# Objektno orijentisano programiranje
## vežbe

Staša Vujičić Stanković

# Scanner - nastavak

- Ulaz može biti *fajl* ili *tok*, uključujući i standardni ulazni tok System.in.
- Ulaz se deli na tokene koristeći kao podrazumevani delimiter beline (' ', '\n', '\t', ...).
- Dobijeni tokeni se mogu konvertovati u vrednosti različitih tipova koristeći razne next... metode.

```
Scanner skener = new Scanner(System.in);
int a = skener.nextInt();
int b = skener.nextDouble();
```

# Najčešće korišćeni konstruktori

- Najčešće korišćeni konstruktori:

**public Scanner(InputStream source)**
**public Scanner(File source)**
**public Scanner(String source)**

# Čitanje celog broja sa standardnog ulaza

- Čitanje celog broja sa standardnog ulaza (System.in):

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

# Čitanje celih brojeva tipa long iz datoteke

- Čitanje celih brojeva tipa long iz datoteke myNumbers, sve dok ih ima
(umesto long može se staviti i neki drugi tip)

```
Scanner sc = new Scanner(new File("myNumbers"));
while (sc.hasNextLong()) {
        long aLong = sc.nextLong();
}
```

# Čitanje iz stringa

- Može se čitati i iz stringa:

```
String input = "1 7 java jdk";
Scanner s = new Scanner(input);
System.out.println(s.nextInt()); // 1
System.out.println(s.nextInt()); // 7
System.out.println(s.next()); // java
System.out.println(s.next()); // jdk
```

# Metodi next*() i hasNext*()

- **next*()** i **hasNext*()** metodi postoje za većinu primitivnih tipova (int, double, long, boolean,...).

- Preskaču karaktere sa ulaza koji odgovaraju delimiterima i pokušavaju da vrate naredni token koji odgovara tipu.

- Metodi **findInLine(String)** i **findWithinHorizon(String, int)** traže zadati obrazac ne uzimajući u obzir delimitere.

# Neodgovarajući token

- Kada se izbaci izuzetak tipa ***InputMismatchException,*** ne prosleđuje se token koji je izazvao izuzetak. Token se može dobiti ili preskočiti primenom nekog drugog metoda.

- U zavisnosti od toga šta je delimiter, mogu biti vraćeni i prazni tokeni.

- Metodom **close()** zatvara se skaner. Kada se Scanner zatvori, zatvoriće se ulazni tok ukoliko isti implementira interfejs *Closeable*.

# Redosled primera

1. ScannerStandardniUlaz.java
2. ScannerUlazDatoteka.java
3. ScannerUlazString.java

# Neke razlike u odnosu na C i C++

- U Javi se svim objektima se pristupa preko **ručke (handle).**

```
String s;          /* s je ručka za pristup objektu tipa String i
                      ima vrednost null, tj. ne referiše ni na jedan
                      objekat klase String */

s = new            /* s je sada ručka (referenca) na konkretan
String("ab");         objekat klase String */

String t = s;      /* s i t su ručke koje se vezuju za isti objekat
                      klase String */
```

# Nizovi

- Deklaracija niza:

  **int[] a;** ili **int a[];**
- Deklaracijom se specificira da je namena promenljive a da referiše na niz celih brojeva, a ne da čuva jednu vrednost tipa int.
- Deklaracijom se ne alocira memorija za sam niz. Veličina niza se ne navodi u deklaraciji!

# Nizovi

- Nakon deklaracije nizovske promenljive, moguće je definisati niz na koji će ona referisati:

```
a = new int[10];
/* ovom naredbom se kreira niz koji može da čuva
10 vrednosti tipa int, a referenca na niz smešta se u
promenljivu a.
Referenca je zapravo lokacija niza u memoriji*/
```

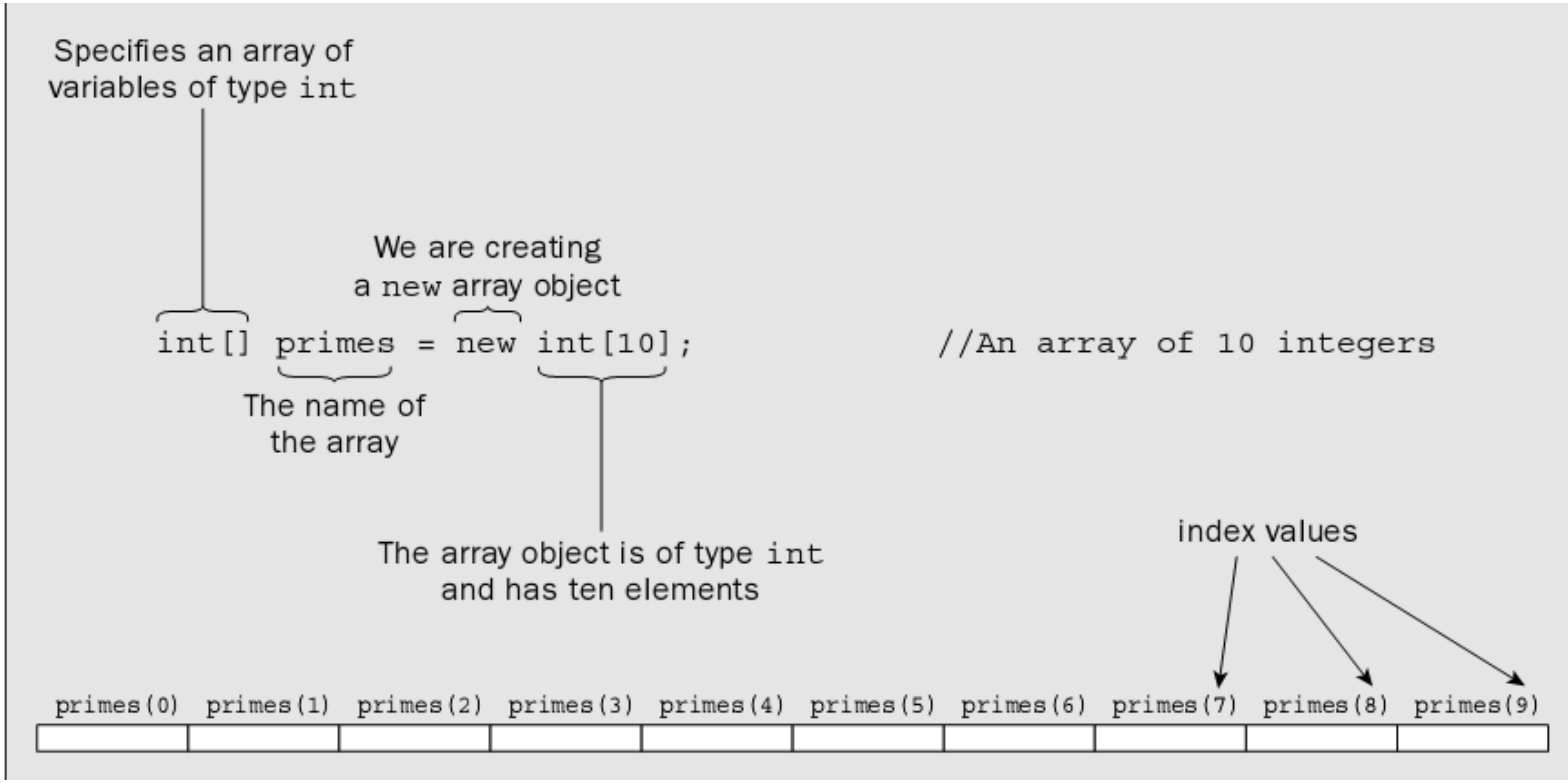
# Nizovi

- Moguće je u jednoj naredbi deklarisati nizovsku promenljivu i definisati niz na koji će ona referisati:

```
int[] a = new int [10];
```

/* upotrebom operatora **new** alocira se memorija za niz i to 4*10=40 bajtova, plus 4 bajta za promenljivu a koja čuva referencu na niz.*/

# Nizovi


Specifies an array of variables of type int
We are creating a new array object
int[] primes = new int[10]; //An array of 10 integers
The name of the array
The array object is of type int and has ten elements
index values
primes(0) primes(1) primes(2) primes(3) primes(4) primes(5) primes(6) primes(7) primes(8) primes(9)

# Nizovi

- Kreiranjem niza upotrebom operatora new, elementi niza se automatski inicijalizuju na podrazumevanu vrednost, koja zavisi od tipa elemenata:
  - numeričke vrednosti – 0
  - boolean – false
  - char – '\u0000'
  - klasni tip - null

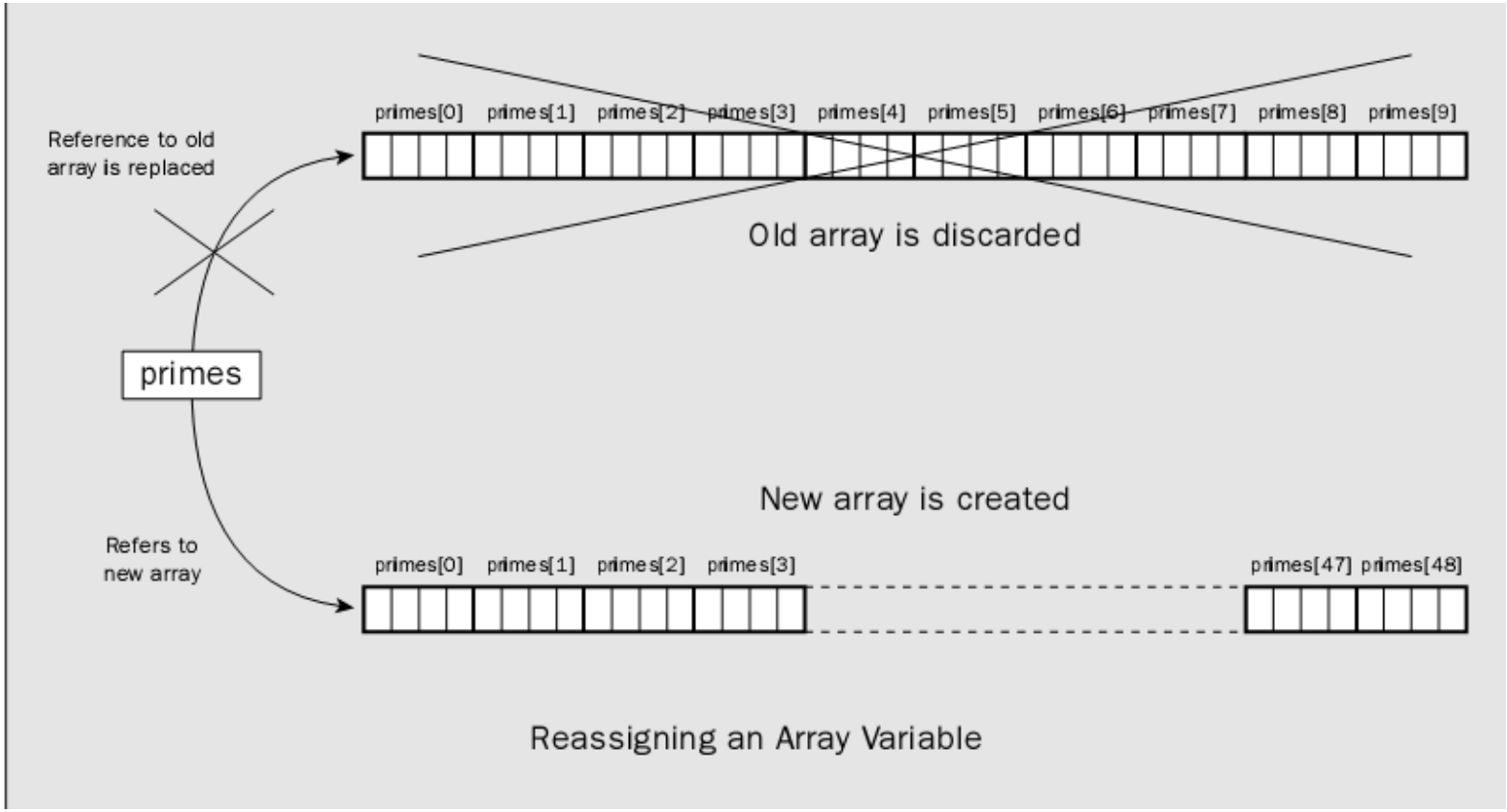
# Nizovi

- Moguće je jednu nizovsku promenljivu koristiti za referisanje različitih *nizova* u različitim delovima programa. Npr:

```
int a[] = new int[10];
  /* a referiše na niz od 10 celih brojeva */
a = new int[50];
  /* sada a referiše na novi niz od 50 celih brojeva */
```

- Međutim, svi nizovi na koje promenljiva može da referiše u programu moraju da sadrže elemente onog tipa koji je specificiran prilikom njene deklaracije. U gornjem primeru a može da referiše samo na celobrojne nizove.

# Nizovi


primes[0] primes[1] primes[2] primes[3] primes[4] primes[5] primes[6] primes[7] primes[8] primes[9]
Reference to old array is replaced
Old array is discarded
primes
New array is created
Refers to new array
primes[0] primes[1] primes[2] primes[3]
primes[47] primes[48]
Reassigning an Array Variable

# Nizovi

- Prilikom deklaracije, moguće je inicijalizovati elemente niza pri čemu će veličina niza biti određena kao broj vrednosti za inicijalizaciju.

```
int[] a = {2,3,4}; /* niz od 3 elementa */
```

- Ako hoćemo sve elemente niza da inicijalizujemo na istu vrednost:

```
import java.util.Arrays.*;
fill(a, 10);
```

# Nizovi

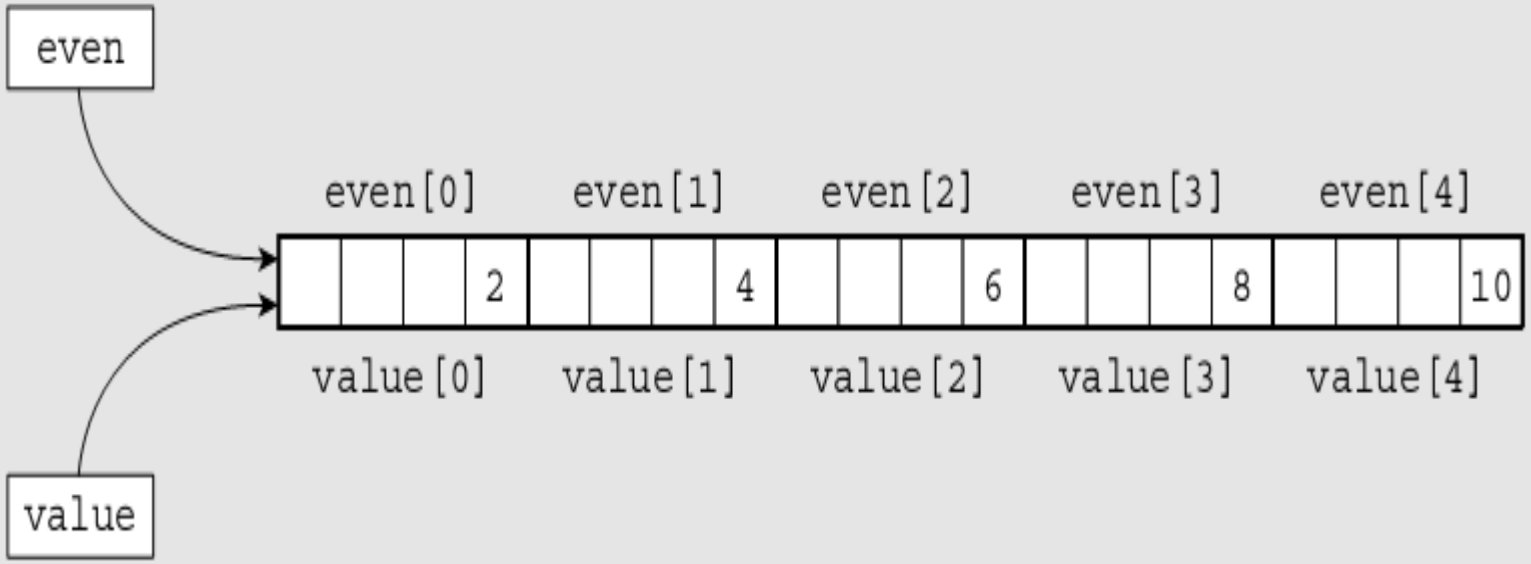
- Moguće je kreirati više ručki na jedan isti niz, što je prikazano na narednom grafiku.
- int[] parni = {2, 4, 6, 8, 10};

  int[] novi = parni;

  // obe promenljive referišu na isti niz

# Nizovi

```
long[] even = (2L, 4L, 6L, 8L, 10L);
```




```
long[] value = even;
```

# Nizovi

- Inicijalizacija elemenata niza
  može se izvršiti upotrebom for ciklusa:

```
double[] samples = new double[50];
for( int i = 0; i < samples.length; i++)
        samples[i] = 100.0;
```

# Nizovi

- Može se koristiti i oblik for ciklusa za kolekcije umesto klasične numeričke forme kada je potrebno pristupiti svim elementima niza.

```
double average = 0.0;
for(double value : samples)
        average += value;
average /= samples.length;
```

- Ovakav oblik for ciklusa omogućuje **samo iteraciju** kroz sve elemente niza, ne može se upotrebiti za pristup elementima niza kako bi se promenila njihova vrednost. U tom slučaju treba koristiti numeričku formu for ciklusa.

# Nizovi nizova

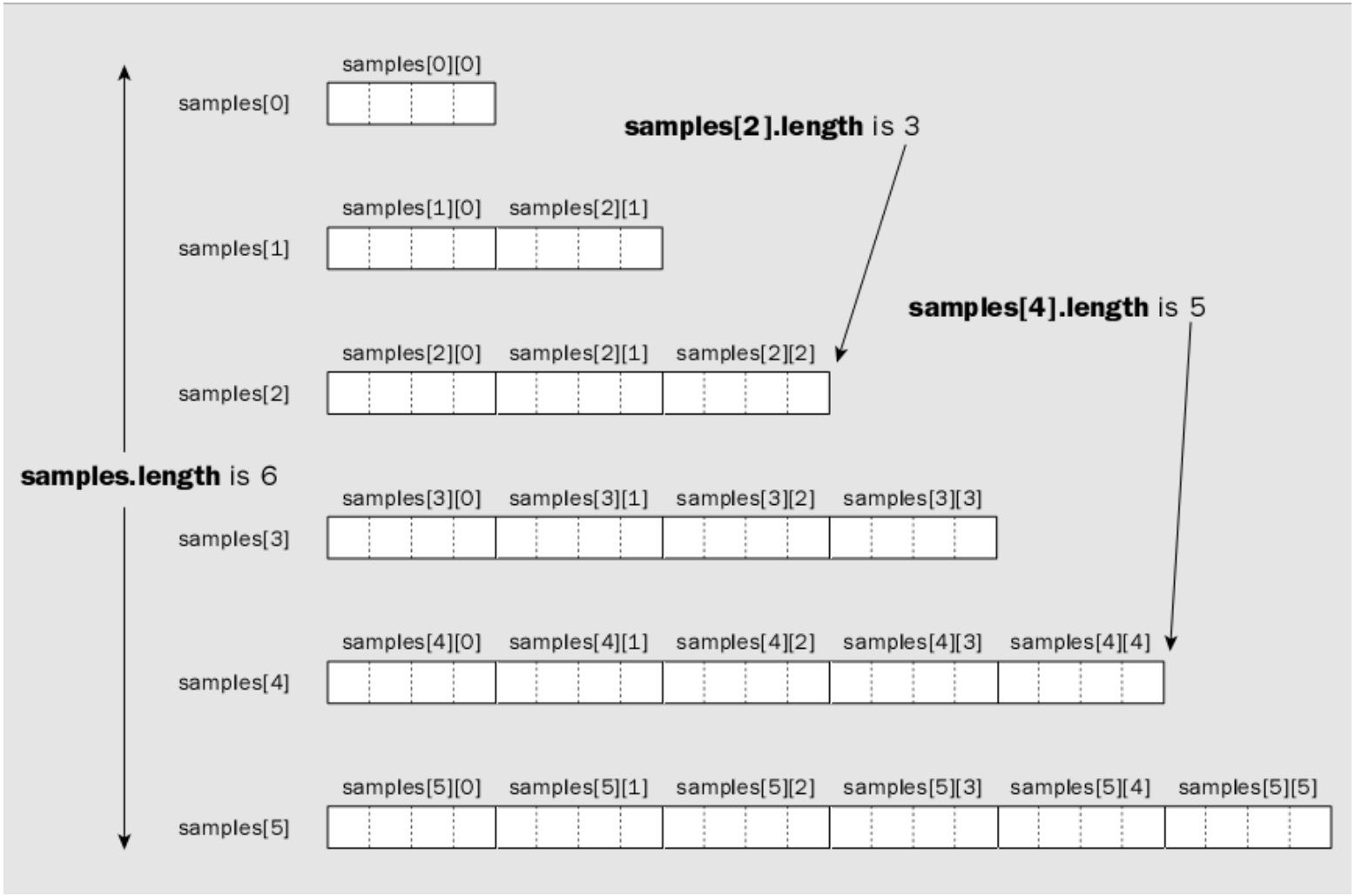
- **int[][] niz = new int[3][5];**
  - Niz od 3 elementa koji su nizovi od 5 elemenata
- Možemo da pravimo i nizove nizova različitih dužina:

```
float[][] samples;
samples = new float[6][];
```

/* promenljiva samples sada referiše na niz od 6 elemenata a svaki od njih može da čuva *referencu* na jednodimenzioni niz */

```
samples[1] = new float[4];
samples[3] = new float[10];
for(int i = 0; i < samples.length; i++)
samples[i] = new float[i + 1];
```

# Nizovi nizova



# Nizovi

- Svaki niz ima polje **length** koje nije metod nego atribut.
- **samples.length = 6**

  **samples[1].length = 2**
- Možemo praviti i nizove ostalih tipova podataka, npr. char...

```
char[] message = new char[50];
  /* svaki element zauzima 2 bajta, jer se
  koristi UNICODE */
java.util.Arrays.fill(message, ' ');
char[] v = { 'a', 'e', 'i', 'o', 'u'};
```

# Zadatak 1.

- Učitavamo broj elemenata niza i njegove elemente i računamo zbir elemenata niza i njihovu aritmetičku sredinu.

# Zadatak 2.

- Sa standardnog ulaza učitavamo elemente niza sve dok se ne unese 0, i računamo zbir elemenata niza i njihovu aritmetičku sredinu.

  Pretpostavlja se da niz neće imati više od 100 elemenata.

# STRINGOVI

- **String literali** su sekvence karaktera između dvostrukih navodnika:

  “Ovo je string literal.”

- String myString = “Ovo je moj string.”;

  /* ovim se deklariše promenljiva myString tipa String i inicijalizuje se referencom na objekat klase String koji sadrži string “Ovo je moj string.” */

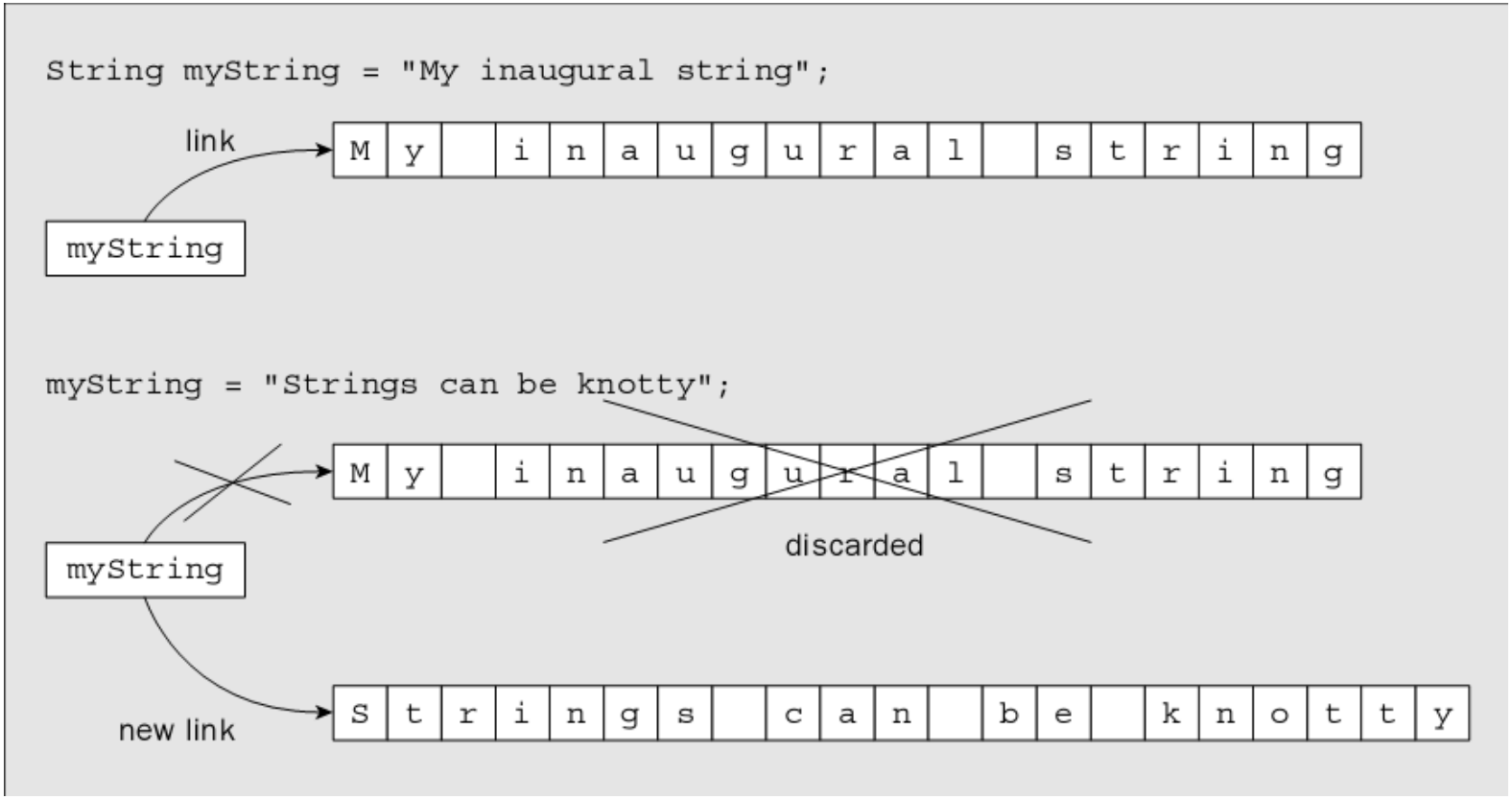
# Stringovi

- U istu promenljivu tipa String može se sačuvati referenca na neki drugi objekat klase String.
- Sam objekat klase String razlikuje se od promenljive kojom se referiše.
  Dakle, u gornjem primeru promenljiva mySting sadrži referencu na objekat klase String,
  a ne sam objekat.

# STRINGOVI

- String objekti imaju svojstvo da ne mogu biti promenjeni (immutable). To znači da se ne može promeniti string koji neki objekat klase String predstavlja.
- Kada se izvršava naredba nad postojećim String objektom kao rezultat se uvek kreira novi objekat klase String.
- Kada se promeni string na koji referiše String promenljiva, odbacuje se referenca na stari string i zamenjuje referencom na novi.

# STRINGOVI


String myString = "My inaugural string";
link
M y i n a u g u r a l s t r i n g
myString
myString = "Strings can be knotty";
M y i n a u g u r a l s t r i n g
discarded
myString
new link
S t r i n g s c a n b e k n o t t y

# Stringovi

- **String nulaString = null;**

  /* String promenljiva koja ne referiše ni na jedan string */
- Kada želimo da odbacimo String objekat na koji trenutno referiše String promenljiva dodelimo joj vrednost **null**. Null će zameniti referencu koja je sadržaj String promenljive, tako da promenljiva neće ukazivati ni na šta.

# STRINGOVI

- Konkatenacija stringova:

  **String mojString =**
  **“Pocetak mog stringa,” + “kraj mog stringa”;**

- Operator + generiše potpuno novi objekat klase String koji je nezavisan od dva polazna i referenca na kreirani objekat smešta se u promenljivu mojString.

- **String dan = “13. “;**
  **String mesec = “Maj”;**
  **String Datum = dan + mesec;** // rezultat je “13. Maj”

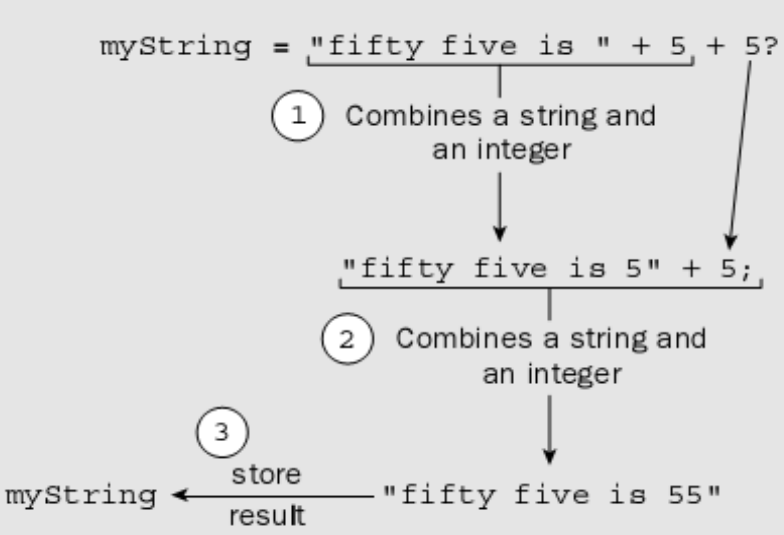
# STRINGOVI

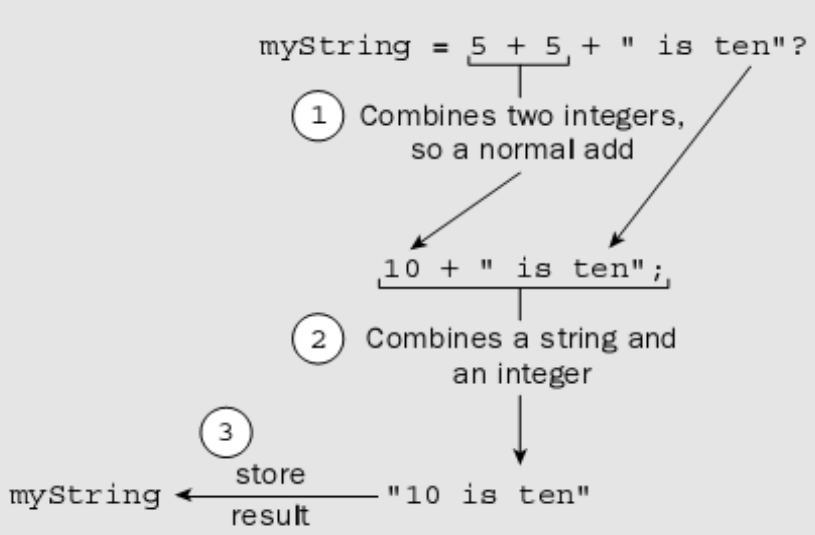
- Može se koristi i **operator +=** za konkatenaciju stringova.

```
String recenica = “Ovo je ”;
recenica += “ moja recenica.”;
/* recenica sada referiše na novi String objekat,
   što ne menja string “Ovo je ” */
```

- Operator + je **levo asocijativan**.

```
String myString;
myString = “fifty five is “ + 5 + 5;
  // “fifty five is 55”
myString = 5 + 5 + “is ten”; // “10 is ten”
```




The associativity of the + operator accounts for the differences between these two statements

# STRINGOVI

- Automatska konverzija primitivnih tipova u String je omogućena zahvaljujući statičkom metodu **toString()** standardnih klasa koje se odnose na primitivne tipove (Byte, Short, Integer, Long, Float, Double, Boolean, Character).
- Kad god je jedan operand operatora + vrednost primitivnog tipa a drugi objekat klase String, kompajler prosleđuje vrednost primitivnog tipa kao argument metodu toString() koji vraća String ekvivalentan polaznoj vrednosti primitivnog tipa.

# STRINGOVI

- String klasa sadrži metod **valueOf()** koji kreira String objekat od vrednosti proizvoljnog primitivnog tipa:

  **String doubleString = String.valueOf(3.1415926);**

- Izraz **string1 == string2** proverava da li obe stringovne promenljive **referišu** na isti objekat. Ako referišu na nezavisne objekte, rezultat izraza je false, bez obzira da li možda ta dva objekta sadrže isti string. Dakle, ovakav izraz ne poredi same stringove, već poredi reference na stringove!

# STRINGOVI

- Da bismo poredili dva stringa na jednakost koristimo metod **equals()**.
- Za poređenje dva stringa na jednakost, ali ignorišući veličinu slova koristimo metod **equalsIgnoreCase()**.
- To su dva metoda koje primenjujemo na jedan objekat uz pomoć operatora ".", a argument je ili string ili stringovna promenljiva.

**string1.equals(string2);**

# STRINGOVI

- Provera da li string počinje drugim stringom: **prviString.startsWith(String)**
- Provera da li se string završava drugim stringom: **prviString.endsWith(String)**
- Da bi dobili dužinu stringa pozivamo metod **length()** – razlika u odnosu na nizove gde je atribut.

# STRINGOVI

- Leksikografsko poređenje stringova vrši se pozivom metoda **compareTo(String)** – on poredi String objekat nad kojim je pozvan sa String objektom koji mu je prosleđen kao argument. Povratna vrednost može biti:
  - < 0 String objekat je manji od argumenta
  - = 0 String objekat jednak argumentu
  - > 0 String objekat je veći od argumenta

# Stringovi

- **substring(int)** – vraća podstring datog stringa koji počinje od indeksa zadatog kao argument.
- **substring(int, int)** – vraća podstring datog stringa od pozicije zadate prvim argumentom do pozicije zadate drugim argumentom.
- **replace(char, char)** – u stringu svaku pojavu prvog karaktera menja drugim karakterom.
- **trim()** – briše beline sa početka i kraja stringa

# STRINGOVI

- Pretraživanje stringova:
  - **indexOf(int ch)**, **indexOf(int ch, int fromIndex)**
  - **indexOf(String str)**, **indexOf(String str, int fromIndex)**
  - **lastIndexOf(int ch)**, **lastIndexOf(int ch, int fromIndex)**
  - **lastIndexOf(String str)**, **lastIndexOf(String str, int fromIndex)**
- Svi metodi vraćaju traženi indeks ili -1 ako odgovarajući objekat nije pronađen

# STRINGOVI

- Kreiranje nizova karaktera od objekata klase String:

  **String text = “Pada kiša”;**

  **char[] textArray = text.toCharArray()**
  **// kreira se niz iz stringa**

- Kreiranje objekata klase String od nizova karaktera:

  **char[] textArray = { ‘P’, ‘a’, ‘d’, ‘a’, ‘ ‘, ‘k’, ‘i’, ‘š’,’a’};**

  **String text = String.copyValueOf(textArray);**

  **String text = new String(textArray);**
  /* poziv konstruktora klase String čime se kreira novi objekat ove klase koji sadrži string sastavljen od karaktera datog niza */

# ZADATAK 1.

```
public class PoredjenjeStringova {
    /*
    * Razlika izmedju poredjenja dve stringovne promenjljive koje pokazuju
    * na dva stringa koja imaju isti sadrzaj, i poredjenja
    * dve stringovne promenjive koje pokazuju na isti string.
    */
    public static void main(String[] args) {
            String string1 = "Dva ";
            String string2 = "studenta";
            String string3 = "Dva studenta";
            // Postavljamo da string1 i string3 referisu
            // na razlicite stringove koji su identicni
            string1 += string2;

            System.out.println("Test br 1");
            System.out.println("string3 je: " + string3);
            System.out.println("string1 je: " + string1);
```

```
// Testiramo da li je u pitanju isti string
        if(string1 == string3)
                System.out.println("string1 == string3 je tacno -" +
                " string1 i string3 pokazuju na isti string");
        else
                System.out.println("string1 == string3 nije tacno -" +
                " string1 i string3 ne pokazuju na isti string");
// Postavljamo da string1 i string3 referisu na isti string
        string3 = string1;

        System.out.println("\nTest br 2");
        System.out.println("string3 je: " + string3);
        System.out.println("string1 je: " + string1);
        if(string1 == string3) // Da li su identicni?
                System.out.println("string1 == string3 je tacno -" +
                " string1 i string3 pokazuju na isti string");
        else
                System.out.println("string1 == string3 nije tacno -" +
                " string1 i string3 ne pokazuju na isti string");
    }
}
```

# ZADATAK 2.

```
public class PoredjenjeStringova2 {
    /*
    * Poredjenje dve stringovne promenljive na jednakost
    sadrzaja.
    */
    public static void main(String[] args) {
            String string1 = "Dva ";
            String string2 = "studenta";
            String string3 = "Dva studenta";
            // Postavljamo da string1 i string3 referisu
            // na razlicite stringove koji su identicni
            string1 += string2;

            System.out.println("Test br 1");
            System.out.println("string3 je: " + string3);
            System.out.println("string1 je: " + string1);
```

```
// Testiramo stringove na jednakost sadrzaja
if(string1.equals(string3)) {
        System.out.println("string1.equals(string3) je
        tacno - " +  “stringovi su jednaki.");
} else {
        System.out.println("string1.equals(string3) je
        netacno - " + " stringovi nisu jednaki.");
}
// Sada postavljamo string1 i string3 da referisu
// na stringove koji se razlikuju samo u velicini slova
string3 = "DVA studenta";

System.out.println("\nTest br 2");
System.out.println("string3 je: " + string3);
System.out.println("string1 je: " + string1);
```

```
// Testiramo stringove na jednakost sadrzaja
if(string1.equals(string3)) {

System.out.println("string1.equals(string3) je
        tacno - " + " stringovi su jednaki.");
} else {

System.out.println("string1.equals(string3) je
        netacno -" + " stringovi nisu jednaki.");
}
```

```
        // Testiramo stringove na jednakost sadrzaja
        // bez obaziranja na velicinu slova
        if(string1.equalsIgnoreCase(string3)) {

        System.out.println("string1.equalsIgnore
        Case(string3) je tacno - " + " stringovi   su
jednaki ako zanemarimo velicinu slova.");
        } else {

        System.out.println("string1.equalsIgnore
        Case(string3) je netacno -" + " stringovi
        su razliciti.");
        }
}
}
```

# ZADATAK 3.

```
public class BrojanjeReci {
    /*
     * Brojimo pojavljivanja reci the i and u zadatom tekstu
     */
    public static void main(String[] args) {
            // Tekst koji analiziramo
            String text = "To be or not to be, that is the question;"
                        + " Whether 'tis nobler in the mind to suffer"
                        + " the slings and arrows of outrageous fortune,"
                        + " or to take arms against a sea of troubles,"
                        + " and by opposing end them?";

int andBr = 0;
int theBr = 0;
int indeks = -1;
String andStr = "and";
String theStr = "the";
```

```
// Trazimo pojavljivanje reci "and"
indeks = text.indexOf(andStr); // Nalazimo prvu pojavu reci "and"
while(indeks >= 0) {
++andBr;
indeks += andStr.length(); // Postavljamo se na mesto posle
                           // pojave tekuceg "and"
indeks = text.indexOf(andStr, indeks);
}

// Trazimo unazad pojavljivanje reci "the"
indeks = text.lastIndexOf(theStr);  // Nalazimo poslednju pojavu reci "the"
while(indeks >= 0) {
++theBr;
indeks -= theStr.length(); // Postavljamo se na mesto pre
                           // pojave tekuceg "and"
indeks = text.lastIndexOf(theStr, indeks);
}
System.out.println("Zadati tekst sadrzi " + andBr + " pojave reci 'and'\n"
+ "i " + theBr + " pojava reci 'the'.");
}
}
```