**Típuskövetkeztetés generikusok példányosításához**

A típus argumentumait a generikus osztályra való hivatkozáshoz kicserélhetjük üres paramétertípusra (<>), amíg a fordító következtetni tud a kontextusból. Ezt a zárójelpárt informálisan gyémántnak nevezzük.

Például vegyük a következő változó deklarációt:

Map<String, List<String>> myMap = new HashMap<String, List<String>>();

JAVA SE 7-ben, helyettesíthetjük a paraméterezett konstruktor típust üres paramétertípussal (<>):

Map<String, List<String>> myMap = new HashMap<>();

Jegyezzük meg, hogy ha ki akarjuk használni az automatikus típuskövetkeztetés előnyeit, akkor meg kell adnunk a gyémántot.

A következő példában a fordító generál egy ellenőrizetlen konverziós figyelmeztetést, mert a HashMap() konstruktor a HashMap primitív típusra hivatkozik, nem a Map<String, List<String>> típusra:

Map<String, List<String>> myMap = new HashMap(); // ellenőrizetlen konverziós hiba

A Java SE 7 korlátozott típuskövetkeztetést támogat generikus példányosításhoz. Csak akkor használhatunk típuskövetkeztetést, ha a paraméterezett konstruktor típusa egyértelműen kiderül a kontextusból. Például a következő nem fordul le:

List<String> list = new ArrayList<>();

list.add("A");

// The following statement should fail since addAll expects

// Collection<? extends String>

list.addAll(new ArrayList<>());

Ne feledjük, hogy a gyémánt gyakran működik metódus hívásoknál is, ugyanakkor javasolt elsősorban változó deklarációnál használni.

Összehasonlításképpen a következő példa lefordul:

// The following statements compile:

List<? extends String> list2 = new ArrayList<>();

list.addAll(list2);

**Típuskövetkeztetés és generikus konstruktorok generikus és nem generikus osztályoknál.**

Megjegyzés: A konstruktorok lehetnek generikusak (más szóval, deklaráljunk saját formális típusú paramétereket) generikus és nem generikus osztályokban is. Nézzük a következő példát:

class MyClass<X> {

<T> MyClass(T t) {

// ...

}

}

Nézzük a MyClass példányosítását, ami a Java SE 7-ben és a korábbi verziókban valid.

new MyClass<Integer>("")

Ez az állítás létrehoz egy példányt a paraméterezett típusú MyClass<Integer>-ből. Az állítás kifejezetten meghatározza az Integer típust formális paramétertípusnak, X a generikus Myclass<X> osztálynak. Megjegyzendő, hogy a generikus osztály konstruktora formális paramétert tartalmaz, T. A fordító String típusra következtet a generikus osztály konstruktorából (mert az aktuális paramétere a konstruktornak egy String objektum).

A Java SE 7 előtti kiadású fordítók képesek következtetni a generikus konstruktor aktuális paraméter típusára, hasonlóan a generikus metódusokhoz. Ugyanakkor, a fordító Java SE 7-ben következtetni tud az aktuális paraméterre a generikus osztály példányosításánál, ha használjuk a gyémántot(<>). Vegyük a következő példát, ami valid Java SE 7-ben és a későbbi verziókban:

MyClass<Integer> myObject = new MyClass<>("");

Ebben a példában a fordító Integer típusra következtet az X-ből a generikus MyClass<X> osztályban. String formális paramétertípusra következtet a T konstruktorból.