PHP és Java különbözőségek

A PHP egy szerveroldali nyelv, tulajdonképpen scriptnyelv, az így megírt fájlok futtatásához a fizikai szervergépre valamilyen PHP fordító kell telepíteni és konfigurálni. Legelterjedtebb manapság még mindig az Apache webszerver, mely a UNIX rendszerekre telepíthető.

A Java nyelven megírt kódok futtatásához is saját környezet kell. A java telepíthető minden operációs rendszerre, valamint komplexitása miatt az egyik legelterjedtebb és leginkább használt nyelv.

Érdemes kiemelni a PHP hiba, -illetve figyelmeztetés kijelzését is, mely szerver, az Apache konfigurációs fáljában oldalon, illetve a PHP fájlban is letiltható, illetve engedélyezhető. Érdemes ezt mindig engedélyezni a fejlesztés során, mivel egyéb esetben egyes hibák elfedésre kerülnek a PHP rugalmassága miatt, így a végső eredmény is hibás lehet. Például az alábbi kódban:

<?php

function multiply ($value) {

$value = $value \* 10;

return $value;

}

$retval = multiply();

Print "Return value is $retval\n";

?>

Látható, hogy a multiply függvény egy paramétert vár, és azzal számol, azonban a meghívásakor nem kerül átadásra paraméter. Ilyenkor a PHP rugalmassága miatt, egyszerűen 0 értékkel fog számolni a program, azaz a végeredmény is nulla lesz.

Azonban ez nyilvánvalóan nem a helyes működés lenne (kivéve természetesen, ha pont a 0 számmal szeretnénk meghívni egyébként is a metódust), ezért érdemes bekapcsolni a hibajelzést. Ilyenkor a program kijelzi, hogy bár a metódus deklarálásakor megadtuk, hogy paramétert várunk, meghívásakor nem adtunk neki paramétert.

Általánosságban a PHP megvalósítása miatt, ha ilyen hibákat vétünk (és ha nincs hiba kijelzés) a PHP mindig megpróbálja megoldani, a programot lefuttatni. A Java ezzel szemben, ha bármilyen hibát vétünk akkor már a kód fordítása során erről értesítést kapunk, a fordítás pedig leáll. Tehát Java nyelvben a fenti kód (itt természetesen osztályt is kell definiálnunk, majd annak a metódusát meghívni):

public class Test {

private value;

Test(int value) {

this.value = value;

}

public void multiply() {

value = value \* 10;

System.out.println("Return value is: " + value);

}

public static void main(String args[]) {

Test test = new Test();

test.multiply();

}

}

Ez a kód már fordításkor hibát fog jelezni, hiszen a konstruktorban zárójelpárban megadtuk, hogy várunk egy integer értéket, viszont a main függvényben az osztály inicializálásakor nem adtunk meg értéket.

Fontos különbség, hogy bár a Java nyelv eredendőleg támogatja az objektum orientáltságot, a PHP azonban nem. Eredetileg ez szerveroldali scriptnyelv volt, csak a 2004-ben kiadott 5-ös verzióval került be az objektum orientáltság. Mivel a PHP akkoriban a legnépszerűbb szerveroldali scriptnyelv volt, sőt még manapság is az, nagyon sok olyan kód készült, melyben objektum orientált leírás nem volt, így ezeket először át kellene alakítani objektum orientált módon, és után lehetne Java kóddá formálni.

Szintaxis

A PHP szintaxisa és a Java szintaxisa néhány tekintetben hasonlóságot mutat, azonban vannak fontos különbségek, melyeket a keresztfordítás során figyelembe kell venni.

Mindkét nyelvben vannak változók, melyek az adatok, értékek tárolására szolgálnak, azonban fontos különbség, hogy míg a Java nyelv megköveteli a változók típusainak meghatározását, addig a PHP esetében nem kell ezt megadni, a változó értéke dönt a típusáról.

A Java nyelvben tehát a típusmegjelöléssel vezetjük be a változót, ezután következik a változó neve, majd = jel után adható meg az értéke. PHP nyelvben ezzel szemben nincs szükség típusmegjelölésre, azonban a változók $ jellel kerülnek bevezetésre, ezután kell a változó nevét megadni.

Tehát például egy változó deklarálása Java nyelvben:

int a = 2;

PHP nyelvben ugyanez:

$a = 2;

A keresztfordítás esetében erre különösen figyelni kell, illetve nehézséget jelenthez, ha PHP nyelvről történik a fordítás Java nyelvre, hogy a változó típusát meg kell adni. Segítséget jelent, hogy a PHP nyelvben megtalálható a gettype() metódus, mely visszaadja egy adott változó típusát.

Mindkét nyelvben megtalálhatók a konstansok, melyek felül nem írható változók, azonban megadásuk itt is más, a Java nyelvben a változó típusa előtt megadott final kulcsszóval definiáljuk a konstanst, általában, konvenció szerint a változó nevét csupa nagybetűvel írva. A PHP-ban ezzel szemben a konstans a define() függvény meghívásával adhatjuk meg, amelynek két paramétere van, az első a konstans változó neve, melyet itt is csupa nagybetűvel írunk, második az értéke.

Külön érdemes megemlíteni a változókat hatáskörük szerinti csoportosításuk alapján. Így a PHP nyelvben az alábbi csoportokat különböztethetjük meg:

* Lokális változók: Olyan változók, melyek osztályokon, illetve metódusokon belül deklarálunk, ilyen változók csak az adott osztályon, illetve metóduson belül használhatók, azon kívül nincs értékük, ott felhasználva NULL értéket, kiíratva üres értéket kapunk.
* Metódus paraméterek: A metódus neve után zárójelpárba írt változónév, csak az adott metóduson belül használható, segítségével lehet átadni változók értékét a metódusnak. Ha a metóduson belül egy ugyanolyan nevű változót deklarálunk, akkor az átadott érték elveszik, a metódus további részében az új értékkel történik a számolás. Metódus meghívásakor érdemes megadni a paramétert, ám (ha nincs hibajelzés bekapcsolva), paraméter nélkül is lefut a metódus és a program.
* Globális változók: a globális változók a program bármely részében elérhetők, tartalmuk módosítható. Ahhoz, hogy egy metóduson belüli változót globális változóként kezeljen meg kell adni a változónév előtt a GLOBAL kulcsszót, ellenkező esetben lokális változóként kerül deklarálásra, és használat előtt inicializálni kell. Figyelni kell rá, hogy egy globális változót két egymás utáni metódusban történő felhasználás esetén, a második metódus már az első által módosított értéket fogja használni.
* Statikus változók: Általánosságban metódusokban használjuk őket. A metódusok lefutása után a metódusokban használt változók megszűnnek, azonban a statikusként definiált változók megmaradnak, így a metódus újra felhasználásakor nem kerülnek újra deklarálásra, hanem a már meglévő értékkel dolgoznak. Megadásuk a változó neve elé írt STATIC kulcsszóval lehetséges.

Bár a Java nyelvben is különböző csoportra oszthatjuk a változókat hatáskörük szerint, és láthatunk átfedéseket a PHP nyelvhez kapcsolódóan, de azért van néhány különbség

* Lokális változók: Automatikusan lokális változó lesz minden olyan változó, melyet valamilyen blokkon belül deklarálunk, legyen az konstruktor, metódus, vagy bármilyen blokk. Ezek csak az adott blokkon belül érhetők el, a blokk lefutása után pedig megszűnnek. Mivel alapértelmezetten csak belül érhető el, ezért módosítót nem kell alkalmazni, illetve deklarálásukkor azonnak inicializálni is kell őket valamilyen értékkel.
* Metódus paraméterként is megadhatunk változókat, így tudunk Java nyelvben is értékeket átadni a metódusoknak, amikkel utána számolni tudunk. A paramétereket itt is a metódus neve után írt zárójelpárban adjuk meg, típusukkal együtt. Itt figyelni kell, hogy a metódus meghívásakor mindenképpen meg kell adni a paramétert is, méghozzá pontosan olyan típusút, amilyet a metódus deklarálásakor megadtunk. Ha nem adunk meg, vagy nem olyan típusút adunk meg, akkor a program már fordításkor hibaüzenetet fog adni, nem fog lefordulni.
* Példányszintű változó: Ez a változó egy osztályhoz kapcsolódik, az osztályon belül kell deklarálni, általában valamilyen módosítót is használunk a deklaráláskor, konvenció szerint private lesz az osztály adattagja, azaz csak az osztályon belül érhető el. Deklaráláskor nem kötelező azonnal inicializálni is, ezt később is megtehetjük, azonban így is azonnal kap értéket, valamilyen 0-t jelző értéket, például referenciaváltozók null, boolean változók false értéket kapnak.
* Statikus változók: Az osztály szintű változók, hasonlóak a példány szintű változókhoz, azonban deklaráláskor meg kell adni a static kulcsszót a változó neve előtt például:  
  private static int a;  
  Az ilyen változókat azért hívjuk osztály szintű változóknak, mert függetlenül attól, hogy egy osztályból hány példány létezik, az ilyen változókból csak egy létezik osztályonként. Deklaráláskor a példányszintű változóhoz hasonlóan automatikusan kap értéket is, ha nem adunk neki külön, és ugyanúgy ahogy az előbbi, ez is 0 értéket kap.  
  Ha módosítója public, akkor más osztályokból is elérhetjük az Osztálynev.Valtozonev formátumot használva.

Láthatósághoz kapcsolódó módosítók

A Java és a PHP nyelvekben is léteznek módosítók, melyekkel a változók, metódusok láthatósági körét lehet beállítani. Fontos különbség, hogy Java nyelvben ezen módosítókat fel lehet használni osztályoknál is, tehát megadható egy osztály láthatósági köre. Ezzel szemben, mivel a PHP nyelvben nincsen külön csomagokba szervezés, ezért az osztályoknál ilyen módosító nem használható, a PHP nyelvben minden osztály pulic lesz.

Az alábbi módosítókat különböztethetjük meg:

|  |  |  |
| --- | --- | --- |
| Módosító | Java | PHP |
| Nincs megadva módosító | Ha nincs megadva módosító, a Java nyelv a csomagszintű láthatóságot engedélyezi, tehát az adott elemeket az azonos csomagban lévő osztályok érhetik el. | Ha nincs megadva módosító egy változó vagy metódus előtt, akkor az alapértelmezetten public lesz, tehát bárki elérheti. |
| Public | Nyilvános módosító esetén az változókat, metódusokat mindenki elérheti. Maguk az osztályok általában public módosítóval rendelkeznek, illetve ritkán egyes változók, főként osztályszintű változók. | Ugyanúgy, mint a Java nyelvben itt is a public módosítóval ellátott tagokat bárki elérheti. Mivel az alapértelmezett módosító is public lesz, ezért, ha nyilvános adattagot szeretnénk deklarálni, akkor nem is kell kiírni a módosítót. |
| Protected | Az adott csomagban lévő osztályok és a leszármazott osztályok érhetik el az adattagokat. | Csak az adott osztályból és a leszármazott osztályokból érhető el az adattag. |
| Private | Csak az adott osztályban lévő metódusok érhetik el az adattagokat, illetve az ilyen módosítóval ellátott metódusokat. Általában ilyen módosítóval vannak ellátva az adattagok. Mivel az objektum orientált programozás egyik alapelve az egységbe zárás, tehát, hogy az adattagokat a külvilág ne lássa, csak az osztály metódusain keresztül lehessen módosítani és mivel ha nem adunk meg módosítót, akkor a Java nyelv alapértelmezetten nem private módosítót ad hozzá, ezért ezt általában ki kell írni. | Csak abból az osztályból érhető el az adattag, ahol deklarálva lett az, a külvilág számára csak esetleg az osztály metódusain keresztül módosítható, ha vannak erre létrehozott metódusai. |