בע"ה

קורס במיני פרוייקט במערכות חלונות (דוט נט)
הנחיות למיני פרויקט –
מערכת לניהול מבחני נהיגה, משרד התחבורה
ה'תשע"ט

תוכן עניינים

מבוא

היעד: כתיבת תוכנה (=פרוייקט) מתפתחת במהלך הקורס.

המטרה: תרגול ויישום של:

- עקרונות שפת #C,
- עקרונות ארכיטקטורה של תוכנה,
- יצירת ממשקי משתמש באמצעות תשתית הפתוח המודרנית WPF
- חשיבה תכנותית ולמידה עצמית.

הערה: ההכוונה במסמך זה היא כללית בלבד, שכן חלק מן הדרישה היא להפעיל חשיבה יצירתית. כמו כן, חלק מהמבחן הסופי יתבסס על נושאים תכנותיים שתתבקשו להתמודד איתם במהלך פיתוח הפרויקט. חשוב להדגיש שציון מתחת ל-85 בקורס זה כמו בכל קורס מעשי אחר איננו משמעותי בעיניהם של המעסיקים בתעשייה...

דגשים:

- העבודה תתבצע אך ורק בזוגות (לא בבודדים ולא בשלישיות). הזוגות יהיו קבועים לכל אורך הסמסטר (לא ניתן לעבור במהלך הסמסטר מקבוצה לקבוצה. לא ניתן להיות שותף עם חבר שרשום בקבוצה שונה. בכל אחד מן המקרים האלה התרגיל ייפסל).
- כל אחד מהשותפים חייב להיות שותף מלא בכל אחד מן השלבים. (במידה ויתברר כי שותף אחד עשה שלב מסוים ואלו שותף אחר עשה את החלק האחר, הציון יהיה פרופורציונלי למספר השלבים שנעשו ע"י כל אחד מן השותפים)
- בסוף הסמסטר, ייבחר פרויקט מצטיין מכל קבוצת תרגול. מבין אלו, ייבחרו הפרויקטים המצטיינים, והם יזכו את יוצריהם בפרס. **המדד העיקרי למצוינות הוא מקוריות, למידה עצמית, וכמובן תכנות נכון**.

תהליך ההגשה:

- הפרויקט מחולק לשלושה שלבים המבוססים זה על זה.
- בסיום כל אחד מן השלבים נקבע מועד הגשה והצגה של הנעשה עד כה.
- בדיקת שני השלבים הראשונים תעשה בצורה כללית (בעיקר תיבדק הרצה תקינה, הערות על איכות העבודה, ועמידה מדויקת בלוח הזמנים הנדרש). הציון על שני שלבים אלו לא יועבר לידי הסטודנטים.
- עם הגשת השלב האחרון והמסכם תתבצע בדיקה יסודית של העבודה שתכלול גם הגנה (מעיין בחינה בע"פ) של שני השותפים על העבודה, כ"א בנפרד.
  הציון הסופי על הפרוייקט יתייחס לכל רכיביו, וכן - לאיכות ההצגה, ולעמידה בזמני ההגשה.
- ההגנה על השלב האחרון תתבצע בשיעור האחרון של הסמסטר ולא יאוחר מכך.
- ציון המעבדה יהיה 50% מהציון הסופי (מורכב מ10% תרגילי בית ו40% פרוייקט) והוא ישוקלל עם ציון המבחן רק בתנאי שהציון במבחן הוא לפחות 55.

הנחיות להגשת הקבצים במערכת המודל:
בכל אחד מן השלבים, יש להעלות, **עוד לפני ההגנה**, קובץ zip עם הפתרון למודל (לפי ההנחיות בתרגיל המבוא):
שם הקובץ יהיה Project01_xxxx_yyyy_dotNet5779
dotNet5779 = שם הקורס והשנה העברית
01 = שם השלב (01,02,03)
xxxx = 4 ספרות אחרונות בתעודת הזהות של בן הזוג הראשון
yyyy = 4 ספרות אחרונות בתעודת הזהות של בן הזוג השני
נא להקפיד על פורמט זה על מנת למנוע מצב של אי קבלת ציון על שלב מסוים.

הערה:
מומלץ ביותר לעבוד עם אחת מתוכנות ה GIT לניהול גרסאות ושיתוף פעולה בין שני בני הזוג.

תיאור כללי

בפרויקט זה נכתוב מערכת (חלקית בלבד) לניהול מבחני נהיגה לשם קבלת רשיון ("טסטים") מטעם משרד התחבורה.

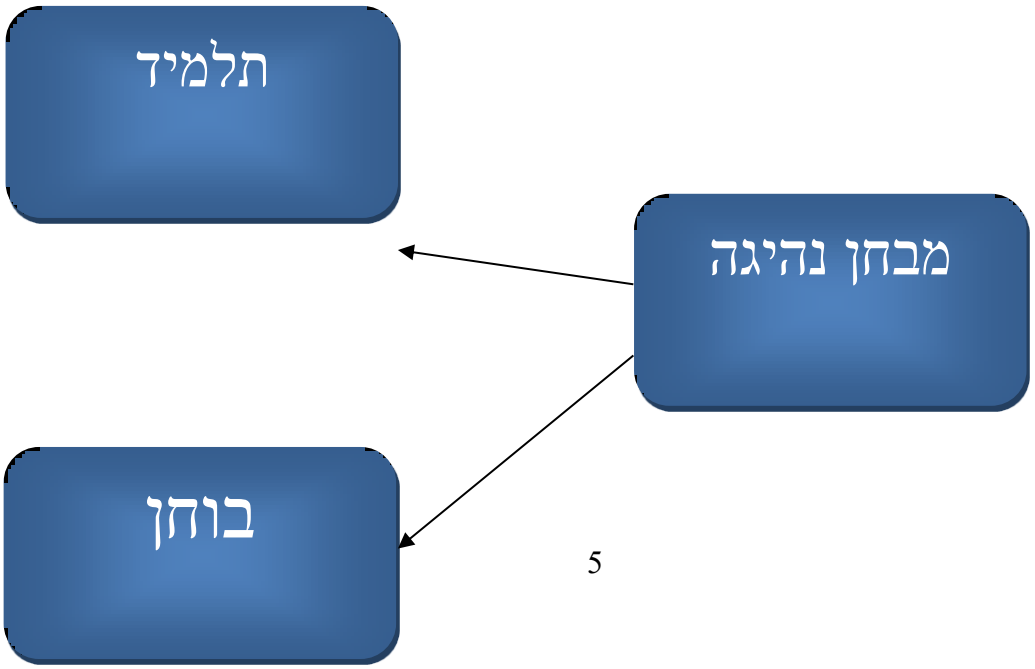
מערכת ניהול ה-"טסטים" הינה רעיון חדש ומהפכני, שהוצע ע"י משרד התחבורה, כחלק מהפרטת מערכת ה-"טסטים", בניסיון:

- להגדיל באופן משמעותי את הזמינות של מבחני הנהיגה
- ולהנגיש אותם בצורה פשוטה לציבור הרחב.

הארגון החדש מנוהל ע"י מערכת אינטרנטית ומכיל מספר ישויות. אנחנו נדמה את המערכת באופן חלקי. אמנם, את הפרויקט הזה עדיין לא נכתוב כאתר אינטרנטי בתבנית של שרת\לקוח (מפני שזה חורג מנושאי הקורס) אך נכתוב את רוב השכבות בצורה כזאת שהרוב יהיה חופף למערכת אינטרנטית מקבילה.

כל תלמיד נהיגה (שעונה על קריטריונים מסוימים), יכול לרשום את עצמו למבחן נהיגה בתאריך ובזמן מסוים שנוח לו, ומכתובת יציאה שנוחה לו.
המערכת משבצת לתלמיד בוחן מתוך מאגר הבוחנים העומד לרשותה, בהתאם לזמינותו של הבוחן.
עם סיום המבחן הבוחן מעדכן את נתוני המבחן – ובין היתר - הוא מעדכן את כתובת היציאה והסיום, ציון הנבחן והערות רלוונטיות.
לארגון יש מאגר בוחנים ענק בפריסה ארצית 24 שעות ביממה.

תרשים המתאר את הישויות במערכת והקשרים בינהן:



תיאור ישויות המערכת

תלמיד

מכילה פרטים כגון: מספר ת.ז., שם משפחה, שם פרטי, תאריך לידה, מין, טלפון, כתובת (רחוב, מספר בית ברחוב, עיר), סוג הרכב עליו למד (רכב פרטי, רכב דו-גלגלי, משאית בינונית, משאית כבדה), סוג תיבת הילוכים (ידני, אוטומטי), שם בי"ס לנהיגה, שם המורה לנהיגה, מספר שיעורי הנהיגה שעבר.

בוחן

מכילה פרטים כגון: מספר ת.ז., שם משפחה, שם פרטי, מין, מספר טלפון נייד, התמחות הבוחן (רכב פרטי, רכב דו-גלגלי, משאית בינונית, משאית כבדה. ניתן להניח שלכל בוחן ישנה רק התמחות אחת) כתובת, טווח מרחק מהכתובת בו הוא מוכן לקיים מבחנים.

מבחן

ישות המקשרת בין התלמיד לבוחן ומכילה את הפרטים הבאים: מספר מבחן, מספר ת.ז. של בוחן, מספר ת.ז. של תלמיד, תאריך המבחן, שעת המבחן, כתובת יציאה, קריטריונים לבדיקה במבחן, ציון המבחן, הערות.

תיאור חלקי הפרויקט בקצרה

<u>שלב ראשון:</u>

נבנה את הפרויקט על פי מודל השכבות.
מקור הנתונים יופיע כפרויקט נפרד ויכיל רשימות (List<T>) של אובייקטים.
הגישה למקור הנתונים תתבצע באמצעות שכבת ה- DAL.
שכבת ה BL תכיל את הלוגיקה הנדרשת,
ושכבת ה UI תהיה ממשק console פשוט.

<u>שלב שני:</u>

פיתוח ממשק המשתמש לממשק ויזואלי גרפי.
תשתית הפתוח הנדרשת של הממשק הוויזואלי הינה WPF.

<u>שלב שלישי:</u>

נשנה את שכבת ה- DAL כך שמקור הנתונים יתבסס על קבצי xml.
כמו כן נעבוד עם אתר חיצוני המספק ערכי מרחק אמיתיים (GoogleMaps). נבצע הורדה של הנתונים באמצעות תהליכון, ונשתמש בהם לפרויקט שלנו.

שלב ראשון – הגדרת הפרויקטים עבור מודל השכבות ומימושן

בחלק זה של הפרויקט נגדיר את כל אחת מן השכבות שיהוו יחד את המיני פרויקט שיאפיין את מערך הטסטים של משרד התחבורה.

שלב זה כולל 4 חלקים כדלהלן:

חלק א' - הגדרת המחלקות עבור הישויות, כלומר שכבת ה- BE.
חלק ב' - הגדרת דרך העבודה מול מקור הנתונים, כלומר שכבת ה- DAL.
חלק ג' – הגדרת הלוגיקה של המערכת, כלומר שכבת ה- BL
חלק ד' – הגדרת ממשק המשתמש של המיני פרויקט, כלומר שכבת ה- UI.

הערה: כל תקלה כתוצאה מפעולה כלשהי, תגרור אחריה טיפול בחריגות לפי מנגנון החריגות הקיים ב- #C. לדוגמא: הכנסת מספר זהות לא תקין, ניסיון לקביעת מבחן עבור תלמיד שלא קיים במערכת.
החריגות צריכות להתבצע עבור כל שכבה בנפרד, ללא תלות בטיפול בחריגה זו בשכבות האחרות (למשל: אם המערכת לא מאפשרת בשכבת ה UI לקבוע מבחן לתלמיד שלא קיים במערכת, שכבת ה DAL לא תסתמך על כך, אלא תבצע גם היא בדיקה שהתלמיד אכן קיים במערכת).
כמובן שעל החריגות גם לעבור משכבה לשכבה במקרה הצורך.

חלק א – ישויות – שכבת ה BE

להלן נפרט שדות שיכללו במחלקות. **חובה** להוסיף שדות ופעולות נוספים.
שים לב! יש להימנע משימוש בפונקציות public במחלקות (מלבד הToString),
הלוגיקה המרכזית מתבצעת בשכבת ה- BL כפי שיפורט בהמשך.

הגדר פרויקט **מסוג Class Library** בשם **BE** ובו המחלקות הרשומות בהמשך.
חובה להגדיר כל מחלקה בקובץ נפרד.

בקובץ נפרד יש להגדיר את כל הטיפוסים הסודרים (enum) הנדרשים (כגון: סוג הרכב (רכב פרטי, רכב דו-גלגלי, משאית בינונית, משאית כבדה), סוג תיבת הילוכים (ידני, אוטומטי), מין, וכדומה)

בקובץ נפרד נוסף יש להגדיר **מחלקה בשם Configuration** שתכלול את כל המשתנים הגלובליים כשדות סטטיים (ראו בהמשך: מספר השיעורים המינימלי, גיל בוחן מקסימלי, גיל נבחן מינימלי, טווח זמן בין מבחן למבחן, וכדומה)

מבנה המתאר כתובת המכיל:

א. רחוב
ב. מספר בניין ברחוב
ג. עיר

מחלקה בשם Tester שמייצגת בוחן ותכלול:

א. מאפיין שמציין את מספר ת.ז. של הבוחן .(חייב להופיע כזה).
ב. מאפיין שמציין את שם המשפחה של הבוחן.
ג. מאפיין שמציין את השם הפרטי של הבוחן.
ד. מאפיין שמציין את תאריך הלידה של הבוחן.
ה. מאפיין שמציין את המין של הבוחן.
ו. מאפיין שמציין את הטלפון של הבוחן.
ז. מאפיין שמציין את הכתובת של הבוחן
ח. מאפיין שמציין את מספר שנות הניסיון של הבוחן.
ט. מאפיין שמציין את מספר המבחנים השבועי המקסימלי האפשרי של הבוחן.
י. מאפיין שמציין את סוג הרכב בו מתמחה הבוחן.

יא. מאפיין המכיל את ימי העבודה בשבוע וטווח השעות של הבוחן (מטריצה בוליאנית, של שעות עגולות בלבד, בין 9:00 ל15:00, בימים א-ה).
יב. מאפיין של מרחק מקסימלי מהכתובת שלו, שבו בוחן יכול לבחון.
יג. מאפיינים נוספים לפי הצורך.
יד. *ToString*.

מחלקה בשם Trainee שמייצגת תלמיד נהיגה ותכלול:

א. מאפיין שמציין מספר הזיהוי של התלמיד. (חייב להופיע כזה)
ב. מאפיין שמציין את השם הפרטי של התלמיד.
ג. מאפיין שמציין את השם המשפחה של התלמיד.
ד. מאפיין שמציין את המין של התלמיד.
ה. מאפיין שיציין את מספר הטלפון הנייד של התלמיד.
ו. מאפיין שמציין את הכתובת של התלמיד
ז. מאפיין שמציין את תאריך הלידה של התלמיד.
ח. מאפיין שמציין את סוג הרכב עליו למד.
ט. מאפיין שמציין את סוג תיבת הילוכים אותה למד.
י. מאפיין שמציין את שם בי"ס לנהיגה בו למד.
יא. מאפיין שמציין את שם המורה לנהיגה אצלו למד.
יב. מאפיין שמציין את מספר שיעורי הנהיגה שעבר.
יג. מאפיינים נוספים לפי הצורך.
יד. *ToString*.

מחלקה בשם: Test שמייצגת מבחן נהיגה (כלומר את הקשר בין הבוחן לתלמיד) ותכלול:

א. מאפיין שמציין את מספר המבחן. **(קוד רץ ייחודי בן 8 ספרות)**
ב. מאפיין שמציין מספר הזיהוי של הבוחן.(חייב להופיע כזה)
ג. מאפיין שמציין מספר הזיהוי של התלמיד.(חייב להופיע כזה)
ד. מאפיין שמציין את התאריך שנקבע לטסט.
ה. מאפיין שמציין את התאריך והשעה של הטסט (ניתן להניח שטסט הוא תמיד באורך שעה ברוטו, כולל הגעת בוחן למקום)
ו. מאפיין שמציין את הכתובת היציאה לטסט.
ז. מאפיינים שמציינים קריטריונים שונים הנבדקים בעת מבחן הנהיגה (שמירת מרחק, חניה ברוורס, התבוננות במראות, איתותים וכדומה), ואת ההצלחה בהם (עבר/נכשל)
ח. מאפיין שמציין את ציון הנבחן (עבר\נכשל).
ט. מאפיין שמציין הערת הבוחן.
י. *ToString*

חלק ב' – DAL במימוש רשימות

להלן נפרט פעולות אפשריות. **חובה** להוסיף פעולות נוספות.

הוסף פרויקט מסוג class library בשם DAL ובו בצע את הפעולות הבאות:

א. הגדר (interface) ממשק בשם Idal .
בממשק הנ"ל הגדר חתימה של פונקציות שימושיות עבור האפליקציה כגון:

- הוספת בוחן.
- מחיקת בוחן.
- עדכון פרטי בוחן קיים.

- הוספת תלמיד.
- מחיקת תלמיד.
- עדכון תלמיד קיים.

- הוספת מבחן
- עדכון מבחן בסיומו

- קבלת רשימת כל הבוחנים.
- קבלת רשימת כל התלמידים.
- קבלת רשימת כל המבחנים.

ב. צור פרויקט חדש בשם: DS והגדר בתוכו מחלקה בשם DataSource שתכיל את הנתונים (רשימות) של הישויות שלנו.
מחלקה זו תכיל 3 רשימות סטטיות (אוספים מסוג list<>) של המחלקות הנמצאות ב- BE.
(הערה: זוהי מחלקה זמנית לשלב זה של הפרוייקט. ניתן לאתחל את הרשימות באמצעות קלט, אך מומלץ לאתחל את הרשימות במספר ערכים בקוד האתחול, כדי להקל על העבודה)

ג. הגדר מחלקה בשם: Dal_imp אשר מממשת את ממשק ה- Idal הנ"ל.
הפונקציות של המחלקה הזאת יעבדו מול הרשימות שבמחלקה DataSource.

דגש - שים לב!

בגלל חיוניות ההפרדה בין השכבות, שכבת ה DAL לא תאפשר לשכבות האחרות גישה למקור הנתונים– כלומר, במקרה הצורך, שכבה זו תשלח עותק של הנתונים, ולא את הנתונים עצמם.

הערה:

על שכבת ה- DAL לוודא שמספר הת.ז. איננו קיים כבר.
שכבת ה- DAL היא זו שמוסיפה לישות "מבחן" מספר מזהה רץ.
על מנת לקבוע את המספר המזהה הרץ ההתחלתי, יש לשמור נתון מתאים במחלקת Configuration

חלק ג' – שכבת ה- BL

הגדר ממשק (interface) בשם: IBL שבו החתימות של השיטות בדיוק כמו ב- IDAL

על שכבת ה BL לבצע את אכיפת הלוגיקה הבאה (וכמובן – עליכם להוסיף לוגיקה משלכם, לפי הגיון פשוט ודרישות המתאימות לפרויקט שלכם):

- אין אפשרות להוסיף בוחן מתחת לגיל 40
- אין אפשרות להוסיף תלמיד צעיר מגיל 18.
- אין אפשרות להוסיף מבחן לפני שעברו 7 ימים מהמבחן הקודם של התלמיד (אם היה).
- אין אפשרות להוסיף מבחן לתלמיד שעשה פחות מ-20 שיעורים.
- אין אפשרות להוסיף מבחן אם אין בוחן זמין לתאריך המבוקש של המבחן, ובתנאי שהבוחן לא שובץ באותו זמן למבחן אחר. במקרה כזה המערכת יכולה להציע לתלמיד זמן חילופי לביצוע המבחן.
- אין אפשרות להוסיף בוחן למבחן אם הבוחן עבר את מספר המבחנים השבועי הקצוב לו.
- לא ניתן לעדכן מבחן, כאשר לא קיימים כל השדות שהבוחן צריך למלא.
- לא ניתן לקבוע לבוחן / תלמיד שני מבחנים באותה השעה.
- לא ניתן לקבוע מבחן על סוג רכב מסוים לתלמיד שכבר עבר בהצלחה מבחן נהיגה על סוג רכב זה.
- בקביעת מבחן יש להתאים בין סוג הרכב שעליו למד התלמיד לבין סוג ההתמחות של הבוחן

הערה: כל הנתונים המספריים (מספר השיעורים המינימלי, גיל בוחן מקסימלי, גיל נבחן מינימלי, טווח זמן בין מבחן למבחן, וכדומה) צריך להיות שמורים במשתנים סטטיים שבמחלקה configuration, על מנת שניתן יהיה לשנותם באופן פשוט, במידה ויוחלט על שינוי בנהלים.

ובנוסף על שכבת ה BL להוסיף את חתימת הפונקציות הבאות:

- פונקציה שמקבלת כתובת ומחזירה את כל הבוחנים שגרים במרחק X ק"מ מהכתובת. (חישוב המרחקים בין כתובות יעשה בעזרת WebRequest שפונה ל url של GoogleMaps בשלב השלישי והסופי של הפרויקט. בשלב זה יש לקבוע את המרחק ע"י הגרלת מספר מתאים).
- פונקציה שמקבלת תאריך ושעה ומחזירה את כל הבוחנים הפנויים באותה שעה. הפונקציה תבדוק האם התאריך והשעה הינם שעות עבודה של הבוחן והאם הבוחן לא תפוס במבחן אחר באותה שעה.
- פונקציה שיכולה להחזיר את **כל המבחנים** שמתאימים לתנאי מסוים (הכוונה שהפונקציה מקבלת delegate שמתאים לפונקציות שמקבלות מבחן ומחזירות bool וכך מוגדר התנאי)

- פונקציה שמקבלת תלמיד, ומחזירה את מספר המבחנים שנבחן בהם.
- פונקציה שמקבלת תלמיד ומחזירה האם הוא זכאי לרשיון נהיגה (האם עבר בהצלחה מבחן נהיגה).
- פונקציה שמחזירה את רשימת כל המבחנים שמתוכננים לפי יום/חודש

הגדר פונקציות המחזירות את הקבוצות הבאות (ע"י שימוש ב-Grouping)
כל אחת מהפונקציות הבאות תקבל משתנה בוליאני שיציין האם להחזיר את התוצאות בצורה ממוינת (איך למיין זה לפי שיקול דעתכם) ערך ברירת מחדל של פרמטר זה יהיה false (לא ממוין)

- רשימת בוחנים מקובצת (Grouping) ע"פ סוג ההתמחות.
- רשימת תלמידים מקובצת (Grouping) ע"פ ביה"ס לנהיגה בו לומדים
- רשימת תלמידים מקובצת (Grouping) ע"פ המורה לנהיגה אצלו לומדים
- רשימת תלמידים מקובצת (Grouping) ע"פ מספר הטסטים שביצעו

הגדר מחלקה חדשה בשכבת ה- BL שתממש את הממשק IBL הנ"ל.

המימוש יכלול שימוש ב Linq to object וביטויי למבדא.

חובה לעשות שימוש בכישרון היצירתיות שלכם ולהוסיף לפחות עוד 6 פונקציות שמתאימות להיות בשכבת ה-BL ועובדות על הנתונים המוחזרים מה- DAL.

יש לעשות שימוש בלפחות 4 ביטויי Linq בשכבת ה- BL- וה- DAL כל אחת – השימוש צריך להיות מגוון, ולכלול שימוש בnew select וב let.
יש לעשות שימוש בלפחות 4 ביטויי למבדה בנוסף לביטויי Linq דלעיל.
וכן לעשות שימוש בanonymous delegates, ושימוש ב- predicates ב- FUNC.

<u>הערות:</u>

על שכבה זו לוודא שמתקיימים חוקים לוגיים בסיסים כמו לדוגמה:

- אם מוסיפים מבחן מסוים, אזי יש לוודא שהתלמיד שנבחן אכן קיים.
- יש לוודא שתלמיד נבחן רק על סוג הרכב שעליו למד.
- לא ייתכן שתלמיד ייכשל ברוב הקריטריונים בטסט, ויקבל ציון עובר.
- וכו...

כאן תתבצע ההגשה הראשונה – אפשר בממשק קונסול או ממשק גרפי פשוט.

לשם הבדיקה של הדברים אנו ניצור פרוויקט זמני בשם PL.
תוכל לממש אותו או בעזרת console application או ממשק גרפי WPF פשוט . אפשר בשלב זה לעשות הכל בחלון אחד.
בכל מקרה עליך לקרוא לפונקציות הנמצאות ב- BL ולבדוק אותן.

<u>הערה:</u>

כדי לעבוד בשלבים מומלץ בהתחלה ליצור משהו שבודק את ה- BE אח"כ את הפונקציונליות ב- DAL כדי לראות שאכן זה עובד ואז לחבר את ה- BL.

כמו כן יש לבצע תיעוד מסודר:

- ✓ מעל הפונקציות באינטרפסים של IDAL ו- IBL תוך שימוש ב \\\
- ✓ להוסיף תיעוד עבור מימושים רק מעל פונקציות יצירתיות ב- BL שאוכפות נהלים מסוימים שהמצאתם כמו למשל "מצא את כל הבוחנים מירושלים שמומחים ברכב פרטי" בקיצור, תיעוד רק עבור דברים שאינם טריוויאליים או שימוש בשאילתות מורכבות. כל השאר אין צורך לתעד.
- ✓ יש להשתמש בשמות משמעותיים למשתנים

שלב שני – שכבת ה- UI

בשלב זה ניצור ממשק גרפי לפרויקט באמצעות תשתית הפתוח WPF.

ניצור פרויקט חדש מסוג : WPF
ונקרא לו **PLWPF**.
כמובן שיהיה עליך ליצור לו reference מתאים ל-BL וכן ל- BE .
עליך להגדיר ישות בשם: BL מטיפוס: IBL
יש לעשות שימוש ב- Factory Method עבור DAL ועבור BL. כמו כן, יש לוודא שאנו לא יוצרים כל פעם מופע חדש של הBL ושל הDAL.

לתכנן את המסכים אשר יקראו ויאפשרו למשתמש לגשת לפונקציונאליות הנמצאת ב- BL.
עליך לתכנן מסך לכל ישות, כאשר יהיו לפחות המסכים הבאים:

- מסך הוספת, עדכון, מחיקה של בוחן.
- מסך הוספת, עדכון, מחיקה של תלמיד.
- מסך הוספת, עדכון , מחיקה של מבחן. כאשר מעוניינים להוסיף "מבחן" יש לחבר בין התלמיד לבין אחד מן הבוחנים האפשריים ע"פ אילוצים רלוונטיים כמו למשל: ימי עבודה, מרחק.
- בנוסף – לפחות עוד 3 מסכים המציגים שאילתות שמימשתם ב- BL.

התכנית תיפתח במסך ראשי המפנה לאפשרויות השונות.
הערה: כאשר ישנו מסך המאפשר הוספת או עדכון ערך כמו "סוג הרכב" וכדו' שכבר קיימים במערכת, יש לאפשר בחירה של שדה זה ע"י פקד ComboBox שבו נקבל רשימה ממנה יש לבחור.

כמו כן כאשר אנו מאפשרים עדכון, לאחר בחירה מתוך רשימת הישויות לעדכון, יופיע המפתח על המסך ויתמלאו יתר השדות באופן אוטומטי, כאשר לא ניתן יהיה כמובן לשנות את ערך המפתח.
מומלץ מאד שניתן יהיה לחפש את הרשומה לעדכון גם על סמך נתונים אחרים בישות (למשל – שם, שם משפחה וכדומה), ולא רק על סמך המפתח (תעודת הזהות)
ישנה אפשרות שהוספה עדכון ומחיקה יהיו למעשה אותו מסך, בהתאם לצורך באותו רגע.

(אין להוסיף פונקציונאליות מעבר לCRUD [=יצירה, קריאה, עדכון, מחיקה] בסיסי לDAL)

כמו כן עליך לדאוג להוסיף זריקה של חריגות במקומות המתאימים ותפיסה של החריגות כדי שהתכנית לא "תעוף" במקרה שפעולה מסוימת נכשלה בזמן הרצה.

בשכבה זו עלייך לטפל בתהליכון המזמן את ה GoogleMaps כך שיפעל מהעצם המזמן אותו. ניתן לדחות את הביצוע של כך לשלב השלישי של הפרויקט.

לאחר השלמת לימוד כל הWPF יש צורך לדאוג שכל האלמנטים שנלמדו בWPF מוטמעים בפרויקט. היכן ואיך לממש, זה תלוי בכל אחד ואחת לפי יצירתיותם.
כמו כן, יש לטפל בחריגות שנזרקות ומטופלות ששייכות לשכבת UI בעצמה כמו למשל שמשתמש לא מילא את כל השדות הנדרשים או שמילא שדות נומריים בתווים וכו'.

הערה חשובה: יש לעשות שימוש ב:
Resources, data binding, converter, data context, dependency property, trigger, תהליכון.

כאן תתבצע ההגשה השניה במספר.

שלב שלישי – עדכון שכבת ה- DAL
מימוש באמצעות Linq-To-XML

הוספה ומימוש מחלקה נוספת ב-DAL המממשת את ה- IDAL באמצעות Linq – to –XML:

עליך להוסיף לפרויקט **DAL** מחלקה נוספת בשם Dal_XML_imp

יש להכין קבצי xml שיחליפו את האוספים שיצרנו כבר (כלומר יחליפו את מקור הנתונים) אחד לכל אחת מהישויות, אשר ייכתבו בפורמט המתאים למבנה הישות אותה הוא מייצג. קבצים אלו יהיו בתוך תיקיה נפרדת ב- solution.

יש ליצור את האתחולים, אפשרות שמירה וטעינה של קובץ וכן אפשרות תשאול ע"י שאילתת Linq.
לאחר מכן יש לממש את כל המתודות של הממשק של ה- IDAL.

לגבי עבודה עם קבצי XML מקומיים, ניתן להיעזר בהדרכה להלן:

עבור אתחול קבצי הXML:
לדוגמה:
studentRoot = new XElement("students");
כעת נוכל כל פעם להוסיף \להוריד ולעדכן אלמנטים בקובץ
במידה והקובץ כבר קיים, פשוט לא ניצור אותו.
הקוד יהיה בערך כך:

```
public XmlSample()         {
            if (!File.Exists(FPath))
                CreateFiles();
        }

        private void CreateFiles()
        {
            studentRoot = new XElement("students");
        }
```

עבור המספר הרץ של חלק מהמחלקות ועוד כמה הגדרות:

הבעיה:
כשמימשנו את המספר הרץ השתמשנו במשתנה סטטי שגדל במהלך התוכנית.
כעת שנשמור את הנתונים ב- xml בפעם הבאה שנפתח את התוכנית אותו משתנה יתאפס שוב ואז המספר הרץ יתחיל מ- 0

הפתרון:
נגדיר קובץ XML בשם config.xml ושם נגדיר את המספר הרץ ועוד הגדרות שנרצה
כל פעם שנשמור אובייקט שמשתמש במספר הרץ נעדכן גם את קובץ ה config.xml
בפעם הראשונה שה dal נוצר הוא ידאג לעדכן זאת במחלקה.

הערה:
עבור קובץ ה- config ועוד קובץ אחד של אחת המחלקות, יש להשתמש ב- linq to xml עבור כל פעולות ההוספה עדכון ומחיקה ושליפה.
במימוש של שאר המחלקות ניתן להשתמש ב- serialize, כלומר לעבוד עם הנתונים ב- list ובסופו של דבר לשמור את זה ל- XML באמצעות xmlSerialize

כאן תתבצע בעז"ה ההגנה הסופית

בהצלחה !

תאריכי הגשות:

הגשה 1 – שלב א (חלקים א-ג עם UI בסיסי). תאריך הגשה: עד לג' בטבת
הגשה 2 – שלב ב (חלק ד, UI עם WPF). תאריך הגשה: עד לכ"ג בטבת
הגשה 3 – לאחר שלב ג' סופי – **תאריך הגשה** : מפגש אחרון בסמסטר (עד לט' בשבט)

(ההגנה בשבוע האחרון בשעות המעבדה). לאחר סיום הסמסטר לא יתבצעו בדיקות.

נספח

דוגמא לחישוב מרחק ומשך זמן נסיעה בין 2 כתובות באמצעות גוגל מפות (google maps)

מצורפת למטה דוגמא של נסיון אחד לחישוב מרחק ומשך זמן נסיעה בין 2 כתובות, הדוגמא כוללת:

1. בניית מחרוזת בפורמט מסויים שמכילה את 2 הכתובת (כתובת באנגלית או בעברית)
2. פניה לשרות גוגל (google service) בעזרת בקשת רשת (WebRequest)
3. קבלת תשובה בפורמט XML
4. ניתוח התשובה, ע"פ האפשרויות הבאות:
   a. לא נתקבלה תשובה, כנראה בגלל עומס ברשת.
   b. נתקבלה תשובה, אך לפחות אחת הכתובות לא תקינה.
   c. נתקבלה תשובה, והיא כוללת מרחק ומשך הזמן בנסיעה בין 2 הכתובות.

מכיוון שהרשת עמוסה לעיתים, אזי בקשת רשת יכולה לארוך כמה חלקי שניות וכן להיענות רק לאחר כמה נסיונות.
לכן, עליכם לעטוף את הנסיון האחד שמודגם בקובץ זה בתוך תהליכון **BackgroundWorker**.
התהליכון ימשיך ויבצע נסיונות לבקשת רשת עד שתתקבל תשובה.
מספר הפניות לשרות המפות החינמי של גוגל מוגבל ומי שעובר אותו מסומן כלקוח מטריד ומפסיק לקבל תשובות.
ההמלצה היא לבצע בתוך התהליכון השהיה של 2 שניות לפחות בין בקשת רשת אחת לשניה ולא לבצע יותר מ2500 פניות ביום.
אם עברתם את מגבלת הפניות ליום אזי ייתכן שלאחר 24 שעות תוכלו לשוב ולפנות. אך אין לדעת...
השתמשו בשרות של גוגל מפות בחוכמה.

לעניין הפרוייקט:
במידה ומאיזושהיא סיבה לא קבלתם תשובה למרחק בין 2 כתובות, אנא השתמשו בערך ברירת מחדל כלשהו שתקבעו.

דוגמת הקוד לנסיון אחד לחישוב מרחק ומשך זמן נסיעה בין 2 כתובות:

```
using System;
using System.IO;
using System.Net;
using System.Xml;

string origin = "pisga 45 st. jerusalem"; //or "אחד העם 100 פתח תקווה" etc.
string destination = "gilgal 78 st. ramat-gan";//or "ז'בוטינסקי 10 רמת גן" etc.

//build the url that includes the 2 addresses
string url = @"http://maps.googleapis.com/maps/api/distancematrix/xml?origins="
+
origin + "&destinations=" + destination +
"&mode=driving&sensor=false&language=en-EN&units=imperial";

//request from google service the distance between the 2 addresses
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

WebResponse response = request.GetResponse();

Stream dataStream = response.GetResponseStream();
StreamReader sreader = new StreamReader(dataStream);
string responsereader = sreader.ReadToEnd();
response.Close();

//the response is given in an XML format
XmlDocument xmldoc = new XmlDocument();
xmldoc.LoadXml(responsereader);

if (xmldoc.GetElementsByTagName("status")[0].ChildNodes[0].InnerText == "OK")
//we have an answer
{
   if (xmldoc.GetElementsByTagName("status")[1].ChildNodes[0].InnerText ==
"NOT_FOUND")
   //one of the addresses is not found
      Console.WriteLine("one of the adrresses is not found");
```

```
    else
    {//2 of the addresses are OK
        //display the returned distance
        XmlNodeList distance = xmldoc.GetElementsByTagName("distance");
        double dist =
Convert.ToDouble(distance[0].ChildNodes[1].InnerText.Replace(" mi", ""));
        Console.WriteLine(dist * 1.609344); //each mile is 1.609344 kilometer

        //display the returned duration
        XmlNodeList duration = xmldoc.GetElementsByTagName("duration");
        string dur = duration[0].ChildNodes[1].InnerText;
        Console.WriteLine(dur);
    }
}
else
//we have no answer, the web is busy, the waiting time for answer is limited
(QUERY_OVER_LIMIT), we should try again (at least 2 seconds between 2
requests)
{
    Console.WriteLine("We have'nt got an answer, maybe the net is busy...");
}
```

דוגמה לXML שמתקבל במידה ומתקבלת תשובה תקינה של מרחק ומשך זמן נסיעה בין 2 כתובות:

XML Visualizer

Expression: InnerXml

Value:


```
<?xml version="1.0" encoding="UTF-8"?>
- <DistanceMatrixResponse>
    <status>OK</status>
    <origin_address>El'ad, Israel</origin_address>
    <destination_address>HaHashmona'im St 4, Petah Tikva,
      Israel</destination_address>
  - <row>
    - <element>
        <status>OK</status>
      - <duration>
          <value>1016</value>
          <text>17 mins</text>
        </duration>
      - <distance>
          <value>11311</value>
          <text>7.0 mi</text>
        </distance>
      </element>
    </row>
  </DistanceMatrixResponse>
```


Close

דוגמה לXML שמתקבל במידה ומתקבלת תשובה אך אחת מהכתובות אינה תקינה:

XML Visualizer

Expression: InnerXml

Value:


```
<?xml version="1.0" encoding="UTF-8"?>
- <DistanceMatrixResponse>
    <status>OK</status>
    <origin_address/>
    <destination_address>El'ad, Israel</destination_address>
  - <row>
    - <element>
        <status>NOT_FOUND</status>
      </element>
    </row>
  </DistanceMatrixResponse>
```


Close