



Advanced Object Oriented Programming & Java

Assignment Replacing Course Exam

Dr. Moshe Deutsch

mdeuts@ruppin.ac.il

Submission Instructions:

1. במטלה שאלה אחת. לכל Public Class בשאלה יש לכתוב קובץ java נפרד C.java כאשר C הוא שם ה class שממומש בקובץ זה.
2. הקוד עבור כל ה Classes חייב להיות Well-Commented!!!
3. בנוסף, יש להעתיק את כל התוכניות במלואן לקובץ WORD אחד. בקובץ זה, עבור כל class שמועתק לשם, יש לציין את שם ה class הממומש. בנוסף, בסיום, יש לצרף כמה דוגמאות מייצגות לפלט ממסך ההרצה. בתחילת הקובץ הנ"ל, יש לציין את שמות המגישים ות.ז. שלהם (במודגש!).
4. יש לכתוב הערות לקוד, התומכות ב javadoc עם קישורים ל Java API Documentation (ראו פסקה אחרונה בתיאור המטלה בהמשך). יש להריץ Javadoc על כל הקבצים ולנתב את הפלט שלו לתת ספריה בשם Doc. אתם נדרשים להריץ את Javadoc בצורה כזו שיווצרו קישורים ל Java API Documentation מהתיעוד שלכם ושייתועדו גם כל ה private Classes ו private methods בפרויקט שלכם (ראו פסקה אחרונה בתיאור המטלה בהמשך).
5. יש לשים את כל ה sources (רק קבצי java) בתת ספריה Src ואת קובץ ה WORD מעל שתי הספריות (Doc ו Src) במבנה הספריות ולכוון בקובץ ZIP או RAR אחד את כל אלה ולהעלותו לאיזור המטלה באתר הקורס.
6. שם קובץ ה-zip יהיה כך: JavaFinProj_Name1_Name2. כאשר: Name1 מציין את השם המלא (שם פרטי + שם משפחה) של סטודנט 1 ו Name2 מציין את השם המלא (שם פרטי + שם משפחה) סטודנט 2. שם הקובץ יהיה באנגלית בלבד!
7. כאשר פותחים את הקובץ המכוון, יש לקבל 3 דברים: JavaFinProj.docx – קובץ ה WORD (סעיף 3), ספריה Src (המכילה את כל ה Java source files – סעיף 5) וספריה Doc (סעיף 4).
8. תאריך אחרון להגשת המשימה: מוצ"ש, תאריך 13/02/2021 – עד השעה 23:59 בלילה. לא ניתן לאחר בהגשה! מעבר לתאריך זה (בשעה הנקובה!), המערכת תיסגר באופן אוטומטי להגשות ומי שלא יגיש עד לתאריך זה ייקבל באופן אוטומטי ציון סופי 0 במשימה.
9. יש להגיש את המטלה בזוגות בהם עבדתם במהלך הקורס. בתחילת קובץ ה Word – יש לרשום במודגש את הפרטים המלאים של המגישים (שמות פרטיים, שמות משפחה ות.ז.). סטודנט אחד מהזוג המגיש יעלה את המשימה לאתר ה Moodle.

10. **אזהרה:** מטלות שתוגשנה לא בהתאם להוראות ההגשה לעיל, תספוגנה הורדה ניכרת בציון הסופי!

*** אזהרה:**

המרכז אקדמי רופין רכש לאחרונה מערכת ממוחשבת לזיהוי העתקות. כל הגשותיכם ייבדקו ע"י המערכת הזו. **כל הגשה, לגביה תהיה התראת העתקה – ולו הקטנה ביותר – תיפסל באופן אוטומטי וכל הסטודנטים הנוגעים בדבר יועלו באופן מיידי לוועדת משמעת!**

Exercise – 100%

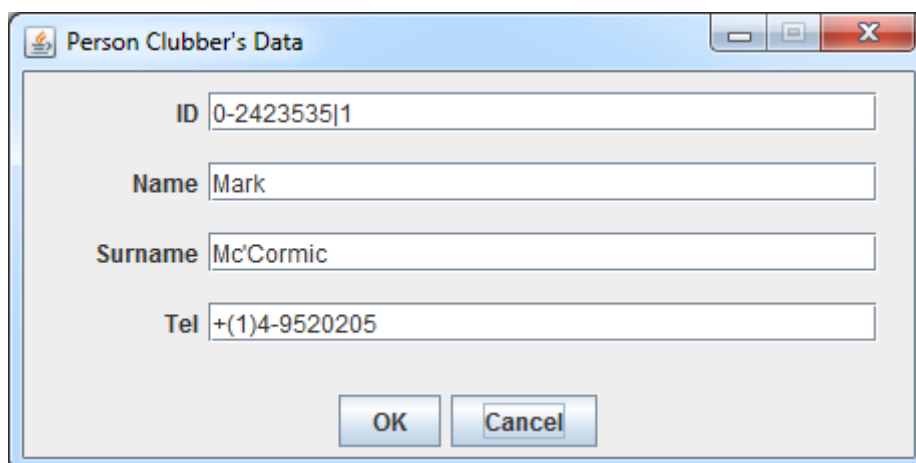
מימוש מערכת מידע לניהול לקוחות עבור מועדון לילה "B.K" בנהריה.

דרישות ותיאור המערכת:

הבעלים של מועדון לילה ה "B.K" בנהריה החליט לרכוש מערכת מידע לניהול הלקוחות הקבועים של המועדון, ע"מ לעדכן אותם בדבר אירועים מיוחדים במועדון וכן להקנות להם הנחה מיוחדת. לשם כך, נפגש מנהל המועדון עם ארכיטקט מערכות תוכנה והוחלט במשותף על הדרישות הבאות:

- כיוון שהמועדון יחסית קטן, מדובר במספר לא גדול של לקוחות קבועים – כפי שנראה בהמשך: יש לכך השלכות על ה Design הקונספטואלי של ניהול המידע, עבור כל לקוח, בשילוב עם ה GUI במערכת (גם כן, עבור כל לקוח!);
- הישויות שעליהן יישמר מידע במערכת סווגו ל 3 הסוגים הבאים: Person (אדם), Student (סטודנט) שהוא סוג של Person ו Soldier (חייל) שהוא גם כן סוג של Person;
- כל המידע הפנימי שנשמר בישויות הללו הנו מידע טקסטואלי!
- המערכת תשמור את כל המידע על הלקוחות השונים בקובץ, כאשר עם עליית המערכת: המידע מהקובץ יועתק לאובייקטים השונים (מ 3 הסוגים שהוזכרו לעיל) ויישמר במאגר פולימורפי – זהו למעשה ArrayList של references לאובייקטים מסוג ClubAbstractEntity, שהוא למעשה abstract class שעומד בראש היררכיית התורשה של כל 3 סוגי הלקוחות במועדון;
- לפני היציאה מן המערכת (exit מהאפליקציה), המערכת תכתוב בחזרה לקובץ את המידע הפנימי שנשמר בכל אובייקט שחי במאגר הפולימורפי הנ"ל. לכן, ישנה חשיבות עליונה לכך **שהמידע הפנימי, השמור בכל אובייקט במאגר, יהיה ווליד (valid) בכל רגע נתון!**

- המערכת תספק תפריט משתמש פשוט של הכנסת מזהה ייחודי של לקוח לחיפוש במאגר הפולימורפי הנ"ל. במידה ונמצא לקוח מתאים, המערכת תציג את פרטי הלקוח **בחלון שמוקדש ללקוח זה**, כאשר לכל לקוח – החלון שמוקדש לו נבנה על פי סוג הלקוח. לדוגמא, מצורפים 3 צילומי מסך: צילום לדוגמא של חלון פרטי לקוח מכל אחד מ 3 סוגי הלקוחות במועדון:



Person Clubber's Data

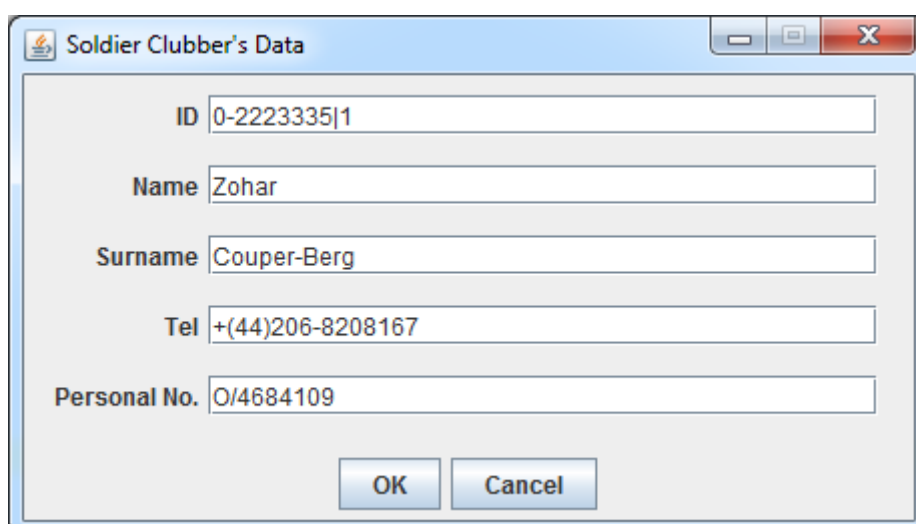
ID 0-2423535|1

Name Mark

Surname Mc'Cormic

Tel +(1)4-9520205

OK Cancel



Soldier Clubber's Data

ID 0-2223335|1

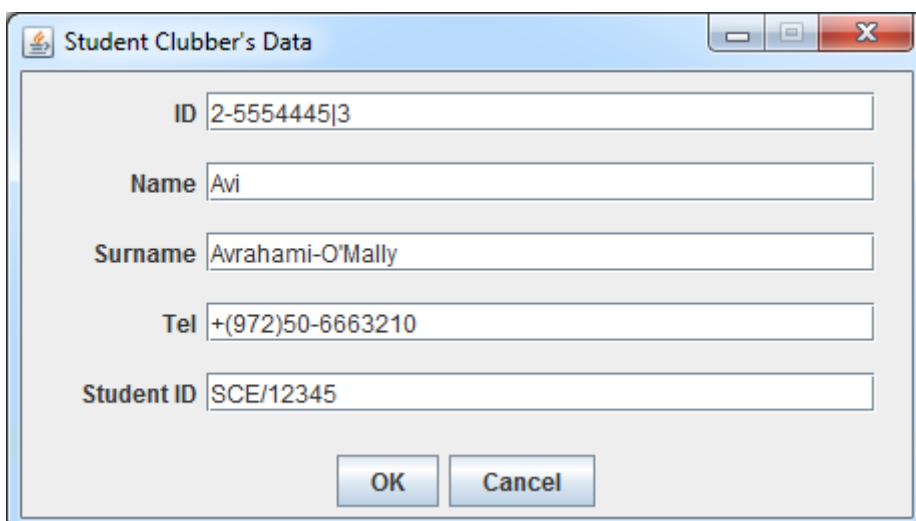
Name Zohar

Surname Couper-Berg

Tel +(44)206-8208167

Personal No. O/4684109

OK Cancel



Student Clubber's Data

ID 2-5554445|3

Name Avi

Surname Avrahami-O'Mally

Tel +(972)50-6663210

Student ID SCE/12345

OK Cancel

• קונספט העבודה:

- כל אובייקט מכל אחד מ 3 סוגי הלקוחות במועדון הוא בעצמו (is a) חלון (JFrame), כאשר לכל אובייקט כזה יש ב "בטן" את המידע (בייצוג String) עבור כל שדה, והמידע הזה מועתק ל JTextFields המתאימים בחלון;
- החלון תמיד יהיה קיים, לכן יש לשחק אך רק עם הנראות שלו (visibility) ולא באמת לאפשר סגירה שלו. לכן:
 - אסור שתהיה אפשרות להגדיל/להקטין את החלון;
 - אסור שתהיה אפשרות לסגור את החלון עם כפתור ה X (האדום - בפינה הימנית העליונה של החלון);
- כאשר החלון במצב מוחבא, תמיד תהיה התאמה מלאה בין המידע ששמור ב "בטן" של האובייקט למידע שמוצג ב JTextFields המתאימים בחלון. לכן:
- לחיצה על כפתור ה OK תבצע וולידציה של כל התכנים של ה JTextFields - לפי הסדר מלמעלה למטה (לפי חוקיות שתיתן בהמשך). אם כולם וולידים, המידע מכל ה JTextFields המתאימים יועתק לשדות המתאימים ב "בטן" של האובייקט והחלון יוחבא. לחילופין, אם הוולידציה נכשלה – תהליך הוולידציה ייפסק ב JTextField הראשון שהוולידציה שלו נכשלה ותסומן כוכבית בצבע אדום לימינו של JTextField זה; כמו כן, במקרה כזה, החלון יישאר נראה ולא יוחבא! (יש לדאוג, כמובן, שבכל שדה שעבר וולידציה לנקות את הכוכבית האדומה לימינו!); דוגמאות:

- לחיצה על כפתור ה Cancel למעשה עושה מה שנקרה rollback, כלומר, כל המידע ששמור ב "בטן" של האובייקט מועתק ל JTextFields המתאימים (לא משנה מה היה ב JTextFields קודם לכן); בנוסף, כל הכוכביות האדומות ינוקו והחלון יוחבא;
- בגדול, תואר פה למעלה התהליך הקונספטואלי – בהמשך נפרק את התהליך לפרטים ונגדיר מי אחראי על מה, ואיך כל התהליך הנ"ל נעשה בצורה פולימורפית!

Design פרטני (ראו גם UML Class Diagram מפורט בעמוד 8):

ClubAbAbstractEntity Class

קונספט כללי:

1. abstract class שהוא יורש ישיר של JFrame.
2. אחראי להכללה של ה GUI של החלון (עבור כל היורשים שלו!), לכן הוא זה שמכיל את שני הכפתורים (OK ו Cancel) והוא גם ממקם אותם באיזור התחתון של החלון. בנוסף, אחראי לספק ממשק (protected method **addToCenter**) שמאפשר רק ליורשים שלו להוסיף GUI Components לחלק המרכזי בחלון.
3. מגדיר 4 abstract methods וכך מחייב את היורשים שלו לממש אותן, על מנת שכל סוג לקוח (יורש) יגדיר לעצמו את הפעולות השונות על המידע וה GUI הספציפי אצלו.
4. השימוש ב 4 abstract methods הללו מאפשר handling (טיפול) גנרי של הכפתורים "OK" ו "Cancel" מתוך ה class **ClubAbAbstractEntity** עבור כל סוגי הלקוחות (יורשים), רק שכל יורש צריך להגדיר כל פעולה כזו, עפ"י מה שמתאים לו וכך (ביתר קלות!) הפעולות המתבצעות מתוך ה handler הריכוזי של "OK" ו "Cancel", במחלקה הזו, יהיו פולימורפיות.

:Instance Variables

ראו ב UML Class Diagram (עמוד 8)

Methods:

Parameterless Constructor – יצירה ואתחול כל ה instance variables הנדרשים ל class זה; יצירה ואתחול של כל ה GUI elements שנדרשים ל class זה; יצירה ואתחול ה handler (מסוג inner class ButtonsHandler) ורישום כ handler עבור שני הכפתורים (OK ו Cancel); מניעת resizing של החלון; מניעת ההשפעה של לחיצה על לחצן ה X (פינה ימנית עליונה של החלון).

protected void addToCenter(Component guiComponent) – מתודה המקבלת Component של GUI כלשהו ומוסיפה אותו לחלק המרכזי של החלון, עפ"י ה Layout שמוצג ב 3 צילומי המסך של החלונות לעיל. המתודה הזו מהווה ממשק שמאפשר ליורשים של **ClubAbAbstractEntity** להוסיף GUI Components לחלק המרכזי בחלון (כל יורש למעשה ייקרא למתודה הזו כאשר הוא מעוניין להוסיף לחלון את כל מה שהוא צריך – זה קורה ב constructor של היורש!).

public abstract boolean match(String key); – מתודה אבסטרקטית המגדירה ממשק ל "התאמה". כל יורש מחוייב לממש מתודה זו שמקבלת מפתח לחיפוש (מסוג String) ומחזיר true/false האם המפתח הזה מתאים למזהה הייחודי של הלקוח או לא.

protected abstract boolean validateData(); - מתודה אבסטרקטית המגדירה ממשק לוולידציה. כל יורש מחויב לממש מתודה זו שמחזירה true/false האם הוולידציה של התכנים של כל ה JTextFields שלו (לפי קריטריונים שיפורטו בהמשך) עברה/נכשלה.

protected abstract void commit(); - מתודה אבסטרקטית המגדירה ממשק לשמירת המידע ב "בטן". כל יורש מחויב לממש מתודה זו שמבצעת שמירת המידע מה JTextFields השונים (ששייכים לו!) לשדות הנשמרים ב "בטן" שלו.

protected abstract void rollBack(); - מתודה אבסטרקטית המגדירה ממשק ל rollback. כל יורש מחויב לממש מתודה זו שמבצעת rollback אצלו – כלומר, כל המידע ששמור ב "בטן" שלו מועתק ל JTextFields המתאימים שלו (ולא משנה מה היה ב JTextFields הללו קודם לכן!).

ה inner class שנקרא ButtonsHandler:

- (1) מהווה handler ריכוזי לשני הכפתורים (OK ו Cancel).
- (2) במתודת ה handling הנדרשת, בסה"כ יש לממש את ההתנהגות הבאה (באמצעות קריאה למתודות האבסטרקטיות המתאימות – שתוארו למעלה):
 - a. אם כפתור ה "OK" נלחץ – יש להפעיל את **validateData**: אם זו החזירה תשובה חיובית, יש לקרוא ל **commit** ולהחביא את החלון. אם זו החזירה תשובה שלילית, פשוט לא לעשות כלום!
 - b. אם כפתור ה "Cancel" נלחץ – פשוט יש לקרוא ל **rollback** ולהחביא את החלון.
- (3) ניתן לראות שהמימוש של ה handler הזה הוא גנרי לחלוטין ונכון לכל סוג לקוח – מה שמשאיר בידי כל יורש לממש את כל 4 המתודות האבסטרקטיות שתוארו לעיל (בהתאם לשדות שקיימים אצלו), כאשר כל אחת מהמתודות הללו ביורש תדאג גם להצגה/ניקוי של הכוכבית האדומה (בהתאם לדרישות שתוארו בסעיף "קונספט העבודה" – עמוד 5).

Classes: Person, Student, Soldier

מה שמתבקש בכל אחד מ 3 ה Classes הללו זה בעצם לספק constructor מתאים עם פרמטרים ולספק מימוש מתאים לכל אחת מ 4 המתודות האבסטרקטיות שהוגדרו ב ClubAbAstractEntity. שימו לב ש Person הוא יורש ישיר של ClubAbAstractEntity. מה שאומר שהוא מחויב לממש את כל 4 המתודות האבסטרקטיות. Student ו Soldier, לעומת זאת, יורשים מ Person שכבר מימש את המתודות הללו (וכאמור Person אינו אבסטרקטי!) זה מחייב אתכם לשני דברים ב Student ו Soldier

- (1) לוודא שמתבצע overriding של 4 המתודות הללו ללא טעויות "הקלדה"!
- (2) ה Overriding שלהן ב Student ו Soldier חייב להיות collaborative! (עפ"י הדרישות בהמשך).

Person Class

:Instance Variables

כל מה שנדרש ע"מ לתמוך בשדות id (ת.ז.), name (שם פרטי), surname (שם משפחה) ו tel (מספר טלפון) עפ"י הדרישות שתוארו לעיל – ראו ב UML Class Diagram (עמוד 8).

Methods:

Constructor with 4 parameters – מקבל 4 פרמטרים מסוג String עבור 4 שדות המידע שלו; יצירה ואתחול כל ה instance variables הנדרשים ל class זה; יצירה ואתחול של כל ה GUI elements שנדרשים ל class זה; הגדרת ה Title של החלון (ראו צילום חלון מתאים למעלה), הגדרת הגודל של החלון עבור Person (450 x 220).

public boolean match(String key); - מחזירה true אם הפרמטר key שווה בתוכנו לשדה id. אחרת, מחזירה false.

protected boolean validateData(); - מתודה זו מתנהגת עפ"י הדרישות שתוארו קודם לכן. הוולידציה על תוכן ה JTextFields שלו מתבצעת באמצעות בדיקת התאמה של כל אחד מהם ל RE (Regular Expression) המתאים. חוקיות Regular Expressions לשדות של Person:

id field RE: digit, followed by character -, followed by 7 digits, followed by character |, followed by a non-zero digit (e.g. 0-2423535|1);

name field RE: capital letter, followed by at least one lower-case letter (e.g. Zohar);

surname field RE: capital letter, followed by 0 or more lowercase letters, followed by 0 or 1 occurrences of: character ' or character - . **This entire**

pattern must be matched at least once (e.g.: Mc'Cormic, Deutsch, O'Brian, Landau-Mc'Donalds, Dor-, M', Mc').

Tel field RE: international phone/mobile-number, e.g.: +(1)4-9520205, +(44)206-8208167, +(972)50-6663210. **First digit after left-bracket must be non-zero (this digit is compulsory!)**; number of digits inside the brackets should be **at most 3**; **first digit after the right-bracket must be non-zero**; number of digits between the right-bracket and the dash character - should be **at most 3**; **first digit after the dash character – must be non-zero**; number of digits after the dash character - **should be 7**.

protected void rollBack(); ו **protected void commit();** - מתודות אלו מתנהגות עפ"י הדרישות שתוארו קודם לכן.

Student Class

:Instance Variables

כל מה שנדרש ע"מ לתמוך בשדה studentID (מספר סטודנט) עפ"י הדרישות שתוארו לעיל – ראו ב UML Class Diagram (עמוד 8).

Methods:

Constructor with 5 parameters – מקבל 5 פרמטרים מסוג String: 4 פרמטרים עבור שדות המידע של Person ופרמטר אחד לשדה המידע הנוסף שלו; יצירה ואתחול כל ה instance variables הנדרשים ל class זה; יצירה ואתחול של כל ה GUI elements שנדרשים ל class זה; הגדרת ה Title של החלון (ראו צילום חלון מתאים למעלה), הגדרת הגודל של החלון עבור Student (450 x 250).

public boolean match(String key); - מחזירה true אם המתודה של Person החזירה true עבור key זה, **או** שהפרמטר key שווה בתוכנו לשדה studentID, **אבל רק החל מהתו** **החמישי ב studentID** (מכיוון שה pattern של studentID מחייב ש 3 התווים הראשונים יהיו אותיות גדולות שמהוות קיצור שם המוסד ואחריהן התו / slash – למשל: RUP/12345, TAU/61142 וכו' – ראו בהמשך).

protected boolean validateData(); - מתודה זו מתנהגת עפ"י הדרישות שתוארו קודם לכן. הוולידציה על תוכן ה JTextField שמציג את מספר הסטודנט, מתבצעת באמצעות בדיקת התאמה שלו ל Regular Expression המתאים. חוקיות Regular Expression לשדה studentID של Student:

studentID field RE: 3 capital letters (representing the institution name shortcut), followed by slash /, followed by 5 digits (first digit must be non-zero). E.g. RUP/12345, HIT/56098, SCE/14606.

protected void rollBack(); ו **protected void commit();** - מתודות אלו מתנהגות
עפ"י הדרישות שתוארו קודם לכן.

Soldier Class

:Instance Variables

כל מה שנדרש ע"מ לתמוך בשדה personalNum (מספר אישי) עפ"י הדרישות שתוארו לעיל –
ראו ב UML Class Diagram (עמוד 8).

Methods:

Constructor with 5 parameters – מקבל 5 פרמטרים מסוג String: 4 פרמטרים עבור
שדות המידע של Person ופרמטר אחד לשדה המידע הנוסף שלו; יצירה ואתחול כל ה instance
variables הנדרשים ל class זה; יצירה ואתחול של כל ה GUI elements שנדרשים ל class זה;
הגדרת ה Title של החלון (ראו צילום חלון מתאים למעלה), הגדרת הגודל של החלון עבור Soldier
(450 x 250).

public boolean match(String key); - מחזירה true אם המתודה של Person החזירה
true עבור key זה, או שהפרמטר key שווה בתוכנו לשדה personalNum.

protected boolean validateData(); - מתודה זו מתנהגת עפ"י הדרישות שתוארו קודם
לכן. הוולידציה על תוכן ה JTextField שמציג את המספר האישי מתבצעת באמצעות בדיקת התאמה
שלו ל Regular Expression המתאים. חוקיות Regular Expression לשדה personalNum
של Soldier:

personalNum field RE: one capital letter of either of the 3 letters: R, O, or C
(shortcut for: Rank, Officer or Chief - נגד, חוגר, קצין או נגד), followed by the
character /, followed by 7 digits (first digit must be non-zero). E.g.
O/4684509, C/5044789

protected void rollBack(); ו **protected void commit();** - מתודות אלו מתנהגות
עפ"י הדרישות שתוארו קודם לכן.

להלן תוכנית ראשית לבדיקת המערכת:

- כרגע, לשם הפשטות ורק לצורכי הבדיקה שלכם את התשתית שתדרשו לכתוב, המתודה **loadClubbersDBFromFile**, שאמורה לקרוא את המידע, לצורך יצירת האובייקטים השונים, מקובץ בינארי – יוצרת באופן ידני 3 אובייקטים (אחד מכל 3 סוגי ה **Classes** המתוארים למעלה). כמו כן, שוב, לצורך פשטות הבדיקה, המתודה **writeClubbersDBToFile**, שאמורה לכתוב את המידע, מהאובייקטים השונים, לקובץ הבינארי, לצורך שמירת המידע לפני שהתוכנית מסתיימת – אינה ממומשת.

// File: NightClubMgmtApp.java

```
import java.util.*;
```

```
public class NightClubMgmtApp
```

```
{
```

```
    //Night-Club Regular Customers Repository
```

```
    private ArrayList<ClubAbstractEntity> clubbers;
```

```
    private Scanner sc;
```

```
    public NightClubMgmtApp()
```

```
    {
```

```
        clubbers = new ArrayList<>();
```

```
        sc = new Scanner(System.in);
```

```
        loadClubbersDBFromFile();
```

```
        manipulateDB();
```

```
    }
```

```
    private void manipulateDB()
```

```
    {
```

```
        String input; boolean found = false;
```

```
        while(true)
```

```
        {
```

```
            System.out.print("Please Enter The Clubber's Key ");
```

```
            System.out.print("or \"exit\" to exit: ");
```

```
            input = sc.nextLine();
```

```
            if(input.trim().equalsIgnoreCase("exit"))
```

```
            {writeClubbersDBToFile(); System.exit(0);}
```

```
            for(ClubAbstractEntity clubber : clubbers)
```

```
                if(clubber.match(input))
```

```
                {
```

```
                    found = true;
```

```
                    clubber.setVisible(true);
```

```

        break;
    }

    if(!found)
        System.out.printf("Clubber with key %s does not exist%n", input);
    else found = !found;
}
} // End of method manipulateDB

private void loadClubbersDBFromFile()
{
    //Read data from file, create the corresponding objects and put them
    //into clubbers ArrayList. For example:
    clubbers.add(new Person("0-2423535|1", "Mark", "Mc'Cormic",
        "+(1)4-9520205"));

    clubbers.add(new Soldier("0-2223335|1", "Zohar", "Couper-Berg",
        "+(44)206-8208167", "O/4684109"));

    clubbers.add(new Student("2-5554445|3", "Avi", "Avrahami-O'Mally",
        "+(972)50-6663210", "SCE/12345"));
}

private void writeClubbersDBtoFile()
{
    //Write all the objects' data in clubbers ArrayList into the file
}

public static void main(String[] args)
{
    NightClubMgmtApp application = new NightClubMgmtApp();
}
} //End of class NightClubMgmtApp

```

משימתכם:

יש לממש במלואם את כל 4 ה classes: Student, Person, ClubAbstractEntity, Soldier, עפ"י הדרישות לעיל.

מתודה שיכולה לעזור לכם בתהליך היא:

public void setTitle(String title)

זוהיא מתודה ששייכת למחלקה Frame (שממנה יורש JFrame) שמאפשרת, כאשר קוראים לה, לשנות את הכותרת של החלון לטקסט שמתקבל בפרמטר המחרוזת (ראו פרטים נוספים ב Java API Documentation).

* לפני שאתם ממשיכים למימוש הסעיפים הבאים, יש לבדוק הייטב את המימוש שלכם, בעזרת התוכנית הראשית שניתנה לכם למעלה.

(b) יש להפוך את מימוש ה Class הראשי, **NightClubMgmtApp**, לחלון, עם ממשק GUI לתקשורת עם המשתמש. עליכם לתכנן ולבחור את דרך עיצוב החלון לשיקולכם – וזאת עפ"י הדגשים והשיקולים שנלמדו במהלך הקורס. עם זאת, עליכם להקפיד על ההנחיות הבאות במימוש זה:

i. יש לממש כפתור "Search" – לחיפוש אובייקט, עפ"י מפתח חיפוש שנקלט מהמשתמש. קליטת המחרוזת לחיפוש תיעשה באמצעות `InputDialog`. הפונקציונאליות של תהליך החיפוש, כתוצאה מלחיצה על כפתור ה "Search", צריכה להיות מבוססת על חלקי הקוד הרלוונטיים במתודה `manipulateDB`. אם לא נמצא אובייקט שמתאים למפתח החיפוש המבוקש, יש להוציא את ההודעה (בפורמט הממומש במתודה `manipulateDB`) ל `MessageDialog` עם אייקון של `Information Message`;

ii. יש לממש אפשרות למשתמש ליצור אובייקט חדש, מכל אחד מ 3 סוגי הלקוחות המוגדרים לעיל (`Person`, `Student` ו `Soldier`). לשם כך, באפשרותכם להשתמש באלמנט/אלמנטי GUI לבחירתכם, שיתאים/יתאימו לדגשים והשיקולים שנלמדו במהלך הקורס. על אלמנט/אלמנטי GUI זה/אלו ליזום את היצירה של כל אחד מ 3 סוגי הלקוחות המוגדרים לעיל (`Person`, `Student` ו `Soldier`) מה Class הראשי **NightClubMgmtApp**. ביצירה עצמה, חובה להשתמש בתשתית הקיימת (שמימשתם בסעיף a) של 4 ה Classes, המתוארים ב UML Class Diagram (עמוד 8). כלומר, יצירת אובייקט ריק (מהסוג המבוקש) והצגת החלק של החלון שלו (שכזכור כבר מובנה בתוכו), ע"מ לאפשר למשתמש להזין פרטים ב `JTextFields` שלו ולשמור את השינויים ע"י לחיצה על כפתור ה "OK" שלו (שהפונקציונאליות שלו כבר ממומשת בכל הרמות!). כמובן, תצטרכו להוסיף שינויים נדרשים בתשתית זו, בכדי לתמוך בתהליך. בנוסף, במקרה כזה כשאובייקט נוצר ריק לשם עדכון הנתונים שלו ע"י המשתמש, כאשר החלק של החלון שלו יוצג (כלומר, יהפוך ממוחבא לנראה) – על כפתור ה "Cancel" להיות במצב `disable`. כמובן, לאחר ה `commit` הראשון, יש להחזיר את כפתור ה "Cancel" בחזרה למצב `enable`.

iii. לעת עתה, השאירו את המתודה `loadClubbersDBFromFile` עם המימוש הנוכחי שלה, אך עליכם לחשוב הייטב (וכמובן לממש!) מהיכן בדיוק צריך לקרוא לה מתוך ה

Class הראשי, **NightClubMgmtApp**, בקונסטלציה החדשה שלו. ***טיפ:** בתיכנונכם, עליכם לחשוב כאילו המתודה `loadClubbersDBFromFile` באמת קוראת את המידע של הלקוחות הרשומים במועדון מתוך קובץ בינארי – ישירות לתוך ה- `ArrayList<ClubAbstractEntity>` שנקרא `clubbers`, כחלק מתהליך עליית המערכת.

iv. לעת עתה, השאירו את המתודה `writeClubbersDBToFile` עם המימוש הנוכחי שלה (קרי, מתודה ריקה), אך עליכם לחשוב היטב (וכמובן לממש!) מהיכן בדיוק צריך לקרוא לה מתוך ה- Class הראשי, **NightClubMgmtApp**, בקונסטלציה החדשה שלו. ***טיפ:** בתיכנונכם, עליכם לחשוב כאילו המתודה `writeClubbersDBToFile` באמת כותבת את המידע של הלקוחות הקיימים במערכת לתוך אותו קובץ בינארי שקראו ממנו את המידע (במתודה `loadClubbersDBFromFile`) – ישירות מתוך ה- `ArrayList<ClubAbstractEntity>` שנקרא `clubbers` – רגע לפני (או תו"כ...) שהחלון הראשי של המערכת (כלומר, החלון של ה- Class הראשי **NightClubMgmtApp**) נסגר.

(c) עליכם לספק מימוש משמעותי למתודות `loadClubbersDBFromFile` ו- `writeClubbersDBToFile`, לקריאה וכתובה (בהתאמה!) של המידע על הלקוחות לאותו קובץ בינארי. הנחיות למימוש:

- i. קראו בקפידה את סעיפים b.iii ו- b.iv (ישירות מעל סעיף זה) – תמצאו שם רמזים מאוד גדולים לאיך לגשת למימושים אלו, מבחינה קונספטואלית.
- ii. זהו נושא שתצטרכו לחקור לגביו באופן עצמאי. קישור לדף אינפורמציה מעולה שייתן לכם הכוונה מפורטת לנושא הוא:

<https://stackoverflow.com/questions/6529872/how-to-write-class-object-to-bin-file>

לאחר שתקראו את הדף הנ"ל, תוכלו להרחיב את הקריאה על ה- Classes הבאים ב- `Java API Documentation`:

1. `FileInputStream`
2. `ObjectInputStream`
3. `FileOutputStream`
4. `ObjectOutputStream`

iii. לצורך הפשטות, שם הקובץ הבינארי יהיה "BKCustomers.dat" ואין צורך לעסוק ב-paths – כלומר, צאו מתוך הנחה שהקובץ נקרא/נכתב מתוך/לתוך הספרייה ממנה אתם מריצים את התוכנית.

(d) עליכם לספק תיעוד מלא של כל הקוד ב Javadoc, עפ"י ההנחיות הבאות:

Enhanced Javadoc Comments:

Similarly to assignment 4, you will require to comment your code, in a similar "spirit" to the comments provided in the "Calculator" Case Study in Ch. 8. Namely, you will need to also use the {@link Classname}, {@link Classname #methodName} and {@link Classname#Methodname}, in order to link your comments in the program to the Java API Documentation (*at least 5 link examples from each of the 3 link-types above!*).

When you run the Javadoc, you should do it in a similar way to what is described in Ch. 8, slide 256. That is, generating the documentation for the *private classes/methods (and above!), as well as linking the documentation to the online Java API Documentation.*

דגשים נוספים למימוש:

- לגבי דרישות ה GUI, שהוצגו בעמודים 4 ו 5: שימו לב למיקומם המדויק של אלמנטי ה GUI בחלונות השונים – בפתרונכם, יש לממש תמונות אלו במדויק, כפי שמופיעים בצילומי המסך בעמודים הללו.
- כל ה MessegDialogs ו ה InputDialogs שהתכנית תפתח למשתמש, חייבים להפתח במרכז החלון, שאליו הם משוייכים.
- על המימוש להיות מודולרי, גנרי, אלגנטי, ללא חזרות/שכפולים של קטעי קוד! ומקיים את כל הדגשים והשיקולים שנלמדו במהלך הקורס.
- כל RE (Regular Expression) שתגדירו, עבור הבדיקות המוגדרות בעמודים 9 – 11, חייב להיות מוגדר בצורה מודולרית ואלגנטית. קרי, ללא חזרות/שכפולים מיותרים של תתי-חלקים בתוכו.

- יש להגיש את התוצר המלא הסופי, לאחר ביצוע כל הסעיפים הנדרשים (החלוקה לסעיפים נועדה עבורכם – לצורך הדרגתיות התוספות והשינויים, ע"מ שתוכלו לבדוק הייטב כל תוספת/שינוי, לפני שאתם ממשיכים הלאה).

Good Luck!