביר את כל הדברים שיכולים להשתבש ו/או 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 ה readme להיות בפורמט word או pdf (**<u>לא notepad !</u>**).
   חי נפשי – אם מישהו מגיש readme כקובץ טקסט פשוט - ירד לו ניקוד מהתרגיל...

3. דווקא בגלל שאין זהו קורס שבו יכנסו לנבכי הקוד ויבדקו כל שורה ושורה, יש להקפיד ביתר שאת על קוד נקי, מסודר, קריא ויעיל. בפרט:
   4. הימנעו משכפול קוד
   5. פונקציות ארוכות מדי (בדר"כ יותר מגודל עמוד)
   6. בחירת שמות גרועים למחלקות, לפונקציות ולמשתנים
   7. הזחה (אינדנטציה) נכונה
   8. imports מיותרים
   9. יש להקפיד להשתמש ב-modifiers בצורה נבונה:
      10. מחלקה שלא אמורים לבנות אובייקטים שלה אמורה להיות מוגדרת כ-abstract
      11. קבועים יש לסמן כ-final
      12. משתנים של המחלקה רצוי להגדיר כ-private אלא אם יש סיבה לגיטימית לבחירה אחרת.
   13. 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.

14. 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...)

15. 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

16. 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:

17. 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 linux/mac – 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: 29.7.23

כללי

מועד הגשה: 29.7.2023 **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 5-6 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();
    }
}
```

Predictions

מוטיבציה

כאשר רוצים להציע הצעת חוק או לבצע שינוי מהותי בהתנהלות המדינה אשר יכול להשפיע על רבים מהאזרחים, פונים למכון המחקר של הכנסת כדי שזה יבצע ניתוח למצב הקיים ויציע את התחזית שלו לשינוי בהתאם או כתלות בהחלת הצעת החוק החדשה.

למכון המחקר של הכנסת כלים רבים לבצע את הניתוח הנדרש ואת ההבנה של "מה יכול לקרות" או "מה תהיה ההשפעה" של מרכיב כזה או אחר בהצעת החוק על פני המציאות.
במקרים מסויימים הוא מבצע סקרים, במקרים אחרים הוא משווה נתונים מספריים, אבל הרבה פעמים הוא גם מבצע סימולציות שונות של המצב הקיים, ובודק כיצד הסימולציות משתנות בהתאם לתנאים ולשינויים המתוארים בהצעת החוק ומנסה לבצע חיזוי של העתיד עם החלת החוק.

כל סימולציה, או תיאור של מצב קיים, מורכבת ומושתתת על פרמטרים רבים, אשר מקיימים בינם לבין עצמם קשרי גומלין.
למשל: אם נעלה את המחיר על סיגריות – האם זה יגרום לאנשים להפסיק לעשן ? (בדקו את זה. זה לא עזר. ☹️)
הקשרים בין הפרמטרים האלה הם לרוב סטטיסטים, לא דטרמיניסטים ויש מידה רבה ובלתי מבוטלת של "הימור" בנסיון להניע את הסימולציה ולהסתמך על פלטיה ותוצאותיה.

דוגמאות לחוקים ולסימולציות אפשריות:

- אם נעלה את המס על שתייה מתוקה, אז ישתו פחות משקאות שכאלה. כפועל יוצא יהיו פחות מקרים של סכרת וכך נוזיל את עלויות הטיפול באזרח ונוריד את העומס על מע' הבריאות.
  בסימולציה אפשרית, בהינתן שיש קשר/מתאם סטטיסטי בין כמות המשקאות לפוטנציאל התפתחות סכרת, כמו גם ידע לגבי כמה מחלת הסכרת "מעמיסה" על מע' הבריאות, הסימולציה תבדוק מה באמת ההיתכנות וכמה מס צריך להעלות כדי לגרום למצב שיוריד את הצריכה של משקאות מתוקים כך שבאמת היא תתורגם, בבוא העת, להפחתת העומס על מע' הבריאות.

- אם נייצר קו רכבת ישיר ממטולה לתל אביב זה יקל על אנשים הגרים שם להגיע למרכז ולהנות מאפשרויות התעסוקה המגוונות.
  בסימולציה אפשרית נוכל לבדוק על סוגי אוכלוסיות מגוונות, אשר כל אחת מהן תבחר להשתמש ולנצל את האפשרות הנ"ל לטובתה במידה מסויימת, כמה באמת קיום של קו שכזה יעלה את שיעור התעסוקה במטולה כעבור פרק זמן מסויים.

בתרגיל זה נדמה מטה-מערכת גנרית היכולה לסייע בהגדרה ובהפעלה של סימולציה שכזו: **Predictions**
המע' תיבנה בשלושה חלקים:
חלק א' – מימוש Predictions כמע' בסיסית; תפעול באמצעות ממשק console.
חלק ב' – העשרת Predictions ביכולות נוספות כמו גם פיתוח ותפעולה באמצעות ממשק גרפי מבוסס Java FX.
חלק ג' – הפיכת Predictions לשרת המאפשר גישה מרחוק למס' סוגי משתמשים.

Predictions – תיאור המע'

מטרת המערכת היא לאפשר הגדרה של "עולם" שבו מתקיימת אוכלוסיה, ולהחיל על אוכלוסיה זו סט של "חוקים" אשר משנים בצורה כזו או אחרת (סטטיסטית) את חלק מהיישויות שבעולם.

המערכת היא מערכת גנרית אשר תאפשר להגדיר את היישויות השונות ואת החוקים שחלים עליהם ובכך למעשה תהווה תשתית וקרקע פורייה לשימוש במספר סימולציות ככל שנדרש.

המערכת מורכבת מכמה חלקים מרכזיים:

World

רכיב זה הינו הרכיב הראשי והמתכלל של סימולציה אחת במערכת.
במסגרת "העולם" של הסימולציה יוגדרו מספר סוגים שונים של יישויות (entity).
כל יישות תופיע כמות נתונה של פעמים. סך האוכלוסיה של העולם היא סך כל מופעי היישויות.

במסגרת הסימולציה נתאר ציר זמן מתקדם. ציר הזמן יתקדם בפסיעה (tick) אחת בכל פעם.
המשמעות האמיתית של משך הזמן של פסיעה אחת לא רלוונטית. זה יכול להיות שניה, חודש, שבועיים – הכל במסגרת הגדרת הסימולציה.
לאורך הסימולציה ציר הזמן ינוע והרכב האוכלוסיה ופרטיה ישתנו בהתאם לחוקים השונים המתקיימים בעולם.
ציר הזמן יתחיל מ 0.

העולם יכול להכיל גם מיקומים פיזים של היישויות המתקיימות בו.
במקרה זה בכל פסיעה של ציר הזמן היישויות השונות יוכלו לנוע במרחב (מרחב דו מימדי) ויחסי הגומלין בינהן יכולים להיות מוגדרים ומושפעים באמצעות המרחק/הקרבה שלהן ליישויות אחרות.

Entity

יישות המייצגת פרט אחד בעולם.
ליישות יש שם, כמו גם הגדרה של כמה מופעים שלה מתקיימים בעולם.
לכל יישות ניתן יהיה להגדיר מספר מאפיינים (properties) שעל פיהם היא נבדלת מרעותה.
כל מאפיין יוגדר באמצעות ההגדרות הבאות:

- שם ייחודי
- סוג: מספר שלם, מספר ממשי, ערך בוליאני, מחרוזת תווים
- מגבלות על הערכים האפשריים של המאפיין (אופציונלי)
- מתן ערכים התחלתיים

(עקבו אחר נספח ג' המתאר את הפרטים הנ"ל בפרוטרוט)

חוקים, פעולות, פונקציות

חוקים הם סט של הגדרות והוראות הפועלות על היישויות השונות המתקיימות בעולם.
המטרה של חוק (ופעולותיו) בסופו של דבר היא לשנות במידה מסויימת חלק מהמאפיינים של היישו(יו)ת עליה הוא מוחל.

לכל חוק יש שם ייחודי.

חוק מכיל "פעולות" (actions) לביצוע.
כל פעולה תגדיר יישות מרכזית עליה/במסגרתה היא עובדת, ותוכל לבצע שינוי כזה או אחר במי מהמאפיינים שלה.
(בהמשך נאפשר גם להגדיר יישות משנית במסגרת הפעלת פעולה ולייצר פעולה הפועלת על כמה יישויות)
ישנן פעולות פשוטות (כמו שינוי ערכו של מאפיין) וישנן פעולות מורכבות כמו הגדרת תנאים, הרג היישות, בריאת יישות חדשה וכו'.
המערכת תתמוך במגוון סוגים של פעולות לביצוע (ראו נספח א' לפירוט כלל סוגי הפעולות).
במידה ויינתנו מספר פעולות במסגרת חוק יש לבצע אותן על פי סדר הגדרתן.

המערכת תתמוך בנוסף במגוון של פונקציות עזר אשר מאפשרות למגדיר הסימולציה להתממשק עם המערכת ולקבל ממנה מגוון מידעים. (ראו נספח ב' לפירוט הפונצקיות הנתמכות)
את הפונצקיות השונות ניתן יהיה לשלב במסגרת הפעולות המוגדרות

לכל חוק יוגדר מרכיב של תזמון הפעלה (activation).
זהו רכיב אשר יקבע מתי והאם החוק מופעל ומוחל על היישויות בעולם.
ה activation מוגדר באמצעות שני מרכיבים:

1. Ticks – מספר שלם; כל כמה פסיעות של ציר הזמן עליו לפעול.
   ערך ברירת המחדל הוא 1, כלומר בכל התקדמות של ציר הזמן.
   (ציר הזמן מתחיל מ 0, והערך הנ"ל מגדיר את הקפיצות בציר הזמן לטובת ההפעלה.
   דוגמא: אם tick = 2 אזי הוא יופעל ב 2,4,6 וכו'..)

2. Probability – מספר עשרוני בין 0 ל 1: באיזו הסתברות על החוק לפעול.
ערך ברירית המחדל הוא 1, כלומר פועל תמיד כשהוא מתאפשר. 0 פירושו לא פועל בכלל. כל מספר באמצע הוא ההסתברות לפעולתו.

משתני סביבה חופשיים (environment)

המערכת תגדיר אוסף של משתני סביבה כלליים התקפים וזמינים לשימוש לכלל היישויות/חוקים המוגדרים בעולם.
משתני הסביבה הנ"ל מהווים משתנים חופשיים שערכים שונים שלהם יכולים להשפיע על התנהלות הסימולציה לכאן או לכאן.
במסגרת הפעלת הסימולציה המשתמש יוכל לשנותם ובכך לבחון איך תנאי התחלה שונים מובילים לתוצאה שונה.

משתני הסביבה יוגדרו כאוסף של Properties במע', בדומה ל Properties המוגדרים עבור כל יישות.

תנאי סיום

המערכת תגדיר אוסף של תנאי סיום שיגרמו לסימולציה לעצור ולהסתיים.
תנאי הסיום האפשריים:

1. לפי זמן: בתנאי זה נגדיר כמה שניות נרצה שהסימולציה תרוץ
2. לפי פסיעות (ticks): בתנאי זה נגדיר כמה ticks נרצה שהמע' תרוץ
3. אינטרקטיבי על פי החלטת המשתמש: בתנאי זה המשתמש יחליט מתי הסימולציה נגמרת.

במידה והוגדרו כמה תנאי סיום – הסימולציה תסתיים ברגע שלפחות תנאי אחד מבינהם מתקיים.

מהלך הסימולציה

הסימולציה תנסה לנבא מה יקרה לאוכלוסיית העולם לאורך הזמן ובהתאם לחוקים השונים המוגדרים בו.

תהליך האתחול

ב tick=0 כלל היישויות יאותחלו, כמו גם במשתני הסביבה החופשיים.
לכל יישות מוגדר כמה מופעים שלה צריכים להתקיים בתחילת הסימולציה.

תהליך יצירת ערכים עבור properties:
הן משתני הסביבה החופשיים (אלה שהמשתמש לא מעניק להם ערכים) והן ה properties של היישויות השונות יכולים להיות מאותחלים בצורה ידנית או אוטומטית.
אם בצורה ידנית – ערך האתחול יינתן בצורה מפורשת במסגרת קובץ ה XML.
אם בצורה אוטומטית הכוונה היא ליצירת ערכים רנדומלים. הנה הדרכים השונות לבצע זאת, על פי סוג המשתנה/property:

- ערכים מספריים (שלמים או מספריים): יאותחלו רנדומלית על פי האלמנט PRD-range שיינתן ב XML.
- ערכים בולאנים יאותחלו רנדומלית ל true או false (אלה האפשרויות היחידות האפשריות בהקשר זה...)
- ערכי מחרוזת (String) יאותחלו רנדומלית בצורה הבאה:
  1. יש להגריל אורך רנדומלי של מחרוזת. אורך מחרוזת יכול להיות כל מספר בין 1 ל 50, כולל.
  2. יש להגריל תווים כאורך המחרוזת הרנדומלי שנקבע בשלב הקודם.
     התווים החוקיים לשימוש הם:
     - אותיות ה abc: a – z ; A - Z
     - ספרות (0-9)
     - סימני הפיסוק הבאים: **()-._?!**
     - תו הרווח

תהליך התקדמות הסימולציה

בכל פסיעה של הזמן (בכל tick), יופעלו כל החוקים ברי ההפעלה.
חוק ייחשב בר הפעלה אם מתקיימים שני התנאים הבאים גם יחד:

1. על פי הגדרת ה activation שלו הגיע מועד הפסיעה להפעלתו.
2. על פי הגדרת ה activation שלו, הוגרל ערך שעומד בהגדרת ההסתברות שלו.
   ערך ה Probability מעיד על ההסתברות להפעלתו של החוק. יש להגריל מספר בין 0 ל 1 (כולל). אם הוא קטן מערך ה probability הרי שאז מתקיימת ההסתברות להפעלתו.

עבור כל חוק בר הפעלה, על פי סדר הגדרתם בקובץ הגדרת הסימולציה, יש להפעיל את כלל הפעולות המוגדרות בו (על פי סדר הגדרתן).

יש לרוץ על כלל מופעי היישויות במע', ולהפעיל את הפעולות בחוקים ברי ההפעלה על כל מופע ומופע על פי המוגדר במסגרתה.
כל פעולה יכולה לבצע במופע היישות (כלומר ב properties שלו) שינויים כאלה ואחרים.
המע' תתמוך במגוון של פעולות אפשריות (ראו נספח א').

תהליך הסיום

את סדר הפעולות הנ"ל מבצעים עבור כל פסיעה ופסיעה, כפי שמוגדר במסגרת הגדרת העולם, עד להתקיימות מי מתנאי הסיום של הסימולציה.

ייתכן כי במהלך הסימולציה יתגלו שגיאות מגוונות:

- פנייה או הסתמכות על שם של entity שלא קיים
- פנייה או הסתמכות על שם של property של entity שלא קיים
- נסיון להשתמש או להמיר ערכים מסוגים שונים שלא על פי המוגדר בנספח ב'.

ועוד...

יש לממש מנגנון כללי לתפיסת שגיאות המתרחשות במהלך הסימולציה.
כאשר שגיאה מתרחשת יש לעצור את הסימולציה ולספק למשתמש הסבר מלא לגבי אופי השגיאה, מתי ומה ההקשר שבו היא קרתה (במסגרת איזה חוק/פעולה; על איזה יישות/מאפיין; בנסיון הפעלה של פונקציה וכו') וזאת כדי שלמשתמש יהיה מושג לגבי מה ואיפה הוא צריך לתקן במסגרת הגדרת הסימולציה.

תוצרי הסימולציה

בגמר הפעלת הסימולציה, יתאפשר למפעיל לקבל מגוון מידעים:

- סקירה על כמות וסוגי היישויות
- בדיקת ממוצע ערכו של מאפיין של יישות מסויימת לאורך זמן
- בדיקת עקביות ערכו של מאפיין של יישות מסויימת לאורן זמן

ועוד...

הנחיות ספציפיות למימוש תרגיל Predictions

1. המטרה היא לבנות מנוע מערכת גנרי, כזה שידע לקבל את הפרטים לגבי אופי המע' (הגדרת סימולציה, כלומר פרטי העולם) 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.

2. כחלק מהמע' תצטרכו לחשוב ולבחור לבד את מבני הנתונים השונים שישרתו את הצרכים של דרישות המע'. זהו לא קורס במבני נתונים או באלגוריתמים, ומבני הנתונים/אלגוריתמים שתבחרו לממש לא חייבים להיות היעילים ביותר או האופטימליים. מספיק שהם יעבדו בצורה **נכונה** (ללא טעויות) ובזמן סביר.

3. ממשקי המשתמש השונים יפעלו מול מנוע המע' אשר יפותח מתרגיל לתרגיל בהתאם לדרישות.
המנוע יכיל, בין היתר, את הגדרות והרצות הסימולציה וכו'

4. כל הממשקים הפונים אל המשתמש החיצוני ומציגים לו מידע ו/או מבקשים ממנו מידע מספרי, חייבים להיות מבוססי ספירה המתחילה מ 1. גם אם פנימית אתם מממשים את מי מהרכיבים כמערך או רשימה (אז בסיס הספירה מתחיל מ 0) – עליכם להקפיד ולודא כי כלפי חוץ "תדברו" אך ורק במונחים של בסיס 1.

5. המע' כולה תתואר בשפה האנגלית בלבד, עם ממשק משתמש המתנהל משמאל לימין (במקומות הרלבנטים)

6. **המע' כולה תיכתב ותורץ בסביבת העבודה של ג'אווה 8.**

פרטים יבשים

צפי תחילת עבודה: **26.7.23**
תאריך הגשה: **17.8.23**
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 Predictions.
את התשתית "תתפעלו" באמצעות ממשק console פשוט המציג תפריט פקודות שדרכו מפעילים את המע'.
בתרגיל תממשו מספר סוגים של פעולות ופונקציות (כפי שמוגדר בנספח א' וב'), תגדירו את היישות הגנרית במערכת, תאפשרו הרצה של סימולציה והצגת תוצאותיה במתכונת מצומצמת.
כל אלה יישבו בתוך "מנוע" המערכת, אשר ידע להגיב לכל פנייה המגיעה משכבת ממשק המשתמש, לעבד את הקלט ולהחזיר פלט רלבנטי.

3. יש לוודא תקינות קלט כחלק מכל אינטרקציה עם המשתמש, ובכל מקום שבו זה רלוונטי:
אם אתם מצפים לקבל מספר – לא לקרוס כי הכניסו לכם בטעות (או בכוונה) טקסט וכו'.
בכל מקרה של תקלה יש להיות מאוד ברורים במסר שמעבירים חזרה למשתמש: מה קרה ? מה הייתה מהות התקלה ? היכן שזה רלבנטי, איך לתקנה וכו'.
חישבו איך להיות ידידותיים למשתמש ולעולם אל תניחו כי מי שמשתמש באפליקציה שלכם הוא מתכנת בעצמו או מישהו שמגיע מהתחום ו"מכיר" איך דברים עובדים לבד.

4. **אין צורך** להשתמש בצבעים שונים במהלך תרגיל זה בעת ההדפסה ל console.
יתרה מזאת, ישנו צפי רב (ניסיון מהסמסטרים הקודמים) כי ניסיון לעשות כן תוך שימוש בספריות צד שלישי קורס אצל הבודק, מעוות את כל תצוגת המסך וגורם לחוסר יכולת לבדוק את ההגשה.
גם אם בדקתם את זה אצלכם וזה עבד.
גם אם בדקתם במחשב של השכנה וזה עבד.

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

5. עליכם לכתוב ממשק משתמש בתצורת console.
ממשק המשתמש יכיל סט סופי של פקודות שדרכן ניתן יהיה להפעיל את המערכת.
אחרי הצגת תפריט הפקודות יש לחכות לקלט מהמשתמש באשר לפעולה אותה הוא רוצה לבצע. לאחר ביצוע הפעולה (שאולי תגרור בקשת קלט נוסף מהמשתמש) יש להציג את הפלט החוזר ממנה (לכל פקודה יש פלט החוזר ממנה) ואז להציג שוב את התפריט וחוזר חלילה.

שימו לב:

- ישנן פקודות שאין הגיון לבצע אותן אם לא קדמו להן פקודות אחרות. במידה וזה קורה יש להציג הודעת שגיאה רלוונטית למשתמש ולאפשר את המשך פעילות המע'.

- בכל המקומות שבהם מציגים "רשימות של דברים" וצריך לאפשר למשתמש לבחור פריט(ים) מרשימה – יש לאפשר בחירה זו ע"י הצמדת מספר לכל אחת מהאפשרויות ולאפשר לו לבחור על פי המספר המזהה של האפשרות מהרשימה (או באמצעות כמה מספרים במקומות הרלבנטים). המספרים יתחילו מ 1 (לא מ 0)
  אין להניח שהמשתמש הולך להקליד לכם מלל חופשי של תיאור האפשרות !

להלן רשימת הפקודות שיש לתמוך בהן:

1. קריאת קובץ פרטי המע'

פקודה זו טוענת את פרטי המערכת מתוך קובץ נתונים בפורמט XML.
קבצי דוגמא מתאימים הועלו מבעוד מועד לאתר הקורס ואתם מוזמנים להורידם ולבחון אותם בהתאם.
(אתם מעודדים לייצר לעצמכם קבצי בדיקה נוספים כדי לבדוק את המע' בצורה יסודית וטובה יותר כאוות נפשכם).

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

הקובץ יהיה קובץ XML שפרטיו וחוקיו המפורטים מובאים בנספח ג' לתרגיל זה.
אתם מצופים לעבור על פרטים אלה ולהתייחס אליהם כחלק אינטגרלי מהגדרת התרגיל.

עליכם לוודא בדיקת קלט לקובץ ה XML ולוודא כי הקובץ מכיל מידע תקין ואמין.
(מובטח כי הקובץ יהיה תקין schema-wise אבל לא בהכרח תקין application-wise...)

בפרט יש לוודא את הפרטים הבאים:
1. הקובץ קיים, והוא מסוג XML (די לבדוק לשם כך כי הוא נגמר בסיומת xml.)
2. במסגרת משתני הסביבה, אין משתני סביבה בעלי שם זהה (שם של משתנה סביבה הוא ייחודי)
3. במסגרת הגדרת יישות בודדת, אין properties בעלי שם זהה (כלומר שם של property במסגרת entity הוא ייחודי)
4. במסגרת הפעולות המוגדרות בחוקים השונים, אין הפנייה ליישות שלא קיימת
5. במסגרת הפעולות המוגדרות בחוקים השונים, אין שימוש ב property שלא קיים עבור היישות שבהקשר.
6. במסגרת פעולות חישוב (calculation \ increase \ decrease), הארגומנטים הניתנים להם הם רק מסוג של מספר.
   שימו לב: זה כולל גם הפניה לפונקציות העזר הנתמכות.

במידה והקובץ לא תקין יש לדווח זאת למשתמש בצורה ברורה כך שניתן יהיה להבין מה לא תקין בקובץ.
אין לקרוס על exception במידה וקובץ מתגלה כאינו תקין ; יש לאפשר למע' להמשיך לפעול במצב זה.
(כחלק מבדיקת התרגיל יטענו למערכת קבצים שאינם חוקיים כדי לבדוק מהי התגובה).

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

דגשים:

- יש לאפשר למשתמש לטעון כמה קבצים אחד אחרי השני (כלומר להפעיל את הפקודה כמה פעמים רצוף).
  כל קובץ תקין "דורס" לחלוטין את כל פרטי הקובץ (התקין) שהיה טעון לפניו במע' (ככל שהיה כזה).
  כל נסיון טעינה של קובץ תקול לא דורס את פרטי הקובץ (התקין) האחרון שהיה במע' (ככל שהיה כזה)

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

2. הצגת פרטי הסימולציה

פקודה זו תאפשר להציג למשתמש מידע מסודר לגבי הסימולציה שהוגדרה בקובץ.
יש להציג את הפרטים הבאים:

יישויות (entities)
עבור כל entity:

- שם
- כמותו באוכלוסיה
- רשימת התכונות (properties) שיש לו.
  עבור כל תכונה
  - שם
  - סוג
  - טווח (אם קיים)
  - האם מאותחל רנדומית או לא

חוקים
עבור כל חוק:

- שם
- מתי הוא מופעל: גם ticks וגם probability (אם מי מהם אינם מוגדרים יש לציין את ערכי ברירת המחדל)
- כמות הפעולות (action) המוגדרות במסגרת החוק ברמה העליונה בלבד (כלומר ללא פעולות מקוננות)
- רשימת שמות הפעולות

תנאי הסיום
רשימת תנאי הסיום המוגדרים לסימולציה.
עבור כל תנאי סיום יש לציין את הפרטים שלו.

3. הפעלת סימולציה

פקודה זו תאפשר להפעיל את הסימולציה.
יש להציג למשתמש את משתני הסביבה ולאפשר לו לספק ערך לכל אחד מהם.
עבור כל משתנה סביבה יש להציג:

- שם
- סוג
- טווח (היכן שרלוונטי)

יש לוודא כי המשתמש מכניס ערכים חוקיים. במידה ולא יש להודיע לו על כך ולאפשר לו להכניסם בשנית.
המשתמש לא חייב לספק ערכים לכלל משתני הסביבה. עבור המשתנים שלא יסופקו להם ערכים – יוגרלו להם ערכים כלשהם על פי הוראות הטווח שניתנו להם (ככל שזה רלוונטי).

כאשר יחליט המשתמש שסיים לתת את כלל הערכים ההתחלתיים למשתני הסביבה שהוא חפץ ביקרם אפשרו לו להפעיל את הסימולציה.
בראשית יש להציג בצורה מסודרת את סך הערכים של משתני הסביבה כולם (שם המשתנה ואז ערכו).
מכאן הסימולציה מתחילה לעבוד תוך קיבוע ערכי ה environment שניתנו לה כך או אחרת.

עבור כל הפעלת סימולציה המע' תייצר מזהה חד חד ערכי המתאר את הפעלתה.
(באמצעות מזהה זה נוכל לחזור ולאתר את תוצאותיה בהמשך).
המזהה יכול להיות ספרור רץ, מחרוזת ייחודית (GUID למשל) או כל דבר העולה על רוחכם.

הפעלת הסימולציה מתרחשת במנוע המע'. המשתמש ממתין עד לסיומה.
אין לעדכן את המסך תו"כ תנועה. המשתמש ממתין מול מסך "שחור" וברגע שהיא תסתיים הוא יראה את תוצאותיה

הסימולציה תרוץ על פי האלגוריתם המתואר בחלק הכללי של המע'.
ההרצה תסתיים ברגע שיתקיים לפחות אחד מתנאי הסיום המוגדרים במסגרתה.

בגמר הרצת הסימולציה יש להחזיר למשתמש את המזהה החד חד ערכי של ההרצה שזה עתה נגמרה, כמו גם לציין את סיבת סיומה (מי מתנאי הסיום התממש).

4. הצגת פרטים מלאים של הפעלת עבר

פקודה זו תאפשר לקבל פרטי הרצה של הרצת-עבר של אחת הסימולציות.
ראשית יש להציג את רשימת כל ההרצות שבוצעו במע'.
על הרשימה להיות מוצגת על פי תאריך ההרצה מהמוקדם אל המאוחר (המוקדמת ביותר ראשונה; המאוחרת ביותר אחרונה).
עבור כל הרצה יש להציג:

- תאריך הרצה. הפורמט: dd-mm-yyyy | hh.mm.ss
- מזהה ייחודי של ההרצה.

יש לבקש מהמשתמש לבחור את ההרצה הרלוונטית שהוא רוצה לראות את תוצאותיה.
(בחירה על ידי מספר של ההרצה, מבוסס 1 כמובן)

לאחר שבחר את ההרצה המבוקשת יש לבקש את אופן התצוגה:
על פי כמות או **הסטוגרמה של property** (ראו תיאור בהמשך)
לאחר איסוף כלל הנתונים (באמצעות בחירה מספרית) יש להציג למשתמש את התוצאות באופן שביקש.

אופני הצגת המידעים למשתמש:

כמות יישויות
יש להציג את כלל היישויות שבמע' ועבור כל אחת את הכמות ההתחלתית שלה ואת הכמות הסופית שלה בסוף ההרצה.
(כמות היישויות יכולה להשתנות במסגרת הסימולציה)

הסטוגרמת מאפיין
בתצוגה זו יש לבחור יישות מרשימת היישויות במע'. לאחר מכן יש להציג למשתמש את ה properties שלה ולבחור את ה Property של היישות שרוצים לראות את ערכו (בחירה מרשימה מספרית).
או אז יש להציג היסטוגרמה המתארת עבור כל ערך – כמה יישויות יש מערך זה.

5. 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'.

2. מנוע המערכת, האחראי על קבלת הפקודות (ממודול ה ui), ביצועם והחזרת פלטים מתאימים.
שימו לב שמודול זה "פסיבי", והוא **רק** מגיב לבקשות ולפקודות המתקבלות ממקורות בלתי ידועים לו (בתרגיל זה מודול #1).
בתרגילים הבאים מקורות נוספים יפנו אליו לקבלת מידע וחשוב מאוד להקפיד על כך **שמודול זה אינו מכיר/מודע למי פונה אליו.**

איך מתחילים ? (המלצה...)

התחילו מהבנת מבנה ופעולת המע' בצורה מוחלטת. הגו על נייר מבנה כללי של המחלקות העיקריות והקשרים והיחסים בינהן.

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 סימולציה, של החוקים השונים, של היישויות, היכולת להריץ סימולציה 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 בבניית האובייקטים העקריים (יישויות, חוקים, פעולות, מנגנון ההרצה). תוכלו להתחיל להרים את המע' גם לפני קריאת הנתונים מקובץ ולוודא כי היא מתחילה לעבוד. ממשו ראשונה דווקא את פקודה 2 שתאפשר לכם לראות כי בידכם המידע הנדרש ואתם מציגים אותו כראוי.
בהתאם להתקדמות, המשיכו במימוש פקודה 1 (על כלל היביטי הטעינה), ו/או עיברו לממש את פקודות הפעלת הסימולציה וצפייה בתוצאותיה (3 ו 4).

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

#	מהות	למה שווה לי ?	כמה שווה לי ?
1	**שמירה וטעינה של המע'** בבונוס זה המצב הנוכחי של המע' (כולל כל ההרצות שכבר בוצעו) 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').
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, או כל דבר נוסף העולה על דעתכם שחשוב שהבודק ידע.
4. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות ותפקידם.
5. יש לכלול בקובץ ה **readme** גם פירוט של המגישים שם, ת.ז. ואי מייל זמין ורלבנטי (!!) – במידה ויהיה צורך ליצור קשר.
6. במידה ומימשתם את מי מהבונוס(ים) לעיל, ציינו את שם הבונוס שמימשתם בראשית קובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !
הגשה באיחור, שאינה באישור, תבטל כל מימוש בונוס. אין להגיש באיחור בשביל להספיק לעשות בונוסים.
תכננו את הזמן בהתאם.

פרטים יבשים

צפי תחילת עבודה: **20.8.23**
תאריך הגשה: **14.9.23**
צפי זמן לביצוע: 4 **שבועות +**
ציון אפשרי מקסימלי: **130**
משקל התרגיל: **35%**
קושי: **מאתגר**

מטרות התרגיל העיקרית

1. מימוש תצוגת ותפעול המע' כאפליצקיה גרפית JavaFX
2. הוספת יכולת להרצת הסימולציות בצורה אסינכרונית
3. תמיכה ביישויות משניות בפעולות
4. תמיכה בקיום מרחב דו מימדי ומיקומי יישויות

דרישות

1. בתרגיל זה תממשו שכבת משתמש גרפית דרכה המשתמש יוכל לתפעל את המע' ולצרוך את המידע.
ל"עולם" יתווסף מרכיב חדש והוא המיקום של כל יישות במרחב, וכך (גם) יחסי הגומלין בין היישויות "הקרובות" יוכלו לבוא לידי ביטוי.
בנוסף תרחיבו את מנוע המע' במימוש סוגים נוספים של פעולות ופונקציות.
הרצת הסימולציה תתרחש מעכשיו בצורה אסינכרונית, פוטנציאלית מקבילית, בבטן השרת, תוך שהמשתמש מתעדכן בהתקדמותה בצורה אינטראקטיבית ורציפה.

2. הרצה אסינכרונית:

הרצת הסימולציה יכולה לקחת זמן, כפונקציה של גודל האוכלוסיה, מורכבות החוקים וכו'.
במידה והמשתמש ירצה להריץ ולבצע כמה סימולציות שונות (אולי עם נתוני פתיחה שונים) וכדי להימנע מממצב שבו אחד חייב לחכות שהרצת הסימולציה הקודמת תסתיים, נבצע את הרצת הסימולציות באמצעות thread'ים עצמאיים תוך שימוש ב thread pool. (כמות ה thread'ים היא נתון המגיע מקובץ ה xml).

ברגע שהמשתמש בוחר להפעיל סימולציה, לאחר שסיפקו לו את כל משתני הסביבה הנדרשים להפעלתה, הרצת הסימולציה לא תתבצע מיידית ע"י ה thread המפעיל (שהוא כמובן ה ____) אלא בסה"כ תוסיף משימה ל threadpool פנימי בתוך מנוע המע' האמון על הפעלת הסימולציות השונות. זה בתורו ישחרר את ה ____ לחזור להמשיך ולתפעל את הממשק הגרפי שיישאר זמין לפעולות המשתמש.
thread פנוי יקבל את משימת ההרצה של הסימולציה ויהיה אחראי להריצה מתחילתה ועד סופה.

היות ולא "נתקעים" עד שהסימולציה נגמרת, פירושו של דבר שעבור כל הרצה קיימת של הסימולציה יש להציג את התקדמותה בצורה רציפה למשתמש עד אשר יגיע תנאי הסיום המבוקש.
יהיה עליכם לייצר Task (כפי שנלמד בכיתה) אשר אמון על התעדכנות בהתקדמות הרצת הסימולציה (ראו פרטים בהמשך).

3. הגדרת העולם תכיל עכשיו גם "מרחב פיזי".
"המרחב" של העולם הוא רשת דו מימדית בעלת גודל מוגדר של כמות שורות ועמודות.
לכל יישות יוגרל בתחילת הסימולציה קורדינטה בה היא ממוקמת. בכל tick היישות תתקדם צעד אחד רנדומלי על גבי המרחב.
הצעדים יכולים להיות ב 4 כיוונים: למעלה, למטה, ימינה, שמאלה. (כלומר לא באלכסון).
בקורדינטה מסויימת יכולה להימצא ברגע נתון, לכל היותר יישות אחת.
המרחב הוא "מעגלי", ולכן אינסופי. במידה ויישות מגיעה לאחד מקצוות המרחב היא פשוט "עוברת" לצד השני שלו.

בהמשך לכך, המע' תגדיר פעולה חדשה, proximity, התאפשר לבצע פעולות מסויימת באם מתקיימת קרבה פיזית ברמה מסויימת בין יישויות (פרטים בנספח א')

4. תמיכה בריבוי יישויות:
במסגרת הגדרת "העולם" המע' תאפשר להגדיר יותר מיישות אחת. כל יישות מגיעה עם ה Properties שלה.
(ייתכנו properties בעלי אותו שם בין יישויות שונות).
מרגע שיש יותר מיישות אחת בעולם, התנהלות הסימולציה והיכולות שלה משתנות במעט וזאת כדי לאפשר מגוון של פעולות וכלים שבהם עוסקים ביותר מיישות אחת.
עד עכשיו במסגרת כל פעולה תמיד צויין מי היא היישות "הראשית" אשר בהקשרה הפעולה מתבצעת.
עכשיו אפשר יהיה להוסיף (במסגרת הגדרה של כל פעולה) גם יישות נוספת, "משנית", שהפעולה מתרחשת בהקשרה.
בבואנו לבחור את היישות המשנית יש לציין בכמה מופעים שלה אנחנו רוצים לעסוק (אחרי הכל מכל יישות יש כמה וכמה מופעים). המע' תאפשר לבחור את כל היישויות, או מספר רנדומלי שלהן, עם אפשרות להגדרת תנאים המאפיינים את היישויות האחרות הנבחרות.

בחירת היישויות הנוספות במסגרת כל פעולה משנה במקצת גם את תהליך הרצת הסימולציה:
במצב רגיל שבו יש רק יישות אחת, החוקים ברי ההרצה (בהתאם להגדרת ה activation) יחולו על כלל מופעי היישויות הרלוונטיות.
במצב בו נבחרות יישויות משניות נוספות לפעולה, הפעלת החוק תעשה עבור כלל מופעי היישויות הרלוונטיות ועבור כל אחת מהן עבור כלל היישויות המשניות כפי שהוגדרו לפעולה (כלומר מכפלה של כמות היישויות).
מנגנון זה יאפשר לנסח כללים ותנאים מורכבים אשר יידמו טוב וקרוב יותר את העולם האמיתי.

עיקבו אחר נספח ג' כדי להבין איך נראית הגדרת יישות נוספת.

5. הרצת סימולציה אינטרקטיבית:
בזמן הרצת הסימולציה המשתמש יקבל מידע על התקדמותה והמדדים השונים שלה (ראו פירוט בהמשך).
המשתמש יוכל לבחור לשלוט בהתנהלות הסימולציה תו"כ שהיא מתבצעת, במסגרת הפעולות הבאות:
1. Pause – השהיה של הסימולציה. במצב זה הסימולציה עוצרת. ניתן לקבל את הפרטים והמידעים שהתקבלו בה עד עכשיו.
הסימולציה עוצרת רק לקראת התחלת הצעד הבא (לא עוצרים באמצע עיבוד של צעד נוכחי).
2. Resume – המשך עבודתה של הסימולציה. פקודה זו מאפשרת להמשיך את הסימולציה מהנקודה שבה הושהתה.
הסימולציה ממשיכה מהצעד אותו היא עמדה לבצע והנתונים שלה ממשיכים לזרום למשתמש.
3. Stop – פעולה זו עוצרת את הסימולציה לחלוטין (לא ניתן לבצע Resume לאחר מכן). גם כאן הסימולציה נעצרת רק בסיום העבודה על הפסיעה הנוכחית.
שימו לב: ניתן לבצע stop גם כשהסימולציה נמצאת ב pause.

6. המע' תתרחב ותתמוך במספר פונקציות חדשות ופעולות חדשות אשר את פרטיהם ניתן לראות בנספח א' וב'

7. לאורך כל השהות במע' ניתן לראות תמיד את המידע הבא (כפי שעולה מן הסקיצה):

- הקובץ הטעון במע'
יש לאפשר טעינה של קובץ מע' (על פי סכמה חדשה המתאימה לקבצים בפורמט של תרגיל 2 בלבד).
הקובץ יכיל הפעם גם מידע לגבי כמות הטרדים כמו גם נתונים על ה"מרחב" של הסימולציה.

עליכם לבצע בדיקות תקינות קלט של הקובץ כפי שבצעתם בתרגיל 1 ובנוסף:
1. יש לוודא כי גודל השורות והעמודות בין 10 ל 100 (כולל).
2. בחוק proximity יש לוודא כי אין הפנייה ליישויות שלא קיימות
3. בחוק replace יש לוודא כי אין הפנייה ליישויות שלא קיימות

אם הקובץ אינו תקין אין לטעון אותו למע' ויש להחזיר הודעה מפורטת למשתמש מדוע אינו תקין.
אם הקובץ תקין הוא נטען למע' בהצלחה וניתן לצפות בפרטיו ולהפעילו בהתאם.

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). **<u>נא לוודא כי הדבר אינו מכשיל אתכם.</u>**

- מידע על מצב ה threadpool הפנימי והעומס במע'.
יש להציג במסודר ובאופן רציף:
  - כמות הסימולציות הממתינות לביצוע (גודל ה QUEUE)
  - כמות הסימולציות המתבצעות כרגע (כמות ה thread'ים שעובדים)
  - כמות הסימולציות שביצוען הסתיים

8. מהלך העבודה:
יכולות האפליקציה בשלב זה ירוכזו בשלושה אפיקים (מסכים) מרכזיים.
עיקבו אחר קובץ מתאר המע' כדי להכיר את המבנה המצופה.
המבנה מגדיר את <u>קוי המתאר הכלליים</u> של האפליקציה ו**חובה להיצמד אליו**.

1. מסך 1 – Details
במסך זה ניתן יהיה לראות את הפרטים השונים של הסימולציה (בדומה למידע שהוצג בפקודה מס' 2 בתרגיל 1).
לבד מהפרטים שהצגתם אי אז, יש להציג גם את פרטי הפעולות השונות במסגרת כל חוק.
שימו לב כי לכל חוק יש את הפרטים הייחודיים לו, ולכן אך הגיוני וראוי כי לכל חוק יהיה רכיב UI מותאם משלו להצגת הפרטים הנ"ל (אין לזרוק את כל הפרטים כטקסט חופשי סתם כך !)

ממליץ להציג את הסימולציה באמצעות עץ המכיל את מרכיביה השונים, כאשר כל מרכיב מתפרק לתתי המרכיבים שלו.
לחיצה על מרכיב בודד מציגה את פרטיו הטכנים. (master-details view)

2. מסך 2 – New Execution
במסך זה המשתמש יוכל להזניק סימולציה חדשה לפעולה.

ראשית, על המשתמש למלא את כל משתני הסביבה החופשיים הנדרשים להפעלת הסימולציה (כשם שהיה בתרגיל 1)
שנית, במסגרת ההפעלה הזו כמויות היישויות השונות בעולם נתונות גם הן להחלטת המשתמש (כלומר אינן מגיעות כחלק מה XML). על כן יש לאפשר למשתמש לקבוע כמה מופעים של כל יישות המוגדרת בקובץ הוא רוצה שיופיעו בעולם.
שימו לב: סך היישויות לא יכול לעלות על גודלו של המרחב. (יש להתריע למשתמש אם הוא מגיע למצב זה)
אפשר להחליט כי מיישות מסויימת ישנם 0 מופעים.

לאחר סיום עדכון הנתונים השונים המשתמש ילחץ על כפתור ייעודי אשר יפעיל את הסימולציה (ויעביר אותו למסך מספר 3).
מאחורי הקלעים הסימולציה מצטרפת לתור הממתינים לביצוע ב threadpool במנוע. רק כשיימצא thread חופשי הוא

ייקח אותה ויבצע אותה עד הסוף.
אם במהלך ההרצה בפועל המשתמש השהה את הסימולציה – ה thread האמון על ביצועה "תקוע" (blocking) וממתין עד שישחררו אותו ותימשך הסימולציה.

בשלב זה המשתמש יכול לחזור ולהפעיל הרצה נוספת של הסימולציה אשר תתווסף להרצה שכבר נמצאת ברקע.
הציגו למשתמש גם כפתור איפוס (clear) המאפשר לנקות את הנתונים שכבר הוכנסו ולהתחיל מחדש.

3. **מסך 3 – Results**

במסך זה המשתמש יוכל לראות את פרטי הרצת הסימולציה. המסך יאפשר לראות גם את הרצות העבר שבוצעו ונגמרו זה לא מכבר, כמו גם את כל ההרצות הנוכחיות שכרגע עוד רצות/מושהות במקביל בבטן המנוע.
יש להציג את רשימת הסימולציות שבוצעו/מבוצעות במע'. עבור כל אחת יש להציג סימן האם היא רצה כרגע או כבר נגמרה.

בבחירת סימולציה מסויימת, יש להציג את הפרטים השונים עליה (בין אם היא כרגע רצה ובין אם היא כבר הסתיימה).

1. פרטי התקדמות
הציגו מידע על התקדמות הסימולציה:
באיזה צעד (tick) היא נמצאת כרגע
מונה זמן ריצה (בשניות)

2. מידע על היישויות
הציגו מידע על כמויות היישויות המתנהלות כרגע בסימולציה:
עבור כל יישות הציגו את שמה ואת הכמות שלה במע'.
(קלאסי לטבלה)

3. ניתוח תוצאות הסימולציה (אפשרי רק עבור סימולציות שנגמרו):
אפשרו למשתמש לנתח את תוצאות הסימולציה על פי האופנים הבאים:
1. הציגו גרף המתאר את כמויות היישויות כפונקציה של ציר הזמן (ticks)
2. אפשרו לבחור יישות ומאפיין שלה ולהציג עבורה מידעים סטטיסטים
   - consistency – מה כמות הפסיעות הממוצעת להישארות אותו הערך.
   - durational value – הצגת מידע ארוך טווח של ערכי ה properties.
   עבור נתונים מספריים: ממוצע ערכם לאורך הסימולציה.
   עבור ערכים בולאנים: כמה פסיעות ערכו היה true וכמה פסיעות ערכו היה false;
   עבור string'ים: הסטוגרמה של כל ערך וכמות הפסיעות שהוא היה מצוי בו.

כאשר סימולציה מגיעה לקיצה (בשל הגדרת תנאי הסיום) יש להודיע על כך למשתמש ולסמן אותה ככזו שהסתיימה הפעלתה.

הציגו למשתמש כפתור המאפשר לו לבחור להריץ מחדש סימולציה שכבר נגמרה. בלחיצה על כפתור זה הוא יועבר חזרה למסך 2 כאשר כל פרטי הסימולציה (משתני סביבה, כמויות יישויות) כבר יהיו בחורים ומלאים על פי המוגדר בסימולציה המקורית וכל מה שיישאר לו לעשות, אם יבחר בכך, זה ללחוץ שוב על ביצוע הסימולציה.

9. 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 הסימולציה הרלוונטית

עיברו לממש בצד המנוע את היכולת להריץ סימולציות במקביל. לאחר/בזמן שזה קורה תוכלו להמשיך במימוש המסך השני שבו תתאפשר הגדרת הריצה של הסימולציה תוך מילוי הפרטים הרלוונטים.

לאחר מכן עיברו לממש את מסך 3 המציג את תוצאות הריצה.

בונוסים

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

#	מהות	למה שווה לי ?	כמה שווה לי ?
1	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	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	תצוגה באמצעות גרפים הציגו את תוצאות הסימולציה לפי properties באמצעות גרף כפונקציה של ה ticks	כשמגלים כמה זה קל – זה מרגיש מדהים !	5 נקודות (ועד ל 100)
4	תצוגה גרפית של המרחב הציגו את המרחב של הסימולציה כרשת עם קורדינטות. לכל יישות צריך להיות צבע משלה וכל משבצת תיצבע בצבע של היישות אשר נמצאת בה. את המרחב יש להציג במסך השלישי. הציגו את המרחב במידה והמשתמש בוחר לבצע הרצה ידנית של הסימולציה. או אז הוא יוכל לראות ממש איך היישויות זזות במרחב על כל צעד ושעל	!!! WTF	10 נקודות **(מעל ל 100)**
5	הרצה לאחור אפשרו הרצה אחורה בזמן של ה ticks. כל הנתונים משתנים בהתאם למצב של העולם כפי שהיה בפסיעה המוגדרת.		10 נקודות **(מעל ל 100)**
6	הרצה ידנית כאשר הסימולציה במצב של pause, המשתמש יוכל להמשיך אותה ידנית ולקדם אותה צעד אחרי צעד (בלחיצה על כפתור מתאים) תוך שהוא יכול לראות בכל צעד את המידעים השונים של הסימולציה ומה השתנה בעקבות הצעד הנ"ל.		10 נקודות **(מעל 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 את שם הבונוס בראשית קובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

פרטים יבשים

צפי תחילת עבודה: **15.9.23**
תאריך הגשה: **19.10.23**
צפי זמן לביצוע: **5 שבועות +**
ציון אפשרי מקסימלי: **110**
משקל התרגיל: **35%**
קושי: **וואו'עליה אלל'בחטי**...

מטרות התרגיל העיקרית

1. מימוש המערכת כאפליקציית client-server.
2. הוספת מנגנון הקצאות ומשתמשים

דרישות

1. בתרגיל זה נממש את היכולת לתפעל את מערכת ה predictions כשרת, המציע את יכולות הרצת הסימולציות עבור משתמשים שונים ומגוונים.
משתמש מסויים יוכל להעלות מספר קבצי סימולציות שונות למערכת. משתמשים אחרים יוכלו לבקש ("לקנות") להריץ סימולציות שונות. כל משתמש רואה רק את הסימולציות שלו, ואין לו גישה להרצות של אחרים.
כל הרצות הסימולציה מתבצעות אך ורק בבטן השרת (שם יושב מנוע המערכת, כמו גם ה threadpool).
המשתמשים השונים יימשכו נתונים ויתעדכנו במצב הסימולציות שלהם מול השרת.

2. השרת (tomcat) יכיל את מנוע המערכת ויחשוף את היכולות בדמות endpoints שונים.

מול השרת יעבדו 2 סוגים של לקוחות/יישויות:

Administrator

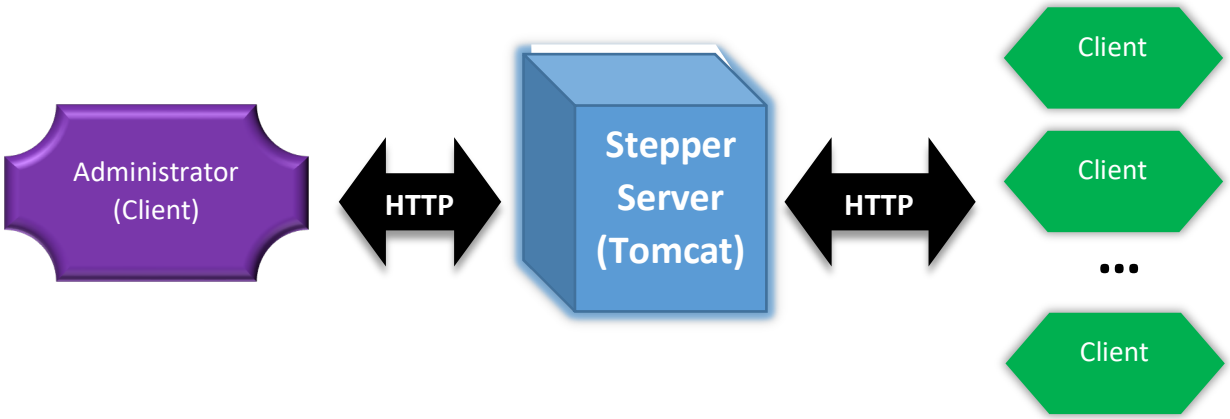
ה Administrator (אדמין, בקיצור, בישראלית) הוא מנהל המע'. יש <u>בדיוק</u> administrator <u>אחד</u> בלבד.
בין יכולותיו של האדמין:

- העלאת קבצים (קובץ בפורמט של תרגיל 3) המכילים הגדרות של סימולציות.
הפעם כל קובץ המכיל הגדרות סימולציה <u>מתווסף</u> למנוע המע' (ולא מחליף את הקובץ הקיים, כפי שהיה בתרגיל 1 ו 2).
כך יכול האדמין להעלות כמה קבצים ולהעשיר את המע' במגוון של סימולציות.
- לתת גישה למשתמש לכמות הרצות של סימולציה מסויימת.
- 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 <u>מספר רב</u> של קליינטים.
בין יכולותיו של הקליינט:

- צפייה בסימולציות האפשריות במע'.
- הגשת בקשה להריץ סימולציה מסויימת כמה פעמים.
- יכולת להפעיל ולהריץ סימולציה (כשם שהיה בתרגיל 2).
- צפייה בתוצאות הרצות העבר של הקליינט עצמו בלבד (כלומר לא של הרצות שבוצעו ע"י משתמשים אחרים)



3. מודל הרשאות ההרצה:
משתמשים יכולים להיחשף לסך סוגי הסימולציות המתקיימים במע', אולם הם אינם רשאים להריצם.
משתמש יוכל להגיש בקשה לאדמין (בתיווך השרת כמובן) שבה הוא מבקש לקבל הרשאה להריץ סימולציה מסויימת כמות פעמים מסויימת.
האדמין מקבל את הבקשות השונות ממשתמשי המערכת ומחליט למי הוא מקצה הרשאה להריץ וכמה פעמים.
(במציאות המשתמש משלם בכסף בשביל הזכות להשתמש במע'; במסגרת התרגיל נוותר על מימוש חלק ניהול החשבונות).
מרגע שהמשתמש קיבל את האישור הוא רשאי להריץ את הסימולציה ככמות פעמים שהוא ביקש אותה. לאחר שסיים להריץ את הכמות שביקש – שוב לא יוכל להריץ את הסימולציה אלא אם יבקש בקשה נוספת.

4. חלק מפרטי המידע במע' צריכים להתעדכן בצורה אוטומטית עבור כל משתמש בצורה של Pull כפי שלמדנו בשיעור והודגם בדוגמא המסכמת (אם בשיטה של all או של delta fetching – כל מקרה לגופו).
אפשר לבצע את ה Pull בטווח זמן של עד 2 שניות, אולם זמן סביר ל pull הוא סדר גודל של חצי שנייה.

5. אין צורך לבצע שמירה של נתונים מעבר למופע הנוכחי של השרת. במילים אחרות כשהשרת יורד - כל הנתונים נעלמים.
לא שומרים את המשתמשים שנרשמו, הסימולציות שהורצו וכו'.

6. יש לדאוג ולוודא כי מסכי המע' לסוגם מתנהגים יפה ב resize ולא מחסירים שום פרט

7. 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 3 (עיקבו אחר נספח ג' כדי לראות את השינויים יחסית לסכמה 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 הסימולציה במסגרת הקובץ התקין יועלו ויתווספו לרשימת הסימולציות המוגדרת בשרת, המוצגת במסך הניהול.
לכל סימולציה מתווסף גם שם ייחודי. אין לאפשר להעלות סימולציות בעלות אותו השם. במידה ומנסים להעלות סימולציה בשם שכבר קיים הקובץ לא נחשב תקין והודעת שגיאה מתאימה צריכה להגיע לאדמין.

גם כאן ינוהל threadpool פנימי בתוך השרת אשר ישמש להרצת הסימולציות. הפעם מי שקובע את כמות ה thread'ים הוא האדמין. לאדמין יהיה מסך ניהול ושליטה של התור במערכת ודרכו הוא יוכל להגדיר את כמות הטרדים.(בפעם הראשונה כשהוא נכנס עליו לקבוע את מספר הטרדים ומאז הם קבועים).

האדמין לא יכול להפעיל סימולציות, אולם הוא יכול לנווט למסך של תוצאות ההרצות כשם שהיה בתרגיל 2.

בנוסף יש לו מסך חדש שבמסגרתו הוא יכול לנהל את כל הקצאות המשתמשים (allocations)
במסך ההקצאות המשתמש יראה מידע על כלל בקשות ההרצה ממשתמשים.
בקשת הרצה כוללת את הפרטים הבאים:

- שם הסימולציה
- שם המשתמש
- כמות הרצות מבוקשת
- תנאי סיום, אחד או יותר (על פי זמן; על פי ticks; על פי החלטת משתמש)

כשבקשה מתקבלת בשרת (ואז מועברת לאדמין) היא מקבלת מאחורי הקלעים מספר בקשה ייחודי (אפשר ספרור רץ).
יש להציג את כלל המידעים של בקשה בצורה בטבלה מסודרת.
האדמין מושך ומתעדכן בבקשות החדשות ובסטטוס הקיימות כל פרק זמן קבוע מהשרת.

בנוסף יש להעשיר את פרטי הבקשה במידעים הבאים:
יש לסמן האם הבקשה פתוחה (ממתינה לאישור/פסילה) או סגורה (האדמין כבר טיפל בה).
במידה ובקשה טרם אושרה ("בקשה פתוחה") יש לאפשר לאדמין לבחור לאשר או לפסול אותה.
אם האדמין אישר את הבקשה יש להציג בשורת הבקשה שני נתונים מספריים נוספים:

- כמה הרצות מתבצעות ע"י משתמש זה בבקשה זו כרגע
- כמה הרצות סה"כ הסתיימו ובוצעו ע"י המשתמש

אם האדמין פסל את הבקשה יש להסיר את שורת הבקשה מהטבלה (הוא סיים לטפל בה).

נוסף על כך האדמין יכול לבחון הרצות עבר שכבר הסתיימו (לא כאלה שרצות כרגע) של כלל המשתמשים.
לשם כך יש להוסיף ולהציג נתון לכל סימולציה של שם המשתמש שהריץ אותה ולאיזה מספר בקשה היא משתייכת.

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).
<u>אין לחייב</u> או להגדיר שימוש בסיסמה או כל רכיב authentication כזה או אחר !

לאחר רישום מוצלח, המשתמש מגיע למסך בו הוא רואה את כלל הסימולציות המתקיימות במערכת (כפונקציה של הקבצים שהעלה האדמין, אם וככל שהוא העלה).
המשתמש יוכל לבחור סימולציה נדרשת ולהגיש בקשה לקבל הרשאה להריצה.
במהלך הגשת הבקשה יש למלא את הפרטים הבאים:

- שם הסימולציה
- כמות הרצות מבוקשת
- תנאי סיום, אחד או יותר (על פי זמן ו/או על פי ticks; או על פי החלטת משתמש)

לאחר אישור הבקשה היא תישלח לשרת לבחינה ע"י האדמין.
הקליינט יוכל לראות טבלה שבה מופיעות כל הבקשות שהוא ביצע בצורה מסודרת.

אם האדמין מחליט לאשר את הבקשה, המשתמש יראה חיווי כי היא מאושרת ואז יוכל גם לראות כמה נתנוים לגבי ההרצות שעוד נותרו לו לבצע.
אם האדמין מחליט לפסול את הבקשה, המשתמש יראה חיווי כי אינה מאושרת.

אם כן, עבור כל בקשה יש להציג:

- מספר סידורי של הבקשה (נקבע בשרת ומתקבל לאחר הגשתה)
- שם הסימולציה
- כמות הרצות מבוקשת
- האם אושרה/נפסלה
- כמה סימולציות רצות כרגע
- כמה סימולציות נגמרו

המשתמש יכול להגיש בכל רגע נתון כמה בקשות שהוא רוצה, אפילו עבור אותה סימולציה (כל בקשה היא בקשה עצמאית)

לאחר שהמשתמש קיבל הרשאה להריץ סימולציות, הוא יכול לבחור את אחת הבקשות שעוד יש לו הקצאה בה ולבחור להריץ את הסימולציה. או אז הוא מועבר למסך ההרצות, אשר זהה למסך שהיה בתרגיל 2. לאחר הזנקת ההרצה הוא יכול לעבור למסך ההסטוריה כפי שהיה בתרגיל 2 ולבדוק את פרטי ההרצה.

ברגע שהמשתמש מיצה את כמות ההרצות כפי שמוגדרת לו בבקשה הוא לא יכול יותר לבצע הרצה של סימולציה זו (אלא עד שיגיש בקשה חדשה ויקבל אישור לה).

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.
האדמין יוכל להעלות קבצים ולראות את הסימולציות הזמינות; המשתמש יוכל לקבל חיווי ולהציג לו את הסימולציות האפשריות

המשיכו בפיתוח מודל ההקצאות בצד המנוע שבשרת. לאחר מכן פתחו את המסך אצל המשתמש המאפשר לו להגיש בקשה, ואת המסך אצל האדמין המאפשר לו להציג בקשות. אחר כך פתחו את יכולת האישור/פסילה של בקשה כפי שהוא בא לידי ביטוי בשני הצדדים.
לבסוף הוסיפו את היכולת להריץ/לראות הרצות כפי שהיה בתרגיל 2. אמור להיות זהה וקליל בשלב זה...

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. (מאוווווווווד דומה לדוגמא הסופית) הצ'ט מתבצע בין המשתמשים הרשומים והמחוברים למערכת בלבד.	היש מדהים מזה ???	5 נקודות (ועד ל 100)
2	מגניב !	שינוי כמות ה thread'ים אפשרו לאדמין לשנות באופן דינמי את כמות ה thread'ים הזמינים במע'. האדמין יוכל להגדיל את כמות הטרדים במידה והוא חושב שהמערכת צפויה להיות עמוס יותר או אם הוא רואה שיש "פקק" של בקשות הרצה ואין מי שיריץ אותם.		10 נקודות **(מעל 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 השונים 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 את שם הבונוס **בראשית** קובץ ה readme כדי שהבודק ידע לבדוק בהתאם.
   בונוס שימומש אבל לא יתועד – לא ייבדק !
   הגשה באיחור, שאינה באישור, תבטל כל מימוש בונוס. אין להגיש באיחור בשביל להספיק לעשות בונוסים.
   תכננו את הזמן בהתאם.

נספח א' – סוגי פעולות (action)

נספח זה מציג את סוגי הפעולות השונות שהמע' תתמוך בהם.
לכל פעולה יש את המאפיינים שלה. המידע השלם על פעולה מגיע כמובן מקובץ ה XML והגדרת הפעולה במלואה עוברת גם דרך ההבנה של איך נראית הגדרתה הגולמית בקובץ ה XML.

לא את כל הפעולות יש לממש מלכתחילה. לכל פעולה מוגדר מאיזה תרגיל היא נדרשת (כמובן שתמיד אפשר להגדיל ראש ולהקדים תרופה למכה...)

לכל פעולה יש יישות ראשית והפעולה מתרחשת במסגרתה (היישות מוגדרת במסגרת הגדרת הפעולה המגיעה מקובץ ה XML).
בכל המקומות שבהם מציינים שם של property – זהו property של היישות הראשית וכו'.

במסגרת פעולה אפשר להגדיר גם יישות משנית שבמסגרתה היא עובדת (החל מתרגיל 2).
פעולות שונות יכולות לבצע התייחסות ליישות המשנית במסגרת הפעלת הפעולה.

פעולות רבות יכולות לקבל מידע הלובש צורות שונות.
נגדיר expression ככזה היכול להכיל מגוון שונה של מידעים:

- הפעלה של פונקצית עזר במע' (ראו נספח ב' לפירוט)
- שם של property של יישות בהקשר מסויים
- ערך חופשי

מובטח כי לא יהיו שמות property כשמות פונקציות העזר המסופקות ע"י המערכת (כך שאפשר יהיה להבדיל בינהם)
מובטח כי לא יינתנו שמות משתני סביבה הזהים לשמות של Properties של היישויות השונות ו/או לשמות פונקציות העזר של המערכת (כך שאפשר יהיה להבדיל בינהם).

הדרך לפענח ערכים מסוג של expression היא בצורה הבאה:

1. אם המילה הראשונה היא שם של פונקציה מוגדרת במערכת – הרי שיש לצפות ולפרק את ה expression על פי חוקי הגדרת הפונקציה.
2. אם המילה הראשונה אינה שם של פונקציה מוגדרת, יש לנסות לחוות את ה expression כשם של property בהקשר של היישות הראשית.
3. אם ה expression אינו נופל במסגרת בדיקה #1 ו #2 – אזי הוא ערך חופשי ויש להשתמש בו בצורה הבאה:
   - אם הוא בהקשר של פעולה מספרית – לוודא שהוא בר המרה למספר;
   - אם הוא בהקשר של פעולה בוליאנית – לוודא כי הוא בדיוק true או false;
   - אם הוא בהקשר של פעולה על string – פשוט לקחת אותו AS IS;

פעולות אשר תכליתן לשנות ערך של property מספרי לא יוכלו לבצע חריגה מהתחום המוגדר והמותר עבור אותו ה Property.
באם פעולה מתיימרת לחרוג מהתחום – היא פשוט מתבטלת ולא מתבצעת בצורה שקופה למשתמש.
מקרה זה אינו נחשב לתקלה ואין צורך לדווח עליו או לעצור את הסימולציה בגללו.

הנה רשימת הפעולות בצורה מרוכזת:

#	שם הפעולה	נדרש החל מתרגיל
1	Increase	1
2	Decrease	1
3	Calculation	1
4	Condition	1
5	Set	1
6	Kill	1
7	Replace	2
8	Proximity	2

1. Increase

תיאור

פעולה זו מאפשרת להגדיל את ערכו של property מספרי של יישות מסויימת בגודל מסויים.

מבנה XML

```
<PRD-action type="increase" entity="ent-1" property="p1" by="3"/>
```

- type: מתאר את סוג הפעולה
- entity: שם של היישות הראשית אשר בהקשרה מבוצעת הפעולה
- property: שם ה property מתוך היישות הראשית שבהקשרה מבוצעת הפעולה.
- by: ביטוי (expression) המתאר בכמה יש להגדיל את ה property המדובר.

2. Decrease

תיאור

פעולה זו מאפשרת להקטין את ערכו של property מספרי של יישות מסויימת בגודל מסויים.

מבנה XML

```
<PRD-action type="decrease" entity="ent-1" property="p2" by="environment(e3)"/>
```

- type: מתאר את סוג הפעולה
- entity: שם של היישות הראשית אשר בהקשרה מבוצעת הפעולה
- property: שם ה property מתוך היישות הראשית שבהקשרה מבוצעת הפעולה.
- by: ביטוי (expression) המתאר בכמה יש להקטין את ה property המדובר.

3. Calculation

תיאור

פעולה זו מאפשרת לבצע חישוב מתמטי על ערכו של property מספרי של היישות הראשית.

מבנה XML

```
<PRD-action type="calculation" entity="ent-1" result-prop="p1">
```

- type: מתאר את סוג הפעולה
- entity: שם של היישות הראשית אשר בהקשרה מבוצעת הפעולה
- result-prop: שם של property מספרי של היישות הראשית שיכיל את תוצאת החישוב.

המערכת תתמוך במגוון של פעולות מתמטיות לביצוע תחת ההגדרה של פעולה זו.
כולן יתרחשו במסגרת היישות הראשית.

- Multiply

  תבצע הכפלה של 2 מספרים. תוצאת ההכפלה תושם ל result-prop

```
<PRD-multiply arg1="p1" arg2="environment(e1)"/>
```

  - Arg1: ארגומנט ראשון לפעולה. זהו expression
  - Arg2: ארגומנט שני לעפולה. זהו expression

- Divide

  תבצע חילוק של 2 מספרים. תוצאת החילוק תושם ל result-prop

```
<PRD-divide arg1="environment(e3)" arg2="3.2"/>
```

  - Arg1: מונה החלוקה. זהו expression
  - Arg2: מכנה החלוקה. זהו expression

4. Condition

תיאור

פעולה זו מאפשרת להגדיר פעולת תנאי.

```
<PRD-action type="condition" entity="ent-1">
```

- type: מתאר את סוג הפעולה
- entity: מתאר את שם היישות הראשית בהקשר הפעולה.

condition יכול להיות מוגדר בשני אופנים הנקבעים על פי ערך ה singularity שלו:

1. Condition פשוט (סופי)

מבנה זה מתאר תנאי פשוט יחידני במע' שניתן להערכה:

```
<PRD-condition singularity="single" entity="ent-1" property="p1" operator="bt" value="4"/>
```

- singularity: מתאר את היות המבנה הנ"ל תנאי פשוט (single)
- entity: שם של היישות שבמסגרתה מבוצע התנאי
- property: שם של property על היישות הראשית אשר תיבחן בהקשר התנאי המדובר
- operator: אופרטור השוואה לבחינת הערך של on. הפעולות הנתמכות:
  - = בחינת שוויון
  - =! בחינת אי שוויון
  - Bt בחינה האם ערכו גדול מ ערך אחר (היכן שרלוונטי)
  - Lt בחינה האם ערכו קטן מ ערך אחר (היכן שרלוונטי)
- value: ערך מולו יושווה המשתנה ב on. זהו expression.

2. Condition מורכב (אגריגטיבי)

מבנה זה מתאר איחוד/הרכבה (אגריגציה) של תנאים אחרים (בין אם פשוטים או מורכבים בעצמם).
האיחוד וההרכבה יכולים להתבצע באמצעות אופרטורים לוגים: and | or
אפשר באמצעותו להרכיב תנאים מורכבים ביותר המכילים רמות קינון עמוקות יותר
מבנה זה לא מכיל תנאי ספציפי לבחינה, כי אם משמש רק לאיחוד והרכבה של תנאים פשוטים יותר

```
<PRD-condition singularity="multiple" logical="or">
```

- singularity: מתאר את היות המבנה הנ"ל מבנה מורכב/מאחד
- logical: מתאר את הפעולה הלוגית המאחדת את כל התנאים הנמצאים בתוכו.
  הפעולות הנתמכות:
  - And: הרכבה באמצעות פעולת "וגם"
  - Or: הרכבה באמצעות פעולת "או"

לאחר הגדרת התנאי, יופיע רכיב ה then אשר מתאר מה יש לבצע במידה והתנאי הוערך להיות אמת.
רכיב זה מכיל הגדרה של פעולות. יכול להכיל גם אוסף של פעולות אחת אחרי השנייה.

יש אפשרות להגדיר גם מבנה של else אשר יופעל במידה והתנאי לא מתקיים.
שימו לב: רכיב ה then חייב להופיע; רכיב ה else לא חייב להופיע (ובמידה והתנאי לא מתקיים – פשוט לא עושים כלום)

דוגמא למבנה מלא של פעולת ה condition:

```
<PRD-action type="condition" entity="ent-1">
    <PRD-condition singularity="multiple" logical="or">
    <PRD-then>
    <PRD-else>
</PRD-action>
```

5. Set

תיאור

פעולה זו מאפשרת לעדכן ערך של property של יישות מסויימת

מבנה ה XML

```
<PRD-action type="set" entity="ent-1" property="p1" value="random(3)"/>
```

- type: תיאור סוג הפעולה
- entity: שם היישות הראשית בהקשרה מבוצעת הפעולה
- property: שם ה property של היישות הראשית שיש לקבוע את ערכו
- value: הערך החדש של ה Property. זהו expression

6. Kill

תיאור

פעולה זו מאפשר להרוג את היישות הראשית שבמסגרתה היא מוגדרת

מבנה ה XML

```
<PRD-action type="kill" entity="ent-1"/>
```

- type: תיאור סוג הפעולה
- entity: שם היישות הראשית שיש להרוג אותה במסגרת הפעולה הנ"ל

7. Replace

תיאור

מאפשר להחליף יישות אחת במשנתה. למעשה הורג יישות אחת, ובורא יישות אחרת במקומה.
את תהליך יצירת היישות האחרת אפשר לברוא מחדש לחלוטין (כמו בתהליך תחילת הסימולציה, על פי הגדרות יצירת היישות) או תוך התבססות מלאה/חלקית על בסיס היישות שנהרגה.

מבנה ה XML

```
<PRD-action type="replace" kill="ent-1" create="ent-2" mode="scratch"/>
```

- type: סוג הפעולה
- kill: שם היישות להרוג
- create: שם היישות שיש לברוא
- mode: מגדיר כיצד לייצר את היישות החדשה.
  יכול להכיל 2 ערכים:
  - scratch: מייצרים את היישות החדשה לחלוטין מחדש על פי חוקי הגדרת יצירת היישות
  - derived: מייצרים את היישות החדשה כך שאם וככל שיש לה properties הזהים בשמם לאלה של היישות שנהרגה – היא לוקחת את ערכם.

8. Proximity

תיאור

פעולה זו מאפשרת לבצע פעולות נוספות על סמך בדיקת קרבה של יישות אחת לאחרת ב"מרחב" הסימולציה.
במסגרת הפעולה יוגדרו מיהם שתי היישויות אשר יש לבדוק את הקרבה בינהן, ומתי שתיהן תחשבנה "קרובות".
במידה ונמצאו 2 יישויות העונות על הגדרות הקרבה ניתן יהיה להגדיר אוסף של פעולות לביצוע עד שתי היישויות.

הקרבה נמדדת במעגלים.
דרגה 1: היישות המשנית נמצאת במי מ 8 המיקומים המקיפים את היישות הראשית (מעגל ראשון)
דרגה 2: היישות המשנית נמצאת במעגל המקיף את המעגל השני (מעגל שני)
וכו'..

מבנה ה XML

```
<PRD-action type="proximity">
  <PRD-between source-entity="ent-1" target-entity="ent-2"/>
  <PRD-env-depth of="1"/>
  <PRD-actions>
</PRD-action>
```

- type: סוג הפעולה
- source-entity: יישות הראשית שיש לבדוק האם היא נמצאת בקרבת יישות אחרת
- target-entity: יישות משנית שיש לבדוק האם היא נמצאת בקרבה ליישות הראשית
- of: מגדיר את מעגל הקרבה שעד אליו (כולל) ייחשבו 2 היישויות כ"קרובות". זהו expressed אשר תוצאתו מספרית

נספח ב' –פונצקיות עזר נתמכות

המערכת תספק מספר פונקציות עזר אשר ניתן להיעזר בהם במסגרת ה expression'ים השונים כדי להשיג מידעים שונים ומגוונים לכל פונקציה יש שם ייחודי.
כל פונקציה מקבלת ארגומנט אחד או יותר ומחזירה תוצאה מסויימת בהקשרו.

הפעלה של פונקציה תוגדר באמצעות המבנה הבא:

<function name>(<expression>,...)

כאשר:

- function name: הוא שם הפונצקיה
- expression: הוא ארגומנט שניתן לפונקציה. כל פונצקיה תגדיר את כמות הארגומנטים ואת סוגם אשר היא מצפה לקבל.

דוגמא:

environment(e1)

במקרה זה מדובר על פונקציה בשם environment והארגומנט נקרא e1
במידה ופונקציה נכשלת היא זורקת שגיאה אשר תטופל במסגרת מנגנון טיפול השגיאות הכללי של הסימולציה.

פונקציות משמשות בעיקר כדי לשלוף/לעדכן ערכים של properties\environment שונים.
במידה ומדובר על type'ים שונים של ערכים אלה, הנה רשימת ההמרות האפשרית שיש לתמוך בה.
(כל מה שלא מופיע כאן ייחשב כשגיאה) :

- ניתן להמיר מספר שלם למספר עשרוני, (אך לא להיפיך)
- ניתן להמיר את המחרוזת "true" לערך הבוליאני true ואת המחרוזת "false" לערך הבוליאני false וחוזר חלילה. (המחרוזות הן case sensitive)

למען הסר ספק: לא ניתן להמיר מספר ל string (כלומר המספר 11 לא יכול להיות מתורגם כ "11")

הנה רשימת פונקציות העזר שהמערכת תומכת בהן, בחלוקה לפי דרישות התרגיל:

#	שם הפונקציה	נדרש החל מתרגיל
1	environment	1
2	random	1
3	evaluate	2
4	percent	2
5	ticks	2

1. **environment**
פונקציה זו תאפשר גישה לערכי משתני הסביבה.
הפונקציה מקבלת ארגומנט אחד ויחיד שהוא שם של משתנה סביבה, ואז מחזירה את ערכו (יהיה אשר יהיה).
במידה ומשתנה הסביבה לא קיים היא זורקת שגיאה אשר תטופל כחלק ממנגנון טיפול השגיאות של הסימולציה.

2. **random**
פונקציה זו תאפשר להגריל ערך רנדומלי כלשהוא.
הפונקציה מקבלת ארגומנט מספרי אחד ויחיד (מספר שלם), והיא תגריל מספר בטווח של 0 ועד הארגומנט (כולל).
התוצר של הפונקציה הוא המספר המוגרל.
במידה והארגומנט אינו מספר היא זורקת שגיאה אשר תטופל כחלק ממנגנון טיפול השגיאות של הסימולציה.

3. **evaluate**
פונקציה זו תאפשר ביצוע הערכה של ערכו של property מסויים של entity אשר בהקשר.
הפונקציה מקבלת ארגומנט אחד ויחיד מהפורמט הבא:

<entity>.<property name>

הפונקציה תחזיר את ערכו של המאפיין, יהיה אשר יהיה.
במידה והארגומנט אינו פונה ל entity שבהקשר, או ה property אינו נמצא היא זורקת שגיאה אשר תטופל כחלק ממנגנון טיפול השגיאות של הסימולציה

4. **percent**

פונקציה זו תאפשר לחשב גודל אחוז מסויים מתוך שלם.
הפונקציה תקבל 2 ארגומנטים מספריים.
ארגומנט ראשון יהיה החלק השלם; ארגומנט שני יהיה האחוז המחושב מתוך החלק השלם.
שני הארגומנטים הם מסוג של expression
התוצר של הפונקציה הוא ערך מספרי המהווה את האחוז מתוך החלק השלם
דוגמא: percent(50, 10) = 5
במידה והארגומנטים אינם מספרים היא זורקת שגיאה אשר תטופל כחלק ממנגנון טיפול השגיאות של הסימולציה.

5. ticks

פונצקיה זו תאפשר למדוד כמה זמן (כלומר כמה ticks) ערך מסויים של property לא השתנה.
לשם כך יש לשנות את מבנה היישות/property (תלוי במימוש) כך שעבור כל ערך של property צריך לשמור מידע הסטורי באיזה tick הוא השתנה ו/או כמה זמן ערכו קבוע.

הפונקציה מקבלת ארגומנט אחד ויחיד מהפורמט הבא:

<entity>.<property name>

שימו לב כי כאן הפונקציה לא מתעניינת בערכו של ה property, אלא רק במשך הזמן שהוא היה קבוע ולא השתנה.

הפנוקציה תחזיר מספר שלם המהווה את כמות ה ticks שבהם הערך הנוכחי לא השתנה.
דוגמא: נניח ש property מסויים קיבל את ערכו הנוכחי בזמן T 50 ומאז לא השתנה ועכשיו אנחנו בזמן T 60, אזי הפונקציה ticks תחזיר 10.

תיאור המע' ומרכיביה בתרגילים השונים נתון באמצעות קובץ XML.

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

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

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

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

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

מבנה המע' מאוגד תחת האלמנט **PRD-world**:

#	סוג	שם	מהות
1	Element	PRD-world	אלמנט זה מכיל את כלל פרטי הסימולציה ומתאר את מאפייני העולם של הסימולציה
2	Element	PRD-environment	מתאר מידע על משתני הסביבה של הסימולציה
3	Element	PRD-entities	מתאר מידע על היישויות השונות בסימולציה
4	Element	PRD-rules	מתאר מידע על החוקים השונים שבעולם
5	Element	PRD-termination	מתאר מידע על תנאי הסיום להרצת הסימולציה

```
<PRD-world>
    <PRD-evironment>
    <PRD-entities>
    <PRD-rules>
    <PRD-termination>
</PRD-world>
```

האלמנט **PRD-environment** מתאר את סך משתני הסביבה של הסימולציה

```
<PRD-evironment>
    <PRD-env-property type="decimal">
        <PRD-name>e1</PRD-name>
        <PRD-range from="10" to="100"/>
    </PRD-env-property>
    <PRD-env-property type="boolean">
    <PRD-env-property type="float">
    <PRD-env-property type="string">
</PRD-evironment>
```

#	סוג	שם	מהות
1	Element	PRD-environment	מתאר את סך כל משתני הסביבה
2	Element	PRD-env-property	מגדיר משתנה סביבה בודד במע'
3	Attribute	type	מתאר את סוג משתנה הסביבה. ערכים אפשריים: • decimal – מספר שלם • float – מספר עשרוני • boolean – ערך בוליאני. יכול להכיל רק true או false • string – מתאר מחרוזת תווים
4	Element	PRD-name	מתאר את שמו של משתנה הסביבה. השם הוא ייחודי. השם יכיל מחרוזת תווים <u>ללא רווחים</u>
5	Element	PRD-range	מתאר טווח חוקי לערכים עבור משתנה הסביבה בהקשרו הוא מוגדר. זהו אלמנט אופציונלי שיכול להופיע רק עבור ערכים מספריים (decimal או float). במידה והוא אינו מופיע – אין מגבלה על הערכים המדוברים.
6	Attribute	from	מתאר גבול תחתון לערך מספרי שיכול להינתן למשתנה סביבה. הגבול כולל את הערך.
7	Attribute	to	מתאר גבול עליון לערך מספרי שיכול להינתן למשתנה סביבה. הגבול כולל את הערך.

האלמנט RPD-entities כולל תיאור של כלל היישויות במע':

```
<PRD-entities>
  <PRD-entity name="ent-1">
    <PRD-population>100</PRD-population>
    <PRD-properties>
      <PRD-property type="decimal">
        <PRD-name>p1</PRD-name>
        <PRD-range from="0" to="100"/>
        <PRD-value random-initialize="false" init="0"/>
      </PRD-property>
      <PRD-property type="float">
      <PRD-property type="boolean">
      <PRD-property type="string">
    </PRD-properties>
  </PRD-entity>
</PRD-entities>
```

#	סוג	שם	מהות
1	Element	PRD-entities	מתאר את סך כלל היישויות המשתתפות בסימולציה
2	Element	PRD-entity	מתאר יישות אחת בסימולציה
3	Attribute	Name	מתאר את שם היישות. השם הוא ייחודי. השם יכיל מחרוזת תווים <u>ללא רווחים</u>
4	Element	PRD-population	מתאר את כמות המופעים של יישות באוכלוסיית הסימולציה
5	Element	PRD-properties	מתאר את סך כל ה Properties של יישות מסויימת
6	Element	PRD-property	מתאר property בודד של יישות
7	Attribute	type	מתאר את סוג ה property. ערכים אפשריים: • decimal – מספר שלם • float – מספר עשרוני • boolean – ערך בוליאני. יכול להכיל רק true או false • string – מתאר מחרוזת תווים
8	Element	PRD-name	מתאר את שמו של ה property של היישות. השם הוא ייחודי במסגרת היישות. השם יכיל מחרוזת תווים <u>ללא רווחים</u>
9	Element	PRD-range	מתאר טווח חוקי לערכים מספריים. זהו אלמנט אופציונלי שיכול להופיע רק עבור ערכים מספריים (decimal או float). במידה והוא אינו מופיע – אין מגבלה על הערכים המדוברים.
10	Attribute	from	מתאר גבול תחתון לערך מספרי שיכול להינתן ל Property מספרי. הגבול כולל את הערך.
11	Attribute	to	מתאר גבול עליון לערך מספרי שיכול להינתן ל property מספרי. הגבול כולל את הערך.
12	Element	PRD-value	מתאר את האופן שבו צריך להעניק ערך ל property של היישות במעמד יצירתה
13	Attribute	random-initialize	ערך בוליאני. מתאר האם יש להתחיל את ה Property המדובר בצורה רנדומלית או לא. במידה והוא מאותחל רנדומלית (true) הערך יילקח בצורה רנדומלית מתוך ה range (במידה והוגדר), או יוגרל רנדומלית לחלוטין במידה ולא הוגדר. במידה והוא לא מאותחל רנדומלית (false) הערך יילקח מה Init
14	Attribute	init	מתאר את הערך שיש לאתחל את ה property של היישות במידה והוא אינו מאותחל רנדומלית

האלמנט **PRD-rules** מתאר את סך החוקים/פעולות שיש לבצע במהלך הסימולציה

```
<PRD-rules>
    <PRD-rule name="r1">
        <PRD-actions>
        <PRD-activation ticks="1" probability="0.4"/>
    </PRD-rule>
    <PRD-rule name="r2">
</PRD-rules>
```

#	סוג	שם	מהות
1	Element	PRD-rules	מתאר את סך כל החוקים המוגדרים במסגרת הסימולציה
2	Element	PRD-rule	מתאר חוק בודד במע'
3	Attribute	name	מתאר את שם החוק. השם הוא ייחודי. השם יכיל מחרוזת תווים ללא רווחים
4	Element	PRD-actions	מתאר את סך כל הפעולות שיש לבצע במסגרת חוק זה, באם הוא מאופשר. תיאור מפורט של מגוון הפעולות ומבני ה XML שלהן ניתן במסגרת נספח א'.
5	Element	PRD-activation	מתאר את התנאים להפעלת החוק. זהו רכיב אופציונלי. במידה ולא מופיע יש להשתמש בברירות המחדל כפי שמוגדרות בסעיפים הבאים.
6	Attribute	ticks	מתאר כל כמה פסיעות התקדמות של ציר הזמן יש להפעיל את החוק. מספר שלם. זהו רכיב אופציונלי. ברירת המחדל היא 1, כלומר כל פסיעה.
7	Attribute	probability	מתאר מה ההסתברות לפעולת החוק. מספר עשרוני בין 0 ל 1, כאשר 0 פירושו שהוא לא מתקיים בכלל; 1 פירושו שהוא מתקיים תמיד; כל מספר באמצע פירושו ההסתברות לפעולתו של החוק. זהו רכיב אופציונלי. ברירת המחדל היא 1, כלומר החוק מתקיים תמיד.

האלמנט PRD-termination מתאר את תנאי הסיום לסימולציה

```
<PRD-termination>
    <PRD-by-ticks count="480"/>
    <PRD-by-second count="10"/>
</PRD-termination>
```

#	סוג	שם	מהות
1	Element	PRD-Termination	מתאר את תנאי הסיום האפשריים לסימולציה. ברגע שתנאי אחד מתקיים – הסימולציה נגמרת. לפחות אחד מתנאי הסיום חייב להופיע; ייתכן כי שניהם יופיעו
2	Element	PRD-by-ticks	מתאר תנאי סיום כעבור כמות ticks מסויימת.
3	Attribute	count	מתאר את כמות הפסיעות שבתומה יש לעצור את הסימולציה
4	Element	PRD-by-seconds	מתאר את כמות השניות שבתומה יש לעצור את הסימולציה
5	Attribute	count	מתאר את כמות השניות שבתומה יש לעצור את הסימולציה

סכמה תרגיל 2

סכמה זו מכיל מספר שינויים מרכזיים:

1. **גודל ה threadpool:**

אלמנט המכיל מידע על גודל ה threadpool לשימוש במסגרת הרצת הסימולציות

```
<PRD-thread-count>3</PRD-thread-count>
```

2. **הגדרת מרחב לסימולציה**

האלמנט PRD-grid מגדיר את המרחב הדו מימדי של הסימולציה:

```
<PRD-grid rows="100" columns="100"/>
```

#	סוג	שם	מהות
1	Element	PRD-grid	מתאר מרחב דו מימדי שבמסגרתו פועלת האוטומציה.
2	Attribute	Rows	מספר שלם. מתאר את כמות השורות. מספר שלם בין 10 ל 100 כולל
3	Attribute	Columns	מספר שלם. מתאר את כמות העמודות. מספר שלם בין 10 ל 100 כולל.

3. **הגדרת יישות משנית**

האלמנט PRD-secondary-entity מגדיר את פרטי היישות המשנית אשר ניתן להתייחס אליה במסגרת הפעולות השונות.
האלמנט הנ"ל יכול להופיע בתוך כל PRD-action שהוא, (אם כי לא תמיד זה יהיה הגיוני).
בהתקיימו הוא מגדיר את סוג היישות המשנית שתהיה זמינה במסגרת הפעולה ומאפשר דרכים לבחירת סך היישויות הנ"ל.
(כאמור הוא משנה גם את לולאת הסימולציה הראשית)

```
<PRD-interactive-entity entity="ent-2">
  <PRD-selection count="4">
    <PRD-condition singularity="single" entity="ent-2" on="p1" operator="bt" value="4"/>
  </PRD-selection>
</PRD-interactive-entity>
```

#	סוג	שם	מהות
1	Element	PRD-interactive-entity	מתאר את סך המידע כיצד לבחור את היישות המשנית. זהו אלמנט אופציונלי ואינו חייב להופיע.
2	Attribute	entity	מתאר את שם היישות המשנית שתשתתף במסגרת הפעולה
3	Element	PRD-selection	מכיל מידע על האופן שבו צריך לבחור את היישויות המשניות לפעולה
4	Attribute	count	מתאר כמה יישויות משניות יש לבחור במסגרת הפעולה הזו. יכול להכיל ערך מספרי שלם או את המילה ALL. במידה ונבחר ALL – הרי שבוחרים את כלל היישויות המשניות בלי שום יוצא מן הכלל. במידה ונבחר ערך מספרי יש לבחור רנדומלית כמות יישויות משניות כפי שמוגדר.
5	Element	PRD-condition	זהו אלמנט המתאר תנאי(ם) שיברור אילו יישויות משניות תבחרנה, במידה וב count נבחר ערך מספרי. זהו אלמנט אופציונלי ואינו חייב להופיע. הגדרתו זהה לאופן שבו הוא מוגדר במסגרת פעולת ה condition (נספח א')

4. **תנאי סיום:**
התווסף תנאי סיום חדש: על פי אינטרקציה עם המשתמש.

```
<PRD-by-user />
```

תנאי זה מגדיר כי הסימולציה נגמרת רק כאשר המשתמש מחליט על כך, במסגרת ההפעלה האינטרקטיבית שלה (כלומר לחיצה על stop).
תנאי זה יכול להופיע לבדו, אך לא עם מי מהתנאים האחרים.

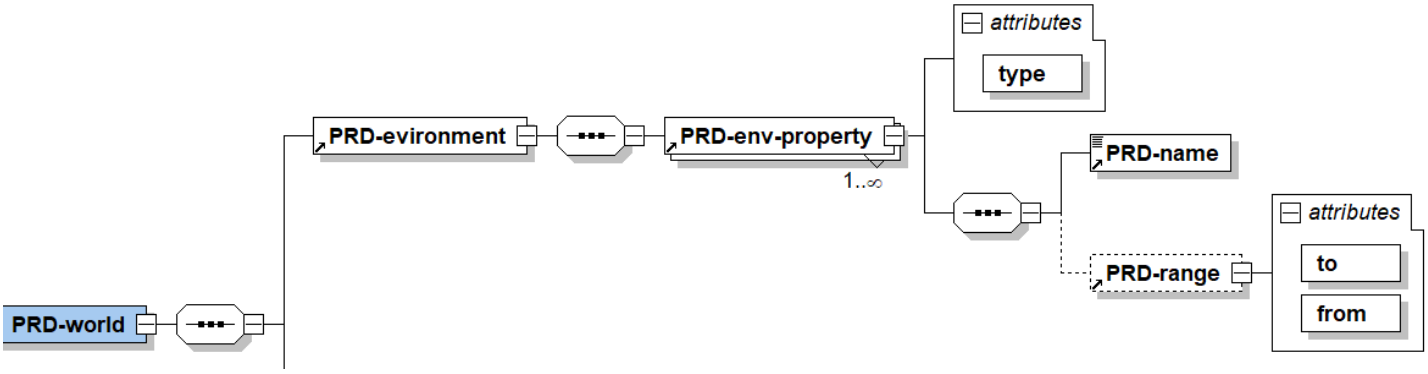
סכמה תרגיל 3

הסכמה מכילה מספר שינויים יחסית לסכמה של תרגיל 2

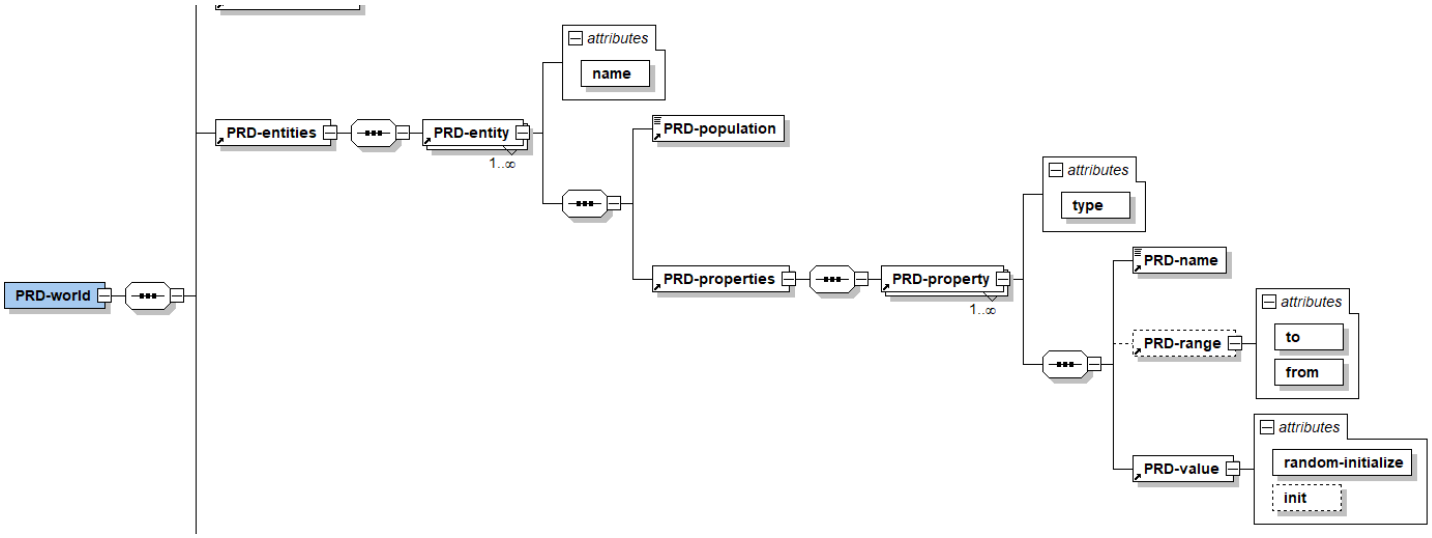
1. הסרת רכיב הגדרת גודל ה threadpool (הוא מוגדר עכשיו ידני ע"י האדמין)
2. הסרת רכיב תנאי הסיום (הם מוגדרים עכשיו במסגרת הבקשה להקצאה)
3. הוספת שם ייחודי לכל סימולציה, בדמות ה attribute על גבי האלמנט world שנקרא name.
השם הוא ייחודי ומכיל מחרוזת ללא רווחים

```
<PRD-world name="master"
```

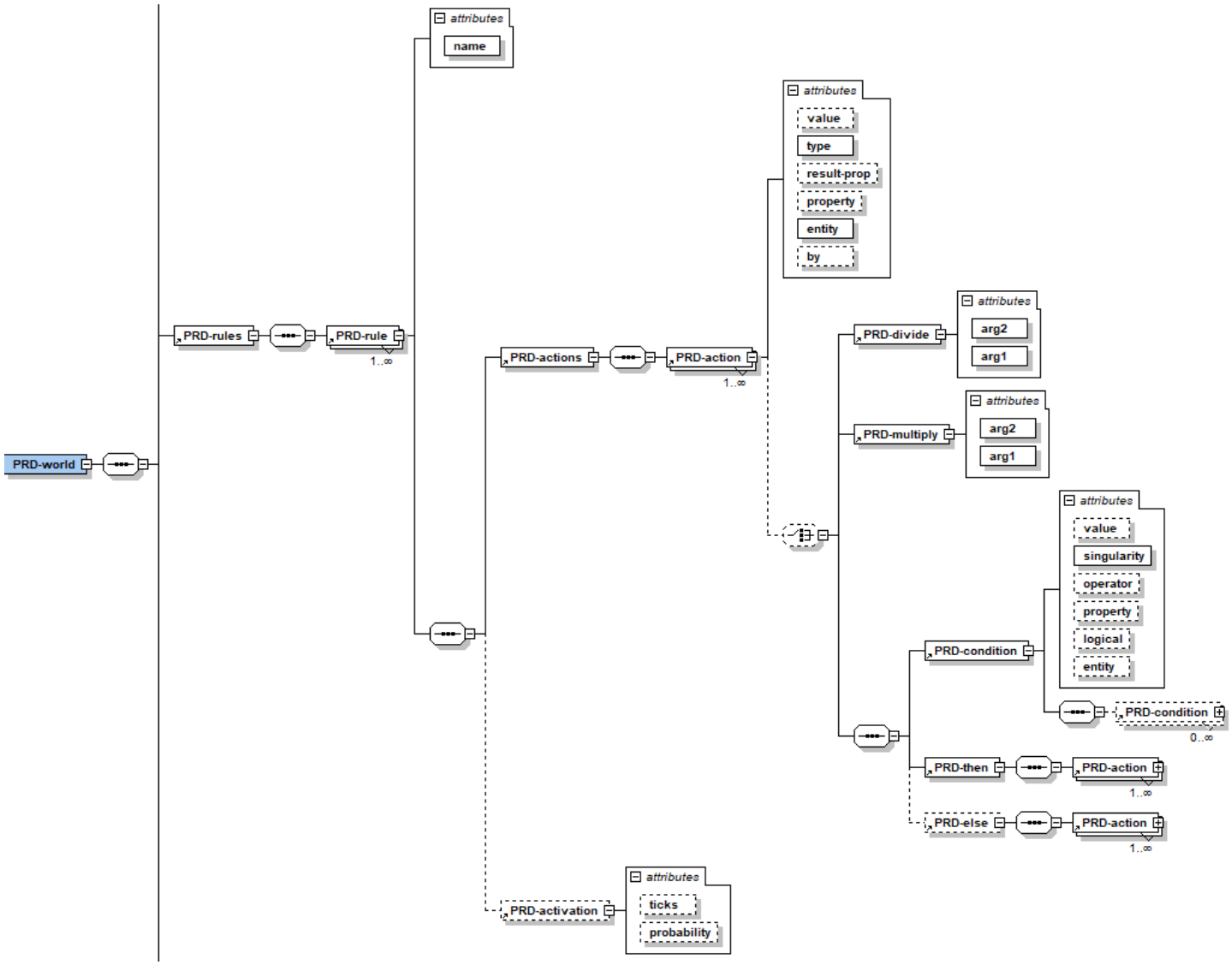
מבנה הסכמה V1 – משתני הסביבה:



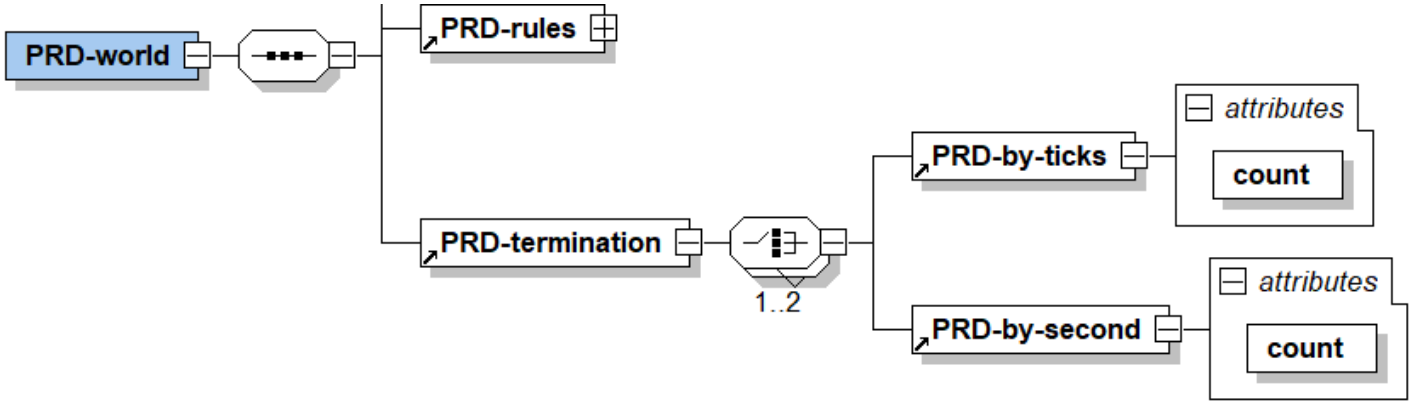
מבנה הסכמה V1 – יישויות:



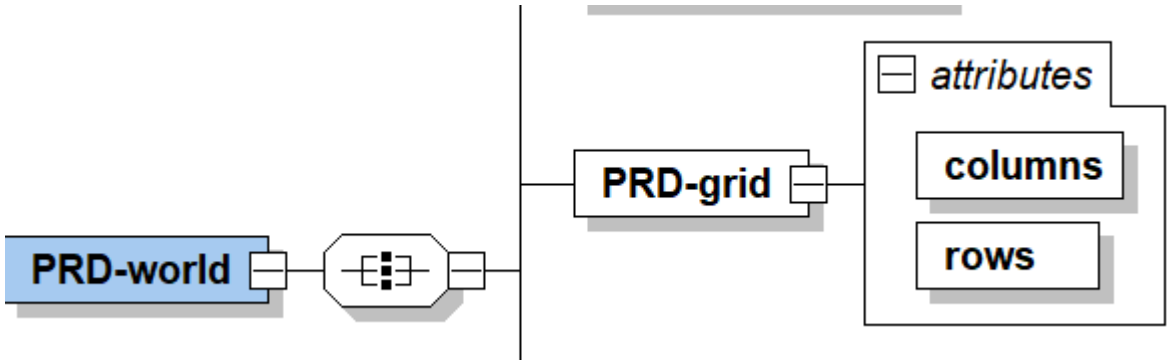
מבנה הסכמה V1 – חוקים:



מבנה הסכמה V1 – תנאי סיום:



מבנה הסכמה V2 – הגדרת המרחב:



מבנה הסכמה V2 – הרחבת ה action: