זמן המבחן 3 שעות + ½ שעה מנהלתית. יש לענות על כל 3 השאלות ע"פ ההוראות. בהצלחה.

לאחר הורדת המבחן ממערכת הבדיקות, פיתחו פרוייקט Java ובתוכו package בשם test. העתיקו לשם את קובצי המקור.

שאלה 1 – שימוש במבני נתונים, Stream ועיבוד נתונים (30 נק')

נתונה לכם המחלקה Employee (אשר אינה לעריכה או להגשה) המייצגת עובד. לכל עובד יש שם, גיל ומשכורת. בקובץ Q1.java עליכם לענות על הסעיפים א,ב,ג במתודות a,b,c בהתאמה.

א. אנו רוצים שבהינתן גיל ואות נקבל רשימה של כל העובדים בגיל הזה, ששם מתחיל באות הזו. נרצה שהרשימה תהיה ממוינת לפי משכורות מהקטנה לגדולה.
לפיכך עליכם לממש את המתודה a() כך שבהינתן Stream של עובדים היא תחזיר לנו Map<Integer, Map<Character, List<Employee>>>.
כלומר, מפה שהמפתח בה הוא גיל והערך הוא מפה שהמפתח שלה הוא אות והערך שלה הוא הרשימה הממוינת לפי משכורת. (15 נק')

ב. ממשו את המתודה b כך שבהינתן stream של עובדים, גיל a ואות c, המתודה תחזיר אמת אם מתוך כל העובדים שגילם גדול (ממש) מ a אין אף עובד ששמו מסתיים ב c. (10 נק')

ג. ממשו את המתודה c כך שבהינתן stream של עובדים היא תחזיר את העובד בעל השם הכי גדול לקסיקוגרפית. (5 נק')

מוד האימון ומוד ההגשה זהים (הקלט אקראי). שאלה זו **תיבדק באופן אוטומטי בלבד**, ולכן עליכם לוודא שהקוד מתקמפל ורץ ללא שגיאות גם אם לא עניתם או הצלחתם לענות על אחד הסעיפים.

שאלה 2 – תבניות עיצוב (40 נק')

סעיף א' (20 נק')

מהמחלקה Employee ניתן ליצור immutable objects. לכן עליכם ליצור את EmployeeBuilder שבאמצעותו נוכל לבנות עובדים באופן איטרטיבי בהתאם ל builder design pattern. הערכים הדיפולטיביים יהיו: שם העובד "default", גיל 18, משכורת 10000. ניתן יהיה לשנות את הערכים הללו באמצעות setters. הלקוח יכול יהיה להפעיל איזו מתודה שירצה ובאיזה סדר שירצה. כשירצה את האובייקט המוכן יקרא למתודה build.

דוגמאות:

```
Employee e=new EmployeeBuilder().build();
```

נוצר לנו אובייקט עם הערכים הדיפולטיביים לעיל.

```
Employee e3=new
EmployeeBuilder().setAge(20).setName("abc").setSalary(15000).build();
```

נוצר לנו עובד בשם "abc" בגיל 20 ובעל משכורת של 15000.

סעיף ב' (20 נק')

ברצוננו לחסוך בזיכרון ולא ליצור אובייקטים זהים של Employee. לכן הפעם עליכם להקפיד שהמתודה build תחזיר Employee חדש ע"פ הפרמטרים של שם, גיל ומשכורת רק אם הוא לא נוצר כבר בעבר (באמצעות EmployeeBuilder). אם נוצר בעבר עובד עם נתונים אלו המתודה build תחזיר רפרנס לעובד הקיים במקום ליצור חדש בדומה ל flyweight design pattern.

מוד האימון ומוד ההגשה זהים (הקלט אקראי). שאלה זו **תיבדק באופן אוטומטי בלבד**, ולכן עליכם לוודא שהקוד מתקמפל ורץ ללא שגיאות.

שאלה 3 – עיבוד מחרוזות, ביטויי למבדה (30 נק')

הגדרה: ביטוי בתצורה של CNF מורכב מפסוקיות (אחת או יותר) המחוברות ביניהם באמצעות פעולת "וגם" (AND). כל פסוקית היא אוסף של ליטרלים (אחד או יותר) המחוברים ביניהם באמצעות פעולת "או" (OR).

דוגמה למחרוזת CNF מתוך ה mainTrain3:

```
"(A == 5) AND (B <= 3 OR C<=10) AND (x ==10 OR eli==40 OR jhkgfjhg <=1000)";
```

כפי שניתן לראות כל פסוקית מוקפת בסוגריים כאשר בין הפסוקיות יש " AND ". בתוך כל פסוקית יש ביטוי בוליאני אחד או יותר המחוברים ע"י " OR ". ניתן להניח את ההנחות הבאות:

- קיים רווח בודד לפני ואחרי AND
- קיים רווח בודד לפני ואחרי OR
- בביטוי הבוליאני
  - שם המשתנה יורכב מאוסף אותיות גדולות ו\או קטנות
  - משתנה יופיע תמיד מצד שמאל וערך תמיד יופיע מצד ימין (למשל A==5)
  - הערכים תמיד יהיו מסוג Integer
  - הביטוי יורכב מ == (שוויון) או מ <= (קטן שווה) בלבד (לא צריך לתמוך בסימנים אחרים)
  - יתכנו רווחים לפני ו\או אחרי הסימן (למשל A==5, A ==5, 5 == A וכדומה)

בד"כ את ערכי המשתנים אנו שומרים ב symbol table. הפעם נשתמש פשוט במפה מ String ל Integer המשייכת את שם המשתנה לערכו.

במחלקה Interpreter עליכם לממש את המתודה interpret אשר בהינתן מחרוזת המכילה ביטוי בתצורת CNF היא תחזיר פרדיקט - אשר בהינתן ערכי המשתנים (מפת ה symbol table) הפרדיקט יחזיר אמת או שקר בהתאמה להאם ההשמה במשתנים מספקת את הביטוי.

לדוגמה:

```
String CNF = "(A == 5) AND (B<= 3 OR C <=10)";

Predicate<Map<String,Integer>> p = Interpreter.interpret(CNF);

Map<String,Integer> symTable=new HashMap<String, Integer>();
symTable.put("A", 5);
symTable.put("B", 4);
symTable.put("C", 10);

p.test(symTable); // true

symTable.put("A", 4);
symTable.put("B", 3);
symTable.put("C", 10);

p.test(symTable); // false
```

<u>טיפ:</u> ל String יש מתודה בשם trim() שמחזירה מחרוזת חדשה ללא white spaces.

מוד האימון מכיל בדיקות פשוטות, ואילו מוד ההגשה מכיל בדיקות עמוקות יותר. במידה ולא התבצעה בדיקה אוטומטית בשל שגיאת ריצה או קומפילציה תתבצע בדיקה ידנית עם קנס של 10 נק'.

הגשה

עליכם להיכנס למערכת הבדיקות למבחן במודול, או בכתובת: https://cktest.cs.colman.ac.il/ ולהגיש ל PTM1 ומועד א' את הקבצים Q1.java, EmployeeBuilder.java, Interpreter.java.

בכל הגשה יש להגיש את כל הקבצים (ולהתייחס לפלט רק של השאלות שעניתם עליהן).

ניתן להגיש במוד אימון ובמוד הגשה כמה פעמים שתרצו עד לסוף המבחן.

בסוף המבחן יש להגיש במוד הגשה ואז במוד הגשה סופית.

אחריה תקבלו מס' אסמכתא בן 4 ספרות. לאחר הגשה במוד זה לא תוכלו להגיש יותר.

אם מתעוררות שאלות

תוכלו לצפות בסרטון המצורף לקובצי המבחן בו אני מסביר את טופס הבחינה. אם בכל זאת מתעוררת שאלה תוכלו לשלוח מייל לכתובת: oop.course.colman@gmail.com

ניתן מענה לשאלות הבהרה בלבד – מה לא מובן בטופס הבחינה.

בפרט, **לא** נענה על שאלות בסגנון של "יש לי באג", "עובד לי במוד אימון אך לא בהגשה" וכדומה. עליכם להתמודד עם מימוש הקוד בכוחות עצמכם.

יושרה אקדמית

אנו מצפים מכם לשמור על יושרה אקדמית ולא להעתיק, להיעזר או לעזור לנבחנים אחרים.

מתבצעת בדיקת העתקות אוטומטית לאחר סיום המבחן, ובמקרה של חשד גם העוזר וגם הנעזר יעלו לוועדת משמעת.

בפרט, אין "להשוות תשובות" ביום המבחן – גם לאחר שסיימתם. מהרגע ששלחתם תשובות למישהו אין לכם שליטה לאן זה יגיע, יש סטודנטים שעדין נבחנים ולא תרצו למצוא את עצמכם כחשודים בהעתקה.

בסופו של יום זהו אינטרס שלכם שכל סטודנט יקבל את הציון המאפיין אותו לאחר שהתאמץ עבורו.

בהצלחה!