

5.

Operátorok

Az operátorok egy, kettő vagy három operanduson hajtanak végre műveletet. Az egyoperandusú operátorokat unáris operátoroknak hívjuk. Például a ++ operátor az operandusát 1-el növeli. A kétoperandusú operátorokat bináris operátoroknak hívjuk. Például az = operátor a jobb oldali operandus értékét a baloldali operandusba másolja. Végül a háromoperandusú operátor három operandust vár. Javában egy háromoperandusú operátor van, a ?: feltételes operátor.

Az unáris operátorok lehetővé teszik a prefix és postfix jelölést is. Az összes bináris operátor infix jelölést alkalmaz, vagyis az operátor az operandusok között szerepel. A háromoperandusú operátor szintén infix jelölést tesz lehetővé. Az operátor mindkét komponense az operandusok között szerepel. A művelet végrehajtása után a kifejezés értéke rendelkezésre áll. Az érték függ az operátortól és az operandusok típusától is. Aritmetikai operátorok esetén a típus alá van rendelve az operandusoknak: ha két int értéket adunk össze, az érték is int lesz. Javában a kifejezések kiértékelési sorrendje mindig balról jobbra történik. Aritmetikai operátorok

A Java programozási nyelvben sokféle aritmetikai operátor áll rendelkezésre lebegőpontos és egész számokhoz. Ezek az operátorok a +, -, *, /, % (maradékképzés).

Implicit konverzió

Amikor egy aritmetikai operátor egyik operandusa egész, a másik pedig lebegőpontos, akkor az eredmény is lebegőpontos lesz. Az egész érték implicit módon lebegőpontos számmá konvertálódik, mielőtt a művelet végrehajtna. A ++ operátor növeli az operandus értékét, a – pedig csökkenti eggyel. Mindkettőt írhatjuk az operandus elé (prefix) és után (postfix) is. A prefix forma esetén először történik az érték növelése vagy csökkentése, majd a kifejezés értéke is a megváltozott érték lesz. Postfix használat esetén a végrehajtás pont fordítva történik.

Relációs operátorok

A relációs operátorok összehasonlítanak két értéket, és meghatározzák a köztük lévő kapcsolatot. Például a != true-t ad, ha a két operandus nem egyenlő. Operátorok: >, >=, <, <=, ==, !=.

Logikai operátorok

A relációs operátorokat gyakran használják logikai operátorokkal együtt, így összetettebb logikai kifejezéseket hozhatunk létre. A Java programozási nyelv hatféle logikai operátort támogat: && (logikai és), || (logikai vagy), ! (logikai nem), & (bitenkénti és), | (bitenkénti vagy), ^ (bitenkénti nem).

Rövidzár kiértékelés

Bizonyos esetekben a logikai operátor második operandusa nem értékelődik ki. Például a következő esetben: | (numChars < LIMIT) && (...). Az && operátor csak akkor ad vissza true-t, ha mindkét operandus true. Tehát ha numChars nagyobb vagy egyenlő, mint LIMIT, akkor && bal oldala false, és a kifejezés visszaadott eredménye a jobb oldali operandus kiértékelése nélkül születik meg. Ilyen esetekben a fordító nem értékeli ki a jobb oldali operandust. Ilyenkor a jobb oldali kifejezés miatt közvetett mellékhatások léphetnek fel, például ha adatfolyamból olvasunk, értékeket aktualizálunk, vagy számításokat végzünk a jobb oldali operandusban. Ehhez hasonlóan, ha a || operátor bal oldali operandusa igaz, felesleges a jobboldalt kiértékelni, nem is fog megtörténni. Ha mindkét operandus logikai, az & operátor hasonlóan viselkedik, mint az &&. Azonban & mindig kiértékelődik és true-t ad vissza, ha mindkét operandusa true. Ha az operandusok boolean-típusúak, | azonos műveletet végez, mint ||. Ezért nem érdemes olyan kódot készíteni, amelyik a jobboldali operandusának kiértékelése során a kiértékelésen túl mást is tesz.

Bitléptető és bitenkénti logikai operátorok

A léptető operátorok bit műveleteket végeznek, a kifejezés első operandusának bitjeit jobbra, vagy balra léptetik. A Java nyelvben használt operátorok: <<, >>, >>>. Mindegyik operátor az első operandus bitjeit lépteti az operátor által megadott irányba a második operandus értékével. Bitenkénti logikai operátorok: & (bitenkénti és, ha mindkét operandus szám; feltételes és, ha mindkét operandus logikai), | (bitenkénti vagy, ha mindkét operandus szám; feltételes vagy, ha mindkét operandus logikai), ^ (bitenkénti kizáró vagy, XOR), ~ (bitenkénti negáció).

Kifejezések

A kifejezések hajtják végre a program feladatát. A kifejezések többek között változók kiszámítására, értékük beállítására és a program futásának ellenőrzésére használjuk. A kifejezéseknek kétféle feladata van: végrehajtani a számításokat, amelyeket a kifejezés alkotóelemei határoznak meg, és visszaadni a számítás végeredményét. Definíció: A kifejezés változók, operátorok és metódushívások olyan sorozata (a nyelv szintaxisát figyelembe véve), amely egy értéket ad vissza. A Java nyelv lehetőséget biztosít összetett kifejezések és utasítások létrehozására kisebb kifejezésekből, amíg a kifejezés egyik részében használt adattípusok megegyeznek a többi rész adattípusaival. Példa: $| x * y * z$. Ebben a példában a sorrend, amely szerint a kifejezés kiértékelődik, nem fontos, mivel a szorzás eredménye nem függ a tagok sorrendjétől, a végeredmény mindig ugyanaz. Azonban ez nem minden kifejezésre igaz. Például a következő kifejezés különböző eredményt ad attól függően, hogy az összeadás vagy az osztást megvalósító operátor hajtódik-e végre elsőként: $| x + y / 100$. Zárójelezéssel meghatározhatjuk, hogy egy kifejezés hogyan értékelődjön ki. Például így tehetjük egyértelművé a végrehajtást az előző példában: $| (x + y) / 100$. Ha határozottan nem jelezzük a sorrendet, amely szerint akarjuk, hogy az összetett kifejezés kiértékelődjön, akkor a sorrendet az operátorok precedenciája fogja meghatározni. A magasabb precedenciával rendelkező operátorok hajtódnak végre elsőként. Operátor precedencia szintek a legmagasabbal kezdve: postfix, unáris, multiplikatív, additív, léptetés, reláció, egyenlőség, bitenkénti és, bitenkénti kizáró vagy, bitenkénti vagy, logikai és, logikai vagy, feltételes, értékadás.

Utasítások

Az utasítások nagyjából a beszélt nyelv mondatainak felelnek meg. Az utasítás egy konkrét futási egységet hoz létre. A következő kifejezéstípusok szervezhetők utasításokba, amelyek pontosvesszővel végződnek: értékadó kifejezések; ++ és – használata; metódushívások; objektumot létrehozó kifejezések. Az utasításnak ezt a fajtáját kifejezés utasításoknak nevezzük. A kifejezés-utasításokon kívül még kétféle típust meg kell említenem: A deklarációs utasítás: létrehoz egy változót. Végrehajtás-vezérlő utasítás: szabályozza, hogy az utasítások milyen sorrendben hajtódnak végre. A for ciklus és az if utasítás jó példák a végrehajtás-vezérlő szerkezetre.

Blokk: nulla vagy több utasítás kapcsos zárójelek között, amely használható bárhol, ahol az önálló utasítások megengedettek.