



תכנות מונחה עצמים

אובייקט - מקום בזיכרון שיש לו סדר וקיים אליו קישור (רפרנט) טיפוס חפץ
שדות - המשתנים של האובייקט

בא' - הפונק' שיוצרת instance של האובייקט

מתודות - הפונק' של האובייקט

מצב פנימי (state) - מאופיין ע"י ערכי השדות

members - שדות ומתודות

Access Modifiers

לשון	מתודה	מחלקה	
ניתן לגשת מכל מקום	ניתן לקרוא/לערוך מכל מקום	ניתן לרשת מכל מקום	public
ניתן לגשת בתיק package	ניתן לקרוא/לערוך בתיק package	-	protected
ניתן לגשת רק באובייקט	ניתן לעשות רק בתיק אובייקט	-	private

reference - מצביע לאובייקט, נשמר ב heap.

Garbage Collector - מוסעל מפי עצם בזמן הריצה באופן אוטומטי ומחק

unreferenced - זיכרון

String - טיפוס מורכב הקיים בשפה. מכיל מתודות רבות כגון length(), indexOf(char) ואף

מאריך - טיפוס מורכב נוסף הקיים בשפה. אובייקט length (שמר בתור שדה (public))

ומכיל שתי מתודות: equals (השוואה ע"פ רפרנט) ! toString

חמשק - Interface

מאפשר הפרדה בין יכולת לבין לממש.
ממשק מכיל חתימות של הסוג


כל class שמממשת ממשק, מתחייבת לממש את כל החתומות שלו.

```
public interface Shape {  
    public double area();  
}
```

```
public class Circle implements Shape {  
    ;
```

פאלימורפיזם: (האפשרות עקרונית למחלקה Circle לבצע את התור Shape מול אותה וזיכוי לריבוי מתודות. קומפילר עושה זאת באמצעות *upcasting*

Static - מאפשר גישה למתודה / שדה בלי יצירת instance שלו.
מתודות סטיות לא יכולות לגשת למתודות / שדות שאינם סטטיים.

מחלקה	מתודה	שדה	 final
לא יכולה שייתנו אותה	לא יכולה להיפרם	לא יכולה לשנות ערך	

Encapsulation
במתודות getter ! setter של שדות מאיפוסים מורכבים
יש ע"צ instance חס ורק אז להתזירו/להשימו

הורשה Inheritance

כאשר מחלקה יורשת מחלקה אחרת, היא מקבלת את כל תכונות האבא שלה. תורשה יוצרת היררכיה

^{implicit} **downcasting** - מצב לא חיוני. הפעלת מתודה של בן אל מופע של האב. גם אם האב (האב) נטו מייצגים של הבן, תתקבל שגיאה קומפילציה. ניתן לעשות downcast רגיל או ניתן להפסיד את מתודות הבן

אם אנושים downcast, לבן כלשהו של היפוס, התכנית תתקמפל. לבן אם לא ניתן לבצע את ה downcast תתקבל שגיאה זמן-ריצה

default - מופשר מיושם של מתודה ושירות בתיק ממשק. default void... ניתן מאוד אך רצוי לממש בתיק המיחלקות

הערה: ניתן להוסיף לממשק שדות סטטיים וסופרים (שניהם בודד), יריד כאלו.

Abstract Classes

מחלקה שמכילה מתודות שאינן ממומשות. ילד בתיק abstract class... מפני כל מתודה לא ממומשת בתיק abstract

כל מחלקה שאינה אבסטרקטית יורשת מחלקה אבסטרקטית מתחייבת לממש את כל תחתיות שאינן מיושם באב

לא ניתן ליצור מופעים של abstract class. (קבל שגיאה קומפילציה אם ננסה לעשות בן.

super - קריאה לבטוי של האב. תיוב להעשיות כשורה היא ושוה של הבטוי של הבן.

@Override - מציין לפני פריסת מתודת האב במחלקת אחר מילדי.

Class Object

המחלקה הפי בסיסית כל מחלקה יורשת אותה. יש לה מתודות toString
! equals (המשווה ע"ש רפרנס)

instance of - בודקת בזמן ריצה האם אובייקט הוא מטיפוס מסוים
Song instance of Student
false

Static & Dynamic Binding

על זכר יש תמיד טיפוס קבוע.

ע"פ ה"ה יש שני טיפוסים:

- טיפוס סטטי: הטיפוס בהתאמה המשתנה. טיפוס קבוע.

- טיפוס דינמי: טיפוס העצם המוצג.

הטיפוס הפינאלי תיב ע"י לזכר של הטיפוס הסטטי.

Overloading

שתי מתודות בעלי שם זהה אך בעלות ארגומנטים שונים (מספר סדר שונה)

הקומפילר ינסה ע"ה"ש את המתודות שבהיות באופן הכי קרוב

על טיפוס ה"אמיתי"

Delegation

הרעיון הוא שבמקום שמחלקה תקבל את כל הפעולות בעצמה, היא מחזיקה
שדה שהוא class אחרת והיא מבצעת את הפעולות

Generic Programming

סגנון פיתוח תוכנה בו אלגוריתמים נכתבים באמצעות טיפוסים שיושגו
בהמשך.