****

Latvijas Universitātes Vadības un uzņēmējdarbības mācību centrs

(LU VUMC)

**JAVA PROGRAMMĒŠANAS PAMATI**

2021

ESF projekts Nr. 8.4.1.0/16/l/001  
‘’Nodarbināto personu profesionālās kompetences pilnveide”



# Nodarbība 05

## Terminu vārdnīca

Objekt-orientēta programmēšana (OOP) : programmēšanas paradigma, kura ir balstīta uz « objektu » koncepcijas. Objekti spēj saturēt sevī datus un kodu : datus lauku veidā, kurus mēdz saukt arī par atribūtiem, un kodu procedūru veidā, kuras mēdz saukt arī par metodēm. Datorzinātnē objekts var būt mainīgais, datu struktūra, funkcija, vai metode, un tādā veidā tas ir vērtība datora atmiņā ar atsauci (adresi), kura glabājas objekta identifikatorā. OOP bāzes principi ir : abstrakcija, iekapsulēšana, pārmantojamība, polimorfisms.

Objekts : tas ir Java klases loceklis, vai instance. Katram objektam ir identitāte, uzvedība un stāvoklis. Objekta stāvoklis tiek glabāts atribūtos (laukos), bet objekta metodes (procedūras un funkcijas) satur sevī kodu, kas definē objekta uzvedību.

Java klase : Klases un objekti ir pamata koncepti objekt-orientēta programmēšanā, kuru jēdziens un būtība apgrozās ap un pietuvojas reālas dzīves objektiem. Java klase, tāpat kā jebkuras OOP valodas klase, ir programmētāja definēts prototips, no kuras tiek veidoti objekti. Tas definē un atspoguļo atribūtu un metožu kopu, kas piemīt visiem šāda tipa objektiem.

Encapsulation : iekapsulēšana. Tā ir OOP koncepcija, kura paredz datu un metožu, kas manipulē ar datiem, sasaistīšanu, kas savukārt ļauj nodrošināt datu un metožu pasargāšanu no ārējās ietekmēšanas un nekorektas izmantošanas. Iekapsulēšana realizē svarīgu OOP datu paslēpšanas koncepciju.

Metode : tas ir koda bloks ar nosaukumu, kurš tiek izpildīts tikai tad kad to izsauc. Tas var saņemt datus izsaukuma brīdī caur parametriem, ja vien tādi ir definēti. Metode var atgriezt rezultātu, vai arī neatgriezt, atkarībā no metodes definīcijas. Metodes objektā definē tā uzvedību, kā arī var piekļūt un operēt ar objekta atribūtiem (laukiem). Metožu mērķis ir funkcionalitātes atkārtota izmantošana.

Method overriding: Metodes pārrakstīšana. Objekt-orientēta programmēšanā tā ir programmēšanas valodas funkcionalitāte, kas nodrošina iespēju apakšklasei (bērna klasei) definēt specifisku implementēšanu metodei, kura jau ir implementēta superklasē (vecākā klasē). Dažas programmēšanas valodas paredz iespēju definēt nepārrakstāmas klases.

Constructor method : Metode-konstruktors. Nosaukums cēlies no angļu valodas vārda construct -- būvēt, veidot. Speciāla metode, kura inicializē jaunizveidota objekta atribūtu vērtības. Ja tā netiek definēta tiešā veidā, Java izveido noklusējuma konstruktoru.

Java package : Java pakotne. Tā ir līdzīgu tipu klašu, interfeisu un apakš-pakotņu kopa. Tā sakārto Java klases nosaukumus telpās, nodrošinot unikālu nosaukumvietu katram tā veidam. Pakotnes Javā tiek iedalītas divās lielās kategorijās : iebūvētas un lietotāju definētas pakotnes. Mūsdienās pastāv daudz iebūvēto pakotņu, piemēram, java, lang, awt, javax, swing, net, io, util, sql u.tt. Lai izveidotu savu pakotni, ir jāsaprot ka Java izmanto OS direktoriju struktūru lai organizētu un glabātu pakotnes.

Java Heap memory : Java kaudzes atmiņa. Tas ir atmiņas apgabals, kurš ir paredzēts objektu uzglabāšanai, kurus izveido JVM dažādu Java programmu izpildes laikā. Šī atmiņa tiek koplietota starp visiem programmas izpildes pavedieniem (threads). Šis atmiņas apgabals tiek izveidots JVM piestartēšanas brīdī. Šīs atmiņas izmērs var būt mainīgs vai nemainīgs, un tas ir konfigurējams lielums. Daudzi konfigurē šā atmiņas apgabala izmēru 2 līdz 8 GB, lai minimizētu atkritumu savākšanas (garbage collection) pauzes.

Java Stack memory : Java rindas atmiņa. JVM sadala atmiņu divos apgabalos : Heap un Stack. Rindas (stack) atmiņas apgabals tiek izmantots statiskas atmiņas un pavediena izpildes nodrošināšanai. Tas satur primitīvas vērtības kas ir metožu specifiskas, kā arī atsauces uz objektiem, kuri tiek uzglabāti kaudzes (heap) atmiņas apgabalā; tas tiek izmantots lai glabātu metožu izpildes kārtību un lokālus mainīgos. Piekļuve šai atmiņai tiek nodrošināta LIFO kārtībā (Last-In-First-Out -- pēdējais-iekšā-pirmais-ārā).