**מעבדה בעיצוב תבניות בתוכנה**

|  |  |
| --- | --- |
| **שם** | **ת"ז** |
| ולדיסלב ברקנס | 317225993 |
| אופק שחר | 204396782 |
| יונתן גוב | 316569458 |

**הוראות הרצה של התוכנה:**

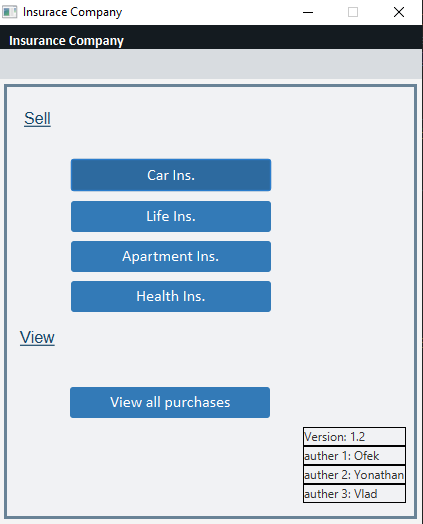
נכנסים לתיקייה release לוחצים פעמיים על קובץ InsuranceCompany.jar.

כדי להריץ מהקוד הפעל את ה main שנמצא ב קובץ App.java המוכל בחבילה gui.

**הוראות משתמש + מסך מהתוכנה:**

**מסך פתיחה:**

בעת פתיחת התוכנה החלון הביאה יעלה.

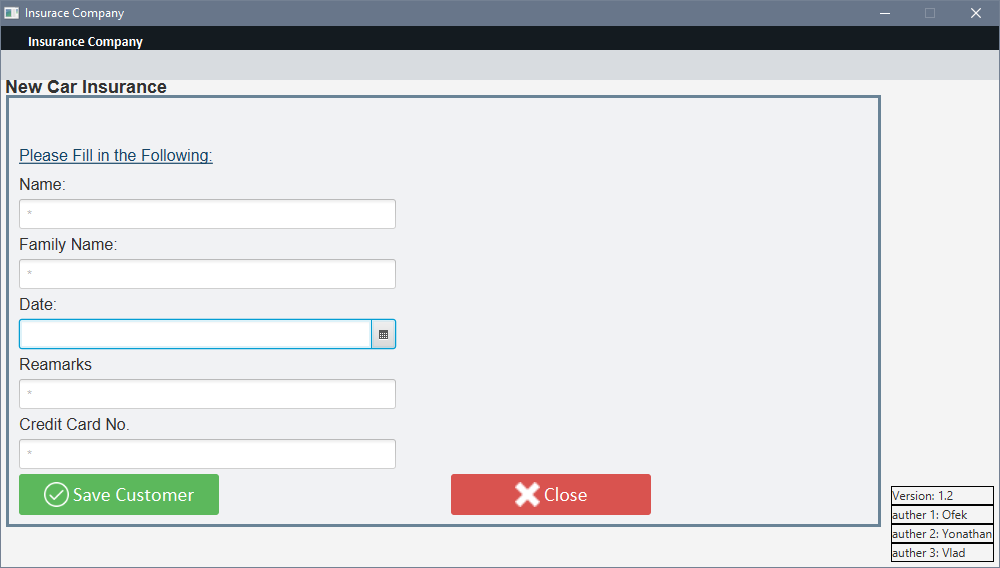
****

בחלון מופיע 4 לחצנים:

* לחצן Car Ins - בעת לחיצה תעבור לחלון טופס רישום עבור ביטוח רכב.
* לחצן Life Ins - בעת לחיצה תעבור לחלון טופס רישום עבור ביטוח חיים.
* לחצן Apartment Ins - בעת לחיצה תעבור לחלון טופס רישום עבור ביטוח הדירה.
* לחצן Health Ins - בעת לחיצה תעבור לחלון טופס רישום עבור ביטוח חיים.
* לחצן View all purchases -כדי לצפות בכל הביטוחים שנעשו עד כה לחץ על כפתור זה.

**יצירת ביטוח חדש:**

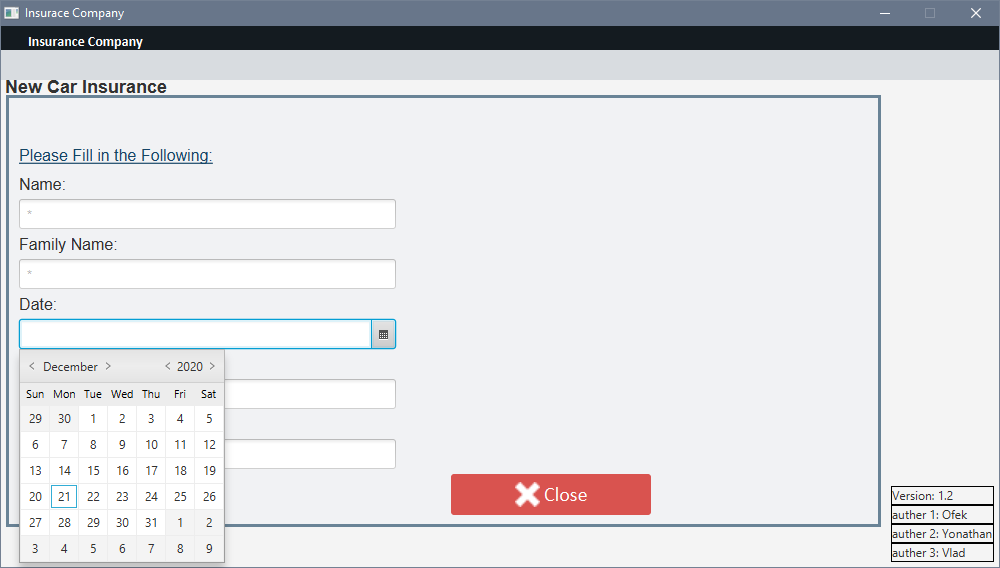
כאשר אנו נמצאים במסך הפתיחה, לחיצה על אחד מהכפתורים : Car Ins , Life Ins , Apartment Ins ,Health Ins יעלה את החלון הבא:

****

כדי להוסיף ביטוח למערכת חייב למלא את כל השדות.

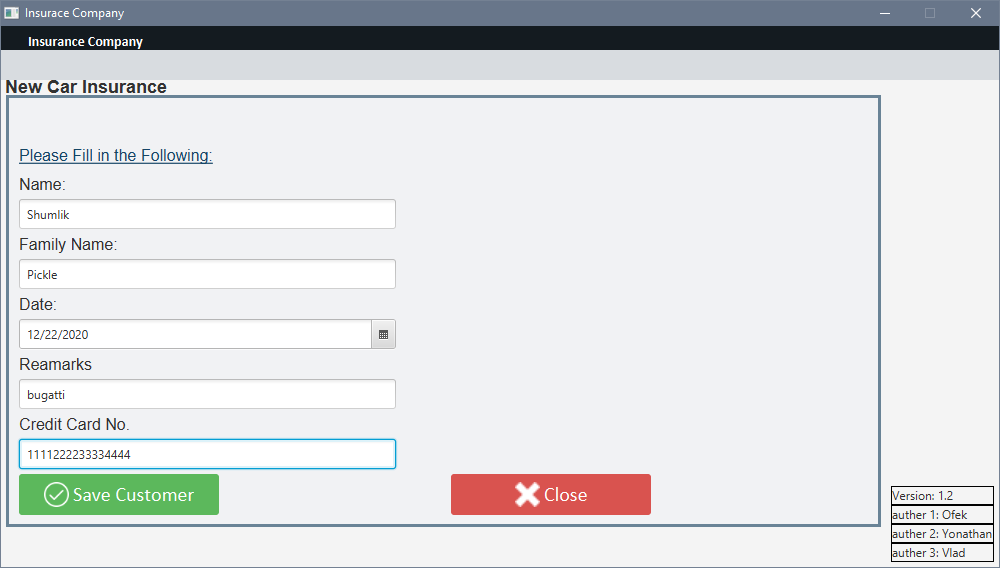
השדות:

* Name - שם פרטי של בעל הביטוח באנגלית
* Family Name - שם משפחה של בעל הביטוח באנגלית
* Remarks - הערות נוספות באנגלית.
* Credit Card No. - מספר כרטיס האשראי
* Date - הכנסת תאריך הביטוח, ניתן לכתוב בצורה : mm/dd/yyyy (כל אות מייצגת ספרה) או ללחוץ על המלבן  שיפתח את החלון הבא:

****

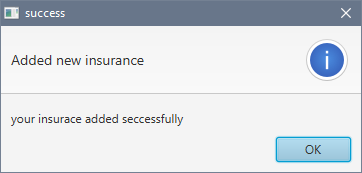
נפתח יומן שם ניתן לבחור את תאריך הביטוח, לחיצה על יום מסוים תרשום את התאריך הנבחר.

מילוי נכון של הטופס יראה כך:

****

לחיצה על הכפתור  תרגום לחזרה לתפריט הראשי ולכל המידע שנרשם בטופס להימחק.  
לחיצה על תבקש מהמערכת לשמור את הטופס.

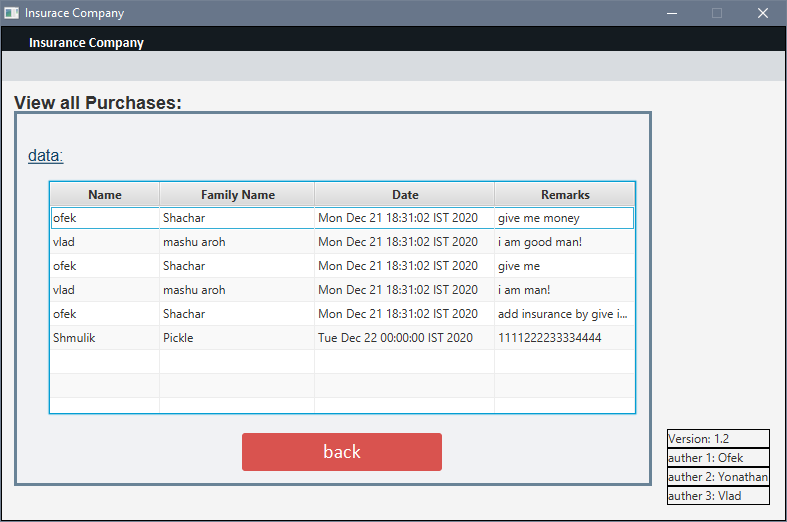
* במקרה שבו הטופס לא מולא כראוי לחיצה על הכפתור לא תעשה דבר.
* במקרה שבו הטופס מולא כראוי והמידע נשמר במערכת ינוקו שורת הטופס ויקפוץ החלון הבא.

****

לחיצה על OK תסגור את החלון הקופץ

**צפייה בביטוחים במערכת:**

כאשר אנו נמצאים במסך הפתיחה, לחיצה על : View all purchases יעלה החלון הבא:

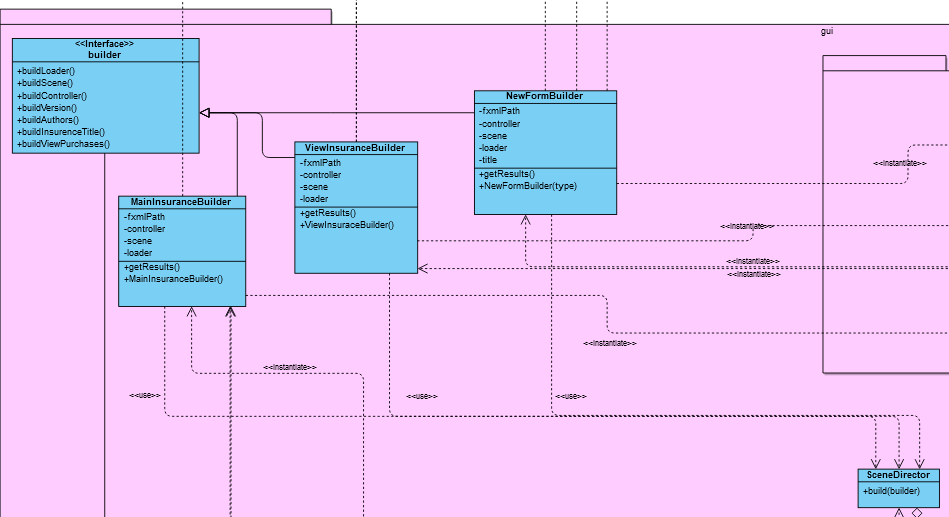
****

בחלון ניתן לראות את כל הביטוחים שנרכשו במערכת.

לחיצה על : תחזיר את המשתמש למסך הפתיחה.

**הסבר לתבניות התכן בפרויקט:**

**Builder**

****

תבנית מפרידה בין יצירה של מסכים בתוכנה ל מופע שנוצר ואיך הוא נוצר.

היתרון המשמעותי בזה שכל שיוצרים מסך אז controller שלו אינו מכיל לוגיקה אילה פונקציונליות כללית בלבד.

לדוגמה בכל controllers יש מתודה של לקבל את כל הכפתורים במסך נועד ש builder Hבנה את הכפתורים לפי הצורך המתאים.

יש לנו באפליקציה 3 מסכים שונים, מסך ראשי, מסך יצירת ביטוח ומסך צפייה בביטוחים שנוצרו.

לכן לכל סוג יש builder משלו MainInsuranceBuilder, ViewInsuranceBuilder ו NewFormBuilder .

כל builder מכיל את שלבי הבניה שוני את Interface .

כדי ליצור מסך ניצור את Scene בעזרת director ו builder המתאים לאחר מכן מגדירים ש Scene הנוצר היה Stage הראשי והחלון מתחלף.

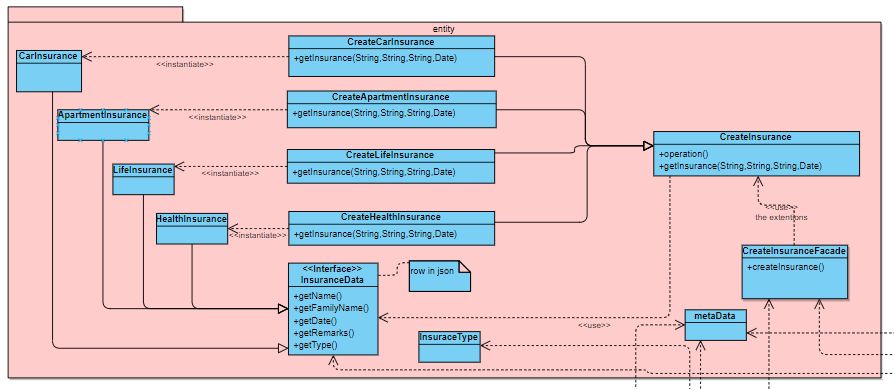
**Facade**

התבנית משמשת כממשק בעבור תת מערכת.

אצלנו ה Facade הוא המחלקה CreateInsuraceFacade . המחלקה מאפשרת למשתמש לא לדעת איך בנוי ה החבילה entity , כלומר הוא לא צריך להכיר את כל המפעלים אלה רק לשלוח איזה ביטוח הוא רוצה (ע"פ enum שנקרא InsuraceType) ויווצר לו הביטוח שהוא רוצה. אם נרצה להמשיך לפתח את המערכת נוכל בקלות להוסיף מפעלים חדשים ו enums מבלי שהמשתמש נאלץ לשנות את הקוד שלו, בנוסף נוכל להשתמש בו כדי להוסיף יצירה של entities נוספים.

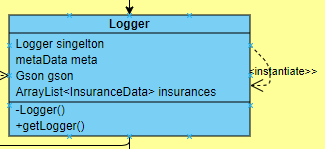
**Abstract Factory**

התבנית משמשת כדי להוסיף שכבת אבסטרקציה נוספת מעל תבנית ה factory שאנחנו מכירים. כלומר יכולים להיות מספר מפעלים המייצרים את אותו המוצר בצורה שונה. כל המפעלים ממשים מפעל על, שאותו המשתמש יכיר. אצלנו יש 4 מפעלים שממש את המפעל CreateInsurance . כל המפעלים מייצרים את המוצר InsuranceData. ה - IsuranceData הוא בעצם ייצוג של ביטוח וכל מפעל מייצר סוג אחר של ביטוח. כרגע ההבדל היחידי הוא הסוג אך בפיתוח עתידי נוכל להוסיף שחשוב המחיר לכל ביטוח נעשים בצורה שונה מבלי שהמשתמש יצטרך לשנות את הקוד שלו.

****

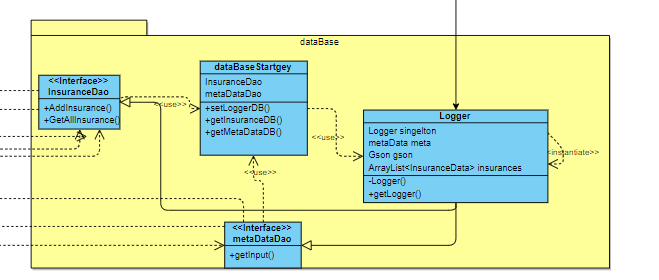
**Singleton**

זוהי מחלקה אשר מאפשרת אובייקט סטטי אחד בלבד מהמחלקה ולא נותנת ליצור עוד אובייקטים מאותו סוג.

****

אנחנו משתמשים במחלקת הLogger אשר מחזיקה נתונים ממאגרי הנתונים בכל רגע ואיננו רוצים שיהיו התנגשויות עם אובייקטים שונים כאלה. בעצם היא מחזיקה אובייקט סטטי מסוג Logger שקראנו לו singelton והבנאי שלה הוא פרטי לכן לא ניתן לקרוא לבנאי שלה הפקודה היחידה שתביא את המופע שלה זה getLogger אשר יחזיר את singleton הסטטי. כדי שלא יהיו בעיות עם חוטים אנחנו יוצרים את האובייקט הזה בזמן שנטענים הקבצים על ידי הצהרה שהאובייקט הוא final static .

**Data Access Object**

התבנית מיועדת בשביל לקשר בין מאגר נתונים כלשהו לתוכנית עצמה. אנו משתמשים בתבנית ומקשרים אותה בין ה Logger שלנו ל2 מאגרי נתונים: אחד מהם זה המידע על התוכנה עצמה (יוצרים גירסה) והשני זה מידע אשר משתנה בזמן אמת על כלל הביטוחים הקיימים. הLogger יודע להשתמש במאגר נתונים מסוג json ל2 מאגרי הנתונים שאנו משתמשים. 

ניתן לראות בדיאגרמה את הקישורים שדיברנו עליהם בנוסף יש dataBaseStartgey אשר נרחיב עליו בהמשך.

הLogger בעצם ממש את הממשק של ביטוחים(InsuranceDAO) ושל המידע הראשי(metaDataDao) וכך הוא ממש את הפעולות בתוכנית אשר מקבלות את המידע ממאגר הנתונים ואשר שולחות מידע למאגר הנתונים והLogger עצמו יודע איך להוסיף ולקרוא מידע באמצעות חבילה שנקראת Gson אשר משתמש בקבצי json. 

כל ממשק כזה(Dao) מקבל או מחזיר את האובייקט מסוג המחלקה אשר משתמשים בו בתוכנית למשל metaData ומי שממש את המחלקה הזאת(Logger) יודע לתרגם את זה בשפה של מאגר הנתונים .

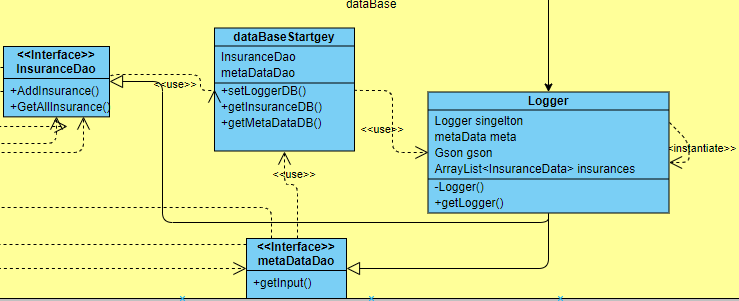
בעצם במחלקות ה- GUI שלנו נשתמש בממשקי ה DAO האלה על מנת לקבל ולעדכן את המידע הרצוי.

המלקה Logger הצהרנו עליה שתהיה final וכך לא יהיה ניתן בכלל להשתמש בה מה שמחייב את השימוש בממשקים השונים metaDataDao ו InsuranceDao. ובעצם כך המתכנת אשר מקבל את כל החבילה כקופסא שחורה איננו צריך להבין איך הlogger עובד זה בגלל השילוב של שתי תבניות העיצוב data access object ו singleton.

אם נרצה להוסיף מאגרי נתונים שונים מה שנצתרך רק שיממשו את ממשק הביטוחים וממשק המידע הראשי.

**Strategy**

התבנית משמשת כדי שיהיה ניתן לבחור שמתודה מסוימת תתממש לפי האסטרטגיה מסוימת ויהיה ניתן לשנות בהתאם כלומר, שמתודה תוכל לממש כמה אחד מתוך כמה אלגוריתמים על פי הצורך של הלקוח.

אצלנו שילבנו את התבנית הזאת כך שהיא לא תהיה רק מתודה אלא שימוש בCLASS שלם. כך שהמחלקה dataBaseStartgey זוהי מחלקה ללא בנאי אשר משתמשים במחלקה עצמה ולא באובייקט שלה, המחלקה הזאת מחליטה באיזה מאגרי נתונים נשתמש. המחלקה הזאת מיועדת לפיתוח עתידי כאשר מאגרי הנתונים לא יהיו json או שיהיו עוד מאגרי נתונים נוספים וגירסאות שונות להם כך המחלקה הזאת תגדיר עם איזה מחלקה שמטפלת במאגרי נתונים אנו נשתמש.

המחלקה מחזיקה את כל ממשקי ה DAO והיא מגדירה איזה מחלקה תממש אותן על ידי החזרת המחלקה אשר ממשת אותן.

אם רוצים להוסיף מימוש למאגר נתונים שונים חדשים יש לעדכן את המחלקה dataBaseStartegy בהתאם.

**דיאגרמת מחלקות:**

דיאגרמת המחלקות להלן נועדה להדגיש את תבניות העיצוב, מצורפת בתיקייה Extras דיאגרמה מלאה יותר.

