מס' גרסה: **7**

שפת ג'אווה ומבוא לפיתוח ווב

Evolutionary Time Table קיץ 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 האפשריים בצורה שווה

תוכן העניינים

[דרישות הקורס 4](#_Toc75385022)

[כללי 4](#_Toc75385023)

[איך להגיש תרגילים באיחור, ולהישאר בחיים 5](#_Toc75385024)

[הנחיות כלליות לכתיבת התרגיל 6](#_Toc75385025)

[תרגיל reflection – תרגיל עצמאי ביחידים (5%) - הגשה: 18.7.2021 8](#_Toc75385026)

[כללי 8](#_Toc75385027)

[מבנה התרגיל 8](#_Toc75385028)

[ניקוד 8](#_Toc75385029)

[איך בודקים ? 9](#_Toc75385030)

[מה מגישים ? 9](#_Toc75385031)

[מקרה בדיקה לדוגמא 9](#_Toc75385032)

[Evolutianry Time Table– שיבוץ מע' שעות באמצעות אלגוריתם אבולוציוני 11](#_Toc75385033)

[מטרת התרגיל(ים) בקורס 11](#_Toc75385034)

[כללי 11](#_Toc75385035)

[אלגוריתם אבולוציוני 11](#_Toc75385036)

[מערכת שעות 12](#_Toc75385037)

[מבנה מערכת השעות 12](#_Toc75385038)

[הנחיות ספציפיות למימוש מערכת Evolutionary Time Table 13](#_Toc75385039)

[תרגיל 1 – מימוש **Evolutionary Time Table**כאפליקציית Console (25%) - הגשה: 8.8.2021 14](#_Toc75385040)

[פרטים יבשים 14](#_Toc75385041)

[דרישות 14](#_Toc75385042)

[חלוקה למודולים 18](#_Toc75385043)

[איך מתחילים ? (המלצה...) 18](#_Toc75385044)

[בונוסים 18](#_Toc75385045)

[סבבה, סיימתי. מה ואיך להגיש ? 19](#_Toc75385046)

[שאלות ותשובות 20](#_Toc75385047)

[תרגיל 2 – מימוש**Evolutionary Time Table** כאפליקציית JavaFX (30%) – הגשה: 4.9.2021 21](#_Toc75385048)

[פרטים יבשים 21](#_Toc75385049)

[דרישות 21](#_Toc75385050)

[חלוקה למודולים 24](#_Toc75385051)

[איך מתחילים ? (המלצה...) 24](#_Toc75385052)

[בונוסים 24](#_Toc75385053)

[סבבה, סיימתי. מה ואיך להגיש ? 25](#_Toc75385054)

[שאלות ותשובות 25](#_Toc75385055)

[תרגיל 3 – מימוש **Evolutionary Time Table** כאפליקציית WEB (40%) – הגשה: 9.10.2021 26](#_Toc75385056)

[פרטים יבשים 26](#_Toc75385057)

[דרישות 26](#_Toc75385058)

[חלוקה למודולים 29](#_Toc75385059)

[איך מתחילים ? 29](#_Toc75385060)

[בונוסים 30](#_Toc75385061)

[סבבה, סיימתי. מה ואיך להגיש ? 30](#_Toc75385062)

[שאלות ותשובות 30](#_Toc75385063)

[נספח א' – תיאור פעולת האלגוריתם האבולוציוני 31](#_Toc75385064)

[נספח ב' – תיאור בעיית מע' השעות בראי האלגוריתם האבולוציוני 33](#_Toc75385065)

[נספח ג' – CROSSOVER 35](#_Toc75385066)

[נספח ד' – Mutations 37](#_Toc75385067)

[נספח ה' – Selection Techniques 38](#_Toc75385068)

[נספח ו' – הגדרות החוקים 39](#_Toc75385069)

[נספח ז' – תיאור מבנה המערכת באמצעות קובץ XML 40](#_Toc75385070)

[סכמת קובץ XML – גרסה I 40](#_Toc75385071)

[סכמת קובץ XML – גרסה II 44](#_Toc75385072)

[סכמת קובץ XML – גרסה III 44](#_Toc75385073)

[נספח ח' – קבצי סכמה לוידוא קובץ ה XML 45](#_Toc75385074)

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)
* **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: 18.7.2021

## כללי

**מועד הגשה: 18.7.2021 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 3-4 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 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).
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 3 - 4 שעות עבודה ו 200 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>

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 זבל 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();  
 }  
}

Evolutianry Time Table– שיבוץ מע' שעות באמצעות אלגוריתם אבולוציוני

## מטרת התרגיל(ים) בקורס

חלק א' – מימוש מנוע האלגוריתם האבולוציוני ופתרון מע' שיבוץ בסיסית, תוך תפעול המע' כאפליקציית console.

חלק ב' – הרחבת מנוע האלגוריתם במספר פיצ'רים, תוך שכלול האילוצים על מע' השעות, ופיתוח תצוגה גרפית על בסיס אפליקציית Java FX (Desktop).

חלק ג' – מימוש אפליקציית web המציעה מע' מקוונת לשיתוף ופתרון מבוזר של מערכות שעות

## כללי

הדרישות "הכלליות" המובאות בהמשך, תקפות לתיאור המע' בכללותה, ומהוות חלק בלתי נפרד מהגדרת כל תרגיל ותרגיל.   
בכל תרגיל נממש רק (עוד) חלק מהדרישות.

בעולם מדעי המחשב ישנם דרכים וטכניקות שונות לפתור סוגים שונים של בעיות באמצעות סוגים שונים של אלגוריתמים.   
בבואנו לפתור בעיה, בדר"כ נשאף לחפש את הפתרון האידיאלי, המושלם (לפעמים גם האחד והיחיד). אכן בחלק ניכר מהבעיות הדבר אפשרי וישים (למשל מיון מערך, סריקה של גרף בצורה מסויימת וכו').   
אולם ישנם סט של בעיות "קשות" לפתרון, גם באמצעות מחשב. בעיות קשות הן בעיות שהפתרון הסטנדרטי להן ירוץ בזמן בלתי סביר (אקספונציאלי) כתלות בגודל הקלט של הבעיה וככל שזה ילך ויגדל (למשל בעיית הסוכן הנוסע ; בעית הגנב ; שח מט וכו')

עבור מכלול הבעיות הנ"ל, ישנן שיטות אחרות, שונות, היכולות לסייע במציאת פתרון. בשיטות אלה אנו מוותרים על השאיפה למצוא את הפתרון המושלם והאידיאלי לבעיה, ומוכנים "להתפשר" על פתרון שהוא "טוב מספיק" (GOOD ENOUGH) לצרכינו.

ישנם מספר רב של אלגוריתמים כאלה, אשר תכליתם "לקרב" אותנו לפתרון ייעודי.   
אחד מהם הוא אלגוריתם אבולוציוני...

## אלגוריתם אבולוציוני

אלגוריתם אבולוציוני שואב השראה מעולם הטבע, ובפרט מטכניקת האבולוציה הנהוגה בו.  
הנה דוגמא:  
חישבו על אוכלוסיית הארנבים במקום מסוים. נניח כי ניתן לאתר ולסווג כל ארנב לפי מספר מאפיניים שונים, לדוגמא: אורך רגליים, גודל אוזניים, חדות שיניים.   
המאפיינים הנ"ל של כל ארנב וארנב מסייעים לו "לשרוד" בטבע: ארנבים עם רגליים ארוכות יוכלו לרוץ מהר יותר ובהסתברות גבוהה יותר להצליח לברוח מטורף מזדמן ; ארנבים עם אוזניים גדולות יותר יכולים לזהות טורף מתקרב ממרחק רב יותר ולהגן על עצמם מבעוד מועד ; ארנבים עם שיניים חדות יותר יוכלו לאכול מזונות שונים (מעבר לחסה) וכך אולי לגלות מקורות מזון נוספים שיצילו אותם אם כל מלאי החסה שבעולם ייכחד...

דמיינו את ארנב העל: ארנב שלו יש רגליים ארוכות ביותר, שיניים חדות ביותר ואוזניים ארוכות ביותר. זהו ארנב שיכול היה לזהות את טורפיו למרחוק ולברוח מבעוד מועד ; אם בכל זאת טורף היה מגיע אליו, בהסתברות גבוהה ביותר הוא יכול היה לברוח ממנו ; אם היה בורח ממנו לאזור חדש על פני כדור הארץ שבו אין חסות – בעזרת שיניו החדות יכול היה לאתר ולאכול מזונות חדשים נוספים ולשרוד באמצעותם.  
בדוגמא זו, הבעיה אשר לה הטבע מנסה למצוא פתרון היא: "האם ניתן לייצר ולהנדס את ארנב העל הזה ?"

  
דמיינו שלב מסויים בתהפתחות אוכלוסיית הארנבים אי שם בעבר.   
נתחיל עם אוכלוסיה של מספר ארנבים (נגיד 100).   
לכל ארנב יש את המאפיינים שלו (אורך רגליים ואוזניים כמו גם חדות שיניים). מי יותר ומי פחות.   
  
הלילה יורד וארנבים עושים מה שארנבים יודעים לעשות הכי טוב.   
למחרת נולדים צאצאים חדשים לכל זוג ארנבים. כל צאצא שכזה מהווה הכלאה מסויימת של המאפיניים השונים שקיבל מהוריו.

צאצא של זוג הורים, אשר להורה ראשון יש רגליים ארוכות, ולהורה שני יש אוזניים ארוכות, יזכה, אולי, להיות ארנב עם רגליים ואוזניים ארוכות גם יחד ! (ייתכן, לא בטוח. רק ייתכן).

לעומתו, צאצא לזוג הורים בעלי רגליים ואוזניים קצרות, אך סביר הוא כי גם הצאצא יהיה כזה, ולא יהפוך להיות נפיל הארנבים...

בשעת בין ערביים להקת נמרים מגיעה לאזור אוכלוסיית הארנבים. הנמרים פורצים בשאגה מקפיאת דם וטובחים ללא רחם באוכלוסיית הארנבים התמימה.   
ארנבים בעלי אוזניים ו/או רגליים קצרות לא הבחינו במתקפה מבעוד מועד, וכשזו קרתה לא הצליחו לברוח מהר מספיק וסופם ידוע.   
לעומתם ארנבים בעלי אוזניים ו/או רגליים ארוכות הבחינו בסכנה המגיעה אליהם מבעוד מועד והתחילו במנוסתם כמה רגעים לפני שהתחילה. בנוסף הם נסו מהר הרבה יותר מכל הארנבים האיטיים האחרים וכך יצא שהם (או לפחות חלקם) צלחו את המתקפה חסרת הרחמים.

בשוך הקרבות, עם שעת ערב, הדלדלה לה אוכלוסיית הארנבים של הבוקר לכדי מחצית. רוב השורדים היו כאלה בעלי המאפיינים הטובים ביותר (רגליים/אוזניים ארוכות) ובנוסף היו גם כמה ארנבים "ממוצעים" שהתמזל מזלם ושרדו את התופת בדרך לא דרך.

הלילה יורד וארנבים עושים מה שארנבים יודעים לעשות הכי טוב... ולמחרת נולד דור צאצאים חדש.  
שימו לב: רוב ההורים של הדור החדש הם כאלה ששרדו את מתקפת הנמרים של הצהריים, כלומר כאלה בעלי מאפיינים השרדותיים טובים יותר. על כן רבים הסיכויים (לא בטוח, אבל אולי...) כי גם צאצאיהם יהיו כאלה שיהנו ממאפיינים אלה (או משילובם), מה שיעניק להם, בתורם, סיכויים רבים יותר לשרוד מתקפות נמרים עתידיות ולהיות אלה שישרדו כדי להוליד את הדור הבא וכו'..

לעומת זאת, רוב הארנבים "הפארשים", החלשים, חסרי המאפיינים ההישרדותיים – פשוט לא שרדו.   
במנגנון טבעי זה, ככל שעוברים הדורות, גדל הסיכוי שמאפיינים שאינם הישרדותיים ייעלמו מן הטבע (ייכחדו) ומאפיינים הישרדותיים ימצאו את דרכם בינות הדורות השונים ואף סביר כי ישתכללו מדור לדור.

זוהי דרכה של האבולוציה, וזוהי גם דרכו של האלגוריתם האבולוציוני.   
עיקבו על [נספח א'](#appendixA) להבנת דרך פעולתו של האלגוריתם האבולוציוני על בוריה.

## מערכת שעות

מערכת שעות היא בעיה "קשה" במדעי המחשב. כלומר, כתלות בכמות האילוצים ומבנה המע' (הקלט), לא ניתן למצוא אלגוריתם שרץ בזמן סביר (פולינומיאלי) שימצא סידור של מע' השעות כך שכל האילוצים מתקיימים (או לחילופין כזה שידע להגיד שאין סידור אפשרי שכזה).

בתרגיל זה נשתמש במנגנון האלגוריתם האבולוציוני כדי לנסות ולמצוא פתרון שהוא good enough לבעיית מע' השעות.   
שימו לב: בהחלט ייתכן כי הפתרון לא יהיה מושלם, כלומר לא כל האילוצים יתקיימו בו.

## מבנה מערכת השעות

מערכת השעות תתאר מצב טיפוסי של תיכון/בית ספר.

המע' תגדיר אסופה של מקצועות, מורים ושכבות/כיתות. כמו כן יוגדרו כמה ימי לימוד וכמה שעות לימוד יש בכל יום במהלך שבוע העבודה.

כל מורה מוסמך ללמד סט מוגדר של מקצועות

כל כיתה/שכבה מגדירה את דרישות הלימוד שלה: איזה מקצוע וכמה שעות לימוד שבועיות של כל מקצוע שכזה היא צריכה לקבל.

כמו כן המע' תגדיר רשימה של חוקים (אילוצים) שהיא צריכה לעמוד בהם (ברמה כזו או אחרת).

פתרון הבעיה יהיה אם כן מציאת שיבוץ (אחד או יותר) המקיים את כל החוקים (או מרביתם) ומציג איזה מורה מלמד איזה מקצוע באיזו השכבה ובאיזה הזמן.

את פתרון מע' השעות ניתן יהיה לצרוך ב 2 אופנים:   
מע' הלימוד מצד המורה המציגה את לו"ז המקצועות שהוא מלמד בכל שכבה בכל שעת לימוד  
מע' הלימוד מצד השכבה (כלומר התלמידים) המציגה את הלו"ז של כל שכבה ושכבה מראי התלמיד.

המע' תוגדר, אם כן, באמצעות הרכיבים הבאים:

1. מקצוע  
   כל מקצוע יכיל:

* מס' מזהה
* שם

1. מורה

כל מורה יוכל ללמד מקצועות מסויימים (לפחות אחד). כל מורה יכיל:

* + מס' מזהה
  + שם
  + רשימה של מס' מזהים של מקצועות שהוא מלמד

1. שכבה/כיתה

שכבה/כיתה מתארת קבוצת תלמידים שצריכים ללמוד מקצועות מסויימים.

כל שכבה תכיל:

* מס' מזהה
* שם השכבה
* מיפוי בין מס' מקצוע לכמות שעות שבעויות הנדרשות עבורו

1. אילוצים / חוקים

החוקים במע' השעות יגדירו את התקנים שמע' שעות תיקנית וחוקית חייבת לעמוד בהם.

החוקים ייתחלקו ל 2 משפחות:

* חוקים קשים (Hard):

חוק קשה הוא כזה שחייבים לעמוד בו אחרת מע' השעות פשוט לא ישימה.   
דוגמה קלאסית לחוק שכזה: מורה לא יכול ללמד ביותר ממקום אחד בו זמנית

* חוקים רכים (Soft):

חוק רך הוא חוק שרצוי שהמע' תעמוד בו, אולם מע' תחשב תקינה גם אם היא לא עומדת בו.  
דוגמה לחוק שכזה: כל מורה צריך לקבל יום חופש אחד בשבוע

1. הגדרה של גודל המע': מס' ימים בשבוע לימוד ומס' שעות לימוד בכל יום.

## הנחיות ספציפיות למימוש מערכת Evolutionary Time Table

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 יעבוד בצורה נכונה (ללא טעויות) ובזמן סביר.**
2. **אינכם נבדקים על מציאת פתרון נכון לבעיית מע' השעות עצמה.   
   מהות התרגיל היא בנייה מנוע תפעול האלגוריתם האבולוציוני ומידול ותפעול בעיית מע' השעות עליו.**

תרגיל 1 – מימוש **Evolutionary Time Table**כאפליקציית Console (25%) - הגשה: 8.8.2021

## פרטים יבשים

צפי תחילת עבודה: **14.7.21** תאריך הגשה: **8.8.21**

צפי זמן לביצוע: **3 שבועות** ציון אפשרי מקסימלי: 130

משקל התרגיל: **25%** 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 מע' השעות כמו גם על מאפייני מנוע האלגוריתם האבולוציוני. המע' תאפשר למשתמש לצפות בפתרונות השונים של מע' השעות המתגלים תוך כדי פעולת האלגוריתם במספר היבטים, כמו גם תאפשר להציץ אל נבכי האלגוריתם ולהבין מה קרה במהלך פעולתו (בכל דור ודור).

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. בתרגיל זה תקימו את מנגנון האלגוריתם האבולוציוני הבסיסי כמו גם תאפשרו את הגדרת מאפייני מע' השעות הכלליים (מורים, מקצועות, שכבות, חוקים וכו')

בתרגיל זה עליכם לממש את האופרטורים האבולוציונים הבאים:

* DayTimeOriented Crossover
* Flipping Mutation

מידע מסודר על אופי פעולת האופרטורים האבולוציונים הנ"ל מופיע בנספחים [ג'](#appendixC) ו[ד'](#appendixD) בהתאמה.

בתרגיל זה תתמכו במנגנון selection פשוט שנקרא Truncation.

פירוט לגבי אופן פעולתו מתואר [בנספח ה'](#appendixE)

בתרגיל זה תתמכו בחוקים הבאים:

* מורה לא מלמד באותה השעה ביום ביותר מכיתה אחת (TeacherIsHuman)
* בכל שעה יש לכל היותר השמה של מורה ומקצוע אחד בלבד (Singularity)
* מורה מלמד אך ורק את המקצועות שהוא מוסמך להם (Knowledgeable)
* כל כיתה מקבלת את מלוא (ובדיוק) דרישות הלימוד שלה (Satisfactory)

פירוט לגבי החוקים ותכליתם מתואר [בנספח ו'](#appendixF)

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

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. ה id של subject, class ו teacher מהווים ספרור רץ (בתוך כל קטגוריה) המתחיל מ 1
  3. חוק לא חוזר על עצמו פעמיים
  4. ברשימת המקצועות שמורה מלמד אין הפנייה למקצוע שלא קיים
  5. ברשימת המקצועות שצריך ללמוד בכל שכבה – אין הפנייה למקצוע שלא קיים
  6. סך שעות הלימוד לכלל המקצועות במסגרת שכבה מסויימת לא עולה על D**∙**H

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

יש לוודא תקינות הנתונים המופיעים בקובץ כפי שהוגדרו בסעיף 3.

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:**

1. 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)

1. פקודה זו מוצגת ומאופשרת תמיד. אפשר לבחור בה בכל רגע נתון במע'.
2. הצגת הגדרת המע' ומאפייני האלגוריתם

פקודה זו תאפשר למשתמש להבין את פרטי המע' כפי שנטענו זה לא מכבר מקובץ הבעיה.

יש להציג את פרטי מע' השעות:

* + 1. רשימת מקצועות.   
       עבור כל מקצוע מס' מזהה ושם. הציגו את המקצועות על פי סדר הספרור הרץ שלהם.
    2. מורים. עבור כל מורה:
       - מס' סידורי
       - רשימת מספרי מקצועות יחד עם שמם אשר הוא מלמד

הציגו את המורים על פי סדר הספרור הרץ שלהם.

* + 1. שכבות. עבור כל שכבה:
       - מס' סידורי
       - רשימת מקצועות נדרשים (מס' ושם) יחד עם כמות שעות לכל מקצוע

הציגו את השכבות על פי סדר הספרור הרץ שלהם.

* + 1. שמות החוקים ליישום במע'. ליד כל חוק האם הוא Hard או Soft

יש להציג את פרטי מנוע האלגוריתם האבולוציוני:

* + 1. גודל אוכלוסיה
    2. טכניקת ה selection (עם קונפיגרוציה מתאימה לה, ככל שניתנה)
    3. סוג ה crossover עם כמות נק' החיתוך
    4. מידע על המוטציות בשימוש. עבור כל מוטציה מידע על ההסתברות להפעלתה ועל הקונפניגורציה המלווה אותה

**דגשים**:

* אין לאפשר פקודה זו אם לא טעון קובץ תקין במע' (או להציג פלט מתאים).

1. הפעלת האלגוריתם האבולוציוני

פקודה זו תפעיל את האלגוריתם האבולוציוני על הגדרת הבעיה שטעונה כרגע במע'.   
באם זו אינה ההפעלה הראשונה של האלגוריתם (כלומר כבר יש תוצאות עבר), יש ליידע את המשתמש בנושא ולבקש את אישורו להפעיל את האלגוריתם מחדש. בהפעלה מחודשת כל המידעים הקודמים נמחקים והכל מתחיל מחדש.  
  
כדי להתחיל את הפקודה יש לקבוע מהו תנאי הסיום. בתרגיל זה נתמוך בתנאי סיום אחד והוא מספר הדורות.   
עליכם לבקש נתון מספרי זה מהמשתמש. נתון זה יהיה מספר שלם וחייב להיות גדול מ 100. (יש לוודא ולהתריע למשתמש במידה וסיפק מידע שאינו מתאים).   
בזמן פעולת האלגוריתם יש לעדכן את המשתמש על קצב התקדמות הדורות (כך הוא יבין שהמע' לא תקועה ומאחורי הקלעים התהליך נע ומתקדם). יש לבקש מהמשתמש כל כמה דורות הוא רוצה להציג את המידע. (מספר שלם חיובי). המידע שיוצג יהיה ה fitness של הפתרון הטוב ביותר בדור המדובר.  
  
מרגע הפעלת האלגוריתם המשתמש ממתין עד שיגיע לתנאי הסיום שלו.   
תו"כ תנועה הוא יקבל עדכון על התקדמות הדורות.   
בסיום פעולת האלגוריתם חוזרים ומציגים את התפריט הראשי.

**דגשים**:

* אין לאפשר פקודה זו אם לא טעון קובץ תקין במע' (או להציג פלט מתאים).

1. צפייה בפתרון הטוב ביותר

פקודה זו תאפשר למשתמש לצפות בנתוני הפתרון הטוב ביותר שנמצא ע"י האלגוריתם.

יש לבקש מהמשתמש קלט המגדיר כיצד יוצג הפתרון:

* גולמי (RAW) – המשתמש יקבל את אוסף החמישיות המגדיר את הפתרון. המבנה של כל חמישיה הוא <D,H,C,T,S>. החמישיות יוצגו כרשימה אנכית ויהיו מסודרות על פי היום, שעה, שכבה, מורה.
* מאופיין מורה (TEACHER) – המשתמש יקבל את מע' השעות של כל מורה (על פי סדר המספר המזהה שלהם).   
  עבור כל מורה תוצג מע' שעות דו מימדית (DH) ובכל תא יהיה כתובים 2 מספרים: מס' הכיתה ומס' המקצוע המלומד בה. כמובן יכולים להיות גם תאים ריקים. במידה ויש יותר משיבוץ באותה השעה יש לכתוב את כל השיבוצים (זו אמנם מע' לא תקינה אבל יש ליידע על כך את המשתמש).
* מאופיין כיתה (CLASS) – המשתמש יקבל את מע' השעות של כל כיתה (על פי סדר המספר המזהה שלהן).   
  עבור כל כיתה תוצג מע' שעות דו מימדית (DH) ובכל תא יהיו כתובים 2 מספרים: מס' המורה ומס' המקצוע הנלמד. כמובן כי יכולים להיות גם תאים ריקים. במידה ויש יותר משיבוץ באותה השעה יש לכתוב את כל השיבוצים (זו אמנם מע' לא תקינה אבל יש ליידע על כך את המשתמש).

בנוסף לפלט הנ"ל יש להציג גם:

* ציון ה fitness שקיבל הפתרון הנ"ל
* סקירה של כל החוקים שהגודרו בפתרון הנ"ל וציוניהם בהקשר פתרון זה.   
  עבור כל חוק יש להציג:
  + שם החוק
  + האם הוא Hard או Soft
  + ציון החוק בהקשר הפתרון הנ"ל
* ממוצע חוקי ה HARD וממוצע חוקי ה SOFT (יש להציג נקודה עשרונית אחת)

**דגשים**:

* אין לאפשר פקודה זו אם לא טעון קובץ תקין במע' (או להציג פלט מתאים).
* אין לאפשר פקודה זו אם טרם הורץ האלגוריתם ועל כן אין בכלל פתרון להציג...

1. צפייה בתהליך האלגוריתם

פקודה זו תאפשר למשתמש להציץ בהתפתחות הפתרונות השונים בדורות שעברו.   
היות ויכולים להיות הרבה דורות, יש להציג את הקפיצה בדורות על פי הנתון שנבחר בעת הרצת האלגוריתם.  
בכל דור שיוצג יש להציג למשתמש את הציון של הפתרון הטוב ביותר שנמצא בדור זה, כמו גם את השיפור/הרעה (כמספר חיובי / שלילי) בציון של הפתרון הטוב ביותר יחסית לפתרון הטוב ביותר בדור הקודם שהוצג (בדור הראשון כמובן אין צורך לספק נתון זה). הרשימה תתחיל מהדור הראשון ועד לדור האחרון.

**דגשים**:

* אין לאפשר פקודה זו אם לא טעון קובץ תקין במע' (או להציג פלט מתאים).
* אין לאפשר פקודה זו אם טרם הורץ האלגוריתם ועל כן אין בכלל פתרון להציג...

1. יציאה מהמע'

פקודה זו יוצאת מהמערכת.

1. בידקו והריצו את האלגוריתם שלכם על קובץ small על גדלים שונים של אוכלוסיה וכמות דורות ומלאו את המידעים הבאים על פי הטבלה (על הטבלה להופיע בקובץ ה readme). יש למלא את הזמן שלקח לכם להריץ כל קונפיגורציה. (לא צריך להיות מדוייקים עם סטופר).  
   על פניו נשמע הגיוני כי לפחות את הקוביות הירוקות הגיוני שתצליחו להריץ.  
   במידה ולא הצלחתם להריץ קונפיגורציה מסויימת בשל תקלה כזו או אחרת (נגמר הזכרון, באג, whatever) או אם ההרצה לוקחת יותר מ 3 דקות – כיתבו ועדכנו זאת בטבלה גם כן (בקצרה).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | כמות דורות | | |
| **1,000** | **10,000** | **100,000** |
| גודל אוכלוסיה | **200** |  |  |  |
| **500** |  |  |  |
| **1,000** |  |  |  |

## 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 הקמת תשתית האלגוריתם האבולוציוני. מומלץ מאוד לעבוד בצורה גנרית כמה שאפשר, כך שמנוע האלגוריתם יוכל לעבוד בצורה עצמאית בלי תלות חזקה (coupling) לבעיית מע' השעות הספציפית של תרגיל זה. חישבו איך לנצל את GENERICs כך שיאפשרו לכם להגדיר את המנוע האלגוריתם בצורה מנוטרלת לחלוטין.

המשיכו בייצור והגדרת מבני הנתונים של בעיית מע' השעות (מורה, מקצוע, כיתה, חוק).   
המשיכו ליכולת לטעון את המידעים השונים מהקובץ ואיתחול מבני הנתונים שלכם במידע הנ"ל.

לאחר מכן תימכו במנגנון הגדרת החוקים השונים. ממשו את החוקים השונים הרלבנטים לתרגיל זה.

לבסוף ממשו את האופרטורים האבולוציונים הרלבנטים לתרגיל זה.

## בונוסים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | סוג | מהות | למה שווה לי ? | כמה שווה לי ? |
| 1 | הקדמת תרופה למכה | תנאי סיום נוסף: by Fitenss  ממשו את תנאי הסיום הנוסף על פי הגעה ל fitness מוגדר.  עיקבו אחר ההוראות לגביו בתרגיל 2 סעיף 10.  במקרה זה, אפשרו למשתמש לבחור אחד מ 2 תנאי הסיום במהלך הפעלת האלגוריתם. | כי גם ככה הולכים לממש זאת בהמשך – אז למה לא ?? | 5 נקודות  (עד ל 100) |
| 2 | הקדמת תרופה למכה | Sizer Mutation: ממשו את אופרטור המוטציה שעליכם לממש כחלק מתרגיל 2. עיקבו אחר הגדרתו לפי [נספח ד'.](#appendixD)  במימוש בונוס זה תקבלו קובץ XML שמכווין לשימוש באופרטור זה | כי גם ככה הולכים לממש זאת בהמשך – אז למה לא ?? | 7 נקודות  (עד ל 100) |
| 3 | הקדמת תרופה למכה | Aspect Oriented Crossover  ממשו את אופרטור ה crossover הנ"ל שעליכם לממש כחלק מתרגיל 2. עיקבו אחר הגדרתו לפי [נספח ג'](#appendixC)  במימוש בונוס זה תקבלו קובץ XML שמכווין לשימוש באופרטור זה | כי הוא אחד הכבדים יותר וחבל על כל בזבוז של זמן במהלך תרגיל 2. אחד שיודע... | 10 נקודות **(מעל ל 100)** |
| 4 | הגדלת ראש | אפשרו למשתמש לשמור את מצב המע' לקובץ.  הוסיפו פקודה לתפריט שתאפשר למשתמש להכניס נתיב שאליו הוא ירצה שמצב המע' יישמר.  עליכם לשמור את הגדרת הבעיה, את תוצאת הרצת האלגוריתם כמו גם את המידע על השלבים השונים במהלך הרצת האלגוריתם.  הוסיפו פקודה נוספת לתפריט שתאפשר למשתמש לטעון את כל המע' ישירות מהקובץ ששמרתם אותה אליה. (שימו לב, זוהי פקודה נפרדת מפקודת החובה שכבר קיימת לטעינת המע' מקובץ ה XML).  לבחירתכם פורמט ואופן שמירת המידע של המע' לקובץ. | טעימה קלה מדרכי שמירת נתונים | 10 נקודות  **(מעל ל 100)** |
| 5 | הוכחה שלמדתם משהו | Thread נפרד:  פעולת האלגוריתם האבולוציוני תתרחש על thread נפרד, ואילו ה main thread לא יצטרך להמתין לסיומה.  במקרה זה לאחר הפעלת פקודה 3 מציגים ישר את התפריט הראשי (ע"י ה main thread) ומאפשרים למשתמש להמשיך לתקשר עם המע' בזמן שהאלגוריתם רץ מאחור.   המשתמש יוכל להפעיל את פקודה מס' 4 באופן עצמאי תו"כ הריצה והיא תציג לו חזרה את פרטי הפתרון הטוב ביותר שנמצא כרגע (כפונקציה של התקדמות האלגוריתם). יש להציג בנוסף לכל המידעים הנוכחים מהו הדור הנוכחי שבמסגרתו נמצא פתרון זה.  המשתמש יוכל להפעיל את פקודה מס' 5 באופן עצמאי תו"כ הריצה והיא תציג לו חזרה את המידעים עבור (עד) 10 קפיצות הדורות האחרונים וההתקדמות שבהם. אם מפעילים את פק' 5 לאחר גמר הפעלת האלגוריתם – היא מציגה את מלוא המידע כפי שהוגדר בפקודה המקורית.  שימו לב: בכל מקרה במימוש בונוס זה פקודה 4 ו 5 אינן blocking !  במידה ובוחרים להפעיל שוב את פקודה מס' 1 או 3 והאלגוריתם עוד רץ ברקע – יש להתריע על כך למשתמש ולבקש את אישורו המפורש לכך. | סוג של הכנה למזגן לקראת העתיד להגיע בתרגיל 2 (אם כי בזעיר אנפין) | 10 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 | 22.7.21 | האם בפקודה 5 אפשר יהיה לבחור להציג את התקדמות הדורות בכל קפיצה שהיא ? | לא. יש להציג את התקדמות הדורות אך ורק על פי נתון הצגת התקדמות הדורות שנבחר ע"י המשתמש בעת הרצת האלגוריתם. הדבר בא למנוע מכם את הצורך לשמור את כל המידעים עבור כל דור ודור ובכך להסתכן בלהגיע לניצול ומיצוי הזיכרון בשל כמויות המידע (נניח שרצים על מליוני דורות...) |
| 2 | 30.7.21 | איך נדע מהו המספר הסביר של כמות הדורות להרצה וגודל האוכלוסיה ? | מלאו את הטבלה של סעיף 7 ובידקו את עצמכם בהתאם לה. יש ציפיה כי לפחות את הקופיגורציות הירוקות תצליחו לרוץ בהצלחה. הוסיפו טבלה זו לקובץ ה readme שלכם. |
| 3 | 2.8.21 | אם מממשים את בונוס ה thread – מה עושים עם פקודה מס' 5 ? | מדפיסים עד 10 התקדמות קפיצת דורות אחורה. מעודכן גם בגוף הבונוס |

תרגיל 2 – מימוש**Evolutionary Time Table** כאפליקציית JavaFX (30%) – הגשה: 4.9.2021

## פרטים יבשים

צפי תחילת עבודה: **9.8.21** תאריך הגשה: **4.9.21**

צפי זמן לביצוע: **4 שבועות** ציון אפשרי מקסימלי: **130**

משקל התרגיל: **30%** 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 רבה יותר על התנהלות האלגוריתם, כמו גם על נראות הפתרונות השונים.

1. שיפורים במנגנון האלגוריתם ואפיון הבעיה:  
   1. בתרגיל זה נשכלל את מנוע האלגוריתם כך שהוא יתמוך גם באליטיזם.   
      אלטיזים היא טכניקה המאפשרת לכמות מסויימת של נבחרים ("האליטה") לשרוד ולעבור לדור הבא as is ללא כל שינויים.

האלטיזים בא לידי ביטוי במהלך תהליך ה selection. הוא מוגדר כמספר שלם וחיובי החסום בגודל האוכלוסיה ומהווה את כמות הפרטים בעלי ה fitness הטוב ביותר שצריכים לעבור לדור הבא כמות שהם.   
בתהליך ה selection תתחילו בלקיחת האליטה לדור הבא ואז את השארית תבחרו באמצעות טכניקת ה selection המוגדרת.  
ה elitism יוגדר בקובץ ה XML כתת אלמנט בתוך הגדרת ה selection  
שימו לב:

* אין חובה להשתמש ב elitism. זהו אלמנט אופציונלי וייתכן כי לא יופיע בהגדרת הקובץ כלל וכלל (מקביל להגדרת elitism = 0)
* פרט באוכלוסיה שנבחר להיות חלק מהאליטה יכול גם להיבחר לשמש כהורה עבור הדור הבא.
* אם אחרי הפחתת ה elitism יש צורך להשלים כמות אי זוגית של צאצאים – ביחרו שרירותית רק את אחד הפתרונות מה crossover האחרון שאתם מבצעים.  
  1. בתרגיל זה תתמכו בשיטת חדשה לביצוע ה selection: Roulete Wheel. פרטים על הטכניקה הזו מובאים [בנספח ה'.](#appendixE)

1. בתרגיל זה תתמכו ב 2 אופרטורים אבולוציונים חדשים:

* Aspect Oreiented Crossover – אופרטור המנסה להקפיד על שימור רכיבים שתורמים לפתרון כמות שהם גם בצאצא הבא
* Sizer Mutation – אופרטור המשנה את כמות החמישיות (מוסיף או מחסיר) בפתרון נתון.

פירוט של אופן פעולת האופרטורים האלה נתון בנספח [ג'](#appendixC) ו[ד'](#appendixD) בהתאמה

1. בתרגיל זה תתמכו בחוקים נוספים על מע' השעות:

* מורה זכאי ליום חופש אחד במהלך שבוע העבודה
* מקצוע לא נלמד ביותר מפרק זמן נתון

עיקבו אחר [נספח ו'](#appendixF) לפירוט לגבי אופן הגדרת ופעולת חוקים אלה

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](#appendixGSII) 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 שבוצעו בתרגיל 1 ובנוסף:

* רכיב ה elitism, אם קיים, קטן מגודל האוכלוסיה

1. לאחר טעינת קובץ תקין אפשרו למשתמש לקבל מידע על הרכב הקובץ (כפי שהיה בתרגיל 1):

עליו להיות מסוגל לקבל מידע על המורים, המקצועות והכיתות במע', כמו גם את המידעים השונים על מאפייני האלגוריתם לביצוע.

1. לאחר טעינת קובץ תקין אפשרו למשתמש להפעיל את האלגוריתם ע"י לחיצה על כפתור ייעודי.

לפני הפעלת האלגוריתם יש לבחור את תנאי הסיום הנדרש.  
בתרגיל זה תתמכו ב 2 תנאי סיום נוספים לאלגוריתם.   
ניתן לבחור יותר מתנאי אחד. התנאי הראשון שמתממש עוצר את האלגוריתם. חובה לבחור לפחות תנאי סיום אחד.

הנה כל תנאי הסיום שיש לתמוך בהם:

* סיום כאשר מגיעים לכמות דורות מוגדרת (כפי שהיה בתרגיל 1)
* סיום כאשר מגיעים לציון fitness מסויים.   
  בתנאי זה יינתן סף fitness מסויים וברגע שמגיעים אליו במי מהדורות – האלגוריתם עוצר.
* סיום כעבור פרק זמן נתון.   
  בתנאי זה יינתן סך הזמן (בדקות) שרוצים שהאלגוריתם יפעל במסגרתו.   
  כשעובר הזמן – פשוט עוצרים ומציגים את התוצר הטוב ביותר.

כמו כן יש לבקש מהמשתמש להכניס את נתון קפיצת הדורות.

הפעלת האלגוריתם תתבצע על גבי thread נפרד מה JAT ותאפשר למשתמש לקבל מידעים "בשידור חי" מדרך פעולתו. המשתמש יוכל לראות בכל רגע נתון את הדור הנוכחי, ואת ה fitness הטוב ביותר הנוכחי שיש כרגע.   
כמו כן יש להציג מכוון התקדמות של האלגוריתם (progress bar) שיגרום למשתמש להבין עוד כמה זמן נשאר לסיום האלגוריתם. יש להציג מכוון התקדמות לכל תנאי סיום שנבחר להשתמש בו.

1. שליטה בזרימת האגלוריתם:  
   במהלך פעולת האלגוריתם, המשתמש יוכל לבצע מגוון פעולות:
   1. הצגת הפתרון הטוב ביותר:  
      באמצעות כפתור ייעודי יישלף הפתרון הטוב ביותר הנוכחי ויוצג למשתמש באותו האופן שבו מוצג הפתרון בתום ריצת האלגוריתם (כפי שמוסבר בהמשך).
   2. השהיית פעולת האלגוריתם:  
      הציעו כפתור ייעודי Pause, אשר יאפשר למשתמש לבחור להפסיק את פעולת האלגוריתם בצורה ידנית כל אימת שירצה.   
      מאחורי הקלעים יש להשהות את פעילות האלגוריתם אך ורק במעבר בין דורות, לפני תחילת עיבוד הדור הבא.   
      אם תנאי הסיום הנבחר הוא "על פי זמן" – אזי עוצרים גם את שעון הזמן עד להמשכת האלגוריתם.  
        
      מרגע שעשה זאת יהיה ביכולתו לשנות מספר פרמרטים כלליים הנוגעים לאופן ביצוע האלגוריתם:
   * שינוי ה probability של ה mutation'ים השונים, כמו גם שאר הארגומנטים של כל מוטציה ומוטציה.
   * שינוי טכניקת ה selection (תוך אספקת פרמטרים רלבנטים אם צריך)
   * שינוי גודל ה elitism (גם אם כזה לא נבחר בהגדרת הקובץ המקורית).
   * שינוי מנגנון ה crossover (תוך אספקת פרמטרים רלבנטים אם צריך)

המטרה של ביצוע השינויים הוא לאפשר למשתמש להתאים את פעולת האלגוריתם תו"כ תנועה כשהוא מבין האם האפיון הראשוני שלו כנראה לא מצליח לספק פתרון מספיק טוב..

1. המשך פעולת האלגוריתם:  
   הציגו כפתור ייעודי Resume, המאפשר להמשיך את פעולת האלגוריתם, תוך התייחסות לשינויים שבוצעו בה (Fכל שהיו כאלה) עד שיגיע ל(מי מ)תנאי הסיום.
2. עצירה מוחלטת של פעולת האלגוריתם:  
   הציגו כפתור Stop אשר יאפשר למשתמש לעצור את האלגוריתם לחלוטין, תוך התעלמות ממצבם של תנאי הסיום השונים.

כך או אחרת, בסיום האלגוריתם, אם באופן טבעי ואם בצורה ידנית, יוצג למשתמש הפתרון הנוכחי הטוב ביותר.

1. הצגת פתרון מלא:

בתרגיל זה תוכלו (סוף סוף !) להציג למשתמש את הפתרון למע' השעות כנדרש.

בהינתן פתרון להצגה, יש לאפשר למשתמש לבחור אם לצפות בו בצורה גולמית (רצפי חמישיות), בחתך של מורה או בחתך של כיתה (כשם שהיה בתרגיל 1)

ב 2 החתכים השונים יש לאפשר למשתמש לבחור את המורה/כיתה המסויימים שהוא רוצה לראות את המע' שלהם (לא עוד מציגים את הכל תמיד !) ולהציג את מע' השעות (DH) כאשר בכל משבצת תינתן זהות המשתנים החופשיים:

בחתך המורה בכל משבצת יהיה מס' הכיתה, שם הכיתה, מס' המקצוע ושם המקצוע שהוא מלמד.

בחתך הכיתה בכל משבצת יהיה מס' המורה, שם המורה, מס' המקצוע ושם המקצוע הנלמד.

שעות ריקות פשוט לא יכילו שום מידע.  
המשתמש יוכל לעבור בין 3 האופנים השונים להצגת המידע בצורה חופשית.

לצד מע' השעות יש להציג את רשימת החוקים תוך חיווי ליד כל חוק מה מאפייניו (שם ופרמטרים אם קיימים) כמו גם מהו הציון שלו בהתאם לפתרון המוצג.

1. כחלק מבדיקת המערכת ישונה גודל המסך (resize) ותיבדק המע' שלכם במסך בגודל שונה. עליכם לדאוג לסידור נכון של רכיבי ה ui ולוודא את תקינותם גם בגודל קטן. מומלץ מאוד להשתמש ב scroll pane (יש ללמוד עליו לבד) כדי להציג תוכן גדול בתוך שטח מסך קטן.

**למען הסר כל ספק ומניעת כל התחכמות שהיא, אין "לטפל" ב resize ע"י כך שפשוט תמנעו מהמסך להיות resizable.**

במידה ומימשתם בונוסים בתרגיל הקודם אין הכרח לגרום להם לפעול גם בתרגיל זה, אולם אם זה מתאפשר זה יחמם את ליבי. (למען הסר ספק, חימום ליבי אינו מתורגם 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 פרטי פתרון. רכיב המציג את פרטי הבעיה. רכיב המציג את פנל השליטה על מהלך האלגוריתם 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 רכיב הבעיה. לאחר מכן בהפעלת האלגוריתם עצמו. אחרי כן בהצגת רכיב הפתרון ורק בסוף בהתעסקות עם כל השליטה הדינמית בתפעול האלגוריתם.

## 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. 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)** |
| 3 | הגדלת ראש חביבה | גרף התפתחות של ה fitness הטוב ביותר כפונצקיה של ההתקדמות בדורות. הציגו את הגרף בסיום פעולת האלגוריתם, לצד הפתרון המנצח | עניין של מה בכך... | **10 נקודות**  **(מעל ל 100)** |
| 4 | הקדמת תרופה למכה | Tournemant  ממשו את טכניקת ה selection המדוברת. פרטי הטכניקה מופיעים [בנספח ה'.](#appendixE) במקרה זה פרטי הטכניקה המדוברת יופיעו כחלק מקובץ ה XML | כי היא מה-זה פשוטה שחבל... | **5 נקודות (מעל ל 100)** |
| 5 | זה נשמע מעניין... | סריקת הפתרונות בכל דור ודור  אפשרו למשתמש לטייל לאורכה של האבולוציה בין קפיצות הדורות שונים ולראות מה היה הפתרון הטוב ביותר בכל דור ודור. בצורה שכזו המשתמש יזכה לראות ממש איך האבולוציה מפתחת ומגדלת את הפתרון מדור לדור.  הציעו לשמתמש כפתורי Prev\next ולחיצה על כל אחד מהם עוברת בין קפיצת הדורות קדימה/אחורה בהתאמה.  בכל מעבר מדור לדור יש לשלוף את הפתרון הטוב ביותר ולהציגו למשתמש בצורה מלאה. יש לאפשר בונוס זה רק בסיום פעולת האלגוריתם (בין אם עצמאית ובין אם ידנית) | כי כל המידע כבר שם... זה רק לזרום עליו... | **15 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 |
|  |  |  |  |

תרגיל 3 – מימוש **Evolutionary Time Table** כאפליקציית WEB (40%) – הגשה: 9.10.2021

## פרטים יבשים

צפי תחילת עבודה: **4.9.21** תאריך הגשה: **9.10.21**

צפי זמן לביצוע: **5 שבועות** ציון אפשרי מקסימלי: **130**

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 נפתח מע' web שתציע למשתמשים ממשק נוח ודינמי לפתרון בעיית מע' השעות ע"י אלגוריתם אבולוציוני.

המע' תאפשר למשתמשים להירשם אליה בצורה דינמית.

כל משתמש יוכל להעלות קובץ (מסוג של תרגיל 3 בלבד) המתאר את מבנה מע' השעות בלבד. משתמשים שונים במע' יוכלו לבחור את מע' השעות השונות, ולהגדיר בעצמם את מאפייני האלגוריתם האבולוציוני ולשגר אותו לפעולה.   
למשתמשים יהיה ממשק תפעול דינמי של האלגוריתם והם יוכלו לשנות את מאפייניו השונים כדי להשיג פתרון אולטימטיבי למע' השעות.   
בכל רגע נתון נוכל לראות מי מהמשתמשים השונים השיג פתרון טוב יותר באמצעות רקיחת השיקוי האבולוציוני הנכון על פי המתכון הסודי שלו...

1. במסגרת תרגיל זה תפתחו עוד 2 חוקים:
   1. לכיתה יש יום חופשי
   2. למורה יש העדפה לעבוד כמות שעות מסויימת במהלך שבוע העבודה. כמות השעות ניתנת כחלק מהגדרת המורה בקובץ ה XML.

עיקבו אחר [נספח ו'](#appendixF) לפירוט לגבי אופן הגדרת ופעולת חוקים אלה

כמו כן במסגרת תרגיל זה תפתחו גם טכניקת selection נוספת שנקראת tournemant.   
פרטי המידע על טכניקה זו מופיעים [בנספח ה'](#appendixE).

1. בתרגיל זה כל משתמש מתחבר למע' מ browser שלו, והאינטראקציה שלו עם המשתתפים האחרים מתבצעת למעשה אך ורק ע"י השרת המתווך בינהם.

פורמלית, יש לתמוך בחווית משחק תקינה מעל chrome בלבד.

גם אם פיתחתם מעל FF או IE (טפו..) – וודאו כי אתם עובדים בצורה מוצלחת מעל chrome.

האפליקציה תיבדק רק מעל chrome.

בקיצור chrome,

קאפיש ?

1. ככלל, כל פרטי המידע במע' צריכים להתעדכן בצורה אוטומטית עבור כל משתמש (=== כל browser) בצורה של Pull כפי שלמדנו בשיעור והודגם בדוגמא המסכמת.
2. אין להסתמך (או לתמוך) ב back כאמצעי למעבר בין המסכים השונים.   
   הניחו כי לא נתעלל באפליקציה ולא נשנה את שורת הכתובת, לא נמחק cookies וכו'.   
   הבדיקה תתבצע אך ורק ע"י שימוש בפונקציונליות שתאפשרו במסכי המשחק השונים (משל היינו הוריכם...)
3. 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. יש לדאוג ולוודא כי מסכי המערכת לסוגם מתנהגים יפה ב resize ולא מחסירים שום פרט.

הם יכולים להיות מכוערים, אבל שישארו פונקציונלים (כשם שהיה נהוג בתרגיל 2).

1. בתרגיל זה ישנם 3 מסכים עיקריים:
   1. **מסך רישום למע' (sign up)**

במסך זה יש לאפשר למשתמש להיכנס למע'.

לכל משתמש יש שם ייחודי לו ורק לו.   
על המע' לוודא ולהתריע במידה ומישהו מנסה להיכנס עם שם כפול.

**אין צורך** לבצע שום תמיכה בעבודה עם סיסמאות או כל אותנטיקציה מכל סוג שהוא.

אחרי רישום מוצלח תעברו למסך מס' 2...

* 1. **מסך ריכוז משתמשים ובעיות מע' שעות**

במסך זה יוכלו המשתמשים השונים לראות מי הם המשתמשים האחרים במע' המחוברים כרגע, כמו גם את סך כל הבעיות של מע' השעות הממתינות לפתרון.

יש להציג מידע על שמות כל המשתמשים הפעילים כרגע במע'.

משתמש יכול לבחור קובץ ממע' הקבצים ולהעלות אותו לשרת (חובה לעבוד עם FileChooser Dialog, כפי שהוצג בכיתה).

משתמש יכול להעלות יותר מקובץ אחד.

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

קובץ ה XML לא יכיל הגדרת הפעלה של האלגוריתם האבולוציוני אם כי רק הגדרה של מאפייני בעיית מע' השעות.

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

בנוסף יש לבדוק כי כמות שעות העבודה הרצויה לכל מורה אינה עולה על DH

במידה והקובץ נמצא תקין, הודעה מתאימה תוחזר למשתמש ומעכשיו כל המשתמשים במע' יכולים לראות את נתוני מע' השעות שהתווספו מהקובץ הזה ומופיעים בטבלת מע' השעות שהמע' מנהלת.

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

המשתמש שהעלה קובץ בהצלחה נחשב "לבעלים" של מע' השעות.

יש להציג מידע על כלל מערכות השעות המוגדרות במע'.   
עבור כל מע' שעות יש להציג:

* + 1. שם המשתמש שהעלה אותה למע'
    2. נתונים מספריים של D, H, C, T, S
    3. כמות חוקי Hard וכמות חוקי Soft מוגדרים בה
    4. כמה משתמשים כבר מנסים לפתור את המע' הנוכחית
    5. מהו ערך ה fitness המקסימלי שהושג כרגע עבור המע' (עבור מע' שאף אחד עוד לא מריץ – אפשר להראות 0)

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

כל משתמש, בין אם העלה קובץ כזה או אחר ובין אם לאו, יכולים לבחור את אחת מהבעיות הממתינות לפתרון ולעבור למסך מס' 3 שבו ניתן יהיה להגדיר, להפעיל ולצפות בתוצאות ריצת האלגוריתם על הבעיה הנבחרת...

* 1. **מסך תפעול האלגוריתם עבור בעיה ספציפית**

במסך זה משתמש יכול לראות את תפעול האלגוריתם האבולוציוני ואת הפתרון המתהווה של מע' השעות.   
  
ראשית יש להציג למשתמש את המאפיינים השונים של מע' השעות המדוברת:  
הציגו את רשימת החוקים המוגדרים בבעיה. עבור כל חוק את שמו והאם הוא Soft או Hard  
הציגו את רשימת המקצועות, המורים והכיתות השונות המוגדרות בפתרון על מאפייניהן השונים.  
(בקיצור, כל המידע המתאר את הבעיה כדי שלמשתמש יהיה מושג על היקיף הבעיה ויוכל להתאים לה את מאפייני האלגוריתם שהוא מוצא לנכון)

שנית יש לאפשר למשתמש לבצע הגדרה וכיוונון של האלגוריתם האבולוציוני. לשם כך יש לאפשר לו לבחור:

* + 1. גודל אוכלוסיה
    2. טכניקת ה selection (עם הפרמטרים השונים, היכן שזה רלבנטי) ;אפשרות לבחור elitism
    3. אופרטור ה crossover עם הגדרות ספציפיות לו אם קיימות
    4. אופרטור ה Mutation עם ההגדרות הספציפיות לו. יש לאפשר לבחור יותר מ mutation אחד.
    5. אוסף של תנאי סיום (כפי שהיה נהוג בתרגיל 2)
    6. גודל קפיצת הדורות

בגמר הגדרת האלגוריתם המשתמש ילחץ על כפתור ייעודי ובכך יאפשר לשרת להפעיל את האלגוריתם עבורו עם המאפיינים שבחר.   
כל הפעלה של אלגוריתם של כל משתמש מתרחשת ע"י thread עצמאי במע' שעליכם להגדיר בעצמכם (שימו לב לא לתקוע ולבזבז את ה thread המשרת את הבקשה !)

מרגע שהאלגוריתם מתחיל לעבוד המע' תמשוך ב pull את פרטי הסטטוס הנוכחי של פעולת האלגוריתם.  
יש למשוך ולהציג את המידע הבא:

* + 1. מס' דור
    2. Fitness הטוב ביותר כרגע

הציגו למשתמש כפתור ייעודי שיאפשר לו, לפי בחירתו, למשוך את פרטי הפתרון המלא הטוב ביותר שיש כרגע במע'.  
כדי להציג את הפתרון, הציגו למשתמש את טבלת לוח השעות המדוברת, עם יכולת לבחור בין חתכים שונים של הפתרון (כפי שהיה בתרגיל 2). בנוסף יש להציג את רשימת החוקים עם התוצאה של כל חוק בהקשר הפתרון המוצג.

כפי שהיה בתרגיל 2, המשתמש יהנה מפאנל פיקוד ושליטה על מהלך פעולת האלגוריתם:

* יכולת להפסיק את האלגוריתם (pause)
* יכולת לשנות את מי מהפרמטרים השונים של האלגוריתם
* יכולת להמשיך את פעולת האלגוריתם (resume)
* יכולת לעצור לחלוטין את הפעלת האלגוריתם (stop)

הציגו למשתמש כפתור נוסף המאפשר לו לחזור חזרה למסך מס' 2 (אין להסתמך על back !)

אם משתמש ביקר בעמוד 3 של בעיה מסויימת וכבר הגדיר/הפעיל בה את האלגוריתם, חזר לעמוד 2 ואז הוא בוחר לעבור שוב לעמוד 3 של בעיה זו – יש להציג לו את המצב הנוכחי של העמוד כפי שהוא כרגע בנבכי המע'.

משתמש יכול לבחור לקנפג ולהפעיל את האלגוריתם על מספר בעיות במקביל.   
כל תפעול של בעיה מתבצע ב thread עצמאי משלה.   
משתמש לא יכול לבחור להפעיל את האלגוריתם מספר פעמים על אותה הבעיה במקביל.

כפי שכבר הבנתם, עבור בעיה מסויימת ייתכן כי ברגע נתון כמה וכמה משתמשים מריצים אותה במקביל.   
בעמוד זה אפשרו למשתמש לקבל מידע לגבי איזה משתמש מריץ בעיה זו ומה מצב הרצתו.

יש להציג את המידעים הבאים:

* שם המשתמש
* פיטנס מקסימלי שהשיג המשתמש
* דור נוכחי שריצת האלגוריתם של המשתמש מצוי בו

## 

## חלוקה למודולים

יש לייצר מודול נוסף שממנו ייבנה war המכיל את כל ה jar'ים הנדרשים (Jar של מנוע המע' ; gson.jar ; אולי אחרים ?).

מודול זה מכיל גם את קוד ה UI (HTML, CSS, JS) וגם את קוד הסרבלטים המקשרים למודול המנוע.

## 

## איך מתחילים ?

אני ממליץ להתחיל במעבר יסודי על הדוגמא המסכמת של הקורס.   
היא תקנה לכם שליטה בסיסית ברזי אפליקצית ווב (בסיסית) ובתפקידים השונים של כל רכיב ושחקן בסיפור.

לאחר מכן התחילו בעבודה על פי סדר העמודים:

ראשית התחילו בעמוד הלוגין. הוא יכול להיות זהה בעיקרו לעמוד 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 עמוד מס' 2. ביחרו פיצ'ר נוסף וממשו אותו בצורה מלאה מה-UI, עבור לסרבלט המתאים בשרת וחזרה ל UI, בו אתם מייצרים את ה HTML בהתאם לתשובה שחזרה מהשרת.

ממליץ מאוד (כמו ב JAVA FX) לייצר עמוד HTML סטטי שמכיל "סקיצה" ראשונית של המידעים שאתם רוצים להציג כך שתוכלו להבין איך הוא אמור להיראות, מי הרכיבים שבו, איך לעצב אותו מבחינת CSS וכן הלאה. זה גם ייתן לכם הבנה ראשונית של מהם כל קריאות ה HTTP שאתם עתידים לייצר מכל רכיב ורכיב, באיזו תדירות הם יוצאים ? איזה סרבלטים משרתים אותם (האם סרבלט אחד לכולם או כמה "סרבלטונים" קטנים ?) וכן הלאה. אחרי שיש לכם סקיצה – תוכלו ב JS להתחיל לייצר את התוכן ה HTML של כל אזור ואזור ולעבוד על רכיב ורכיב "לעומק" (UI -> SERVER -> UI).

כך מסיימים את עמוד 2 ואז עוברים באותו אופן לעמוד 3. אחרי שכל התשתית והשלד הראשי של האפליקציה בנוי – מוסיפים את כל הכפתורים השונים ומפתחים כל פיצ'ר עד סופו.

## 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 | הגדלת ראש קלילה | אפשרו למשתמש "להציץ" לנבכי הפעלת האלגוריתם של משתמש אחר.  בבונוס זה במסך 3 של בעיה מסויימת, כשניתן לראות כי יש עוד הרצות במקביל של משתמשים אחרים, הציגו למשתמש 2 כפתורים ייעודים:   1. הצצה לנבכי הגדרת האלגוריתם הנוכחית של המשתמש המסויים. הציגו את שלל המידעים השונים שיאפשרו להבין מה הייתה תצורת הפעלת האלגוריתם 2. הצצה לפתרון הטוב ביותר שיש כרגע עבור משתמש מסויים | בתכנון נכון – אמור להיות פיסת עוגה ולא יותר... | 10 נקודות  (ועד ל 100) |
| 2 | הגדלת ראש בלתי ניתנת להכלה במגניבותה ! | בעמוד מס' 3 הציגו גרף המתאר את שינוי ה fitness כפונקציה של הדורות והתקדמות האלגוריתם (כשם שהיה בבונוס של תרגיל 2). חיקרו ועיבדו אם איזו ספריית JS שתמצאו לנכון (או בצורה native). יש למשוך את נתוני הגרף מהשרת כל פרק זמן קבוע, כשם שמושכים את כל שאר המידעים. | כי בכל זאת אנחנו מזרזים את האבולוציה... אז לא נשוויץ ?? | **10 נקודות**  **(מעל ל 100)** |
| 3 | הגדלת ראש סופר דופר ! | ממשו את שכבת ה UI של תרגיל 3 כחלק מאפליקציית Java FX.  אפשר ורצוי כמובן להסתמך על האפליקציה הקיימת שכבר מימשתם בתרגיל 2.  בבונוס זה יש לאפשר למשתמש להירשם למע', לאחר מכן לייצר עבורו עמוד (java fx) המקביל לעמוד 2 שבו יוכל לראות את פרטי הבעיות השונות, כפי שמתואר בעמוד מס' 2 (אפשר לוותר על הצגת רשימת המשתמשים במע'). משם יוכל לבחור את אחת הבעיות, להגדיר עבורה את מאפייני האלגוריתם ולהריצו (הרצה ותפעול אלגוריתם על בעיה כבר ממומש לכם כחלק מתרגיל 2...)  עליכם לבחור ולהשתמש כאן ב HTTP Client שיפעל מול אותו השרת, כפי שקורה בתרגיל 3 (למעשה בונוס זה ייבדק כך שחלק מהמשתמשים הם ב browser וחלקם ב java fx.) | [Oh My God !!!](https://www.youtube.com/watch?v=iMs9feeSknk) | **20 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. קבצי קוד המקור של הפרויקט שלכם (גם צד השרת וגם צד הלקוח).
2. 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.

**בפרט חייב להכיל קובץ ה readme את ה url שאיתו מנווטים אל מסך ה login של המשחק שלכם.**

1. יש לכלול בקובץ ה **readme** גם תיעוד והסבר כללי (וממצה) של המחלקות העיקריות החדשות ותפקידם. (חישבו מה יסייע לבודקת להיכנס ביתר קלות לקוד שלכם ולהבין מי נגד מי...)
2. במידה ומימשתם את מי מהבונוסים לעיל, ציינו זאת בקובץ ה readme כדי שהבודקת תדע לבדוק בהתאם.

בונוס שימומש אבל לא יתועד – לא ייבדק !

הגשה באיחור, שאינה באישור, תבטל כל מימוש בונוס. אין להגיש באיחור בשביל להספיק לעשות

בונוסים. תכננו את הזמן בהתאם.

## שאלות ותשובות

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

נספח א' – תיאור פעולת האלגוריתם האבולוציוני

כאמור, האלגוריתם האבולוציוני שואב את השראתו מאופן פעולת האבולוציה בטבע.   
לכל פרט בטבע יש גנים המגדירים את אופן פעולתו. הגנים הללו מקודדים את המאפיינים השונים של הפרט בטבע.   
במקבילת האלגוריתם האבולוציוני הגנים הם הדרכים לתאר פתרון אחד אפשרי של הבעיה.  
אוכלוסיית הפרטים בטבע, שבה לכל פרט יש גן משלו והוא בעל מאפיינים ייחודיים לו מוקבלת באלגוריתם למגוון של פתרונות שונים לבעיה (לא כולם נכונים או טובים)

כל הפרטים בטבע מתחרים על משאב משותף ודל (נגיד מזון – אין מספיק לכולם) ורק הפרטים בעלי המאפיינים המתאימים ביותר למעשה שורדים. מנגנון זה נקרא מנגנון הברירה הטבעית.   
במקבילת האלגוריתם האבולוציוני נצטרך להגדיר דרך שבה נבחן עד כמה כל פתרון ופתרון מתאים לבעיה המסויימת שעלינו לפתור. רצוי שנוכל להתבטא בצורה רציפה (מתאים יותר/פחות) ולא בצורה דיכוטומית (כן/לא) כדי לאפשר מנעד של פרטים שיזכו לשרוד...

הפרטים בטבע ששרדו את מנגנון הברירה הטבעית הם אלה שייבחרו (בעל כורחם) לייצר את הדור הבא. צאצא של זוג הורים מורכב מהמרכיבים הגנטיים של 2 ההורים האלה ומהווה סוג של הכלאה בינהם. 2 הורים בעלי מאפיינים השרדותיים גבוהים (כאלה ששרדו את מנגנון הברירה הטבעית) יורישו את הגנים (הטובים) שלהם לצאצאיהם וכך הצאצאים יהווה הכלאה טובה יותר (בשאיפה) של 2 הוריהם.   
במקבילת האלגוריתם נצטרך להגדיר טכניקה שבה בהינתן 2 פתרונות (זו הורים) נוכל לייצר מהם 2 פתרונות חדשים שיבוססו, כך או אחרת, על בסיס המרכיבים של הפתרונות ההורים

כחלק מלידת הצאצא, בהתסברות מסויימת, ייתכן וחלק מהגנים שלו עוברים שינוי אקראי כזה או אחר. לתהליך זה קוראים מוטציה. המוטציה בטבע יכולה לגרום לגן של הצאצא להיות טוב עוד יותר מזה של הוריו (ופרטים אחרים באוכלוסיה) או לחילופין להרע את מצבו ולהפוך אותו לגן מוחלש (שסביר שאם זה המצב הוא ייעלם בברירה הטבעית של הדור הבא).   
במקבילה האלגוריתמית נגדיר תהליך של מוטציה שמקבל לידיו פתרון (צאצא) אחד ובהסתברות מסויימת בוחר חלק מסויים בפתרון ופשוט משנה אותו, כך או אחרת. בתהליך הרבייה ויצירת הדור הבא יכולים להתבצע יותר מסוג אחד של מוטציה על גבי צאצא (פתרון) אחד.

בסוף תהליך זה יהיה בידינו את הדור הבא, אשר סטטיסטית, משופר ומותאם יותר מדור ההורים שיצר אותו להתמודד עם המשאבים הדלים ולשרוד את מנגנון הברירה הטבעית של יום המחרת.  
תהליך זה חוזר על עצמו מראשית עידן החיים שוב ושוב ושוב...

האלגוריתם האבולוציוני, אם כן, מורכב ממספר שלבים החוזרים על עצמם שוב ושוב עד שמגיעים לתנאי סיום רצוי.  
הנה תיאור פורמלי של פעולת האלגוריתם:

1. **אוכלוסיה ראשונית (Initial Population)**

בראשית פעולת האלגוריתם נגדיר את גודל האוכלוסיה שבתוכה תבוצע האבולוציה.   
ככל שהאוכלוסיה גדולה יותר – כך יהיה יותר מגוון של פתרונות וסיכוי למצוא את הפתרון הנכון, אולם כך גם האלגוריתם יפעל לאט יותר.

האלגוריתם מתחיל מיצירה רנדומלית של מספר פתרונות אפשריים לבעיה כגודל האוכלוסיה.   
שימו לב כי בהחלט ייתכן ולא כל הפתרונות "חוקיים" או הגיוניים.

1. **הערכת תאימות (Fitness Evaluation)**

כל פתרון מוערך ונקבע לו ערך תאימות (fitness). ערך זה מתאר כמה הפתרון הנ"ל "מתאים" לפתרון הבעיה.   
אפשר להחליט שרירותית מהם גבולות ה fitness (לדוגמא: מספר בין 0 ל 100 או מספר בין 0 ל 1) כמו גם האם fitness גבוה נחשב טוב יותר או בדיוק הפוך.   
לצורך קביעת ה fitness יש לספק פונקציה שמקבלת כקלט פתרון בודד ומחזירה כפלט את המספר המדובר.  
  
בסוף שלב זה נקבל מעין מיפוי בין כל פתרון ל fitness שלו.

1. **בדיקת תנאי סיום**

בשלב זה נבדוק האם הגענו לתנאי הסיום של האלגוריתם. יש מגוון של תנאי סיום לאלגוריתם.   
אם כן נחזיר את הפתרון (או הפתרונות) בעלי ערך התאימות הגבוה ביותר שנמצא עד עכשיו ונסיים.  
אם לא – נמשיך הלאה לשלבים הבאים...

1. **ברירה טבעית**

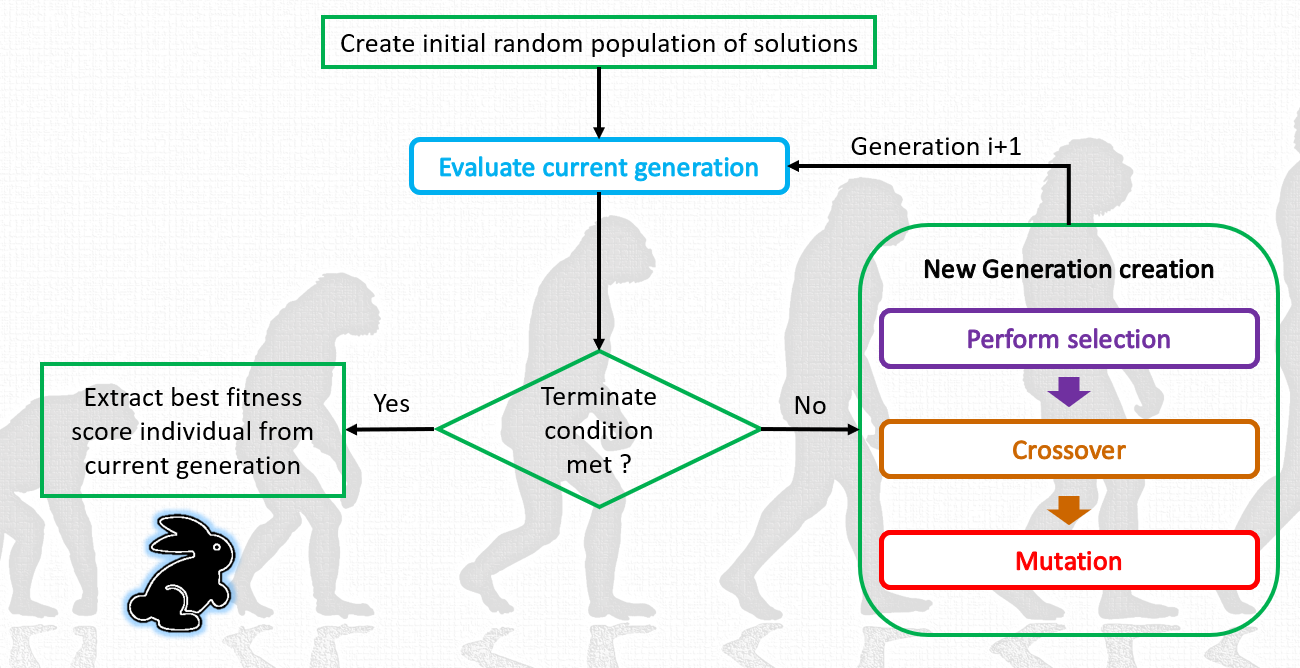
נבחר קבוצה של פתרונות בעלי ערך תאימות גבוה ביותר. פתרונות אלה ייחשבו "שורדים" ויהוו את הבסיס ליצירת "הדור הבא" (ובשאיפה הטוב יותר, המתאים יותר) של הפתרונות.  
ישנן דרכים שונות ומגוונות לביצוע בחירת הפתרונות (Selection) שייקראו "הורים" ואיכות האלגוריתם מוגדרת, בין היתר, בטכניקת הבחירה הנ"ל.  
כל טכניקת selection תבחר כמות מסויימת של הורים. שימו לב אין חובה לבחור כמות הורים ככמות האוכלוסיה.

1. **ייצור הדור הבא**

מבין הפתרונות הנבחרים בשלב הקודם נגריל 2 פתרונות שיהוו הורים. נייצר מ 2 ההורים 2 פתרונות חדשים שיהוו סוג של הכלאה בין המרכיבים השונים של הפתרונות ההורים.   
מנגנון יצירת הצאצאים מההורים נקרא Crossover. ישנם הרבה סוגים של crossovers וטכניקות שונות המגדירות כיצד לייצר זוג פתרונות מזוג פתרונות אחר.   
  
לאחר שלב זה נפעיל בהסתברות מסויימת תהליך של Mutation (אחד או יותר) על כל אחד מהצאצאים.   
Mutation הוא אופרטור אבולוציוני נוסף שנועד להגביר את המגוון של הפתרונות ולאפשר להימנע מהתכנסות לאוכלוסיה בינונית. ה Mutation בוחר פן מסויים של הפתרון (הצאצא) ומשנה רכיב מסויים בו בהסתברות מסויימת.

נמשיך בתהליך הנ"ל עד שנייצר את כמות צאצאים חדשה כגודל האוכלוסיה.   
במהלך התהליך ניתן לבחור את אותו ההורה שוב ושוב כדי לייצר ממנו צאצאים (מונוגמיה היא המצאה של האדם...).   
קבוצת הצאצאים החדשה נחשבת "לדור הבא".   
עתה נחזור חזרה לשלב 2, כאשר האוכלוסיה שלנו מונה את הדור הבא בלבד...

בתהליך איטרטיבי זה, לאורך הדורות, לאורך פרק זמן מסויים, בתקווה ובשאיפה, לאט לאט, נגדל ונתכנס לפתרון טוב ומתאים יותר ויותר לבעיה איתה התחלנו.



נספח ב' – תיאור בעיית מע' השעות בראי האלגוריתם האבולוציוני

תיאור פתרון בודד

מע' השעות מוגדרת ע"י טבלה דו מימדית של ימים (D) כפול שעות לימוד (H) ביום. גודל הטבלה יגדיר גם את גודל הפתרון שלנו.

עבור שכבה/כיתה מסויימת (C), בכל שעה של כל יום יש לציין איזה מורה (T) מלמד איזה מקצוע (S) ע"י ציון מס' מורה ומס' מקצוע.   
בהחלט ייתכנו שעות ריקות שבהם אף מורה לא שובץ ללמד אף מקצוע.   
  
ניתן לייצג מידע זה באמצעות רביעיה של נתונים: T,S,H,D

היות ובמע' מוגדרים מס' שכבות, הרי שכדי לייצג פתרון מלא של בית הספר כולו (המכיל את כל השכבות) נוכל להשתמש באוסף של חמישיות:

<D, H, C, T, S>

הנה מענה על כמה שאלות מעניינות:

**כמה חמישיות שונות סה"כ יכולות להתקיים ?**

**TF** = D \* H \* C \* T \* C

**כמה חמישיות צריכות להופיע בפתרון תקין ?**

סך כמות כל דרישות הלימוד לכל המקצועות בכל הכיתות (**TSR** = **T**otal **S**tudy **R**equirements)

**כמה פתרונות אפשריים אפשר לייצר ?**

בהנחה שרוצים לייצר פתרון המכיל בדיוק TSR חמישיות, שלא חוזרות על עצמן, הרי שעלינו לבחור מבין סך החמישיות TF סה"כ TSR חמישיות. חמישייה אינה צריכה לחזור על עצמה ואין כמובן חשיבות לסדר בחירת החמישיות.

**TF**

**TSR**

**TF !**

**TSR! (TF! – TSR!)**

**=**

כדי לחשב זאת נוכל להשתמש בבינום של ניוטון... (כן. הוא.):

ייצור פתרון בודד

כדי לייצר פתרון יש להגריל את כמות החמישיות שבו. לא בכל הפתרונות צריכים להיות את אותה כמות החמישיות כמובן. חישבו איך להגדיר ולמצוא מהו סך החמישיות הנדרש למציאת פתרון חוקי ושלם ושאפו לייצר פתרונות בעלי כמות החמישיות הנ"ל.  
  
כדי לייצר חמישיה יש להגריל את כל אחד מהערכים שלה, בטווח המוגדר לו.  
גם כאן ייתכן ואין הגיון להגריל את כל סוגי החמישיות (למשל עבור מורה T להגריל S שהוא לא מלמד בכלל). חישבו (ובעיקר התנסו) בטכניקות שונות של הגדרת החמישיות. לפעמים דווקא חמישיות "לא נכונות" יכולות לייצר מגוון באוכלוסיה וליצור את הדרך להגיע לפתרון מתאים יותר.

תיאור פונקציית ערך התאימות

במע' יוגדרו מספר חוקים אשר ימדדו את איכות הפתרון. כל חוק יתרכז בהיביט אחד מסויים של הפתרון.   
במסגרת תיאור הבעייה נציין באיזה חוקים ברצונינו לעמוד.   
התוצאה של כל חוק היא ערך מספרי בין 0 ל 100. ככל שהמספר גבוה יותר, פירושו כי עמדנו בדרישות החוק בצורה מלאה יותר. ציון 100 פירושו כי עמדנו בכל דרישות החוק. ציון 0 פירושו כי כל רכיבי הפתרון לא עומדים בדרישות החוק.

ישנם 2 סוגים של חוקים: חוק נוקשה (Hard) וחוק מקל (Soft).   
חוקים נוקשים הם כאלה שאי עמידה בהם פשוט לא תאפשר את פעילות המע' (למשל מורה המלמד באותה שעה ביום בכמה שכבות שונות).   
חוקים קלים הם כאלה שיהיה נחמד לעמוד בהם – אבל לא נורא אם לא.   
כדי למדל את היחס הנ"ל ייקבע משקלם של סך החוקים הנוקשים מתוך הציון הכללי של הפתרון.

פונקציית בדיקת ערך התאימות (fitness function) תקבל לידה פתרון מסויים ואת רשימת החוקים. הפונקצייה תפעיל את כל החוקים על הפתרון הנ"ל ותשמור את תוצאת הפעלת כל חוק על הפתרון.   
לאחר מכן תחשב ממוצע של כל ציוני החוקים לפי הקטגוריה שלהם (HARD ו SOFT) ולבסוף תחשב ממוצע משוקלל של 2 הממוצעים על פי היחס המוגדר בינהם בתיאור הבעיה.   
התוצאה תהיה מידת תאימות הפתרון לבעיה. התוצאה תהיה גם היא מספר בין 0 ל 100. ציון 100 אומר שמצאנו פתרון שעונה על כל החוקים המוגדרים. ציון 0 אומר שמצאנו פתרון שאינו עונה על אף אחד מהחוקים המוגדרים...

יצירת הדור הבא

Crossover

כדי לייצר פתרונות חדשים יש צורך בזוג הורים אשר מרכיביהם השונים יילקחו בדרך כזו או אחרת כדי לייצר את הצאצא.

לפני ביצוע כל Crossover נסדר את הפתרונות ההורים (אוספי החמישיות) בצורה מסויימת (כל crossover והטכניקה שלו).   
נגריל נק' חיתוך (אחת או יותר) ואז נבצע שחלוף של הרכיבים בין 2 ההורים כדי לייצר את הצאצא.   
בסיום פעולת ה Cross over יהיו בידינו 2 פתרונות חדשים שהם שעטנז של 2 הפתרונות ההורים.

Mutation

כל פתרון "צאצא" יועבר דרך סדרה של Mutations (אחד או יותר).  
כל Mutation יקבל כקלט את הפתרון הצאצא ויחזיר את אותו הפתרון כפלט, ייתכן עם שינוי קל במי ממרכיביו (על פי הסתברות מסויימת).  
כל Mutation נבדל באופי השינוי שהוא מבצע במי מהמרכיבים השונים של הפתרון.

נספח ג' – CROSSOVER

במסגרת התרגיל תדרשו לממש מס' טכניקות שונות לביצוע CROSSOVER. הנה תיאור מפורט של איך כל אחת מהן אמורה לעבוד.

כל האופרטורים עובדים טכנית בצורה דומה ומה שמשנה הוא בעיקר סדר החמישיות ההתחלתי.

אופן הפעולה:  
האופרטור מסדר את כל החמישיות של 2 ההורים בצורה מסודרת כך שהם ממוינים בצורה כלשהיא (נקבעת ע"י האופרטור הקונקרטי בשימוש). אם ישנה כניסה ללא חמישיה מתאימה – היא עדיין קיימת ונותרת ריקה. האופרטור הנ"ל מניח את 2 החמישיות הממוינות של 2 ההורים אחת מתחת לשנייה בצורה הבאה:

**. . .**

Parent 1

k

j

i

n

4

3

2

1

**< >**

**< >**

**< >**

**< >**

**< >**

**. . .**

Parent 2

**< >**

**< >**

**< >**

**< >**

בשלב הבא האופרטור מגריל את נק' החיתוך כפי שנדרש בהגדרת קובץ ההפעלה.   
נק' החיתוך הם מיקומים בסידור כל החמישיות וככאלה הן נעות בין 1 ל בין גודל הפתרון:

**. . .**

Parent 1

k

j

i

n

4

3

2

1

**< >**

**< >**

**< >**

**< >**

**< >**

**. . .**

Parent 2

**< >**

**< >**

**< >**

**< >**

כדי לייצר את הפתרונות הצאצאים עוברים מן ההתחלה ועד לנק' החיתוך הקרובה.   
לוקחים את החלק של ההורה הראשון לצאצא הראשון ואת החלק של ההורה השני לצאצא השני.   
ממשיכים לנק' החיתוך הבאה אך הפעם הופכים את הסדר: החלק של ההורה הראשון הולך להמשך הצאצא השני ואילו החלק של ההורה השני הולך להמשך הצאצא הראשון.  
כך ממשיכים בין נק' החיתוך (או עד שמגיעים אל הסוף) ובכל מקטע הופכים את סדר שיוך מקטעי ההורים לצאצאים.

התוצר הוא 2 צאצאים המורכבים מערבובי החלקים של ההורים, כפי שהוגדרו בנק' החיתוך:

Offspring 1

**. . .**

k

j

i

n

4

3

2

1

**< >**

**< >**

**< >**

**< >**

**< >**

Offspring 2

**. . .**

**< >**

**< >**

**< >**

**< >**

ה crossovers השונים נבדלים באופן הסידור הראשוני של החמישיות של ההורים. מרגע שהסידור בוצע – המשך העבודה (נק' חיתוך, שחלופים) פועלים באותו האופן.

**שם: DayTimeOriented**

אופן הפעולה:   
אופרטור זה מסדר את כל החמישיות בטבלה בגודל של DH. סדר הופעת החמישיות הוא על פי היום/שעה/כיתה/מורה המצויינות בהן.  
מתחילים מן היום הראשון ובתוכו כל השעות על פי הסדר (1..H). כך פועלים עד שמגיעים ליום D. אם יש כניסה בטבלה (Dn, Hm) שאין בה חמישיה שבה היום והשעה הם n, m (בהתאמה) הרי שאז היא נותרת ריקה.

**שם: AspectOriented**

אופן הפעולה:  
אופרטור זה מסדר את החמישיות כך שיעודדו מצב שבו היביט מסויים של הפתרון יקבל קדימות על פני היביט אחר.

פרמטרים:

1. היביט פעולה (Orientation) – יכול להיות אחד מ 2 הערכים הבאים: TEACHER ; CLASS

היביט מורה:

בגישה זו מסדרים את כל החמישיות על פי כל המורים (1.. T). עבור כל מורה t, יהיו DH כניסות ושיבוץ החמישיות עבור המורה t יבוצע על פי DH שלהן. (גם כאן, כמובן יהיו הרבה כניסות ריקות עבור כל מורה).

בהמשך העבודה יוגרלו נק' החיתוך בתוך כל מקטע של כל מורה. עבור כל מקטע של כל מורה יווצרו הצאצאים.

בשלב האחרון לוקחים את הצאצאים שנוצרו בכל מקטע של כל מורה ומאחדים אותם לכדי צאצא אחד. מתחילים מ T1 עד Tt. במידה ועבור כניסת DH יש יותר מחמישיה אחת מתאימה – פשוט מגרילים איזו לקחת.

היביט כיתה:

גישה זו דומה מאוד לגישת היביט על פי מורה, רק שבה מסדרים את כל החמישיות ראשית על פי כל הכיתות (1..S). גם כאן עבור כל כיתה יהיו DH כניסות וכו'.

נספח ד' – Mutations

מוטציות הן אופרטורים הפועלים על פתרון מסויים ומשנים רכיב מסויים בו. האופרטור אינו פועל תמיד, כי אם בהתסברות מסויימת. מרגע שההסתברות חלה - לכל אופרטור יש את דרך הפעולה שלו.

**שם: Flipping**

פרמטרים:

1. מקסימום חמישיות לשינוי (MaxTupples) – מספר שלם.
2. רכיב לשינוי (Component) – אחת מהאותיות הבאות ב upper case בלבד: D,H,C,T,S

דרך הפעולה:

אופרטור זה פשוט משנה את אחד הרכיבים בחמישיה (D,H,C,T,S) בצורה רנדומלית, בחמישיה רנדומלית.

ראשית האופרטור יגריל את סך החמישיות לביצוע המוטציה בהן. מקסימום מוגדר ע"י MaxTupples  
שנית האופרטור יגריל את זהות החמישיות הרלבנטית.  
לבסוף האופרטור יגריל את הרכיב הנבחר בחמישיה המדוברת למספר אחר מתוך סך האפשרויות (D,H, C, T, S)

**שם: Sizer**

פרמטרים:

1. סך הכל חמישיות לשינוי (TotalTupples). מספר שלם (חיובי או שלילי)

דרך הפעולה:

אופרטור זה אחראי על הוספה או קיצוץ של חמישיות מסך הפתרון.  
  
אם הפרמטר הוא חיובי – הרי שזהו מספר החמישיות שהאופרטור הולך להוסיף (ובכל מקרה לא יותר מ DH)  
כדי לייצר חמישיה – האופרטור פשוט יגריל אותה מבין הטווחים (D,H, C, T, S) ויוסיף אותה לפתרון.  
  
אם הפרמטר הוא שלילי – הרי שזהו מספר החמישיות שהאופרטור הולך להסיר (ובכל מקרה לא יישארו פחות מ D חמישיות)

האופרטור יגריל מספר בטווח של 1 עד TotalTupples וזו תהיה הכמות של החמישיות אותם הוא מוסיף/גורע/

נספח ה' – Selection Techniques

Selection הוא אחד המרכיבים החשובים ביותר בפעולתו של האלגוריתם האבולוציוני. ככל שבחירת ההורים המעמידים את הדור הבא תהיה איכותית יותר – כך יגדל הסיכוי לקבל צאצאים טובים לפחות כמו הוריהם, אם לא טובים מהם’ וכך תקצר הדרך אל הפתרון הנכסף.

ישנן טכניקות רבות לביצוע selection. אנו נתמוך בטכניקות הבאות:

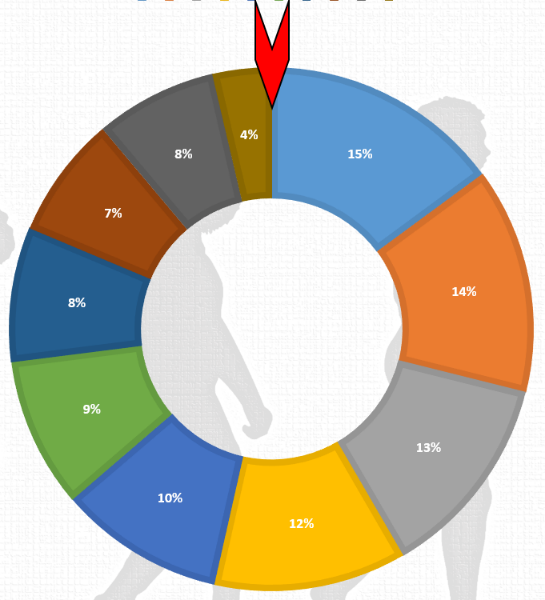
**שם: Truncation**

פרמטרים:

1. % בחירה (topPercent). מספר שלם בין 1 ל 100 (כולל)

בטכניקה זו מסדרים את כל הפתרונות על פי סדר ה fitness שלהם מהגבוה אל הנמוך.   
לאחר מכן בוחרים את % המועמדים הנדרשים מראשית הטבלה (בעלי ה fitness הגבוה ביותר) על פי ה topPercent שהוגדר.

דוגמא:  
אם ה top percent שניתן הוא 10 והאוכלוסיה מונה 50 פרטים, הרי ש 10 % מ 50 הם 5 ועל כן נבחר את 5 הפתרונות הטובים ביותר להיות ההורים שיעמידו את הדור הבא.  
במידה והאחוז מתוך האוכלוסיה אינו מספר שלם – קחו את החלק השלם בלבד (floor)

**שם: RouletteWheel**

בטכניקה זו בוחרים כמות הורים כגודל האוכלוסיה. כל זוג הורים יוליד 2 צאצאים במקומו.   
בשיטה זו מסדרים את כל הפתרונות על גלגל דמיוני כך שכל פתרון מקבל שטח יחסי ל fitness שלו.   
עכשיו נסובב את הגלגל ונמתין שיעצר. נבחר את ההורה שהגלגל נעצר בשטח שלו.  
  
ככל שה fitness של פתרון מסויים גדול יותר יחסית לזה של חבריו האחרים לאוכלוסיה – כך יגדל הסיכוי כי הוא ייבחר יותר פעמים להעמיד צאצאים. יחד עם זאת, טכניקה זו נותנת עדיין סיכוי גם לפרטים "פחות מוצלחים" באוכלוסיה להיבחר ולהעמיד צאצאים וכך נשיג, אולי, שונות רבה יותר בדור הבא

שימו לב כי בשיטה זו בהחלט ייתכן כי הורה מסויים ייבחר יותר מפעם אחת.

פרמטרים: אין

**שם: Tournament**

פרמטרים:

1. בורר טורנירים (Predefined tournament equalizer – pte). מספר בין 0 ל 1

בטכניקה זו בוחרים כמות הורים כגודל האוכלוסיה.  
נבחר 2 פתרונות בצורה רנדומלית. לכל פתרון יש את ה Fitness שלו.  
נגריל מספר בין 0 ל 1.

אם המספר גבוה מה PTE – נבחר את הפתרון עם ה fitness הגבוה יותר  
אם המספר קטן מה PTE – נבחר את הפתרון עם ה fitness הנמוך יותר

שימו לב כי בשיטה זו בהחלט ייתכן כי הורה מסויים ייבחר יותר מפעם אחת.

נספח ו' – הגדרות החוקים

במהלך התרגיל תתמכו בחוקים הבאים. לכל חוק יש שם ייחודי שיופיע בדיוק כך בתוך ה XML.

* 1. מורה לא מלמד פעמיים באותה שעה

**שם:** TeacherIsHuman

**תיאור:** מורה הוא רק בן אדם. לא ייתכן כי הוא מלמד באותה השעה והיום בכמה כיתות שונות

**פרמטרים:** אין

* 1. בשעה מסויימת מלמד לכל היותר מורה אחד

**שם**: Singularity

**תיאור**: בשעה ויום מסויימים בכיתה מסויימת יכול ללמד לכל היותר מורה אחד מקצוע אחד (או בכלל לא).

**פרמרטים**: אין

* 1. מורה מלמד רק את המקצועות שהוא מוסמך להם

**שם**: Knowledgeable

**תיאור**: מורה מלמד את רק את המקצועות שהוא מוסמך להם

**פרמטרים**: אין

* 1. כיתה מקבלת את כל שעות הלימוד שלה

**שם**: Satisfactory

**תיאור**: כיתה מקבלת בדיוק את מלאי שעות הלימוד שהיא צריכה עבור כל מקצוע ומקצוע

**פרמטרים**: אין

* 1. למורה יש יום חופשי

**שם**: DayOffTeacher

**תיאור**: למורה יש לפחות יום אחד בשבוע העבודה שבו הוא לא מלמד בכלל

**פרמטרים**: אין

* 1. מקצוע לא נלמד יותר ממספר שעות רצוף

**שם**: Sequentiality

**תיאור**: מקצוע לא נלמד יותר מכמות שעות רצוף. כמות השעות תינתן כפרמטר חיצוני.   
חישוב רצף השעות כמובן מדבר במסגרת יום מסויים (ולא חוצה ימים).

האמור מדבר כמובן על היביט הכיתה. (אם מורה מלמד ספרות 7 שעות רצוף ביום – זה בסדר).

**פרמטרים**: TotalHours - כמות השעות שאין ללמד את המקצוע בצורה רצופה יותר ממנה. מספר חיובי שלם.

* 1. לכיתה יש יום חופשי

**שם**: DayOffClass

**תיאור**: לכיתה יש לפחות יום חופשי אחד שבו היא לא לומדת כלל.

**פרמטרים**: אין

* 1. למורה יש העדפה על כמות שעות לימוד בשבוע עבודה

**שם**: WorkingHoursPreference

**תיאור**: לכל מורה מוגדר כמות שעות רצויה לעבודה במהלך שבוע העבודה כולו. החוק מקבל ציון מקסימלי כאשר המורה עובד בדיוק את כמות השעות הנדרשת. לא יותר ולא פחות.  
כמות השעות מוגדרת בקובץ ה XML כחלק מהמידע של כל מורה.

**פרמטרים**: אין

נספח ז' – 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.

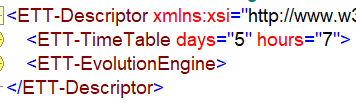
ביחנו היטיב את קבצי הדוגמא שהועלו למע' ה Mama וודאו כי אתם מבינים את פרטיהם ומבניהם.

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 ולא על רווחים בתחילתה/סופה

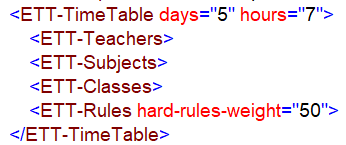
## סכמת קובץ XML – גרסה I

ראשית הקובץ מתארת מבנה של הגדרת בעיית מע' השעות כמו גם הגדרת מאפייני האלגוריתם האבולוציוני



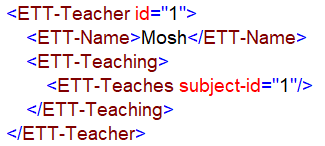
|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ETT-Descriptor | מתאר את ראשית קובץ ה XML |
| 2 | Element | ETT-TimeTable | מבנה המתאר בפנים את המאפיינים של בעיית מע' השעות |
| 3 | Element | ETT-EvolutionEngine | מבנה המתאר בפנים את המאפיינים של האלגוריתם האבולוציוני |

**ETT-TimeTable**



|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Attribute | days | מתאר את כמות ימי העבודה במע' השעות. מס' חיובי שלם |
| 2 | Attribute | hours | מתאר את כמות שעות העבודה בכל יום במע' השעות.  מס' חיובי שלם |
| 3 | Element | ETT-Teachers | מבנה המתאר את המידע על כל המורים המוגדרים במע'.  מכיל לפחות אלמנט אחד של ETT-Teacher המתאר מורה אחד במע' |
| 4 | Element | ETT-Subjects | מבנה המתאר את המידע על כל המקצועות המוגדרים במע' מכיל לפחות אלמנט אחד של ETT-Subject המתאר מקצוע אחד במע' |
| 5 | Element | ETT-Classes | מבנה המתאר את כל השכבות/כיתות המוגדרות במע' מכיל לפחות אלמנט אחד של ETT-Class המתאר כיתה/שכבה אחת במע' |
| 6 | Element | ETT-Rules | מבנה המתאר את כל החוקים/אילוצים המוגדרים במע' מכיל לפחות אלמנט אחד של ETT-Rule המתאר חוק/אילוץ אחד במע' |
| 7 | Attribute | hard-rules-weight | מספר המתאר את משקל ממוצע החוקים מסוג של Hard יחסית לחוקים של Soft. מספר שלם בין 0 ל 100 כולל |

**ETT-Teacher**

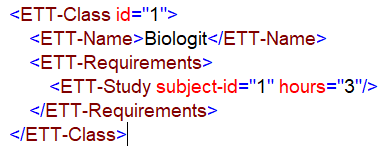
****

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ETT-Teacher | מתאר מורה אחד במע' |
| 2 | Attribute | id | מתאר מס' מזהה ייחודי של מורה. מס' חיובי שלם. |
| 3 | Element | ETT-Name | מתאר את השם של המורה. מחרוזת תווים באנגלית. יכולה להכיל רווחים |
| 4 | Element | ETT-Teaching | מבנה המתאר את המידע של כל המקצועות שמורה זה מוסמך ללמד. יכול להכיל אחד או יותר מופעים של אלמנט ETT-Teaches |
| 5 | Element | ETT-Teaches | אלמנט המתאר מקצוע אחד שמורה זה מוסמך ללמד |
| 6 | Attribute | subject-id | מתאר את מספרו הייחודי של המקצוע אותו מורה זה מוכשר ללמד |

**ETT-Subject**

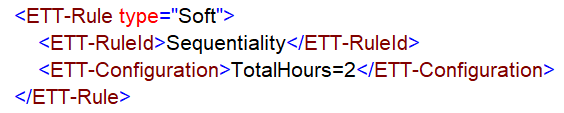
|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ETT-Subject | מתאר מקצוע אחד במע' |
| 2 | Attribute | id | מתאר מס' מזהה ייחודי של מקצוע. מס' חיובי שלם. |
| 3 | Attribute | name | מתאר את השם של המקצוע. מחרוזת תווים באנגלית. יכולה להכיל רווחים |

**ETT-Class**

****

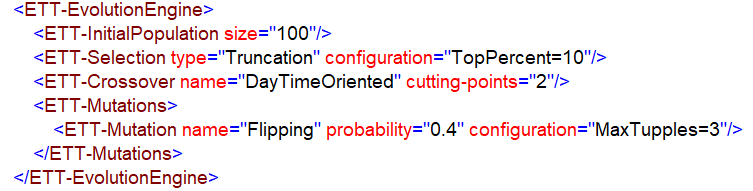
|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ETT-Class | מתאר כיתה/שכבה אחת במע' |
| 2 | Attribute | id | מתאר מס' מזהה ייחודי של כיתה/שכבה. מס' חיובי שלם. |
| 3 | Element | ETT-Name | מתאר את השם של הכיתה/שכבה. מחרוזת תווים באנגלית |
| 4 | Element | ETT-Requirements | מתאר את רשימת דרישות הלימוד של הכית/שכבה. יכיל אלמנט אחד או יותר של ETT-Study |
| 5 | Element | ETT-Study | מתאר דרישת לימוד אחת של מקצוע |
| 6 | Attribute | subject-id | מתאר הכוונה למס' המקצוע שנדרש ללמוד בכיתה/שכבה זו. מס' חיובי שלם |
| 7 | Attribute | hours | מתאר את כמות השעות שכיתה/שכבה זו אמורים ללמוד |

**ETT-Rule**

****

|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ETT-Rule | מתאר הגדרה של חוק אחד במע' |
| 2 | Attribute | type | מתאר את סוג החוק. ערכים אפשריים: Hard או Soft |
| 3 | Element | ETT-RuleId | מתאר את סוג החוק להפעלה. ערכים אפשריים:   * TeacherIsHuman * Singularity * Knowledgeable * Satisfactory |
| 4 | Element | ETT-Configuration | אלמנט אופציונלי שאינו חייב להופיע, המתאר קונפיגורציה ספציפית.  מכיל ערכים בפורמט של key=value.  אם יש יותר מערך אחד הם יופרדו באמצעות , |

**ETT-EvolutionEngine**

****

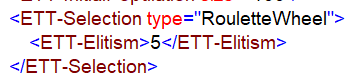
|  |  |  |  |
| --- | --- | --- | --- |
| # | סוג | שם | מהות |
| 1 | Element | ETT-EvolutionEngine | מתאר הגדרה של מאפייני האלגוריתם האבולוציוני להפעלה |
| 2 | Element | ETT-InitialPopulation | אלמנט המתאר את גודל האוכלוסיה הראשונית |
| 3 | Attribute | size | מתאר את גודל האוכלוסיה במהלך פעולת האלגוריתם. מס' חיובי שלם. |
| 4 | Element | ETT-Selection | מתאר את רכיב ה selection המופעל באלגוריתם האבולוציוני |
| 5 | Attribute | type | מתאר את סוג ה selection המונהג באלגוריתם זה. אחד מהערכים הבאים:   * Truncation * RouletteWheel |
| 6 | Element | ETT-Crossover | מתאר את ה crossover שייעשה בו שימוש במהלך הפעלת האלגוריתם |
| 7 | Attribute | name | מתאר את סוג ה crossover. אחד מהערכים הבאים:   * DayTimeOriented * AspectOriented |
| 8 | Attribute | cutting-points | מתאר את כמות נק' החיתוך לשימוש כחלק מה crossover. מס' חיובי שלם |
| 9 | Element | ETT-Mutations | אלמנט המתאר את כל ה mutation'ים שייעשה בהם שימוש במהלך ריצת האלגוריתם. מכיל אחד או יותר אלמנטים מסוג של ETT-Mutation |
| 10 | Element | ETT-Mutation | מתאר מוטציה אחת |
| 11 | Attribute | name | מתאר את סוג המוטציה. אחד מהערכים הבאים:   * Flipping * Sizer |
| 12 | Attribute | probability | מתאר את ההסתברות להפעלת המוטציה. מספר עשרוני בין 0 ל 1 כולל. |
| 13 | Attribute | configuration | מתאר קונפיגורציה ספציפית.  מכיל ערכים בפורמט של key=value.  אם יש יותר מערך אחד הם יופרדו באמצעות ,  אלמנט זה יופיע תמיד בהגדרה של mutation. אלמנט זה יכול להופיע (אך לא חייב) בהגדרה של selection ו crossover |

## 

## סכמת קובץ XML – גרסה II

עדכונים מגרסה קודמת:

1. לאלמנט ETT-RuleId התווספו 2 ערכים אפשריים נוספים:
   1. DayOffTeacher
   2. Sequentiality

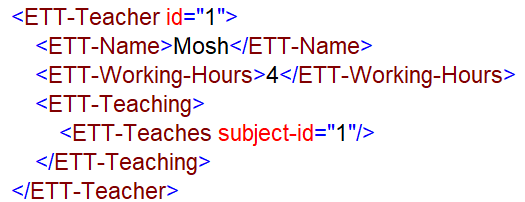


1. לאלמנט ETT-Selection נוסף תת אלמנט חדש: ETT-Elitism המכיל מס' חיובי שלם
2. לאלמנט ETT-Selection, לשדה type, נוסף ערך חוקי נוסף: **RouletteWheel**

## 

## סכמת קובץ XML – גרסה III

עדכונים מגרסה קודמת:

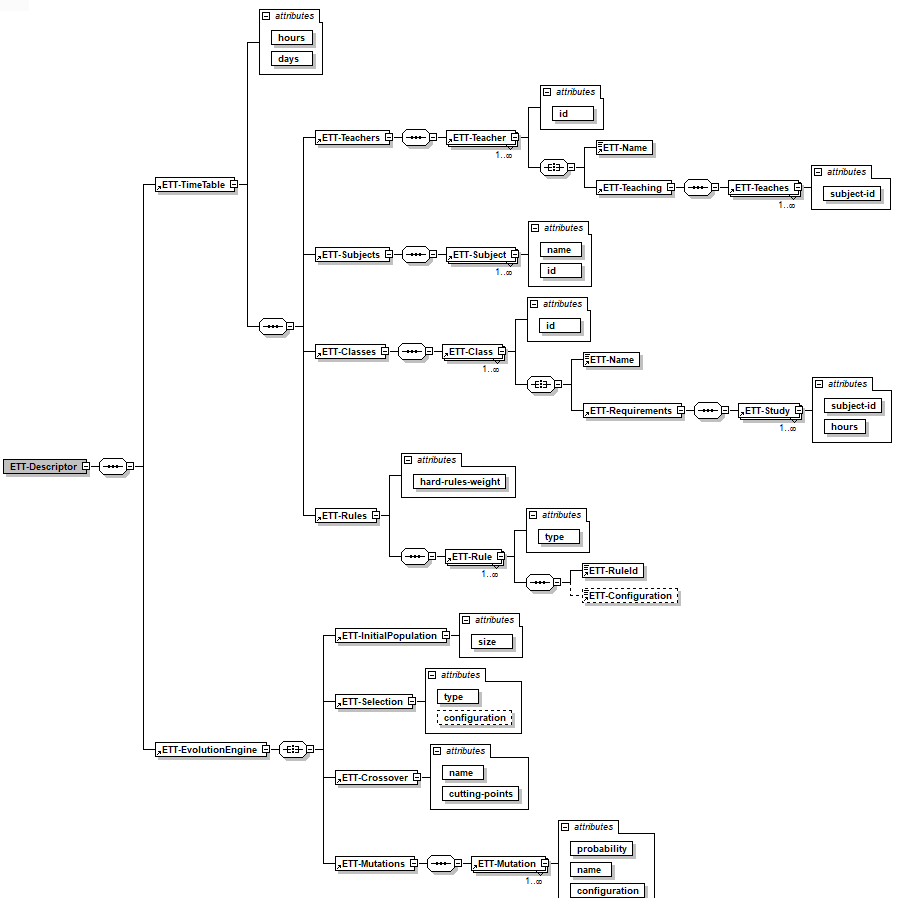
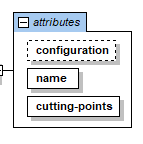
* + 1. **אין הגדרה של מאפייני האלגוריתם האבולוציוני (ETT-EvolutionEngine).
    2. הגדרת מורה הורחבה ומכילה אלמנט ETT-Working-Hours המכיל מס' חיובי שלם.

נספח ח' – קבצי סכמה לוידוא קובץ ה XML

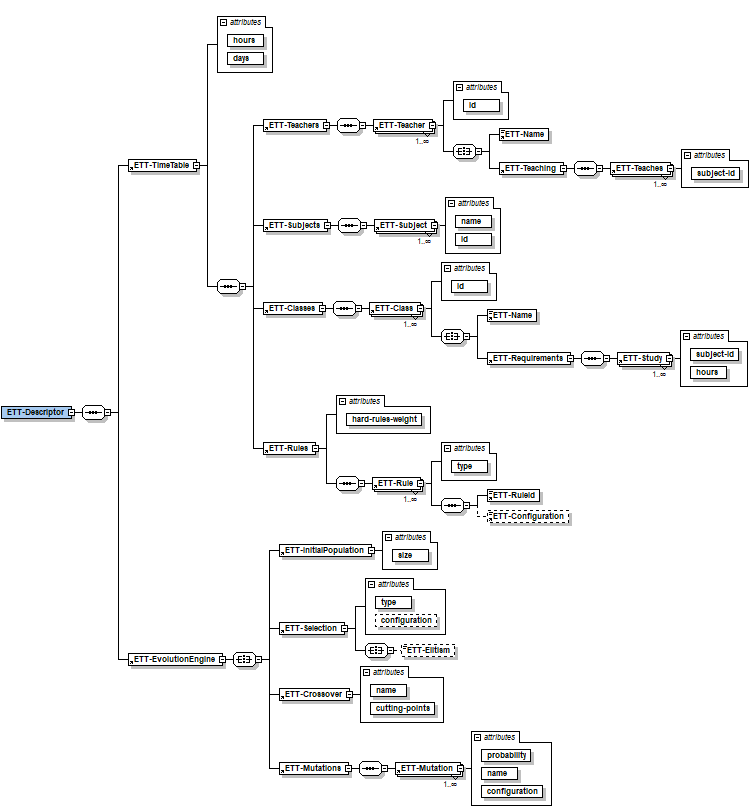
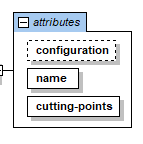
קובץ ה XML מקושר לסכמה שמגדירה את חלקיו השונים ואת חוקיותם.

להלן תרשימים המתארים את מבנה הסכמות בגרסאותיהם השונות:

### סכמה – גרסה 1



### סכמה – גרסה 2



### סכמה – גרסה 3

