**הורשה inheritance**

**האספקט המסתורי ביותר של OOP.**

**המלצה:**

**על מנת להבין הורשה תמיד תשאלו "בשביל מה זה טוב?"**

**המוטיבציה הראשונית של הורשה היא: "איך להימנע משימוש חוזר בקוד ע"י שכפול הקוד?"**

**נניח יש לך תוכנה למכונית פרטית ואז יש לך צורך לממש תוכנה למשאית.**

**הבעיות לא נפתרות רק בגלל שהשתמשת בהורשה.**

* **בהורשה יש מחלקות אב ומחלקות נגזרות.**
* **לפעמים מחלקת האב היא מחלקה מלאה בזכות עצמה ולפעמים לא.**
* **מחלקת אב לא מלאה נקראת מחלקה אבסטרקטית.**
* **מחלקה אבסטרקטית קיימת (בדרך כלל) על מנת לצבור קוד משותף לגורמים ספציפיים דומים.**
* **מתי מחלקת האב היא מחלקה מלאה? בדרך כאשר מדובר במקרה פרטי של גורם העומד בפני עצמו.**

**לדוגמא, מטריצה ומטריצה סימטרית.**

**מנקודת ראות טכנית יותר:**

* **הורשה יכולה להיות שרשרשת של הורשות**
* **ב-Java נתמך רק הורשה יחידה: מחלקה יכולה לרשת רק ממחלקה אחת. (הורשה מרובה קיימת בשפות אחרות כמו C++).**
* **לעיתים קרובות חלק מהקוד של מחלקת האב "לא מתאימה" למחלקה הנגזרת, והיתר כן. ניתן לדרוס רק את החלק ה"לא מתאים".**
* **ב-Java דיאגרמת ההורשה היא תמיד עץ.**
* **במחלקה נגזרת יש את כל המרכיבים שיש במחלקת האב (שדות ומתודות) אבל בהחלט יתכנו: שדות ומתודות חדשות + דריסה של מתודות (בלבד) של מחלקת האב.**
* **שדות ומתודות ה-Public של האב הם public גם של הבן.**
* **שדות ומתודות ה-private של אב קיימות בבן אבל אינן נגישות ישירות בבן (מחלקת הבן הוא כאילו קוד לקוח של האב).**
* **קיימת רמת נגישות המבדילה בין קוד מימוש מחלקות בן (וצאצאים) של האב לבין קוד לקוח אחר: רמת נגישות protected. עושים את רק כשיש סיבה לכך (דריסה במימוש פנימי למשל).**
* **שבירת התנגשויות בשמות אפשר לממש ע"י מילת המפתח super בצורה דומה ל-this (גם הודות להורשה יחידה).**

**לדוגמא:**

**class A**

**{**

 **void m()**

 **{**

 **}**

**}**

**class B extends A**

**{**

 **int x;**

**void m()**

 **{**

 **X =0;**

 **super.m();**

 **}**

**}**

* **בשפה הזו כל מחלקה באופן משתמע היא מחלקה נגזרת של מחלקה מובנית בשם Object. (זה מאפשר "משתנה המסוגל להצביע לכל דבר" כמו void \* של C).**
* **משתנה של מחלקה מסוימת יכול בפועל להצביע על מופע של כל מחלקת צאצא שלו.**

**למשל:**

**class A**

**{**

**}**

**class B extends A**

**{**

**}**

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

**a.m();**

* **הסיבה שמשתנה של האב יכול להצביע על מופע של הבן הוא שקוד (לקוח) של האב יכול להתייחס רק לשדות שקיימות גם בבן.**
* **קוד של האב יכול למנוע דריסה של מתודה בבן ע"י הגדרתה private או final.**
* **בבנאי של המחלקה הנגזרת בדרך כלל נקרא הבנאי של מחלקת האב. זה נעשה ע"י קריאה ל-super(…) עם סוגריים וחייב להיות הפקודה הראשונה בבנאי הבן. אם לא נכתוב כלום יקרא בנאי האב ללא פרמטרים אם ישנו כזה (אחרת הודעת שגיאה).**