תוכנה 1 – חורף 2023/24

תרגיל מספר 6

collection framework, design של תוכנה

<u>הנחיות כלליות:</u>

קראו בעיון את קובץ נהלי הגשת התרגילים אשר נמצא באתר הקורס.

את התרגיל הבא צריך להגיש באופן הבא:

- הגשה במערכת ה-Git תתבצע על פי ההנחיות שראיתם בתרגול 1.
  צרו את ה repository שלכם מתוך הקישור הבא:
  https://classroom.github.com/a/2y0o2VbV

  יש לוודא שבתיקיית הגיט שלכם נמצאים הקבצים הבאים:
  a. קובץ פרטים אישיים בשם details.txt המכיל את שם המשתמש שלכם ב Moodle ואת מספר תעודת הזהות שלכם.
  b. 2 תיקיות src ו-resources. יש להגיש את ה-repository ואת מבנה התיקיות שבתוכו בדיוק באותה היררכיה שקיבלת אותם

- הגשה במערכת ה Moodle (http://moodle.tau.ac.il/): עליכם להגיש את קובץ הטקסט assignment.txt ובו קישור לgit repository האישי שלכם.

<u>הנחיות כלליות נוספות לתרגיל:</u>

1. מומלץ ראשית לקרוא את כלל ההוראות עבור החלק שאתם ניגשים לפתור על מנת לוודא שהמימוש שלכם יהיה מותאם בצורה הטובה ביותר לדרישות התרגיל.
2. בכל אחד מחלקי התרגיל ניתן להוסיף שירותים ומחלקות לפי הצורך, אך אין לשנות חתימות של שירותים קיימים והגדרות של מנשקים (גם אין להוסיף או להוריד throws).
3. אין הגבלות על import-ים או שימוש במבני נתונים מסוימים, אך שימו לב שאתם מוחקים import-ים מיותרים או כאלה שאינם מתקמפלים בנובה.
4. בכל חלק קיים טסטר קצר המבצע בדיקות שפיות (חפשו בגוגל sanity tests). כדאי ומומלץ להוסיף בדיקות משלכם שכן הטסטרים הם בסיסיים ביותר ולא בודקים את כל המקרים.

חלק א' (50 נק')

בתרגיל זה עליכם לממש מבנה נתונים של היסטוגרמה באמצעות אוספים גנריים. נגדיר היסטוגרמה בתור מבנה נתונים אשר סופר מופעים של עצמים מטיפוס T כלשהו (טיפוס גנרי). הקוד ימומש בחבילה `.il.ac.tau.cs.sw1.ex6.histogram`

לדוגמא, עבור אוסף האיברים הבא: 1, 2, 3, 1, 2, ההיסטוגרמה תכיל את האיברים 1, 2, 3 ואת מספר המופעים שלהם.

יחד עם קבצי התרגיל מסופק לכם הממשק Histogram את השירותים הבאים (כולל החוזה של כל שירות):

```
public void addItem(T item);
public boolean removeItem(T item);
public void addAll(Collection<T> items);
public int getCountForItem(T item);
public void clear();
public Set<T> getItemsSet();
public int getCountsSum();
```

א. השירות `addItem` מוסיף מופע אחד של הפריט item להיסטוגרמה.

ב. השירות removeItem מוריד מופע אחד של הפריט item. השירות מחזיר true אם הפריט הוסר, ו false אם הפריט לא היה קיים, ולכן גם לא הוסר.

ג. השירות `addAll` מוסיף אוסף של פריטים להיסטוגרמה (האוסף יכול להכיל את אותו הפריט יותר מפעם אחת. בנוסף, לא ניתן להניח דבר על הקיום של פריטים אלה בהיסטוגרמה לפני הקריאה ל addAll).

ד. השירות getCountForItem יחזיר את מספר הפעמים שהאיבר item נספר. אם item הוא פריט שלא קיים בהיסטוגרמה, יוחזר הערך 0.

ה. השירות `clear` ירוקן את ההיסטוגרמה מכל האיברים והספירות (כלומר, לאחר clear, השירות `getCountForItem` יחזיר ספירה 0 לכל איבר).

ו. השירות `getItemsSet` יחזיר אוסף מטיפוס Set אשר מכיל את כל האיברים בהיסטוגרמה אשר מספר המופעים שלהם גדול מ-0, ללא הספירות שלהם.

ז. השירות getCountsSum יחזיר את סכום המופעים של איברי ההיסטוגרמה.

בנוסף לשירותים שפורטו כאן, המנשק Histogram מממש את Iterable, כך שתצטרכו לממש את הפונקציה iterator כפי שיוסבר בהמשך.

סעיף 1 (25 נק'):

ממשו את המחלקה HashMapHistogram אשר מממשת את המנשק Histogram עבור כל טיפוס T המממש את המנשק Comparable (כלומר, T יכול לקבל ערך של כל מחלקה המממשת את המנשק Comparable. נזכיר כי הטיפוסים המובנים הבסיסיים כמו Integer ו String מממשים מנשק זה). לדרישה הזו יש סיבה אותה ניראה בהמשך.

פרקטית, זה אומר שנגדיר את HashMapHistogram באופן הבא:

```
public class HashMapHistogram<T extends Comparable<T>> implements
Histogram<T>
```

משמעות הגדרה הזו: המנשק Histogram מצריך פרמטר גנרי. את הפרמטר הגנרי נגדיר כ T, ונוסיף עליו את האילוץ שהוא צריך לממש את המנשק Comparable<T>. כלומר, T יהיה פרמטר גנרי מתאים אם הוא Comparable עם עצמים אחרים מטיפוס T. את התחביר הזה ניראה בהמשך הקורס.

המימוש יעשה באמצעות הכלה (aggragation) של HashMap, כלומר, כל מופע של HashMapHistogram יכיל שדה מטיפוס HashMap. שדה זה יהיה אחראי על שמירת הספירות עבור כל אובייקט מטיפוס T.

סעיף 2 (25 נק')

המנשק Histogram יורש מהמנשק Iterable, מה שמחייב את HashMapHistogram לממש את השירות iterator(). הגדרת המנשק היא:

```
public interface Histogram<T> extends Iterable<Map.Entry<T, Integer>>
```

כלומר, האיטרטור שההיסטוגרמה אמורה להחזיר עובר על זוגות (מטיפוס Map.Entry). הזוגות מייצגים מפתחות וערכים בהיסטוגרמה. תזכורת - השירות entrySet של המחלקה Map מחזיר אוסף של איברים מטיפוס Map.Entry.

האיטרטור יחזיר את הזוגות על פי הסדר הטבעי של המפתחות (האיברים מטיפוס T). לדוגמא – אם ההיסטוגרמה מכילה מחרוזות, הסידור יהיה לפי סדר לקסיקוגרפי עולה. אם ההיסטוגרמה מכילה מספרים שלמים, הסידור יהיה סידור המספרים בסדר עולה.

לצורך כך עליכם לממש:

א. מחלקה חדשה המממשת את המנשק Iterator. שם המחלקה הוא HashMapHistogramIterator והשלד שלה נתון לכם. אין צורך לממש את השירות remove.
ב. כדאי לממש גם Comparator בשביל סידור האיברים שהאיטרטור יחזיר. ניתן לממש מחלקה זו כמחלקה פנימית במחלקת האיטרטור או כמחלקה בקובץ Java נפרד משלה.

שימו לב, ניתן, ואף כדאי, להעביר למחלקות ה-Comparator וה-Iterator את המידע הרלוונטי מתוך המופע של HashMapHistogram וכן להשתמש בהכלה של אוספים לפי הצורך.
להבין איזה מידע כל אובייקט צריך לקבל הוא חלק מהאתגר מהתרגיל, חישבו על כך בעת בניית המחלקות.

שלד כל המחלקות אותן אתם נדרשים לממש נתון לכם בחבילה il.ac.tau.cs.sw1.ex8.histogram המופיעה בקבצי התרגיל.

העזרו ב HashMapHistogramTester בשביל לבדוק את עצמכם, והוסיפו לו בדיקות משלכם.

חלק ב' (30 נק')

בחלק זה נתרגל עבודה עם אוספים (Collections) . הקוד ימומש בחבילה

`il.ac.tau.cs.sw1.ex6.collections` במחלקה `CollectionsExercise` שם כבר יש לכם את חתימות הפונקציות הרלוונטיות.

<u>סעיף 1 (10 נק')</u>

ממשו את השירות processDicts. השירות מקבל רשימה של Map-ים שבהם המפתח הוא Character והערך הוא Set<String>.

בכל מילון המפתחות הם תווים, וכל תו ממופה לset של מחרוזות. נאמר שתו c ממופה למחרוזת s במילון מסויים אם הזוג c_set:'c' מופיע במילון וגם s היא איבר ב c_set.

השירות יחזיר Map שיענה על הדרישות הבאות:

1. המפתחות בפלט יהיו איחוד מפתחות המילונים ברשימת הקלט.
2. הערך עבור מפתח c כלשהו יהיה המספר המקסימלי k של מילונים בהם c מופה למחרוזת מסויימת s.

לדוגמה, עבור הרשימה mapsList שבה שלושה מילונים.

mapsList = [{'a': {"tt"}, 'c': {"gg"} } , { 'a': {"kk", "mm" }, 'b': {"hh"} } , { 'a':{"kk"}, 'b': {"yy"} }]

הפונקציה תחזיר את המילון {'a': 2, 'b': 1, 'c': 1}. שלושת המילונים ב mapsList מכילים שלושה מפתחות שונים ('a', 'b', 'c'). עבור 'a' – המחרוזת "kk" מופיעה בקבוצה הממופה ל 'a' בשני מילונים שונים. 'b' אמנם מופיע בשני מילונים שונים, אך המחרוזות המופיעות בקבוצות הממופות ל 'b' הן שונות בין שני המילונים.

<u>סעיף 2 (10 נק'):</u>

בהנתן lst, רשימת מחרוזות המייצגות משפטים, נרצה לבנות לכל מילה x המופיעה במשפטים ב lst, <u>מילון אשר מכיל את כל המילים שהופיעו מיד אחריה במשפט כלשהו</u>, ואת מספר הפעמים שאותה המילה הופיעה אחרי x.

לדוגמה, עבור הרשימה הזו: lst = ["I love java", "java is the best", "love java is love"]

אחרי המילה love הופיעה פעמיים המילה java. אחרי המילה java מופיעה פעמיים המילה is (המילה java אמנם מופיעה שלוש פעמים, אבל פעם אחת מהן היא היתה המילה האחרונה במשפט). אחרי המילה is מופיעות פעם אחת שתי מילים שונות: the, love.

ממשו את השירות analyzeText אשר מקבל רשימת משפטים lst ומחזיר מילון שבו המפתחות הם כל המילים שהופיעו במשפטים, והערכים הם מילונים שמכילים את כל המילים שהופיעו אחרי המפתח, ואת מספר המופעים שלהם. עבור מילה שהופיעה רק בסוף משפט הערך המתאים יהיה מילון ריק. דוגמת שימוש:

```
Map<String, Map<String, Integer>> res = analyzeText(lst);

System.out.println(res.get("love")); //{"java" : 2}

System.out.println(res.get("is")); // {"the": 1, "love": 1}

System.out.println(res.get("best")); // {}
```

הניחו כי הקלט חוקי וכל מחרוזת ב lst רק מילים המכילות אותיות, ובין כל שתי מילים מפריד רווח בודד. ניתן להניח כי כל מחרוזת ב lst מכילה לפחות מילה אחת.

<u>סעיף 3 (10 נק'):</u>
ממשו את הפונקציה weirdSorting אשר מקבלת רשימה (List) של אובייקטים מטיפוס Integer ומחזירה רשימה חדשה שהיא הרשימה המקורית, ממוינת לפי cos שארית החלוקה במספר ראשוני נתון p (שנתון כקלט לפונקציה). כלומר, מספר a גדול ממספר b אם ורק אם cos שארית החלוקה של a ב p-גדול מ cos שארית החלוקה של b ב p . אין לשנות את המערך המקורי.

בסעיף זה ניתן להשתמש ב Collections.sort בלבד.

חלק ג' (20 נק')

בחלק זה נתרגל עיצוב (Design) של תוכנה. בשאלה זו אין צורך לספק מימושים. אנו נרצה לראות את מבנה המחלקות והמנשקים שיצרתם, את חתימות הפונקציות, הנראות שבחרתם לכל פונקציה והאם הפונקציה היא פונקציה דורסת. אמנם אין כאן דרישה למימוש, אבל כדי להימנע משכפול קוד עתידי נרצה שתבחרו בקפידה בסוג המחלקות והמנשקים. בפרט, אם הגיוני לבחור במחלקה אבסטרקטית הגדירו כזו. עבור כל בחירת design שעשיתם שנראית לכם מהותית/עקרונית מספיק, פרטו בהערות של הפונקציה/מחלקה/מנשק מדוע ביצעתם דווקא את הבחירה הזו. מותר להוסיף מנשקים ומחלקות אחרים שלא קיימים במפורש בתיאור התרגיל, כמו גם להניח שהם קיימים בחוץ. על התוצר הסופי שלכם לשבת בחבילה `il.ac.tau.cs.sw1.ex6.design`.

עליכם לעצב את מודול ההודעות של תוכנת chat כלשהי (דוגמת `Whatsapp, Telegram` וכדומה).

- על התוכנה (כרגע) לתמוך בסוגי ההודעות הבאים: מלל (טקסט), תמונה, הודעות קוליות, סרטונים ומסמכים.
- על הdesign לאפשר הוספה עתידית של סוגים נוספים של הודעות.
- הניחו שהbackend של התוכנה כבר קיים. כלומר יש מודול (Module) שמעבד ומציג את ההודעות למסך (אין צורך להגיש אותו). הוא מצפה למצוא בכל הודעה את נתוני המסגרת הבאים: שולח ההודעה, מען ההודעה וזמן השליחה.
- הניחו שקיים מנשק/מחלקה `User` וזה הטיפוס שמייצג את שולח ההודעה ומען ההודעה.
- זמן השליחה מיוצג כמחרוזת.
- כל הודעה תכיל פונקציות `getter` לנתוני המסגרת.
- כמובן שלכל עצם שמייצג הודעה יש שדה שמחזיק את המידע (התוכן) של ההודעה הספציפית. נתעלם בשאלה זו מתחזוקת התוכן וייצוגו.
- כל הודעה תכיל פונקציית `draw` שמציגה את ההודעה על המסך.[1] כדי להציג הודעה על המסך, אפשר ורצוי להשתמש במנשק `IDrawable` (שנתון לכם בשלד). הלוגיקה להצגת כל סוג של הודעה היא ייחודית לסוג ההודעה
- כל הודעה ניתנת למחיקה. הלוגיקה למחיקה זהה לכל ההודעות.
- כל הודעה תהיה ניתנת להורדה. הלוגיקה להורדת כל סוג של הודעה היא ייחודית לסוג ההודעה.
- הודעות קוליות וסרטונים ניתן לנגן. הלוגיקה לניגון כל סוג של הודעה כזאת היא ייחודית לסוג ההודעה.
- כל הודעה ניתנת להצפנה בפרוטוקול DES אבל הודעה מסוג "מסמך" נצפין בפרוטוקול AES.

[1] הדרישות בתרגיל הן הפשטה מאוד גסה של המציאות. בפועל כתיבה של תוכנית כזו מסובכת בהרבה. כתרגיל מחשבתי, ממליצים לחשוב כיצד תוכנה כזו תמומש בפועל.

בהצלחה!