**Genericele**

Pelle Remus-Nicolae

SDI Anul 1 Semestrul 2

**Tipurile generice** adaugă stabilitate codului, făcând mai multe dintre erorile acestuia detectabile în timpul compilării.

Pe scurt, genericele permit tipurilor (clase și interfețe) să fie parametri atunci când definesc clase, interfețe și metode. La fel ca parametrii formali mai cunoscuți utilizați în declarațiile de metodă, parametrii de tip vă oferă o modalitate de a reutiliza același cod cu intrări diferite. Diferența este că intrările la parametrii formali sunt valori, în timp ce intrările la parametrii de tip sunt tipuri.

Un tip generic este o clasă sau o interfață generică care este parametrizată pe tipuri. Prin convenție, numele parametrilor de tip sunt o singură literă mare. Acest lucru contrastează puternic cu convențiile de denumire a variabilelor pe care le cunoașteți deja și cu un motiv întemeiat: Fără această convenție, ar fi dificil să se facă diferența dintre o variabilă de tip și o clasă obișnuită sau un nume de interfață.

Cele mai utilizate nume de parametri de tip sunt:

* E – Element (utilizat des de Java Collections Framework)
* K – Cheie
* N – Număr
* T – Tip
* V – Valoare
* S, U, V etc – al 2-lea, al 3-lea, al 4-lea tip etc

**Invocarea și instanțierea unui tip generic:** Pentru a face referire la clasa generică Box din cod, trebuie să efectuați o invocare de tip generic, care înlocuiește T cu o anumită valoare concretă, cum ar fi Integer:

|  |
| --- |
| Box<Integer> integerBox = new Box<Integer>(); |

Vă puteți gândi la o invocare de tip generic ca fiind similară cu o invocare obișnuită a unei metode, dar în loc să transmiteți un argument unei metode, transmiteți un tip ca și argument - Integer în acest caz - clasei Box în sine.

**Diamantul:** În Java SE 7 și versiunile ulterioare, puteți înlocui argumentele de tip necesare pentru a invoca constructorul unei clase generice cu un set gol de argumente de tip (<>) atâta timp cât compilatorul poate determina sau deduce argumentele de tip din context . Această pereche de paranteze unghiulare, <>, se numește informal diamant. De exemplu, puteți crea o instanță a casetei <Integer> cu următoarea instrucțiune:

|  |
| --- |
| Box<Integer> integerBox = new Box<>(); |

**Metodele generice** sunt metode care introduc propriii parametri de tip. Acest lucru este similar cu declararea unui tip generic, dar domeniul de aplicare al parametrului de tip este limitat la metoda în care este declarat. Sunt permise metode generice statice și nonstatice, precum și constructorii de clase generice.

Sintaxa pentru o metodă generică include o listă de parametri de tip, în paranteze unghiulare, care apare înaintea tipului de returnare a metodei. Pentru metodele generice statice, secțiunea parametrului tip trebuie să apară înainte de tipul de returnare al metodei.

**Parametrii de tip delimitați:** Pot exista momente când doriți să restricționați tipurile care pot fi utilizate ca argumente de tip într-un tip parametrizat. De exemplu, o metodă care operează pe numere ar putea dori doar să accepte instanțe de Număr sau subclasele acestuia. Pentru aceasta sunt folosiți parametrii de tip delimitați.

Pentru a declara un parametru de tip delimitat, enumerați numele parametrului de tip, urmat de cuvântul cheie extends, urmat de limita superioară a acestuia, care în acest exemplu este Number. Rețineți că, în acest context, extends este utilizat în sens general pentru a însemna fie „extends” (ca în clase), fie „implement” (ca în interfețe).

|  |
| --- |
| public <U extends Number> void inspect(U u) |

**Delimitări multiple:** <T extends B1 & B2 & B3>

**Restricții la generice:**

* Nu se pot instanția tipurile generice cu tipurile primitive
* Nu se pot crea instanțe de parametri de tip
* Nu se pot declara câmpuri statice ale căror tipuri sunt parametri de tip
* Nu se pot utiliza distribuții sau instanțe cu tipuri parametrizate
* Nu se pot crea tablouri de tipuri parametrizate
* Nu se pot crea, prinde sau arunca obiecte de tipuri parametrizate
* Nu se poate supraîncărca o metodă în care tipurile de parametri formali ai fiecărei suprasarcini se șterg la același tip brut