תרגיל בית מס. 3
מימוש גרפים
פולימורפיזם, static, final
מערכים, אוספים, equals, compare, חריגות
להגשה עד יום חמישי 08.05 למניינם
ההגשה בזוגות במוודל עד השעה 23:00
משקל התרגיל: 6 נק'

התרגיל עוסק במימוש גרפים (חלק ראשון), מערכים ואוספים (חלק שני), חריגות (חלק שלישי) ותיעוד קוד (חלק רביעי).

<u>אופן ההגשה:</u>
הורידו את הקבצים הנתונים במטלה במוודל, ושימו אותם בפרוייקט חדש שהכנתם תחת התיקייה הראשית src.
כל המחלקות שתרשמו בפרוייקט יהיו תחת התיקייה הראשית src אלא אם כן יש הוראות מיוחדות אחרות (ראו חלק ג').
אפשר גם להוריד את הפרוייקט מהגיטהב – פרוייקט OOP-HW3-Graph בקישור:
https://github.com/michalHorovitz/OOP2022Public

יש לארוז בקובץ zip אחד את כל קבצי ה-java בפרוייקט (מלבד קבצי הבדיקה ב-Junit)
וקובץ pdf אחד ובו תשובותיכם לשאלות הפתוחות. שם הקובץ צריך להיות:

40_HW3_123456789_987654321.zip

כאשר *123456789* ו-*987654321* הם מספרי הזהות (בני 9 ספרות כל אחד, גם אם מתחילים ב-0) של המגישים.

שם קובץ ה-pdf עם התשובות יהיה – hw3.pdf, והוא יהיה כלול בתוך ה-zip.
שימו לב כי הקבצים מחלק ג' שנמצאים בתת תיקייה צריכים להופיע עם תת התיקייה שלהם.
כלומר כאשר פותחים את ה-zip המוגש נקבל תיקייה המכילה את קבצי ג'אווה של חלקים 1-3, ותיקייה ובה הקבצים של חלק 4.

לא תתקבל עבודה שאינה מתקמפלת בבודק האוטומטי.

בנוסף, בבודק האוטומטי יש מספר טסטים ודוגמאות על מנת שתוכלו לבדוק את הקוד שכתבתם עוד לפני הגשתו. בתחילת הבדיקה של התרגיל שלכם, אנחנו נערוך את אותם הבדיקות האוטומטיות. לפיכך, שימו לב כי הינכם מקבלים OK עבור הבדיקות האלו.

<u>תיאור והסבר על גרפים:</u>
ניתן לקרא על גרפים בויקיפדיה: גרף (תורת הגרפים) ויקיפדיה.
גרפים מתארים קשרים בין ישויות, ותיתקלו בהם הרבה בהמשך לימודיכם בתואר.
בתרגיל זה נעסוק בשני סוגים עיקריים של גרפים: גרף מכוון וגרף לא מכוון.
כל הגרפים שנעסוק בהם הם ללא משקולות או תוויות אחרות על הקשתות.

חלק ראשון – מימוש גרפים ושימוש בכך:

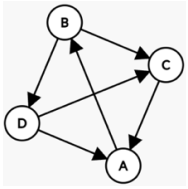
שלב א': מימוש IGraph<V>

בחלק זה עליכם תחילה לממש את המנשק הגנרי IGraph הנתון, עבור גרפים מכוונים ולא מכוונים.
הפרמטר הגנרי V הוא הטיפוס של הקדקודים בגרף, פרמטר זה מרחיב את Comparable<V>, וזאת על מנת להגדיר סדר על הקדקודים להדפסה אחידה בכל ההרצות.

עליכם להגדיר 2 מחלקות הממשות את IGraph<V>.

1. DirectedGraph<V extends Comparable<V>> -עבור מימוש של גרף מכוון.
2. UndirectedGraph<V extends Comparable<V>> -עבור מימוש של גרף לא מכוון.

הדפסת הגרף נעשית לפי התיעוד במנשק Igraph בדיוק, למשל, עבור הגרף המכוון הבא:



יודפס (כאשר \t מתאר הוספת רווח טאב - tab):

```
DirectedGraph:\tA:{B} B:{C,D} C:{A} D:{A,C}
```

אנחנו נממש את הגרפים ע"י השדה:

```
private SortedMap<V, SortedSet<V>> vertices;
```

המכיל מיפוי מקדקוד (מפתח, מטיפוס V), לקבוצת השכנים שלו, כלומר מיפוי מקדקוד (מפתח) לקבוצת הקדקודים שיש קשת מהקדקד אליהם (ערך). למשל בגרף למעלה:
מ-D יש קשתות ל-A ול-C, לכן עבור המפתח D הערך המותאם הוא הקבוצה {A,C}.

הטיפוס הגנרי V ירחיב את Comparable<V>, וזאת על מנת להגדיר סדר על הקדקודים להדפסה אחידה בכל ההרצות.

במחלקות אלו בנאי אחד ללא פרמטרים, המאתחל גרף ריק.

חישבו איך אפשר לממש את שני סוגי הגרפים (מחלקות DirectedGraph ו- UndirectedGraph) בצורה יעילה ללא שכפול קוד.
אם יש צורך הוסיפו מחלקות ומתודות, ובלבד שתקיימו את הדרישות בתרגיל.

במימושכם שימו לב לשימוש בעקרונות תכנות מונחה עצמים, והשתמשו בדברים שנלמדו לפי הצורך (final,static,abstract,overloading,overriding, שרשור בנאים, ועוד).

בשלב זה, הניחו כי הארגומנטים המתקבלים במתודות ובבנאים תקינים.

שלב ב': שימוש בגרפים

השלימו את הגדרת המחלקה Person, כך ש-equals,hashcode יהיו תלויות רק ב-id,
ובנוסף, ההשוואה תהיה לפי id כך שמיון לפי השוואה זו יהיה לפי id בסדר עולה.

שלב ג': בדיקות לגרפים

הריצו את שתי מחלקות הבדיקות של הגרפים:

GraphTestDirected, GraphTestUndirected

שלב ד': שאלות על המימוש

1. תנו דוגמה בתרגיל ליחס הורשה, ליחס הפשטה, ליחס הכלה, לדריסה (overriding) של מתודה, לפולימורפיזם, ולהעמסה (overloading) של מתודה.
אם אין לכם דוגמה לאחד מן הדברים, ציינו זאת.

2. האם ניתן להגדיר יחסי הורשה בין המחלקות DirectedGraph ו- UndirectedGraph? הסבירו. באיזו דרך בחרתם להשתמש.

3. אנו מעוניינים שקטע הקוד הבא ידפיס את המספר "1".
האם זה הפלט של הרצת קטע הקוד על המימוש שלכם? הסבירו.

```
Set<IGraph<String>> setGraphs = new HashSet<IGraph<String>>();
setGraphs.add(new DirectedGraph<String>());
setGraphs.add(new DirectedGraph<String>());
System.out.println(setGraphs.size());
```

אם לא, הסבירו מה יש לשנות/להוסיף בקוד שלכם על מנת שקטע קוד זה אכן ידפיס 1.
<u>אין צורך לממש, אלא רק לתאר אלו שינויים ועדכונים בקוד נדרשים.</u>

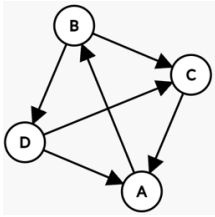
חלק ב': קלט ומערך

שלב א': מקדים

הוסיפו למחלקה GraphUtils מתודה המקבלת מחרוזת המתארת גרף שקדקודיו הם מטיפוס String. המחרוזת המתארת את הגרף, היא באותו הפורמט כמו תוצאת toString על הגרף.
המתודה הזו מחזירה אובייקט מטיפוס IGraph<String>.
למשל: אם נקבל את המחרוזת הבאה (\t מתאר הוספת רווח טאב - tab):

```
DirectedGraph:\tA:{B} B:{C,D} C:{A} D:{A,C}
```

אז יוחזר גרף מכוון כדלהלן:



בנאי זה יכול להיעזר ב- Scanner עם שימוש במתודה useDelimiter("\\t") המוגדרת במחלקה Scanner. ניתן לראות הסבר על מחלקה זו בסוף התרגיל.
בשלב זה, הניחו כי הקלט תקין.

שלב ב': קלט

בחלק זה המערכת תקלוט רשימה של גרפים מתוך קובץ לפי פורמט מיוחד, ותבצע מספר דברים.
בסיום החלק הזה מופיע הסבר ועזרה לניהול קלט-פלט עבור התרגיל.

קובץ הקלט הינו קובץ טקסט, בו כל שורה מגדירה גרף אחד ששמות הקדקודים בו הם מטיפוס String.
כל שורה היא מחרוזת המתארת גרף באותו הפורמט כמו תוצאת toString על אובייקט גרף כזה.

הגדירו מחלקה GraphsHandler ובה מתודה main אשר מקבלת כארגומנט ראשון את שם קובץ הקלט (כולל הנתיב שלו). תפקידה של מתודה זו לקרוא את קובץ הקלט, לבנות את כל הגרפים על פי ההוראות שבו, לשמור אותם במיכלים ואז להדפיס את כל הגרפים באופנים שונים.
כל גרף יודפס בשורה נפרדת. המחרוזת המתארת את הגרף מתקבלת מהפעלת מתודת toString() על הגרף.
בשלב זה ניתן להניח כי הפרמטרים הנשלחים ל-main תקינים, וכן שפורמט הקובץ תקין, כולל הערכים בפנים.

נעשה שימוש במספר סוגים של מיכלים:

```
1. List<IGraph<String>> list
2. SortedSet<IGraph<String>> sortedSet;
```

יש לקרא את הקובץ פעם אחת, ותוך כדי הקריאה למלא את המיכלים האלו באופן הבא.
list: יכיל את כל הגרפים בסדר הפוך לסדר קריאתם.
set: יכיל את כל הגרפים בסדר עולה לפי מספר הצמתים בגרף. שני גרפים עם אותו מספר צמתים יופיעו לפי סדר קליטתם.

אסור לקרוא את הקובץ יותר מפעם אחת, וכן אסור להגדיר מיכלים נוספים עבור הקלט (מלבד לטיפול בחריגות – ראה חלק 4 בתרגיל).

שלב ג': כתיבת קבצי פלט

1. הדפסת כל הגרפים לפי סדר הפוך מהופעתם בקובץ הקלט. הדפסה זו תעשה ע"י ה-list:

```
for (IGraph<String> Graph : list) {
      //write a line with Graph to the file
}
```

הדפסה תהיה לקובץ GraphsOutList.txt

שימו לב כי עליכם להגדיר נכונה את list כך שהסדר הנכון אמור להתקבל מהדפסה באמצעות לולאות for דלעיל.

2. הדפסת כל הגרפים בסדר עולה לפי מספר הקדקודים בגרף.
הדפסה זו תעשה שתי פעמים בשני אופנים שונים:
   a. ע"י ה-sortedSet. הדפסה זו היא לקובץ בשם GraphsSortOutSet.txt
   b. ע"י ה-list. הדפסה זו היא לקובץ בשם GraphsSortOutList.txt

אם הגדרתם נכון את ה-set אז הסדר הנכון אמור להתקבל מהדפסה באמצעות לולאות for כדלהלן:

```
for (IGraph<String> Graph : sortedSet) {
      //write a line with Graph to the file
}
```

לגבי ה-list. המטרה היא לשנות את הסדר ב-list כך שהפלט של הלולאה

```
for (IGraph<String> Graph : list) {
      //write a line with p to the file
}
```

יהיה על פי הסדר כנדרש.
אין להגדיר מערכי עזר או אוספי עזר נוספים עבור הקלט.

ניתן לבצע זאת באמצעות שתי מתודות במחלקה Collections:
   a. `Collections.reverse(List<?> list)` - הופכת את הסדר ב-list.
   b. `Collections.sort(List<T> list, Comparator<? super T> c)` ממיינת את ה-list.
   לצורך מיון זה, נשלח מימוש למנשק Comperator<IGraph<String>>.

קבצי קלט-פלט לדוגמה מופיעים באתר.
קבצי הטקסט כולם יהיו באותה תיקייה של קבצי ה-java.

בהמשך הקורס, נלמד על קלט-פלט בצורה יסודית, וכן על טיפול בשגיאות - קובץ לא נמצא וכד'.
בתרגיל זה עליכם להשתמש בקלט-פלט על פי ההוראות דלהלן.

הסבר והוראות עבור השימוש במחלקה Scanner ובקלט:

כדי להשתמש במחלקה Scanner יש לכתוב בראש הקובץ import java.util.Scanner;

כדי לקרא מקבוץ נבצע Scanner sc = new Scanner (new File(filename.txt));

וכדי "לקרא" ממחרוזת str נבצע Scanner sc = new Scanner (str);

ניתן לקרא שורה אחר שורה באופן הבא:

```
while (sc.hasNextLine())
        String line = sc.nextLine();
```

בסיום השימוש באובייקט מטיפוס Scanner יש לסגור אותו ע"י הרצת sc.close();

המחלקה Scanner יכולה לסייע גם בקריאת הפרמטרים שבשורה המופרדים באמצעות תווים מיוחדים. אם מעבירים ל-Scanner את המחרוזת עצמה, אפשר "לקרוא" אותה בדומה לקריאת קובץ, באמצעות המתודות, hasNext() ו-next(). תיעוד המחלקה מופיע בקישור הבא:
https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

כדי לפענח את הקלט ולבנות את הגרפים, סדר הדברים הוא כזה:

א. יש לקרוא את קובץ הקלט שורה אחר שורה.
את קובץ הקלט ניתן לקרוא באמצעות המחלקה Scanner.
במתודה ה-main ניתן לרשום: Scanner sc = new Scanner (new File(filename.txt));
כאשר fileName הוא מחרוזת המכילה את שם קובץ הקלט, למשל "GraphsIn.txt",
והקובץ נמצא בתיקייה שבה נמצאים קבצי ה-java.
אם מתקבלת הודעה שגיאה על קובץ לא קיים, ניתן לנסות לבצע את השורה הבאה במקום השורה הקודמת: Scanner sc = new Scanner(new File("./src/"+ filename.txt));
בנוסף נוסיף את שורות הייבוא הבאות:

```
import java.io.File;
import java.io.FileNotFoundException;
```

כדי לקרוא שורה מהקובץ ולשים את תכנה במשתנה line אפשר לכתוב:

```
String line;
while (sc.hasNextLine())
        line = sc.nextLine();
```

(אם מתעוררת בעיה הנוגעת ל-unhandled exception,
תנו ל-eclipse לפתור לכם את הבעיה: לחצו על add throws declaration.
כלומר נקבל כי חתימת המתודה main בינתיים היא כדלהלן:

```
public static void main(String[] args) throws FileNotFoundException
```

ב. ניתוח וקריאה של השורה. אתם רשאים לפתור זאת בכל דרך הנראית לכם נכונה, אך גם כאן מומלץ להשתמש במחלקה Scanner כפי הדוגמה לעיל.

הסבר והוראות עבור השימוש ב-פלט בתרגיל:

בחלק זה נשתמש במחלקה Writer. כדי להשתמש במחלקה זו יש לכתוב בראש הקובץ:

```
import java.io.Writer;
import java.io.FileWriter;
```

במתודה ה-main ניתן לרשום:

```
Writer wr = new FileWriter ("GraphsOut.txt");
```

כתיבה נעשית באמצעות המתודה write למשל:

```
wr.write("Hello");
wr.write("\n"); //moves the cursor to a new line
```

בסיום יש לבצע שתי שורות

```
wr.flush();
wr.close();
```

כמו-כן, מימוש זה ידרוש שינוי של החתימה של המתודה main:

```
public static void main(String[] args) throws IOException
```

אם הקובץ נכתב אצלכם לתיקייה אחרת, נסו את השורה הבאה:

```
Writer wr = new FileWriter ("./src/"+"GraphsOut.txt");
```

במקום השורה המופיעה לעיל.

חלק ג' –טיפול בחריגות

בחלק זה נוסיף טיפול במקרי שגיאה בנתונים שבקובץ באמצעות חריגות (ירושה מ-Exception).

סוגי השגיאות בהן יש לטפל, למשל:

1. שורה בה הסוג של הגרף אינו DirectedGraph ואינו UndirectedGraph.
2. שורה בפורמט אחר מהנדרש, (למשל, סוג הגרף, ותיאור הגרף מופרדים באמצעות ":" במקום באמצעות \t)
3. שורה בה המחרוזת לתיאור הגרף אינה נכונה בגלל תוספת של פרמטרים, הפרדה לא נכונה, פורמט לא מדויק.
4. אין צורך לטפל בחריגות אחרות.

הוסיפו למחלקות שכתבתם בחלקים הקודמים טיפול בשגיאות אלו.
המטרה שעבור כל שורת קלט לא תקינה – תיזרק חריגה מטיפוס HW3Exception (מחלקה שתגדירו בתוך תיקיית ה-src).
התכנית לא תעצור בגלל שורות לא חוקיות. התכנית תדלג על שורה שבגינה נזרקה חריגה ותמשיך לקרא את הקובץ ולטפל בשורות הבאות על אף שמתגלות חריגות בחלק מהשורות.

התכנית תוציא כפלט קובץ טקסט עם כל החריגות.
הקובץ יכיל את כל השורות הבעייתיות עם פרטים על השגיאה שנוצרה בעקבות זאת (תיאור השגיאה שנוצרה ומספר שורה בקובץ הקלט).
שם הקובץ יהיה errorsGraphs.txt
שאר הדברים, כמו הארגומנטים הנשלחים ל-main וכד', ניתן להניח כי הם תקינים.
באתר ישנה דוגמה לקובץ קלט ולקבצי פלט ושגיאות המתאימים לקלט.
קבצי החריגות שלכם צריכים להיות דומים (לא חובה שיהיו זהים) לקבצים שבדוגמה.

בהצלחה!