תרגיל בית מס. 3
מימוש תאריכים ושעות
פולימורפיזם, static, final, immutable
מערכים, אוספים, equals, compare,
חריגות ותיעוד

להגשה עד יום חמישי 24.12 למניינם
ההגשה בזוגות במוודל עד השעה 23:00
משקל התרגיל: 7 נק'

התרגיל עוסק במימוש תאריכים ושעות (חלק ראשון), מערכים ואוספים (חלק שני), חריגות (חלק שלישי) ותיעוד קוד (חלק רביעי).
התרגיל כולל מימוש ושאלות על המימוש (בחלקים א' ו-ב'). יש לממש חלק אחר חלק, שכן כל חלק מתייחס לחלקים הקודמים.
במימוש החלק הראשון, נשתמש ב-Time שהכרנו בתרגיל בית קודם, ונתונה לכם עכשיו באופן קצת שונה (מחלקה Time נתונה לכם עם מימוש).
אין לשנות את המחלקות והמנשקים הנתונים לכם.

<u>אופן ההגשה:</u>
הורידו את הקבצים הנתונים במטלה במוודל, ושימו אותם בפרוייקט חדש שהכנתם תחת התקייה הראשית src.
כל המחלקות שתרשמו בפרוייקט יהיו תחת התקייה הראשית src אלא אם כן יש הוראות מיוחדות אחרות (ראו חלק ג').
יש לארוז בקובץ zip אחד את כל קבצי ה-java שכתבתם וקובץ pdf אחד ובו תשובותיכם לשאלות הפתוחות. שם הקובץ צריך להיות: 28_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, והוא יהיה כלול בתוך ה-zip.
שימו לב כי הקבצים מחלק ג' שנמצאים בתת תקייה צריכים להופיע עם תת התקייה שלהם.
כלומר כאשר פותחים את ה-zip המוגש נקבל תקייה 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.
הבדיקות יתפרסמו בהמשך, ועל כך תבוא הודעה.

<u>חלק ראשון: מימוש המנשק IDate והמחלקה DateTime</u>

<u>בחלק זה עליכם תחילה לממש את המנשק IDate הנתון.</u>

עליכם להגדיר 2 מחלקות הממשות את IDate.
1. GregorianDate -עבור מימוש של תאריך גרגוריאני.
2. JewishDate -עבור מימוש של תאריך עברי.

המתודה toString – עבור GreorgianDate תדפיס:
Gregorian date dd month yyyy
כאשר dd מתאר את היום בחודש בשתי ספרות, month מתאר את החודש בקיצור (*), ו-yyyy מתאר את השנה ב-4 ספרות.
(*)החודשים ירשמו כך (כולל הנקודה): Jan. Feb. Mar. Apr. May. Jun. Jul. Aug. Sep. Oct. Nov. Dec.

ועבור JewishDate תדפיס:

Jewish date dd month yyyy

כאשר החודשים ירשמו כך (כאשר מספר החודש הוא 1-12 לפי הסבר הבא משמאל לימין):

Nisan, Iyar, Sivan, Tamuz, Av, Elul, Tishrei, Heshvan, Kislev, Tevet, Shvat, Adar

אם מדובר בשנה מעוברת אזי יודפס Adar A (חודש 12), ו- Adar B (חודש 13).

לרשותכם יש את המחלקה **DateUtils** – ובה מתודות עזר שונות לשימושכם. אין לשנות מחלקה זו.

במחלקות GregorianDate ו-JewishDate יהיו מספר בנאים כדלהלן (נפרט את הבנאים של GregorianDate, הבנאים של JewishDate יוגדרו באופן דומה).

```
1. public GregorianDate(int day, int month, int year)
2. public GregorianDate(GregorianDate date)
3. public GregorianDate(String date)
```

הבנאי השלישי מקבל מחרוזת מהפורמט "dd/mm/yyyy". כאשר dd,mm,yyyy הם מספרים המתארים את התאריך (יום,חודש,שנה). הבנאי בונה את האובייקט לפי זה. בנאי זה יכול להיעזר ב-Scanner עם שימוש במתודה useDelimiter("/") המוגדרת במחלקה Scanner. ניתן לראות הסבר על מחלקה זו בהמשך התרגיל, וכן יש דוגמה לשימוש בזה במחלקה Time הנתונה לכם.

חישבו איך אפשר לממש את הדברים 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).

רמז: כדאי להגדיר מחלקה נוספת– שממנה ירשו GregorianDate ו-JewishDate.
המטרה "לתפוס" את המשותף שבמחלקות GregorianDate ו-JewishDate במחלקה אחת.

<u>כעת השלימו את מימוש המחלקה DateTime</u>

מחלקה זו מתארת **<u>מועד</u>** - תאריך עם שעה.
נתון לכם שלד של המחלקה DateTime. השלימו את המימוש שלה.
שימו לב שבבנאי המקבל כפרמטרים date ו-time, אין להעתיק את המצביעים date ו-time, אלא לבצע העתקה עמוקה.
המתודה toString תחזיר את התאריך עם השעה לפי הפורמט – date.toString, time.toString, כאשר date.toString זה המחרוזת של date המתקבלת מהפעלת toString שלה, ובדומה time.toString עבור החלק של השעה.

<u>שאלות - החלק השני:</u>

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

```
Set<IDate> setDates = new HashSet<IDate>();
setDates.add(new GregorianDate(10,10,2020));
setDates.add(new GregorianDate(10,10,2020));
System.out.println(setDates.size);
```

אם לא, הסבירו מה יש לשנות/להוסיף בקוד שלכם על מנת שקטע קוד זה אכן ידפיס 1.
אין צורך לממש אצלכם.
4. אובייקט נקרא Immutable אם הוא לא ניתן לשינוי אחרי יצירתו.

א. אובייקט ניתן לשינוי אחרי יצירה דרך מתודות המיועדות לכך (למשל set) או דרך שדות בעלי הרשאה שאינה פרטית. חישבו על דרכים נוספות לשינוי אובייקט אחרי יצירה (היזכרו בתרגיל בית 2).

ב. מבין המחלקות שמימשתם עד עתה, מאלו מחלקות ניתן ליצור אובייקטים ניתנים לשינוי ומאלו מחלקות ניתן ליצור אובייקטים שאינם ניתנים לשינוי?
ג. מה היתרון בתכונת Immutable? ומהו החיסרון בכך?

חלק שני: קלט ומערך

בחלק זה המערכת תקלוט רשימה של תאריכים מתוך קובץ לפי פורמט מיוחד, ותבצע מספר דברים.
כמו-כן הנכם נדרשים לענות על שתי שאלות לגבי המימוש (מופיעות בסוף החלק).
בסיום החלק הזה מופיע הסבר ועזרה לניהול קלט-פלט עבור התרגיל.
קבצי קלט-פלט יפורסמו בהמשך.
קובץ הקלט הינו קובץ טקסט, בו כל שורה מגדירה תאריך אחד.
הפרמטרים בשורה מופרדים באמצעות #, והסדר הוא כדלהלן:
האות הראשונה בכל שורה אומרת באיזה סוג מדובר: g (-gregorian), j (-jewish)
אח"כ # מפריד.
אח"כ, נקבל את ערכי התאריך בפורמט dd/mm/yyyy.
אח"כ עם נרצה להגדיר גם שעה ולא רק תאריך, יופיע עוד מפריד #, ולאחריו תופיע השעה בפורמט: hh:mm:ss כאשר השעה נספרת מ- 0-23.

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

נעשה שימוש בשלשה סוגים של מיכלים:

```
1. DateTime[] arr
2. List<DateTime> list
3. SortedSet<DateTime> set;
4. Map<DateUtils.DateType, Set<DateTime>> map;
```

שלב א': קלט

יש לקרא את הקובץ פעם אחת, ותוך כדי הקריאה למלא את שלשת המיכלים האלו באופן הבא.

arr: יכיל את כל המועדים לפי סדר קריאתם.
list: יכיל את כל המועדים בסדר הפוך לסדר קריאתם.
Set: יכיל את כל המועדים לפי סדר כרונולוגי מהמוקדם למאוחר.
map: המפתחות הם מסוג DateUtils.DateType ומכילים את כל הסוגים של תאריכים (לוח גרגוריאני, עברי וכו').

יתכנו מספר שורות בקובץ עם אותו ערך DateType, ולכן הערך עבור כל מפתח הוא **קבוצה** של מועדים שהיו בקובץ עם אותו סוג (גרגוריאני, עברי וכו'). **הקבוצה** של המועדים צריכה להיות ממויינת לפי סדר כרונולוגי מהמוקדם למאוחר.
אם התאריך מופיע בקלט ללא שעה – נתייחס לזמן הזה כאל שעה 00:00:00.

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

שלב ב': כתיבת קבצי פלט:

1. הדפסת כל המועדים לפי סדר הופעתם בקובץ הקלט. הדפסה זו תעשה ע"י ה-arr:

```
for (DateTime date : arr) {
      if (date==null) break;
      //write a line with date to the file
}
```

ההדפסה תהיה לקובץ datesOutArr.txt. שימו לב כי אינכם מדפיסים שורות null.

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

```
for (DateTime date : list) {//write a line with date to the file}
```

ההדפסה תהיה לקובץ datesOutList.txt

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

3. הדפסת כל המועדים **לפי הסדר הכרונולוגי**, מהמוקדם למאוחר. אם אותו מועד מופיע מספר פעמים, נדפיס עם אותו התאריך יודפסו אחד אחרי השני **על פי סדר הופעתם בקובץ הקלט**.
הדפסה זו תעשה שלש פעמים בשלשה אופנים שונים:
   a. ע"י ה-set. הדפסה זו היא לקובץ בשם datesSortOutSet.txt
   b. ע"י ה-list. הדפסה זו היא לקובץ בשם datesSortOutList.txt
   c. ע"י ה-arr. הדפסה זו היא לקובץ בשם datesSortOutArr.txt

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

```
for (DateTime date : set) {
      //write a line with date to the file
}
```

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

```
for (DateTime date : 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:
`Collections.reverse(List<?> list)` הופכת את הסדר ב-list.
`Collections.sort(List<T> list, Comparator<? super T> c)` מסדרת את ה-list.
לצורך מיון זה, נשלח מימוש מנשק Comperator.
הטיפול ב- arr דומה לטיפול ב-list, ע"י שימוש ב - Arrays.sort במקום ב-Collections.

4. הדפסת המועדים של הלוח הגרגוריאני לפי סדר כרונולוגי שלהם, ואח"כ הדפסת המועדים של הלוח העברי לפי סדר כרונולוגי שלהם.
אם הגדרתם נכון את ה-map אז הסדר הנכון אמור להתקבל מהדפסה באמצעות לולאות for כדלהלן:

```
for (Set<DateTime> setDatesType : map.values()) {
      for (DateTime date : setDatesType) {
            //write a line for date and time
      }
}
```

שם קובץ הפלט הזה יהיה datesSortOutMap.txt

קבצי קלט-פלט לדוגמה מופיעים באתר.
<u>קבצי הטקסט כולם יהיו באותה תיקייה של קבצי ה-java.</u>
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.
<u>הסבר והוראות עבור השימוש במחלקה Scanner ובקלט:</u>
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(). ניתן להשתמש גם במתודה nextShort() שקוראת את המספר הממשי הבא ומחזירה אותו כבר מוכן כ-short, וכן במתודה hasNextShort() על מנת לוודא שהערך הבא הוא short. יש מתודות דומות נוספות. תיעוד המחלקה מופיע בקישור הבא:
https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

כדי לפענח את הקלט ולבנות את המועדים, 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 "datesIn.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
```

בהמשך עבור הפלט נוסיף ונשנה חתימה זו להיות

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

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

<u>הסבר והוראות עבור השימוש ב-פלט בתרגיל:</u>
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 ("datesOut.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/"+"datesOut.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 משתמשים באות שונה מ- g,j לציון סוג התאריך.
2. שורה בפורמט אחר מהנדרש, למשל, סוג התאריך, התאריך והשעה מופרדים באמצעות ":" במקום באמצעות #), שורה בה הנתונים אינם מספריים כנדרש, שורה בה יש יותר מידי/פחות מידי נתונים, ועוד ועוד.
3. שורה בה המחרוזת לתיאור התאריך אינה נכונה בגלל תוספת של פרמטרים, הפרדה לא נכונה, פורמט לא מדויק.

הוסיפו למחלקות שכתבתם בחלקים הקודמים טיפול בשגיאות אלו.
המטרה שעבור כל שורת קלט לא תקינה – תיזרק שגיאה.
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.
יש שלש סוגי חריגות:

1. עבור מחרוזת תאריך (date) לא טובה - תיזרק חריגה מסוג InputDateException
2. עבור מחרוזת שעה (time) לא טובה - תיזרק חריגה מסוג InputTimeException
3. עבור שאר החריגות - תיזרק חריגה מסוג InputLineException

בשורה בה מופיעות מספר חריגות, נזרוק את החריגה מהסוג שמספרו קטן יותר. למשל, אם מופיעות חריגה מסוג 1 וגם חריגה מסוג 2 – תיזרק חריגה מטיפוס 1.

כל המחלקות האלו יהיו בתת-תיקייה בשם exceptionsTypes היושבת בתוך תיקיית ה-src הראשית.
התכנית תוציא כפלט קובץ טקסט עבור כל אחד מסוגי השגיאות. הקובץ יכיל את כל השורות הבעייתיות עם פרטים על השגיאה שנוצרה בעקבות זאת (תיאור השגיאה שנוצרה. מספר שורה בקובץ הקלט). שמות הקבצים יהיו כשמות המחלקה עם סיומת: txt.
שאר הדברים, כמו הארגומנטים הנשלחים ל-main וכד', ניתן להניח כי הם תקינים.
באתר ישנה דוגמה לקובץ קלט ולקבציי פלט ושגיאות המתאימים לקלט.
קבצי החריגות שלכם צריכים להיות דומים לקבצים שבדוגמה.

חלק ד' - תיעוד הקוד

ישנם שתי דרכים לתעד את הקוד. שתי הדרכים משמשות יחדיו **למטרות שונות**.
דרך אחת באמצעות java-doc – ראו תיעוד Time בתרגיל בית 2. תיעוד זה נחשף יחד עם חתימת המתודות ה-public. מטרת תיעוד זה היא להודיע למשתמש במתודות על אופן השימוש במתודות ומה מטרתן. במקרה של מתודה דורסת, ניתן להסתפק לפעמים בתיעוד כזה רק במחלקת האב (ואז מבצעים הפניה לתיעוד זה במחלקת הבן).
יש כלי אוטומטי ב- eclipse המוסיף Javadoc. סמנו את המתודה/משתנה/מחלקה בתוך קובץ ה-java ואז לחצן ימני בעכבר – Source ואז Generate Element Content. בד"כ קיצור המקלדת לכך הוא ctrl+alt+j. פעולה זו יוצרת Javadoc בסיסי לאלמנט הנבחר, אותו עליכם לערוך.
<u>במקרה של מתודה דורסת, פעולה אוטומטית זו יוצרת הפניה לתיעוד במחלקת האב.</u>

אופן התיעוד האחר הינו ע"י הוספת שורה שלמה או רק סוף שורה לאחר שני קווים נטויים, למשל:
IDate date; //this is a field which represents the ….
תיעוד זה נועד להסביר למתכנת חלקים קשים יותר בקוד, נקודות עדינות במימוש וכד'.

הוסיפו תיעוד מהסוג הראשון במחלקה: DateTime מחלק א'. הוסיפו תיעוד לכל המתודות והבנאים וכן תיעוד בראש הקובץ למחלקה.
בנוסף, תעדו באופן התיעוד השני את המחלקה DatesHandler מחלק ב'.
אין להגזים בתיעוד (בפרט לא בסוג השני). בחירת שמות נכונים, וארגון הקוד בצורה נכונה חוסכים המון שורות תיעוד מיותרות. אם יש צורך בתיעוד ארוך בכדי להבין את הקוד, סימן שהקוד אינו מאורגן בצורה מיטבית.

בהצלחה!