|  |
| --- |
| **Teme:**   * **Strukture : sklad,vrsta (**Utrjevanje dela s tabelami**)** * LIFO, delovanje sklada * Vrste, realizacija krožne vrste, vrsta s stražarjem, * Dodatne operacije na krožnih vrstah |

**Naloga 1**

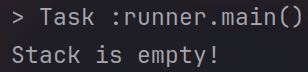
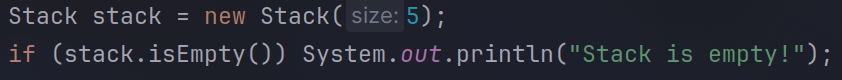
Vzemite implementacijo sklada kot je bila izvedena na urah predmeta Računalništvo03.

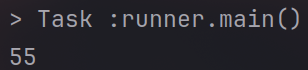
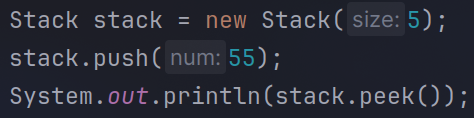
Implementaciji dodajte metodi (če še niso spisane):

1. isEmpty() - vrne true ali false v odvisnosti od tega ali sklad prazen ali poln
2. int peek() – ki vrne vrednost z vrha sklada, vendar ne ažurira kazalca vrha sklada

Spišite demonstracijo delovanja sklada oz. navedenih metod, v stilu :

S sklada lahko vzamem element zgolj, če ta ni prazen

 Na vrhu sklada se nahaja vrednost 55



**Naloga 2**

V pomnilniškem modelu za vse morebitne javanske programe na voljo samo eno področje za implementacijo sklada. V primeru, da zaganjamo 3 programe, je potrebno to področje razdeliti na 3 individualne (neprekrivne, neodvisne) sklade, ki vsak zase ustrezno deluje.

Spišite javanski program, ki bo izvedel simulacijo delovanja zahtevane realizacije. Pri tem predpostavite, da celotno področje sklada definiramo kot eno samo tabelo pozitivnih celih števil velikosti 50. Simulacija mora vsebovati polnjenje/praznenje posameznih skladov in vizualizacijo trenutne vsebine posameznega sklada.Delček simulacije npr. izgleda kot:

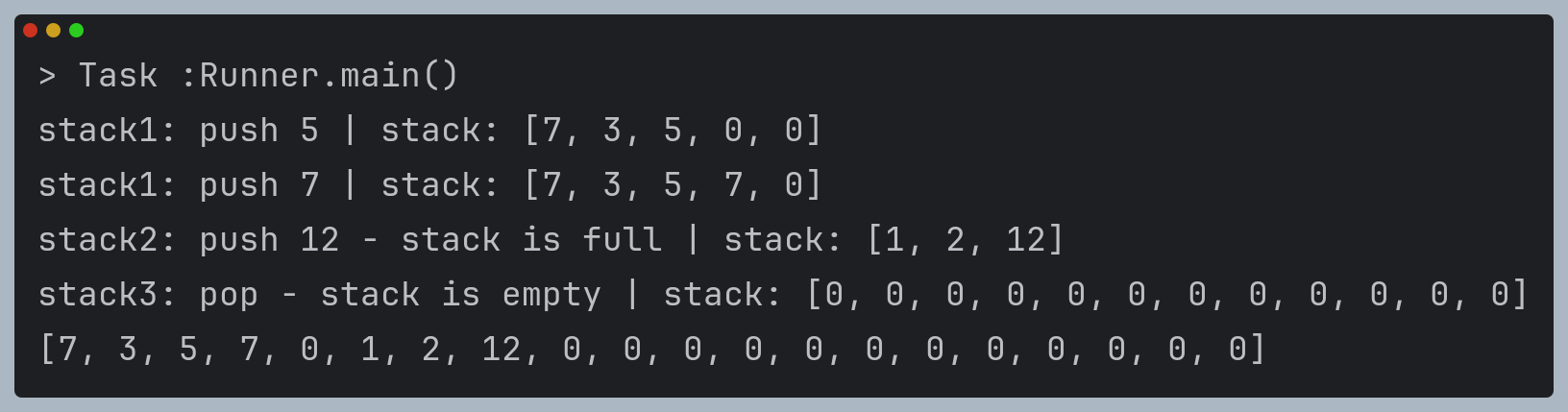
Sklad1 : dodajam 5 : vrednosti na skladu so : 7,3,5

Sklad1: dodajam 7 : vrednosti na skladu so: 7,3,5,7

Sklad2: dodajam 12-sklad poln : vrednosti na skladu so: 1,3,9,7,6,5, …, 2

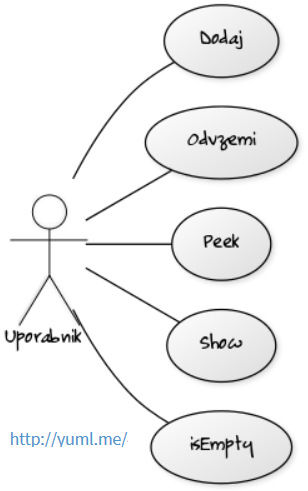
Sklad3: odvzemam-sklad je prazen:

….



**Naloga 3**

Na vrsti pozitivnih celih števil z ničlo, bi želeli realizirati mehanizme, ki so podani z naslednjim diagramom rabe:

iz diagram je razvidno, da uporabnik pri rabi 'programa' lahko uporablja 5 različnih mehanizmov (Dodaj,Odvzemi,Peek,Show,isEmpty). Očitno so to mehanizmi programa, se pravi, da so verjetno realizirani kot metode, in da bi bile uporabne s strani uporabnika, morajo biti njemu dosegljive, se pravi javne.

Diagramu pravimo Diagram primerov rabe oz. Use-case diagram. Kot tak je specifican kot eden izmed diagrama standarda UML (Unified Modeling Language).

V splošnem diagram kaže, na kakšen način akter/Uporabnik lahko uporabi program oz. katere mehanizme lahko kliče. Velikokrat se uporablja za opis vmesnikov, pri načrtovanju menijev, …

Specifikacija in opis mehanizmov metod se nahaja v dokumentu prepisa, na hitro pa takole:

dodaj/1 doda s parametrom podan element v krožno vrsto

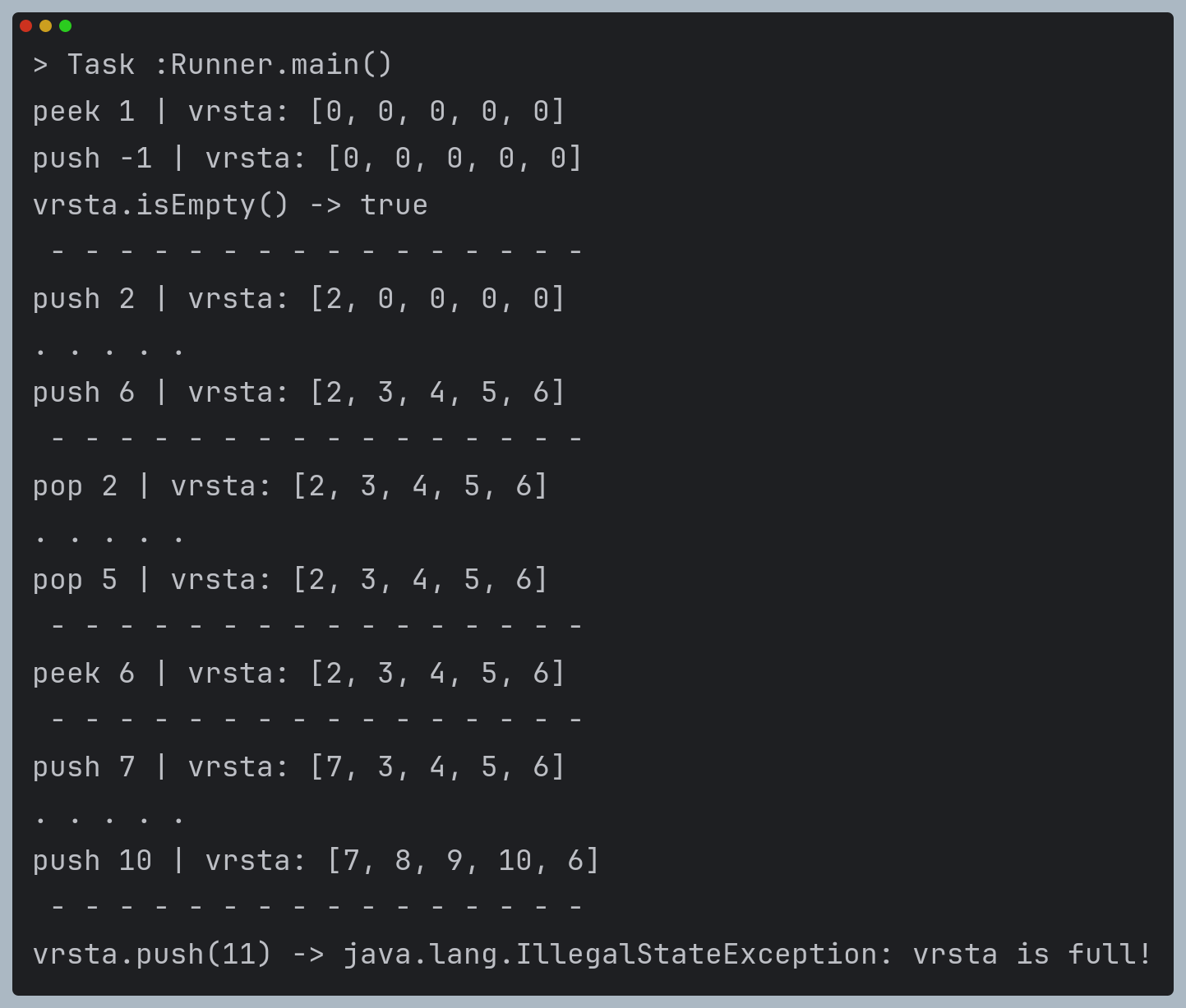
odvzemi/0 vzame el. Iz glave vrste; vrne -1 če nima kaj vrniti

peek/0 vrne vrednost elementa glava, pri tem ne vpliva na vrednosti kazalcev vrste; vrne

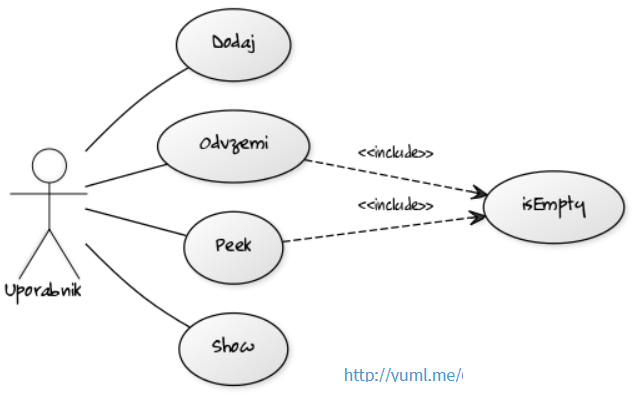
1 če elementa ni

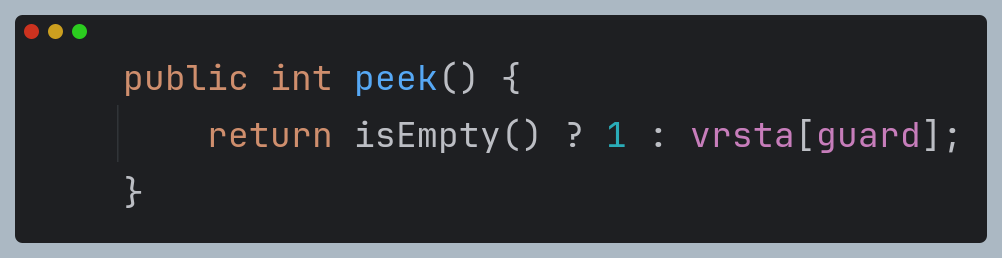
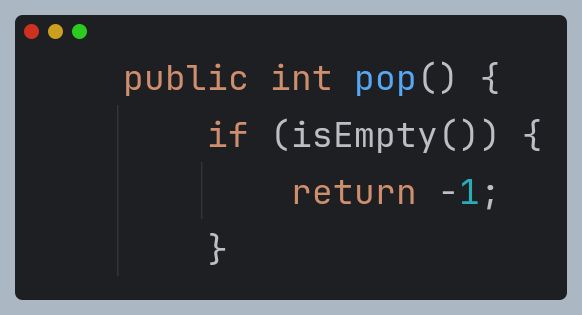
show/0 prikaže trenutno vsebino vrste v obliki [5,1,3]

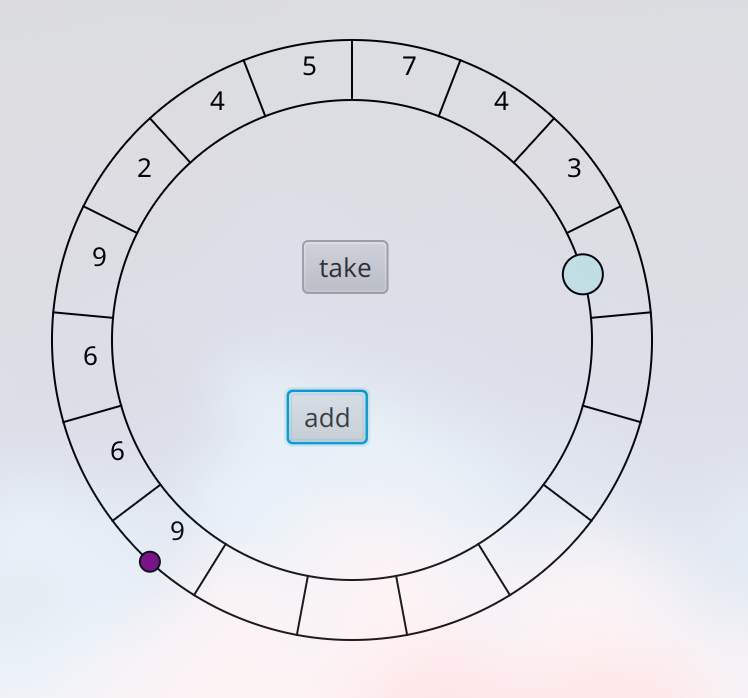
isEmpty/0 vrne true, če je vrsta prazna

Spišite demonstracijski program, ki bo pokazal pravilnost delovanja vseh /vsake posebej/ implementirane metode. (odvzemi -> pop) & (dodaj -> push)

**Naloga 4**

Delna realizacija naloge 3, podana v predpisih ni najboljša. Problematična sta mehanizma odvzemanja elementa in vpogled v vrednost elementa glava vrste 'peek'. Ti metodi vračata vrednost -1 v primeru, da nimata kaj vrniti. Mehanizma bi želeli spremeniti tako, da bi bila mehanizma obeh metod pogojena, z metodo isEmpty, kot to kaže desni diagram. Naše znanje direktne implementacije desne slike še ne pokriva, zato naj bo sprememba naslednja: iz diagram sledi, da metoda isEmpty ni več javna metoda, njena funkcionalnost pa naj v obeh metodah v nadaljevanju zamenja obstoječe testiranje pogoja.

**Naloga 5[[1]](#footnote-2)**

Želeli bi grafično demonstracijo delovanja krožne vrste opredeljene s predhodnima nalogama. Pri tem naj bo za vizualizacijo uporabljena tehnologija javaFX. Kot primer si lahko ogledate primer enostavne vizualizacije iz repozitorija dodatnih vaj.

1. Naloga ni obvezna. Če vas boljša ocena ne zanima, vam je pač ni potrebno izvesti [↑](#footnote-ref-2)