**Predicate – DELEGATE שיכול להצביע לכל פונקציה ש:**

* *מקבלת פרמטר אחד.*
* *סוג הפרמטר בתוך <>.*
* *מחזירה ערך BOOL.*

***Action –***

* *יש להעביר פרמטרים מופרדים בפסיק , 1-16 פרמטרים.*
* *סוג הפרמטר בתוך <>.*
* ***אי אפשר*** *להחזיר ערך .*

**Func *–***

* *מקבלת 1-16 פרמטרים מופרדים בפסיקים.*
* *סוג הפרמטר בתוך <>*
* *הפרמטר האחרון קובע את סוג הערך המוחזר, כל השאר הם פרמטרים שהפונקציה מקבלת*
* *חובה להעביר את סוג הטיפוס המוחזר, אולם לא חובה לקבל פרמטרים לפונ' שניצור.*

***3 עקרונות OOP –***

*1****) הורשה INHERITANCE –***

הורשה מאפשרת לממש את עקרון ה- **reuse** – שימוש חוזר בקוד ללא צורך לכתוב אותו מספר פעמים.

***2) הכמסה ENCAPSULATION –***

**כימוס**, הנקרא לעיתים **הסתרת מידע**, מאפשר הסתרת כל המרכיבים הפנימיים של האובייקט ומאפשר גישה אליהם אך ורק דרך הפונקציות של האובייקט המהוות את הממשק.   
מכאן שאנו מגנים על מאפייני האובייקט באי מתן גישה ישירה אליהם, אלא אך ורק דרך פונקציות ממשק שהוגדרו לאותו אובייקט והן לא מאפשרות לבצע במאפייניו שינויים בלתי הולמים או חוקיים.

כימוס מספק את הגבול שבין מאפייניו הפנימיים של האובייקט לבין הממשק החיצוני שלו, בכך שהוא לא מאפשר שינוי באופן ישיר של משתני האובייקט, אלא רק דרך פונקציות הממשק שלו.

*3)* ***פולימורפיזם רב צורתיות*** *–*

- נותנת ל[מתכנת](https://he.wikipedia.org/wiki/%D7%9E%D7%AA%D7%9B%D7%A0%D7%AA) את היכולת לממש [אלגוריתמים](https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D) ו[מבני נתונים](https://he.wikipedia.org/wiki/%D7%9E%D7%91%D7%A0%D7%94_%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D) לשימוש כללי, ולגזור מהם צורות שימוש שונות בהתאם לעצמים ולנסיבות המשתנות, באופן מפורש או מרומז.

*- ניתן ליצור אובייקט מסוג אב ולהצביע על כל אובייקט לא משנה אם הוא אב או בן.*

*- כאשר בן יורש מאבא הוא יכול לגשת לכל המשתנים שלא PRIVATE*

***מהו OOP*** *–*

*מתדולוגית פיתוח קוד שמפתחת תוכנה בצורה של עצמים על מנת לאפשר קוד אינטואיטיבי שמדמה את העולם האמיתי ברמת הקוד.*

***הבדלים בין מחלקה אבסטרקטית לממשק ונקודות משותפות בניהן.***

***מחלקה אבסטרקית*** *–*

*האם היורש הוא גם המחלקה האבסטרקטית ? מחלקה ראשונית מופשטת.*

*לא ניצר ליצור ממנה מופעים*

*אפשר לרשת ממנה*

*אפשר לרשת מחלקה אבסטרקטית רק אם המחלקה שיורשת גם אבסטרקסית*

*ממשק אומר אני מחייב את מיי שממש אותי לקיים פעולות מסוימות.*

*תמיד יתחיל INTERFACE עם I גדולה*

***נקודות משותפות : לדעת 3***

*1)משניהם לא ניתן ליצור מופעים.*

*2)בשתיהם לא ניתן לכתוב פונקציה ללא מימוש.*

*3)משניהם ניתן ליצור משתנה מהטיפוס שלהן.*

***ההבדל בין מחלקה ASTRACT לבין ממשק (INTERFACE) :***

*1) INTERFACE יכול לממש מחלקה אחת .מחלקה אחת יכולה לממש כמה INTERFACES*

*2) באבסטרקטית היורש שלה חייב להיות גם מסוג האבסטרקטית (האם גם אני מסוג המחלקה ? או שרק יש בי את המאפיינים שלה ואני לא קשור אלייה?), INTERFACE אפשר להפעיל על מי שיכול לממש את הפונקציות ללא צורך במהות משותפת.*

*3) לממשק אין בכלל CTOR לעומת במחלקה אבסטרקטית יש CTOR שיתרחש כאשר יוצרים אובייקט מהסוג של מי שירש מהמחלקה האבסטרקטית*

*4) לממשק אי אפשר משתני PRIVATE אפשר רק PUBLIC ובמחלקה אבסטרקטית אפשר PRIVATE וגם PUBLIC*

*5)בABSTRACT CLASS מותר להוסיף משתנים עם ערך התחלתי ובINTERFACE אסור.*

***לדעת – מה ההבדל בין מחלקה למופע של מחלקה?***

* ***מחלקה-*** *תבנית ליצירת מופעים שמכילה משתנים ופעולות, אינה תופסת מקום בזיכרון.*
* ***מופע של מחלקה –*** אובייקט שיצרנו מהתבנית של המחלקה. האובייקט יכיל בתוכו את כל הפונקציות והמאפיינים שהוגדרו במחלקה*.*
* **מה ההבדלים בין – REF וVAL**
* REF – כל משתנה שמכיל בתוכו הפנייה (כתובת) – פוינטר.(סטרינג מערך אובייקט וכו..)
* VAL – מכיל את הערך ישירות בתוך המשתנה.*(כל ערך מספרי, ערך בוליאני , CHAR)*
* *כאשר מבצעים השוואה בין ערכי REF בודקים ששניהם מצביעים לאותו אובייקט.*
* *כאשר מבצעים השוואה בין ערכי VAL (==) משווים בין הערך בתוך המשתנה.*
* *בשליחה לפונקציות – כאשר שולחים VAL – הערך לא ישתנה מחוץ לפונקציה.*

*לעומת REF שאם נשלח בעזרתו לפונקציה גם בסוף הפונקציה יישמרו השינויים*