פעולות סטטיות

הגדרה:
פעולה (או מתודה) היא יחידת קוד שנועדה לבצע משימה מסוימת, כגון חישוב, עדכון נתונים, או החזרת ערך. פעולות יכולות לקבל פרמטרים ולהחזיר ערכים.

למה להשתמש בפעולות?

1. הפחתת חזרתיות ושימוש חוזר בקוד (DRY - Don't Repeat Yourself):
   - פעולות מאפשרות להימנע מכתיבת קוד חוזר, ובכך לחסוך זמן, להקטין טעויות ולייעל את תחזוקת הקוד. ניתן להכניס קוד לפעולה ולהשתמש בה שוב ושוב במקום לשכפל את הקוד בכל מקום. זה גם מקל על שמירת עקביות, שכן כל שינוי בקוד נעשה במקום אחד.
   - דוגמה: במקום לחזור על חישוב a + b במקומות שונים בתוכנית, ניתן להגדיר פעולה Add(a, b) ולקרוא לה בכל מקום שצריך.
2. שיפור קריאות והפשטת הקוד:
   - פעולות מקלות על קריאת הקוד, מכיוון שהן ממקדות את הלוגיקה ומפשטות את מבנה התוכנית. כל פעולה מבצעת משימה אחת ברורה, דבר שמייעל את הבנת הקוד ומקל על התחזוקה שלו.
   - דוגמה: פעולות כמו CalculateTotal() או ProcessOrder() מאפשרות להבין בקלות מה קורה בתוכנית, במקום לקודד את כל הלוגיקה בתוך קטעי קוד ארוכים.
3. בדיקות יחידה (Unit Testing):
   - פעולות מאפשרות לבדוק כל חלק בנפרד, מה שמקל על תהליך הבדיקות. ניתן לבדוק את הפעולות באופן מבודד ולוודא שכל פעולה פועלת כראוי, דבר שמייעל את תהליך הפיתוח.
   - דוגמה: פעולה כמו ValidateUserInput() יכולה להיבדק בצורה מבודדת כדי לוודא שהיא פועלת כמצופה.

פעולות המחזירות ערך-

כתיבת פעולות המחזירות ערך היא חלק מרכזי בתכנות, ויש להן יתרונות רבים. הסיבה העיקרית לכתיבת פעולות כאלו היא שהיא מאפשרת לבצע חישובים או מניפולציות על נתונים, ולאחר מכן להחזיר את התוצאה שניתן להשתמש בה בהמשך התוכנית.

פעולות שאינן מחזירות ערך-

פעולות המשמשות להדפסה/עדכון. בכותרת הפעולה במקום טיפוס ערך מוחזר יופיע void.

קבלת פרמטרים בפעולה-

פרמטרים הם משתנים שמוגדרים בתוך סוגריים בעת הגדרת הפעולה, והם יכולים להיות מכל סוג נתון כמו int, double, string ועוד. הפעולה משתמשת בפרמטרים האלה כדי לבצע את הפעולה שלה, ולאחר מכן היא עשויה להחזיר ערך או לבצע שינוי/הדפסה.

יש לשים לב שהשמות הניתנים למשתנים בהגדרת הפעולה הם רק שמות מקומיים לפעולה עצמה.
כשנקרא לפעולה נוכל לשלוח לה פרמטרים בכל שם שהוא בתנאי שהטיפוס זהה לטיפוס הפרמטרים המוגדרים בכותרת הפעולה.

<u>הגדרת פעולה-כותרת ובלוק הפעולה</u>

כותרת פעולה מוגדרת ע"י:

1. הרשאת גישה-public/private
2. במקרה של פעולה סטטית-המילה השמורה static
3. טיפוס ערך מוחזר-במקרה שהפעולה מחזירה-טיפוס הערך המוחזר, אחרת-void.
4. שם-מרמז על מהות הפעולה. שם פעולה יתחיל באות גדולה.
5. פרמטרים-פרמטרים הנשלחים לפעולה בעת הפעלת (קריאה) הפעולה. אין חובה לקבל פרמטרים. הפרמטרים יופיעו בסוף כותרת הפעולה, בתוך (). כשהפעולה לא תקבל פרמטרים, יגיעו בסוף כותרת הפעולה () -סוגריים ריקות. אם הפעולה הוגדרה כמקבלת פרמטרים-**חובה** לשלוח אותם בעת קריאה לפעולה.

(כרגע-כל כותרות הפעולות שנגדיר יתחילו ב public static).

דוגמאות:

כותרת פעולה המדפיסה hello world:

```
public static void Print()
```

*void פעולה שאינה מחזירה
*Print שם הפעולה
*() הפעולה לא מקבלת פרמטרים.

כותרת פעולה המקבלת שני מספרים שלמים ומחזירה את סכומם:

```
public static int Sum(int x, int y)
```

*int פעולה המחזירה ערך מטיפוס int
*Sum שם הפעולה
*(int x, int y) פרמטרים שהפעולה מקבלת

בלוק הפעולה

קוד הפעולה-הלוגיקה שלה, ייכתב לאחר כותרת הפעולה בין { }.

במקרה של פעולה המחזירה ערך, חובה לכתוב את משפט ה return-מה יחזור מהפעולה. טיפוס הערך המוחזר חייב להיות זהה לטיפוס המוגדר בכותרת הפעולה.

חובה לכתוב משפט return לכל מצב, ז"א, אם משפט ה return נכתב בתוך תנאי/לולאה, חובה לכתוב אחד נוסף מחוץ לו. מכיוון שהקומפלייר צריך לוודא שבכל מצב הפעולה תחזיר ערך במקרה שהגדרנו בכותרת הפעולה טיפוס ערך מוחזר ולא void.

כאשר הקומפיילר קורא את משפט ה return, הוא מחזיר את הערך מיידית ומפסיק את ביצוע הפעולה.

<u>קריאה לפעולה</u>

קריאה לפעולה מתבצעת באמצעות ציון שמה, ו (). כאשר במקרה והפעולה מצפה לקבל פרמטרים, יש לשלוח אותם בתוך הסוגריים.

לדו':

```
Sum(8,12);
```

קריאה לפעולה בשם Sum המצפה לקבל 2 מספרים שלמים. במידה והפעולה מחזירה ערך, נוכל להציב את הקריאה לפעולה לתוך משתנה, להדפיס כך שיודפס הערך המוחזר, להשוות ועוד.

דוגמאות: (בהנחה שהפעולה מחזירה ערך מטיפוס שלם)

```
int num=Sum(8,12);

Console.WriteLine(Sum(8,12));

if(Sum(8,12)>10)...
```

חשוב!

לא ניתן לבצע השמה לתוך קריאה לפעולה:

```
Sum(8,12)=4; X
```

לסיכום:

שימוש בפעולות חיוני כדי להקל על תחזוקת הקוד, להפחית חזרתיות ולייעל את הבדיקות. זהו כלי חשוב לארגון הקוד וליצירת פתרונות ברורים ומובנים. עם זאת, חשוב להשתמש בהן בצורה נכונה וממוקדת כדי למנוע חזרתיות מיותרת או ביצועים נמוכים. כדאי להקפיד על שמות ברורים, תיעוד נאות ושימוש בפעולות ממוקדות כדי להפיק את המרב מהשימוש בהן.