236703 – תכנות מונחה עצמים, חורף 2023-24

תרגיל בית 2 – היכרות עם Java

כללי

1. מועד ההגשה : **19/2/2024** בשעה **23:55**
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: **ג'וליאן**. שאלות על התרגיל יש לשלוח במייל julianmour@campus.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 סטודנטים רעבים בטכניון, המאפשרת לסטודנטים ליצור חברויות אחד עם השני, לדרג מסעדות בקרבה לטכניון, לקבל המלצות מחברים על מסעדות ועוד.

חלק א'

בחלק זה נממש את הבסיס של המערכת: סטודנטים ומסעדות.

RestaurantImpl

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

- RestaurantImpl(int id, String name, int distFromTech, Set<String> menu) - בנאי המקבל מספר זהות המסעדה, השם שלה, המרחק מהטכניון וקבוצה של מנות בתפריט (שמות של מנות). ניתן להניח שיתקבלו פרמטרים חוקיים (מספרים אי שליליים, תפריט לא ריק וכו').

- distance() - מחזירה את מרחק המסעדה מהטכניון.

- rate(HungryStudent s, int r)- הסטודנט שהתקבל כפרמטר מדרג את המסעדה. הדירוג הינו מספר שלם בתחום [0,...,5]. אם המספר שהתקבל אינו דירוג מתאים, תיזרק חריגת RateRangeException. אם הסטודנט כבר דירג את המסעדה, דירוגו מתעדכן. על המתודה להחזיר את המופע של האובייקט 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(Restaurant r) - משווה את המסעדה עם מסעדה אחרת שמתקבלת כפרמטר, לפי הסדר הטבעי שיוגדר בהמשך.

HungryStudentImpl

מחלקה זו תממש את ההתנהגות של הממשק HungryStudent אשר סופק לכם. מחלקה זו מייצגת סטודנט בודד. בנוסף, הממשק HungryStudent מרחיב את הממשק Comparable<HungryStudent> המאפשר השוואה בין אובייקטים בעזרת המתודה compareTo. אופי המימוש מתואר בהמשך. המחלקה תספק את ההתנהגות הבאה:

- HungryStudentImpl(int id, String name) - בנאי המקבל מספר זהות ושמו של הסטודנט ומאתחל סטודנט עם ערכים אלו.

- favorite(Restaurant) - המסעדה שהתקבלה כפרמטר נהיית מועדפת ע"י הסטודנט. אם המסעדה אינה מדורגת ע"י הסטודנט, יש לזרוק חריגה מסוג UnratedFavoriteRestaurantException. המתודה תחזיר את המופע של האובייקט על מנת לאפשר שרשור של פעולות.

- favorites() - מחזירה את המסעדות המועדפות ע"י הסטודנט.

- addFriend(HungryStudent s) - מוסיפה קשר חברות ברשת החברתית אל הסטודנט שהתקבל כפרמטר. <u>קשר "חברות עצמית" אינו חוקי.</u> במקרה כזה יש לזרוק חריגה מסוג SameStudentException. 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 הסטודנט.

- favoritesByRating(int r) - מחזירה אוסף של כל המסעדות המועדפות ע"י הסטודנט, עם דירוג ממוצע גבוה מ (או שווה ל) זה שהתקבל כפרמטר. סדר המעבר על האוסף המוחזר הינו 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 הסטודנט, אשר נמצאות במרחק קצר מ (או שווה ל) זה שהתקבל כפרמטר. סדר המעבר על האוסף המוחזר הינו 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: כדי לממש את שתי המתודות הללו, מומלץ להיזכר בממשק Comparator<T> שלמדתם בתרגול, ולהשתמש במחלקות המממשות את הממשק. בממשק יש מתודה אחת שחובה לממש - compare. לחילופין, נסו לשלב lambda expression במימוש זה. היעזרו בפעולות filter ו sorted של שטפים (Streams).

- equals(Object o) - עליכם לדרוס את המתודה equals שהוגדרה לראשונה בObject, כפי שנלמד בתרגול. שני סטודנטים יחשבו שווים אמ"מ יש להם אותו מספר מזהה.
- toString() - מחזירה מחרוזת המתארת את המסעדה - מוסבר בהמשך.
- compareTo() - משווה את הסטודנט עם סטודנט אחר שמתקבל בפרמטר, לפי הסדר הטבעי שיוגדר בהמשך.

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

חלק ב' - מימוש הממשק HamburgerNetwork

בחלק זה נממש את המערכת הכוללת. במערכת נוכל להוסיף סטודנטים, להגדיר חברויות בין סטודנטים ולקבל מידע על קשרים בין סטודנטים, ועל המסעדות העדיפות עליהם. שימו לב שקשר חברות בין אנשים במערכת הינו 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 סטודנט, והקשתות הן קשרי החברות בין הסטודנטים. אנו נרצה להשתמש בעובדה זו בהמשך.

HamburgerNetworkImpl

המחלקה תספק את ההתנהגות הבאה:

- HamburgerNetworkImpl() - מאתחל את המערכת.

- joinNetwork(int id, String name) - מקבלת נתונים של סטודנט, מייצרת מופע של HungryStudent בהתאם ומוסיפה אותו למערכת, לבסוף מחזירה את המופע שיצרה. אם קיים כבר סטודנט עם אותו מספר מזהה במערכת, יש לזרוק חריגה StudentAlreadyInSystemException.

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

- registeredStudents() - מחזירה אוסף של הסטודנטים הרעבים במערכת.

- registeredRestaurants() - מחזירה אוסף של המסעדות במערכת.

- getStudent(int id) - מחזירה מופע של HungryStudent שבמערכת עם מספר המזהה המבוקש. אם לא קיים סטודנט כזה במערכת, יש לזרוק חריגת StudentNotInSystemException.

- getRestaurant(int id) - מחזירה מופע של Restaurant שבמערכת עם מספר המזהה המבוקש. אם לא קיימת מסעדה כזו במערכת, יש לזרוק חריגת RestaurantNotInSystemException.

- addConnection(HungryStudent s1, HungryStudent s2) - מוסיפה קשר חברות בין שני סטודנטים רעבים במערכת. <u>זכרו שקשר חברות הינו סימטרי!</u> אם אחד מהם לא קיים במערכת, יש לזרוק חריגת StudentNotInSystemException. אם שני הפרמטרים מייצגים אותו אדם, יש לזרוק חריגת SameStudentException. אם החברות כבר קיימת, יש לזרוק ConnectionAlreadyExistsException. המתודה תחזיר את המופע של האובייקט על מנת לאפשר שרשור של פעולות.

- favoritesByRating(HungryStudent s) - מחזירה אוסף של מסעדות, אשר יכלול את כל המסעדות המועדפות של כל החברים של הסטודנט שהתקבל כפרמטר, שנמצאות במערכת. על האיטרציה בstream הנוצר מהאוסף לעמוד בדרישות הבאות:
  - סדר המעבר על החברים הינו לפי סדר עולה של מספרי זהות.
  - עבור כל חבר, סדר המעבר על המסעדות המועדפות הינו לפי דירוג ממוצע, בסדר יורד. עבור מסעדות עם דירוג ממוצע זהה, הסידור הינו 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 (ביחד עם שאר המסעדות המועדפות ע"י החבר הראשון).
- האוסף מכיל מסעדות מועדפות של חברים ישירים של הסטודנט בלבד. כלומר, לא יופיעו באוסף מסעדות המועדפות על הסטודנט עצמו ולא על חברים של חברי הסטודנט.

אם הסטודנט לא נמצא במערכת, יש לזרוק חריגה מסוג StudentNotInSystemException.

- favoritesByDist(HungryStudent s) - מחזירה אוסף של מסעדות, אשר יכלול את המסעדות המועדפות של חבריו של הסטודנט שהתקבל כפרמטר, הנמצאות במערכת. על האיטרציה בstream הנוצר מהאוסף לעמוד בדרישות הבאות:
  - סדר המעבר על החברים הינו לפי סדר עולה של מספרי זהות.
  - עבור כל חבר, סדר המעבר על המסעדות המועדפות הינו 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 (ביחד עם שאר המסעדות המועדפות ע"י החבר הראשון).
  - האוסף מכיל מסעדות מועדפות של חברים ישירים של הסטודנט בלבד. כלומר, לא יופיעו באוסף מסעדות המועדפות על הסטודנט עצמו ולא על חברים של חברי הסטודנט.

אם הסטודנט לא נמצא במערכת, יש לזרוק חריגה מסוג StudentNotInSystemException.

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

- getRecommendation(HungryStudent s, Restaurant r, int t) - אנו נאמר כי מסעדה היא מומלצת-t ע"י סטודנט, אם הוא נמצא במרחק לכל היותר t קשרי חברויות מסטודנט אחר אשר מעדיף מסעדה זו. המתודה מחזירה ערך בוליאני האם המסעדה r מומלצת-t ע"י המשתמש s. אם s לא נמצא במערכת, תיזרק חריגת StudentNotInSystemException. אם r אינה במערכת, יש תיזרק חריגת RestaurantNotInSystemException. אם t שלילי, תיזרק חריגת ImpossibleConnectionException.
  - מספר קשרי החברויות בין סטודנט לעצמו מוגדר להיות 0.
  - יש להיזהר מקריאות רקורסיביות עמוקות בלתי נשלטות (לולאות אינסופיות).
  - רמז: חשבו על הדרכים למעבר על גרפים.

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 בממשק 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 לaddConnection, ולא ישירות דרך מופע של HungryStudent.

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 (**ג'וליאן**) 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 (חוץ מאלה של package OOP.Solution כמובן) ואין להגיש test-ים.
- הגשה שלא לפי ההוראות תגרור הורדת ציון בהתאם.

בהצלחה !!!