יחידה 7- ירושה – כללי דריסה

* לא ניתן לבצע דריסה לשיטה שהוגדרה כfinal.
* על הגישה הדורסת להיות זהה או "ליברלית" יותר מהגישה הנדרסת.

זאת אומרת, אם מאפיין הגישה בשיטה הנדרסת (מחלקת האב) הוא public, מאפיין הגישה בשיטה הדורסת לא יכול להיות private

* אם יש ערך מוחזר בשיטה הדורסת הוא חייב להיות מאותו הטיפוס המוחזר במחלקה הנדרסת (מחלקת האב) או תת טיפוס שלו.
* הארגומנטים המתקבלים צריכים להיות בדיוק מאותו הסוג של השיטה הנדרסת

ניתן להגדיר מחלקה שלמה כfinal – ואז לא ניתן לרשת אותה או לבצע בה שינויים.

10.12 – שיעור 8 עם שי תבור

הבנאי של הבן חייב לפנות קודם לפנות לבנאי של האב. רק אחרי שהבנאי של האב יתבצע, יתבצע גם הבנאי של הבן.

אם לא מציין אחרת, הבנאי של הבן יפנה לבנאי הריק של האב.

מחלקות מופשטות

Public abstract class Person

מחלקה שאי אפשר ליצור ממנה אובייקטים.

אפשר להגדיר משתנים מסוג מחלקה מופשטת, ואפשר לרשת ממנה:

Person p;

כל מחלקה שיורשת ממחלקה מופשטת, מחויבת לממש (לדרוס) את כל השיטות המופשטות המוגדרות. **אלא אם כן**, מחלקה מופשטת יורשת ממחלקה מופשטת- ואז היא **יכולה**, אבל **לא חייבת**. הבן הראשון בתור שאינו מופשט, יצטרך לממש את כל השיטות של האבות המופשטים שלו שלא מומשו.

במחלקה מופשטת השיטות לא חייבות להיות מופשטות. אך אם יש במחלקה לפחות שיטה אחת מופשטת- על המחלקה להיות מופשטת.

למחלקה מופשטת יש בנאי – אך הוא לא מופעל ע"י new אלא על ידי super (קריאה מהבנאי של המחלקה היורשת). זאת אומרת, המימוש של מחלקה מופשטת תבוא לידי ביטוי רק ע"י מחלקה שירשה אותה.

פולימורפיזם – אובייקט מוגדר מסוג אחד, אבל בפועל מתנהג כאובייקט מסוג אחר.

אפשר להסתכל על אובייקט אחד בכמה נקודות השקפה.

אובייקט מסוג ACADEMIC הוא גם PERSON .

לא משנה מאיפה נכנסת לאובייקט – הפעלת שיטות תתבצע לפי "כח המשיכה" (קשירה דינמית).

אף על פי שמסתכלים על אובייקט דרך האבא שלו, ניתן להפעיל שיטות של הבן (שלו בעצמו) רק **אם** לאבא יש שיטה בשם זהה.

במידה ולאב אין שיטה בשם זהה אך בכל זאת נרצה להפעיל את השיטה מה נעשה?

(Casting)

**שאנחנו מפעילים new אוטומטי נוצר אובייקט חדש וערכיו, אם ועד שיעודכנו, יקבלו ערכי ברירת מחדל.**

**Instanceof operator:**

Object name instanceof Object name2

שיטה שמקבלת 2 אובייקטים ומחזירה ערך בוליאני של אמת או שקר אם האובייקט הראשון (משמאל)הוא סוג של האובייקט השני.

**המחלקה Object**

* בג'אווה כל מחלקה יכולה לרשת ממחלקה אחת בלבד.
* לכל מחלקה בשפה חייב להיות "מחלקת על" ממנה היא ירושת.
* מחלקה שאיננה מציינת באופן מפורש מאיזו מחלקה היא יורשת, יורשת באופן דיפולטיבי מהמחלה Object.

המשמעות לכך שהמחלקה Object נמצאת בראש עץ הירושה של כל המחלקות:

* כל מחלקה בג'אווה למעט Object, יורשת את אותן שיטות מהמחלקה Object.
* משתנה מטיפוס Object יכול להצביע על כל טיפוס אחר המוגדר בשפה.
* שיטה המקבלת כפרטמר משתנה מטיפוס Object יכולה למעשה לקבל כל סוג טיפוס.

תכונות במחלקה:

לכל האובייקטים יש כתובת, לכן תכונת הכתובת נמצאת במחלקה.

שיטות (קיימות שיטות נוספות):

Public Boolean equals(Object obj) – בודקת אם שני אובייקטים זהים ומחזירה נכון או לא. בעצם המימוש של השיטה נוגע **לכתובות** של 2 האובייקטים, בדיוק כמו האופרטור ==. שיטה זו בדרך כלל נועדה להידרס על מנת לבצע השוואות של ערכים ולא של כתובות.

Public String toString()- מחזירה את הכתובת של האובייקט.

**שאלה:** אם אני רוצה להשתמש במשתנים של אובייקט שמיוצג על ידי מצביע כללי יותר – אני צריך לוודא שאותו אובייקט הוא מהסוג המבוקש, ובמידה וכן לעשות לו down casting? או שאפשר בלי המרה?

**תשובה:** לא. עלינו לעשות לו down casting לאובייקט המבוקש עצמו. להציב אותו במשתנה (מבציע) מסוג האובייקט המבוקש. קומפיילר לא יודע שהif וידא את זה, שאותו אובייקט הוא מהסוג המבוקש. לכן מבחינת הקופיילר, גם אם לא נשתמש בif ונעשה downcasting לא נכון- הקופיילר לא יעצור אותנו, אך תהיה שגיאת זמן ריצה.

Casting הדבר היחיד שהוא עושה זה להחליף נקודת השקפה. מסתכל על **אותו** אובייקט דרך הקאסטינג שביצענו.

**המרה לא חוקית מתגלה בזמן ריצה**!

Downcasting תמיד צריך להיות מפורש. הקומפילר לעולם לא יעשה down casting באופן אוטומטי.

במקרה של שיטה שמקבלת אובייקט מסוג של אבא, אך מופעלת ע"י אובייקט של הבן- השיטה תפעל עליו, מכיוון שכל בן הוא גם אבא.