VEŽBA BR. 4

Klasa string i nizovi objekata

Cilj vežbe: Upoznavanje sa klasom String, njenim metodama i nizovima objekta

U dosadašnjim primerima su se nizovi zakova predstavljali korišćenjem primitivnog tipa "**char**" i oznake za nizove (**char[]**). Ovo, međutim, nije najbolji način za rad sa nizovima znakova. Često je potrebno nad određenom reči ili izrazom uraditi nee složene operacije – izdvojiti smao jedan deo reči ili izraza, proveriti da li se u izrazu nalazi neka reč, zameniti neko slovo nekim drugim, spojiti neke reči u rečenice, itd. Ako bi se koristio „**char**" niz, za sve ove operacije bi trebalo napisati posebne metode. Čak bi se i poređenje jednakosti dva niza slovo po slovo moralo implementirati na isti način. U Javi postoji predefinisana klasa za rad sa nizovima znakova koja već ima implementirane mnoge od ovih funkcionalnosti.

To je klasa **String.**

Deklaracija String promenljive sse vrši na isti način kao i za bilo koju drugu promenljivu (String je klasa pa njen naziv počinje velikim slovom):

```
String nazivPromenljive;
```

Sa obzirom na to da je u pitanju klasa, **svaka promenljiva ovog tipa je objekat pa se mora inicijalizovati** pre nego što se može koristiti. Inicijalizacija se može uraditi na nekoliko načina. Prvo, moguće je incijalizovati objekat klase String korišćenjem konstruktora.

```
String nazivPromenljive = new String("neki niz znakova");
```

Konstruktor ove klse kao parametar prima neku String vrednost. Potrebno je primetiti da se **String vrednosti uvek pišu pod dvostrukim znacima navoda.** Nasuprot tome, "**char**" vrednosti se pišu pod jednostrukim znacima navoda.

String promenljive se mogu inicijalizovati i na kraći način. Efekat prethodne i naredne naredbe za inicijalizaciju je isti.

String **nazivPromenljive = "neki niz znakova"**;

Ovakav vid inicijalizacije je **moguće ostvariti samo kada su u pitanju String objekti i ne važi za druge klase.** Moguć je još jedan oblik inicijalizacije String promenljive koji je koristan u onim situacijama kada je niz promenljivih tipa „**char**" potrebno pretvoriti u String objekat iste sadržine. U ovom slučaju, konstruktor klase String kao parametar dobija niz promenljivih tipa „**char**".

char[] niz;

…

String s = **new** String(niz);

U ovom slučaju je efekat inicijalizacije isti kao i za objekte drugih klasa. Pre inicijalizacije, promenljiva ima „**null**“ vrednost. Pri inicijalizaciji se u memoriji računara zauzima prostor odgovarajuće veličine za smeštanje konkretne String vrednosti.

Pored **null** vrednosti, String promenljiva može da sadrži i **prazan string.** to je String vrednost koja **ne sadrži nijedan znak (čak ni prazna mesta) ali je promenljiva inicijalizovana.**

```
String nazivPromenljive  = “”;
```

Potrebno je skrenuti pažnju i na to šta se dešava kada se String promenljivoj koja već ima neku vrednost dodeli nova vrednost. Iako se čini logičnim da će se iskoristiti memorijski prostor koji je već zauzet, to se ne dešava. Umesto toga, **inicijalizuje se novi String objekat i on prima novu vrednost, a objekat koji sadrži staru vrednost se oslobađa pomoću „garbage collection“ mehanizma.**

String vrednosti su, u stvari, nizovi znakova pa se često dešava da je potrebno spojiti više vrednosti u jedan String. U pitanju je **nadovezivanje (concatenation) String vrednosti** i vrši se korišćenjem operatora za nadovezivanje (koji se piše isto kao i znak za sabiranje – „+“). Ovaj prostor se može koristiti isključivo sa String vrednostima.

```
nazivPromenljive = “neki niz znakova” + “neki drugi niz znakova”;
```

Operator za nadovezivanje String vrednosti je zapravo već korišćen u okviru „**println**“ komande da bi se formatirala neka složenija izlazna poruka. Tu se može primetiti još jedna karakteristika ovog operatora a to je da će **sve vrednosti koje nisu tipa String biti pretvorene u odgovarajući niz znakova i nadovezane na postojeću String vrednost.** Jedini preduslov je da je **makar jedna vrednost u celom izrazu String.**

Primer 1: Nadovezivanje stringova

```
public class Stringovi {

	public static void main(String[] args) {
		String s1 = "Dobar";
		String s2 = " dan.";
		String s3 = s1+s2;

		System.out.println(s3);
	}


	}
```

Koju vrednost dobija promenljiva ***s3*** nakon izvršenja programa?

Primer 2. Nadovezivanje stringova

```
	int temperatura = 16;

	String s4 = "Napolju je "+temperatura+" stepeni!";
```

Koju vrednost dobija promenljiva ***s4*** i šta se dešava sa promenljivom ***temperatura***?

Primer 3. Nadovezivanje stringova

```
1. String s5 = 28 + 42.66;
2. String s6 = "122.46" + 43;
3. s6 = 185 + "144.42";
4. s6 = "" + 122 + 285.58;
```

Koji od prethodnih izraza nije funkcionalan i zašto?

Šta se dešava u izrazu br. 2?

Šta se dešava u izrazu br. 3?

Šta se dešava u izrazu br. 4?

Poređenje jednakosti

Poređenje jednakosti za proste tipove podataka je relativno jednostavno, i vrši se pomoću operatora za poređenje jednakosti „==“. Sa obzirom na to da je String klasa, **String vrednosti se ne mogu porediti korišćenjem operatora za poređenje jednakosti.** Ako bi se to uradilo, **poredile bi se samo adrese objekata u memoriji a ne i sam sadržaj.** Umesto toga, potrebno je koristiti „**equals**“ **metodu** koja poredi dva String-a slovo po slovo i vraća „**true**“ ako su jednaki, a „**false**“ ako nisu. Pri tome, obraća se pažnja na velika i mala slova. Ova metoda se koristi na sledeći način:

```
promenljiva1.equals(promeljiva2);
```

Operator za poređenje jednakosti se može korisitit jedino kada se želi proveriti da li String promenljiva ima „**null**“ vrednost tj. da li je inicijalizovana ili ne.

```
promenljiva1 == null;
```

Primer 4. Poređenje dve string vrednosti

```
public class PoredjenjeStringova {

	public static void main(String[] args) {
		String s1 = "Uporedna recenica!";
```

```
		String s2 = new String("Uporedna recenica!");

		if(s1 == s2) {
			System.out.println("Uporedne vrednosti su jednake!");
		}else {
			System.out.println("Uporedne vrednosti nisu jednake!");
		}

		if(s1.equals(s2)) {
			System.out.println("Uporedne vrednosti su jednake!");
		}else {
			System.out.println("Uporedne vrednosti nisu jendake!");
		}
	}


	}
```

Šta se dešava u prvoj ***IF*** komandi?

Šta se dešava u drugoj ***IF*** komandi?

Druge metode klase String

Pošto je već rečeno da klasa String omogućava rad sa nizovima znakova na jendostavniji način, pružajući dodatne funkcionalnosti. Međutim, nekada je potrebno pojedinačno proći kroz sve znakove u String promenljivoj kao da je u pitanju običan niz znakova. Svaki znak String vrednosti **ima svoj indeks** a indeksi kreću od nule i tu se vidi sličnost sa običnim nizovima. Međutim, znakovima se ne pristupa koriććenjem uglastih zagrada kao kod nizova, već upotrebom metode ***charAt.*** Dužina string vrednosti se dobija upotrebom metode ***length.***

Primer 5. Napraviti klasu za prebrojavanje znakova koja ima:

- Statičku metodu ***prebroj*** koja kao parametar prima String i vraća koliko puta se u njemu pojavljuje slovo **'a'**.

```
class PrebrojavanjaZnakova{
	int prebroj(String s1) {
		int brojac = 0;
		for(int i = 0; i<s1.length(); i++) {
			if(s1.charAt(i)=='a') {
				brojac++;
			}
		}
		return brojac;
	}
}
```

```
public class Znakovi {

	public static void main(String[] args) {

		PrebrojavanjaZnakova pz = new PrebrojavanjaZnakova();
		String s2 = "Danas je lep dan!";
		System.out.println("Slovo a se u zadatoj recenici pojavljuje
"+pz.prebroj(s2)+" puta");
	}


	}
```

Objasniti ***Primer 5.***

Tabela metoda za rad sa nizovima znakova

Tabela 1. Metode klase String

Naziv metode	*Opis*
charAt(int indeks)	Vraća znak iz String promenljive koji je na određenoj poziciji.
compareTo(String s)	Poredi dve String vrednosti leksikografski (abecedno) i vraća pozitivan broj ako je prva vrednost “posle” druge vrednosti, negativan broj ako je “pre”, a nulu ako su jednaki.
compareToIgnoreCase(String s)	Radi isto što i prethodna metoda, ali ne obraća pažnju na to da li su u pitanju velika ili mala slova.
endsWith(String s)	Proverava da li se prvi String završava nizom znakova koji se nalazi u Stringu s i vraća “true” ako je tačno, a “false” ako je netačno.
equals(Object o)	Proverava da li je prvi String jednak unetom Stringu i vraća “true” ako jeste ili “false” ako nije. Kao parametar ova metoda bi trebalo da dobije String bez obzira na to što piše da je parametar tipa Object.
equalsIgnoreCase(String s)	Radi isto što i prethodna metoda samo se ne obraća pažnja na velika i mala slova, na primer (“DAN” i “dan” će biti isto protumačeni).
indexOf(char c)	Pronalazi i vraća indeks prvog pojavljivanja znaka koji je unet kao parametar u String

	vrednosti. Ako se taj znak ne može naći, vraća -1.
indexOf(String s)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
indexOf(char c, int index)	Pronalazi i vraća indeks prvog pojavljivanja znaka koji je unet kao parametar u String vrednosti. Pretraživanje počinje od pozicije u Stringu koja je uneta u vidu indeksa.
indexOf(String s, int index)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
lastIndexOf(char c)	Pronalazi i vraća indeks poslednjeg pojavljivanja znaka koji je unet kao parametar u String vrednosti. Ako se taj znak ne može naći, vraća -1.
lastIndexOf(String s)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
lastIndexOf(char c, int indeks)	Pronalazi i vraća indeks poslednjeg pojavljivanja znaka koji je unet kao parametar u String vrednosti. Pretraživanje se vrši unazad od poslednje pozicije do pozicije koja je uneta u vidu indeksa.
lastIndexOf(String s, int indeks)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
length()	Vraća dužinu String vrednosti.
replace(char c1, char c2)	Pronalazi sva pojavljivanja znakova kojisu jednaki vrednosti promenljive c1 i menja znakom iz promenljive c2.
split(String s)	Deli početni String na više String vrednosti (Vraća niz String vrednosti). Deljenje se vrši na onim mestima gde se naiđe na izraz koji je dat kao argument. ako se npr. kao argument unese prazno mesto (“ ”), podeliće početnu vrednost na pojedinačne reči.
startsWith(String s)	Proverava da li prvi String počinje nizom znakova koji se nalazi u Stringu s i vraća “true” ako je tačno, a “false” ako je netačno.
substrings(int pocetniIndeks)	Vraća deo početne String vrednosti počev od znaka sa indeksom “pocetniIndeks” pa do kraja.
substring(int pocetniIndeks, int krajnjiIndeks)	Vraća deo početne vrednosti počev od znaka sa indeksom “pocetniIndeks” pa do znaka sa indeksom “krajnjiIndeks” ali ne uključujući i taj znak.
toLowerCase()	Pretvara sva velika slova String vrednosti u mala slova, i vraća tako promenjen string.
toUpperCase()	Pretvara sva mala slova u velika, i vraća tako izmenjen string.

trim()	"Odseca" znakove koji predstavljaju prazna mesta sa početka i kraja String vrednosti i vraća tako izmenjen string.

Zadaci za samostalni rad studenata

Zadatak 1. Napraviti klasu ***BrojacReci*** koja ima:

- Statičku metodu koja kao parametar prima tekst i vraća ukupan broj reči u tekstu. Svake dve reči u tekstu su međusobno odvojene jednim praznim mestom.
- Statičku metodu koja prebrojava i vraća broj pojavljivanja reči "lep". tekst je dat u vidu parametra. Svake dve reči u tekstu su međusobno odvojene jednim praznim mestom.
- Statičku metodu koja prebrojava i vraća broj pojavljivanja određene reči u tekstu. I tekst i reč koja se traži su dati u vidu parametra. Potrebno je ne uzimati u obzir da i su u pitanju velika ili mala slova. Takođe, potrebno je uzeti u obzir situaciju kada je tražena reč na kraju rečenice (rečenica se uvek završava tačkom). Primer: U tekstu „Lep dan danas. Danas je lep dan." reč „danas" se pojavljuje dva puta.
- Statičku metodu koja vraća reč koja se u tekstu pojavljuje najveći broj puta. Tekst je dat u vidu parametra.
- Statičku metodu koja vraća reč koja se u tekstu pojavljuje najmanji broj puta. Tekst je dat u vidu parametra.

Napraviti klasu ***TestBrojacReci.*** Za tekst "Danas je lep dan. Bas lep." proveriti i ispisati na ekranu: reč koja se pojavljuje nejmanji broj puta i reč koja se pojavljuje najveći broj puta.

Zadatak 2. Potrebno je kreirati klasu ***OceneStudenata*** koja ima:

- Atribut ocene koji je tipa niz Stringova. Svaki element niza predstavlja po jednog studenta i njegovu ocenu u vidu stringa. Taj string je u formatu „*BROJ-INDEKSA IME PREZIME OCENA*“. Primer: „42/17 Pera Peric 7“.
- Konstruktor koji incijalizuje atribut ocene na određenu vrednost koja je data u vidu parametra tipa String. U okviru ovog stringa se nalaze ocene za više studenata odvojene tačka-zarezom npr. „42/17 Pera Peric 7;43/17 Milos

Milosevic 6“. Potrebno je razdvojiti ovaj string i popuniti elemente niza. Ako je String null, ispisuje se greška na ekranu.

- Metodu koja ispisuje na ekranu podatke o svim studentima.
- Metodu koja ispisuje na ekranu samo podatke o onim studentima koji su pali na ispitu.
- Metodu koja ispisuje na ekranu samo podatke o onim studentima koji su položili ispit.
- Metodu koja ispisuje na ekranu samo podatke o onim studentima koji su fakultet upisali 2017 godine.

Napraviti klasu ***TestOceneStudenata*** koja rkeira jedan objekat klase Ocene Studenata i puni ga vrednostima u formatu gore definisanom. Ispisati na ekranu studente koji su položili ispit, zatim one koji nisu položili, pa na kraju one koji su fakultet upisali 2017. godine.

Zadatak 3. Napraviti klasu ***AnalizatorReciRecenica*** koja ima:

- Atribut tekst koji je tipa String koji ima početnu vrednost “nepoznat”.
- Matodu **getText** koja vraća trenutnu vrednost atributa tekst.
- Metodu **setText** koja postavlja vrednost atributa tekst na novu vrednost. Ovaj tekst je dat u vidu parametra. Dodeljivanje se vrši isključivo ako je novi tekst različit od null, u suprotnom se ispisuje poruka o grešci na ekranu.
- Metodu koja vraća deo teksta bez prvog slova.
- Metodu koja vraća deo teksta bez prva četiri slova.
- Metodu koja vraća deo teksta bez prve reči. Svake dve reči u tekstu su odvojene jednim praznim mestom. Ako tekst ima samo jednu reč, potrebno je vratiti null vrednost.
- Metodu koja vraća deo teksta bez prve rečenice. Sve rečenice se završavaju tačkom. Ako tekst nema nijednu rečenicu, potrebno je vratiti null.
- Metodu koja vraća poslednju reč iz teksta. Svake dve reči u tekstu su odvojene jednim praznim mestom. Ako tekst ima samo jednu reč, potrebno je vratiti ceo tekst.
- Metodu koja vraća samo poslednju rečenicu teksta. Sve rečenice se završavaju tačkom. Ako tekst nema nijednu rečenicu, ili ako ima jednu rečenicu, potrebno je vratiti null.
- Metodu koja vraća deo teksta bez prvog i poslednjeg slova.
- Metodu koja vraća deo teksta od trećeg do pretposlednjeg slova (uključujući treće ali ne uključujući pretposlednje slovo).
- Metodu koja vraća drugu reč iz teksta. Svake dve reči u tekstu su odvojene jednim praznim mestom. Ako tekst ima samo jednu reč, potrebno je vratiti null.
- Metodu koja proverava da li se novi tekst nalazi na početku teksta (tj. da li tekst počinje izrazom datim u novom tekstu). Ako počinje, metoda vraća FALSE. Novi tekst je dat u vidu parametra.
- Metodu koja proverava da li se novi tekst nalazi na kraju teksta (tj. da li se neki tekst završava izrazom datim u novom tekstu). Ako se završava, metoda vraća TRUE, u suprotnom FALSE. Ako je novi tekst jednak null metoda vraća FALSE. Novi tekst je dat u vidu parametra.

Napraviti klasu ***TestAnalizatorReciiRecenica*** koja kreira jedan objekat klase *AnalizatorReciiRecenica* i unosi u njega tekst: “Sunce sija. Nebo je plavo. Nema nijednog oblaka.”. Ispisati na ekranu poslednju reč i rečenicu iz teksta.

Nizovi objekata

Kod nizova koje smo objasnili na početku vežbe je uvedeno prećutno ograničenje – elementi niza su ouvek bili prostog tipa podataka. U Javi je moguće napraviti niz čiji elementi nisu prostog tipa, već je **svaki element po jedan objekat**. Rad sa nizovima objekta je skoro pa potpuno isti kao i sa običnim nizovima. Jedine razlike se svode na to da se svaki pojedinačni element tretira malo drugačije jer je u pitanju objekat.

Deklaracija niza objekata je potpuno ista kao i za običan niz:

```
tip_podatka[] nazivPromenljive;
```

Sa obzirom na to da su elementi niza objekti neke klase, tip podatka je upravo naziv te klase:

```
nazivKlase[] nazivPromenljive;
```

Inicijalizacija niza je u ovom slučaju potrebna, i vrši se na isti način kao i za običan niz:

```
nazivPromenljive = new NazivKlase[ceo_broj];
```

Prva razlika u odnsu na obične nizove se može videti pri inicijalizaciji. Kod inicijalizacije niza čiji su elementi prostog tipa, sama inicijalizacija niza rezerviše sav potreban memorijski prostor za svaki element niza i sa nizom se može odmah raditi. Kada je u pitanju niz objekata, inicijalizacija niza samo rezerviše memorijski prostor za pokazivače na objekte, ali **ne inicijalizuje same objekte.** Drugim rečima, **posle inicijalizacije niza objekata, svaki element ima vrednost „null“ i sa njim se može raditi dok se ne inicijalizuje**. U nekim situacijama je potrebno izvršiti pojedinačnu inicijalizaciju niza odmah na početku, a nekad se samo unose ceć inicijalizovani objekti na odgovarajuća mesta u nizu (u tom slučaju se smatra da je mesto u nizu „prazno“ ako ima „null“ vrednost).

Pristup elementima niza se vrši preko indeksa, a maksimalni apacitet niza se dobija pozivanjem kamande „length“:

```
nazivPromenljive[indekss]

nazivPromenljive.length
```

Druga razlika u odnosu na obične nizove se sastoji u tome što se nad elementima niza objekata mogu direktno pozivati metode. Svaki element je po jedan objekat, pa se pozivi njegovih metoda mogu izvršiti ovako:

```
nazivPromenljive[indeks].nazivMetode(…argumenti…);
```

Primer 4. Napraviti javnu klasu ***TemperaturaMesta*** koja ima:

- Privatni atribut *naziv* koji predstavlja naziv mesta. Početna vrednost za ovaj atribut je „nepoznat“.
- Privatni atribut *temperatura* koji predstavlja dnevnu temperaturu za to mesto.
- Odgovarajuće javne *get* i *set* metode za ova dva atributa.

Napraviti javnu klasu ***DnevnaPrognoza*** koja ima:

- Privatni atribut *temperature* koji predstavlja niz objekata ***TemperaturaMesta*.**
- Javni konstruktor koji kao parametar prima broj koji predstavlja maksimalan broj mesta na koje se prognoza odnosi. Ako je uneti broj veći od nule, potrebno je inicijalizovati atribut *temperature* na taj kapacitet. Ako je uneti broj nula ili manji od nule, kapacitet se postavlja na 10 i ispisuje se poruka o grešci na ekranu. U svakom slučaju je potrebno inicijalizovati svaki element niza.
- Javnu metodu *imaSlobodnihMesta* koja vraća TRUE ako u nizu ima slobodnih mesta za unos temperature, a FALSE ako nema. Mesto u nizu je slobodno ako umesto naziva mesta na koje se odnosi prognoza stoji String “nepoznat”.
- Javnu metodu *unesi* koja kao ulazni parametar prima naziv mesta i temperaturu i unosi te podatke na prvo slobodno mesto u nizu. Ako u nizu nema slobodnih mesta, metoda ispisuje poruku o tome na ekranu.
- Javnu metodu *izbaci* koja kao ulazni parametar prima naziv mesta i izbacuje podatke o tom mestu i njegovoj temperaturi iz niza. Izbacivanje se vrši tako što se kao naziv mesta postavlja reč “nepoznat” a temperatura dobija vrednost 0. Ako u nizu nema mesta sa unetim nazivom, ništa se ne dešava.
- Javnu metodi *ispisi* koja na ekranu ispisuje celokupnu dnevnu prognozu.

Napraviti klasu ***TestDnevnaPrognoza*** koja kreira jedan objekat klase ***DnevnaPrognoza*** kapaciteta 4 i u njega unosi podatke o temperaturama za četiri mesta: “*Beograd” (17C), “Niš” (22C), “Novi Sad” (13C)* i *“Gnjilane” (15C).* Ispisati podatke o svim gradovima i temperaturama.

```
package t;

public class TemperaturaMesta {
	private String naziv  = "nepoznat";
	private int temperatura;

	public String getNaziv() {
		return naziv;
	}

	public void setNaziv(String naziv) {
		this.naziv = naziv;
	}

	public int getTemperatura() {
		return temperatura;
	}

	public void setTemperatura(int temperatura) {
		this.temperatura = temperatura;
	}

}
```

```java
package t;

public class DnevnaPrognoza {
	private TemperaturaMesta[] temperature;

	public DnevnaPrognoza(int brojMesta) {
		if(brojMesta>0)
			temperature = new TemperaturaMesta[brojMesta];
		else {
			temperature = new TemperaturaMesta[10];
			System.out.println("Greska!");
		}

		//pojedinacna inicijalizacija svakog elementa niza
		for(int i = 0; i<temperature.length; i++)
				temperature[i] = new TemperaturaMesta();

	}

	public boolean imaSlobodnihMesta() {
		for(int i = 0; i < temperature.length; i++)
			if(temperature[i].getNaziv().equals("nepoznat"))
			return true;



		return false;
	}

	public void unesi(String naziv, int temperatura) {
		if(!imaSlobodnihMesta())
			System.out.println("Nema slobodnih mesta!");
		else
			for(int i = 0; i<temperature.length; i++)
				if(temperature[i].getNaziv().equals("nepoznat")) {
					temperature[i].setNaziv(naziv);
					temperature[i].setTemperatura(temperatura);
					break;
				}
	}

	public void izbaci(String naziv) {
		for(int i=0; i<temperature.length; i++)
			if(temperature[i].getNaziv().equals(naziv)) {
				temperature[i].setNaziv("nepoznat");
				temperature[i].setTemperatura(0);
				break;
			}

	}

	public void ispisi() {
		for(int i =0; i<temperature.length; i++)
			System.out.println("Mesto:
	\t\t"+temperature[i].getNaziv()+"\nTemperatura:
	\t"+temperature[i].getTemperatura());

	}
}
```

```
package t;

public class TestDnevnaPrognoza{
	public static void main(String[] args) {

	DnevnaPrognoza d = new DnevnaPrognoza(4);
	d.unesi("Beograd", 17);
	d.unesi("Nis", 22);
	d.unesi("Gnjilane", 24);
	d.unesi("Mitrovica", 45);
	d.ispisi();
	}
}
```

Konstruktor klase **DnevnaPrognoza** inicijalizuje atribut „temperature“ koji predstavlja niz objekata klase **TmperaturaMesta**. Pored inicijalizacije niza, ovaj konstruktor inicijalizuje svaki element niza (objekat) pojedinačno. To se vrši pozivanjem konstruktora klase **TemperaturaMesta** za svaki element. Tek posle ove inicijalizacije se elementi niza mogu koristiti. Metoda *imaSlobodnihMesta* proverava da li je neko mesto u nizu slobodno na taj način što poredi da li atribut „naziv“ objekta koji se nalazi na tom mestu u nizu ima vrednost „nepoznat“. Tu se može videti da se vrednost njegovog atributa „naziv“ dobija pozivanjem metode *getNaziv* (*temperature[i].getNaziv()*). Sa obzirom da ova metoda vraća naziv (String) potrebno ga je uporediti sa vrednošću „nepoznat“ što se vrši pozivanjem *equals* metode. Ceo logički izraz u okviru IF komande zbog toga izgleda relativno složeno pa glasi: *temperature[i].getNaziv().equals(“nepoznat”)*.

Zaključak

U Nišu _____

Potpis _____