Web Technology
(KCS-602)
Unit 1

Prepared By

Abhishek Kesharwani

Assistant Professor,UCER Naini,Allahabad

Java String

- **Java String** provides a lot of concepts that can be performed on a string such as compare, concat, equals, split, length, replace, compareTo, intern, substring etc.
- In java, string is basically an object that represents sequence of char values.
- 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";
```

- java.lang.String class
- The java String is immutable i.e. it cannot be changed but a new instance is created.

There are two ways to create String object:

- By string literal
- By new keyword

1) String Literal

- Java String literal is created by using double quotes.

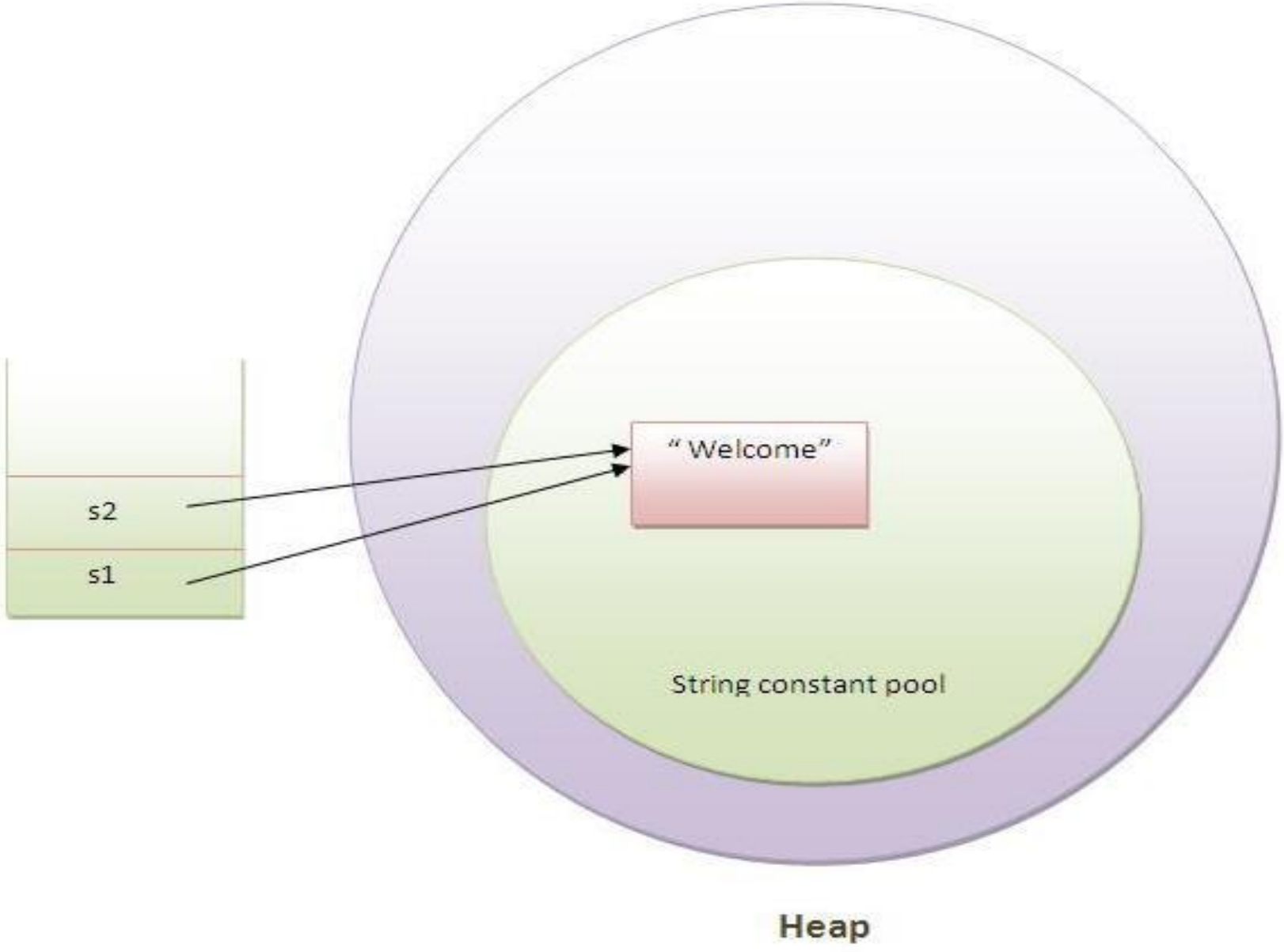
For Example:

```
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 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";//will not create new instance
```


s2
s1
" Welcome"
String constant pool
Heap

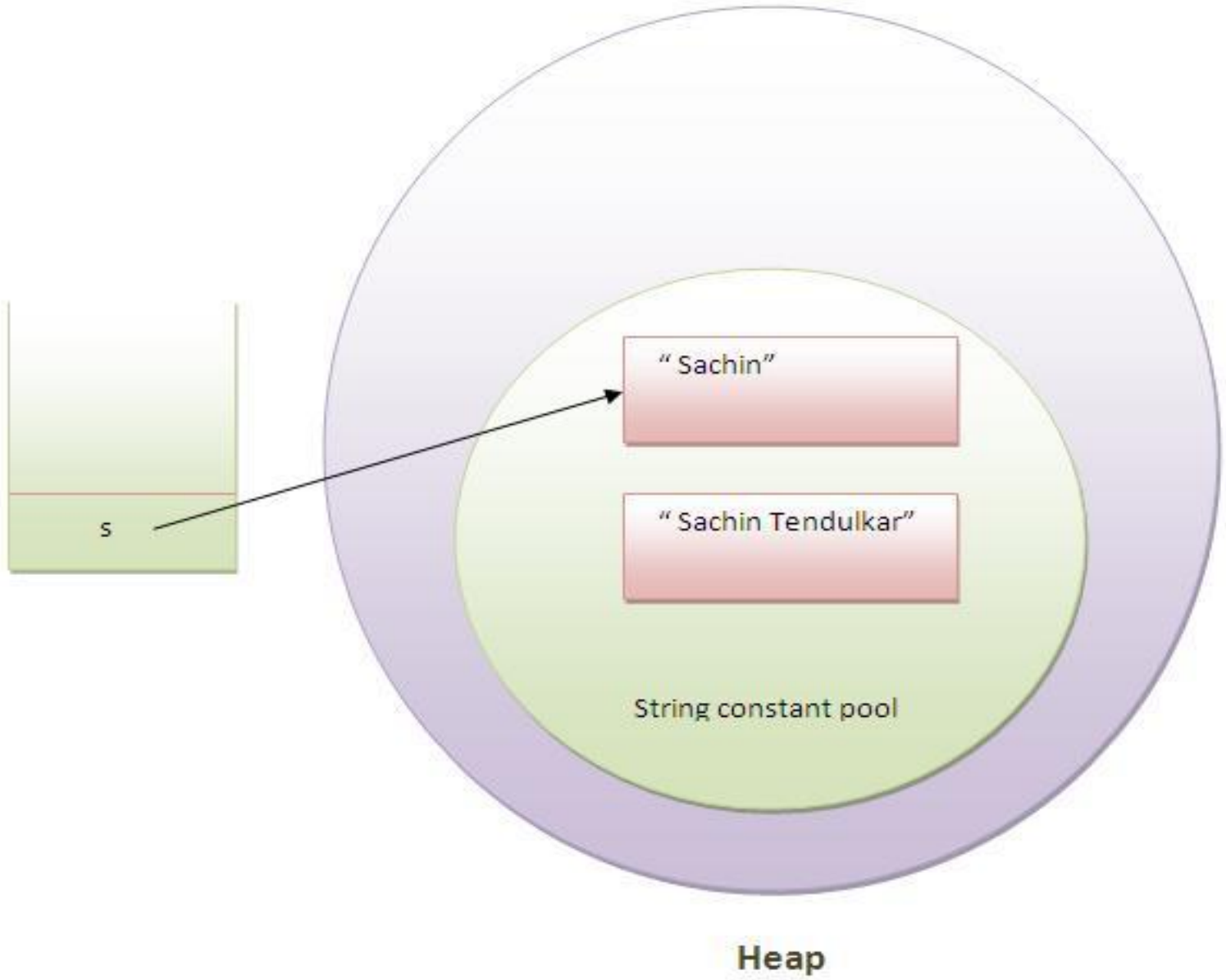
- In the above example only one object will be created.
- Firstly JVM will not find any string object with the value "Welcome" in string constant pool, so it will create a new object.
- After that it will find the string with the value "Welcome" in the pool, it will not create new object but will return the reference to the same instance.

Method	Description
char charAt(int index)	returns char value for the particular index
int length()	returns string length
String substring(int beginIndex)	returns substring for given begin index
String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index
boolean contains(CharSequence s)	returns true or false after matching the sequence of char value
boolean equals(Object another)	checks the equality of string with object
boolean isEmpty()	checks if string is empty
String concat(String str)	concatenates specified string
String replace(char old, char new)	replaces all occurrences of specified char value
String trim()	returns trimmed string omitting leading and trailing spaces
String toUpperCase()	returns string in uppercase.
int indexOf(int ch)	returns specified char value index

Immutable String in Java

- 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.

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


" Sachin"
" Sachin Tendulkar"
s
String constant pool
Heap

1) String compare by equals() method

- The String equals() method compares the original content of the string. It compares values of string for equality. String class provides two methods:
- **public boolean equals(Object another)** compares this string to the specified object.
- **public boolean equalsIgnoreCase(String another)** compares this String to another string, ignoring case.

```
class Teststringcomparison1{
 public static void main(String args[]){
   String s1="Sachin";
   String s2="Sachin";
   String s3=new String("Sachin");
   String s4="Saurav";
   System.out.println(s1.equals(s2));//true
   System.out.println(s1.equals(s3));//true
   System.out.println(s1.equals(s4));//false
 } }
```

Output: true true false

```
class Teststringcomparison2{
 public static void main(String args[]){
   String s1="Sachin";
   String s2="SACHIN";

   System.out.println(s1.equals(s2));//false
   System.out.println(s1.equalsIgnoreCase(s2));//true
 }
}
```

Output: false true

2) String compare by == operator

The = = operator compares references not values.

```
class Teststringcomparison3{
 public static void main(String args[]){
   String s1="Sachin";
   String s2="Sachin";
   String s3=new String("Sachin");
   System.out.println(s1==s2);//true
                                (because both refer to same instance)
  System.out.println(s1==s3);
                        //false(because s3 refers to instance created in non
    pool)
 }
}
```

3) String compare by compareTo() method

The String compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Suppose s1 and s2 are two string variables. If:

s1 == s2 :0

s1 > s2 :positive value

s1 < s2 :negative value

```
class Teststringcomparison4{
 public static void main(String args[]){
   String s1="Sachin";
   String s2="Sachin";
   String s3="Ratan";
   System.out.println(s1.compareTo(s2));//0
   System.out.println(s1.compareTo(s3));//1(because s1>s3)
   System.out.println(s3.compareTo(s1));//-1(because s3 < s1 )
 } }
```

Output:0 1 -1

String Concatenation in Java

In java, string concatenation forms a new string *that is* the combination of multiple strings.

There are two ways to concat string in java:

By + (string concatenation) operator

By concat() method

1) String Concatenation by + (string concatenation) operator

Java string concatenation operator (+) is used to add strings. For Example:

```
class TestStringConcatenation1{
 public static void main(String args[]){
   String s="Sachin"+" Tendulkar";
   System.out.println(s);//Sachin Tendulkar
 } }
```

Output:Sachin Tendulkar

Java String split

```
public class SplitExample{
public static void main(String args[]){
String s1="java string split method by javatpoint";
String[] words=s1.split(“ ");//splits the string based on whitespace
for(String w:words){
System.out.println(w);
}
}}
```

Java String endsWith

```
public class EndsWithExample{
public static void main(String args[]){
String s1="java by javatpoint";
System.out.println(s1.endsWith("t"));
System.out.println(s1.endsWith("point"));
}}
```

Output:

true true

substring() method

```
public class SubstringExample{
public static void main(String args[]){
String s1="javatpoint";
System.out.println(s1.substring(2,4));//returns va
System.out.println(s1.substring(2));//returns vatpoint
}}
```

Java String join() method example

```
public class StringJoinExample{
public static void main(String args[]){
String joinString1=String.join("-
   ","welcome","to","javatpoint");
System.out.println(joinString1);
}}
```

welcome-to-javatpoint