236703 – תכנות מונחה עצמים, חורף 2024-25

תרגיל בית 2 – היכרות עם Java

כללי

1. מועד ההגשה : **17/12/2022** בשעה **23:55**
2. מטרת התרגיל היא הכרות עם Java, עבודה עם Exceptions, Collections, Comparable/Comparator, Streams, Iterable/Iterator ומימוש מנשקים.
3. קראו היטב את ההוראות, במסמך זה ובקוד שניתן לכם.
4. אחראי על התרגיל: **ג'וליאן**. שאלות על התרגיל יש לשלוח במייל julianmour@campus.technion.ac.il עם הנושא: "236703 HW2".
5. הקפידו על קוד ברור, קריא ומתועד ברמה סבירה. עליכם לתעד כל חלק שאינו טריוויאלי בקוד שלכם.
6. מהירות ביצוע אינה נושא מרכזי בתרגילי הבית בקורס. בכל מקרה של התלבטות בין פשטות לבין ביצועים, העדיפו את המימוש הפשוט.
7. הימנעו משכפול קוד והשתמשו במידת האפשר בקוד שכבר מימשתם.
8. **כדי להימנע מטעויות, אנא עיינו ברשימת ה- FAQ המתפרסמת באתר באופן שוטף.**

מבוא

בתרגיל זה תצטרכו לממש מערכת לניהול שירים ומשתמשים – TechnionTunes.

חלק א'

בחלק זה נממש את הבסיס של המערכת: משתמשים ושירים. לכל משתמש ושיר יהיה מזהה ייחודי, שיקרא id או מספר זהות במסמך זה.

`UserImpl`

מחלקה זו תממש את ההתנהגות של המנשק User אשר סופק לכם. מחלקה זו מייצגת משתמש בודד.

באופן כללי, מלבד מאפייני המשתמש, המשתמש גם 'מחזיק' קבוצה של השירים שאותם דירג (כולל דירוג כמובן)

המנשק User מרחיב את המנשק Comparable<User> המאפשר השוואה בין אובייקטים בעזרת המתודה compareTo. אופי המימוש מתואר בהמשך.

המחלקה תספק את ההתנהגות הבאה:

- `UserImpl(int userID , String userName , int userAge)` - בנאי שמקבל את ה- ID , שם וגיל (שנים) של משתמש חדש. אפשר להניח שמתקבלים פרמטרים חוקיים:
  - `userAge >= 0`
  - `userID >= 0`
  - `username != null`

- `getID()` – מחזירה את מספר ה- ID של המשתמש.

- `getName()` – מחזירה את שם המשתמש.

- `getAge()` – מחזירה את גיל המשתמש (בשנים).

- `rateSong(Song song, int rate)` – המשתמש מדרג שיר כלשהו.
  - אם `rate` **אינו** בתחום [0,10] – תזרק החריגה `IllegalRateValue`.
  - אם השיר כבר דורג על ידי המשתמש – תזרק החריגה `SongAlreadyRated`.
  - **אסור** למתודה זו לערוך את `song`.

- `getAverageRating()` – מחזירה את הדירוג הממוצע של המשתמש (ממוצע על הדירוגים שנתן לשירים השונים). אם המשתמש עדיין לא דירג אף שיר - המתודה תחזיר 0.

- `getPlaylistLength()` – מחזירה את האורך הכולל (בשניות) של כל השירים שהמשתמש דירג. אם המשתמש עדיין לא דירג אף שיר - המתודה תחזיר 0.

- `getRatedSongs()` – מחזירה collection של השירים שהמשתמש דירג, **ממוין** לפי:
  1 – הדירוג שהמשתמש נתן לשיר בסדר יורד
  2 – אורך השיר בסדר עולה
  3 – מספר זהות בסדר יורד

- `getFavoriteSongs()` – מחזירה collection של כל השירים שהמשתמש דירג בציון של 8 ומעלה (כולל 8) <u>ממוינים</u> לפי מספרי הזהות שלהם בסדר עולה.

- `AddFriend(User friend)` - מוסיפה קשר חברות חדש בין שני המשתמשים.
  - אם שני המשתמש הנוכחי כבר חבר של `friend` - תזרק החריגה `AlreadyFriends` .
  - קשר "חברות עצמי" אינו חוקי - תזרק חריגה `SamePerson` .

  <u>הערה</u>: פעולה זו מוסיפה קשר חברות **חד כיווני**.

- `favoriteSongInCommon(User user)` – מחזירה `true` <u>אם ורק אם</u> שני המשתמשים חברים ויש לפחות שיר אחד משותף שהם מעדיפים (דירוג 8 ומעלה) .

- `getFriends()` – מחזירה map בו כל מפתח הוא חבר וערך המפתח הוא מספר השירים השונים שאותו חבר דירג.

<u>הערה :</u> User מממש את Comparable<User> ולכן אין בעיה שהמפתחות יהיו מסוג User.

- equals(Object o) - עליכם לדרוס את המתודה equals שהוגדרה לראשונה ב- Object, כפי שנלמד בתרגול. משתמשים יחשבו זהים אם ורק אם יש להם אותו מספר מזהה.

- compareTo(User user) – ההשוואה בין שני משתמשים תתבצע לפי ההשוואה בין המספרים המזהים שלהם.

SongImpl

מחלקה זו תממש את ההתנהגות של המנשק Song אשר סופק לכם. מחלקה זו מייצגת שיר בודד.

בנוסף, המנשק Song מרחיב את המנשק Comparable<Song> המאפשר השוואה בין אובייקטים בעזרת המתודה compareTo. אופי המימוש מתואר בהמשך. המחלקה תספק את ההתנהגות הבאה:

- SongImpl(int songID , String songName , int length , String singerName) - בנאי שמקבל את ה- ID, שם, אורך (שניות) ושם הנגן של שיר חדש. ניתן להניח שמתקבלים פרמטרים חוקיים:
  - songID >=0
  - length >=0
  - songName != null
  - singerName != null

- getID() – מחזירה את מספר הזהות של השיר.

- getName() – מחזירה את שם השיר.

- getLength() – מחזירה את אורך השיר (בשניות).

- getSingerName() – מחזירה את שם הנגן.

- rateSong(User user, int rate) - המשתמש מדרג שיר כלשהוא .
  - אם rate **אינו** בתחום [0,10] – תזרק החריגה IllegalRateValue.
  - אם המשתמש כבר דירג מקודם את song – תזרק החריגה SongAlreadyRated.
  - **אסור** למתודה זו לערוך את user.

- getRaters() – מחזירה Collection של משתמשים (User) שדירגו את השיר, **ממוין** באופן הבא:
  1- לפי דירוג בסדר יורד.
  2- לפי גיל בסדר עולה.
  3- לפי מספר זהות בסדר יורד.

- getRatings() – מחזירה map בו המפתחות הם הדירוגים השונים ששיר זה קיבל והערך של כל מפתח הוא set שמכיל את כל המשתמשים שנתנו דירוג זה לשיר הנ"ל (ללא חשיבות לסדר).

- getAverageRating() – מחזירה את הדירוג הממוצע ששיר זה קיבל. אם השיר עדיין לא דורג - המתודה תחזיר 0.

- equals(Object o) - עליכם לדרוס את המתודה equals שהוגדרה לראשונה ב- Object, כפי שנלמד בתרגול. שני שירים יחשבו שווים אמ"מ יש להם אותו מספר מזהה.

- compareTo(Song song) - ההשוואה בין שני שירים מתבצעת לפי ההשוואה בין המספרים המזהים שלהם .

<u>הערה</u>**:** עליכם לדרוס את המתודה compareTo המוגדרת במנשק Comparable כפי שראיתם בכיתה. המתודה מגדירה יחס סדר בין המשתמש/השיר הנוכחי למשתמש/שיר שהתקבל כפרמטר. עליה להחזיר ערך שלילי אם המשתמש/השיר הנוכחי בעלת מזהה קטן משל המשתמש/השיר שהתקבל, חיובי אם להיפך, ו- 0 אם המזהים הם שווים.

חלק ב' - מימוש המנשק TechnionTunes

בחלק זה נממש את המערכת הכוללת. במערכת נוכל להוסיף משתמשים, שירים, להגדיר חברויות בין משתמשים ולקבל מידע על קשרים בין המשתמשים ועל השירים העדיפים עליהם. שימו לב שקשר חברות בין משתמשים במערכת **<u>יהיה</u> קשר סימטרי**, כלומר אם A חבר של B, אז גם B חבר של A. קשר זה איננו רפלקסיבי, כלומר אדם אינו יכול להיות חבר של עצמו (וכמובן שאיננו טרנזיטיבי).

ממשק זה מרחיב את הממשק Iterable<Song> שמספק לנו דרך לעבור על כל השירים במערכת בסדר הבא:

1- לפי אורך בסדר עולה.

2- לפי מספר זהות בסדר עולה.

<u>הערה</u>: ניתן להניח שבמהלך המעבר לא יתווספו שירים חדשים למערכת.

בהמשך נראה שהמערכת מדמה גרף של חברויות בו כל צומת הוא משתמש, והקשתות הן קשרי החברות בין המשתמשים. אנו נשתמש בעובדה זו בהמשך.

TechnionTunesImpl

המחלקה תספק את ההתנהגות הבאה:

- TechnionTunesImpl() – מאתחלת מערכת ריקה .

- AddUser(int userID , String userName , int userAge) – מוסיפה משתמש חדש למערכת.
  - אם המשתמש כבר קיים - תזרק החריגה UserAlreadyExists .

- ניתן להניח שמתקבלים פרמטרים חוקיים:
  - `userID >= 0`
  - `username != null`
  - `userAge >= 0`

- `getUser(int id)` – מחזירה את המשתמש בעל מספר זהות id (האובייקט עצמו ("רפרנס") ולא העתק שלו).
  - אם אין משתמש בעל מספר זהות זהה - תזרק החריגה `UserDoesntExist` .

- `makeFriends(int id1,int id2)` – מוסיפה קשר חברות בין שני המשתמשים בעלי מספרי זהות שהתקבלו.
  להזכירכם, קשר זה יהיה סימטרי.
  - אם שני המשתמשים כבר חברים - תזרק החריגה `AlreadyFriends`.
  - אם התקבל מספר זהות שלא מתאים לאף משתמש במערכת – תזרק החריגה `UserDoesntExist`.
  - קשר "חברות עצמי" אינו חוקי - תזרק החריגה `SamePerson`.

- `addSong(int SongID , String songName , int length ,String SingerName)` – המתודה מקבלת את ה- ID, שם, אורך (שניות) ושם הנגן של שיר חדש ותוסיף אותו למערכת.
  - אם קיים במערכת שיר זהה במערכת - תזרק החריגה `SongAlreadyExists` .
  - אפשר להניח שמתקבלים פרמטרים חוקיים:
    - `songID >=0`
    - `length >=0`
    - `songName != null`
    - `singerName != null`

- `getSong(int id)` – מחזירה את השיר בעל מספר זהות id (האובייקט עצמו ("רפרנס") ולא העתק שלו).
  - אם לא קיים שיר בעל מספר זהות זה במערכת - תזרק החריגה `SongDoesntExist` .

- `rateSong(int userId, int songId, int rate)` – משתמש בעל מספר זהות `userId` מדרג את השיר בעל מספר זהות `songId`. במקרים לא חוקיים, יש לזרוק חריגות בסדר הבא:
  - לזרוק `UserDoesntExist` אם המשתמש לא קיים.
  - לזרוק `SongDoesntExist` אם השיר לא קיים.
  - לזרוק `IllegalRateValue` אם ערך הדירוג לא חוקי.
  - לזרוק `SongAlreadyRated` אם השיר כבר דורג.

- sortSongs(Comparator<Song> comp) – מחזירה collection של כל השירים במערכת ממוינים לפי comp.

- getIntersection(int IDs[]) – מחזירה set המכיל את כל השירים <u>שדורגו</u> על ידי <u>כל</u> המשתמשים המתאימים למספרי הזהות שהתקבלו.
  - אם התקבל מספר זהות שלא מתאים לאף משתמש במערכת – תזרק החריגה UserDoesntExist .

- getHighestRatedSongs(int num) – מחזירה collection של לכל היותר num שירים שממוין באופן הבא:
  1- לפי דירוג ממוצע בסדר יורד.
  2- עדיפות שנייה (עבור דירוג ממוצע זהה): לפי אורך בסדר יורד.
  3- עדיפות שלישית (עבור דירוג ממוצע ואורך זהים): לפי מספר זהות בסדר עולה.

- getMostRatedSongs(int num) - מחזירה collection של לכל היותר num שירים שממוין באופן הבא:
  1- לפי מספר הדירוגים ששיר קיבל בסדר יורד.
  2- עדיפות שנייה: לפי אורך בסדר עולה.
  3- עדיפות שלישית: לפי מספר זהות בסדר יורד.

- getTopLikers(int num) - מחזירה collection של לכל היותר num משתמשים שממוין באופן הבא:
  1- לפי דירוג ממוצע שמשתמש נתן בסדר יורד.
  2- עדיפות שנייה: לפי גיל בסדר יורד.
  3- עדיפות שלישית: לפי מספר זהות בסדר עולה.

- canGetAlong(int userId1,int userId2) – מתודה זו מחזירה האם שני משתמשים מסתדרים אחד עם השני.
  שני משתמשים מוגדרים כ- *מסתדרים אחד עם השני* באופן הבא:
  1- כל משתמש מסתדר עם עצמו.
  2- שני משתמשים מסתדרים אחד עם השני אם קיים מסלול בגרף החבריות של המערכת שבו לכל שני משתמשים (צמתים) עוקבים קיים לפחות שיר אחד מועדף.

שימו לב:
- אם אחד המשתמשים לא קיים במערכת - תזרק החריגה UserDoesntExist.
- מספר קשרי החברויות בין משתמש לעצמו מוגדר להיות 0.
- יש להיזהר מקריאות רקורסיביות עמוקות בלתי נשלטות (לולאות אינסופיות)
- רמז נוסף: חשבו על הדרכים למעבר על גרפים שלמדתם בקורס אלגוריתמים.

<u>הערה:</u> כאשר עובדים עם הממשק TechnionTunes **ניתן להניח** שהמשתמשים/שירים לא ישתנו על ידי גורם חיצוני, כלומר רק TechnionTunes קוראת ל- rateSong.

טיפול בשגיאות

על מנת לפתור את התרגיל, תצטרכו להשתמש ב- Java Exceptions. נושא זה יילמד לעומק בהמשך הקורס, לצורך תרגיל זה עליכם להשתמש בתכונות בסיסיות בלבד. בדומה ל ++C, על מנת להצהיר על קטע קוד שעלול לכלול שגיאה בתוכנית יש לעטוף אותו בבלוק **try**, וקטע טיפול בשגיאה בבלוק **catch**. בניגוד ל- ++C, ניתן "לזרוק" אך ורק אובייקטים מטיפוס היורש (ישירות או בעקיפין) מהטיפוס Throwable. ישנם כמה סוגים של טיפוסי חריגות, כאשר בתרגיל זה עליכם להשתמש אך ורק ב- checked exceptions. בחריגות מסוג זה, מתודה המכילה קטע קוד אשר עלול לייצר חריגה, חייבת להצהיר על כך. ההצהרה היא חלק בלתי נפרד מהחתימה של המתודה.
לדוגמה:

```
class AwesomeException extends Exception { … }

class A {
      public void f() throws AwesomeException { … }
}
```

המתודה f, מצהירה על כך שהיא עלולה לייצר חריגת AwesomeException בעת ריצתה.

על מנת לזרוק חריגה מהטיפוס AwesomeException יש להשתמש במילה השמורה **throw**:

```
if (x < 0) {
    throw new AwesomeException();
}
```

הנחיות ורמזים למימוש

- סדר זריקת השגיאות הוא לפי סדר ההצהרה שלהם ב-method.
- בכל פעם שרוצים להחזיר collection/map/set של משתמשים/שירים/מספרים אבל אין איברים שמקיימים את תנאי הסעיף אתם אמורים להחזיר מבנה (collection/map/set) ריק .
- מומלץ להשתמש בתכונות החדשות של Java 8 בפרט $\lambda - expr$, streams.
- ניתן להוסיף מחלקות עזר ו/או מחלקות אבסטרקטיות (במידת הצורך).
- ניתן להניח שבעת איטרציה על האוספים, לא יתווספו איברים חדשים לאוסף. כמו כן, ניתן להניח שלא ישתנה האוסף שעליו עוברים.
- ניתן להניח שבזמן בדיקת המערכת, יתווספו קשרי חברות רק באמצעות קריאה makeFriends, ולא ישירות דרך מופע של User.
- לגבי החזרת עותק של אוסף - מדובר על shallow copy. כלומר, שינוי האוסף המוחזר לא יגרום לשינוי המערכת. שינוי עצמים מתוך האוסף המוחזר לא ייבדקו, ולכן <u>אין</u> דרישה לממש deep copy.

  - רמז: כדי לממש את שתי המתודות הללו, מומלץ להיזכר במנשק Comparator<T> שלמדתם בתרגול, ולהשתמש במחלקות המממשות את המנשק. במנשק יש מתודה אחת שחובה לממש - compareTo. לחילופין, נסו לשלב lambda expression במימוש זה. היעזרו בפעולות filter ו- sorted של שטפים (Streams).

בדיקות אוטומטיות ע"י JUnit

עם התרגיל סופקה לכם מחלקת בדיקות הנקראת Example, המשתמשת בספרייה JUnit. אנו ממליצים להפעיל את הבדיקות האלו על הפתרון שלכם, ולכתוב בדיקות נוספות באמצעות ספרייה זו כדי להקל על מלאכת הבדיקה.

הצוות ממליץ לכתוב את הבדיקות לפני המימוש של כל מודול.
זוהי המלצה בלבד וכך או כך אין להגיש בדיקות.

לנוחיותכם, המצגת מסדנת ה- IDE + git, נמצאת באתר הקורס, ובה הסבר מפורט על התחלה עם JUnit ב IntelliJ.

דרישות והערות כלליות

1. אין לשנות את הקבצים המצורפים (של package OOP.Provided). הבודק האוטומטי דורס את הקבצים ע"י הגרסה המצורפת.
2. עליכם לוודא שהמתודה `equals` עונה על החוזה שלה כפי שנלמד בתרגול עבור המחלקות בהן נדרשתם להגדיר אותה.
3. יש לתעד את כל המחלקות ואת כל המתודות בפתרון באופן סביר (כל דבר שאינו מובן מאליו יש לתעד).
4. כל המחלקות שלכם צריכות להיות ממומשות ב-package OOP.Solution.
5. אין להשתמש בספריות חיצוניות לצורך הפתרון. ניתן להשתמש במחלקות מתוך java.util.
6. אין להדפיס לערוץ פלט הסטנדרטי או לערוץ השגיאות הסטנדרטי. אם אתם משתמשים בפלט לצורך בדיקות, הקפידו להסיר את ההדפסות לפני ההגשה.
7. הקפידו להסיר שורות ייבוא לקבצים או ספריות שאינם חלק מהקוד שניתן לכם או שהנכם משתמשים בו. (למשל ייבוא לקבצי בדיקות).
8. לתרגיל מצורף קובץ בשם Example.java שמכיל דוגמה להרצה של המערכת. חובה לוודא שה-test שבקובץ מתקמפל ועובר עם ההגשה (אם הוא לא, אז בסיכוי גבוה ה-test-ים הרשמיים לא יעברו). אין לצרף את Example.java להגשה.

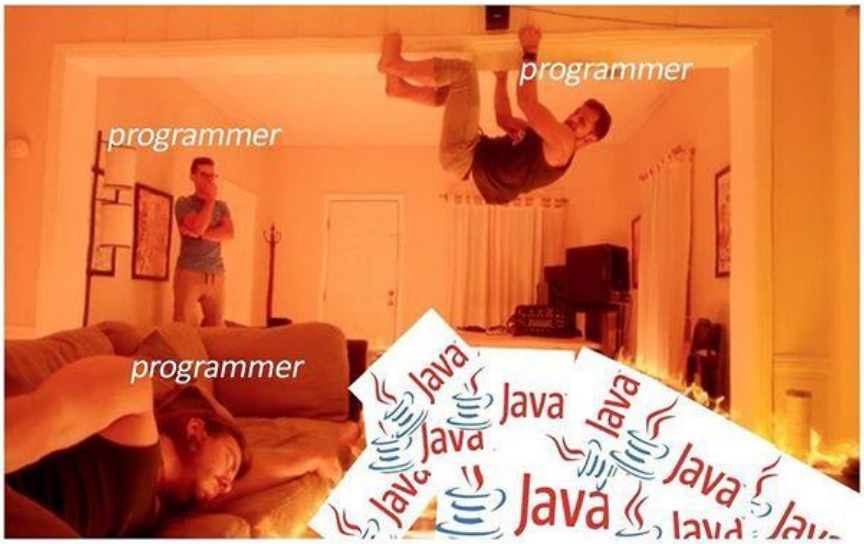
הוראות הגשה

- בקשות לדחייה יש לשלוח למתרגל האחראי על הקורס (**ג'וליאן**) בלבד, עם הכותרת "**236703 HW2**". מכיוון שבקורס מדיניות איחורים - ראו מידע באתר - דחיות יאושרו רק מסיבות לא צפויות או לא נשלטות (כמו מילואים).
- יש להגיש קובץ בשם $OOP2\_ID1\_ID2.zip$ המכיל:
  - קובץ בשם readme.txt בפורמט הבא:

Name1 id1 email1
Name2 id2 email2

  - על ה-zip להכיל את כל קבצי הקוד שכתבתם לצורך התרגיל.
- הימנעו משימוש בתיקיות בתוך ה-zip ומהגשת קבצים שבחבילות אחרות.
- אין להגיש את הקבצים המצורפים לתרגיל (מוץ מאלה של package OOP.Solution כמובן) ואין להגיש test-ים.
- הגשה שלא לפי ההוראות תגרור הורדת ציון בהתאם.

THE FLOOR IS JAVA



בהצלחה!