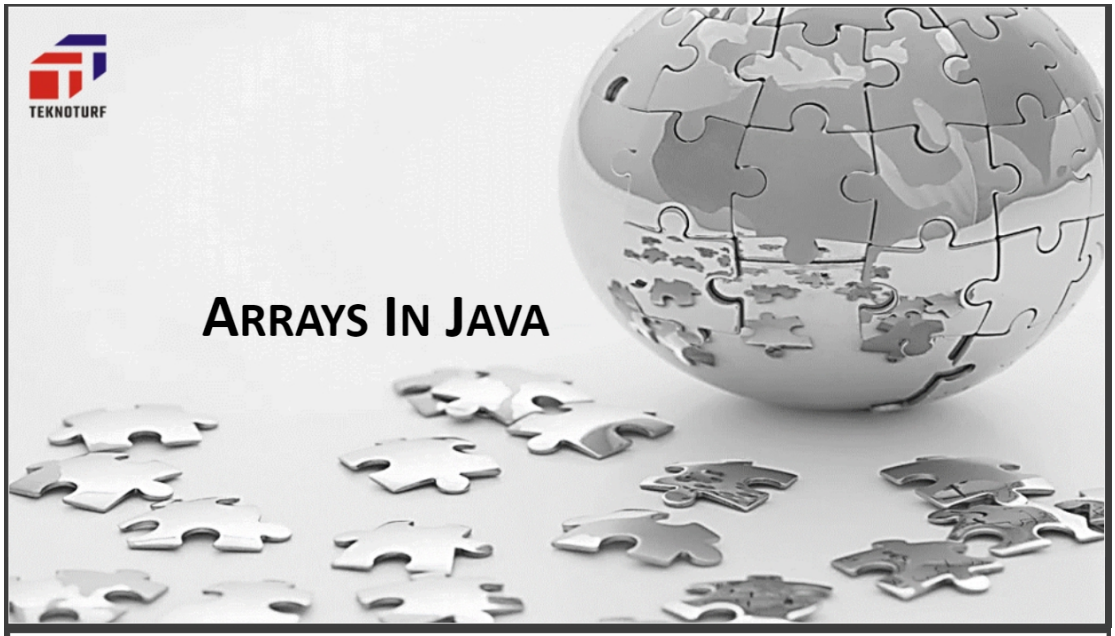
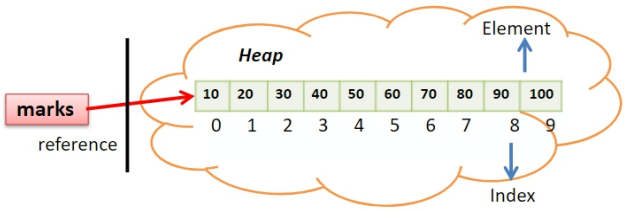


Arrays

TEKNOTURF

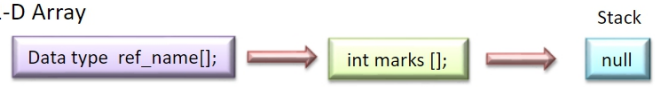
- Array is an ordered collection that stores elements of the same type.
- Array elements are stored contiguously.
- Array in java is index based; the first element of the array is stored at 0 index.
- Array is a Reference data type.



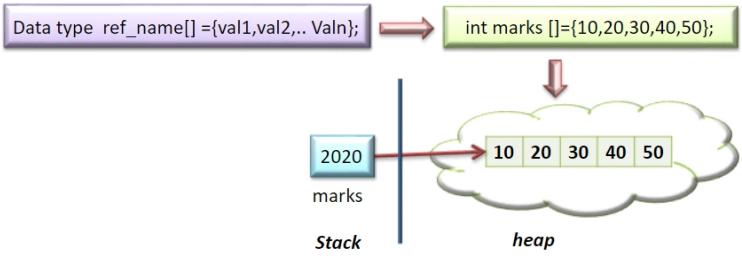
One dimensional Array

TEKNOTURF

Declaring 1-D Array

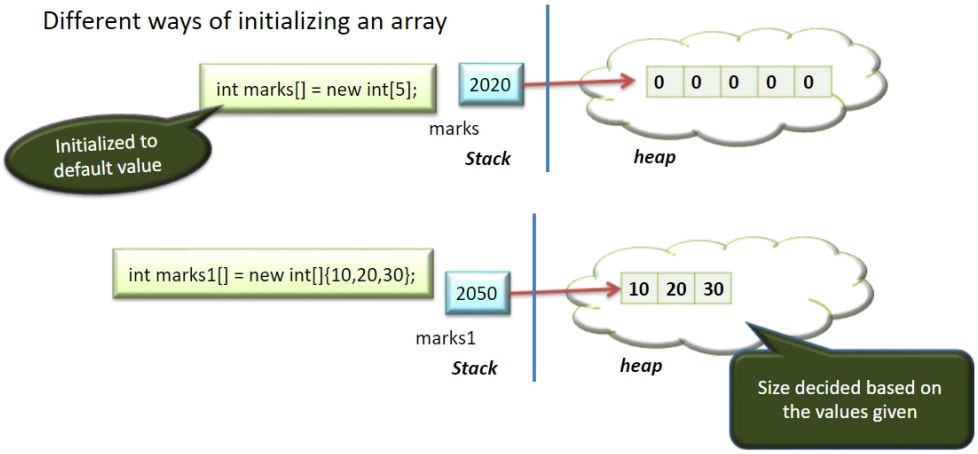


Initializing Array



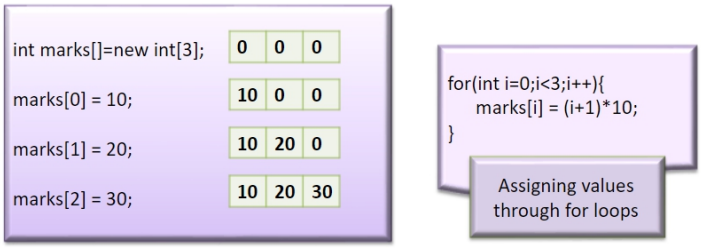
One dimensional Array

Different ways of initializing an array



One dimensional Array

Assigning Elements in array




```
for(int i=0;i<3;i++){
    marks[i] = (i+1)*10;
}
```

Assigning values through for loops

One dimensional Array

Length property

- Gets the length of the array ie. The number of elements in an array

<array_name>.length

```
int marks[]=new int[3];
for(int i=0;i<marks.length;i++){
    marks[i] = (i+1)*10;
}
```

Iterates the loop for 3 times

One dimensional Array

TEKNOTURF

Accessing elements from array

```
for(int i=0;i<marks.length;i++){
        System.out.println(marks[i]);
}
```

(OR)

Enhanced for loop introduced in Java 5

```
for (data_type variable: array_name)
```

```
for(int m : marks) {
    System.out.println(m);
}
```

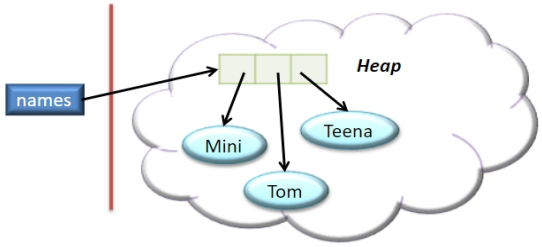
One dimensional Array

TEKNOTURF

Creating a String Array

```
String names[] = new String[3];
names[0] = new String("Mini");
names[1] = "Tom";
names[2] = new String("Teena");
```

Arrays for reference data types hold the address of the array elements



Array Of Objects

TEKNOTURF

Array can also store objects

Example

```
Employee empList[]=new Employee[3];
```

The array variable empList[] does not hold an array of Employee objects, instead it holds an array of Employee reference variables.

To make these variables hold the object, we have to create the object as

```
empList[0]=new Employee();
empList[1]=new Employee();
empList[2]=new Employee();
```

Array Of Objects - Example

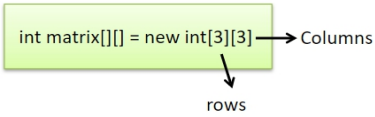
```
class Employee {
      private int empId;
      private String name;
      //Write constructor, getters and setters
}

public class ObjectArrayDemo {
      public static void main(String[] args) {
                Employee empList[]=new Employee[3];
                Scanner sc=new Scanner(System.in);
                for(int index=0;index<empList.length;index++){
                        System.out.print("Enter the id : ");
                        int id=sc.nextInt();
                        System.out.print("Enter the name : ");
                        String name=sc.next();
                        empList[index]=new Employee(id, name);
                }
      for(int index=0;index<empList.length;index++){
                System.out.println("Index is "+index);
                System.out.println("ID is  "+
                        empList[index].getEmpId() +
                        "Name is " +
                        empList[index].getName());
      }
   }
}
```

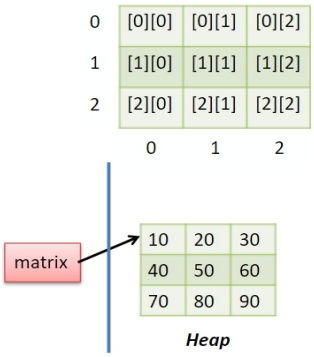
Output

```
Enter the id : 101
Enter the name : Pooja
Enter the id : 102
Enter the name : Pinky
Enter the id : 103
Enter the name : Tina
Index is 0
ID is 101  Name is Pooja
Index is 1
ID is 102  Name is Pinky
Index is 2
ID is 103  Name is Tina
```

2-D Array

- Data is stored in the form of rows and columns
- Also known as matrix form





2-D Array classification

T	O	M
S	A	M
R	A	M

M	I	N	I	
T	O	M		
T	E	E	N	A

Rectangular array

- Arrays that have elements of the same size
- char names[][] = new char[3][3];

Jagged array

- each row of the array may contain different lengths
- char names[][] = new char[3][];
  names[0] = new char[4];
  names[1] = new char[3];
  names[2] = new char[5];

2-D Array - Example

Example - To store price list of 3 products by 2 vendors

```
int price[][]=new int[3][2];

price[0][0]=90;price[0][1]=85;
price[1][0]=180;price[1][1]=173;
price[2][0]=590;price[2][1]=490;

for(int outer=0;outer<price.length;outer++) {
        System.out.println("Product index "+(outer+1));
        for(int inner=0;inner<price[outer].length;inner++){
                System.out.print("Vendor"+(inner+1)+
                        " - "+price[outer][inner]+"   ");
        }
        System.out.println();
}
```

Output

```
Product index 1
Vendor1 - 90  Vendor2 - 85
Product index 2
Vendor1 - 180 Vendor2 - 173
Product index 3
Vendor1 - 590 Vendor2 - 490
```

Common pit falls

Common mistakes we do...

01

```
int []marks=new int[5];
marks[6]=20;
```

Line 2 will lead to a run time exception

BOUNDS CHECKING

We cannot access the element out of the boundary

02

```
int []marks;
marks[0]=10;
```

Line 2 will lead to a compile time error.

ARRAY AS LOCAL VARIABLE

If we access without initializing, it leads to a compile time error.

03

```
class Student {
    private float marks[];
    public void findAverage()
    {
    //below line will lead to runtime exception
        for(int i=0;i<marks.length;i++)
        {
            //logic for finding average
        }
    }
}
public class Test {
    public static void main(String[] args) {
        Student sobj=new Student();
        sobj.findAverage();
    }
}
```

ARRAY AS INSTANCE VARIABLE

marks array will have the default value as null, If we access without initializing, it will lead to a compile time exception

04

```
int array1[]={1,2,3,4,5};
int array2[]={6,7,8,9,10};
array2=array1;
array2[0]=20;
System.out.println(array1[0]);
```

Line 5 will display the output as 20.

ARRAY COPY

When one array is assigned to another, only the reference is copied not the values

Array as a Function Argument

```
class SortArray {

    public void sort(int []nos)
    {
        int n=nos.length;
        for(int i=0;i<n;i++)
            for(int j=0;j<n-1-i;j++)
            {
                if(nos[j]>nos[j+1]){
                    int temp=nos[j];
                    nos[j]=nos[j+1];
                    nos[j+1]=temp;
                }
            }
    }
}
public class Test {
    public static void main(String[] args) {
        SortArray sobj=new SortArray();
        int numbers[]={20,12,50,60};
        sobj.sort(numbers);
        for(int n:numbers)
        {
            System.out.println(n);
        }
        }
}
```

Output

12

20

50

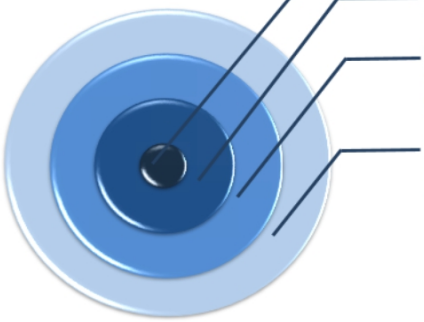
60

Note: Changes done using nos has been reflected to numbers

Java passes "objects by reference" and not by values

- Array is passed as a reference and not as the copy of individual elements
- Changes done using array reference will be reflected in the original array
- The argument in the called method header must be of array type

Variable Arguments



Variable Arguments (varargs) is a newly implemented feature of Java 5.0

It allows a method to accept zero or more arguments

If the number of arguments to a method are of same type, but the number of arguments is unknown, varargs is a better approach

Syntax

return_type method_name (data_type ... variable_name) { ..... }

Variable Arguments

```
public static void main(String[] args)
{
    System.out.println("The sum is " +
                       sum(10,20,30));
}
public int  sum (double... numbers)
{
    int total = 0;
    for (int i = 0; i < numbers.length; i++)
      total += numbers[i];
    return total;
}
```

Argument passed to a method is converted into an array of the same-typed values

sum(10,20,30) ⟺ sum(new int[] {10,20,30})

The rules for varargs

- There can be only one variable argument in the method
- Variable argument must be the last argument

Variable Arguments - Example

```
class VarargsExample
{
    static void display(int num, String... values)  {
        System.out.println("number is "+num);
        for(String s:values){
            System.out.println(s);
        }
    }

    public static void main(String args[]){
        display(500,"hello");//one argument
        display(1000,"my","name","is","varargs");//four arguments

    }
}
```

Output :

number is 500
hello
number is 1000
my name is varargs

Arrays

Arrays is a class in java.util package

Arrays class has a collection of static methods to work with arrays like sorting and searching

All methods in Arrays are static and hence have no constructor.

Few methods in Arrays class are

- equals
- copyOf
- sort
- binarySearch

ARRAYS- CLASS - METHODS

Methods	Description
Arrays.sort(int [] a)	Sorts the specified array into ascending numerical order.
Arrays.sort(int[] a, int fromIndex, int toIndex)	Sorts the specified range of the array into ascending order.
Arrays.fill(int[] a, int value)	Assigns the specified int value to each element of the array.
Arrays.fill(int[] a, int fromIndex, int toIndex, int val)	Assigns the specified int value to each element of the specified range of the array.
Arrays.binarySearch(int[] a, int key)	Searches the specified array of ints for the specified value using the binary search algorithm.
Arrays.binarySearch(int[] a, int fromIndex, int toIndex, int key)	Searches a range of the specified array of ints for the specified value using the binary search algorithm.
Arrays.toString(int[] a)	Returns a string representation of the contents of the specified array.

Arrays.equals

boolean equals(array1, array2)

- This method compares two arrays
- Both arrays should be of same data type and one dimensional
- Return true if both arrays contain same elements in the same order

```
int  a[]={2,4,6,8,10};
int b[]={2,4,6,8,10};
System.out.println(Arrays.equals(a,b)); //returns  true

int  c[]={2,4,6,8,10};
int d[]={10,8,4,2,6};
System.out.println(Arrays.equals(a,b)); //returns  false
```

Arrays.copyOf

TEKNOTURF

array copyOf(originalArray, newLength)

- This method returns an array which is a copy of the originalArray
- It copies elements from 0 to newLength and puts them in the new array
- If newLength exceeds the length of originalArray, it pads those positions with default value.

```
int  num1[]={2,4,6,8,10};
int num2[]=Arrays.copyOf(num1,3);

for(int num : num2)
System.out.println(num);   // prints 2 4 6

Integer num3[]={1,2,3};

Integer num4[]=Arrays.copyOf(num3,5);
 for(Integer num : num4)
System.out.println(num);   // prints  1 2 3 null null
```

Arrays.sort

TEKNOTURF

void sort(array)

- This method sorts the array in ascending order
- If the array is an array of objects, it sorts them in ascending order according to their natural ordering

```
int  num1[]={12,41,32,16,10};
Arrays.sort(num1);

for(int num : num1)
    System.out.println(num);  // prints 10 12 16 32 41

int  num2[]={82,41,32,16,10,46,72,89};
Arrays.sort(num2,1,4); //sorts the numbers from 1  (inclusive) to 4 (exclusive)

for(int num : num2)
System.out.println(num);  //prints 82 16 32 41 10 46 72 89
```

Arrays.binarySearch

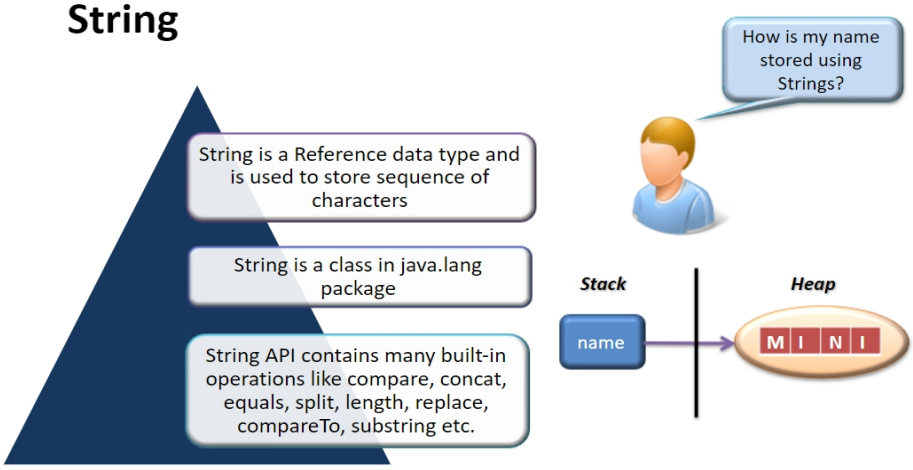
TEKNOTURF

int binarySearch(array,key)

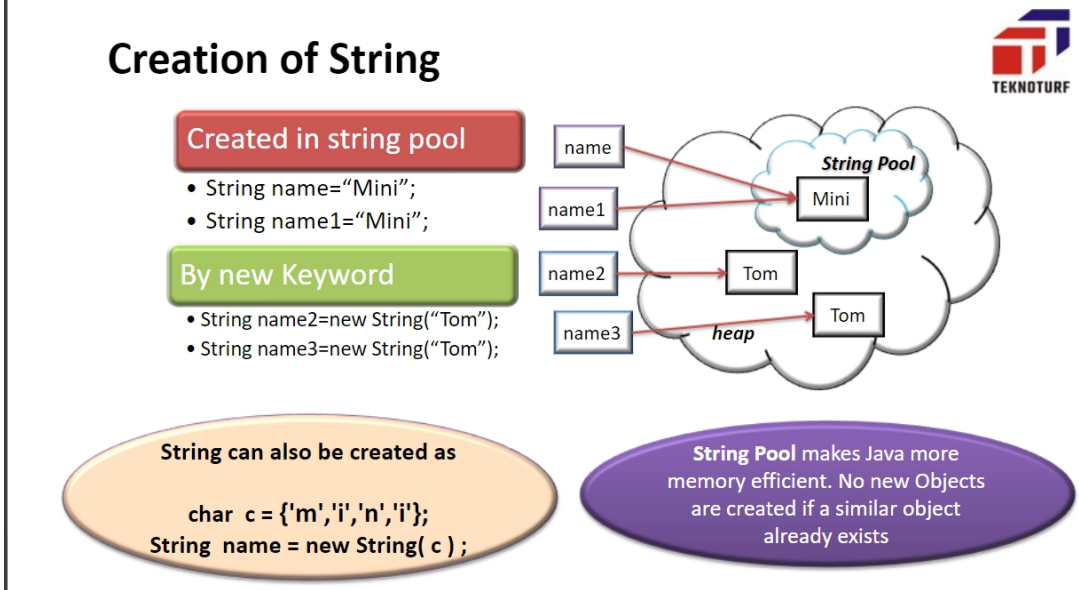
- This method searches for the specified key in the array.
- For this, the array should be sorted prior to this method call.
- Returns the index position where the key is found
- If key not found, returns -1
- If array not sorted, result is undefined

```
int  num[]={82,41,32,16,10,46,72,89};
System.out.println(Arrays.binarySearch(num, 41));  // output unpredictable

Arrays.sort(num);
System.out.println(Arrays.binarySearch(num, 41)); // prints 3
```

String



Creation of String



String

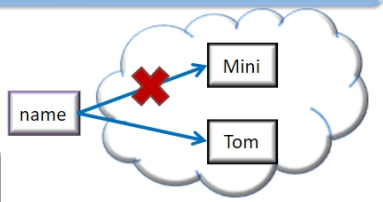
String Objects are immutable

Immutable means something that cannot be modified or changed.

Once string object is created, its data or state can't be changed but a new string object is created.

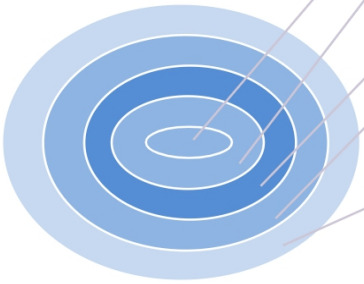
String name="Mini";

name="Tom"



The Object "Mini" is unmodified. Instead a new Object "Tom" is created and mapped with name reference.

String



String class uses character array to store text

To create an empty String

String s = new String() or String s = "";

Like primitive, String can be directly assigned with a String literal

String concatenation can be done by using the '+' operator

String str= "Java"+ "Program" ;

When a String is assigned a string literal directly, it is stored in String pool

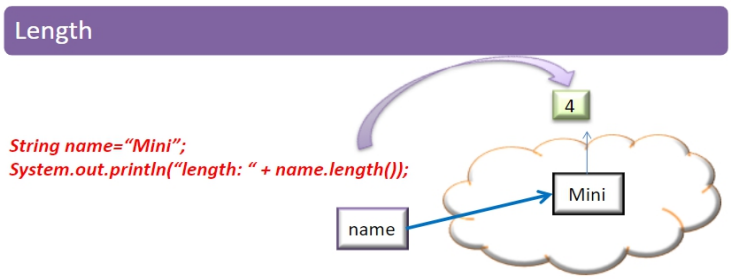
String are immutable

String Functions

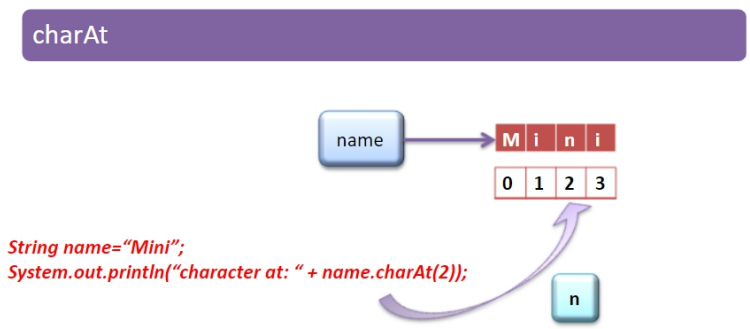
String API contains many built-in methods. Some of the methods are

Function	Function Signature	Purpose
length	int length()	returns string length
charAt	char charAt(int index)	returns char value for the particular index
equals	boolean equals(Object another)	checks the equality of string with another String object
substring	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index
concat	String concat(String str)	concatenates specified string

length Function



charAt Function



The index should be valid – between 0 and length of string -1

If not, it will throw StringIndexOutOfBoundException when the program is executed.

charAt Function

- This code uses charAt function to find the number of vowels in a word got as input
- When a single character is to be got as input, use charAt as

  char choice = sc.next()***.charAt(0);***

```
Scanner sc = new Scanner(System.in);
String word=sc.next();
 int cnt=0;
for(int i=0;i<word.length();i++)  {
    char c = word.charAt(i);
     if(c=='a' || c=='e' || c=='i' ||
                    c=='o' || c=='u')  {
            cnt++;
    }
}
System.out.println("No. of vowels " +
                                  cnt);
```

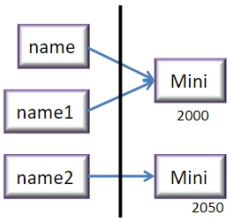
equals and equalsIgnoreCase

TEKNOTURF

equals

equalsIgnoreCase – case insensitive check

```
String name="Mini";
String name1="Mini";
String name2=new String("Mini");
System.out.println(name == name1);   → true
System.out.println(name == name2);   → false
System.out.println(name.equals(name2)); → true
```



== checks for the address of the string Objects to be compared

equals checks for the values inside the objects to be compared

equals and equalsIgnoreCase

TEKNOTURF

equals and equalsIgnoreCase

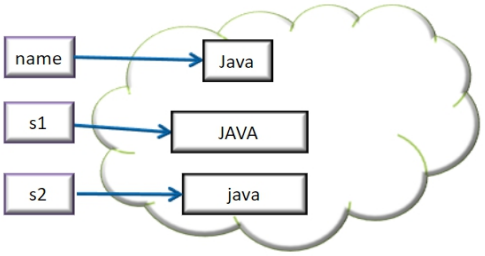
```
String str1 = "Welcome";
String str2 = "Welcome";
String str3 = new String("Welcome");
System.out.println(str1==str2);          → true
System.out.println(str1==str3);          → false
System.out.println(str1.equals(str2));   → true
System.out.println(str1.equals(str3));   → true
```

Case conversion Function

TEKNOTURF

toUpperCase and toLowerCase

```
String name = "Java";
String s1  = name.toUpperCase();
String s2 = name.toUpperCase();
System.out.println(s1+"  "+s2);
```




```
Scanner sc = new Scanner(System.in);
String  choice = sc.next();
If(choice.toLowerCase().equals("yes")
{
      //some code
}
```

indexOf and lastIndexOf Function

indexOf and lastIndexOf

indexOf method searches for the first occurrence of a character or substring.

lastIndexOf Searches for the last occurrence of a character or substring.

If present, returns the index of the character or substring

If not present, returns -1.

```
String email="peter123@gmail.com";
  int pos1=email.indexOf('@'); //8. If it is -1, then mail id invalid

  //When accessing index of character not present -
    int pos2=email.indexOf('#');   //-1, as the character is not present
```

To ensure '.' appears after'@' in email

```
String email="peter.123@tek.co.in";
  int pos1=email.indexOf('@'); //9
  int pos2=email.lastIndexOf('.'); //16
```

indexOf and lastIndexOf Function

indexOf and lastIndexOf

These mehtods are overloaded as

indexOf(char ch) indexOf(char ch,int startIndex)

indexOf(String str) indexOf(String str,int startIndex)

indexOf and lastIndexOf methods are case-sensitive

```
String email="peter123@gmail.com";
int pos1=email.indexOf("gmail"); //9.

    String text="HelloWorldHello";
    int pos1=text.indexOf("ll"); //2
    //When accessing index of String not present -
    int pos2=text.indexOf("xx"); //-1, as the String is not present
```

To ensure '.' appears after'@' in email

```
String email="peter.123@tek.co.in";
  int pos1=email.indexOf('@'); //9
  int pos2=email.lastIndexOf('.'); //16
```

Few Search Functions

contains, startsWith and endsWith

contains method checks if one String is present in another String. If yes returns true. Else returns false.

contains method is case sensitive

To make it case insensitive, convert both the string to upper or lower case

```
  String str="She sells sea shells on the sea shore";
  If(str.contains("sea")
      System.out.println("Yes");
  else
   System.out.println("No");
```

To compare as case insensitive

```
If(str.toLowerCase.contains("sea")
    System.out.println("Yes");
```

Few Search Functions

TEKNOTURF

contains, startsWith and endsWith

startsWith method checks if a String starts with a specific substring

endsWith method checks if a String ends with a specific substring

The return type is boolean

```
String empId  = "TEK254DEV";
If(empId.startsWith("TEK")  {
   if(empId .endsWith("DEV")
       System.out.println("Developer");
   else if(empId.endsWith("TES")
       System.out.println("Tester");
 }
 else
   System.out.println("Invalid  ID");
```

To compare as case insensitive

```
If(empId.toLowerCase().startsWith("tek")
   System.out.println("Valid");
```

Convert String to Character Array

TEKNOTURF

toCharArray()

toCharArray method converts the invoking string into a sequence of characters

The length of the array returned by this function will be the lenth of the string

```
Scanner sc = new Scanner(System.in);
 String word=sc.next();
  int cnt=0;
  char ch[]=word.toCharArray();
  for(char  c : ch) {
      if(c=='a' || c=='e' || c=='i' ||  c=='o' || c=='u')  {
              cnt++;
      }
  }
  System.out.println("No. of vowels " + cnt);
```

Can use either a normal for loop or a for each loop

Replace Function

TEKNOTURF

replace

replace(char old,char new) - Replaces all occurrences of old character with the new character

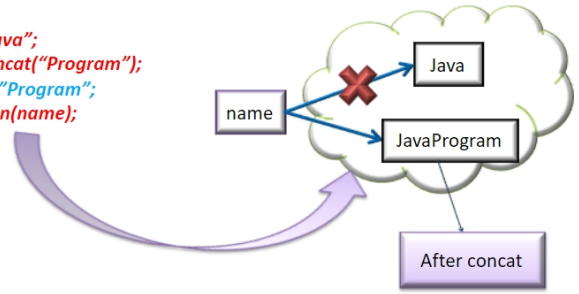
```
String text  = "helloworld";
String res = text.replace('l','w');
   System.out.println(res);
//Output - hewwoworwd
```

```
String sentence1= "Welcome to Java";
String sentence2= "Welcome      to      Java";
String  result = sentence2.replace(" ","");
if(sentence1.equals(sentence2))
     System.out.println("Exact match");
else if(sentence1.replace(" ","").equals(result))
      System.out.println("Same with space difference");
else if(sentence1.equalsIgnoreCase(sentence2))
      System.out.println("Same with case difference");
else
      System.out.println("No Match");
```

concat Function

concat

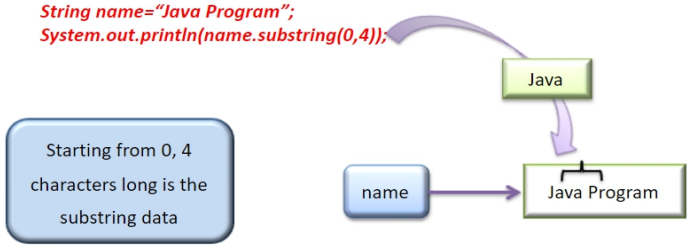
```
String name = "Java";
name = name.concat("Program");
//name = name+"Program";
System.out.println(name);
```



substring Function

substring

```
String name="Java Program";
System.out.println(name.substring(0,4));
```



split Function

split

split() method splits this string against given regular expression and returns a String array.

The regular expression is a String that represents the delimiting regular expression

A regular expression is a String representing the sequence of characters that defines a search pattern , specifically used for string matching

"Java PHP Python".split(" "); - will return an array of size 3, by splitting the string based on space

split Function

TEKNOTURF

split

```
Scanner sc=new Scanner(System.in);
String str=sc.nextLine();

String arr[]=str.split(",");

for(String s : arr)
{
System.out.println(s);
}
```

If input is
Kapil Dev,Tendulkar,MS Dhoni,Kohli

Output will be :
Kapil Dev
Tendulkar
MS Dhoni
Kohli

Few static methods in Character class

TEKNOTURF

isLetter, isDigit, isLetterOrDigit

isLetter, isDigit and isLetterOrDigit are static methods in Character class.

We can access them directly using class name Character

- isLetter(char) determines if the passed character is an alphabet or not.
- isDigity(char) determines if the character is a digit or not and
- isLetterOrDigit(char) determines if the character is alphanumeric or not.
- The return type of the above functions are boolean.

Few static methods in Character class

TEKNOTURF

isLetter, isDigit, isLetterOrDigit

```
Scanner sc=new Scanner(System.in);
String sentence =sc.nextLine();  //Use nextLine for a
String with space

//To convert a String to a character array
char arr[] = sentence.toCharArray();

int alphabets=0,digits=0,splChar=0;

for(int i=0;i<arr.length;i++)
{
    if(Character.isLetter(arr[i]))
        alphabets++;
 else if(Character.isDigit(arr[i]))
   digits++;
 else
   splChar++;
}
    System.out.println("Number of alphabets : "+alphabets);
    System.out.println("Number of digits : "+digits);
    System.out.println("Number of special characters : "+
                                                splChar);

for(char x : arr) {
    if(Character.isLetterOrDigit(x))
        System.out.println( x+" is alphanumeric");
|
```

Few static methods in Character class

isDigit to find the sum of digits in a String

```
Scanner sc=new Scanner(System.in);
String sentence =sc.nextLine();

//To convert a String to a character array
char arr[] = sentence.toCharArray();

int total=0;
for(int i=0;i<arr.length;i++)
{
    if(Character.isDigit(arr[i]))
    {
        total=total+Integer.parseInt(arr[i]+"");
       // arr[i]+"" – converts character in arr[i] to String
       // To convert that String to int use Integer.parseInt method
}
```

StringBuffer

- StringBuffer is used to create a mutable String
- StringBuffer class is thread-safe
- Creating a StringBuffer Object

StringBuffer sb = new StringBuffer("Java");

StringBuffer Function

StringBuffer sb=new StringBuffer("Java");

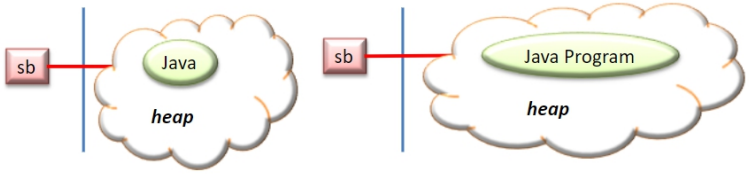
Function signature	Purpose	Example
char charAt(int index)	returns the character at the specified position.	System.out.println(sb.charAt(2)); // v
int length()	returns the length of the string	System.out.println(sb.length()); // 4
String substring(int beginIndex, int endIndex)	returns the substring from the specified beginIndex and endIndex.	String s = sb.substring(0,4); System.out.println(s); // Java
StringBuffer reverse()	used to reverse the string	sb.reverse(); System.out.println(sb); // avaJ

StringBuffer Function

- append
  - StringBuffer append(String s)
  - Appends the specified string with given string.

Original String is modified

```
StringBuffer sb = new StringBuffer("Java");
sb.append(" Program");
System.out.println(sb);
```

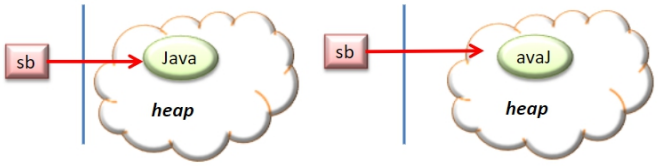


StringBuffer Function

- reverse
  - StringBuffer reverse(String s)
  - Reverses the specified string

Original String is modified

```
StringBuffer sb = new StringBuffer("Java");
sb.reverse();
System.out.println(sb);
```



StringBuffer to String

- StringBuffer object can be converted to String using toString()
- **toString functions return the String representation of reference Object**
- toString method is present in the Object class.

StringBuffer name= new StringBuffer("Raghu");

String sname=name.toString();

System.out.println(sname.equals("Raghu"));

To check whether the given StringBuffer Object's value is equal to Raghu, we need to convert StringBuffer to String and use equals method

StringBuffer Function

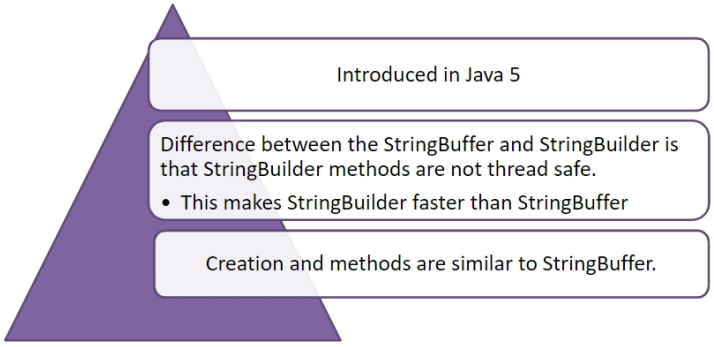
TEKNOTURF

reverse

```
Scanner sc=new Scanner(System.in);
String str=sc.next();
StringBuffer sb=new StringBuffer(str);
sb.reverse(); // will reverse the string
String reverse=sb.toString();
System.out.println("The reversed string is "+ reverse);

//To check for Palindrome (case – insensitive)
if(str.equalsIgnoreCase(reverse))
    System.out.println(str+" is a Palindrome");
else
    System.out.println(str+" is not a Palindrome");
```

StringBuilder

TEKNOTURF



Regular Expression

TEKNOTURF

- Java 4 introduced the support for regular expression
- Regular expressions (RegEx) are string patterns that describe a text
- RegEx used to define string patterns, using which we can search, extract and edit a text
- RegEx plays a vital role in string manipulations
- Example

" **[a-z]+**" is a regex pattern that matches a sequence of one or more lower case alphabets

Java RegEx

Regular Expression

TEKNOTURF

The API for Regex is available in java.util.regex package

This package has 1 interface and 3 classes. The classes are :

- Pattern
- Matcher
- PatternSyntaxException

Pattern and Matcher class

Pattern class

- Is a compiled version of regular expression
- Created as
  Pattern pattern = Pattern.compile(pattern);

Matcher class

- Matcher Object is a regex engine that interprets the pattern and matches the regular expression against the text
- Created as
  Matcher matcher = pattern.matches(text);

Regular Expression

Sample Code

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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

            Pattern pattern=Pattern.compile("[a-z]+");
            Matcher matcher=pattern.matcher("Welcome to Java 1.8");
            while(matcher.find()){
                System.out.println("Found "+matcher.group());
            }
    }
}
```

Output :
Found elcome
Found to
Found ava

Character classes

- Used to define what the pattern should look for

RegEx	Description	Example
[abc]	a, b, or c	Pattern.matches("[abc]", "c")
[^abc]	Any character except a, b, or c	Pattern.matches("[^abc]", "x")
[a-zA-Z]	a through z or A through Z, inclusive	Pattern.matches("[a-zA-Z]", "V")
[a-d[m-p]]	a through d, or m through p: [a-dm-p]	Pattern.matches("[a-dm-p]", "n")
[a-z&&[def]]	d, e, or f	Pattern.matches("[a-z&&[def]]", "e")
[a-z&&[^bc]]	a through z, except for b and c: [ad-z]	Pattern.matches("[a-z&&[^bc]]", "d")
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z]	Pattern.matches("[a-z&&[^m-p]]", "x")

Character classes

- Example

```
import java.util.regex.Pattern;

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

            System.out.println(Pattern.matches("[abc]", "c"));            //true
            System.out.println(Pattern.matches("[abc]", "ab"));           //false
            System.out.println(Pattern.matches("[a-z]", "x"));             //true
            System.out.println(Pattern.matches("[^a-z]", "1"));            //true
            System.out.println(Pattern.matches("[^a-z]", "12$"));          //false
            System.out.println(Pattern.matches("[a-z&&[aeiou]]", "x")); //false
            System.out.println(Pattern.matches("[a-z&&[^m-q]]", "o")); //false

   }
}
```

Quantifiers

- Specifies the number of occurrence of a character

RegEx	Description	Example
X?	X occurs once or not at all	*Pattern.matches("[a-z]?","")*
X+	X occurs once or more times	*Pattern.matches("[a-zA-Z]+","Welcome")*
X*	X occurs zero or more times	*Pattern.matches("[0-9]*","2018")*
X{n}	X occurs n times only	*Pattern.matches("[0-9]{3}","517")*
X{n,}	X occurs n or more times	*Pattern.matches("[a-z]{5,}","looks good")*
X{y,z}	X occurs at least y times but less than z times	*Pattern.matches("[a-z]{5,10}","welcome")*

Meta characters

- Used as short codes

RegEx	Description	Example
.	Any character (may or may not match terminator)	Pattern.matches(".", “(")
\d	Any digits, short of [0-9]	Pattern.matches("\\d", "1")
\D	Any non-digit, short for [^0-9]	Pattern.matches("\\D", "a")
\s	Any whitespace character, short for [\t\n\x0B\f\r]	Pattern.*matches("\\s"," ")*
\S	Any non-whitespace character, short for [^\s]	Pattern.*matches("\\S","z")*
\w	Any word character, short for [a-zA-Z_0-9]	Pattern.*matches("\\w","_")*
\W	Any non-word character, short for [^\w]	Pattern.*matches("\\W","#")*
\b	A word boundary	Pattern.*matches(“young\\b”,“young")*
\B	A non word boundary	Pattern.*matches(“young\\B”,“youngest")*

Regular Expression using String method

- Validation of input with regex can be done using String method – matches
- Syntax : originalString.matches(regex)
- Example

```
import java.util.regex.Pattern;

public class RegExDemo {

	public static void main(String[] args) {
		//Validating a name with alphabets and space
		System.out.println("Rosy".matches("[a-zA-Z ]+"));		//true
		System.out.println("David kumar".matches("[a-zA-Z]+"));  //true
		System.out.println("David 234".matches("[a-zA-Z]+"));	//false

	}
}
```

Validating Data

The sequence of characters received as input can be validated against a pattern using

- Character class
- Quantifiers
- Meta Characters

Validating Data

Example

```
import java.util.regex.Pattern;

public class RegExDemo {

	public static void main(String[] args) {
		System.out.println(Pattern.matches("[a-zA-Z ]+","Positive Thinking"));  // true
		System.out.println(Pattern.matches("[a-zA-Z ]+","Java 1.8"));  // false
		System.out.println(Pattern.matches("[0-9]{10}","9874563210"));  //true
		System.out.println(Pattern.matches("[0-9]{10}","987456321"));  //false
		System.out.println(Pattern.matches("[A-Z]{3}[/][0-9]{2}","IBM/24")); //true
		System.out.println(Pattern.matches("[A-Z]{3}[/][0-9]{2}","tek/24")); //false
		System.out.println(Pattern.matches("[\\d]{5,}","953687")); //true
		System.out.println(Pattern.matches("[\\d]{5,}","3687"));  //false
	}
}
```

Validating Data

Example - Validating a phone number
Should contain 10 digits
Should start with a digit 7 / 8 / 9

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegExDemo {

        public static void main(String[] args) {
            System.out.println(Pattern.matches("[7-9][0-9]{9}", "9875632145"));  // true
            System.out.println(Pattern.matches("[7-9][0-9]{9}", "98756321451"));  // false
            System.out.println(Pattern.matches("[7-9][0-9]{9}", "987563214"));  // false
            System.out.println(Pattern.matches("[7-9][0-9]{9}", "5875632145"));  // false
        }
}
```