|  |
| --- |
| ***Teme:***   * ***Uporaba obstoječih razredov za deloz nizi znakov***   + Nizi znakov: razredi:     - String,     - ( StringBuffer, StringBuilder);   + Ustvarjenje niza   + Določanje dolžine niza, / length()   + Primerjava nizov / compareTo, equals   + (tabele nizov)   + Naslavljanje posameznih znakov niza / charAt   + Podnizi v nizu:     - Izločanje dela niza /substring     - Identifikacija podniza v nizu /indexOf   + posebnosti (java.lang.Runtime)     - nadzor porabe in razpoložljivosti pomnilnika   + Razredne knjižnice. |

|  |
| --- |
| **Ref:**  <http://docs.oracle.com/javase/17/docs/api/java.base/java/lang/String.html>, apr.2017, apr.2022  <http://docs.oracle.com/javase/17/docs/api/java.base/java/lang/StringBuffer.html>, apr. 2017, apr.2022  [http://docs.oracle.com/javase/17/docs/api/java.base/java/lang/Runtime.html, apr](http://docs.oracle.com/javase/17/docs/api/java.base/java/lang/Runtime.html,%20apr). 2017, apr.202 |

**Opombe: obvezna je izvedba in oddaja prvih 11 nalog te vaje ! Izvedba ostalih nalog in realizacij v poročilu je zaželena. Rok za oddajo je 14 od dneva objave tegale besedila.**

Rešitve naj pričakovano kažejo obvladovanje dela z nizi in metodami razreda String, če z besedilom naloge ni navedeno drugače.

**Naloga 1**

*Konstruiranje niza ( uporaba konstruktorjev razreda) iz praznega niza, iz obstoječega niza, iz ....*

Napišite demonstracijski program, ki pokaže, kako lahko niz znakov (String) naredite iz konstantnega niza znakov(literala), iz obstoječega niza, iz tabele znakov, prazen niz znakov. Nize tudi izpišite.

**Naloga 2**

Napišite program, s katerim ustvarite en sam niz in mu nato 1000x 'spremenite' vrednost. Inicialna (začetna) in vse nove vrednosti nizov so poljubne. Po vsaki spremembi izpišite velikost uporabljenega in velikost prostega pomnilnika.

//Runtime.getRuntime().freeMemory();

//Runtime.getRuntime().totalMemory();

**Naloga 3**

*Indeksiranje elementov niza (charAt) in metoda za vračanje dolžine niza (length() )*

1. Napišite metodo izpisiPrvaInZadnjaDva, ki kot parameter dobi niz znakov (String) in izpiše zaporedno najprej prva dva, nato zadnja dva znaka.
2. Napišite metodo vrniPrvaInZadnjaDva, ki kot parameter dobi niz znakov (String) in kot niz vrne enako zaporedje znakov kot jih izpiše zpisiPrvaInZadnjaDva.

(\* v primeru, da znakov ni dovolj, naj izpiše/vrne ustrezno manj znakov\*)

**Naloga 4**

*Ugotavljanje, ali se v nizu nahaja črka, podniz, ter kje se nahaja ( indexOf() )*

1. Napišite metodo ***boolean aliJeVNizu(String niz, char c);*** , ki ugotovi ali se v nizu ***niz*** nahaja znak ***c.***
2. Napišite metodo ***boolean aliJeVNizu(String niz, Stringr podniz);*** , ki ugotovi ali se v nizu ***niz*** nahaja podniz ***podniz.***
3. Napišite metodo ***int kjeiJeVNizu(String niz, Stringr podniz);*** , ki ugotovi na kateri poziciji v nizu ***niz se*** nahaja podniz ***podniz***. V primeru, da ga ni, naj metoda vrne vrednost -1.
4. Napišite metodo ***int kolikoJeVNizu(String niz, char c);*** , ki ugotovi, kolikokrat se v nizu ***niz*** nahaja znak ***c.***
5. Napišite metodo ***int kolikoJeVNizu(String niz, String podniz);*** , ki ugotovi, kolikokrat se v nizu ***niz*** nahaja podniz ***podniz.***

**Naloga 5**

Orišite ( in pokažite z demonstracijskim programom) vse načine stikanja dveh ali več nizov.

**Naloga 6**

*Rezanje niza, brisanje podniza iz niza (String)*

Realizirajte metode nad nizi, ki jih podajajo naslednja imena. Pri tem upoštevajte, da mora biti niz, ki ga manipulirate, parameter metode:

1. brisiPrvoPojaviteNiza npr. : *(static) String brisiPrvoPojaviteNiza(String original, String brisan);*
2. brisiZadnjoPojavitevNiza
3. brisiVsePojavitvePodniza
4. vrniPrvoPolovicoNiza
5. vrniZadnjoPolovicoNiza
6. vrniSrednjoTretjinoNiza
7. odreziZadnjihNZnakovNiza

**Naloga 7**

*Zamenjava besede v nizu z novo besedo*

1. String zamenjajPrviNizSPodnizom(String niz, String osnovni, String novi); v nizu niz zamenja prvi podniz osnovni s podnizom novi.
2. String zamenjajZadnjiNizSPodnizom(String niz, String osnovni, String novi);
3. String zamenjajVsePodnizeZNizom(String niz, String osnovni, String novi);

**Naloga 8**

*Primerjava nizov compareTo(), equals() : enakost, katera je leksikalno večja, manjša*

S primeri ponazorite podobnosti in razlike navedenih metod.

**Naloga 9**

*Generiranje nizov z naključno vsebino, generiranje nizov iz elementov tabele nizov, …*

Realizirajte metode:

1. String generirajNizCrke(int dolzinaNiza); // zgolj A..Z in a..z
2. String generirajNizCrkeStevilke(int dolzinaNiza); // A..Z in a..z in 0..9
3. String generirajNizStevilke(int dolzinaNiza); // 0..9
4. String generirajNizCrkeVelike(int dolzinaNiza);
5. String generirajNizCrkeMale(int dolzinaNiza);
6. String generirajNizZnaki(int dolzinaNiza); // poljuben znak s kodo spodnjega zloga od 33..127

**Naloga 10**

*Tabela nizov, inicializacija (naloga 9)*

Kreirajte program, ki bo ustvaril tabelo desetih nizov, njihovo vsebino naključno generiral z naključno izbiro ene izmed metod naloge 9 in tabelo na koncu izpisal na zaslon. Tipično bosta aplikacijo sestavljali (poleg deklaracije tabele) dva zaporedna klica metod:

String[] t:

generirajNizeVTabelo(t);

izpisiTabeloNizov(t);

**Naloga 11**

*Tabela nizo, iskanje*

Realizirajte metode:

1. boolean aliJeVTabeliNiz(String[] t, String niz); // se niz nahaja v tabeli t
2. String vrniNajkrajsiNizVTabeli(String[] t); // vrne niz z najmanjso dolzino
3. String vrniNajdaljsiNizVTabeli(String[] t);
4. String vrniNajmanjsiNizVTabeli(String[] t); // vrne leksikalno najmanjsi niz
5. String vrniiNajvecjiNizVTabeli(String[] t);
6. boolean aliTabelaNizovVsebujeDuplikate(String[] t); // sta mogoče v tabeli vsaj dva niza enaka
7. String vrniPrviDupliciranNizVTabeliNizov(String[] t); // vrne vrednost dupliciranega niza

**Naloga 12**

*Tabela nizov, razvrščanje*

Napišite metodi:

1. void razvrstiTabNizovPoDolziniNizov(String[] t);
2. void razvrstiTabNizovLeksikalno(String[] t);

prva naj razvrsti nize v tabeli po njihovi dolžini in najkrajšega do najdaljšega, druga po velikosti oz. Vrednosti nizov (leksikalno).

**Naloga 13**

*Formiranje knjižnjice vseh metod predhodnih nalog*

Kreirajte samostojen razred z imenom OperacijeNadNizi. Razred naj vsebuje vse metode iz nalog 2-12. Pri tem morajo biti vse metode v razredu ***public*** in ***static***.

Nato spišite demonstracijski program, ki bo nad izbranih nizom/izbranimi nizi izvedel demonstracijo izvedbe vsaj 65% vseh metod razreda OperacijeNadNizi.

1. Oba razreda (aplikacija in Operacije…) naj bosta v isti izvorni datoteki.
2. Vsak razred naj bo v svoji lastni datoteki.
3. Vsak razred naj bo v svoji datoteki; pri tem aplikaciji kot prvo vrstico datoteke dodajte vrstico: package aplikacija; in datoteki OperacijeNadRazredi kot prvo vrstico vsebino: package operacijenadnizi;

Spišite ugotovitve o prevajanju in izvajanju vseh treh variant. Spišite tudi vse popravke, ki jih morate dodati za realizacijo variante c.

1. Izdelajte javansko dokumentacijo o razredu OperacijeNadRazredi !!

**Naloga 14**

V splošnem so metode razreda String hitre, kadar gre za pregledovanje vsebine, izločanje (‚parsing‘) in razbijanje niza na podnize. Ker pa so nizi v osnovi nespremenljivi, pa so počasni pri operacijah, kjer je rezultat operacije spremenjena dolžina niza, ker se spremeba dolžine vedno izvede z ustvarjenjem novega niza. Pri teh operacijah sta učinkovitejša razred StringBuffer in StringBuilder, ki predstavljata spremenljiva (mutable) objekta. StringBuffer je strarejši (Java 1.4), StringBuilder novejši in v določenih primerih hitrejši; oba pa poleg vseh metod razreda String zagotavljata še operacije tipa apend/insert/delete (substring in concat sta tako tudi v string realizirana z StringBuilder). StringBuffer garantiran, da se bodo nizi obnašali ustrezno tudi pri večnitnem izvajanju (sinhronizacija,..), vendar to plačamo s hitrostjo.

1. Vse metode 5,6 in 7 naloge predelajte tako, da vse tipe vračalnih vrednosti in parametrov zamenjate z:
   1. StringBuffer
   2. StringBuilder

Pri tem imenom metod dodajte sufiks Buffer ali Builder v odvisnosti od tega, kater mehanizem boste uporabili. Optimizirajte izvajanje metod s tem, da v realizaciji zahtevanih mehanizmov uporabite morebitne učinkovitejše metode novo uporabljenih razredov

1. Dodajte vse metode, izdelane v točki a) naloge 14 v knjižnico metod ustvarjeno v nalogi 13.

|  |
| --- |
| ***Teme:***   * ***Nizi utrjevanje***   + Nizi znakov   + Tabela nizov   + Funkcije za delo z nizi   + Razred Scanner, funkcija split()   + Razreda iz paketa java.time : Instant, Duration   <https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html> , april 2017  <https://docs.oracle.com/javase/8/docs/api/java/time/Instant.html> , april 2017 |

**Naloga 15**

Naloga 14 zahteva alternativno realizacijo metod nalog 5,6,7. V tej nalogi bi želeli ‘preveriti’ učinkovitos posameznih realizacij; tako prostorko kot tudi časovno. Izhodišče za prostorkost naj bo naloga 2; pomnilni bomo pomerili pred izvajanje in po izvajanju 1000 ponovites istega postopka, prav tako pa bomo merili čas z razliko med zaćetnim in končnim časom. Rezultat naj bo tabelo (ali naj jo generira testni program ali naj bo spisana v poročilo), ki bo za vsak mehanizem za vse tri variantepodal diferenco v stanju zasedenosti pomnilnka in diferenco v izvajalnem času.

Za merjenje časa ne bomo uporabili ure sistema (System.currentTimeMillis() ali System.nanoTime()) temveč uporabite:

Instant zacetek = Instant.now();

// tesni postopek

Instant konec = Instant.now();

long pretekloCasa = Duration.between(zacetek, konec).toMillis();

|  |
| --- |
| **Besedilo se nanaša na naloge 16-20:**  **Javanska metoda dobi vrednosti dveh parametrov tipa String. Njena naloga je razbiti niz, podan s prvim parametrom na podnize tako, da vsakega naslednjega prične niz, podan z vrednostjo drugega parametra.**  **Npr. a(„olalainolala“,“la“) ------> { „o“ , „la“ , „laino“ , „la“ , „la“ }**  **V vsaki izmed nalog realizirate po dve metodi:**   1. **Metoda vizualizira (izpiše) posamezne nize na zaslon, vsakega v svoji vrstici** 2. **Metoda vrne tabelo dobljenih nizov klicočemu programu** |

**Naloga 16**

Realizacija zahtevane funkcionalnosti z uporabo razreda Scanner

**Naloga 17**

Realiozacija zahtevane funkcionalnosti z uporabo metode split() razreda String.

**Naloga 18**

Realizacija naloge z uporabo metod razreda String, brez uporabe metode split() in brez rabe razreda Scanner.

**Naloga 19**

Spišite javanski program, ki bo demostriral uporabo funkcij nalog 16,17,18. Zagotavljati mora tudi branje ustrezno dolgega niza, ter vizualizacijo rezultatov procesiranja metod, če ta (zahtevano) ni implementirana znotraj zahtev posameznih nalog.

**Naloga 20[[1]](#endnote-1)**

Realizirajte javanski program kot vizualno aplikacijo. Besedilno polje za vnos ‘dolge besede’, besedilno polje za vnos ‘delitvenega’ niza, besedilno polje čez več vrstic za izpovanje rezultatov metod. Realizacija šestih metod bo zahtevala verjetno 6 gumbov, za izvedbo vsake izmed metod po enega. Vizualizacija naj bo izvedena s FX tehnologijo.

**Naloga 21[[2]](#endnote-2)**

Iz ukazne vrstice preberite 8 besed.

1. Napišite funkcijo, iz izpiše najkrajšo besedo (besede).
2. Napišite funkcijo, iz izpiše besedo (besede), ki ima najmanj samoglasnikov.
3. Napišite funkcijo, iz izpiše leksikografsko gledano prvo in zadnjo besedo.

Testirajte delovanje funkcij, zabeležite testne podatke in dobljene rezultate.

1. Naloga ni obvezna, vpliva pa na oceno [↑](#endnote-ref-1)
2. Naloga ni obvezana (testna naloga, rekapitulacija, vsebovana v prvim naboru nalog) [↑](#endnote-ref-2)