

תוכנה 1 – אביב תשע"ו

תרגיל מספר 5

הנחיות כלליות:

- קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.
- הגשת התרגיל תעשה במערכת ה-moodle בלבד (<http://moodle.tau.ac.il/>).
- יש להגיש קובץ zip יחיד הנושא את שם המשתמש ומספר התרגיל (לדוגמא, עבור המשתמש aviv יקרא הקובץ aviv_hw5.zip). קובץ ה-zip יכיל:
 - א. קובץ פרטים אישיים בשם details.txt המכיל את שמכם ומספר ת.ז.
 - ב. קבצי ה-java של התוכניות אותם התבקשתם לממש.

הגשת מחלקה עם חבילות: יש לכווץ בתוך קובץ ה-zip שאתם מגישים את כל היררכיית התיקיות מתחת ל-src באקליפס. למשל, כדי להגיש את המחלקה sw1.pac.MyClass העתיקו את התיקיה sw1 שמתחת ל-src כולל כל מה שבתוכה לתוך קובץ ה-zip.

בתרגיל זה נעסוק במערכת של המלצות על סרטים ונממש שיטה מאוד טריוויאלית למתן המלצות על סרטים על סמך של דירוגים של משתמשים שונים במערכת. Amzone ו Netflix הן שתי מערכות גדולות אשר משתמשות בדירוגי משתמשים לצורך מתן המלצות על התכנים שהן מציעות. כמובן שהאלגוריתמים שבהם מערכות אלא משתמשות הם יותר מתוחכמים ממה שאנחנו ניראה.

מידע של סרטים, משתמשים ודירוגים

באתר הקורס ניתן למצוא שלושה קבצי טקסט (עם הסיימות dat) אשר מכילים מידע על סרטים, משתמשים והדירוגים שלהם. קבצי מידע אלה נוצרו מתוך בסיס הנתונים MovieLens שנוצר על ידי קבוצת מחקר מאוניברסיטת מיניסוטה (Department of Computer Science and Engineering at the University of Minnesota). בסיס הנתונים הזה זמין בחינם למטרות מחקר, וניתן הוריד אותו בשלמותו מהקישור הבא:

<http://grouplens.org/datasets/movielens>

ניתן לצפות בתוכנו של כל קובץ באמצעות כל עורך טקסט.

בתרגיל זה נשתמש בשלוש מחלקות: Movie, User, MoviesRecommendations. מחלקות אלה נמצאות ב il.ac.tau.cs.sw1.ex5. שימו לב – כמו בתרגיל 4, אין להשתמש במבני נתונים גנריים כמו Set, Map, List וכו'.

המימוש של המחלקה User נתון בחלקו, עליכם להשלים אותו וכן להשלים את מימושו של שתי המחלקות האחרות. כמו כן, ניתן ואף כדאי להוסיף מתודות עזר על פי הצורך.

מערכת המלצה על סרטים

(1) [10 נקודות] ממשו את המתודה הסטטית `getUserFromString` המקבלת מחרוזת בפורמט הבא:

`id::gender::ageGroup::occupation::zipCode`

ומחזירה אובייקט `User` ע"פ הנתונים במחרוזת.

בין כל שני שדות מפרידים שני סימני נקודותיים. לשדה `gender` יש שני ערכים אפשריים: `M` ו-`F`. שאר השדות מכילים ספרות.

דוגמא למחרוזת קלט אפשרית:

`1::F::1::10::48067`

הניחו כי פונקציה זו תקבל קלט חוקי בלבד, כלומר, בפורמט המתואר.

חתימת המתודה:

```
private static User getUserFromString(String userString)
```

(2) [5 נקודות] ממשו את המתודה הסטטית `loadUserData` אשר מקבלת כפרמטר שם של קובץ המכיל נתוני משתמשים ומחזירה מערך של `Users` אשר נוצרו על סמך הנתונים אלה. כל שורה בקובץ מתאימה לפורמט שהוגדר בסעיף הקודם, כך שעליכם להשתמש במתודה `getUserFromString` על מנת לייצר את המשתמשים.

המערך שיווצר יהיה מערך שבו בתא ה-`i` ישמר המשתמש ש-`id` שלו הוא `i`. כלומר, האינדקס האחרון במערך יהיה מספר ה-`id` הגדול ביותר בקובץ. באינדקס עבורו לא קיים משתמש עם אותו ה-`id` (למשל, האינדקס 0) ישמר הערך `null`.

הנחות נוספות:

- ניתן להניח שכל מספר `id` מופיע פעם אחת בלבד בקובץ הקלט.
- ניתן להניח שהקובץ ממויין לפי `id` (לא חיוני למימוש).
- ניתן להניח שלא יהיה מספר `id` הגדול מ-10,000.
- המתודה מצהירה של זריקת חריג (שגיאה) בגלל העבודה עם קבצים.

חתימת המתודה:

```
public static User[] loadUserData(String fileName) throws Exception
```

(3) [5 נק'] השלימו את המימוש של המחלקה `Movie` על פי המפרט הבא:

משמעות	שדה
מזהה יחודי של הסרט	<code>int movieId</code>
שם הסרט	<code>String title</code>
שנת יציאת הסרט	<code>int year</code>
רשימת קטגוריות אליהן הסרט משתייך.	<code>String[] categories</code>

הנכם רשאים להוסיף שדות נוספים לפי ראות עיניכם.

הגדירו את בנאי המחלקה על פי החתימות הבאות:

```
public Movie(int id, String title, int year, String[] categories)
```

ממשו גם getter-ים (מתודות get) עבור שדות של המחלקה.

ניתן להעזר ב Source->Generate Getters and Setter על מנת לייצר את הקוד של ה getter-ים. לנוחותכם, ממשו את המתודה toString עבור המחלקה Movie. מימוש זה אינו חובה ואנחנו לא מגדירים פורמט אחיד לייצוג האובייקט באמצעות מחרוזת.

(4) [5 נק'] ממשו את מתודת המופע belongsToCategory המקבלת מחרוזת המייצגת קטגוריה ומחזירה true אם הסרט משתייך לקטגוריה זו ו false אחרת. עליכם לבצע את הבדיקה ללא חשיבות ל case (כלומר, הקטגוריות Fiction ו fiction הן למעשה אותה הקטגוריה).

חתימת המתודה:

```
public boolean belongsToCategory(String category)
```

(5) [10 נקודות] ממשו את המתודה הסטטית getMovieFromString המקבלת מחרוזת בפורמט הבא:

```
id::title::categories
```

ומחזירה אובייקט מטיפוס Movie הנוצר על פי הנתונים במחרוזת. בין כל שני שדות, כמו עבור משתמשים, מפרידים שני סימני נקודותיים. השדה id הוא שדה מספרי.

השדה title מכיל את שם הסרט (שם הסרט יכול להכיל אותיות, מספרים, סימני פיסוק וכו'). בסוף המחרוזת מופיעה בסוגריים שנת יציאת הסרט.

השדה categories מכיל את הקטגוריות אליהן משתייך הסרט. בין כל שתי קטגוריות מפריד התו " | " דוגמא למחרוזת קלט:

```
1::Toy Story (1995)::Animation|Children's|Comedy
```

הנחות והערות:

- הפונקציה מקבלת קלט בפורמט חוקי, ובפרט, כל סרט ישתייך לפחות לקטגוריה אחת.
 - על מנת לפצל ולהחליף מחרוזות על פי סוגריים והסימן " | " השתמשו ב \ \ לפני תווים אלה.
- חתימת המתודה:

```
public static Movie getMovieFromString(String movieStr)
```

(6) [5 נקודות] ממשו את המתודה הסטטית loadMoviesData אשר מקבלת כפרמטר שם של קובץ המכיל נתוני סרטים ומחזירה מערך של איברים מטיפוס Movie אשר נוצרו על פי הנתונים אלה. הנחות והנחיות זהות לאלה בסעיף (2), עבור המתודה המקבילה לטעינת משתמשים.

חתימת המתודה:

```
public static Movie[] loadMoviesData(String fileName) throws Exception
```

(7) [5 נקודות] כעת, נעבור למחלקה MoviesRecommendation.

השלימו את הבנאי של המחלקה MoviesRecommendation כך שיאתחל את השדות הבאים:

שדה	משמעות
Movie[] movies	מערך הסרטים במערכת
User[] users	מערך המשתמשים במערכת
int[][] ratings	טבלת דירוגים של משתמשים לספרים.

האיבר ratings[i][j] מכיל את הדירוג של היזר ה i לסרט ה j.
--

חתימת הבנאי:

```
public MoviesRecommendations(User[] users, Movie[] movies, int[][] ratings)
```

(8) [5 נקודות] ממשו את המתודה הסטטית updateSingleRating המקבלת מחרוזת בפורמט הבא:
 userID::movieID::rating::timestamp

בין כל שני שדות מפרידים שני סימני נקודותיים (::). כל השדות מכילים ערכים מספריים בלבד. הערך של timestamp אשר מופיע מחרוזת לא ישמש לשום דבר ולכן עליכם להתעלם ממנו.

בנוסף למחרוזת זו, המתודה מקבלת שלושה פרמטרים נוספים – מערך של משתמשים (User), מערך של סרטים (Movie) ומערך דו מימדי של מספרים שלמים (int) אשר בו נעדכן את הדירוג הנוכחי.

עליכם לטפל במקרה שבו המחרוזת מכילה מידע לא תקין – מספר סרט או מספר משתמש אשר לא קיים. לצורך כך, מועברים המערכים שמכילים את כל המשתמשים והסרטים. במידה והדירוג נעשה עבור סרט או משתמש לא חוקי, עליכם להתעלם ממנו ולא לעדכן את מערך הדירוגים. במקרה זה, המתודה תחזיר את הערך false. אחרת, אם העדכון הצליח, המתודה תחזיר את הערך true.

הניחו כי מימדי מערך הדירוגים תואמים למערכי המשתמשים והסרטים.

חתימת המתודה:

```
private static boolean updateSingleRating(String ratingsString, int[][] ratingsTable, User[] users, Movie[] movies)
```

(9) [5 נקודות] ממשו את המתודה הסטטית loadRatingsData אשר מקבלת שלושה פרמטרים: שם קובץ, מערך של משתמשים (User) ומערך של סרטים (Movie) ומחזירה מערך דו מימדי אשר מכיל את דירוגי הסרטים של כל המשתמשים. גודל המערך המוחזר יגזר מאורכי מערכי המשתמשים והסרטים. השתמשו ב updateSingleRating על מנת לעדכן את מערך הדירוגים.

הנחות:

- יתכנו מספרי משתמשים/סרטים אשר לא מופיעים ברשימת המשתמשים/סרטים. הטיפול צריך להעשות במתודה שמומשה בסעיף 8 על פי ההנחיות.
- עבור כל זוג של משתמש/סרט תופיע שורת דירוג יחידה.
- הקובץ אינו בהכרח ממויין (לא אמור להשפיע על המימוש).

חתימת המתודה:

```
public static int[][] loadRatingsData(String fileName, User[] usersArray, Movie[] moviesArray)
```

(10) [5 נקודות] ממשו את מתודת המופע getAverageRatingForUser אשר מקבלת מספר משתמש (id) ומחשבת את הדירוג הממוצע שלו עבור הסרטים אותם דירג.

עליכם לוודא שמספר המשתמש מתאים למשתמש חוקי. במידה והמשתמש לא דרג אף סרט או שהמשתמש לא קיים, החזירו את הדירוג NO_RATING (ראו הגדרת הקבוע בראש המחלקה)

חתימת המתודה:

```
public double getAverageRatingForUser(int userId)
```

(11) [10 נקודות] ממשו את המתודה `getAverageRatingForMovie` אשר מקבלת מספר מזהה של סרט (`id`), קבוצת גיל (מספר שלם) ועיסוק (מספר שלם) ומחשבת את הדירוג הממוצע שלו בקרב כל המשתמשים אשר שייכים לאותה קבוצת הגיל והעיסוק. אם הועבר הערך `NO_FILTER` עבור פרמטר של קבוצת גיל ו/או עיסוק, לא נבצע סינון לפי פרמטר זה. כלומר, אם נרצה לקבל את הדירוג הממוצע של הסרט על פני כל המשתמשים, נעביר את הקבוע `NO_FILTER` עבור שני הפרמטרים.

במידה ואף משתמש בקבוצה שהוגדרה לא דירג ספר זה, או שלא קיים ספר עם `id` זה, היא תחזיר את הדירוג `NO_RATING`.

נשתמש בהעמסה בשביל לממש שתי אופציות: האופציה ראשונה מקבלת שלושה פרמטרים, האופציה השנייה מקבלת רק פרמטר יחיד, מזהה הסרט, ומחזירה את הדירוג הממוצע ללא סינון על פי קבוצת גיל או עיסוק. על מנת למנוע שכפול קוד, ממשו את אחת הפונקציות באמצעות קריאה לשניה.

```
public double getAverageRatingForMovie(int movieId, int ageGroup, int occupation)
```

```
public double getAverageRatingForMovie(int movieId)
```

(12) [10 נקודות] כעת נממש את המתודה `getAverageRatingByMovieCategory` אשר מקבלת את הפרמטר `category` המייצג מחרוזת אחת או יותר, ומחזירה את ממוצע הדירוגים הממוצעים של כל הסרטים המשתייכים לכל קטגוריות אלה (כלומר, כמת "וגם" בין הקטגוריות). במידה ולא קיים אף סרט מדובר המתאים לכל הקטגוריות הללו, יש להחזיר את הערך `NO_RATING`.
חתימת המתודה:

```
public double getAverageRatingByMovieCategory(String... category)
```

הסימון המיוחד ... (3 נקודות) מאפשר לנו לקרוא למתודה עם מספר משתנה של פרמטרים (במקרה הזה, מספר משתנה של מחרוזות).

מידע נוסף ניתן למצוא ב:

<https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html>

תחת הסעיף Arbitrary Number of Arguments

(13) [10 נקודות] ממשו את המתודה `getHighestRatedMovieInAgeGroup` אשר מקבלת כפרמטר מספר משתמש (`id`), ומחזירה את הסרט עם הדירוג הממוצע הגבוה ביותר בקרב קבוצת הגיל שלו. החזירו את הסרט עם הדירוג הממוצע הכי גבוה, גם אם המשתמש קרא (ודירג) סרט זה. במידה ויש מספר סרטים עם הדירוג הכי גבוה, ביחרו שרירותית אחד מהם.

במידה ולא קיים משתמש עם מספר זה, הפונקציה תחזיר את הערך `null`.

חתימת המתודה:

```
public Movie getHighestRatedMovieInAgeGroup(int userId)
```

(14) [10 נקודות] לבסוף, נדפיס את ההמלצה לקובץ.

ממשו את המתודה `printRecommendationToFile` אשר מקבלת מספר משתמש (`id`) ומחרוזת שהיא נתיב לקובץ ומדפיסה את ההמלצה בפורמט הבא:

```
The recommended Movie for you is: XXX  
The movie's average rating among your age group is: XXX  
The movie's average rating among your occupation group is: XXX  
The movie's average rating among all users is: XXX
```

ההמלצה נקבעת על ידי שימוש ב `getHighestRatedMovieInAgeGroup` (הפונקציה מהסעיף הקודם) בסוף המשפט האחרון ניתן להוסיף ירידת שורה, לבחירתכם.

ניתן להניח שהפורמט של המחרוזת שמייצגת את שם הקובץ הוא חוקי, כלומר, המחרוזת אינה ריקה ומכילה שם חוקי של קובץ או נתיב תקין.
דוגמא לפלט אפשרי (עבור המשתמש עם id =1). יתכנו כמה פלטים אפשריים עבור משתמש זה.

The recommended Movie for you is: When Night Is Falling
The movie's average rating among your age group is: 5.0
The movie's average rating among your occupation group is: 5.0
The movie's average rating among all users is: 3.740740740740741

חתימת המתודה:

```
public void printRecommendationToFile(int userId, String fileName) throws  
Exception
```

השתמשו במתודות שמומשו עבורכם על מנת לייצר את מחרוזות הפלט.

טסטר

לתרגיל זה מצורפת מחלקת טסטר. על מנת להריץ את הטסטר עליהם לעדכן את מיקום שלושת קבצי הנתונים בהתאם למיקומם על המחשב שלכם. עדכנו את שלושת הקבועים שמכילים את הנתונים הנכונים לקבצים.

הוסיפו בדיקות נוספות מלבד הבדיקות המופיעות בטסטר. לצורך הבדיקות החדשות כדאי לעבורך את קבצי הנתונים כך שתוכלו לבדוק את עצמכם בקלות (למשל – במקום לבדוק מה ממוצע הדירוגים של ספר כלשהו, ואז לבדוק לאילו קבוצות גיל שייכים המדרגים, הוסיפו נתונים משלכם – הוסיפו/עדכנו משתמשים לפי קבוצת הגיל המבוקשת, ספרים חדשים ואת הדירוגים שתרצו לבצע עליהם את החישובים).

בהצלחה!