Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to CSE, EEE, EIE, ICE, IT)

UNIT NO 2

INHERITANCE AND INTERFACES

2.9 STRINGS

COMPUTER SCIENCE & ENGINEERING

UNIT - II
Strings in Java

TOPICS PLAN

- Introduction
- What is String?
- How to create a String?
- Methods of String class
- Immutable String
- StringBuffer class
- StringBuilder class
- String vs StringBuffer
- StringBuffer vs Builder
- Creating Immutable class
- Video link
- MCQ Link

Introduction

- In Java, string is basically an object that represents sequence of char values.
- Strings are used for storing text.
- A String variable contains a collection of characters surrounded by double quotes:
- **Example:** String greeting = "Hello";
- An array of characters works same as Java string.
- **For example:**

```
char[] ch={'j','a','v','a'};

String s=new String(ch);
```

is same as:

```
String s="java";
```

Definition of String

- **What is String in java?**
- Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters.
- The java.lang.String class is used to create a string object.
- **Example:**
- String greeting = "Hello world!";

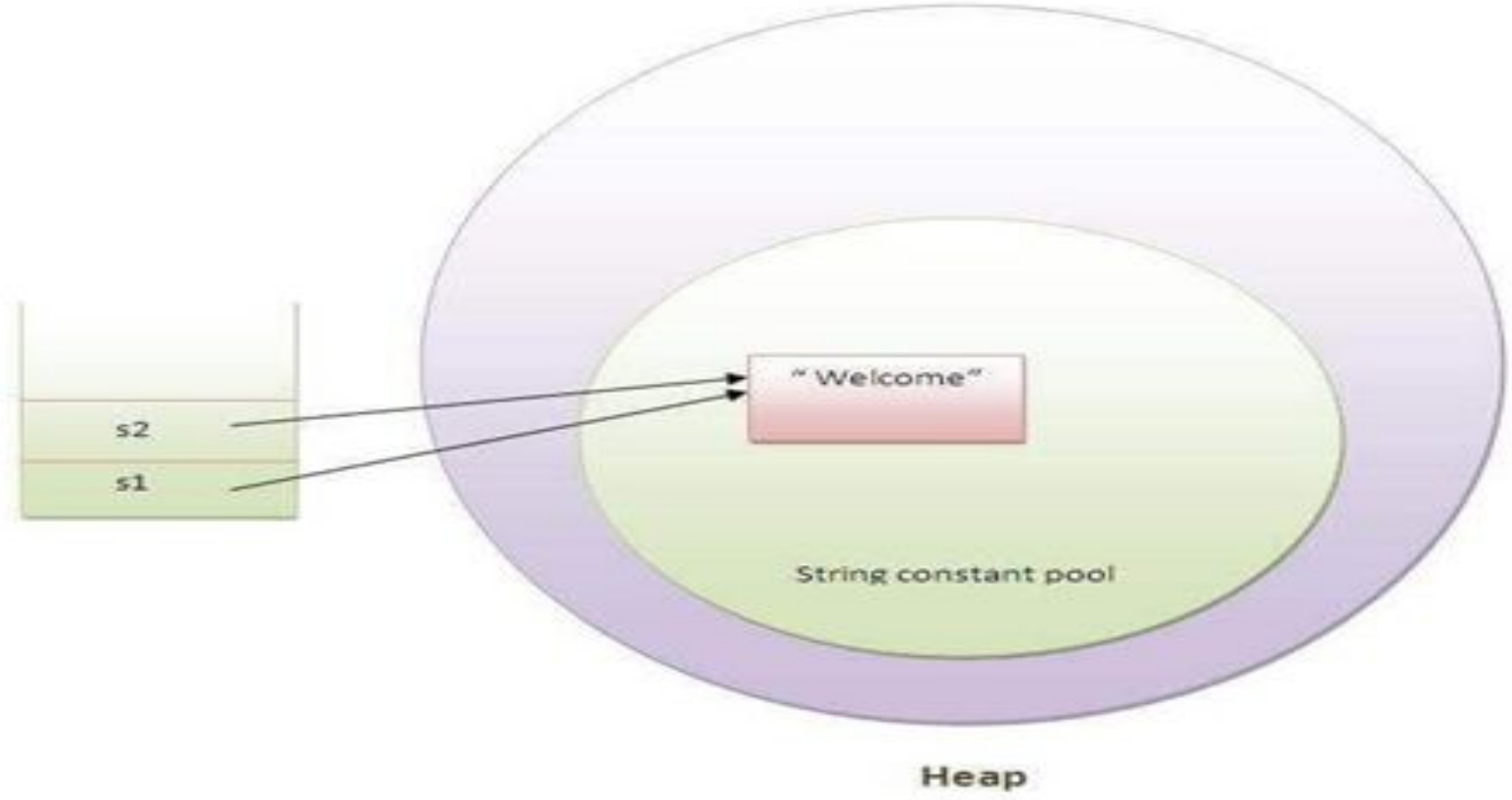
String Creation

- How to create a string object?
- There are two ways to create String object:
  - 1. By string literal
  - 2. By new keyword

String Creation by string literal

- Java String literal is created by using double quotes.
- **For Example:**
  - String s="welcome";
- Each time you create a string literal, the JVM checks the "string constant pool" first.
- If the string already exists in the pool, a reference to the pooled instance is returned.
- If the string doesn't exist in the pool, a new string instance is created and placed in the pool.
- **For example:**
  - String s1="Welcome";
  - String s2="Welcome";//It doesn't create a new instance

String Creation by string literal



String Creation by string literal

- In the above example, only one object will be created.
- Firstly, JVM will not find any string object with the value "Welcome" in string constant pool, that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.
- Note: String objects are stored in a special memory area known as the "string constant pool".
- Why Java uses the concept of String literal?
- To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

String Creation by new keyword

- String s=new String("Welcome");//creates two objects and one reference variable
- In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool.
- The variable s will refer to the object in a heap (non-pool).

Example of String creation

```
public class StringExample{
public static void main(String args[]){
String s1="java";//creating string by java string literal
char ch[]={'s','t','r','i','n','g','s'};
String s2=new String(ch);//converting char array to string
String s3=new String("example");//creating java string by new keyword
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);  } }
```

Output:

```
java
strings
example
```

Java String Methods

- The java.lang.String class provides many useful methods to perform operations on sequence of char values.

No.	Method	Description
1.	char charAt(int index)	returns char value for the particular index
2.	int length()	returns string length
3.	static String format(String format, Object... args)	returns a formatted string.

Java String Methods

No.	Method	Description
4.	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.
5.	String substring(int beginIndex)	returns substring for given begin index.
6.	boolean contains(CharSequence s)	returns true or false after matching the sequence of char value.
7.	static String join(CharSequence delimiter, CharSequence... elements)	returns a joined string.

Java String Methods

No.	Method	Description
8.	boolean equals(Object another)	checks the equality of string with the given object.
9.	boolean isEmpty()	checks if string is empty.
10.	String concat(String str)	concatenates the specified string.
11.	String replace(char old, char new)	replaces all occurrences of the specified char value.

Java String Methods

No.	Method	Description
12.	String replace(CharSequence old, CharSequence new)	replaces all occurrences of the specified CharSequence.
13.	static String equalsIgnoreCase(String another)	compares another string. It doesn't check case.
14.	String[] split(String regex)	returns a split string matching regex.
15.	String[] split(String regex, int limit)	returns a split string matching regex and limit.

Java String Methods

No.	Method	Description
16.	int indexOf(int ch)	returns the specified char value index.
17.	int indexOf(int ch, int fromIndex)	returns the specified char value index starting with given index.
18.	int indexOf(String substring)	returns the specified substring index.
19.	int indexOf(String substring, int fromIndex)	returns the specified substring index starting with given index.

Java String Methods

No.	Method	Description
20.	String toLowerCase()	returns a string in lowercase.
21.	String toLowerCase(Locale l)	returns a string in lowercase using specified locale.
22.	String toUpperCase()	returns a string in uppercase.
23.	String toUpperCase(Locale l)	returns a string in uppercase using specified locale.
24.	String trim()	removes beginning and ending spaces of this string.

Java String Method - charAt()

- The java string charAt() method returns *a char value at the given index number*.
- The index number starts from 0 and goes to n-1, where n is length of the string.
- It returns StringIndexOutOfBoundsException if given index number is greater than or equal to this string length or a negative number.
- **Java String charAt() method example**
- public class CharAtExample{
- public static void main(String args[]){
- String name="java";
- char ch=name.charAt(2);//returns the char value at the 2nd index
- System.out.println(ch);
- }}
- **Output:**

  v

Java String Method - compareTo()

- The java string compareTo() method compares the given string with current string lexicographically. It returns positive number, negative number or 0.
- It compares strings on the basis of Unicode value of each character in the strings.
  - If first string is lexicographically greater than second string, it returns positive number (difference of character value).
  - If first string is less than second string lexicographically, it returns negative number and
  - if first string is lexicographically equal to second string, it returns 0.

Java String Method - compareTo()

```
public class CompareToExample{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="meklo";
String s4="hemlo";
String s5="flag";
System.out.println(s1.compareTo(s2));//0 because both are equal
System.out.println(s1.compareTo(s3));//-5 because "h" is 5 times lower than "m"
System.out.println(s1.compareTo(s4));//-1 because "l" is 1 times lower than "m"
System.out.println(s1.compareTo(s5));//2 because "h" is 2 times greater than "f"  }}
```

Output:

0
-5
-1
2

Java String Method - concat()

- The java string concat() method *combines specified string at the end of this string*.
- It returns combined string. It is like appending another string.
- Java String concat() method example
- public class ConcatExample{
- public static void main(String args[]){
- String s1="java string";
- s1.concat("is immutable");
- System.out.println(s1);
- s1=s1.concat(" is immutable so assign it explicitly");
- System.out.println(s1);
- }}
- *Output*
- *java string*
- *java string is immutable so assign it explicitly*

Immutable String

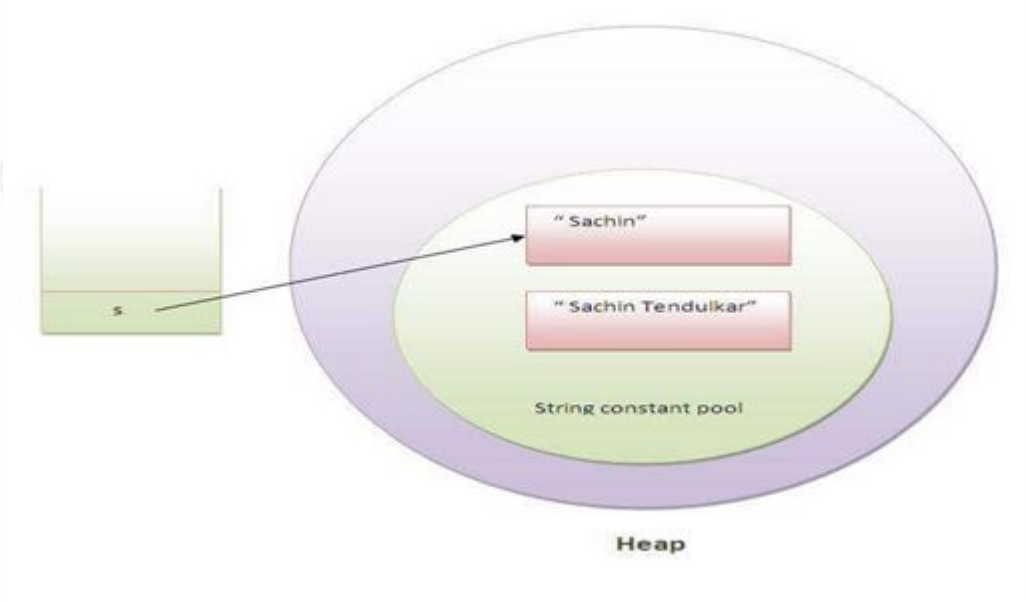
- In java, string objects are immutable. Immutable simply means unmodifiable or unchangeable.
- Once string object is created its data or state can't be changed but a new string object is created.
- **Example:**

```
class Testimmutablestring{
 public static void main(String args[]){
   String s="Sachin";
   s.concat(" Tendulkar");//concat() method appends the string at the end
   System.out.println(s);//will print Sachin because strings are immutable objects   }
}
```

- **Output:Sachin**

Justification of Immutable String

- Now it can be understood by the diagram given below. Here Sachin is not changed but a new object is created with sachintendulkar. That is why string is known as immutable.



Justification of Immutable String

- As you can see in the above figure that two objects are created but s reference variable still refers to "Sachin" not to "Sachin Tendulkar".
- But if we explicitely assign it to the reference variable, it will refer to "Sachin Tendulkar" object.
- For example:
- class Testimmutablestring1{
- public static void main(String args[]){
- String s="Sachin";
- s=s.concat(" Tendulkar");
- System.out.println(s);
- } }
- Output:Sachin Tendulkar

Why string objects are Immutable?

- Because java uses the concept of string literal.
- Suppose there are 5 reference variables, all refers to one object "sachin".
- If one reference variable changes the value of the object, it will be affected to all the reference variables.
- That is why string objects are immutable in java.

What is mutable string?

- A string that can be modified or changed is known as mutable string.
- StringBuffer and StringBuilder classes are used for creating mutable string.

Java StringBuffer class

- Java StringBuffer class is used to create mutable (modifiable) string.
- The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.
- Note: Java StringBuffer class is thread-safe i.e. multiple threads cannot access it simultaneously. So it is safe and will result in an order.

Constructor of StringBuffer class

Constructor	Description
StringBuffer()	creates an empty string buffer with the initial capacity of 16.
StringBuffer(String str)	creates a string buffer with the specified string.
StringBuffer(int capacity)	creates an empty string buffer with the specified capacity as length.

methods of StringBuffer class

Method	Description
append(String s)	is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
insert(int offset, String s)	is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.

methods of StringBuffer class

Method	Description
delete(int startIndex, int endIndex)	is used to delete the string from specified startIndex and endIndex.
reverse()	is used to reverse the string.
replace(int startIndex, int endIndex, String str)	is used to replace the string from specified startIndex and endIndex.
length()	is used to return the length of the string i.e. total number of characters.

methods of StringBuffer class

Method	Description
delete(int startIndex, int endIndex)	is used to delete the string from specified startIndex and endIndex.
reverse()	is used to reverse the string.
replace(int startIndex, int endIndex, String str)	is used to replace the string from specified startIndex and endIndex.
length()	is used to return the length of the string i.e. total number of characters.

Example of append() method of StringBuffer class

- StringBuffer append() method
- The append() method concatenates the given argument with this string.
- class StringBufferExample{
- public static void main(String args[]){
- StringBuffer sb=new StringBuffer("Hello ");
- sb.append("Java");//now original string is changed
- System.out.println(sb);//prints Hello Java
- }
- }

Example of replace() method of StringBuffer class

- StringBuffer replace() method
- The replace() method replaces the given string from the specified beginIndex and endIndex.
- class StringBufferExample3{
- public static void main(String args[]){
- StringBuffer sb=new StringBuffer("Hello");
- sb.replace(1,3,"Java");
- System.out.println(sb);//prints HJavalo
- }
- }

StringBuilder class

- Java StringBuilder class is used to create mutable (modifiable) string.
- The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized.
- It is available since JDK 1.5.

Constructor of StringBuilder class

Constructor	Description
StringBuilder()	creates an empty string Builder with the initial capacity of 16.
StringBuilder(String str)	creates a string Builder with the specified string.
StringBuilder(int length)	creates an empty string Builder with the specified capacity as length.

Methods of StringBuilder class

Method	Description
public StringBuilder append(String s)	is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
public StringBuilder insert(int offset, String s)	is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.

Methods of StringBuilder class

Method	Description
public StringBuilder replace(int startIndex, int endIndex, String str)	is used to replace the string from specified startIndex and endIndex.
public StringBuilder delete(int startIndex, int endIndex)	is used to delete the string from specified startIndex and endIndex.
public StringBuilder reverse()	is used to reverse the string.

Example of inert() Methods of StringBuilder class

- The StringBuilder insert() method inserts the given string with this string at the given position.
- class StringBuilderExample2{
- public static void main(String args[]){
- StringBuilder sb=new StringBuilder("Hello ");
- sb.insert(1,"Java");//now original string is changed
- System.out.println(sb);
- }
- }
- output
- prints HJavaello

Example of inert() Methods of StringBuilder class

- The StringBuilder insert() method inserts the given string with this string at the given position.
- class StringBuilderExample2{
- public static void main(String args[]){
- StringBuilder sb=new StringBuilder("Hello ");
- sb.insert(1,"Java");//now original string is changed
- System.out.println(sb);
- }
- }
- output
- prints HJavaello

Example of reverse() Methods of StringBuilder class

- The reverse() method of StringBuilder class reverses the current string.
- class StringBuilderExample5{
- public static void main(String args[]){
- StringBuilder sb=new StringBuilder("Hello");
- sb.reverse();
- System.out.println(sb);
- }
- }
- **output:**
- prints olleH

Difference between String and StringBuilder class

String	StringBuffer
String class is immutable.	StringBuffer class is mutable.
String is slow and consumes more memory when you concat too many strings because every time it creates new instance.	StringBuffer is fast and consumes less memory when you concat strings.
String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method.	StringBuffer class doesn't override the equals() method of Object class.

Difference between StringBuffer and StringBuilder class

StringBuffer	StringBuilder
StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.

Video Link for Strings in Java

- https://www.youtube.com/watch?v=4l50UaPca7Y
- https://www.youtube.com/watch?v=N63JCXwdd14&list=PL9ooVrP1hQOHb4bxoHauWVwNg4FweDItZ&index=19

MCQ LINK

- **String Quiz Link**
  - https://forms.gle/X9su5sB6iapm6Z229