**פולימורפיזם**

* **שימוש חוזר בקוד מבלי לשכפל אותו ולעדכן**
* **עוצמה מרבית לקוד ספריה**
* **גם ניסיון לאפשר למתכנתים לבטא בקוד רעיון מבלי להתחייב בפרטים.**

**מחלקות אבסטרקטיות**

**Abstract classes.**

**לעיתים קרובות מחלקה שהיא קוד משותף למספר מחלקות איננה רחבה דיה להיות מחלקה מלאה.**

**למשל, עובד, עובד אקדמי ועובד מנהלי במכללה.**

**יכול להיות שהמחלקה "עובד" לא מלאה, ולא ניתן להקצות לה מופע.**

**האמצעי מחלקה אבסטרקטית מאפשרת לנו:**

* **להגדיר שדות**
* **לתכנת מתודות**
* **להכריז על מתודות שאין אנו מממשים ו"לקרוא" להם בקוד של מתודות המחלקה.**
* **מחלקה אבסטרקטית: לא ניתן להקצות דינמית, נחשבת למחלקה בירושה וניתן להכריז על משתנה מסוגה שיכולה להצביע בפועל על מופעים של מחלקות צאצא שלה.**
* **מחלקה שיורשת ממחלקה אבסטרקטית חייבת לדרוס את המתודות האבסטרקטיות אלא אם כן הן בעצמן אבסטרקטיות.**

**abstract class A**

**{**

 **abstract void am();**

 **void m()**

 **{**

 **am();**

 **}**

**}**

 **class B extends A**

 **{**

 **@override**

 **void am()**

 **{**

 **}// am**

 **}**

**A a = new B();**

**a.m();**

**ממשקים interfaces**

* **מהווים מעין מחלקות אבסטרקטיות שמכילות רק הכרזות של מתודות אבסטרקטיות.**
* **מהווים מעין חוזה שמחלקה יורשת חייבת לקיים.**
* **ניתן להגדיר משתנים מסוג ממשק היכולים להצביע למופעים של מחלקות הממשות אותם.**
* **משמשות בעיקר לתאום קודים שעושים את אותה פעולה בהקשרים שונים (למשל, "ביצוע תשלום").**
* **יכול לשמש תאום מאמץ תכנותי על פני קבוצות מתכנתים עצמאיים.**
* **מחלקה יכולה לרשת רק ממחלקה אחת אבל ממספר כלשהוא של ממשקים.**

**דוגמא**

**interface I1**

**{**

 **void1 m1();**

**}**

**interface I2**

**{**

 **void1 m2();**

**}**

**Class A implements I1,I2**

**{**

**public void m1()**

**{**

**}// m1**

**public void m2()**

**{**

**} // m2**

**} // A**

**הכרזה הכי כללית:**

**Class B extends A inplements I1,I2,I3,…**

**אפשר לרשת מכמה ממשקים אבל רק ממחלקה אחת (מלאה או אבסטרקטית, לא משנה)**