תכנות מונחה עצמים

* **מהן שלושת היסודות לתכנות מונחה עצמים?**

הורשה, הכמסה (getter setter),פולימורפיזם *(רב צורתיות – ניתן ליצור אובייקט מסוג אב ולהצביע על כל אובייקט לא משנה אם הוא אב או בן.) (רב צורתיות – ניתן ליצור אובייקט מסוג אב ולהצביע על כל אובייקט לא משנה אם הוא אב או בן.)*

***הסבר של אנה על פולימורפיזם*: כאשר בן יורש מאבא הוא יכול לגשת לכל המשתנים שלא !!! PRIVATE לעומת זאת משתנה שהוא Protected ניתן לגשת גם לחלקים מהם הוא יורש.**

**בנוסף כאשר בן יורש מאבא הוא קורא בצורה אוטומטית ל constructor- (הבנאי) של האבא.**

**כאשר אני יוצר אובייקט מסוג בן מתבצעת גם קריאה מסוג האב.**

* **הסבר את שלושת היסודות לתכנות מונחה עצמים:**
* **הורשה** ( **INHERITANCE**):
  + הורשה הינה אחד מהעקרונות החשובים ביותר בתכנות מונחה עצמים (OOP). הורשה מאפשרת לממש את עקרון ה - **reuse** - כלומר שימוש חוזר בקוד ללא צורך לכתוב אותו מספר פעמים.
* **הכמסה (getter setter)** **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.
* **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 תכנות מונחה עצמים?**

מתודולוגית פיתוח קוד שמפתחת תוכנה בצורה של עצמים (אובייקטים) על מנת לתת קוד אינטואיטיבי שמדמה את העולם האמיתי ברמת הקוד, במילים אחרות ביצוע של האנשה ומניעה של חזרה על שורות של קוד.dry code-"don’t repeat yourself"

* **מה ההבדל בין מחלקה לבין מופע מחלקה?**
* **מחלקה** היא תבנית ליצירת מופעים ומשתנים ,והיא **אינה תופסת מקום בזיכרון**.
* בעוד מופע הוא INSTANCE(התייחסות) של אותה המחלקה שמכיל את המשתנים + הפונקציות ותופס מקום בזיכרון (בHEAP-).

**הסבר נוסף:**

* ***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 ותופס מקום בזיכרון (בHEAP-)*.*

* **ref type ו-value type?**
* **REF**- כל **משתנה שמכיל** בתוכו **הפנייה** (כתובת) –פוינטר. (סטרינג ,מערך, אובייקט וכו'..).
* **VAL** – **מכיל** את ה**ערך** ישירות **בתוך המשתנה**. *(כל ערך מספרי, בוליאני , CHAR).*
* **מה ההבדלים בין REF ל- ? VAL**

1. משתנה מטיפוס REF הוא משתנה שמכיל הפנייה או פוינטר לערך, בעוד VAL הוא ערך שמכיל ישירות את הערך שניכנס בתוכו.
2. VAL יכול להיות או מספרי(number) או בוליאני (Boolean) או צ'אר (char) ויהיה לו ערך מינימום ומקסימום ותוכנו יושב ישירות בתוך התא בזיכרון, בעוד REF הוא מצביע לתאים בזיכרון.
3. בהשוואה בין ערכי REF מתבצעת השוואה בין התאים שמצביעים להם בזיכרון, בעוד VAL יבדוק את הערך בכל משתנה.
4. אם שולחים משהו מסוג VAL לפונקציה אז השינוי לא ישמר בעוד בשיחה של REF אז מה שייגשו אליו זה הכתובת והשינוי בערך ישתנה.

* **מה ההבדל בין מחלקה ASTRACT לבין ממשק (INTERFACE) ?**

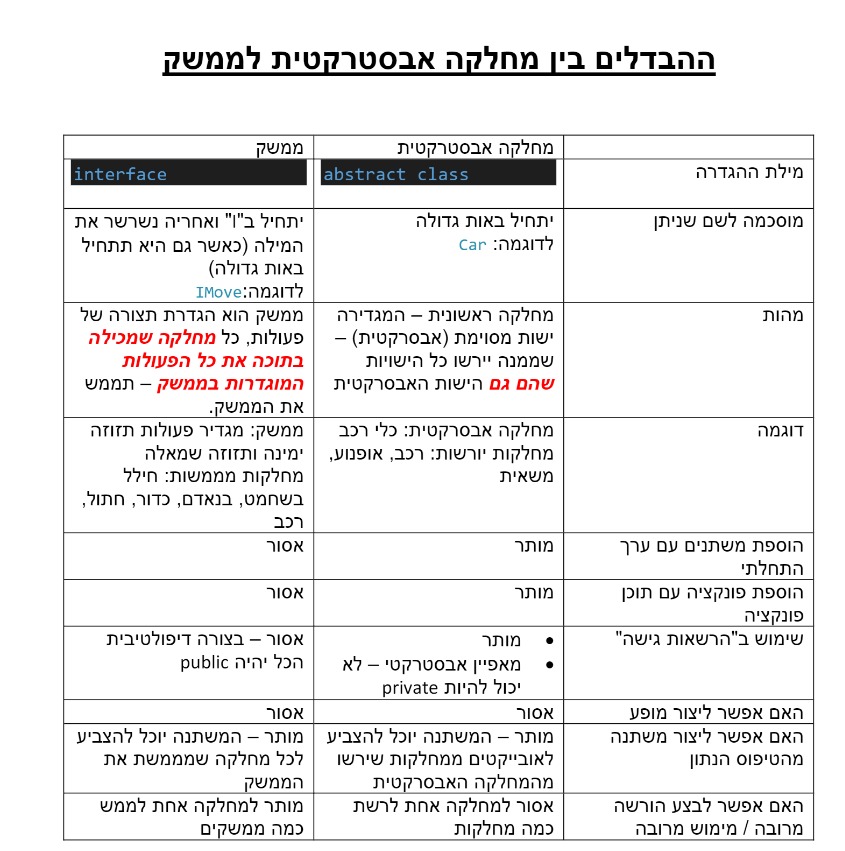
1. INTERFACE (ממשק) יכול לממש מחלקה אחת. לעומת זאת מחלקה אחת יכולה לממש כמה INTERFACES (ממשקים).

2. באבסטרקטית היורש שלה חייב להיות גם מסוג האבסטרקטית (האם גם אני מסוג המחלקה ? או שרק יש בי את המאפיינים שלה ואני לא קשור אליה?), 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. ב- "ABSTRACT CLASS" מותר להוסיף משתנים עם ערך התחלתי וב-INTERFACE אסור.

* **ההבדלים בין מחלקה אבסטרקטית וממשק (5 הבדלים):**
* מחלקה אבסטרקטית תהיה מחלקת בסיס שתוריש את תוכנותיה(משתניה) למחלקה שהיא מאותה המהות, בעוד ממשק היא פעולה שכל ישות יכולה לבצע.
* מחלקה אבסטרקטית ניתן להוריש פעם אחת בעוד ממשק אפשר להוריש כמה שנרצה.
* ניתן לממש פונקציות בתוך פונקציות אבסטרקטיות בעוד בממשק לא ניתן
* במחלקה אבסטרקטית יש בנאי ובממשק אין.
* מחלקה אבסטרקטית תתחיל באות גדולה לא משנה איזה, בעוד שממשק תתחיל תמיד עם האות "I" ואחריה נשרשר את המילה(כאשר גם היא תתחיל באות גדולה).
* **נקודות משותפות בין** **מחלקה אבסטרקטית וממשק(3 הבדלים)**:
* משניהם לא ניתן ליצור מופעים.
* בשתיהם לא ניתן לכתוב פונקציה ללא מימוש.
* משניהם ניתן ליצור משתנה מהטיפוס שלהן.