**Delegates**

**מעין תחליף למושג פוינטרים לפונקציה**

**ההכרזה**

public **delegate double func(double x);**

**func הוא מעין typedf של פוינטר לפונקציה שמקבלת פרמטר ממשי ומחזירה ממשי.**

**ב-C# שם של פונקציה ללא הסוגריים () זה ה"כתובת" של הפונקציה.**

**קוד נוסח**

 **func f = new func(Functions.poly);**

**גורם ל-f להצביע על הפונקציה poly .**

**שימוש בזה**

**y =f (x);**

**תתבצע הסתעפות למתודה ש-f מצביע באותו רגע.**

**הגדרת מחלקה יורשת:**

**Class שם מחלקה חדשה: שם מחלקת הבסיס**

**{**

 **פרטי השדות נוספים / דורסים של המחלקה החדשה**

**}**

**ב-C# בניגוד ל-Java אבל בדומה ל-C++, יש שני סוגים של דריסת מתודות בהורשה:**

* **דריסה סטטית**
* **דריסה וירטואלית**

**ב-Java קיימת רק הדריסה הוירטואלית.**

**ההבדל בין שני סוגי הדריסה הוא כאשר יש לנו משתנה מסוג מחלקת האב המצביע בפועל על מופע של הבן.**

**בדוגמא שלנו זה משהו כמו**

**point p = new point2(1.0, 2.0);**

**מה קורה אם משתנה מסוג האב, קורא למתודה הקיימת באב ונדרסת בבן?**

**בדוגמא זה משהו כמו**

**p.show();**

**לכאורה, יש לנו 2 אפשרויות:**

* **לקרוא למתודה של האב (לפי הגדרת המשתנה)**
* **לקרוא למתודה של הבן (לפי מה שמוצבע בפועל).**

**ב-Java אין חוכמות, זה הולך לפי המופע המוצבע (כלומר לפי הבן).**

**ב-C#, C++ זה תלוי בקוד המקור, לפי סוג הדריסה.**

**דריסה סטטית:**

**המתודה באב ללא מאפין מיוחד,**

**המתודה בבן מוגדרת new**

**דריסה וירטואלית:**

**המתודה באב מוגדרת virtual**

**המתודה בבן מוגדרת override**

**השורה התחתונה:**

**כאשר ממשמים במחלקה מתודה שיש סיכוי כלשהוא שיידרס במחלקה נגזרת, יש להגדיר אותה virtual.**

**לי לא ידוע סיבה לממש דריסה סטטית (מעבר ליעילות).**

**רשומות struct ב-C#**

**מעין מחלקות שהמופע שלהן מוקצה על המחסנית.**

**ב-Java משתנים פרימיטיביים (תווים, שלמים, ממשיים) מוקצים על המחסנית וכל מופע אחר - על הערימה (הקצאה דינמית).**

**ב-C# הקצאת הזיכרון היא כמו ב-Java למעט חריג אחד: struct. מעבר לשיטת ההקצאה משתני struct הם כמו משתני class אבל עם מגבלות מסוימות.**

**ניתן לממש בנאי ל-struct ולקרוא לו בהקצאה דינמית אבל לא ניתן לממש בנאי ללא פרמטרים ל-struct.**

**אפשר להגדיר מתודות המקבלות פרמטר מסוג struct אבל אז הפרמטר מועבר by value כלומר יש שכפול של ה-struct ולא השמת פוינטר כמו ב-class.**

**חריגות**

**אולי ההבדל העיקרי בין Java ל-C# בנושא החריגות הוא שב-Java מתודה יכולה להכריח את הקורא לה לעשות זאת בבלוק try וב-c# אין מנגנון כזה.**

**בשפת C# (כמו ב-Java) ניתן להתגונן מפני חריגת מערכת ע"י try-catch. פירוש הדבר שבהתרחש תקלה כזו, השליטה תעבור לנקודה בתוכנית הנקבעת ע"י המתכנת.**

**סדרות בלוקים של catch חייבים להיות בסדר עולה של היררכיה של גזירה (מחלקות בן צריכים להקדים מחלקות אב).**

**חריגות משתמש הוא אמצעי נוח להתמודד עם בדיקות תקינות.**