236703 – תכנות מונחה עצמים, חורף 2019

תרגיל בית 2 – היכרות עם Java

כללי

1. מועד ההגשה : **28/11/2018** בשעה **23:59**
2. מטרת התרגיל היא הכרות עם Java, עבודה עם Exceptions, Collections, Comparable/Comparator, Streams ומימוש מנשקים.
3. קראו היטב את ההוראות, במסמך זה ובקוד שניתן לכם.
4. אחראי על התרגיל: **גיא**. שאלות על התרגיל יש לשלוח במייל guy-suday@cs.technion.ac.il עם הנושא: "HW2 236703".
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 תדרשו לממש רשת חברתית של פרופסורים רעבים בטכניון, המאפשרת להם ליצור חברויות אחד עם השני, לדרג בתי בוריטו בקרבה לטכניון, לקבל המלצות מחברים על בתי בוריטו (מעתה ייקראו "מסעדות") ועוד.

חלק א'

בחלק זה נממש את הבסיס של המערכת: פרופסורים ומסעדות.

CasaDeBurritoImpl

מחלקה זו תממש את ההתנהגות של המנשק CasaDeBurrito אשר סופק לכם. מחלקה זו מייצגת מסעדה בודדת. המנשק CasaDeBurrito מרחיב את המנשק Comparable<CasaDeBurrito> המאפשר השוואה בין אובייקטים בעזרת המתודה compareTo. אופי המימוש מתואר בהמשך. המחלקה תספק את ההתנהגות הבאה:

- CasaDeBurritoImpl(int id, String name, int dist, Set<String> menu) - בנאי המקבל את מזהה המסעדה, את שמה, את מרחקה מהטכניון וקבוצה של מנות בתפריט (שמות של מנות). ניתן להניח שיתקבלו פרמטרים חוקיים (מספרים אי שליליים, תפריט לא ריק וכו').

- getId() – מחזירה את מזהה המסעדה.

- getName() – מחזירה את שם המסעדה.

- distance() - מחזירה את מרחק המסעדה מהטכניון.

- boolean isRatedBy(Profesor p) – מחזירה true אמ"מ המסעדה הנוכחית דורגה ע"י הפרופסור שהתקבל כפרמטר.

- rate(Profesor p, int r) - הפרופסור המתקבל כפרמטר מדרג את המסעדה. הדירוג הוא מספר שלם בתחום [0,...,5]. אם המספר שהתקבל אינו דירוג חוקי, תיזרק חריגת RateRangeException. אם הפרופסור כבר דירג את המסעדה, דירוגו מתעדכן. על המתודה להחזיר את המופע של האובייקט (this) כדי לאפשר שרשור פעולות.

- numberOfRates() - מחזירה את מספר הדירוגים של המסעדה.

- averageRating() - מחזירה את ממוצע הדירוג של המסעדה. אם אין דירוגים, המתודה תחזיר 0.

- equals(Object o) - עליכם לדרוס את המתודה equals שהוגדרה לראשונה ב Object, כפי שנלמד בתרגול. מסעדות יחשבו שוות אמ"מ יש להן אותו מספר מזהה.

- toString() - מחזירה מחרוזת המתארת את המסעדה - מוסבר בהמשך.

- compareTo(CasaDeBurrito c) - משווה את המסעדה עם מסעדה אחרת שמתקבלת כפרמטר, לפי הסדר הטבעי שיוגדר בהמשך.

ProfesorImpl

מחלקה זו תממש את ההתנהגות של המנשק Profesor אשר סופק לכם (שימו לב שהשם נכתב בספרדית ולא באנגלית לאורך התרגיל). מחלקה זו מייצגת פרופסור בודד. בנוסף, המנשק Profesor מרחיב את המנשק Comparable<Profesor> המאפשר השוואה בין אובייקטים בעזרת המתודה compareTo. אופי המימוש מתואר בהמשך. המחלקה תספק את ההתנהגות הבאה:

- ProfesorImpl(int id, String name) - בנאי המקבל מספר זהות ושמו של הפרופסור ומאתחל פרופסור עם ערכים אלו. ניתן להניח שיתקבלו פרמטרים חוקיים.

- getId() – מחזירה את מזהה הפרופסור.

- favorite(CasaDeBurrito c) - המסעדה שהתקבלה כמפרמטר נהיית מועדפת ע"י הפרופסור. אם המסעדה אינה מדורגת ע"י הפרופסור, יש לזרוק חריגה מסוג UnratedFavoriteCasaDeBurritoException. המתודה תחזיר את המופע של האובייקט על מנת לאפשר שרשור של פעולות.

- favorites() - מחזירה את המסעדות המועדפות ע"י הפרופסור. על שינויים באוסף זה לא להשפיע על אוסף המסעדות המועדפות על הפרופסור.

- addFriend(Profesor p) - מוסיפה קשר חברות ברשת החברתית אל הפרופסור שהתקבל כפרמטר. **קשר "חברות עצמית" אינו חוקי**. במקרה כזה יש לזרוק חריגה מסוג SameProfesorException. אם קיים כבר קשר חברות ביניהם, יש לזרוק חריגה מסוג ConnectionAlreadyExistsException. המתודה תחזיר את המופע של האובייקט על מנת לאפשר שרשור של פעולות.

  getFriends() - מחזירה אוסף המכיל את חבריו של הפרופסור. על שינויים באוסף זה לא להשפיע על אוסף החברים של הפרופסור.

- filteredFriends(Predicate<Profesor> p) – מחזירה אוסף המכיל את חבריו של הפרופסור שעונים על הפרדיקט p. על שינויים באוסף זה לא להשפיע על אוסף החברים של הפרופסור.

- filterAndSortFavorites(Comparator<CasaDeBurrito> c, Predicate<CasaDeBurrito> p) - מחזירה אוסף של כל המסעדות המועדפות ע"י הפרופסור, אשר עונות על תנאי הפרדיקט p . סדר המעבר על האוסף המוחזר הוא לפי ה Comparator**.** כלומר, **זהו לא הסדר הטבעי המוגדר ע"י compareTo**!

- favoritesByRating(int r) - מחזירה אוסף של כל המסעדות המועדפות ע"י הפרופסור, **עם דירוג ממוצע גבוה מ (או שווה ל) זה שהתקבל כפרמטר**. סדר המעבר על האוסף המוחזר הוא לפי דירוג מסעדות אלו בסדר יורד. עבור מסעדות עם דירוג זהה, הסידור המשני יהיה לפי מרחק מהטכניון, בסדר עולה. סידור שלישי (עבור מסעדות שנמצאות במרחק זהה, ובעלות דירוג זהה) יהיה לפי מספר מזהה בסדר עולה. כלומר, **זהו לא הסדר הטבעי המוגדר ע"י compareTo!**

- favoritesByDist(int r) - מחזירה אוסף של כל המסעדות המועדפות ע"י הפרופסור, **אשר נמצאות במרחק קצר מ (או שווה ל) זה שהתקבל כפרמטר**. סדר המעבר על האוסף המוחזר הוא לפי המרחק של המסעדות מהטכניון לפי סדר עולה. עבור מסעדות שנמצאות במרחק שווה מהטכניון, סידור משני של המסעדות יהיה לפי דירוג בסדר יורד, וסידור שלישי (עבור מסעדות שנמצאות במרחק זהה, ובעלות דירוג זהה) יהיה לפי מספר מזהה בסדר עולה. כלומר, **זהו לא הסדר הטבעי המוגדר ע"י compareTo**!

- רמז: ניתן להשתמש במתודה `filterAndSortFavorites` למימוש `favoritesByRating`,`favoritesByDist`. כמו כן, היעזרו בפעולות `filter` ו `sorted` של שטפים (`Streams`).

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

- `toString()` - מחזירה מחרוזת המתארת את המסעדה - מוסבר בהמשך.

- `compareTo(Profesor p)` - משווה את הפרופסור עם פרופסור אחר שמתקבל בפרמטר, לפי הסדר הטבעי שיוגדר בהמשך.

- **סדר טבעי:** עליכם לדרוס את המתודה `compareTo`, המוגדרת במנשק `Comparable` כפי שראיתם בכיתה. המתודה מגדירה יחס סדר בין המסעדה/הפרופסור הנוכחי/ת למסעדה/פרופסור שהתקבל/ה כפרמטר. עליה להחזיר ערך שלילי אם המסעדה/פרופסור הנוכחי/ת בעלת מזהה קטן משל המסעדה/הפרופסור שהתקבל/ה, חיובי אם להיפך, ו-0 אם המזהים הם שווים.

חלק ב' - מימוש המנשק `CartelDeNachos`

בחלק זה נממש את המערכת הכוללת. במערכת נוכל להוסיף פרופסורים, להגדיר חברויות בין פרופסורים ולקבל מידע על קשרים בין פרופסורים, ועל המסעדות העדיפות עליהם. שימו לב שקשר חברות בין אנשים במערכת הוא קשר סימטרי, כלומר אם A חבר של B, אז גם B חבר של A, אך איננו רפלקסיבי. כלומר, אדם אינו יכול להיות חבר של עצמו (וכמובן שאיננו טרנזיטיבי).

בהמשך נראה שהמערכת מדמה גרף של חברויות בו כל צומת הוא פרופסור, והקשתות הן קשרי החברות בין הפרופסורים. אנו נרצה להשתמש בעובדה זו בהמשך.

`CartelDeNachosImpl`

המחלקה תספק את ההתנהגות הבאה:

- `CartelDeNachosImpl()` - מאתחל את המערכת.

- `joinCartel(int id, String name)` - מקבלת נתונים של פרופסור, מייצרת מופע של `Profesor` בהתאם ומוסיפה אותו למערכת, לבסוף מחזירה את המופע שיצרה. אם קיים כבר פרופסור עם אותו מספר מזהה במערכת, יש לזרוק חריגה `ProfesorAlreadyInSystemException`.

- addCasaDeBurrito(int id, String name, int dist, Set<String> menu) - מקבלת נתונים של מסעדה, מייצרת מופע של CasaDeBurrito בהתאם, ומוסיפה אותה למערכת, לבסוף מחזירה את המופע שיצרה. אם קיימת כבר מסעדה עם אותו המזהה במערכת, יש לזרוק חריגה CasaDeBurritoAlreadyInSystemException.

- registeredProfesores() - מחזירה אוסף של הפרופסורים שבמערכת. על שינויים באוסף זה לא להשפיע על המערכת.

- registeredCasasDeBurrito() - מחזירה אוסף של המסעדות שבמערכת. על שינויים באוסף זה לא להשפיע על המערכת.

- getProfesor(int id) - מחזירה רפרנס ל-Profesor שבמערכת עם מספר המזהה המבוקש. אם לא קיים פרופסור כזה במערכת, יש לזרוק חריגת ProfesorNotInSystemException.

- getCasaDeBurrito(int id) - מחזירה רפרנס ל-CasaDeBurrito שבמערכת עם מספר המזהה המבוקש. אם לא קיימת מסעדה כזו במערכת, יש לזרוק חריגת CasaDeBurritoNotInSystemException.

- addConnection(Profesor p1, Profesor p2) - מוסיפה קשר חברות בין שני פרופסורים רעבים במערכת. **זכרו שקשר חברות הוא סימטרי**. אם אחד מהם לא קיים במערכת, יש לזרוק חריגת ProfesorNotInSystemException. אם שני הפרמטרים מייצגים אותו אדם, יש לזרוק חריגת SameProfesorException. אם החברות כבר קיימת, יש לזרוק ConnectionAlreadyExistsException. המתודה תחזיר את המופע של האובייקט על מנת לאפשר שרשור של פעולות.

- favoritesByRating(Profesor p) - מחזירה אוסף של מסעדות, אשר יכלול את כל המסעדות המועדפות של כל החברים של הפרופסור שהתקבל כפרמטר, שנמצאות במערכת. על האיטרציה ב-stream הנוצר מהאוסף לעמוד בדרישות הבאות:
  - סדר המעבר על החברים הוא לפי סדר עולה של מספרי זהות.
  - עבור כל חבר, סדר המעבר על המסעדות המועדפות הוא לפי דירוג ממוצע, בסדר יורד. עבור מסעדות עם דירוג ממוצע זהה, הסידור הוא לפי מרחק מהטכניון, בסדר עולה. סידור שלישי יהיה לפי מספר מזהה בסדר עולה.
  - יש לעבור על המסעדות המועדפות של חבר אחד לפני המעבר לחבר הבא.
  - אין כפילויות באוסף. אם חבר מעדיף מסעדה R, וחבר שני מעדיף את אותה המסעדה, אזי R מופיע בסדר פעם אחת בדיוק (בנוסף לשאר המסעדות המועדפות ע"י החבר הראשון).
  - האוסף מכיל מסעדות מועדפות של חברים ישירים של הפרופסור בלבד. כלומר, לא יופיעו באוסף מסעדות המועדפות על הפרופסור עצמו ולא על חברים של חברי הפרופסור.

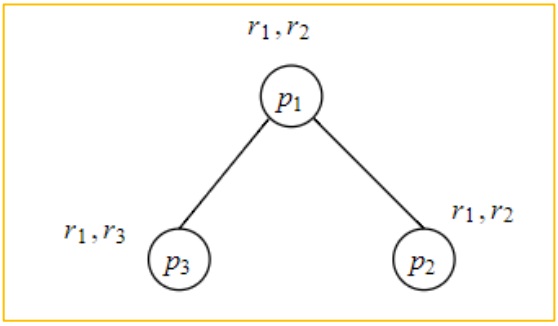
אם הפרופסור לא נמצא במערכת, יש לזרוק חריגה מסוג ProfesorNotInSystemException.

- `favoritesByDist(Profesor p)` - מחזירה אוסף של מסעדות, אשר יכלול את המסעדות המועדפות של חבריו של הפרופסור שהתקבל כפרמטר, הנמצאות במערכת. על האיטרציה ב-`stream` הנוצר מהאוסף לעמוד בדרישות הבאות:
  - סדר המעבר על החברים הוא לפי סדר עולה של מספרי זהות.
  - עבור כל חבר, סדר המעבר על המסעדות המועדפות הוא לפי מרחק מהטכניון, בסדר עולה. עבור מסעדות במרחק זהה, הסידור המשני יהיה לפי דירוג ממוצע בסדר יורד. סידור שלישי יהיה לפי מספר מזהה, בסדר עולה.
  - במעבר עוברים על כל המסעדות המועדפות של חבר לפני שעובר לחבר הבא.
  - אין כפילויות באוסף. אם חבר מעדיף מסעדה R, וחבר שני מעדיף את אותה המסעדה, אזי R מופיע בסדר פעם אחת בדיוק (בנוסף לשאר המסעדות המועדפות ע"י החבר הראשון).
  - האוסף מכיל מסעדות מועדפות של חברים ישירים של הפרופסור בלבד. כלומר, לא יופיעו באוסף מסעדות המועדפות על הפרופסור עצמו ולא על חברים של חבריו הפרופסור.

אם הפרופסור לא נמצא במערכת, יש לזרוק חריגה מסוג `ProfesorNotInSystemException`.

- `getRecommendation(Profesor p, CasaDeBurrito c, int t)` - נאמר כי מסעדה היא מומלצת-t ע"י פרופסור, אם הוא נמצא במרחק לכל היותר t קשרי חברויות מפרופסור אחר אשר מעדיף מסעדה זו. המתודה מחזירה ערך בוליאני האם המסעדה c מומלצת-t ע"י המשתמש s. אם s לא נמצא במערכת, תיזרק חריגת `ProfesorNotInSystemException`. אם c אינה במערכת, יש תיזרק חריגת `CasaDeBurritoNotInSystemException`. אם t שלילי, תיזרק חריגת `ImpossibleConnectionException`.
  - מספר קשרי החברויות בין פרופסור לעצמו מוגדר להיות 0.
  - יש להזהר מקריאות רקורסיביות עמוקות בלתי נשלטות (לולאות אינסופיות).
  - רמז: חשבו על הדרכים למעבר על גרפים.

- `getMostPopularRestaurantsIds()` – נאמר שפרופסור תורם t נקודות למסעדה, אם יש לו t חברים (לא כולל עצמו) שמסעדה זו מועדפת עליהם. כמות הנקודות הכוללת של מסעדה במערכת היא סכום התרומות של כל הפרופסורים.
המתודה מחזירה את רשימת המזהים של המסעדות הפופולריות ביותר (ניקוד מקסימלי במערכת).
הרשימה תהיה ממוינת בסדר עולה (לפי מזהה המסעדה).
<u>שימו לב</u>: אם מסעדה לא מועדפת על אף פרופסור, הניקוד שלה הוא 0.
דוגמה: פרוספור p1 תורם 2 נקודות למסעדה r1 (החברים שלו p2,p3 מעדיפים אותה), נקודה אחת ל-r2 ונקודה אחת ל-r3 (חבר p2 מעדיף את r2 וחבר p3 מעדיף את r3).
פרופסור p2 תורם נקודה אחת ל-r1,r2 (החבר p1 מעדיף את r1,r2).
פרוספור p3 תורם נקודה אחת ל-r1,r2 (החבר p1 מעדיף את r1,r2).
סך כל הניקוד: 4 נק' ל-r1, 3 נק' ל-r2, 1 נק' ל-r3.
לכן תוחזר רשימה המכילה את המזהה 1.



- `toString()` - מחזירה מחרוזת המתארת את המסעדה - מוסבר בהמשך.

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 נפד 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

- המתודה toString(): כנהוג ב Java, על המתודה להחזיר תיאור של האובייקט **בהתאם לפורמט המוגדר מראש במנשק המסופק**. שימו לב, על מנת לממש את מתודות אלו בקלות, עליכם לחשוב כיצד לבנות את המחלקות ואילו שדות יהיו לאובייקטים מטעמן.
- 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. למשל, בעת המעבר על אוסף המסעדות שמתקבל מקריאה ל-favoritesByRate לא יתווספו דירוגים למסעדות אחרות.
- ניתן להניח שבזמן בדיקת המערכת, יתווספו קשרי חברות רק באמצעות קריאה ל-addConnection , ולא ישירות דרך מופע של Profesor.

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.

<u>אין חובה לבדוק את התרגיל כלל, וכך או כך אין להגיש בדיקות.</u>

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 (נתן) בלבד. 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.

בהצלחה !!!