סיכום שיעור תכנות מונחה עצמים

1. **Immutable**:

כל מחלקה שיש בה אובייקט שהמשתנים שלו לא משתנים, נקרא **Immutable** (לדוגמא: כל אובייקט מסוג String נחשב כ- **Immutable**).

1. **Extend**

כאשר נרצה לקבל ("ירושה") ממחלקה אחרת, נעשה **Extend** למחלקה שממנה נרצה ליירש את המשתנים/מאפיינים/פונקציות/בנאים.

כאשר עושים Extends כל מאפייני המחלקה שאותה אנחנו יורשים עובר אלינו למרות שלא הצהרתי על המשתנים או יצרתי method.

כל "רכיב" שהוא public במחלקה שממנה אנחנו יורשים עובר למחלקה שיורשת.

1. **Super :**

במידה ויש לנו 2 method **עם אותו השם,** אחת במחלקה שממנה "יורשים" ואחת במחלקה שיורשת, ואנחנו רוצים לקרא ל- method שנמצאת במחלקת שממנה "יורשים" אנחנו נקרא ל-method ע"י **super** לדוגמא : super.ToString.

אם לא נוסיף super לקריאה לפונקציה, באופן דיפולטיביjava יקרא ל- method שנמצא

במחלקה הנוכחית ובעצם יקרא לעצמו ונקבל שגיאה stakoverflow (חריגה מהזיכרון שהקצתה ה- os ל(java-

1. **Final method:**

* כאשר מגדירים final על method שנמצאת במחלקת "אב" אפשר לקרא לmethod- הזאת.

אבל אי אפשר לשכתב אותה.

* כאשר מגדירים class כ-final מחלקה זו אינה ניתנת לירושה.

1. **זמן קומפילציה/זמן ריצה:**

כאשר יוצרים אובייקט חדש, בזמן קומפילציה java מחשיב את האובייקט לפי סוג האובייקט שהצהרנו עליו ולכן רק method שנמצאות במחלקה שממנה יצרנו את האובייקט יהיה ביכולתנו לממש, אלא אם כן ה-method נמצאת גם במחלקה שיורשת ולכן בזמן ריצה ה-method שתרוץ היא ה-method שנמצאת במחלקה שיורשת.

דוגמא: person one = new employee יוצר לנו אובייקט עם יכולות להריץ פונקציות שנמצאות רק ב- person למרות שהגדרנו שאובייקט זה לקוח מ- employee, אם זאת אם ב-employee נמצאת פונקציה

שקיימת גם בperson, בזמן ריצה java (jvm) יריץ את ה-method שנמצאת ב- employee.