**תכנות מונחי עצמים**

**איתן רון**

**שעות קבלה:**

**ימי א' 19-20 חדר M320**

[**http://math.haifa.ac.il/ronn/java/wordfiles**](http://math.haifa.ac.il/ronn/java/wordfiles)

**תכנות מונחי עצמים**

**Object Oriented Programming**

**(OOP)**

**גישה לכתיבת תוכניות**

**שפות תכנות OOP הם בסך הכול שפות המעמידות אמצעים שתומכים בגישה.**

**תכנון ותכנות OOP היא למעשה פורמליזציה (מעין הגדרה רשמית) של שיטות עבודה שנהוגות בכתיבת תוכנה להבדל מתוכניות.**

**תוכנית היא קוד שמוגדרת ע"י פעולה מסוימת (כמו למשל למיין קובץ).**

**תוכנה היא קוד המוגדרת ע"י משימה (למשל להיות מעבד תמלילים).**

**תוכנה היא קוד שמתעדכנת בגרסאות חדשו גם משום שפרטי המשימה משתנים.**

**הקוד של תוכנה מחייב כל הזמן יכולת לעבור שינויים.**

**הגישה המסורתית לפיתוח תוכניות נקראית הגישה הפרוצדורלית או מונחית פעולה**

**Procedural**

**Action driven**

**בעיקרו של דבר הפיתוח של תוכנית בגישה המסורתית בצורה של שרשרת:**

**משימה => הגדרת פעולות => אלגריתמים של יישום הפעולות => להגדרת מבני נתונים שישרתו את האלגוריתמים => קוד ביצועי.**

**סוג כזה של פיתוח נקרא top-down כל פעם יורדים לרמה אבסטרקטית פחות.**

**זה השיטה הקלה ביותר לפתח תוכנית שרצה.**

**הבעיה היא שזה לא מתאים לתוכנה שמוציאה גרסאות חדשות כל הזמן: היא רגישה מאד לשינויים.**

**דרך להתמודד לפחות עם חלק מהבעיות הוא לשנות את נקודת ההשקפה למעין bottom-up בניגוד ל-top-down.**

**ב-OOP מתייחסים אל תוכנית כאוסף של רכיבים : מגדירים את הקוד במושגים של רכיבים, פעולות על רכיבים ויחסי גומלין בין רכיבים.**

**היתרון שבכך, הוא שבתוכנה, הרכיבים הם הגורם הכי יציב.**

**הדבר מקנה יציבות מרבית לחלק מהקוד ומידור מרבי של השינויים.**

**נקודת השקפה אפשרית אחת של OOP היא: "כול רכיב הוא קוד ספרייה".**

**לפחות היסטורי, ב-OOP, שלוש עקרונות בסיסיים:**

* **Encapsulation – כימוס הסתרת מימוש פנימי**
* **Inheritence- הורשה – סוג שימוש בקוד קיים.**
* **Polymorphism – "רב צורתיות".**

**עקרון הכימוס – הגבלת הגישה לרכיבים פנימיים של רכיב רק לקוד מסוים.**

**עקרון הכימוס מבדיל עבור הרכיב בין קוד מימוש לקוד לקוח.**

**קוד הלקוח רק משתמש ברכיב ואין לו חלק במימוש.**

**ניסיון של קוד לקוח לגשת לרכיבי מימוש של רכיב לא תעבור קומפילציה.**

**הקומפילר של השפה מאפשר לנו לכפות נורמה.**

**קוד המימוש מחולק לקוד פנימי וקוד ממשק.**

**קוד הלקוח יכול לגשת רק לממשק.**

**שינוי ברכיב שלא משנה את הממשק אינו יכול לגרום לצורך בשינוי בקוד הלקוח.**

**עקרון ההורשה:**

 **מאפשר להגדיר סוג של רכיב על בסיס רכיב קיים מבלי לשכפל קוד.**

**יעיל במיוחד בהגדרת רכיבים חדשים שהם מקרים פרטיים או הרחבה של רכיבים קיימים.**

**פולימורפיזם - "רב צורתיות"**

**קוד מתפרש אחרת בנסיבות שונות.**

**בעיקרו של דבר - עקרון שחוסך בצורה דרסטית שכפול קוד.**

**אחד המטרות של OOP הוא להקל על פיתוח קוד חדש בעזרת קוד קיים.**

**שכפול קוד והתאמתו הוא הצורה הפחות יעלה של שימוש חוזר בקוד.**

**פולימופיזם מביא למינימום את הצורך בשכפול קוד.**

**הדרך שהוא עושה זאת קשור להורשה.**

**המטרה של פולימורפיזם:**

**חלק (גדול) מהקוד של תוכנה מבטא את הרעיון שרוצים שיתבצע, מבלי לפרט איך.**