Object Oriented Programming

OOP 6 : String & Arrays

Hirra Anwar

Content by Dr. Abdul Ghafoor

National University of Sciences and Technology (NUST)

School of Electrical Engineering and Computer Science

Islamabad

- String Class & Methods
- String Buffer & Methods
- Arrays

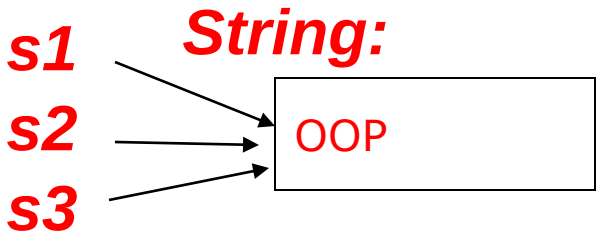
- Set of character, it contains special characters, alphabets, numbers.
- String class exists in java.lang package
- String supports two operators + and +=
  - Other Java classes do not provide these overloaded operates

- String in java is handled by two classes
  - String
  - StringBuffer

- Anonymous object of a class String
- Enclosed in double quotes
  - For example
  - System.out.println(“Hello”)
  - String myStr=“Hello”;
- Do not have constructor
  - Java manages
- Usage:
  - Assign to a String reference
  - Pass to the functions (methods, constructors)

- String is Immutable class
- Immutability Once created, a string cannot be changed: none of its methods changes the string. Such objects are called immutable.
- Immutable objects are convenient because several references can point to the same object safely
- There is no danger of changing an object through one reference without the others being aware of the change.

```
String s1 =“OOP”;
String s2 =“OOP”;
String s3 =“OOP”;
```



```
String s1 = "Hello";
String s2 = s1;
s1 = s1.replace('e', '0');
System.out.println(s1);
System.out.println(s2);
System.out.println(s1 == s2);
```

- An empty String has no characters.
- It's length is 0.

```
String word1 = "";
String word2 = new String();
```

- Not the same as an uninitialized String.

```
private String errorMsg;
```

errorMsg is null Empty strings

- String()
- String(char chars[])
- String(char chars[],int start, int numChars)
- String(String x)
- String(byte bytes[])
- String(byte bytes[],int start, int numChars)

int length()

Usage :

<string>.length();

Examples :

S.o.P("Object".length());

s1.length();

name.length();

Adding Strings together, Concatenation '+' operator can be used to concatenate two strings

String *concat(String other)* method can also be used

Examples :

1. *String s1 = "Hello"+"How are You"+20+20;*

   // s1 will be "HelloHowareYou2020"

2. *String s2 = "Object"*

   String s3 = "Programming"

   String s4 = s2 + s3 OR String s4 = s2.concat(s3);

3. *System.out.println("xyz".concat("oop"));*

4. *String s1 = 20+20+"Hello"+"How are You";*

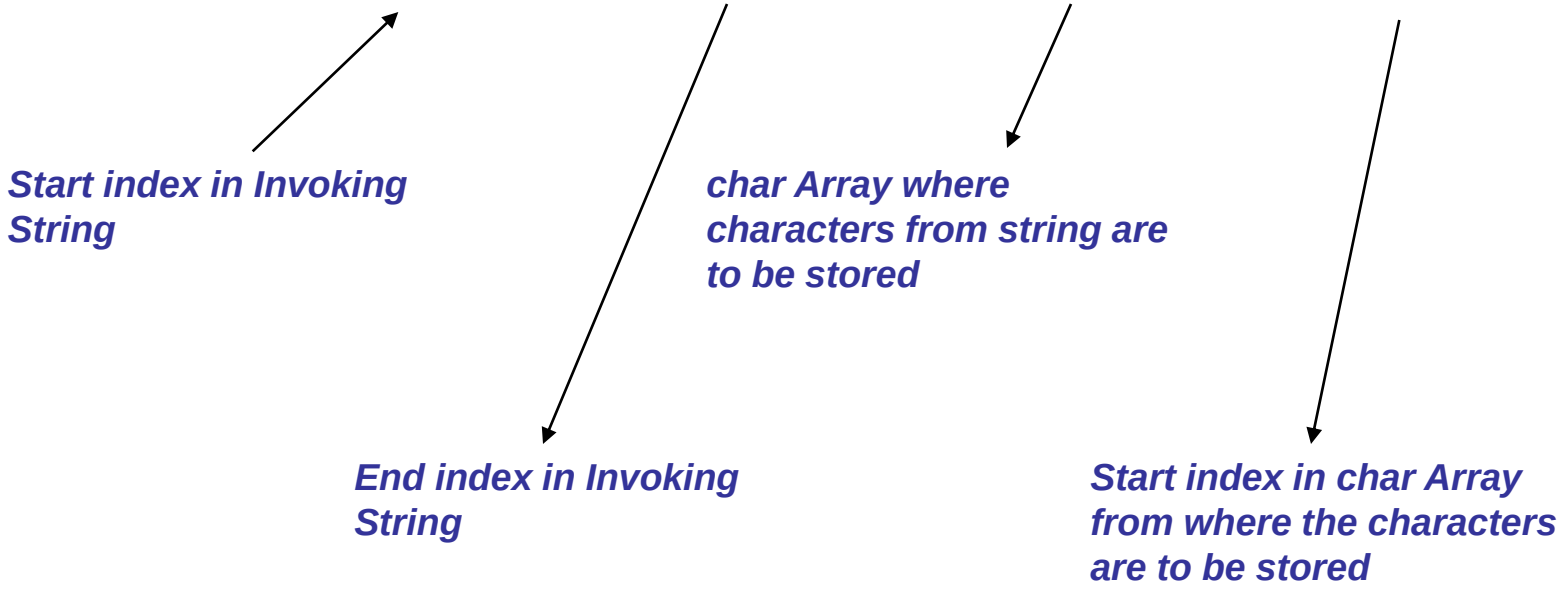
   // s1 will be "40HelloHowareYou"

Extracting a Single Character from a String char charAt(int where)

- charAt() method extracts a character from a string at a given index
- <<where>> parameter should be within range (0 to stringlength-1).
  - Otherwise StringIndexOutOfBoundsException will be thrown
- Examples :

```
char ch = "xyz".charAt(1);  // ch will be 'y'
char ch = "xyz".charAt(3);
//StringIndexOutOfBoundsException
```

- To Extract more than one character we can use getChars() method
- Syntax:
- void getChars (int sourceStart, int sourceEnd, char target[ ], int targetStart)



boolean equals(Object str)

boolean equalsIgnoreCase(String str)

Examples :

(i) *"xyz".equals("abc");* << false>>

(ii) *"xyz".equalsIgnoreCase("XYZ")* << true>

(iii) *s1.equals(s2)* << returns if s1 and s2 are equal>>

(iv) *s1.equalsIgnoreCase(s2)*

<< returns if s1 and s2 are equal by ignoring case>>

equals Vs ==

- Used searching first/last occurrences of a character / substring
- Return the index of character or substring if found otherwise -1
- These two methods are overloaded in several different ways

int indexOf(int ch) / int lastIndexOf(int ch)

int indexOf(String str) / int lastIndexOf(String str)

int indexOf(int ch, int startIndex) / int lastIndexOf(int ch, int startIndex)

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

String s1 = "Now is the time for all good men to come forward to aid their country";

```
System.out.println(s1.indexOf('t'));        7
System.out.println(s1.lastIndexOf('t'));    66

System.out.println(s1.indexOf("to"));       33
System.out.println(s1.lastIndexOf("to"));   49

System.out.println(s1.indexOf("to",35));        49
System.out.println(s1.lastIndexOf("to",35));    33
```

- String substring(int startIndex)
- Returns a substring from invoking string starting form startIndex up to last of the invoking string
  - *"Islamabad-Pakistan".substring(7);*
  - startIndex <= invoking string length – 1.

- *String substring(int startIndex, int endIndex)*
  - Returns a substring from invoking string starting form startIndex up to endIndex-1
  - endIndex > startIndex and both should be within permitted range.
  - *"Islamabad-Pakistan".substring(2,6);*

- String toLowerCase()
- String toUpperCase()
- Examples :

1. S.O.P(“object”.toUpperCase());
2. S.O.P(“OBJECT”.toLowerCase());

```
class StringTest
{
        public  static  boolean  isPalinDrome(String str)
        {
                // This method returns true if str is palindrome otherwise false
        } // End of isPalinDrome()
        public static void removeDuplicates(String  values)
        {
                // Removes duplicates  characters from values for example feeling ,. Remove one e
        } // End of removeDuplicates()
        public  static  int  exists(String value, String name)
        {
                // Returns the index of name in values if exists otherwise returns -1
        } // End of exists()
        public  static  String  getUnion(String name1, String name2)
        {
                // Returns the union of name1, name2
        }// End of getUnion()
        public  static  String  getintersection(String name1, String name2)
        {
                //  Returns the intersection of name1, name2
        }// End of getintersection()
        public  static  void  main(string args[])
        {
                //  Call all above methods
        }// End of main()
}// End of class StringTest
```