

יום ראשון

חזרה על נושאים ניבחרים משנה שעברה

- הסיבות שבגללן התפתחה תורת התכנות מונחה עצמים קשורות בעיקר לרמת הסיבוך הכבדה שתוכניות מסחריות הגיעו אליה. כאשר התוכנית מסובכת, אי אפשר לראות את היער מרוב עצים.
- מתכנתים ומעצבי מערכות תוכנה חיפשו דרך לגשר בין העולם ה"אמיתי" ועולם הקוד (תכנות), כדי שהתוכנית תהיה קריאה ומובנת.
- חיפשו גם דרך לפתח תוכניות גדולות בשלבים, למשל מערכת הפעלה שכוללת עשרות מליוני שורות של הוראות בשפת תכנות, ניבנו בשלבים, על ידי צוותים שונים.
- התכונות של תכנות מונחה עצמים
 - Data encapsulation - שמירת המידע במעטפת מקומית, כך שרק מספר קטן של הוראות בתוכנית יכולות לשנות אותו. זה אומר שאם משהו בנתונים השתבש, אפשר בקלות למצוא את הפקודה 'האשמה' (תזכרו בטבלת מעקב). לעיתים זה ניקרא גם: Data hiding
 - Inheritance - הורשה. אנו לא רוצים בכל תוכנית להמציא מחדש את הגלגל. נניח שהגדרנו מחלקה כללית שניקראת: 'כלב'. נתנו לכלב אפשרות להגדיר תכונות כמו גובה, משקל ושם. כעת ברצוננו להגדיר מחלקה 'כלב פודל'. מן הסתם, גם לפודל יש גובה, משקל ושם, אז במקום להגדיר במחלקה 'כלב פודל' שוב את: גובה, משקל ושם - אנחנו פשוט נירש את התכונות האלה מהמחלקה 'כלב', ונוסיף עוד תכונות כמו "כל כמה זמן לספר את הפודל" - שזו תכונה אופיינית לפודלים, אבל לא לכל כלב. זה אומר שבהורשה, למעשה יורשים את כל התכונות והפעולות של המחלקה הבסיסית (super class או base class) ומוסיפים עוד תכונות ופעולות, כלומר עושים את המחלקה החדשה (sub-class או derived class) יותר פרטית ומסוימת.
 - Polymorphism - רב-צורני. אם יש אפשרות להגדיר פעולה שתוכל לפעול על אובייקטים מסוגים (מחלקות) שונים, ותתן תוצאה דומה, אבל ספציפית עבור אותו אובייקט. לדוגמה פונקציה שמציירת צורה, כאשר הצורה פעם יכולה להיות עיגול, ופעם אחרת ריבוע או מלבן וכו'. זה דבר שמפשט את מראה התוכנית ומשפר את מידת קריאותה.
 - Abstraction או בעברית: הפשטה. האפשרות לקחת עצם או דבר מורכב ולחלק אותו לחלקים פשוטים יותר, ובכך להקל על תכנות כל חלק בניפרד (דומה קצת לפונקציות).
- שיעורי בית
- להשלים את התוכנית שמחפשת מספרים משוכללים.