מחלקות בתוך מחלקות - inner classes
חלק שלישי

תחום הגישה של מחלקה המוגדרת בתוך פונקציה

ל class המוגדר בתוך פונקציה אסור לגשת למשתנים הלוקאליים שלה אלה אם הם קבועים. כמו class פנימי רגיל שאינו סטטי, יש לו מצביע לאובייקט היוצר, והוא יכל להתייחס לשדות שלו.
לדוגמה:

```
class Check {
   int _a=7;
   void foo() {
         final int a=8;
         int a1=9;
         class A{
            void f() {
                  System.out.println(_a);
                  System.out.println(a);
                  // System.out.println(a1); //c.error
            }
         }
   }
}
```

כאשר הפונקציה היא סטטית, ה - class הפנימי גם הוא סטטי וככזה אין לא הכרות בזמן ריצה עם האובייקט היוצר:

```
class Check {
   int _a=7;
   static void foo() {
         final int a=8;
         int a1=9;
         class A{
            void f() {
                  //System.out.println(_a);  //c.error
                  System.out.println(a);
                  // System.out.println(a1); //c.error
            }
         }
   }
}
```

מחלקה אנונימית

ישנם מקרים רבים בהם יש צורך להגדיר בתוך פונקציה מחלקה חדשה המממשת ממשק מסוים, וליצור אובייקט יחיד מסוגה. ניתן לעשות כל זאת בעזרת התחביר שכבר ראינו, אך מכיוון שבעצם אין צורך, במקרה כזה, לתת שם למחלקה החדשה, קיים ב - Java תחביר מקוצר.
התחביר המקוצר מאפשר יצירת אובייקט מטיפוס ללא שם.
הקוד הבא הוא דרך מקוצרת לכתוב את אחת הדוגמאות שכבר ראינו:

```
interface HasFoo {
   public void foo();
}

 class A {
  static HasFoo getAnom() {
         HasFoo p =
            new HasFoo() {
                  public void foo() {
                     System.out.println("foo of an anonymous class");
                  }
            };
            return p;
   }
}
```

התחביר ליצירת אובייקט אנונימי, עלול להיראות מוזר. הוא מתחיל כביכול ביצירת אובייקט מסוג HasFoo, פעולה שהיא כמובן לא אפשרית מכיוון ש - HasFoo הוא רק ממשק. בהמשך מופיע קוד המגדיר מחלקה חדשה הממשת את HasFoo. למחלקה החדשה אין שם, אך האובייקט היחיד שנוצר ומוצבע ע"י p, הוא מסוגה.

ניתן כמובן לקצר ולכתוב:

```
static HasFoo getAnom() {
         return
            new HasFoo() {
                  public void foo() {
                     System.out.println("foo of an anonymous class");
                  }
            };
}
```

אותו תחביר מקוצר של יצירת אובייקט מסוג מחלקה אנונימית הממשת ממשק, קיים גם עבור יצירת אובייקט ממחלקה אנונימית היורשת ממחלקה קיימת:

```
class B{
   public void foo() {System.out.println("B.foo()");}
}

class A{
  static B getAnom() {
         return
            new B() {
                  public void foo() {
                     System.out.println("Anonymous.foo()");
                  }
            };
   }
}
```

שימוש במחלקות אנונימיות נפוץ מאד בקוד של ממשקים גרפיים. בקוד כזה, יש הרבה פעמים צורך ליצור אובייקט יחיד מטיפוס המוגדר במיוחד בשבילו.

קבצי ה - class. הנוצרים מקומפילציה של מחלקות פנימיות

כידוע, הקומפיילר של Java מקבל קבצי מקור ב - Java, ומייצר קבצים בעלי סיומת ".class" המכילים את התרגום ב - byte code. Byte-code היא השפה המובנת לאלמנט הנקרא Java Virtual Machine או JVM, שיודע לקרוא byte code ולהריץ אותו. עבור כל מחלקה או ממשק, מייצר הקומפיילר קובץ ".class" נפרד. גם אם המחלקות או הממשקים הוגדרו באותו קובץ של קוד מקור, הקומפיילר ייצר עבורם קבצי byte-code נפרדים. מחלקה או ממשק ששמו A לדוגמה, יתורגם לקובץ הנקרא A.class. אם ל- A מחלקה פנימית בשם B היא תתורגם לקובץ בשם A$B.class. ההפרדה ע"י סימן ה - $ היא חלק מהגדרות השפה. אם הוגדרה מחלקה אנונימית באחת הפונקציות של A, היא תתורגם לקובץ A$1.class. אם ב - A מוגדרות מספר מחלקות אנונימיות הן תתורגמנה לקבצים עם שמות דומים וההבדל ביניהם יהיה רק במספר הסידורי.