אירועים – Events

אירוע היא דרך מסויימת לחשוף ולאפשר עבודה עם Delegate, בצורה המאפשרת ותומכת ב"תכנות מונחה אירועים".

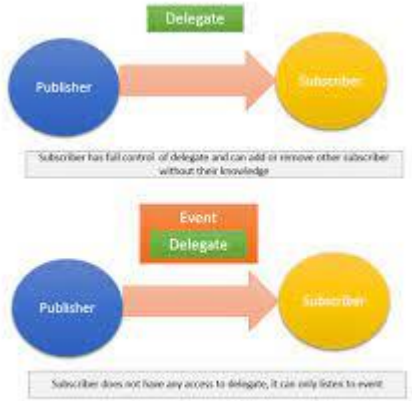
מה זה תכנות מונחה אירועים? תכנות מונחה אירועים זה פיתוח אפליקציה שמגיבה כאשר מתרחשים פעולות מסויימות שהוגדרו מראש ושלהן מפתח המחלקה מאפשר להגיב.

אירוע מאפשר לאוביקט לסמן שהתרחש אירוע מסויים. באמצעות Events מממשים תבנית עיצוב מוכרת בשם observer (תבנית מהתחום שנקרא Design Patterns).

המחלקה שמעוררת את האירוע נקראת Publisher (מפיץ), והמחלקה שמקבלת או תופסת את האירוע (את הסיון של ה Publisher עם המידע המצורף אליו), נקראת – Subscriber ("המנוי/הנרשם").

כמה מחלקות יכולות להירשם לאירוע בודד. בדרך כלל המפיץ מעורר את האירוע כשמשהו קורה, והנרשם שמעוניין לדעת על התרחשות האירוע הזה, נרשם לאירוע ומטפל בו (מגיב לו) כשהוא מתרחש.

לדוגמה: אם יש לי מחלקה שמנהלת רשימת סטודנטים, המחלקה מאפשרת להוסיף סטודנט, וחושפת אירוע הוספת סטודנט, כך שמחלקות אחרות יכולות לבצע פעולה כאשר התווסף סטודנט למערכת. למשל, מחלקה אחרת תפקידה לשלוח מייל לסטודנט ולמרצה כאשר דווח שהתווסף סטודנט, מחלקת ניהול חשבונות ידווחו שסטודנט התווסף, יקבלו את פרטי הסטודנט וישלחו לו בדואר חשבונית, ומחלקה אחרת תעדכן את אתר האינטרנט של המכללה על כך שתלמיד נוסף הצטרף.



התמונה מהאתר: https://dailydotnettips.com/back-to-basic-events-in-c/

ב #C אירועים מיושמים באמצעות מנגנון של delegate ומהווים מעטפת ל delegate.

אירועים הם members של מחלקות (כמו properties) שלא ניתן להפעילם מחוץ למחלקה שהגדירה אותם גם אם הגדרנו את רמת הגישה public. אירוע שהוכרז כ public יאפשר למחלקות אחרות שימוש ב- + = ו - = באירוע, אך ה invocation מותר רק מתוך המחלקה שהצהירה על האירוע (ה delegate עצמו יכול להיות מוגדר במחלקה אחרת, הוא בדרך כלל מוגדר בפני עצמו ולא בתוך מחלקה כלשהי).

כל מה שנעשה עם אירועים ניתן היה לעשות גם עם delegates אבל התחביר של events גם מגן על מי ש"נרשם" לארוע משינוי ע"י מחלקה אחרת, וגם המונחים שלו תומכים בתפיסה של רישום לארוע, ולכן מקובל להשתמש בו בניגוד לסתם delegates לאירועים תלויי זמן ובאירועים שחוזרים על עצמם במהלך התוכנית.

הצהרה על אירוע

נצהיר בשני שלבים:

1. נגדיר delegate

2. ניצור משתצנה מסוג ה delegate עם מילת המפתח event.

דוגמה:

```
public delegate void Notify();  // delegate

public class ProcessBusinessLogic
{
    public event Notify ProcessCompleted; // event, pay tension to thevent keyword

}
```

בואו נראה דוגמה, איך אובייקט מעורר אירוע:

```
public delegate void Notify();  // delegate

public class ProcessBusinessLogic
{
    public event Notify ProcessCompleted; // event

    public void StartProcess()
    {
        Console.WriteLine("Process Started!");
        // some code here..
        OnProcessCompleted();
    }

    protected  virtual  void  OnProcessCompleted()  //protected  virtual
method
    {
        //if ProcessCompleted is not null then call delegate
        ProcessCompleted?.Invoke();
    }
}
```

EventArgs

מקובל פעמים רבות ב #C להשתמש להעברת מידע עם האירוע ב EventArgs, פעמים רבות נירש את EventArgs ונוסיף מאפיינים משלנו שנעביר עם ה EventArgs. זו מחלקה שמוגדרת ב namespace שנקרא system ומשמשת אותנו לטובת העברה של מופעים שלה כחלק מהחתימה של ה delegates של האירועים שלנו.

Event Handler

ישנם delegates מובנים בשפה לטובת שימוש באירועים, ה delegate המובנים מייצגים אירועים כלליים או אירועים ספציפיים שנרצה להתשמש בהם כאשר נפתח אירוע דומה.

כזה הוא EventHandler וגם EventHandler<T> הגרסה הגנרית של EventHandler. כפתורים וקונטרולים רבים ב Windowsforms משתמשים ב delegate הנ"ל.

אפשר לראות שימוש בהם בדוגמה הבאה:

```
class Program
{
    public static void Main()
    {
        ProcessBusinessLogic bl = new ProcessBusinessLogic();
        bl.ProcessCompleted += bl_ProcessCompleted; // register with an
event
        bl.StartProcess();
    }

    // event handler
    public static void bl_ProcessCompleted(object sender, EventArgs e)
    {
        Console.WriteLine("Process Completed!");
    }
}

public class ProcessBusinessLogic
{
    // declaring an event using built-in EventHandler
    public event EventHandler ProcessCompleted;

    public void StartProcess()
    {
        Console.WriteLine("Process Started!");
        // some code here..
        OnProcessCompleted(EventArgs.Empty); //No event data
    }

    protected virtual void OnProcessCompleted(EventArgs e)
    {
        ProcessCompleted?.Invoke(this, e);
    }
```

במאמר זה הועתקו דוגמאות רבות, ותורגמו הסברים מהאתר: https://www.tutorialsteacher.com/csharp/csharp-event

סיכום

1. אירוע הוא מעטפת סביב delegate ותלויה בו.

2. השתמש במילת המפתח "event" עם משתנה מסוג הdelegate כדי להכריז על אירוע.
3. השתמש ב EventHandler delegate או ב- EventHandler<TEventArgs> מובנה לאירועים נפוצים.
4. מחלקת ה publisher מעלה אירוע, ומחלקת ה subscriber נרשמת לאירוע ומספקת את מתודה שתטפל באירוע, כלומר, תגיב לו.
5. מקובל לקרוא לשיטה (מתודה) המעעוררת אירוע עם ה Prefix "On".
6. החתימה של שיטה המגיבה חייבת להתאים לחתימת ה delegate.
7. הרשמה לאירוע באמצעות אופרטור +=. בטל את ההרשמה באמצעות האופרטור -=. לא ניתן להשתמש באופרטור =.
8. העבר נתוני אירועים באמצעות EventHandler<TEventArgs>
9. השתמשו במחלקת הבסיס של EventArgs ליצירת מחלקת נתוני אירועים מותאמים אישית.
10. ניתן להכריז על אירועים סטטיים, וירטואליים, seald ומופשטים (אבסטרקטיים).
11. interface יכול לכלול את האירוע כ member.
12. אם ישנם מספר Subscribers הם יקראו אחד אחרי השני (באופן סינכרוני).

תרגילי כיתה ובית

1. כתוב מחלקה עם מאפיין שם, אם קיבלנו שם ארוך מ 10 תווים נעורר אירוע TooLongNameEvent.

2. כתוב תוכנית שמאפשרת להקליד מספרים לתוך משתנה מסוג מספר שלם שוב ושוב, אם הוקלד המספר 999 ייזרק אירוע LuckyNumberWasEntered. מחלקה אחרת תירשם לאירוע ותדפיס "Lucky number was entered".

3. המכללה החליטה שכל סטודנט חמישי שייקלט למערכת יקבל הנחה, כתוב תוכנית שמאפשרת להזין סטודנטים למערכת. על כל סטודנט חמישי שייקלט נזרוק אירוע. מחלקה אחרת תתפוס את האירוע ותדפיס: " You get 5% discount".

4. כתוב מחלקה Shape שמייצגת צורה על מסך, למחלקה יהיה ארוע Outch שמייצג שאובייקט אחר כלשהו במסך נגע בשטח הצורה. נתמוך בצורות: נקודה וריבוע. (המחלקה צריכה לחשוף פונקציה שתבדוק מיקום של צורות אחרות במסך נגיד כוכבית שיכולה לנוע עם תנודות העכבר, הפונקציה תעורר את האירוע אם קיבלנו מיקום לבדיקה והתברר שהאובייקט הזר פגע בשטח הצורה).

5. כתוב תוכנית המייצגת פתרון תוכנה של בנק, התוכנית מאפשרת למשתמשים לבצע לוגאין למערכת. נחלק את זה כך: נגדיר מחלקה User עם המאפיינים: שם משתמש וסיסמה. הגדר Hard coded מספר משתמשים במערכת.

כתוב מחלקה שמאפשרת למשתמש לנסות לבצע לוגאין למערכת, אם הפרטים של המשתמש והסיסמה נכונים, נזרוק אירוע SuccessLoginEvent, אחרת, UnsuccesfulLoginEvent, ונעביר את הסיבה של אי ההצלחה למי שיירשם לאירוע ("User name does not exist" , "Wrong password").

נכתוב UI שמאפשר להזין שם משתש וסיסמה, מחלקת ה UI מעבירה נתונים למחלקה שבודקת את הפרטים, ומדפיסה הודעה מתאימה אם הלוגאין הצליח או נכשל (מחלקת ה UI מדפיסה).

6. בהמשך לתרגיל הקודם, נאפשר ללקוחות הבנק - "המשתמשים" שההזדהות שלהם הצליחה, להפקיד כספים לבנק. (כרגע אתם תאפשרו להקליד את סכום ההקלדה, "בעולם האמיתי" הסכום יתקבל ע"י הרכיב בכספומט שסופר את הכסף או סורק את ההמחאה). נוסיף את סכום ההפקדה לחשבון המשתמש (יש ליצור מחלקה שמייצגת חשבון). כאשר יתווסף סכום לחשבון יתעורר אירוע הפקדה, עם פרטי המפקיד וסכום ההפקדה.

מחלקה אחרת תתפוס את האירוע ותציג כמה כסף יש ללקוח בחשבון.

מחלקה נוספת המטפלת במאזני הכספים של הבנק, תתפוס את האירוע ותציג כמה כסף מופקד בבנק כעת מכלל הלקוחות.

7. **תרגיל למידה עצמית**: הוסיפו לאירוע שכתבתם בפתרון לשאלה מס' 2 מידע שעובר עם האירוע.
העבירו את מספר הפעמים שהוכנס בהם מספר אחר עד שהוקלד 999.
השתמשו ב EventHandler הגנרי! (חיפוש קצר עם גוגל).