Liepājas Valsts tehnikums

2PT kursa audzēknis

Edžus Krūmiņš

**Mainīgie, to datu tipi, pamatdarbības darbam ar tiem programmēšanas valodā Java**

Mācību materiāls eksāmenam programmēšanā

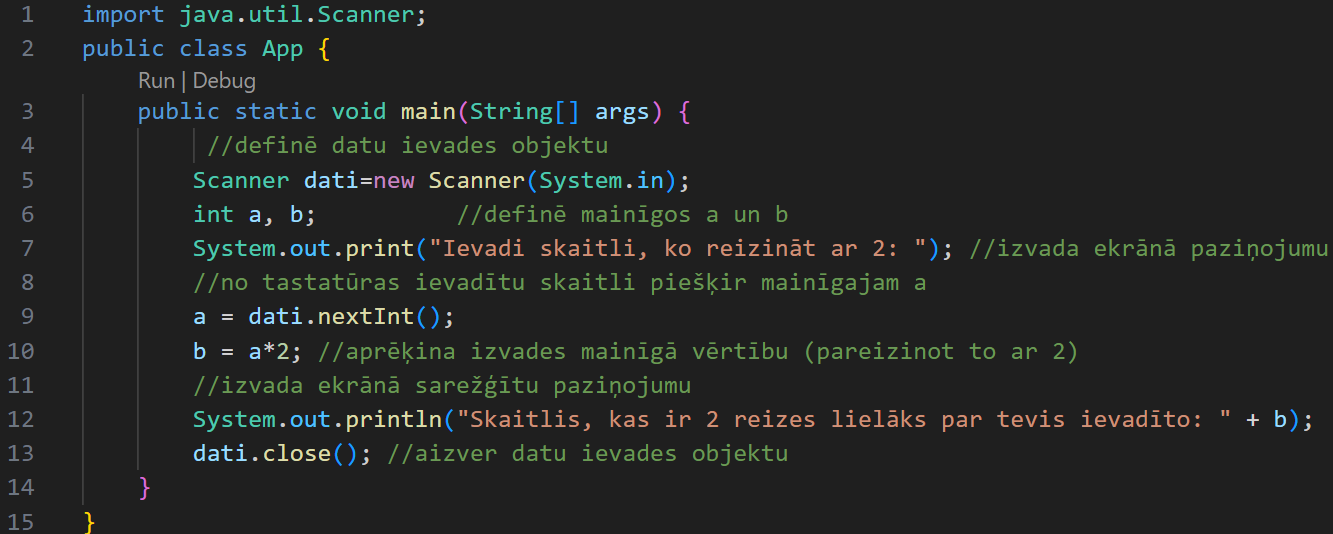
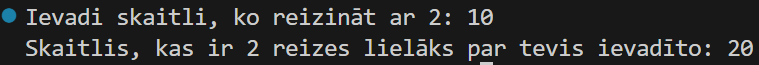
Liepāja 2025

**Mainīgie un to izmantošana**

Mainīgie lielumi programmā tiek izmantoti noteikta veida informācijas saglabāšanai un apstrādei. Tos var izmantot, lai kodā pēc iespējas vienkāršāk izmainītu kādu vērtību, lai veiktu ar tiem dažādas darbības (saskaitīšana, atņemšana, atdalīšana, u.c.) un lai vienkāršotu kodu, dotu iespēju tam kļūt dinamiskam, utt.

Piemēram, lietotājs var ievadīt kādu skaitli, ko programma saglabā ar nosaukumu “Skaitlis”. Lietotājs vēlas, lai šis skaitlis tiktu izvadīts atpakaļ, tikai 2 reizes lielāks. Programma mainīgo “Skaitlis” pareizina ar 2 un izvada konsolē.  
Šajā gadījumā nav svarīgi, kādu skaitli lietotājs ievadījis, jo tas vienmēr tiks izvadīts konsolē 2 reizes lielāks, pateicoties tam, ka programma izmanto mainīgos.

**Mainīgo definēšana, koda piemērs**

Tālāk aplūkosim, kā tas varētu izskatīties kodā…**** 

1. Varam redzēt, ka vispirms iesākam, definējot datu ievades objektu – tas ir vajadzīgs, lai mēs caur konsoli varētu “sarunāties” ar lietotāju jeb ļaut viņam ievadīt dažādas vērtības.
2. Tad definējam mainīgos a,b – definējot skaitļus bez decimālām vērtībām, izmanto int, taču par datu tipiem parunāsim nedaudz vēlāk.
3. Nākamais solis ir printēt konsolē paziņojumu un saņemt atbildi no lietotāja – izmantojam definēto datu ievades objektu dati.nextInt().
4. Visbeidzot aprēķinam mainīgā b vērtību, pareizinot to ar 2, izvadām konsolē paziņojumu ar rezultātu – labā prakse ir aizvērt datu ievades objektu pēc tā izmantošanas ar dati.close().

**Mainīgo datu tipi**

Tagad aplūkosim mainīgo datu tipus. Tie iedalās 3 lielās kategorijās – veselo skaitļu tipi, reālo skaitļu tipi, loģiskais tips un simbolu tips. Katram skaitļu tipam ir arī noteikts aizņemtās atmiņas apjoms un iespējamās darbības, kuras ar tiem var veikt.  
Mainīgajam datu tipam boolean nav praktiski definēts aizņemtās atmiņas apjoms un tā iespējamās darbības nav veicamas tikai savstarpēji, bet gan kā salīdzināšanas un loģiskās operācijas.

|  |  |  |  |
| --- | --- | --- | --- |
| Tips | Piemērs | Aizņemtās atmiņas apjoms | Iespējamās darbības |
| **byte** | *21* – paredzēts ļoti mazu skaitļu uzglabāšanai, kas neaizņem daudz vietas. | 8 biti (1 baits) | Saskaitīt (+) Atņemt (-) Reizināt (\*) Dalīt(/) Dalīt pēc moduļa (%)  **(veselo skaitļu tips)** |
| **short** | *21* – domāts nedaudz lielāku skaitļu uzglabāšanai, taču mazākiem kā int. | 16 biti (2 baiti) |
| **int** | *21* – visbiežāk izmantotais datu tips, lai uzglabātu veselos skaitļus. | 32 biti (4 baiti) |
| **long** | *21* – lietots, lai uzglabātu ļoti lielus skaitļus. | 64 biti (8 baiti) |
| **float** | *21.25* – lieto, lai uzglabātu skaitļus ar komatiem. | 32 biti (4 baiti) | Saskaitīt (+) Atņemt (-) Reizināt (\*) Dalīt(/)  **(reālo skaitļu tips)** |
| **double** | *21.25* – izmanto līdzīgi kā Float, taču lai uzglabātu pat lielākus skaitļus. | 64 biti (8 baiti) |
| **boolean** | *True* – izmanto, lai pārbaudītu, vai vērtība ir True (patiesa) vai False (aplama). | Teorētiski, tas atmiņā aizņem 1 bitu, taču praktiski tas nav precīzi definēts. | Salīdzināšanas operācijas: Vienāds (==) Nav vienāds (!=) Lielāks (>) Mazāks (<) Lielāks vai vienāds (>=) Mazāks vai vienāds (<=)  Loģiskās operācijas: Loģiskais UN (&&) Loģiskais VAI (||) Loģiskais NĒ (!)  **(loģiskais tips)** |

**boolean lielumu A un B veikto loģisko darbību rezultāti**

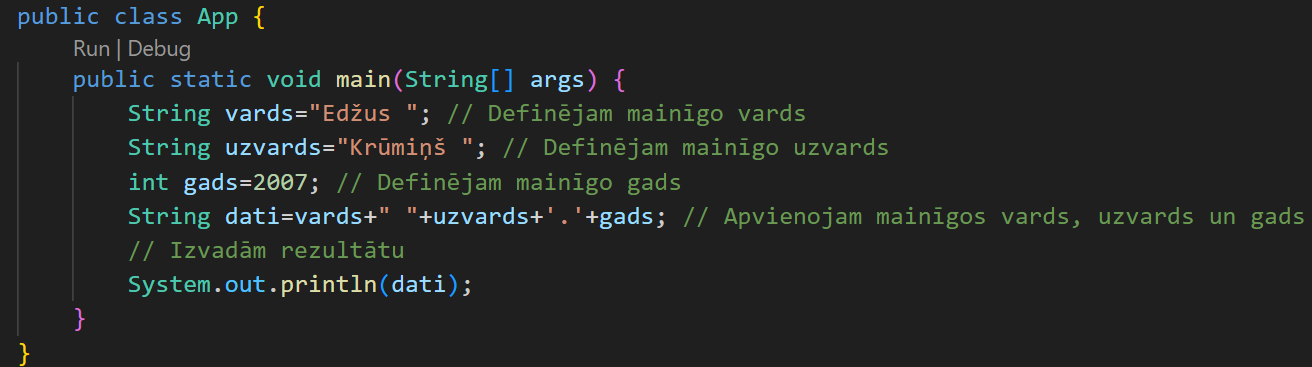
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **!A** | **A || B** | **A && B** |
| True | True | False | True | True |
| True | False | False | True | False |
| False | True | True | True | False |
| False | False | True | False | False |

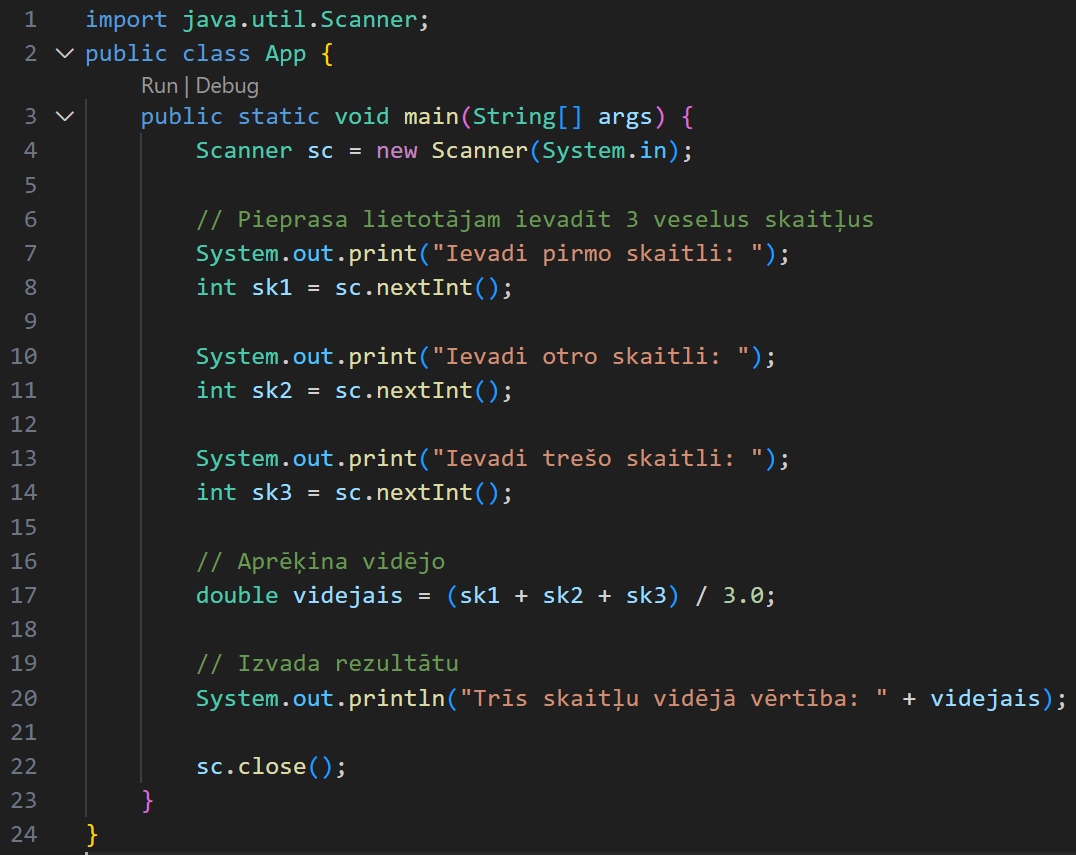
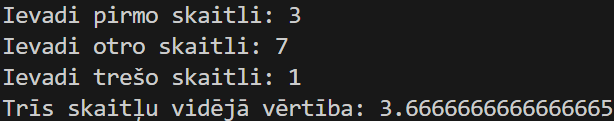
**Simbolu tips**

char tipam atbilst viens Unicode simbols, kas atmiņā aizņem 16 bitus (2 baitus). Simbolu tipa konstante ir simbols, kas ietverts apostrofos, piemēram: 'A', '2', 'k'.

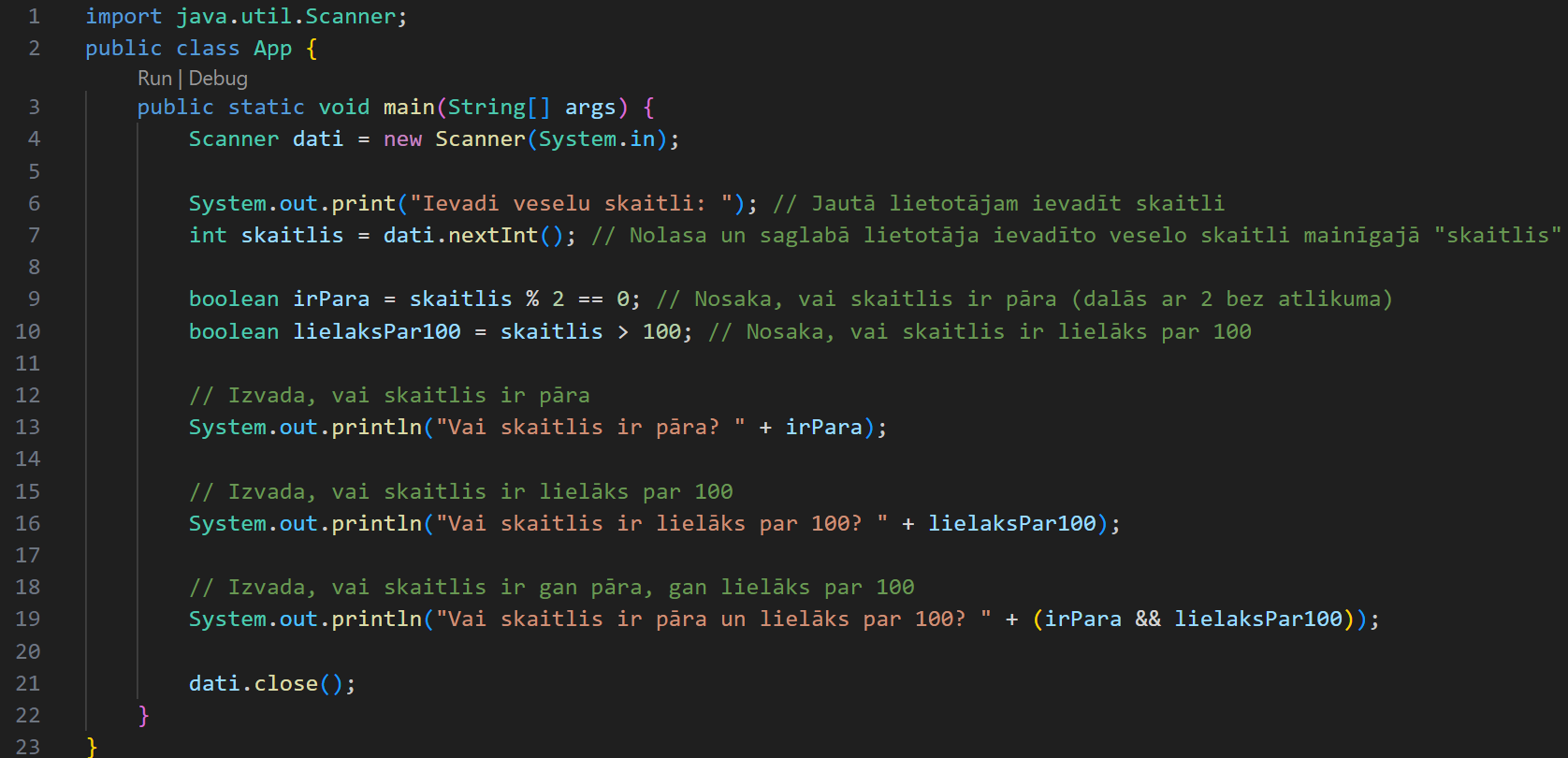
Simbolu virknes veido, izmantojot String tipa mainīgos. Simbolu virkne ir simboli, kas ietverti pēdiņās, piemēram: "A", "123", "", "cena".  
Izmantojot operatoru "+", simbolu virknes var apvienot gan savā starpā, gan ar citu tipu lielumiem.

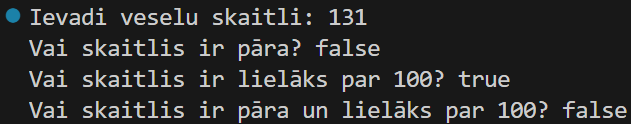
Koda piemērs:

   
  
Šis ir vienkāršs koda piemērs, ar kura palīdzību var parādīt to, kā String tipa mainīgos var apvienot gan savā starpā, gan ar citu tipu lielumiem.  
Ievēro, ka atšķirībā no citu datu tipiem, String datu tipu definē ar **lielo** sākuma burtu ‘S’. Pārējiem mainīgo datu tipiem tas tā nav.

**Noslēgumā aplūkosim vēl 2 uzdevumus un to koda piemērus.**  
  
Izveidosim programmu, kas pieprasa lietotājam ievadīt trīs veselus skaitļus un aprēķina to vidējo aritmētisko vērtību. Rezultātu izvadām konsolē.  
  
 

1. Vispirms, kā jau iepriekš, mēs definējam Scanneri, lai lietotājs varētu ievadīt datus. Šoreiz to sauksim par sc.
2. Tad jautājam lietotājam ievadīt pirmo, otro un trešo skaitli, visus šos skaitļus saglabājot mainīgajos. (Ievēro, ka mainīgos varam definēt tieši, kad ieglabājam tajos rezultātu, ne tikai koda sākumā).
3. Nākamais solis ir aprēķināt vidējo aritmētisko vērtību šiem skaitļiem. Tam mēs izmantojam mainīgo double (jo rezultāts visdrīzāk nebūs vesels skaitlis). Tieši šī iemesla dēļ, mēs dalām visus 3 int tipa mainīgos ar 3.0, jo izmantojam mainīgo double.
4. Visbeidzot izvadām rezultātu konsolē, atpakaļ lietotājam un aizveram Scanneri ar sc.close().

Pēdējo uzdevuma piemēru aplūkosim saistībā ar boolean mainīgo…  
  
Izveidosim programmu, kas prasa lietotājam ievadīt veselu skaitli.  
Programmai ir jāizvada:  
Vai skaitlis ir pāra?  
Vai tas ir lielāks par 100?  
Vai tas ir pāra un lielāks par 100?



1. Pēc Scanner definēšanas, jautājam lietotājam ievadīt skaitli un saglabājam to mainīgajā int skaitlis.
2. Tālāk pārbaudām uzdevumā dotos nosacījumus attiecībā pret ievadīto skaitli. Definējam divus boolean tipa mainīgos un pārbaudām, vai skaitlis ir pāra un to, vai tas ir lielāks par 100.
3. Visbeidzot mēs izvadām rezultātus konsolē ar boolean vērtībām (True/False). Ievēro, kā pēdējā System.out.println mēs tajā pašā rindā pārbaudām, vai skaitlis ir gan pāra, gan lielāks par 100 un izvadām to vērtību. Līdz ar to, mēs neizmantojām tik daudz koda rindiņas un uztaisījām šo kodu optimizētāku.
4. Kā jau vienmēr, pēdējais solis šādās Scanner programmās ir aizvērt to. Šoreiz, izmantojot dati.close().

**Kopsavilkums**

Mainīgie ir jebkuras programmas pamatelements. Tie ļauj saglabāt, apstrādāt un izmantot datus visdažādākajos veidos – sākot ar vienkāršu skaitļu ievadi līdz sarežģītām loģiskām darbībām. Java valodā katram mainīgajam ir noteikts datu tips, kas nosaka, kādus datus tas var saturēt un kādas operācijas ar tiem ir iespējamas. Saprotot šo pamatu, iespējams veidot loģiskas, funkcionālas un efektīvas programmas.

Šajā materiālā tika aplūkoti gan dažādi datu tipi, gan piemēri to lietošanai, ieskaitot reālus uzdevumus, kas nostiprina izpratni par tiem. Prasmīga mainīgo izmantošana ir svarīgs solis ceļā uz profesionālu programmētāja izaugsmi.

**Informācijas avoti**

* Java mācību materiāli no Liepājas Valsts tehnikuma mācību kursa <https://skolo.lv/pluginfile.php/78273365/mod_resource/content/0/Skaitliskie%20un%20simboliskie%20datu%20tipi.pdf>
* W3Schools – Java Tutorial: <https://www.w3schools.com/java/>
* GeeksforGeeks – Java Variables: <https://www.geeksforgeeks.org/variables-in-java/>
* Paša izstrādātie un saglabātie koda piemēri mācību nolūkos