

מס' גרסה: **16**

פיתוח תוכנה מבוסס java

G.P.U.P – Generic Platform for Utilizing Processes סתיו 2021 -

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 וההטיות בצורה שווה

תוכן העניינים

[דרישות הקורס 4](#_Toc86776617)

[כללי 4](#_Toc86776618)

[איך להגיש תרגילים באיחור, ולהישאר בחיים 5](#_Toc86776619)

[הנחיות כלליות לכתיבת התרגיל 6](#_Toc86776620)

[תרגיל reflection – תרגיל עצמאי ביחידים (5%) - הגשה: 2.11.2021 8](#_Toc86776621)

[כללי 8](#_Toc86776622)

[מבנה התרגיל 8](#_Toc86776623)

[ניקוד 8](#_Toc86776624)

[איך בודקים ? 9](#_Toc86776625)

[מה מגישים ? 9](#_Toc86776626)

[מקרה בדיקה לדוגמא 9](#_Toc86776627)

[**G**eneric **P**latform for **U**tilizing **P**rocesses– **P.U.P.G** 11](#_Toc86776628)

[מטרת התרגיל(ים) בקורס 11](#_Toc86776629)

[פרולוג 11](#_Toc86776630)

[P.U.P.G 12](#_Toc86776631)

[הנחיות ספציפיות למימוש מערכת GPUP 14](#_Toc86776632)

[תרגיל 1 – מימוש **P.U.P.G** כאפליקציית Console (20%) - הגשה: 27.11.2021 15](#_Toc86776633)

[פרטים יבשים 15](#_Toc86776634)

[דרישות 15](#_Toc86776635)

[חלוקה למודולים 18](#_Toc86776636)

[איך מתחילים ? (המלצה...) 18](#_Toc86776637)

[בונוסים 19](#_Toc86776638)

[סבבה, סיימתי. מה ואיך להגיש ? 19](#_Toc86776639)

[שאלות ותשובות 19](#_Toc86776640)

[תרגיל 2 – מימוש **G.P.U.P** כאפליקציית JavaFX (35%) – הגשה: 15.1.2022 20](#_Toc86776641)

[פרטים יבשים 20](#_Toc86776642)

[דרישות 20](#_Toc86776643)

[חלוקה למודולים 23](#_Toc86776644)

[איך מתחילים ? (המלצה...) 23](#_Toc86776645)

[בונוסים 23](#_Toc86776646)

[סבבה, סיימתי. מה ואיך להגיש ? 24](#_Toc86776647)

[שאלות ותשובות 24](#_Toc86776648)

[תרגיל 3 – מימוש **G.P.U.P**כאפליקציית WEB (40%) – הגשה: 22.2.2022 25](#_Toc86776649)

[פרטים יבשים 25](#_Toc86776650)

[דרישות 25](#_Toc86776651)

[אפליקציית מנהל (Admin) 26](#_Toc86776652)

[אפליקציית פועל (Worker) 28](#_Toc86776653)

[חלוקה למודולים 31](#_Toc86776654)

[איך מתחילים ? 31](#_Toc86776655)

[בונוסים 32](#_Toc86776656)

[סבבה, סיימתי. מה ואיך להגיש ? 32](#_Toc86776657)

[שאלות ותשובות 33](#_Toc86776658)

[נספח א' – G.P.U.P Tasks 34](#_Toc86776659)

[נספח ב' – אפילוג (סוף הסיפור על G.P.U.P) 36](#_Toc86776660)

[נספח ג' – דוגמאות לפעולות על גרף ה target'ים 37](#_Toc86776661)

[נספח ד' – תיאור מבנה המערכת באמצעות קובץ XML 38](#_Toc86776662)

[סכמת קובץ XML – גרסה I 38](#_Toc86776663)

[סכמת קובץ XML – גרסה II 39](#_Toc86776664)

[סכמת קובץ XML – גרסה III 40](#_Toc86776665)

[נספח ה' – קבצי סכמה לוידוא קובץ ה XML 41](#_Toc86776666)

דרישות הקורס

## כללי

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

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

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

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

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

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

הניקוד על כל תרגיל נזקף לזכות הסטודנטים שבצעו אותו בלבד.

1. טרם הגשת התרגיל המתגלגל הראשון, תידרשו לשלוח מייל שבו מידע על צוות מגיש/י התרגיל.   
   במידה ויחול שיוי בציוות בתרגילים הבאים – יש לשלוח עדכון מתאים במייל לקראת ההגשה של התרגיל הבא.
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 איידע אתכם ואאפשר 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 להארכה.
2. במקרה של מחלה או מילואים יש להוסיף סריקה של אישור המחלה או המילואים להגשת התרגיל; בהגשת תרגיל באיחור בשל שירות מילואים יש לצרף להגשת התרגיל קובץ סרוק של טופס השחרור מהמילואים (ולא את צו הקריאה).
3. במקרה של בקשה להארכה (מכל סיבה שהיא, לרבות מילואים ומחלה) יש לפנות למרצה מראש על מנת לקבל אישור (במייל או פרונטלית בכיתה)
4. 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.
5. סטודנט המגיש תרגיל באיחור של עד שבוע עם סיבה מוצדקת ישלח את הנימוקים לאיחור ביחד עם התרגיל.
6. ניתן לערער על ציון של תרגיל לכל היותר שבוע מיום פרסום המשוב והציון במע' המאמא.

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

## הנחיות כלליות לכתיבת התרגיל

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

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

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 intelij (או כל ide אחר) ולכן זו אפילו לא אופציה.

חיסכו ממני (ומכם) את כתיבת המייל המבקש זאת.

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

הקובץ יכיל:

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

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

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

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

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

* כל הקלט והפלט בתרגילים השונים יהיה באנגלית בלבד.

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

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

* יש לדאוג שבאף מקרה לא ייזרק Exception שלא טופל ושיגרום לתעופה של התוכנית; יש לטפל ב-Exceptions בנקודה שבה ניתן לעשות כן.
* 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. המטרה היא לתרגל את הנושאים המרכזיים הנלמדים בקורס, ולא לתפור מוצר לפי דרישות של לקוח. על כן, בכל מקום שלא מופיעה דרישה מדויקת – מוטל עליכם לבחור בדרך ההגיונית ביותר שנראית לכם ולציין את בחירתכם בקובץ ה Readme אשר מוגש עם התרגיל. אם יש ספק לגבי אופן פעולתכם אתם מעודדים לשאול האם הפתרון שאתם חושבים לתת לסוגיה מסוימת הוא קביל ולגיטימי (שאלות בפורום, מייל למרצה וכו)
* התרגיל מתקיים כולו במסגרת ג'אווה גרסה 8. הקפידו להוריד, לעבוד, לקמפל ולהריץ עם הגרסה המתאימה בלבד.
* **ווידוא הגשת התרגיל טרם הגשתו:**
* **יש לוודא כי ההגשה שלכם רצה היטיב על מע' נקייה, באופן שבו גם הבודק יריץ אותה,** על מערכת נקייה וללא תוצרי לוואי אחרים של הפעלות קודמות שלכם.
  + **הבודק יבצע את הבדיקה על מע' windows 10. כל מי שמפתח על גבי mac/linux – זכותכם – אבל גם חובתכם לוודא כי אתם רצים היטיב על windows 10. למען הסר ספק, לא תתבצע בדיקה על מע' הפעלה אחרת. כמו כן לא תהיה התחשבות בתקלות שמקורם רק בשל עבודה על מע' הפעלה שונות (ולא שאמורות להיות תקלות כאלה..)**
  + **יש לוודא כי כל קבצי הבדיקה השונים שהועלו ל mama נטענים בהצלחה ע"י המע' שלכם טרם ההגשה. בדיקת הבודק תתחיל מבדיקה בסיסית המבוססת בצורה גסה על קבצים אלה. חבל ליפול Level 0 על שטות שיכולתם לעלות עליה בשנייה עוד בשלב הפיתוח.**
  + **כאמור, הגשה חוזרת בשל תקלות level 0 תתחיל מראש מציון של 90. בלי שום יוצא מן הכלל. הקדימו תרופה למכה.**
* **בחלק מהתרגילים ניתנת אפשרות למימוש דרישות בונוס.**
  + **ישנם 2 סוגי בונוסים:**
    1. **בונוס בתוך טווח התרגיל - יכול להביא אתכם לכל היותר לציון 100, ולחפות במקרה והורדו לכם נקודות בשל תקלות.**
    2. **בונוס מחוץ לטווח התרגיל – יכול להעלות את ציונכם אף מעבר ל 100 (וכן, יש כפל מבצעים לטובת הלקוח).**
  + **בכל מקרה יש לבצע את הבונוס אם ורק אם סיימתם את כל דרישות הבסיס ההכרחיות לתרגיל.**
  + **חלק מהבונוסים בתרגילים השונים הם כאלה שנועדו "להקדים תרופה למכה" – מימוש דרישה בתרגיל n אשר בכל מקרה תגיע כדרישה חובה בתרגיל n+1.**

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

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

* + **פירוט הבונוסים, משקלם ונקודותיהם מפורט בגוף התרגיל הספציפי.**
  + **אם כבר מממשים בונוס, יש לממש את כולו, עפ"י דרישתו כדי לזכות במלוא הניקוד שהוא מקנה. בכל מקרה ההחלטה על ניקוד הבונוס היא בידי הבודק/מרצה בלבד (אל דאגה, 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: 2.11.2021

## כללי

**מועד הגשה: 2.11.2021 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();  
 }  
}

**G**eneric **P**latform for **U**tilizing **P**rocesses– **P.U.P.G**

## מטרת התרגיל(ים) בקורס

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

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

חלק ג' – ביצוע משימות בצורה מבוזרת תוך הסתמכות על ארכיטקטורת Client-Server

## פרולוג

בשנת 2009 התחלתי לעבוד בחברת Applied Materials במשרה מלאה (כג'וניור).  
השתלבתי במוצר ה Mask Inspection אשר פותח (בעיקר) ב (unmanaged) C++ , באמצעות visual studio 6.   
המוצר היה מורכב מ(בערך) 130 פרוייקטים מסוגים שונים (dll, tlb, exe וכו').

כלל המע' הייתה מתוארת באמצעות קובץ טקסט שבו כתבו המתכנתים את הנתיבים אל הפרוייקטים השונים המרכיבים את המוצר. הנתיב היה אל קובץ dsp – קובץ המתאר פרוייקט ב visual studio 6. ככל שעבר הזמן הקובץ הזה הלך והתארך (והגיע, כאמור, לבערך 130 פרוייקטים ב 2009)  
כדי לקמפל, לבנות ולייצר מופע של המע', המתכנת היה מריץ סקריפט מיוחד (VBS) שקורא את הנתונים מקובץ הטקסט, שורה אחרי שורה, ומפעיל את הקומפיילר של C++ (MSDEV) על כל אחד מהפרוייקטים.   
בסיום (בשאיפה מוצלח) של הקימפול של פרוייקט X היה עובר הסקריפט לפרוייקט הבא בתור אחריו וחוזר חלילה עד למעבר מלא על כלל הפרוייקטים...

התהליך שבו בונים את כל המוצר נקרא בפשטות build.

מודל הפיתוח הנ"ל הוכח כבעייתי ביותר:

* תהליך ההרצה היה סדרתי. קימפול מלא של כל המוצר היה לוקח 11 שעות (!) ועל כן בדר"כ הושאר לעבודה במהלך הלילה כדי שיהיה מוכן בבוקר. אם היה נכשל – היו מגלים זאת רק בבוקר (!!), מנסים לתקן, ומפעילים עוד build בלילה הבא... (!!!)   
  וחוזר חלילה.   
  כן.
* בין הפרוייקטים השונים התקיימו תלויות כאלה ואחרות.   
  אם פרוייקט X תלוי בפרוייקט Y חובה לקמפל (ובהצלחה...) את פרוייקט Y לפני שמתחילים לקמפל את פרוייקט X.  
  המתכנתים שהוסיפו פרוייקטים חדשים לא תמיד ידעו איזה פרוייקט תלוי במי (ובוודאי לא ידעו למה).   
  בדר"כ הוסיפו את הפרוייקט שלהם בסוף קובץ הטקסט והריצו את הסקריפט עד שראו שעבר בהצלחה.   
  אם נכשל (וספציפית בגלל בעיית תלויות) – היו תוקעים את הפרוייקט שלהם איפשהוא באמצע ו... מקווים לטוב.   
  כן.
* לעיתים היו תקלות במהלך הבילד, ואחד הפרוייקטים באמצע נכשל להתקמפל (לאו דווקא בגלל בעיית תלויות).   
  כישלון בבניית פרוייקט אחד היה חודל ועוצר את כל תהליך הבילד (!).   
  הדבר היה מתגלה רק למחרת, (בשאיפה) מתוקן בהצלחה ו... צריך להריץ את כל הבילד מחדש.   
  לא הייתה אפשרות להתחיל רק מהנקודה האחרונה שבה נכשלנו.   
  כן.
* לאף אחד לא היה באמת מושג מהי מפת התלויות השונות בין הפרוייקטים. הסוגיה הזו יכולה להיות סופר קריטית במקרים שבהם מגיעים למסקנה שצריך לשנות/לתקן/להנדס מחדש פרוייקט מסויים.   
  איך יודעים מי הם כלל הפרוייקטים התלויים בו ? האם שינוי בו יגרור שינוי בפרוייקטים אחרים ? כמה יהיה זה "מסוכן" לבצע שינויים בפרוייקט כזה או אחר ? ניתוח ידני של התלויות כפי שהן נגזרות מקובץ התיאור של כל אחד מהפרוייקטים (dsp) התברר להיות מסובך ביותר.

## P.U.P.G

GPUP היא מע' גנרית לייעול ושיפור תהליכים.   
המע' נוצרה בעקבות מכלול הבעיות המתוארות לעיל, והורחבה בהמשך לתמוך במגוון פתרונות לתהליכים אחרים הקשורים גם הם לתהליך הפיתוח.

הרעיון הבסיסי העומד בלבה של GPUP מושתת על 2 רכיבים עקריים:

**ניהול של גרף תלויות של מידעים שונים:**

חישבו על גרף (מתורת הגרפים, לא ממתמטיקה), המכיל צמתים המקושרים לצמתים אחרים.

כל צומת בגרף נקרא target. ל target יש שם והוא יכול גם לשאת במסגרתו מידע חופשי שיכול לשמש את המע' בעתיד (ראו פרטים בהמשך).

כל target יכול להיות תלוי ב target'ים אחרים, או שאחרים יכולים להיות תלויים בו.  
Target יכול להיות תלוי ביותר מ target אחד ויכולים להיות תלויים בו יותר מ target אחד. (התלויות ממודלות בדמות קשרים בין הצמתים בגרף)

Target'ים שאינם תלויים באף target אחר – נקראים עלים (Leafs) ; Target'ים שאינם נדרשים עבור אף taget אחר נקראים שורשים (Roots) ; Target'ים שגם תלויים באחרים וגם נדרשים עבור אחרים – נקראים אמצעיים (Middles).  
target'ים שאינם נדרשים עבור אף אחד ואינם תלויים באף אחד נחשבים עצמאיים (independent).  
יכולים להיות מספר בודד או רב של leafs, roots, middles, independents (שכן מדובר על גרף ולא על עץ).

הנה גרף תלויות לדוגמא:

**Root**

**Middle**

**Leaf**

**Top-Down  
Depends On**

**Bottom-Up  
Required For**

**Independent**

בהינתן גרף תלויות, GPUP תדע להציע למשתמש מגוון כלים לניתוח והסקת מסקנות מן הגרף. בין היתר (אך לא רק):

* בחינה קלה ומהירה של כל target ומה סטטוסו (leaf | root | middle | independent). במי הוא תלוי. מי תלוי בו. תלויות ישירות, תלויות עקיפות וכו'.
* איתור מסלול בין 2 tareget'ים: אפשר יהיה לאתר בקלות את מפת הקשרים בין 2 target'ים.   
  יכול להיות יותר ממסלול אחד בין 2 target'ים.
* איתור מעגלים בגרף.
* תרחישי :What-if מי הם סך ה target'ים שכך או אחרת, בצורה ישירה או עקיפה, מושפעים/תלויים ב target מסויים

**ביצוע משימות (Tasks) על גרף התלויות:**

משימה (task) היא פיסת קוד לביצוע הפועלת “על” target ועל המידע האגור בתוכו.   
המשימה מקבלת כ input את המידע האגור ב target מסויים ועושה איתו משהו.

מודל ביצוע המשימות ב GPUP עובד בצורה של Bottom-Up:

ראשית מתחילים לבצע את המשימה על כל ה target'ים שאינם תלויים באף target אחר. זה יכלול כמובן את כל ה Leafs וה Independents.   
כל משימה יכולה להסתיים בהצלחה (Success), כשלון (Failure), או בהצלחה עם אזהרה (Warning).  
ברגע שעיבוד משימה על target מסתיים בהצלחה (גם עם אזהרה) – הוא מוריד חסם אחד (את עצמו) מהפעלת המשימה על כל ה target'ים אשר תלויים בו. ברגע שכל התלויות של target מסויים עובדו בהצלחה – הרי שגם הוא מוכן עכשיו לעיבוד ע"י ה task...

מטבע הדברים, בתחילת הפעלת משימה, יכולים להיות מועמדים רבים לריצה.   
למעשה אין כל מניעה להריץ במקביל את ה task על כולם – היות והם עצמאיים ולא תלויים באף אחד.   
כמו כן תו"כ הפעלת משימה יכולים להיפתח ולהתאפשר לביצוע גם ה middles וה roots (ואפילו במקביל אחד לשני ול leafs או ל independents) וככל שנוכל להריץ במקביל את המשימה על כל target שמאופשר לריצה – הרי שנוכל לסיים את עיבוד ה task על כל הגרף בצורה מהירה יותר.

תו"כ הרצת ה task (בשאיפה במקביל) על גבי הגרף נוכל לאסוף מידעים שונים ומגוונים ולדעת בכל רגע נתון כמה היינו יעילים והצלחנו למצות את כל המשאבים העומדים לרשותינו אל מול מצבים שבהם יש משאבים שמוכנים לביצוע task אבל אין להם יכולת להתקדם. (חישבו על היחס בין ה target'ים המאופשרים לביצוע לכמות המשאבים שיכולה לבצעם ברגע נתון.)

במידה וביצוע ה task על אחד ה target'ים נכשל, הרי שכל ה target'ים התלויים ב target זה "ומעלה" לא יוכלו להתבצע. אין הדבר מפריע ל task לרוץ על target'ים אחרים המאופשרים להרצה ובכך למעשה למצות כמה שאפשר את ההרצה של ה task על כל ה target'ים האפשריים.

במהלך ההרצה כל target יכול להימצא באחד מהמצבים הבאים:

**FROZEN** – targets שעוד אי אפשר לרוץ עליהן מאחר ועדיין לא עברו בהצלחה כל תלויותיהן.   
**SKIPPED** – targets שלעולם לא נוכל לרוץ עליהן מאחר ויש מי מתלויותיהן (אחת או יותר) שהעיבוד שלה כבר נכשל.

**WAITING** – targets הנמצאים בתור, וברגע שיהיה משאב פנוי ניתן יהיה לבצע אותן. (בתחילת ההרצה כל ה Leafs וה Independents מוכנסים לתור...)

**IN PROCESS** – targets שכרגע מתבצעת עליהם המשימה (כאמור, יכולים להיות מספר כאלה)

**Finished** – targets שהרצתם כבר הסתיימה. לכל target שהרצתו הסתיימה אפשר לראות אם הוא הסתיים בהצלחה, כישלון או עם אזהרות.

בזמן ביצוע של משימה על כל target נוצר פלט המתאר את מהלך פעולת המשימה:

1. פלט המבשר על התחלת הפעלת המשימה על ה target המדובר
2. פלט המתאר את המידע החופשי הנישא על גבי ה target (אם וככל שקיים כזה)
3. פלט המבשר על סיום הפעלת המשימה על ה target המדובר.   
   יש לציין את האופן שבו הסתיימה ההרצה: Success | Warning | Failure
4. אם בסוף העיבוד של המשימה על ה target המדובר (בין אם בהצלחה ובין אם בכשלון) יש target(ים) שתלוי(ם) בו ישירות שעכשיו "נפתח(ו)" לביצוע (כלומר כל ה target'ים שהם תלויים בהם כבר עובדו עם תוצאה כזו או אחרת) – יש לציינם.   
   שימו לב: יש לציין אותם בכל מקרה, וללא שום קשר למצבם (Skipped\Frozen)
5. אם עיבוד ה target מסתיים ב Failure, בנוסף לסעיף 4, יש לכלול גם את את כל ה target(ים) העקיפים (עד ל Roots), שלא יוכלו להתבצע בכלל. גם כאן, כאמור, ללא קשר למצבם (Skipped\Frozen).

המידעים הנ"ל רלבנטים לכל סוגי המשימות באשר הן.

בין פלט 2 ל 3 כל task ייתן את הפלט הייחודי לו כפי שמוגדר [בנספח א'.](#GPUP_TASKS)

בסיום ההרצה אפשר יהיה לקבל דו"ח מסודר מה רץ בהצלחה, מה נכשל (ולמה) ומה בכלל לא אופשר להרצה (בשל מערך התלויות).   
ברצות המשתמש הוא יוכל להפעיל מחדש את עיבוד ה task על הגרף – אבל הפעם רק על ה target'ים שנכשלו/שהיו חסומים לביצוע.

במהלך התרגיל נממש מספר סוגים של משימות (המתוארות כולן [בנספח א'](#GPUP_TASKS).)

## הנחיות ספציפיות למימוש מערכת GPUP

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

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

1. **כחלק מהמע' תצטרכו לחשוב ולבחור לבד את מבני הנתונים השונים שישרתו את הצרכים של דרישות המע'. זהו לא קורס במבני נתונים או באלגוריתמים, ומבני הנתונים/אלגוריתמים שתבחרו לממש לא חייבים להיות היעילים ביותר או האופטימליים. מספיק שהם יעבדו בצורה נכונה (ללא טעויות) ובזמן סביר.**
2. **המע' כולה תיכתב ותורץ בסביבת העבודה של ג'אווה 8**

תרגיל 1 – מימוש **P.U.P.G** כאפליקציית Console (20%) - הגשה: 27.11.2021

## פרטים יבשים

צפי תחילת עבודה: **31.10.21** תאריך הגשה: **27.11.21**

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

משקל התרגיל: **20%** קושי: **סביר**

**מטרת התרגיל העיקרית**

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

## דרישות

1. בתרגיל זה תממשו את מנוע המערכת, אשר יודע לקבל נתונים על מבנה גרף ה targets ויאפשר למשתמש להפעיל עליו משימה אחת בלבד (משימת הסימולציה).

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

1. פרטיה הטכנים של המערכת יינתנו מקובץ XML (כמפורט [בנספח](#xml) ד').

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...)

במידה והקובץ מתגלה כתקול, יש להודיע על כך בצורה מסודרת ונאותה למשתמש.   
יש לפרט מהי סיבת התקלה (ולא להסתפק במשפט סתום-בלום כמו: "הקובץ תקול...").

בפרט יש לוודא את הפרטים הבאים:

* 1. הקובץ אכן קיים, והוא מסוג XML (די לבדוק לשם כך כי הוא נגמר בסיומת xml)
  2. שם של target הוא ייחודי ולא חוזר על עצמו
  3. במהלך תיאור תלויות של target (בין אם DependsOn או RequiredFor) יש לוודא כי ה target אכן קיים.
  4. אין קונפליקט בין תיאור סוגי התלויות:  
     if x dependsOn y then y can’t be dependsOn x  
     if x requiredFor y then y can’t be requiredFor x

1. בתרגיל זה תתמכו במשימה אחת בלבד: משימת הסימולציה.

משימה זו למעשה מקבלת target ולא עושה איתו כלום אלא רק מעבירה פרק זמן המדמה עיבוד.   
בסיום "העיבוד" (לכאורה) המשימה תחזיר תוצאה שנבחרת בצורה רנדומלית.

פרטי המשימה מוסברים [בנספח א'](#GPUP_TASKS).

1. בהפעלת משימה יווצר בספריית העבודה של GPUP (הנתיב אליה מתואר גם הוא בקובץ ה XML) ספרייה הנושאת את שם ה task וחותמת זמן ההרצה. לתוך ספריה זו יווצרו כל הפלטים של הרצת ה task. שם הספרייה יהיה:

<task name> - <Execution date in format: dd.mm.yyyy HH.MM.SS>

למשל:

Simulation - 23.04.2024 23:22:45

(שימו לב כי זהו לא בהכרח הכתיב של הפורמט המלא בג'אווה. זו רק אלוסטרציה. חפשו על SimpleDateFormat)

הפלט של כל הרצה של המשימה על target מסויים יודפס למסך כמו גם יישמר בקובץ ייעודי תחת ספריית ההרצה של ה task. שם הקובץ המתאר את פעולת ה task על target מסויים יראה כך:

<target name>.log

למשל:

A.log

1. יש לוודא תקינות קלט כחלק מכל אינטרקציה עם המשתמש בכל מקום שבו זה רלבנטי:

אם אתם מצפים לקבל מספר – לא לקרוס כי הכניסו לכם בטעות (או בכוונה) טקסט וכו'.

בכל מקרה של תקלה יש להיות מאוד ברורים במסר שמעבירים חזרה למשתמש: מה קרה ? מה הייתה מהות התקלה ? היכן שזה רלבנטי, איך לתקנה וכו'.

חישבו איך להיות ידידותיים למשתמש ולעולם אל תניחו כי מי שמשתמש באפליקציה שלכם הוא מתכנת בעצמו או מישהו שמגיע מהתחום ו"מכיר" איך דברים עובדים לבד. (זה הזמן לחשוב על...)

1. **אין צורך** להשתמש בצבעים שונים במהלך תרגיל זה בעת ההדפסה ל console.

יתרה מזאת, ישנו צפי רב (ניסיון מהסמסטרים הקודמים) כי ניסיון לעשות כן תוך שימוש בספריות צד שלישי קורס אצל הבודק,

מעוות את כל תצוגת המסך וגורם לחוסר יכולת לבדוק את ההגשה.

גם אם בדקתם את זה אצלכם וזה עבד.

גם אם בדקתם במחשב של השכנה וזה עבד.

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

1. עליכם לכתוב ממשק משתמש בתצורת console.

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

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

**שימו לב**: ישנן פקודות שאין הגיון לבצע אותן אם לא קדמו להם פקודות אחרות. במידה וזה קורה יש להציג הודעת שגיאה רלבנטית למשתמש ולאפשר את המשך מהלך פעילות המע'.

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

יש לוודא תקינות הנתונים המופיעים בקובץ כפי שהוגדרו בסעיף 2.

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

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

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

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

**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 מידע כללי על גרף ה targets

פקודה זו תאפשר למשתמש לקבל מידע על כלל הגרף.

יש להציג:

1. כמות target'ים
2. כמה target'ים הם| Independents Leaf | Middle | Root

**דגשים:**

* + - אין לאפשר הרצת פקודה זו אם טרם נטען קובץ (תקין) במע'.

1. הצגת מידע על target

פקודה זו תאפשר למשתמש לקבל את המידעים השונים על target שהוא בוחר.

יש לבקש מהמשתמש את שם ה target. במידה והוא לא קיים – יש להודיע על כך בצורה מסודרת למשתמש.

אם ה target קיים במע', יש להציג עליו את הפרטים הבאים:

1. שם ה target
2. מיקום ה target: Leaf | Middle | Root | Independent
3. רשימת כל ה target'ים שהוא תלוי בהם בצורה ישירה (אם לא תלוי באף target – יש לציין זאת).
4. רשימת כל ה target'ים שהוא נדרש להם בצורה ישירה (אם לא נדרש לאף target – יש לציין זאת).
5. מידע כללי שנישא על גביו (אם קיים כזה. אם אין יש לציין זאת)

**דגשים:**

* + - אין לאפשר הרצת פקודה זו אם טרם נטען קובץ (תקין) במע'.
    - אם המשתמש הכניס target שלא קיים – הודיעו לו על כך בצורה נעימה ומכובדת והישארו במסגרת הפקודה.
    - בכל מקרה אפשרו למשתמש לחזור חזרה לתפריט הראשי (אם אין לו מושג איך קוראים ל target'ים למשל)

1. מציאת קשר (מסלול) בין 2 targets

פקודה זו תאפשר למשתמש להכניס 2 שמות target'ים (מקור ויעד) כמו גם את סוג היחס המבוקש בין המטרות (Depends On או Required For) והמע' תציג האם יש רצף של תלויות בין 2 ה target'ים הללו, על פי היחד המבוקש.

שימו לב: יכול להיות יותר ממסלול אחד. יש להציג את כל המסלולים האפשריים.

ראו [נספח ג'](#graph_examples) לדוגמאות.

**דגשים:**

* + - אין לאפשר הרצת פקודה זו אם טרם נטען קובץ (תקין) במע'.
    - אם המשתמש הכניס target שלא קיים – הודיעו לו על כך בצורה נעימה ומכובדת והישארו במסגרת הפקודה.
    - בכל מקרה אפשרו למשתמש לחזור חזרה לתפריט הראשי (אם אין לו מושג איך קוראים ל target'ים למשל)

1. הפעלת משימה

פקודה זו תאפשר למשתמש להפעיל את משימת הסימולציה על הגרף.

אפשרו למשתמש להכניס את הפרטים הנדרשים לטובת הפעלת משימת הסימולציה (כפי שהם מוגדרים [בנספח א'](#GPUP_TASKS)).

אפשרו למשתמש לבחור האם להתחיל את העיבוד מחדש (From Scratch) או לבצעו רק על השארית הנדרשת מההרצה הקודמת (ככל שהייתה כזו) – Incremental.   
(אם בוחרים incremental ולא היה עיבוד מקדים – הרי שזה כאילו בחר from scratch. יש ליידע על כך את המשתמש)

לאחר מילוי כל הפרטים המע' תתחיל לבצע את ה task על גרף ה targets.   
המנגנון ירוץ כפי שמתואר בחלק הכללי (מ Leafs to Roots).   
כחלק מההרצה תיווצר הספרייה הייעודית עבור הרצה זו ובתוכה יווצרו הקבצים הרלבנטים כפי שמוגדר בסעיף 4.

ההרצה תתבצע בצורה סידרתית. המשתמש יוכל לראות את הפלטים של הריצה של ה task על כל target.

בסיום ההרצה על כל הגרף יש להציג את סיכום ההרצה. הסיכום יכלול:

* כמה זמן לקחה כל ההרצה (פורמט של HH:MM:SS)
* סטטיסטיקה על תוצאות עיבוד ה targets (כמה הסתיימו ב Success | Warning | Failure | Skipped)
* רשימת כל ה target'ים ועבור כל אחד מהם:
  + שם
  + תוצאת הריצה: Success | Warning | Failure | Skipped
  + זמן ריצה (אם היה כזה) - פורמט של HH:MM:SS

**דגשים:**

* + - אין לאפשר הרצת פקודה זו אם טרם נטען קובץ (תקין) במע'.

1. יציאה מהמע'

פקודה זו יוצאת מהמערכת.

## חלוקה למודולים

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

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

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

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

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

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

## איך מתחילים ? (המלצה...)

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

התחילו בבניית התשתית המתארת את גרף ה targets. החליטו באיזה מבני נתונים תשתמשו ואיך למדל את התלויות השונות בין ה target'ים השונים.

המשיכו בעבודה על פקודות 2,3 ו 4 אשר יגרמו לכם להתחכך ולהבין אם מבני הנתונים שבחרתם לשמירת הגרף אכן הולמים את הדרישה ועומדים בציפייה. עבור כל פקודה בצעו את המסלול המלא החל מה UI, העברת המידע למנוע, קבלת התשובה והצגתה למשתמש.

עיברו לעבוד על היכולת להריץ task. ממשו את מנגנון ההרצה של ה task. הגדירו את task הסימולציה והפעילו את התוכנית בעזרתו.

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | הקדמת תרופה למכה | מעגלים:  ממשו את היכולת לאתר אם target מסויים נכלל במעגל או לא.  אם כן יש להציג את פרטי המעגל. ייתכן והוא שותף ביותר ממעגל אחד. יש להציג אחד מהם ואין צורך להציג את כולם. לטובת בונוס זה הוסיפו פקודה חדשה בתפריט. | כי גם ככה תממשו את זה בהמשך... | 5 נקודות (עד ל 100) |
| 2 | מגניב לאללה ! | שמירת מצב מע':  בבונוס זה תוסיפו פקודה נוספת לתפריט שתאפשר למשתמש לשמור את כל מצב המע' לקובץ. פורמט הקובץ נתון לבחירתכם. עליכם לבקש מהמשתמש נתיב מלא לשם קובץ. אתם יכולם לבחור איזו סיומת שתרצו עבור הקובץ.  יש לשמור את כל המצב של המע', ובפרט את נתוני גרף המטרות הטעונים כרגע במע', נתוני הריצה האחרונה שבוצעה (ככל שבוצעה כזו וכו').  בנוסף הוסיפו למשתמש פקודה נוספת שתאפשר לו בעליית המע' לבחור לטעון את המצב האחרון שנשמר (במקום לטעון קובץ XML כרגיל).  זו יכולה להיות פקודה נוספת או כתת סעיף בפקודה מס' 1 של טעינת הקובץ.  המשתמש יכניס נתיב מלא לקובץ (ללא הסיומת) שנשמר באמצעות הפקודה הקודמת.  לאחר הטעינה המע' חוזרת למצב הקודם בו שהתה בצורה מדוייקת. המשתמש יוכל לשאול שאלות על הגרף שטעון בה, להריץ עליה טסקים (אפילו incremental במידה וכבר הייתה שמורה ריצה) וכו'. | 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, טבלת תוצאות הריצה מסעיף 7, כמו גם כל דבר נוסף העולה על דעתכם שחשוב שהבודק תדע.
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 מהבונוסים לעיל, ציינו זאת בקובץ ה readme כדי שהבודק ידע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

## שאלות ותשובות

|  |  |  |  |
| --- | --- | --- | --- |
| # | תאריך | שאלה | תשובה |
| 1 | 2.11.21 | האם יש משמעות לסדר המטרות בפקודה מס' 4 | אכן. יש להגדיר את מטרת המקור ומטרת היעד. בנוסף יש להגדיר את סוג היחס המבוקש בין 2 המטרות הנ"ל (Depends On או Required For) ואז להציג את כל המסלולים בין המקור אל היעד על פי היחס המדובר. |

תרגיל 2 – מימוש **G.P.U.P** כאפליקציית JavaFX (35%) – הגשה: 15.1.2022

## פרטים יבשים

צפי תחילת עבודה: **12.12.21** תאריך הגשה: **15.1.22**

צפי זמן לביצוע: **5 שבועות** ציון אפשרי מקסימלי: **120**

משקל התרגיל: **35%** קושי: **מאתגר**

**מטרות התרגיל העיקרית**

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

## דרישות

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

בתרגיל זה אין יותר סדרה של "תפריטים" כפי שהיו בתרגיל 1. כל ממשק המשתמש הוא ממשק גרפי בלבד, המבוסס על JavaFX. כל האינטראקציה עם המשתמש מתרחשת דרך פעולותיו על גבי הפקדים השונים שתציגו לו (כפתורים, תיבות טקסט וכו').   
בתרגיל זה נאפשר למשתמש שליטה ומידע רב יותר על מבנה הגרף, הרצת משימות ועל הפלטים השונים.

1. בתרגיל זה תתמכו ב serial sets:

Serial set מגדיר קבוצה של targets שאין לבצע הרצה שלהן במקביל.   
(ייתכן וכולן מסתמכות על משאב משותף והרצתן במקביל יכולה להרוס הרצה של אחרות וכן הלאה).

target יכול להשתתף בכמה serial sets או בכלל לא.

המידע לגבי serial set מגיע מתוך קובץ ה XML.

1. העבודה במע' תחל בטעינת קובץ XML (כשם שהיה בתרגיל 1).

מרבית המידע נשאר זהה לקובץ ה XML של התרגיל הקודם, עם מעט שינויים ותוספות הנדרשים לתמיכה בתרגיל 2.

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

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

יש לבצע בדיקת תקינות על תוכנו של קובץ ה XML. כל הבדיקות כפי שבצועו בתרגיל 1 ובנוסף:

* יש לוודא כי אין הכוונה ל target שאינו מוגדר דרך המבנה החדש של ה serial set.
* אין Serial Sets בעלי אותו שם.

1. הצגת מידעים:

אפשרו למשתמש לראות את מבנה הגרף בצורה "נעימה" יותר:  
הציגו טבלה המכילה את כל ה targets בצורה "שטוחה". (למתעניינים לגבי תצוגת עצית – ראו בונוס 5).

מומלץ לעבוד עם table view. יש דוגמא בסיסית עליו בקובץ הדוגמאות של java fx (תחת advance) אבל תכלס כדאי וצריך ללמוד על אופן פעולתו לבד)

לטבלה יהיו העמודות הבאות:

* שם ה target
* מיקום (Leaf | Middle | Root | Independent)
* כמות Depends On ישירים | סה"כ כמות depends On (ישירים + כמות תלויות טרנזטיביות)
* כמות Required For | סה"כ כמות required For (ישירים + כמות תלויות טרנזטיביות)
* מידע חופשי שנישא על גבי ה target (ככל שקיים כזה)
* כמות ה serial sets שהוא משותף להם

אפשרו למיין את הטבלה על פי כל אחת מהעמודות.

הציגו אזור של סיכומים:

* כמות target'ים
* כמה בכל מיקום (Root | Middle | Leaf | Independents)

הציגו מידע לגבי ה Serial Sets הקיימים במע'. עבור כל Serial Set:

* שם
* רשימת המטרות הנכללות בו
  1. הציעו למשתמש מגוון פעולות על הגרף:
     1. מציאת מסלול בין 2 target'ים: אפשרו למשתמש לבחור 2 מטרות (אחת מקור והראשונה יעד) ואת היחס בינהן ואז להציג את כל המסלולים האפשריים הקיימים בין 2 המטרות (כשם שהיה בתרגיל 1)
     2. מציאת מעגל: אפשרו למשתמש לבחור target והציגו האם הוא חלק ממעגל.   
        ייתכנו הרבה מעגלים. יש להציג אחד מהם.
     3. What-if: אפשרו למשתמש לבחור target מסוים ולדעת מי היא קבוצת כל ה target(ים) המושפעת ממנו.

אפשרו למשתמש לבחור האם הוא מעוניין בכיוון של מעלה (required for) או בכיוון של מטה (depends on), ואז הציגו את רשימת כל ה targets מכל הרמות על פי בחירת המשתמש.

ראו [נספח ג'](#graph_examples) לדוגמאות

* 1. בתרגיל זה תתמכו בהרצת משימה נוספת: משימת הקימפול.   
     לשם הפעלת משימה זו יטען כמובן גרף המכיל target'ים המתאימים לה.  
     פרטי המשימה והגדרותיה נמצאים [בנספח א](#GPUP_TASKS)'.
  2. הרצת משימה על הגרף:

הגדירו כפתור ייעודי שבאמצעותו המשתמש יתחיל את תהליך הרצת המשימות על הגרף.

הרצת משימה תכלול מספר שלבים:

* + 1. אפשרו למשתמש לבחור את ה targets עליהם הוא רוצה לבצע את המשימה.   
       המשתמש יוכל ביתר קלות לבחור את כל ה targets, חלק מהם (בחירה פרטנית של target בודד אחד אחרי השני) או לבחור באמצעות טכניקת ה what-if: בחירה של target מסויים + סימון כיוון (dpends on \ required for) והמע' תבחר אוטומטית את כל ה targets המושפעים (כולל טרנזטיביות). באותו אופן אפשרו למשתמש לבטל בכפתור אחד את הבחירה של כל ה target'ים הבחורים כרגע.  
       (אני ממליץ לעבוד עם מצב של checkbox ברשימת המטרות דרכם ניתן יהיה לסמן אילו מטרות חפצים ביקרם)
    2. בחירת המשימה לביצוע (סימולציה או קימפול).   
       לכל משימה אפשרו למלא את הפרמטרים הרלבנטים לה (כמוגדר [בנספח א'](#GPUP_TASKS))
    3. בחירת מאפיינים להרצה:
* האם להריץ מההתחלה (from scratch) או בצורת incremental. שימו לב: Incremental מניח כי יש סטטוס הרצה קודם לכל המטרות הנבחרות. עליכם לוודא כי אכן לכל המטרות שנבחרו להרצה (בסעיף 7.1) יש סטטוס הרצה קודם ורק אז לאפשר Incremental. אם לא כך הדבר – יש להודיע למשתמש ולעדכן אותו כי הריצה תהיה בתצורה של from scratch.
* בחירת כמות ה thread'ים להרצה  
  מספר שלם החל מ 1 ועד לנתון ה Max-parallelism (המגיע מקובץ ה XML)
  + 1. יש להציג למשתמש כפתור ייעודי שבאמצעותו הוא ייבחר להפעיל את הרצת ה task.  
       ברגע שמתחילים לבצע את ה task יש לספק את המידעים השונים על תהליך הרצתו:
* אפשרו למשתמש לקבל חיווי חי עבור סטטוס ההרצה של כל target:

FROZEN | SKIPPED | WAITING | IN PROCESS | FINISHED

(אני ממליץ לעבוד עם סדרה של רשימות/טבלאות שה target'ים נעים בינהם בהתאם למצבם)

כמו כן עבור FINISHED, אפשרו למשתמש להבין – האם הסתיימו בהצלחה, כשלון או עם אזהרות.

(אפשר למשל באמצעות צבע שונה לכל אחד מהם, או עמודה נוספת שמציגה אם הצלחה/אזהרות/כשלון)

* אפשרו למשתמש לראות progress bar המתקדם בהתאם לכמות ה target'ים שעיבודם הסתיים (כך או אחרת) ו/או נמצאים במצב של skipped.
* אפשרו למשתמש לבחור target במהלך הריצה, (מאחת מהקבוצות מעלה) ולקבל עליה את המידע הרלבנטי:
* שם
* סטטוס (Leaf | Middle | Root | Independent)
* אם הוא משתייך ל serial set(s) – את שם (שמות) הסטים.
* אם הוא frozen – ספקו את רשימת התלויות שהוא ממתין לסיומן (המוצלח) כדי לאפשר לו להיפתח לביצוע.
* אם הוא waiting - האם/כמה זמן (ms) הוא כבר ממתין בתור
* אם הוא Skipped – ספקו את רשימת התלויות שעיבודן נכשל ובגללן הוא skipped.
* אם הוא in process – ספקו מידע לגבי כמה זמן כבר נמשך העיבוד
* אם הוא Finished – ספקו מידע לגבי תוצאת הרצתו.
* התייחסות ל Serial Sets:  
  במידה ויש להתחיל ביצוע target אולם הוא משתייך ל serial set וכרגע ה serial set "סגור" להרצה (כלומר יש כרגע target מה serial set שכבר עובדים עליו) תוכלו לבחור איך לנהל את הטרד המדובר:  
  1. המתנה עד שה serial set ייפתח לביצוע.  
  2. החזרת ה target לתור ומשיכת target אחר לביצוע (אך שימו לב לפוטנציאל ה busy wait הטמון בגישה זו)

1. פאנל שליטה דינמית בהרצה:

יש להציג למשתמש את היכולת להשהות (pause) את ההרצה. במצב זה כל ה thread'ים עוצרים את פעולתם. שימו לב כי אין הכרח כמובן לעצור ולחדול את הטרד מיידית, ואם הוא כרגע מבצע משימה כזו או אחרת זה בסדר שהוא ייעצור אחריה ופשוט לא ימשיך למטרה הבאה בתור.   
בזמן העצירה המשתמש יכול לנוע בין הרשימות השונות ולבדוק את הנתונים השונים של המע'.   
יש להציג למשתמש את היכולת להמשיך את ההרצה (resume). כל ה thread'ים ממשיכים את פועלם מהנקודה שבה עצרו.

1. בסיום ההרצה יש להנגיש את תוצאותיה בצורה מסודרת:   
   האם נגמר בהצלחה או כישלון ; כמויות ופירוט של targets ב skipped | finished successfully | finished warning | finished with failure
2. 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 לחימום ליבו של הבודק ו/או להעלאת נקודות).

## חלוקה למודולים

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

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

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

## איך מתחילים ? (המלצה...)

התחילו בהגיית המראה הכללי של המע'. ואז המשיכו לייצר את המע' ב java fx על פי הגדרתה הראשונית.

התחילו בהוספת כפתור של טעינת הקובץ. יצרו את המצב שבו אתם מצליחים לטעון את הקובץ בהצלחה.   
עכשיו כשהמידעים כבר בפנים, התחילו לעבור על תצוגת המידעים השונים.

התחילו בהגיית איך כל רכיב נראה (רכיב המציג את רשימת ה targets ; רכיב המציג את שלב ההרצה ; רכיב המציג את פנל השליטה על מהלך ההרצה וכו'), ואז צרו את ה FXML, css, controller של הרכיבים הללו בהתאם לדרישות השונות.   
עכשיו שלבו מופעים של הרכיבים האלה באפליקציה הראשית שלכם. ממליץ להתחיל ראשית בהצגת הגרף וביצוע כל הפעולות עליו. רק אחרי כן המשיכו בכל הקשור לסוגיית ההרצה. גם כאן, התחילו ביכולת להריץ רק את משימת הסימולציה (כפי שהיה בתרגיל 1) ורק בסוף התקדמו להתעסק עם משימת הקומפליציה.

## בונוסים

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | פשוט מגניב ! | אנימציות  ממשו לפחות 2 סוגי אנימציות לבחירתכם, כפי שנלמד בכיתה. תעדו בקובץ ה README איזה אנימציות מימשתן, והיכן צפוי המשתמש להיתקל בהן.  האנימציה צריכה להתבצע לכל היותר במשך 2 שניות  יש לאפשר גם לנטרל את האנימציות הנ"ל, כך שהיא לא תאיט את התקדמות המערכת | כי זה מה-זה קל... | 5 נקודות  (ועד ל 100) |
| 2 | הגדלת ראש נאה | החלפת skin למע'  בבונוס זה תממשו (לפחות) 2 ערכות צבעים נוספות על הערכה הדיפולטית, ותאפשרו למשתמש להחליף את ערכות הצבעים.  אני לא מגדיר במדויק מה אמור להשתנות במסגרת המעבר מ skin ל skin – אבל דאגו לבצע שינוי מהותי שלא יוכל לחמוק מעינו של המשתמש הסביר (הכי טוב: ביחנו זאת על הוריכם) | כי זה בכל זאת תרגיל בממשק גרפי – ומה יותר גרפי מזה ??? | **5 נקודות**  **(ועד ל 100)** |
| 3 | מכובד. בהחלט. | הצגת גרף ה targets בצורה ויזואלית.  יש לעבוד עם [GraphViz](https://graphviz.org/) וללמוד לבד כיצד היא מתופעלת.  יש להציג למשתמש כפתור ייעודי שבו הוא יבקש לראות תצוגה ויזואלית לגרף ה targets.יש לאפשר למשתמש לבחור נתיב מלא ושם קובץ בו יישמרו התצוגה הויזואלית לקובץ תמונה בפורמט כזה או אחר. ליד קובץ התמונה יישמר תחת אותו שם עם סימות viz גם קובץ הטקסט שממנו נבנה הגרף. | כך או אחרת נתקלים מתישהוא בחיים בצורך לצייר גרפים ואז graph viz עולה כאופציה (חינמית) יעילה ביותר... שווה להכיר כבר מעכשיו | 10 נקודות **(מעל ל 100)** |
| 4 | די מתבקש כדי לקבל את המירב מן המע' | הצגת מידע על ניצול משאבים מקביליים.  בכל פעם שתור ה targets המוכנים לביצוע משתנה (target מתווסף או מוסר( יש לחשב את כמות הטרדים הפנויים.  יש להוסיף את הפלט הנ"ל במהלך ההרצה (רצוי בטאב משלו).  הפלט יהיה טבלאי:  כל שורה היא שינוי במצב התור (target נכנס או יוצא)  העמודות יכללו:   * חותמת זמן אבסלוטית: HH:MM:SS * זמן מתחילת ההרצה: HH:MM:SS * כמות thread'ים פנויים מסך ה thread'ים במע'. אפשר גם להציג באחוזים. (ככל שנמוך יותר זה אומר שתמיד ה thread'ים עובדים.( * כמות targets בתור (מסמן את כמות העבודה שעוד ממתינה לביצוע) | כל הנתונים כבר בידיכם... מה הבעיה להציג אותם ?? | 5 נקודות  (ועד ל 100) |
| 5 | נו.. בכל זאת זה גרף היררכי... | הצגת גרף ה targets בצורה היררכית  יש לאפשר להציג את גרף ה targets באמצעות תצוגה היררכית של tree view. (זאת בנוסף לאפשרות של רשימה שטוחה. לא במקום !)  יש לאפשר למשתמש להחליט אם הוא רוצה לראות את העץ בכיוון של מעלה למטה (כלומר Roots או Independents בראשית והם נפתחים עד ל leafs) או מכיוון מטה למעלה (כלומר leafs בראשית (יחד עם Independents) והם נפתחים עד ל roots)  גם כאן יש להציג למשתמש עבור כל target את סטטוסו ואת שמו | היכירות עם רכיב ה TREE VIEW שכל מי שמפתח ממשק גרפי מגיע להכירו בהזדמנות כך או אחרת... | 5 נקודות (ועד ל 100) |
| 6 | קצת טריקי | עדכון כמות הטרדים תו"כ ביצוע משימה:  כשהמשתמש מבצע השהיה (pause) של המשימה, אפשרו לו להגדיל/להקטין את כמות ה thread'ים הזמינים למע', (בין 1 ועד לנתון ה max parallelism). ואז כשהוא יעשה resume המע' תפעל עם פחות/יותר טרדים על פי מה שהוגדר.  (במידה ועובדים עם thread pool – קראו את [זה](https://stackoverflow.com/questions/25022810/java-executorservice-changing-fixedsize) ! יכול להיות שקל יותר למימוש עם עובדים ישירות עם blocking queue) | ידרוש מכם כניסה והבנה עמוקה יותר של משטר הטרדים, הסנכרון והתזמון שבינהם. Price less ! | **10 נקודות** (מעל ל 100) |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

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

בונוס שימומש אבל לא יתועד – לא ייבדק !

## שאלות ותשובות

|  |  |  |  |
| --- | --- | --- | --- |
| # | תאריך | שאלה | תשובה |
| 1 | 20.12.21 | איך מתנהג Incremental בכל הקשור להרצה רק על חלק מהמטרות | כאשר מריצים incremental הוא תמיד מניח מאחורי הקלעים את קיומה של הרצה קודמת שתוצאותיה ידועות ולכן ניתן להתחשב בהן. במידה ונתקלים במצב שבו המשתמש בחר לעשות incremental על סט של מטרות – יש לבצע בדיקה מאחורי הקלעים שאכן עבור כל אחת מהמטרות הנבחרות יש סטטוס הרצה קודם ואם אכן לכולן יש כזה – אז יתאפשר לבצע incremental על המטרות שנבחרו. אחרת אפשר יהיה לבצע רק from scratch. כך המשתמש יוכל לבחור את אותן המטרות ולהריץ עליהן כמה פעמים incremental אם שינה ולו מטרת אחת (הוריד הוא הוסיף) – אבוד לו והוא חייב להתחיל מ from scratch. מפורט כעת גם כחלק מסעיף 7.3 |
| 2 | 20.12.21 | האם ניתן לבחור להרצה גם מטרה/מטרות בודדות ללא הגדרת תלות בינהן ? | כן. מפורט ומתואר כחלק מסעיף 7.1 |

תרגיל 3 – מימוש **G.P.U.P**כאפליקציית WEB (40%) – הגשה: 22.2.2022

## פרטים יבשים

צפי תחילת עבודה: **16.1.22** תאריך הגשה: **22.2.22**

צפי זמן לביצוע: **5 שבועות +** ציון אפשרי מקסימלי: **112**

משקל התרגיל: 40**%** קושי: **ווא'עליה אלל'בחטי**...

**מטרות התרגיל העיקרית**

1. מימוש המערכת כאפליקציית web.
2. מימוש משתמשים שונים במע'
3. מימוש הגדרת האלגוריתם בצורה דינמית

## דרישות

1. בתרגיל זה נשדרג את G.P.U.P לעבוד בצורה מבוזרת בארכיטקטורת שרת-לקוח.  
   השרת יכיל את מנוע המע' עצמה, אולם ביצוע המשימות יתבצע ע"י הקליינטים שיתפקדו "כפועלים".  
   המע' תגדיר 2 סוגי משתמשים: מנהל (administrator) ופועל (worker).   
   המנהל יוכל להעלות קבצים לשרת המתארים מופע של גרף, ולהגדיר את העבודה שיש לבצע עבורם (task עם הפרמטרים הרלבנטים וכו').   
   הפועל ייתחבר למע' ויוכל לקבל מהשרת הוראות ביצוע של משימה על target. הפועל יבצע את המשימה ויחזיר את התשובה (ואת כלל הפלטים) חזרה אל השרת שבעצם ירכז את כלל ביצוע המשימה על הגרף (השרת לא מבצע בעצמו את המשימות – אלא רק מעביר אותם לפועלים לביצוע).   
   הפועל יהיה זכאי לתשלום עבור פעולתו, כפי שיוגדר במחירון המשימות (חלק מקובץ ה XML).
2. הן הפועל והן המנהל ("הקליינטים" של השרת) יוצגו בדמות אפליקציית java fx עצמאית (אפלקיציה לכל אחד מהם). התקשורת בינן לבין השרת תתבצע באמצעות httpclient בדומה לדוגמא המסכמת.   
   ברגע נתון יכולים להתחבר אחד או יותר אדמינים כמו גם אחד או יותר פועלים.  
   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. זמן סביר ל pull הוא סדר גודל של חצי שנייה.

Admin  
(Client)

Worker  
(Client)

G.P.U.P  
Server

Admin  
(Client)

**HTTP**

**HTTP**

Admin  
(Client)

**...**

Worker  
(Client)

1. אין צורך לבצע שמירה של נתונים מעבר למופע הנוכחי של השרת. במילים אחרות כשהשרת יורד - כל הנתונים נעלמים.

לא שומרים את המשתמשים שנרשמו, הבעיות שהועלו, הפתרונות שנמצאו וכו'.

1. בתרגיל זה אין צורך לתמוך בפיצ'ר של serial set.
2. במסגרת הפעלת המע' תוקצה לכם על השרת תקייה אחת ויחידה **c:\gpup-working-dir** שבתוכה תוכלו לנהל את כלל הפעולות של המע' (כפי שהיה בתרגיל 1 ו 2).

עליכם לוודא את קיום התקייה בהעלאת השרת ולייצר אותה במידה והיא לא קיימת (היא לא תהיה קיימת !)  
**שימו לב**: המידע על ספריית העבודה כבר לא מופיע ב XML כפי שהיה עד כה בתרגיל 1 ו 2.

## אפליקציית מנהל (Admin)

האפליקציה תיפתח **במסך לוגין** שבו יידרש המשתמש לבחור לעצמו שם.   
אין לאפשר רישום כפול תחת שם שכבר קיים במע'.

לאחר מסך הלוגין המשתמש יעבור למסך הבא – **מסך ה dashboard.**   
במסך זה מוצגים כלל הגרפים שהועלו למע' כמו גם ירוכזו המשימות-לביצוע.

יש להציג את רשימת כל המשתמשים במע'. עבור כל משתמש יש להציג את שמו ואת תפקידו.   
(קלאסי ל all the information all the time)

פעולות שהמשתמש (admin) יוכל לבצע:

* + - העלאת קובץ לשרת

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

בהגיעו לשרת, הקובץ יעבור בדיקת תקינות (כל הבדיקות הרלבנטיות כפי שהוגדרו בתרגיל 1 ו 2).

במידה והקובץ נמצא תקין, הודעה מתאימה תוחזר למשתמש ומעכשיו כל האדמינים במע' יכולים לראות את נתוני הגרף.

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

משתמש יכול להעלות יותר מקובץ אחד.

עבור כל קובץ ("גרף") שהועלה יש להציג ("נתוני הגרף"):

* 1. שם הגרף
  2. שם המשתמש שהעלה את הגרף
  3. כמות ה targets הכללית ; כמות ה targets בחלוקה ל leaf | middle | root | Independents
  4. מחירון המשימות של הגרף
     + המשתמש יוכל לבחור (כל) גרף ולקבל עליו את המידעים והפעולות כפי שהוגדרו בתרגיל 2 סעיף 3,4,5 (זה יכול ואמור להיות, פיזית, אותו UI בדיוק !).
     + משימה-לביצוע:

משימה-לביצוע (execution) היא יישות המתארת הפעלה של משימה על (תת) גרף המוגדר במע'.  
המשתמש יוכל לבחור את אחד מהגרפים שמתקיימים במע' (כל אחד מהם ; לאו דווקא את זה שהוא העלה) ולהגדיר משימה שצריך לבצע עליו. יש לתת למשימה-לביצוע שם. השם הוא ייחודי ויש לוודא כי לא מנסים להכניס משימה בעלת שם קיים.  
יש לאפשר למשתמש לבחור להפעיל אך ורק את המשימות הנקובות במחירון (כפי שהוגדר ב XML)

על המשתמש לבחור את המשימה ולמלא את הפרטים שלה כמו גם את בחירת המטרות שישתתפו בריצה.   
הגדרת משימה חדשה לביצוע היא כמובן במצב של from scratch (ואין צורך לאפשר למשתמש לבחור עבורה מצב incremental)

לכשיסיים, המשימה-לביצוע תיחשף תיווצר בשרת ותיחשף לכלל המשתמשים.   
כשהמשימה רק נוצרת היא בסטטוס "חדשה" (בהמשך הסטטוס שלה יכול להשתנות בהתאם להתקדמותה. מפורט בהמשך).

עבור כל משימה-לביצוע יש להציג:

* 1. שם המשימה
  2. שם המשתמש שמייצר אותה
  3. שם הגרף שנכלל במשימה
  4. כמות ה targets הכללית ; כמות ה targets בחלוקה ל leaf | middle | root | Independents
  5. מחיר המשימה הכולל (מחיר העבודה עבור כל target כפול סך המטרות שנבחרו לביצוע)
  6. כמות הפועלים הרשומים לה. בהתחלה 0.  
     ראו תיאור באפליקציית הפועל כדי להבין את תהליך הרישום של פועל למשימה.
  7. סטטוס המשימה.

יש להציג את המידע הנ"ל עבור כלל המשימות-לביצוע שהוגדרו במע' (גם אלה שהוגדרו ונשמרו ע"י מנהלים אחרים).   
כמו כן יש להציג גם משימות שפעולתם נסתיימה כבר.

* הרצה חוזרת / Incremental:   
  האדמין יוכל לבחור לקחת משימה-לביצוע קיימת, כזו שכבר הסתיימה (אם בצורה מלאה או אם בצורה חלקית - כזו שלא כל המטרות בה בוצעו בהצלחה) ולבחור להפעיל אותה מחדש – אם מחדש לחלוטין (from scratch) או אם בתצורה של Incremental.   
  בעשותו כן הוא למעשה משכפל את המשימה-לביצוע למשימה-לביצוע חדשה שתתבצע בהתאם לאופי הנבחר:  
  - from scratch – כל המטרות מאופסות וההרצה מתחילה מחדש לחלוטין (כאילו היה מייצר בעצמו משימה-לביצוע חדשה ובוחר את אותן המטרות ואותו ה task עם אותם הפרמטרים בשנית)  
  - Incremental – המטרות וסטטוסן הקודם "מועתקות" למשימה-לביצוע החדשה והיא מתבצעת (בצורה Incremental) מהפוזיציה הזו (כמובן עם אותו ה task וההפרמטרים).

השם של המשימה "המועתקת" יכלול את השם המקורי של המשימה בתוספת ספרור אוטומטי (1, 2, 3 וכן הלאה).

כמובן שביצירת משימה-לביצוע "מועתקת" אין צורך להטריד את המשתמש בשנית בהכנסת נתוני ה task להרצה והפרמטרים שלו וכו'. (אחרי הכל אנחנו באים לעזור לו לעשות re-run...)

רשימות אלה (של המשתמשים, הגרפים האפשריים והמשימות לביצוע המתקיימות במע') מתעדכנות ב live ומציגות את מצב המע' הנוכחי כפי שהוא בכל רגע נתון (יש לעבוד באחת משיטות ה Pull שלמדנו. המלצתי: all)

המשתמש יוכל לבחור את אחת מהמשימות-לביצוע שהוא הגדיר ולקבל מידע מגוון עליה במסך הבא: **מסך פירוט משימה-לביצוע**:

במסך זה ה admin יוכל לעבוד ולקבל מידע במסגרת משימה אחת לביצוע.

מרגע שנכנסים למסך, האדמין יוכל לנהל את הפעלת המשימה (להפעיל, להשהות, להמשיך, לעצור).   
ברגע שהמשימה-לביצוע מופעלת, כל ה target'ים המוכנים לפעולה נכנסים לתור, אולם בניגוד לתרגיל 1 ו 2 – הם לא מתבצעים ע"י השרת עצמו, כי אם אך ורק ע"י ה"פועלים".   
האדמין הוא זה שמקבל את מלוא מידע ההפעלה והביצוע של המשימה (כמו שהיה בתרגיל 2) – איזו target נמצאת באיזה שלב, מי מבצע אותה, וכו'

האדמין יראה מידע לגבי המשימה-לביצוע:

* 1. שם המשימה
  2. שם גרף ה target'ים
  3. כמות ה targets הכללית ; כמות ה targets בחלוקה ל leaf | middle | root | Independents
  4. סה"כ כמות פועלים רשומה
  5. סה"כ targets שמבוצעים כרגע (ע"י הפועלים) יחסית ל targets שיכולים להיות מבוצעים (כלומר כמה ממתינים בתור).
  6. מדד התקדמת בביצוע המשימה כפונצקיה של קצב עיבוד ה targets בתצורה כזו או אחרת  
     (יכול להיות נתון מספרי או Progress bar – כרצונכם)

אפשרו למשתמש פאנל שליטה שיאפשר לו להפעיל (play), להשהות (pause), להמשיך (resume) ולעצור (stop) את תפעול המשימה:

* לחיצה על play תסמן את המשימה-לביצוע ככזו שהופעלה ורק מכאן ואלך אפשר להפיץ את הטרגטים שלה לפועלים (יתואר בהמשך). סטטוס המשימה משתנה ל"פעילה".
* כאשר המשתמש יבחר לעשות pause, הוא פשוט לא יקצה יותר target'ים לפועלים. סטטוס המשימה ישתנה ל"מושהית"  
  **שימו לב:** ה target'ים שכבר מתבצעים ע"י הפועלים ימשיכו להתבצע (אין צורך לעצור את הפועלים) – אך הם פשוט לא יקבלו משימות חדשות. (גם אם ממתינות כאלה בתור)
* לחיצה על resume (אפשרי רק אחרי לחיצה קודמת על pause) תאפשר את המשך עבודת המע' כנדרש.   
  סטטוס המשימה ישתנה ל"פעילה"
* לחיצה על stop תחדול לחלוטין את ביצוע המשימה (ללא יכולת להמשיכה). סטטוס המשימה ישתנה ל"נעצרה באמצע".  
  גם כאן ה target'ים שכבר מתבצעים ע"י הפועלים ימשיכו להתבצע – אולם בפעם הבאה שהפועל יגיע לבקש עבודה יש צורך ליידע את הפועל כי ביצוע משימה זו הופסק ונחדל מבעוד מועד.

המשתמש יוכל לקבל חיווי מלא לגבי מצבה של כל target במהלך הביצוע של המשימה (כל המידעים הרלבנטים, כפי שהוגדרו בסעיף 7.4 מתרגיל 2).

כשביצוע משימה נגמר (בין אם בהצלחה מלאה או חלקית) סטטוסה משתנה ל"נסתיימה".  
בסיום ביצוע משימה הציגו את המידעים הרלבנטים כפי שהיה נהוג בתרגיל 2.   
(יש לשאוף להשתמש ברכיבים ובמסכים שכבר יצרתם בתרגיל 2 לשם כך. אין סיבה לעבוד כפול ולייצר עבודה מיותרת !)

המשתמש יקבל כפתור שיאפשר לו לחזור חזרה למסך 2 (dashboards).

**שימו לב:** ביצוע וניהול המשימה לא נעצר בשל מעבר חזרה למסך הקודם.

## אפליקציית פועל (Worker)

האפליקציה תיפתח במסך לוגין שבו יידרש המשתמש לבחור לעצמו שם. (אין לאפשר רישום כפול תחת שם שכבר קיים במע'.)  
כמו כן יש לספק את מספר המשאבים המקביליים (טכנית, טרדים) שהוא מוכן להקצות ולתרום. מספר שלם בין 1 ל 5.

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

במסך ה dashboard של הפועל יוצגו 2 רשימות מרכזיות:

1. רשימת כל המשתמשים במע': עבור כל משתמש יש להציג את שמו ואת תפקידו.
2. רשימת המשימות-לביצוע המוגדרות במע'.

המשתמש רואה את כל המשימות-לביצוע הקיימות במע' – של כלל המנהלים שיצרו אותם.

על כל משימה-לביצוע יש להציג:

1. שם המשימה-לביצוע
2. שם המשתמש שיצר אותה
3. סוג ה task לביצוע במשימה
4. כמות ה targets הכללית ; כמות ה targets בחלוקה ל leaf | middle | root | Independents
5. מחיר המשימה עבור כל target
6. סטטוס המשימה:
   1. חדשה: האדמין רק יצר אותה
   2. מופעלת: האדמין לחץ play
   3. מושהית: האדמין לחץ pause
   4. נעצרה-באמצע: האדמין לחץ stop
   5. נסתיימה: ביצוע המשימה נגמר בצורה רגילה בין אם בהצלחה חלקית או מלאה, ללא התערבות חיצונית ע"י האדמין
7. כמות הפועלים הרשומים למשימה
8. חיווי האם הוא עצמו רשום לביצוע המשימה הנ"ל או לא.

רשימות אלה מתעדכנות ב live ומציגות את מצב המע' הנוכחי כפי שהוא בכל רגע נתון   
(יש לעבוד באחת משיטות ה Pull שלמדנו. המלצתי: all)

בנוסף יש להציג למשתמש זה את סך הקרדיטים שצבר בגין ביצוע משימות. מספר זה צריך להתעדכן בצורה רציפה ולשקף את הנתון האמיתי.

**אופן ביצוע המשימות אצל הפועל:**

הפועל יוכל לבחור את אחת מהמשימות-לביצוע ולהירשם כפועל זמין לעבודה עבורה.   
הפועל יכול להירשם רק למשימה שהיא בסטטוס חדשה, מופעלת או מושהית (כלומר לא יכול להירשם למשימות שבפועל כבר הסתיימו, בין אם הסתיימה בהצלחה או בין אם נעצרה באמצע ע"י האדמין).

פועל יכול להירשם ליותר ממשימה-לביצוע אחת, ואף ליותר משימות מכמות הטרדים שיש לו.  
מרגע שהוא רשום למשימה הוא יקבל מידעים קונקרטים על הפרמטרים של המשימה (ה task) עצמה.

**שימו לב**: פעולת הרישום היא עצמאית והיא לא כוללת את קבלת העבודות לביצוע מהשרת. אין שום סיבה שרישום ייכשל או יידחה ע"י השרת.

כל פועל יתחבר לשרת ויבקש לקבל target'ים לביצוע בהתאם לבקשת הרישום שלו למשימות-לביצוע.   
השרת ישלח לו בתשובה את ה target'ים הנדרשים לביצוע.   
הפועל יבצע את המשימה (שאת פרטיה הוא כבר מכיר ויודע עוד ממעמד הרישום) באמצעות מי מה thread'ים שהוא הקצה לכך (המספר ניתן כחלק ממסך הלוגין).   
בסיום עיבוד target הוא יעדכן חזרה את השרת בתוצאות העיבוד (איך היא הסתיימה, לוגים וכו').   
השרת יקבל את הלוגים ויישמור אותם כקבצים בספריית העבודה המתאימה לביצוע המדובר. (כפי שהיה בתרגיל 2)  
בסיום ביצוע משימה על target מסוים הפועל מקבל קרדיט על פי "מחירון" המשימות (כפי שהוא התקבל בקובץ ה XML)

אחריות הפועל היא לנהל את משטר ה thread'ים שלו ולבקש מהשרת target'ים לביצוע עד לכמות ה thread'ים הפנויים.

פועל יכול לבחור להשתתף ביותר ממשימה-לביצוע אחת, אך סך הטרדים שלו מוקצים לסך כל המשימות שהוא מבצע.

בפנייה לשרת הפועל יצהיר כמה target'ים הוא רוצה לקבל (לכל היותר ככמות הטרדים הפנויים שלו). השרת יישלח לו חזרה target'ים לביצוע, בהתאם לרישום שלו למשימות האפשריות.   
השרת ינסה לחלק ככל האפשר לפחות target אחד מכל משימה שהוא רשום אליה.

בהחלט ייתכן כי לא יהיו משימות לביצוע עבור חלק מהרישומים.

הגו בעצמכם את מודל העבודה מול השרת (ופרטו אותו כמובן בקובץ ה readme):  
מתי יפנה הפועל לקבל עוד משימות מהשרת ? האם כל פרק זמן קבוע (אני ממליץ !) ? האם כל אימת שמתפנה טרד ? האם שילוב כלשהוא של השניים ?

האם על השרת להחזיר לפועל גם טרגטים שאין עליהם עבודה אמיתית לביצוע ? (נגיד כאלה שמי מילדיהם כבר נכשל ולכן הם כבר skipped ?) אולי ראוי לחסוך את המאמץ ובהיתקל בטרגטים כאלה כבר "לפתור" אותם בצד השרת ?

**מסך ריכוז משימות לביצוע:**

ברגע נתון הפועל יכול לבצע משימות שונות על target'ים שונים.   
יש לשקף מידע על פועלו הנוכחי הכללי של הפועל למשתמש. המידע הזה יופיע במסך הבא: **מסך ריכוז המשימות-לביצוע**

**מידעים ופעולות אפשריים למשתמש:**

* יש לשקף את כמות הטרדים העמוסים/פנויים מתוך הסה"כ בכל רגע נתון.
* מידע על כלל ה target'ים שפועל זה ביצע (בעבר וכרגע)

עבור כל target שכזה יש להציג את המידע הבא:

* + שם המשימה-לביצוע לה הוא שייך
  + שם המשימה להרצה (task: סימולציה/קומפליציה)
  + שם ה target המבוצע
  + סטטוס עיבוד ה target (עבור כאלה שמבוצעות כרגע: in processing ; עבור כאלה שהסתיימו: האם בהצלחה/כשלון וכו')
  + לוגים של ביצוע ה target המדובר (רלבנטי רק עבור target'ים המבוצעים כרגע)
  + מחיר שהתקבל עבור ביצוע המשימה (רלבנטי רק עבור target'ים שעיבודם כבר נגמר).

**שימו לב**: זהו מידע שכל פועל מחזיק על עצמו. אפשר בהחלט לנהל אותו ברמת הפועל עצמו ולא בהכרח לשאוב אותו בכל רגע מהשרת

יש להציג את סך הקרדיטים שקיבל הפועל עד עכשיו

* מידע על המשימות-לביצוע שהפועל רשום אליהן כרגע.   
  על כל משימה-לביצוע יש להציג את הפרטים הכללים הבאים:

1. שם המשימה
2. סה"כ כמות פועלים רשומה
3. מדד התקדמות בביצוע המשימה. יכול להיות נתון מספרי (x / y) או Progress bar – כרצונכם.

**שימו לב**: ההתקדמות היא של כל המשימה, כפי שהשרת רואה אותה, לא רק בכמות המשימות שקיבל פועל זה עבורה.

1. כמות target'ים שהפועל הנ"ל ביצע כבר במסגרת המשימה הזו
2. סך הקרדיט שניתן לפועל בגין ביצוע המשימה הנ"ל

* המשתמש (הפועל) יכול לבחור משימה-לביצוע מהרשימה הנ"ל ולבצע מס' פעולות בגינה:

1. להשהות (pause) את השתתפותו:  
   במצב זה הפועל לא יבקש עוד target'ים מהשרת לביצוע (עבור משימה זו).   
   אין צורך לעצור את ביצוע ה target'ים הנוכחים שהוא כבר עליהם.
2. להמשיך את השתתפותו במשימה (resume). רלבנטי כמובן רק אם עשו pause.
3. להסיר את השתתפותו מהמשימה (stop \ unregister).  
   במצב זה הפועל לא יבקש עוד target'ים מהשרת לביצוע (עבור משימה זו).   
   הפועל יורד ממניין הרשומים לביצוע עבור המשימה. (מידע שמשתקף גם אצל האדמינים השונים).  
   המידע על משימה-לביצוע זו למעשה ייעלם מהמסך שהפועל רואה. יש לבקש אישור מפורש מהמשתמש לביצוע פעולה זו.  
   אם ירצה הוא יוכל לחזור למסך ה dashboard ולבצע רישום חוזר כפועל למשימה בהמשך.

* יש לאפשר למשתמש לחזור חזרה למסך מס' 2 של ה dashboard.   
  **שימו לב:** חזרה למסך זה לא חודלת או מפריעה את המשך הביצוע של המשימות-לביצוע ברקע.

## 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 שמהם ייבנו 2 ה jar של אפליקציות ה client.

## איך מתחילים ?

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

לאחר מכן התחילו בעבודה על פי סדר העמודים:

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

תו"כ כתיבת עמוד זה תתרגלו את שלב ה UI ל שרת וחזרה ל UI.   
זה גם ידרוש מכם להקים בצד השרת אובייקט כזה או אחר שמנהל את המשתמשים וכן הלאה.

לאחר מכן התחילו לעבוד על מסך מס' 2 בגרסת האדמין. הדבר יהיה דומה יותר למה שכבר יש לכם בתרגיל 2 ולכן יהיה קל יותר. המשיכו למסך מס' 2 בגרסת הפועל. כאן כבר תתרגלו עבודה מול השרת כגורם מתווך בין 2 סוגי המשתמשים: תוכלו לתרגל מצב שבו האדמין יוצר משימה-לביצוע והפועל רואה ומקבל אותה. זה גם יהיה המקום שבו יש לכם קצת עבודת UI "חדשה".

המשיכו לביצוע מסך 3 של האדמין (ללא פעולות השליטה) – ולו רק כדי שתוכלו לייצר תוכן של משימות לפועלים.  
המשיכו לביצוע מסך 3 של הפועל (ללא פעולות השליטה) – כדי שתוכלו לנסח את החלפות המידעים בינהם ולראות שמשימה פועלת הלכה למעשה.

רק עכשיו ממשו את יכולת השליטה (pause, resume, stop) של כל אחד מהצדדים.  
לסיום, שכשל המע' כבר חיה עובדת ובועטת, התעסקו עם כל הסוגיה של הרצה חוזרת/Incremental.

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

לא מן הנמנע כי 2 סוגי הקליינטים יצטרכו להשתמש בקוד משותף (למשל כל תפעול וגישה ל http client). מאוד הגיוני לייצר מודול נוסף (== jar) שיחזיק את סך הקוד המשותף ל 2 האפלקיציות (וייכלל כמובן ב 2 האפליקציות). כמו כן כלל היכולות שהיו ארוזות במנוע האחד והיחיד בתרגיל 1 ו 2 למעשה מתפצלות כרגע בין השרת ל worker. לבחירתכם האם לעבוד עם מנוע אחד ש 2 הרכיבים הנ"ל מכירים (והוא מכיל את סך כל היכולות המשותפות לשניהם) או לפצל ל 2 מנועים קטנים, בהתאם לדרישות של כל סוג של רכיב.

ממליץ מאוד לאורך כל העבודה להקפיד לעבוד ולבחון את עצמכם דרך postman כאמצעי וידוא כי הסרבלטים עובדים כראוי ללא תלות בקליינט כזה או אחר. תוכלו לוודא כי צד השרת עובד טוב (up to a degree) טרם כתיבת החלק הרלבנטי בקליינט (וכמובן יקדם אתכם בדרך לבונוס מס' 2)

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | הגדלת ראש מגניבה ! | Chat:  אפשרו למשתמשים לקיים chat בינהם תו"כ התחברותם למע'.  ה chat צריך להתנהל כולו אל מול השרת בצורה שבה כולם רואים את מה שכולם כותבים. (מאוווווווווווד דומה לדוגמא הסופית)  הצ'ט מתבצע בין המשתמשים הרשומים והמחוברים למערכת בלבד. וצריך להיות פעיל הן בעמוד 2 והן בעמוד 3. | היש מדהים מזה ??? | 10 נקודות  (ועד ל 100) |
| 2 | מתנה, ממני אליכם | Postman Collection  עליכם לייצר Postman Collection שיאפשר למשתמש העצמאי (זה שלא רוצה לעבוד עם קליינט גרפי) לעבוד גם כן מול השרת שלכם.  על ה collection שלכם להיות מתועד היטיב בקובץ ה readme (כולל דוגמאות הפעלה) ועליו להכיל את היכולות הבאות:   * אפשרו למשתמש להירשם כ"פועל" תוך מתן שם וכמות טרדים * אפשרו למשתמש לקבל חזרה את רשימת כל המשתמשים הקיימים במע' עם תפקידיהם * אפשרו למשתמש לקבל חזרה את רשימת כל "המשימות-לביצוע" * אפשרו למשתמש להירשם כפועל עבור משימה-לביצוע. יש לוודא כי הוא מקבל חזרה פרטים כלליים על אופי המשימה * אפשרו למשתמש לבקש מהשרת משימות קונקרטיות לביצוע כחלק מהמשימה-לביצוע שהוא נרשם אליה * אפשרו למשתמש לעדכן את השרת בתוצאת עיבוד של משימה. * אפשרו למשתמש להתפטר מביצוע משימה. | כדי שתוכלו לדבר על זה בראיון עבודה, כמובן... | 7 נקודות **(מעל ל 100)** |
| 3 | הגדלת ראש סבירה | פועל נוטש את העמדה: אפשרו למע' להתמודד עם מצב שבו פועל נוטש (בצורה לא מכובדת) את עבודתו (כלומר אפקטיבית מישהו פשוט סוגר את האפליקציה של הפועל). במצב זה על השרת לזהות זאת ולהקצות את כל המשימות והעבודות שכבר ניתנו לפועל לשאר הפועלים (ככל שקיימים כאלה). על השרת להסיר את הפועל מרשימת הפועלים המוכרים (כאילו הפועל עשה Logout), ולדווח על כך לכל האדמינים במע' שהפועל היה אחראי על ביצוע משימותיהם.  חישבו על דרך מעניינת שבה השרת ידע לזהות פועל סורר. אני ממליץ על נוהל דיווח סדרתי של הפועל לשרת כל פרק זמן קבוע. כשדיווח שכזה לא מגיע אחרי פרק זמן מוגדר – זו אינדיקציה לכך שהפועל סרח... | תרחיש סביר ביותר במציאות היומיומית | 5 נקודות **(מעל ל 100)** |

## סבבה, סיימתי. מה ואיך להגיש ?

יש להגיש קובץ zip המכיל:

1. WAR אחד בדיוק שיושם בספריית tomcat ויעבור deployment אוטומטי. על WAR זה לכלול את כל התלויות שלכם,

אין להניח כי יסופקו לכם תלויות מבחוץ (למשל Gson וכו').

1. 2 ספריות עבור הקליינטים השונים: admin ו worker. בכל ספרייה יישבו כל ה jar'ים הנדרשים כדי לתפעל את הקליינט הרלבנטי. יש לספק לכל קליינט קובץ batch שמפעיל אותו (כמו שהיה בתרגיל 1 ו 2).
2. אם ביצעתם את בונוס מס' 2, יש לספק גם Postman Collection שניתן יהיה לייבא ולהפעיל אל מול השרת שלכם בכתובת localhost:8080
3. קבצי קוד המקור של הפרויקט שלכם (גם צד השרת וגם צד הלקוחות).
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 בקובץ 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 | 2.2.22 | האם שם משימה לביצוע הוא ייחודי ? האם צריך לוודא כי לא ניתן שם כפול ? | השם הוא ייחודי ובוודאי צריך לוודא כי אין כבר משימה בשם זה.  מעודכן [כאן](#EX3_Q1) |
| 2 | 2.2.22 | האם הפועל יכול להירשם למשימה בכל סטטוס | לא. הפועל יכול להירשם רק למשימה חדשה, פעילה או מושהית.  מעודכן [כאן](#EX3_Q2) |
| 3 | 2.2.22 | האם בפרטי משימה-לביצוע יש להציג את כל הפועלים שנרשמו לה או את אלה שכרגע מבצעים עבודה כלשהיא | יש להציג את סך הפועלים שנרשמו למשימה (בלי קשר להאם הם כרגע גם ממש מבצעים עבודה עבור המשימה). שימו לב כי פועל יכול להסיר רישום. במצב זה כמובן מורידים אותו מסך הרשומים למשימה. מעודכן [כאן](#EX3_Q3). |
| 4 | 2.2.22 | מהם הסטטוסים השונים שהמשימה יכולה להימצא בהם ? | כאשר האדמין רק מייצר משימה – היא מתחילה כמשימה חדשה. בשלב זה ה worker'ים יכולים להירשם אליה אולם אפקטיבית היא עדיין לא פעילה. מרגע שהאדמין מפעיל אותה (לוחץ על Play) היא הופכת לפעילה ורק אז אפשר להזרים target'ים שלה ל worker'ים (על פי דרישתם ויכולתם).  תיאור הסטטוסים השונים מפורט [כאן](#EX3_Q4) |
| 5 | 2.2.22 | האם פועל יכול להירשם למספר משימות-לביצוע ? האם יותר מכמות הטרדים שיש לו ? | כן. פועל יכול להירשם לכמה משימות-לביצוע שהוא רוצה. עצם הרישום שלו לא בהכרח אומר שהוא יקבל גם עבודה (כלומר הרצת task על מטרות) במסגרתן. מתואר [כאן](#EX3_Q5) |
| 6 | 2.2.22 | האם הפועל יכול לנהל בעצמו את המידע לגבי ה target'ים שהוא ביצע או שצריך גם אותו למשוך מהשרת ? | אמנם המידע הנ"ל יכול גם להתנהל בצד השרת – אולם כלל המידע הנדרש מופיע וקיים אצל הפועל (זה מידע של הפועל בלבד) ולכן אין שום מניעה (ואולי אף רצוי כדי להוריד עומס מהשרת) להציגו "מבטן הפועל", ולא למשוך אותו כל הזמן מהשרת.  מעודכן גם [כאן](#EX3_Q6) |

נספח א' – G.P.U.P Tasks

במסגרת התרגיל תוכלו לנהל ב GPUP מספר משימות (Task). לכל משימה יכולים להיות ארגומנטים ופרמטרים משלה.

הנה תיאור המשימות השונות שייתמכו במהלך כל התרגילים:

**Simulation Task**

משימה זו למעשה לא מבצעת משהו מעניין אלא רק מושכת זמן בביצועה על כל target כדי לסמלץ ביצוע עבודה כלשהיא.

באמצעותה ניתן יהיה להפעיל את המע' "על יבש" ולבדוק את פעולתה בהתאם. המשתמש יוכל לקבוע כמה זמן המשימה "עובדת" על כל target (בפועל מבצעת sleep) ומה יהיו הסיכויים שהיא תסתיים בהצלחה או בכישלון.

המשימה יכולה לעבוד על כל גרף של targets. אין לה שימוש או דרישה קריטית לקיום נתונים נוספים על גבי ה target.

פרמרטים חופשיים ל task

1. זמן עיבוד לכל target. מספר שלם שיינתן ב ms.
2. האם זמן העיבוד הוא רנדומלי (חסום בנתון מס' 1) או קבוע (בדיוק נתון מס' 1)
3. ההסתברות שהרצת משימה על target תסתיים ב success. מספר עשרוני בין 0 ל 1.   
   המשלים ל 1 הוא הסיכוי שעיבוד המשימה יסתיים ב failure
4. אם משימה נגמרת ב success, מה ההסתברות שזה יהיה success with warnings ? מספר עשרוני בין 0 ל 1.

מידעים שיש להדפיס במהלך ריצת task על כל target:

* + - מידע על הזמן שה task הולך "לישון"
    - שורת מידע לפני "השינה".
    - שורת מידע אחרי "השינה"

**Java Compiler Task**

משימה זו מפעילה את הקומפיילר של ג'אווה (javac) על קבצי קוד מקור ומקמפלת קובץ בודד ל .class המתאים לו.

גרף ה targets יתאר למעשה את הקשרים בין קבצי java השונים בתוך פרוייקט מסויים, כך שראשית יקומפלו המחלקות שלא דורשות אף תלות (זולת ה JDK) ולאחריהן המחלקות הנסמכות עליהן וכן הלאה.

לשם הפעלת javac תצטרכו לספק 3 ארגומנטים:

1. -d <target folder> : מכיל הפניה לספריה שאליה יווצרו תוצרי הקימפול. הספרייה חייבת להיות קיימת מבעוד מועד.
2. -cp <target folder> : מכיל הפניה לספריה שבה נמצאים תוצרי הקימפול אותם יחפש הקומפיילר כשייתקל בתלויות של קובץ בעת שהוא מקמפלו.
3. <path to java file> : הפנייה לקובץ ה java המלא אותו הקומפיילר צריך לקמפל. ההפניה צריכה להכיל את הסיומת .java

הנה דוגמא לשורת הפעלה, המקמפלת את הקובץ file.java הנמצא תחת הספרייה path/to/java אל תוך ספריית out ומורה לקומפיילר לחפש מחלקות מקומפלות גם כן בספריית out (יחסית למיקום ב java-project)

..\java-project> **javac -d out -cp out path/to/java/file.java**

Javac הוא process חיצוני. כדי להפעיל process חיצוני באמצעות קוד בג'אווה בידקו את המחלקה [ProcessBuilder](https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/ProcessBuilder.html) ולימדו כיצד לעבוד איתה.  
(הנה עוד [מדריך](https://www.baeldung.com/java-lang-processbuilder-api). ממליץ לבצע כמה נסיונות בתוכנית עזר קטנה בצד)

מבנה ה target המתאים ל task זה:

* GPUP-User-Data יכיל את ה FQN של המחלקה (ה package כולל שם המחלקה).
* עליכם להמיר את ה FQN של המחלקה לנתיב על מע' הקבצים. הנתיב שתקבלו הוא המיקום שבו יימצא קובץ ה java יחסית לנתיב הספרייה שקיבלתם בפרמטר הראשון המופיע למטה.
* ניתן להניח כי המשימה הזו תופעל **אך ורק** על גרף שמכיל את המידע הנדרש הנ"ל. אין צורך לוודא זאת.

פרמטרים ל task:

1. נתיב לספרייה הראשית שבמסגרתה (תחתיה) נמצא כל קוד המקור לקימפול. לדוגמא: ספריית src  
   הספרייה מצופה להיות קיימת. עליכם לבדוק זאת ולהתריע במידה ולא.
2. נתיב לספריה בה צריכים לייצר את הקבצים המקומפלים.   
   הספרייה לא חייבת להיות קיימת. במידה והיא לא קיימת - יש לייצרה.

מידעים שיש להדפיס במהלך ריצת task על כל target:

* + - מידע על זהות הקובץ המקומפל.
    - מידע על שורת ההפעלה המלאה של הקומפיילר שה task הולך לבצע
    - אם התהליך – נכשל יש להדפיס את כל הפלט של javac
    - מידע על כמות הזמן שלקח לקומפיילר לעבוד (בין אם סיים בהצלחה או בכשלון) את הזמן יש להציג ב ms (מילי סקונד)

נספח ב' – אפילוג (סוף הסיפור על G.P.U.P)

המע' נוסתה והוטמעה בהצלחה בתוך applied וחוללה הבדל מהותי בתהליך ובשיטות הפיתוח:

1. ביצוע checkout של כל הפרוייקטים (עבדנו מול clear case) לקח במקור כ 11 דקות.  
   עם G.P.U.P זה לקח דקה וחצי.
2. ביצוע undo checkout (עוד פעולה רווחת) – לקח במקור 20 דקות.  
   עם G.P.U.P זה לקח עשר דקות
3. בילד מלא של כל המע' לקח במקור 11 שעות (== לילה)

עם G.P.U.P זה לקח בממוצע 4.5 שעות (עם שיא של קצת פחות מ 4 שעות לפעמים). הדבר הוביל לשינוי דרך העבודה: מפתח יכול היה להגיע בבוקר ולבצע בילד מלא ועד הצהריים כבר היו לו תוצאות. אם נכשל יכול היה לתקן את התקלות ולהריץ שוב בצורה Incremental ואז עד אחה"צ כבר הייתה לו סביבה עובדת, מה שזירז מהותית את קצב הפיתוח.

1. ביצוע טסטים למע'.   
   לפני כן מבוצע היה בתהליך ידני על ידי מעטים שידעו כיצד.

עם G.P.U.P קל היה להגדיר, לראשונה, את גרף התלויות בין הטסטים השונים ולאפשר לכולם להריץ אותם בקלות כחלק מתהליך הפיתוח. הרצת הטסטים (~ 500 מקרים) באמצעות 4 טרדים על מכונה ייעודית לקחה סדר גודל של 10 דקות.  
גם זה שינה מהותית את תהליך הפיתוח: היות וניתן היה בקלות לבצע הרצות טסטים – כל קוד חדש של המפתח, כל תיקון באג, כל פיצ'ר חדש – עבר דרך הרצת הטסטים ובכך קיבל כל המוצר תו איכות חזק הרבה יותר. כמו כן ככל שעבר הזמן קל היה לעבות ולהרחיב את מערך הטסטים שיכלול פיסות קוד נוספות.

1. בזכות G.P.U.P ניתן היה ביתר קלות לחשוב ולצאת לפרוייקטים של עדכון, פירוק, וניתוק תלויות. קל היה לדעת איזה פרוייקט תלוי במי, כמה ולמה ; מה עומק התלויות וכמה הן מורכבות ; ובהתאם לזה לצאת למהלך של פירוק כמה מה battle-necks שאפשרו, בתורם, יצירת תהליך בילד מהיר עוד יותר...

הכל היה נראה מעולה, אולם סוף דבר היה לא כל כך חיובי (עבור G.P.U.P לפחות).

את המע' פיתחתי (במצטבר במשך שנה בערך) לבדי לחלוטין על מחשבי הפרטי בזמני הפרטי.

לא ביקשתי או דרשתי מ applied שום תשלום נוסף על עצם השימוש בה (אם כי היה נחמד לקבל תגמול כספי כזה או אחר בגינה). הדבר היחיד שביקשתי הוא שהמע' תישאר ותחשב "שלי". אולם, לפחות אז, היו נהוגים חוקי "זכויות יוצרים" די דרקוניים המגדירים כי **כל** פועלו של עובד החברה נחשב כרכוש החברה ללא קשר למקום/זמן/טכנולוגיה/זיקה שבה הוא משתמש.

לא שהיו לי תכנונים כבירים בהקשר זה, אולם לא רציתי לחסום את דרכי פעולותי העתידיות במידה וארצה להתקדם עם G.P.U.P בצורה עצמאית (בעיקר חשבתי להופכה לפרוייקט קוד פתוח בבוא מן הימים).   
על הרקע הזה יחסי עם applied materials עלו על שרטון. עו"ד נלקחו מטעם 2 הצדדים (עלה לי 600 ש"ח רק פגישת ייעוץ בנושא הגנת פטנטים !) ולבסוף הוחלט לחתום על הסכם המתיר לי להשתמש, להפעיל ולהפיץ את G.P.U.P כל עוד לא מדובר על תחום בעל זיקה לתחומי פעולתה של applied (ייצור מכונות לתהליך יצירת צ'יפים – אל דאגה בכיוון הזה).

למיטב ידיעתי עדיין משתמשים ב G.P.U.P בחטיבה בה עבדתי.  
לפני כמה שנים התקשרה אלי ידידה משם וסיפרה לי זאת והייתה צריכה עזרה קלה בתפעול המע'...

לאחר applied עברתי לעבוד ב HP במוצר תוכנה מבוסס ג'אווה (סוף סוף !) השנה כבר הייתה 2011 והטכנולוגיות השונות כבר הציעו פתרונות דומים/משלימים למהפך ש G.P.U.P עשתה ב applied. נסיון להטמיע את G.P.U.P גם ב HP (בעולם הג'אווה) הסתבר כיעיל במידה פחותה ובתכלס כאן נגמר הפרוייקט (הוציאו לי את כל הרוח מהמפרשים... ☹)

אגב, היום חוקי זכויות יוצרים בתחום התוכנה קצת "מקלים" יותר לכיוון העובד (בעיקר עקב התפתחות האפיק של קוד פתוח ותרומה לו) ובהחלט יש הכרה כי העובד (המתכנת) רשאי להיות שותף בפרוייקטים פיתוחיים כאלה או אחרים (עם מגבלות שונות).  
עדיין, אם/כשתחתמו על חוזה עם מעסיק תוכנה, גדול כקטן, קחו סוגיה זו בחשבון (גם אם כרגע אין לכם שום תכנונים קונקרטים בכיוון) וודאו ודעו את המצב החוזי והמשפטי של הסוגייה הזו.

נספח ג' – דוגמאות לפעולות על גרף ה target'ים

**Root**

**Middle**

**Leaf**

**Top-Down  
Depends On**

**Bottom-Up  
Required For**

**Independent**

**מטרות ומעגלים:**

**מטרה: A**

מעגל: אין

**מטרה: L**

מעגל: יש (E,L,G)

**מסלולים בין מטרות:**

**מקור: A יעד: K יחס: Depends On**

מסלול: A->E->K

**מקור: I יעד: B יחס: Required For**

מסלול: I->J->D->B

מסלול: I->B

**מקור: B יעד: E יחס: Required For**

אין מסלול

**מקור: K יעד: A יחס: Depends On**

אין מסלול

**מקור: E יעד: G יחס: Depends On**

מסלול: E->L->G

נספח ד' – תיאור מבנה המערכת באמצעות קובץ XML

תיאור מבנה המערכת מפורט כחלק מקובץ ה XML

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

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

היכן שמצויין case insensitive הכוונה היא שאין חשיבות ל case של האותיות באנגלית. במקרה זה הערך milk זהה לערך MiLk

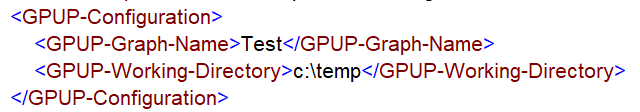
היכן שמצויין שהמחרוזת יכולה להכיל רווחים – המדובר הוא רק על רווחים בתוך המחרוזת. אם מופיעים רווחים בתחילתה/סופה יש להתעלם מהם (רמז המטודה trim() על המחלקה String)

## סכמת קובץ XML – גרסה I

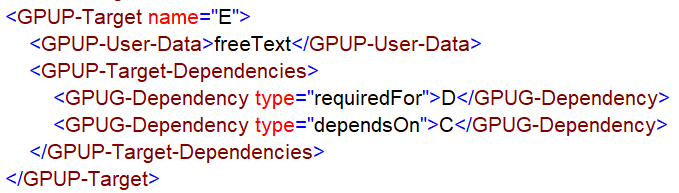
Text

Description automatically generatedראשית הקובץ מתארת מבנה של הגדרת גרף מטרות של G.P.U.P.

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | GPUP-Descriptor | מתאר את ראשית קובץ ה XML |
| 2 | Element | GPUP-Configuration | מבנה המתאר בפנים מאפיינים כלליים של המע' |
| 3 | Element | GPUP-Targets | מבנה המתאר בפנים את גרף המטרות והקשרים בינהם |

**GPUP-Configuration**

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | GPUP-Graph-Name | מכיל את השם של גרף המטרות. השם הוא ייחודי והמע' לא יכולה להחזיק יותר מגרף אחד באותו השם. יכול להכיל רווחים. case insensitive. |
| 2 | Element | GPUP-Working-Directory | מכיל נתיב מלא (במע' windows) לספריית העבודה של gpup. אין להניח כי הספרייה קיימת. יכול להכיל רווחים. |

**GPUP-Target**

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | GPUP-Target | מתאר target אחד במע' |
| 2 | Attribute | name | מתאר שם של מטרה. השם יכול להכיל רווחים.  השם הוא case insensitive |
| 3 | Element | GPUP-User-Data | אלמנט המאפשר להכיל טקסט חופשי מכל סוג שהוא (המידע ש target "נושא" על גבו). זהו אלמנט אופציונלי ואינו חייב להופיע. |
| 4 | Element | GPUP-Target-Dependencies | אלמנט המתאר את התלויות שיש ל target זה ב target'ים אחרים המתוארים בגרף.  זהו אלמנט אופציונלי ואינו חייב להופיע. |
| 5 | Element | GPUG-Dependency | אלמנט המתאר תלות אחת של target באחרת. התוכן שלו הוא שם ה target בה תלויים |
| 6 | Attribute | type | מתאר את אופי התלות.  יחס של depdensOn – ואז נאמר כי ה target תלוי ב target מסויים.  יחס של requiredFor – ואז נאמר כי ה target נדרש עבור target מסויים.  יכיל בדיוק אחד מ 2 הערכים: dependsOn | requiredFor  אין מניעה ש 2 target'ים יתארו שניהם את התלויות בינהם ביחסים הופכיים:  **X** dependsOn **Y** ; **Y** requiredFor **X** |

## סכמת קובץ XML – גרסה II

עדכונים מגרסה קודמת:

1. האלמנט GPUP-Configuration מכיל אלמנט נוסף שנקרא GPUP-Max-Parallelism המתאר את כמות המשאבים המקביליים (הטרדים) שהמע' עובדת איתם. זוהי כמות מקסימלית. מספר שלם גדול מ 1.
2. Logo

   Description automatically generated with medium confidenceהגדרת serial-sets

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | GPUP-Serial-Sets | מתאר את הגדרת כל הקבוצות הסדרתיות במע'. זהו אלמנט אופציונלי שאינו חייב להופיע. |
| 2 | Element | GPUP-Serial-set | מתאר קבוצה סדרתית אחת |
| 3 | Attribute | Name | שם של ה serial set.  מחרוזת תווים. יכולה לכלול רווחים |
| 4 | Attribute | targets | מתאר את רשימת ה target'ים הנכללים בקבוצה. הרשימה מופרדת בתו פסיק (,) הרשימה לא תהיה רשימה ריקה ותכיל לפחות target אחד. גם כאן ההכוונה ל target'ים היא case insensitive. |

## סכמת קובץ XML – גרסה III

עדכונים מגרסה קודמת:

1. האלמנט GPUP-Configuration אינו מכיל יותר את GPUP-Max-Parallelism ו GPUP-Working-Directory
2. Graphical user interface, text

   Description automatically generatedהאלמנט GPUP-Configuration מכיל אלמנט נוסף שנקרא GPUP-Pricing המתאר את מחירון עיבוד targets עבור כל משימה המוגדרת במע':

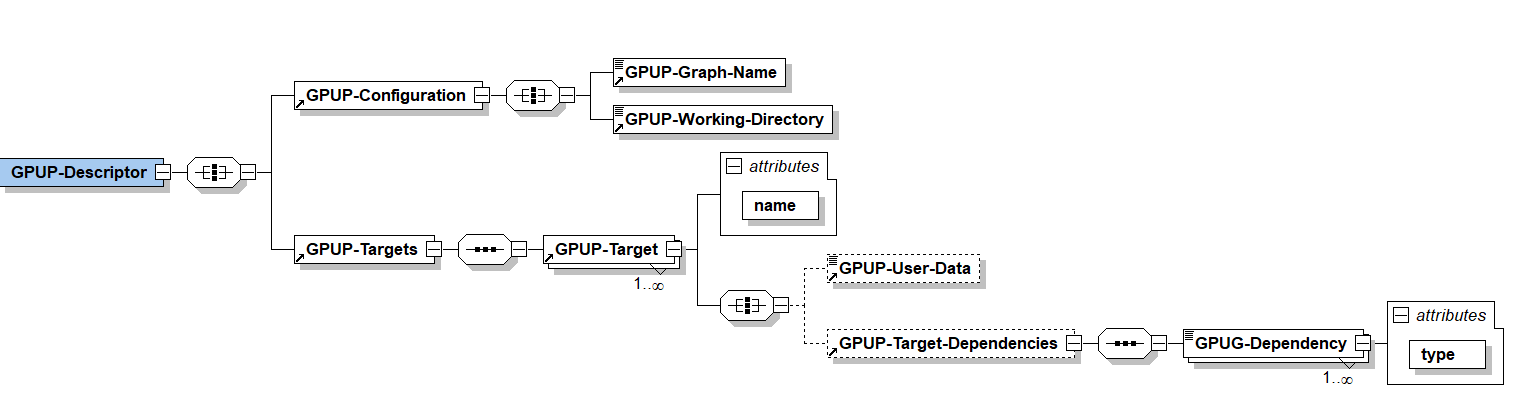
|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | GPUP-Task | מתאר מחיר של משימה אחת |
| 2 | Attribute | name | מתאר את שם המשימה.  יכיל בדיוק אחד מ 2 הערכים: Simulation | Compilation |
| 3 | Attribute | price-per-target | מתאר את מחיר עיבוד target במשימה זו.  מספר חיובי שלם |

נספח ה' – קבצי סכמה לוידוא קובץ ה XML

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

להלן תרשימים המתארים את מבנה הסכמות בגרסאותיהם השונות:

### סכמה – גרסה 1



### סכמה – גרסה 2

Graphical user interface

Description automatically generated

### סכמה – גרסה 3

Graphical user interface, diagram

Description automatically generated