****

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 08

## 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.

Java interface : Java interfeiss. Tas ir abstrakts tips jeb datu struktūra, kurš tiek izmantots lai definētu klases uzvedību. Tādā veidā interfeiss nodrošina specifikāciju kopu, ko citām klasēm jāimplementē. Vairāki Java interfeisi var būt implementēti ar vienu Java klasi. Pats interfeiss nevar iekļaut metožu implementāciju. Interfeisu var iedomāties kā līgumu, kas nosaka implementējošas klases uzvedības pamatnostādnes, kuras tai būs spēkā un kuras tā nevarēs ignorēt. Visas interfeisa metodes ir publiskas un abstraktas.

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.

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.

Inheritance : pārmantošana. Tas ir OOP mehānisms, kurš nodrošina viena objekta vai klases veidošanu uz cita objekta (prototipēšanas pārmantošanas veids) vai klases (klases pārmantošanas veids) bāzes, saglabājot līdzīgu implementāciju. Cita pārmantošanas definīcija ir : jaunu klašu (apakš-klašu) atvasināšana no citām eksistējošām klasēm (bāzes klasēm, vai super-klasēm), tādā veidā formējot klašu hierarhiju. Lielākā OO programmēšanas valodu daļā objekts, kurš ir izveidots caur pārmantošanu ("bērna objekts"), iegūst visas "vecāka objekta" metodes un atribūtus, ar šādiem izņēmumiem : konstruktori, destruktori, pārrakstīti operatori un bāzes klases "draugu" metodes. Pārmantošana nodrošina programmētājiem šādas iespējas : veidot jaunas klases, paplašinot citas klases; definēt jaunu implementāciju, saglabājot pamata uzvedību; atkārtoti izmantot kodu; neatkarīgi paplašināt oriģinālu programmnodrošinājumu caut publiskām klasēm un interfeisiem; organizēt kodu hierarhiski, veidojot klašu koku.

Polimorfisms : OOP koncepcija, kura paredz viena interfeisa izmantošanu dažādu tipu vienībām, vai viena nosaukuma izmantošanu dažādu tipu apzīmēšanai. Citiem vārdiem, polimorfisms ir objekta spēja pieņemt dažādas formas. Pastāv dažādi polimorfisma veidi; OO programmēšanas valodās visbiežāk sastopams polimorfisma veids ir kad vecāka klases atsauce tiek izmantota, lai norādītu uz bērna klases objektu. Piemērs : Student student = new Student(); Person person = new Student();

Abstrakcija : OOP koncepcija, kura paredz tikai būtisku datu "parādīšanu", un nebūtiskas informācijas "paslēpšanu". Tas ir abstrakcijas galvenais mērķis : nevajadzīgu detaļu paslēpšanā no lietotājiem.

Operators : tā ir konstrukcija, definēta programmēšanas valodā, kura uzvedas kā funkcija, bet atšķiras sintaktiski vai semantiski. Vispārīgi vienkārši operatoru piemēri ir aritmētiski operatori (piemēram +, -, /), salīdzināšanas operatori (piemēram >, >=, <), loģisku operāciju operatori (piemēram AND, arī zināms pār && atšķirībā no valodas), u.tt.

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.

Getter method : Saņēmējmetode. OOP paradigmā ir pieņemts ka objektu atribūtiem nav jābūt publiskiem, proti, ka nevajag dot ispēju tiem piekļūt tiešā veidā caur objekta identifikatoru. Piekļuve tiek organizēta izmantojot speciālu metodi, kurai, būdamai klases sastāvdaļai, ir piekļuve privātiem klases atribūtiem. Metodi, kura atgriež konkrēta atribūta vērtību, sauc par saņēmējmetodi (getter no angļu valodas vārda get).

Setter method : Piešķīrējmetode. OOP paradigmā ir pieņemts ka objektu atribūtiem nav jābūt publiskiem, proti, ka nevajag dot ispēju tiem piekļūt tiešā veidā caur objekta identifikatoru. Piekļuve tiek organizēta izmantojot speciālu metodi, kurai, būdamai klases sastāvdaļai, ir piekļuve privātiem klases atribūtiem. Metodi, kura piešķir konkrētam atribūtam vērtību, sauc par piešķīrējmetodi (setter no angļu valodas vārda set).