בס"ד תשפ"ג סמסטר א'

**מיני פרויקט במערכות חלונות**

*שלב 7*

מטרות השלב

עבודה עם תהליכונים גולמיים

עבודה עם BackgroundWorker

יצירת סימולטור

שימוש בחסימת שימוש סימולטני (בו זמנית) של מתודות אובייקט ושל קטעי קוד מתוך תהליכונים שונים

הנחיות לביצוע התרגיל והגשתו

* חובה להשתמש בכלי לניהול גרסאות git ובאתר github.org
* חובה לבצע את השלב באותו המאגר ובאותו ה-Solution כמו שלב 0 (מקדים)
* חובה להגיש במודל קישור על פי הנחיות ההגשה
* חובה להקפיד על פורמט זה על מנת למנוע מצב של אי קבלת ציון על תרגיל מסוים
* חובה לקרוא את התיאור הכללי של הפרויקט לפני תחילת העבודה על כל שלב.

### 

הקדמה

בשלב זה נוסיף תהליך של סימולציה למערכת שלנו, על מנת לדמות פעולות שנעשות לאורך חיי המערכת.

לצורך כך נתנסה בעבודה עם ריבוי תהליכונים.

**תיאור כללי של הסימולטור**

התפקיד של הסימולטור הוא ליצור הדמייה בזמן אמת של התקדמות דינמית של מצב ישויות הפרויקט לפי הדרישות המוגדרות בתאירו הכללי של הפרויקט. הסימולציה תכלול הצגת שעון רץ (המתקדם כל שניה) והצגת בחירה ו-\או התקדמות של מצב אובייקט הישות שנבחר באותו הרע. יתבצע עדכון אוטומטי של מצב ונתוני הישויות בכל החלונות. במקרים המתאימים יוצג progress bar של התקדמות אובייקט הישות המתאימה שיתעדכן פעם בשניה. **החלוקה של המטלות תחולק לבסיס ולבונוסים בפירוט בהמשך ההנחיות.**

התמיכה בסימולטור תפותח בשתי מקומות - בשכבת התצוגה (PL) ובפרויקט נפרד של הסימולטור בעצמו. שכבת ה-PL "תתקשר" עם הסימולטור, והסימולטור יעבוד מול BL (יבקש ממנה נתונים וישלח אליה עדכונים). מבנה תלויות הפרויקטים מבחינת הוספת הסימולטור יראה ככה: PL ⇒ Simulator ⇒ BL.

כמו כן, לבונוס: בשכבת BL יוצרו אירועים (event delegate) פרטיים עבור הישות שהסימולטור יעשה הדמייה עליה, ובממשק של BL יוספו מתודות רישום\ביטול של מתודות השקפה (observers) עבור האירועיים האלה. שכבת ה-PL תרשום את המתודות שלה לאירועים האלה על מנת לעדכן את נתוני הישויות בחלונות הפתוחים כנ"ל.

**הנחיות מפורטות**

### 

### **סימולטור:**

1. פרויקט הסימולטור
   1. הוסיפו ב-Solution פרויקט חדש מסוג ספרית מחלקות בשם Simulator
   2. עשו לחיצה כפולה על שם הפרויקט ב-Solution Explorer (ייפתח לעריכה קובץ ניהול הפרויקט עם סיומת csproj) - יש להוסיף לו את השורות הבאות (המודגשות) בסוף האלמנט PropertyGroup:

*<Project Sdk="Microsoft.NET.Sdk">*

*<PropertyGroup>*

*<TargetFramework>net6.0</TargetFramework>*

*<ImplicitUsings>enable</ImplicitUsings>*

*<Nullable>enable</Nullable>*

**<BaseOutputPath>$(SolutionDir)\bin\</BaseOutputPath>**

**<OutputPath>$(SolutionDir)\bin\</OutputPath>**

**<AppendTargetFrameworkToOutputPath>false</AppendTargetFrameworkToOutputPath>**

**<AppendRuntimeIdentifierToOutputPath>false</AppendRuntimeIdentifierToOutputPath>**

**<UseCommonOutputDirectory>true</UseCommonOutputDirectory>**

*</PropertyGroup>*

*</Project>*

* 1. בתפריט ההקשר של הפרויקט (לחיצה על כפתור ימני כאשר הסמן מעל שם הפרויקט) בחרו מאפיינים (Properties) וודאו שהשדה Default namespace ריק (תחת כותרת Application>General, כתוצאה מהשורה המודגשת האחרונה לעיל), ובשדה Base output path (תחת כותרת Build>Output, כתוצאה מהשורה המודגשת הראשונה לעיל) מופיע:

**$(SolutionDir)\bin\**

* 1. **נ.ב.** יתר ההוספות שעשינו לעיל לא משתקפות בממשק המאפיינים של פרויקט ולכן עשינו אותם ע"י עריכה ישירה בתוך הקובץ csproj
  2. בתפריט ההקשר של הפרויקט בחרו הוספת הפניות פרויקט (...Project Reference), ובו סמנו את הפרויקט BL
  3. בתפריט ההקשר של הפרויקט בחרו …Build Dependencies>Project Dependencies וודאו שהסימון דומה לסימון ההפניות כנ"ל
  4. הוסיפו את הפרויקט Simulator לתלויות פרוייקט (project dependencies) של פרויקט PL

1. להחלטתכם יצירת החוזים של הפרויקט (ממשק + טיפוסים) - המשך ההנחיות יהיו ללא ממשק לצורך פשטות ההנחיות. מי שירצה לעשות עם ממשק - צריך להקפיד על הכללים בדומה למה שנעשה בשכבת BL.
2. **תיצרו מחלקה Simulator**
   1. המחלקה תהיה סטטית

* 1. במחלקה תהיה מתודה ציבורית ל**הפעלת ההדמיה**
     1. המתודה תיצור תהליכון גולמי שיבצע את ההדמיה
  2. כל התכונות במחלקה - פרטיות
  3. במחלקה תהיה מתודה ציבורית לעצירת סימולציה ותכונה מתאימה שתוגדר עם volatile
  4. במחלקה יהיו תכונות event delegate עבור:
     1. השלמת עצירת הדמיה
     2. כל עדכון בבחירת אובייקט ישות או השלמת התהליך המבוצע עם אובייקט ("עדכון התקדמות" ההדמיה)
     3. כל אירוע נוסף שתמצאו לנכון
  5. מתודות ציבוריות עבור רישום וביטול לאירועים האלה (הפרמטר - delegate עבור המתודה שתירשם לאירוע המתאים)

1. הסימולטור יבצע את ההדמיה כמתואר בתיאור הכללי של הפרויקט

### 

### **שכבת תצוגה:**

1. בחלון הראשי של הפרויקט יש להוסיף כפתור הפעלת הדמיה
2. בלחיצה על הכפתור ייפתח חלון חדש עבור הצגת הסימולטור
   1. בחלון יהיה כפתור סיום הדמיה
   2. לא תתאפשר סגירת החלון ע"י לחיצה על סימן "X" של החלון
   3. בחלון יוצג שעון רץ של הדמיה שיתעדכן פעם בשניה
   4. בחלון יוצג אובייקט שכרגע מנוהל ע"י ההדמיה, המצב הקודם שלו והמצב הבא שלו (כולל זמן תחילת תהליך מעבר וזמן צפוי להשלמתו)
      1. בונוס: יוצג progress bar של התקדמות התהליך שיתעדכן פעם בשנייה לפחות
   5. שימו לב : פתיחת חלון הסימולטור לא חוסמת חלונות אחרים.
3. **בפתיחת חלון הסימולטור**:
   1. תצרו פועל רקע (Background Worker) והפעילו אותו (RunWorkerAsync)
   2. מטלת הרקע של פועל הרקע (DoWork) יבצע את הפעולות הבאות:
      1. רשמו מתודות משקיפות (**ראו בהמשך**) לאירועי הסימולטור
      2. הפעילו את הסימולטור ע"י זימון מתודת ההפעלה (ראו [לעיל](#7shr94x87ahi))
      3. תעשו לולאה על דגל עצירת פועל הרקע (CancellationPending)
      4. בתוך לולאה דאגו להתעוררות כל שניה ובקשת דיווח התקדמות (ReportProgress) מתאימה
   3. עדכון התצוגה (ProgressChanged) יוכל לבצע את העדכונים כדלהלן
      1. דאגו לזיהוי סוג בקשת עדכון ולהעברת מידע מתאים כנ"לעדכון שעון
      2. עבור סוג בקשת עדכון מומלץ להשתמש תכונת Percentage של הארגומנט
      3. עבור מידע נוסף מומלץ להשתמש בתכונת UserState של הארגומנט
      4. אם יש מספר אובייקטים\נתונים במידע לעדכון - מומלץ להשתמש ב-Tuple או ברשימה ArrayList (המימוש שלכם ניתן להחלטתכם)
      5. יבצעו העדכונים הבאים (לפחות):
         1. עדכון אובייקט
         2. עדכון מצב אובייקט (אם נדרש)
         3. לבונוס: עדכון התקדמות (עבור progress bar)
   4. בסיום הדמיה (RunWorkerCompleted):
      1. ביטלו את רישום מתודות ההשקפה לאירועי סימולטור
      2. השלימו את עדכוני התצוגה ו-\או נתוני התצוגה בהתאם לדרישות והצרכים שלכם
4. **בחלון הסימולטור** הוסיפו מתודות השקפה של תבנית עיצוב Observer עבור אירועי סימולטור כפי שמופיע
   1. חתימות יהיו מתאימות למה שנדרש באירועים ( event delegate) של הסימולטור
   2. משקיף סיום הדמיה יפעיל את המתודה של הדלקת דגל עצירת פועל הרקע (CancelAsync)
   3. משקיף עדכון מהדמיה יבצע בקשת דיווח התקדמות (ReportProgress) מתאימה תוך העברת מידע מתאים לעדכון התצוגה
5. לבונוס - בכל **שאר** החלונות המתאימים הקשורים לשינוי מצבי ישויות בהדמיה:
   * 1. הוסיפו מתודות השקפה (תבנית עיצוב Observer) לאירועי הוספה\עדכון\מחיקה של אובייקטים של הישויות המתאימות
     2. צרו פועל רקע:
        1. במטלת הרקע של פועל הרקע :
           1. רשמו את המתודות לאירועי שכבת BL ע"י מתודות חדשות מתאימות בממשק מתאים של BL (**ראו בהמשך**)
           2. רשמו מתודת עצירת פועל רקע בסימולטור
           3. המתינו לסיום (תוך התעוררות פעם בשניה)
        2. בעדכון התקדמות - עדכנו את התצוגה בהתאם לשינוי הנדרש ע"י הפעלת דיווח התקדמות
     3. בסגירת חלון (**לפני השלמת הסגירה!**) דאגו לעצירת פועל הרקע בצורה לגיטימית!

### **שכבה לוגית ושכבת נתונים:**

עקב שימוש באותם הנתונים מתוך תהליכונים השונים (הסימולטור, התהליכון הראשי של האפליקציה, פועלי רקע), שלמות הנתונים עלולה להיפגע (מכיוון שלמשל שנמשוך נתונים באמצע העדכון שלהם - ואז נקבל נתונים משובשים). לכן אנו חייבים לדואוג ל**מניעת שיבוש שלמות הנתונים עקב פניות בו זמנית ממספר תהליכונים**. את הפרטים תלמדו בהמשך התואר בקורס "**עקרונות מערכות הפעלה**" בנושאים: סנכרון בין תהליכים, שימוש במשאבים משותפים, קטע קריטי, וכו'.

1. דאגו לחסימת כניסה סימולטנית (בו זמנית) למתודות ציבוריות של שכבת הנתונים ע"י הוספת אטריבוט סנכרון לפני כל מתודה ציבורית בשני המימושים של שכבת הנתונים:

**using System.Runtime.CompilerServices;**

**...**

**class Student**

**{**

**[MethodImpl(MethodImplOptions.Synchronized)]**

**public int Add(int studentId) ...**

**...**

**}**

1. בשכבה הלוגית BL דאגו לחסימת כניסה למתודות ציבוריות שמבצעות בקשות\הוספה\עדכון\מחיקה של נתונים כלשהם כמו בסעיף לעיל
2. עניין נוסף בשכבה הלוגית BL:
   1. פעולות מסוימות דורשות עדכון מתואם של מספר ישויות נתונים
   2. לכן יש לעטוף סדרת פעולות האלה בבלוק חסימת גישה לשכבת הנתונים
   3. החסימה תתבצע עבור אובייקט שכבת מימוש הנתונים (שבדרך כלל שמרתם אותו בתכונה פרטית בשם dal\_ או s\_dal או dal - בדוגמא הבאה נשתמש בשם dal\_ לצורך הפשטות) בצורה הבאה:

**lock(\_dal)**

**{**

**Requests to the data layer //*סדרת פניות לשכבת הנתונים***

**}**

* 1. אם ביצעתם את התהליך של סדרת פניות לשכבת נתונים ע"י מתודה עזר פרטית, אזי הוראת המתודה היא הוראה יחידה שעבורה ניתן לרשום:

**lock(\_dal) calling a private method //*זימון מתודת עזר***

* 1. בשאילתות נתונים מרובים:
     1. השאילתא לא מתבצעת באמת בעת זימון (ביצוע מושהה כפי שלמדתם)
     2. לכן אי אפשר לחסום את כל העבודה עם הנתונים המרובים - וזה גם לא יעיל
     3. לכן יש לבצע חסימות איבר-איבר (אחד-אחד):
        1. הוציאו את הכנת אלמנט הישות לפעולה נפרדת שבין
        2. בצעו את החסימה כנ"ל בתוך המתודה
        3. בימים הקרובים נוציא המלצות יותר מפורטות בעניין הזה

1. לבונוס בשכבת BL:
   1. הוסיפו במימושים מתאימים של ממשקים מתאימים אירועי השקפה (event delegate) עבור עדכון\הוספה\מחיקה של כל הישות המתאימה המנוהלת ע"י הדמייה
   2. הוסיפו בממשקים מתאימים מתודות רישום\ביטול של משקיפים ומימשו אותם בהתאם
   3. שילבו הפעלת האירועים בכל מקום נדרש על מנת לאפשר עדכון תצוגות מתאימות