**תרגיל בית מספר 5 - Generics & Streams**

**שם הקורס: עיצוב ותכנות מ"ע להנדסה, מספר קורס: 157109**

# רקע ומטרות

מערכות המלצה ממלאות תפקיד מרכזי בעזרה למשתמשים לגלות תכנים שהם עשויים לאהוב – מספרים ועד מסעדות. מערכות אלו מנתחות כמויות גדולות של מידע על העדפות המשתמשים ומציעות פריטים שמתאימים לטעמם.

בתרגיל זה, תבנו מערכת המלצות בשפת Java תוך שימוש ב-Stream API בלבד – **בלי לולאות!** המערכת צריכה לתמוך בטיפוסים שונים, באמצעות שימוש ב-Generics.

במסגרת התרגיל תתנסו בעיבוד מידע, ניתוח דירוגים, חישוב הטיות ודמיון בין משתמשים, והמלצה על הפריטים המתאימים ביותר עבור כל משתמש. כל זאת תוך שימוש בפעולותStream .

מערכת ההמלצה תוגדר באופן גנרי, שיאפשר לה לעבוד עם סוגים שונים של פריטים – לדוגמה:

RecommenderSystem<Book> bookRec = new RecommenderSystem(...);

RecommenderSystem<Restaurant> restaurantRec = new RecommenderSystem(...);

השימוש ב-Generics מספק בטיחות טיפוסים, קוד ניתן לשימוש חוזר, ותמיכה טובה יותר ב־IDE ובקומפיילר.

# קבצי קלט

ניתנים לכם שלושה קבצי קלט, כאשר ההפרדה בין הערכים היא באמצעות Tab. אפשר להניח שהקבצים תקינים.

1. קובץ ratings.txs: כל שורה מייצגת דירוג בפורמט הבא:

<user\_id> <item\_id> <rating>

* + user\_id ו-item\_id הם מספרים שלמים
  + rating הוא מספר שלם בין 1 ל-5

1. קובץ users.tsv: כל שורה כוללת תיאור של משתמש בפורמט הבא:

<user\_id> <user\_name> <age> <gender>

1. קובץ books.tsv כל שורה כוללת תיאור של משתמש בפורמט הבא:

<book\_id> <book\_name> <author> <pages>

# דרישות כלליות

* **אסור להשתמש בלולאות (for, while, do-while) או בפונקציית forEach של Stream.**
* יש להשתמש ב־Stream API של Java לכל עיבוד הנתונים.
* יש להשתמש ב-Generics כך שמערכת ההמלצות תוכל לפעול עם כל סוג פריט שמממש את הממשק Item.
* הקוד צריך להיות בנוי בצורה מודולרית, עם פונקציות עזר ברורות שניתן לבדוק אותן בנפרד.

# אסטרטגיות המלצה

במסגרת התרגיל תממשו שלושה סוגים של מערכות המלצה:

1. מערכת המלצה מבוססת פופולריות
   * ממליצה על פריטים פופולריים במיוחד לפי ממוצע הדירוגים שלהם.
2. מערכת המלצה מבוססת פרופיל
   * ממליצה לכל משתמש לפי הפרופיל המבוסס על מין וגיל.
3. מערכת המלצה מבוססת דמיון
   * ממליצה לכל משתמש באופן אישי לפי משתמשים עם דפוסי דירוג דומים.

# דרישות מימוש

ניתנות לכם המחלקות הבאות. **אין לשנות את מחלקות אלה ואין להגיש אותן**:

* User – כולל שדות מספר מזהה, שם, גיל ומין.
* Item – ממשק שממומש על ידי:
  + Book – כולל שדות מספר מזהה, שם, מחבר ומספר עמודים.
* Rating – כולל שדות מזהה משתמש, מזהה פריט ודירוג (מספר בין 1 ל־5)

עליכם לממש את המחלקות הבאות (נתון לכם שלד של המחלקות, ובהן חתימות הפונקציות שעליכם לממש):

## 1. המחלקה המופשטת RecommenderSystem

הגדירו את המחלקה המופשטת על פי הדרישות הבאות:

**שדות:**

* Map<Integer, User> users – מיפוי מזהי משתמשים לאובייקטים מסוג User
* Map<Integer, T> items – מיפוי מזהי פריטים לאובייקטים מהטיפוס T
* List<Rating<T>> ratings – רשימת דירוגים
* **מומלץ להוסיף מבני עזר שיסייעו לכם בביצוע הפעולות.**

**בנאי:** מאתחל את המערכת על סמך המשתמשים, הפריטים והדירוגים, ובונה את המבנים הפנימיים.

**פעולה מופשטת** המחזירה את עשרת הפריטים המומלצים ביותר עבור המשתמש הנתון:

public abstract List<T> recommendTop10(int userId);

## 2. מחלקת PopularityBasedRecommender

ממשו תת מחלקה של RecommenderSystem בשם PopularityBasedRecommender

**מימוש פונקציית ההמלצה:**

עבור מזהה משתמש נתון:

* סננו פריטים שקיבלו לפחות 100 דירוגים ושלא דורגו עדיין על ידי המשתמש.
* חשבו את ממוצע הדירוגים עבור כל אחד מהפריטים הללו.
* החזירו את 10 הפריטים בעלי ממוצע הדירוג הגבוה ביותר, בסדר יורד. במקרה של שוויון, העדיפו את הפריטים עם יותר דירוגים. במקרה של שוויון, מיינו לפי שם הפריט.

בנוסף, המחלקה כוללת את הפונקציות הבאות לחישוב הדירוג הממוצע ומספר הדירוגים של פריט:

public double getItemAverageRating(int itemId)

public double getItemRatingsCount(int itemId)

## 3. מחלקת ProfileBasedRecommender

ממשו תת מחלקה של RecommenderSystem בשם ProfileBasedRecommender

**מימוש פונקציית ההמלצה:**

עבור מזהה משתמש נתון:

* **מצאו משתמשים תואמים:**
  + בעלי אותו מין
  + הפרש גיל של עד ±5 שנים (כולל)
* **סכמו את הדירוגים לפי פריט:**
  + לכל פריט, אספו את כל הדירוגים שלו מהמשתמשים התואמים.
  + שמרו רק פריטים שקיבלו לפחות 5 דירוגים בקבוצה זו.
  + חשבו את ממוצע הדירוגים של כל אחד מהפריטים הללו.
* **החזירו את עשרת הפריטים** שעדיין לא דורגו על ידי המשתמש, עם ממוצע הדירוג הגבוה ביותר, בסדר יורד. במקרה של שוויון, העדיפו את הפריט עם יותר דירוגים. במקרה של שוויון גם במספר הדירוגים, מיינו לפי שם הפריט.

בנוסף המחלקה כוללת את הפונקציה הבאה, המחזירה את רשימת המשתמשים התואמים למשתמש:

public List<User> getMatchingProfileUsers(int userId)

## 4. מחלקת SimilarityBasedRecommender

ממשו תת מחלקה של RecommenderSystem בשם SimilarityBasedRecommender.

**מימוש פונקציית ההמלצה:** עליכם לממש שיטה מבוססת דמיון תוך הסרת הטיות, לפי השלבים הבאים:

**שלב 1: הסרת הטיות (Bias Removal)**

כאשר משתמשים מדרגים פריטים, המספרים שהם נותנים מושפעים מגורמים כלליים – לא רק מהעדפה אישית. גורמים אלו נקראים **הטיות**, ויש להסיר אותם לפני שניתן להשוות בין משתמשים ולחשב את הדמיון ביניהם.

* **הטיה גלובלית (Global Bias)**ממוצע הדירוגים בכל המערכת (כל המשתמשים וכל הפריטים). ערך זה מבטא עד כמה הדירוגים במערכת גבוהים או נמוכים. זו ההטיה הבסיסית – יש להחסיר אותה מכל דירוג.
* **הטיית פריט (Item Bias)**יש פריטים שפופולריים יותר מאחרים. על מנת לחשב את הטיית הפריט:
  + מחסרים את ההטיה הגלובלית מכל דירוג.
  + מחשבים את ממוצע הדירוגים של כל פריט – זו הטיית הפריט.
* **הטיית משתמש (User Bias)**חלק מהמשתמשים נוטים לדרג גבוה או נמוך יותר מאחרים. יש משתמשים שמדרגים הכל 5 ויש כאלה שרק פריטים בודדים יקבלו אצלם דירוג מעל 3. על מנת לחשב את הטיית המשתמש:
  + מחסרים גם את ההטיה הגלובלית וגם את הטיית הפריט מכל דירוג.
  + מחשבים ממוצע ממוצע הדירוגים של כל משתמש – זו הטיית המשתמש.

לאחר הסרת כל שלוש ההטיות, מתקבל **דירוג נטול הטיות**, שישמש אותנו לחישוב הדמיון:

bias-free rating = rating − global bias − item bias − user bias

בעת חיזוי דירוג עבור משתמש, יש להוסיף חזרה את שלוש ההטיות (הגלובלית, של הפריט, ושל המשתמש) כדי לחשב את הדירוג הסופי.

**שלב 2: חישוב דמיון בין משתמשים (User Similarity)**

עבור מזהה משתמש נתון:

* מצאו משתמשים אחרים שדרגו **לפחות 10 פריטים משותפים** איתו.
* לכל משתמש מתאים, חשבו את הדמיון:
  + השתמשו ב[**מכפלה סקלרית**](https://he.wikipedia.org/wiki/%D7%9E%D7%9B%D7%A4%D7%9C%D7%94_%D7%A1%D7%A7%D7%9C%D7%A8%D7%99%D7%AA) **(dot product)** בין הדירוגים נטולי ההטיות על הפריטים המשותפים.

**שלב 3: חיזוי דירוגים (Predict Ratings)**

* מצאו את **10 המשתמשים הכי דומים** (באמצעות הדמיון משלב 2).
* מצאו את הפריטים שלפחות חמישה מהמשתמשים הדומים דירגו ושהמשתמש הנתון לא דירג.
* חיזוי הדירוג עבור הפריטים שנמצאו באמצעות **ממוצע משוקלל**
  + השתמשו בדירוגים נטולי ההטיות של המשתמשים הדומים.
  + המשקל הוא ציון הדמיון.
* הוסיפו בחזרה את ההטיה הגלובלית, הטיית הפריט, והטיית המשתמש כדי לקבל את הציון הסופי.

**שלב 4: 10 ההמלצות הטובות ביותר**

* החזירו את 10 הפריטים עם הדירוג הצפוי הגבוה ביותר. במקרה של שוויון, העדיפו פריטים עם יותר דירוגים כלליים במערכת. במקרה של שוויון גם במספר הדירוגים, מיינו לפי שם הפריט.

בנוסף לפונקציית ההמלצה, המחלקה כוללת את הפונקציות הבאות:

* public void getGlobalBias(): הדפסת ההטיה הגלובלית.
* public void getItemBias(int itemId). הדפסת הטיית הפריט. זכרו שהטיית הפריט מחושבת לאחר שמחסרים את ההטיה הגלובלית.
* public void getUserBias(int userId). הדפסת הטיית המשתמש. זכרו שהטיית המשתמש מחושבת לאחר שמחסרים את ההטיה הגלובלית ואת ההטיות של הפריטים.
* public double getSimilarity(int u1, int u2)

## 5. מחלקת בדיקה MainApp

מחלקה זו נועדה לסייע לכם לבדוק את הקוד שלכם. **אין להגיש מחלקה זו.**

על מנת להשתמש במחלקה, עליכם להשלים את אתחול מבני הנתונים הנדרשים עבור מערכת ההמלצה.

**דגש חשוב**: השלד כולל רק את הפונקציות שייבדקו בבודק האוטומטי. **מומלץ מאד להוסיף פונקציות עזר נוספות כדי לחלק את הפעולות המורכבות לחלקים פשוטים.**

בהצלחה!