String In Java

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

- **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.
- The java.lang.String class implements *Serializable*, *Comparable* and *CharSequence* interfaces.
- The java String is immutable i.e. it cannot be changed but a new instance is created. For mutable class, you can use StringBuffer and StringBuilder class.

Create String Object

There are two ways to create String object:

- By string literal
- By new keyword

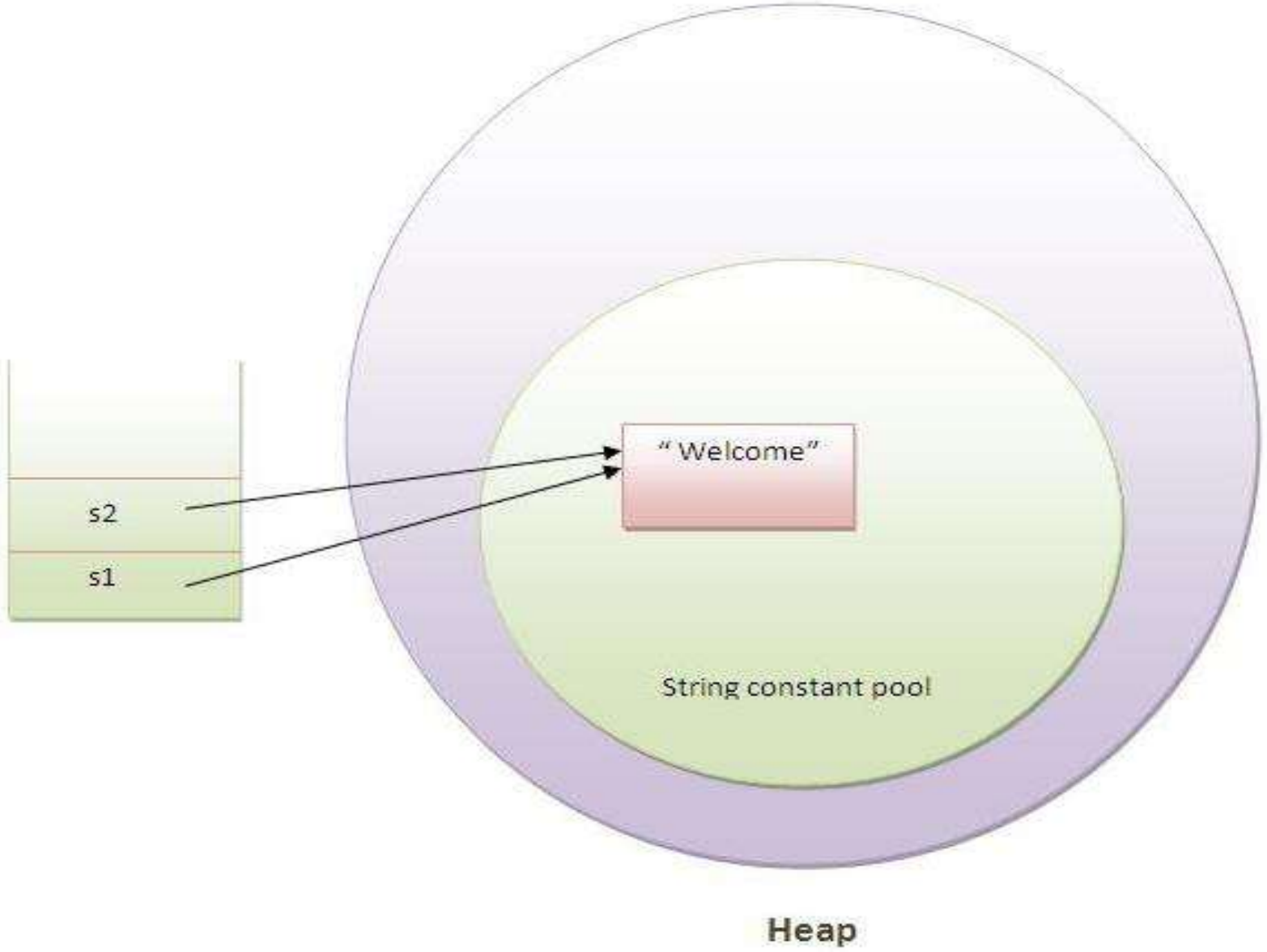
String Literal

- 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

By new keyword

String s=new String("Welcome");//creates two objects and on e reference variable

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 heap(non pool).

```
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
//creating java string by new keyword
String s3=new String("example");
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}
}
```

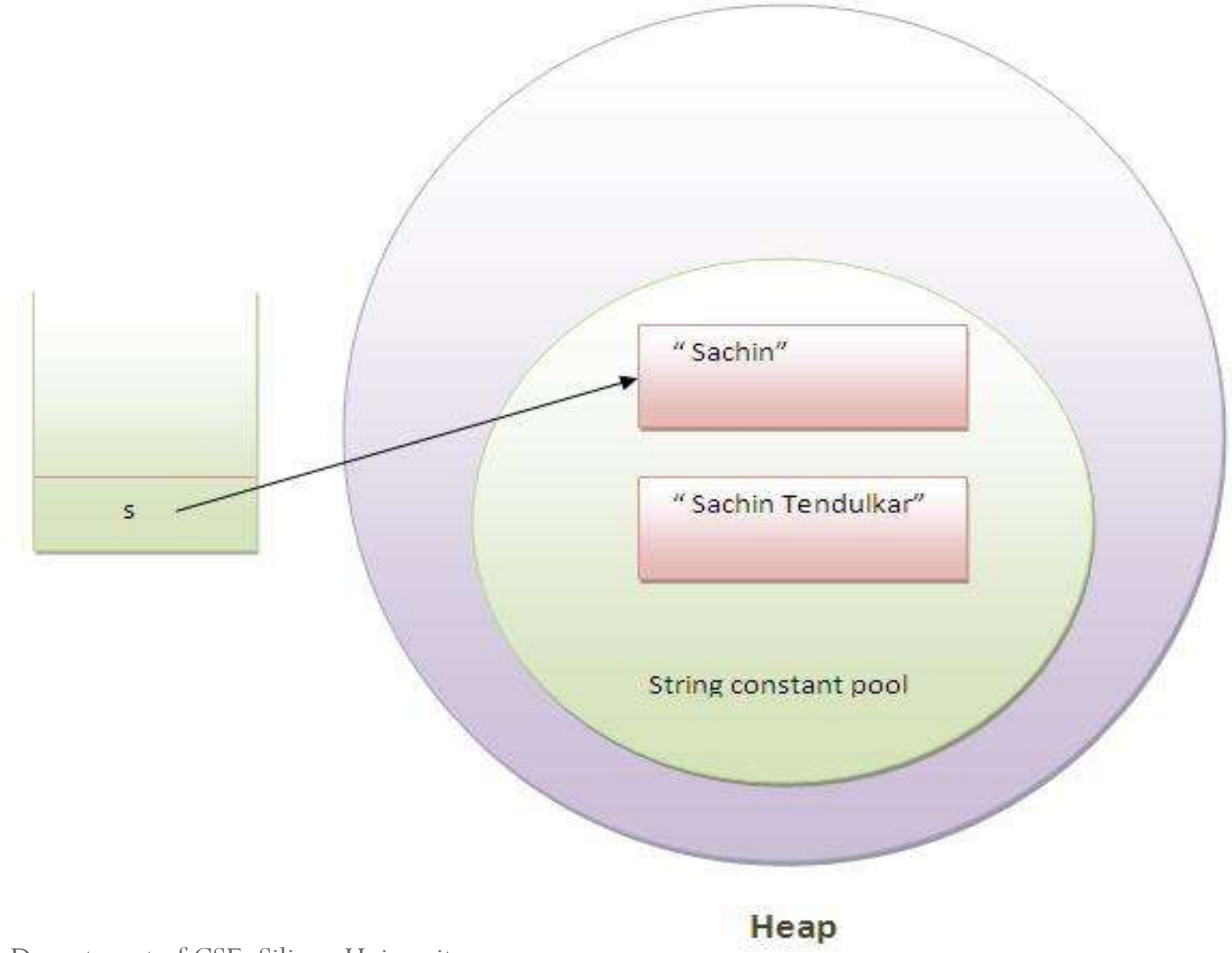
O/P:-**java strings example**

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.

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

Output: Sachin


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

String compare

- We can compare string in java on the basis of content and reference.
- There are three ways to compare string in java:

1. By equals() method
2. By == operator
3. By compareTo() method

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

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");
//true (because both refer to same instance)
   System.out.println(s1==s2);
   System.out.println(s1==s3);//false(because s3 refers to instance c
reated in nonpool)
 }
}
```

- **Output**: true false

By compareTo()

- 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

- 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

Concatenation by + operator

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

- **Output**: Sachin Tendulkar

Concatenation by concat() method

```
class TestStringConcatenation3{
 public static void main(String args[]){
   String s1="Sachin ";
   String s2="Tendulkar";
   String s3=s1.concat(s2);
   System.out.println(s3);//Sachin Tendulkar
  }
}
```

- **Output**: Sachin Tendulkar

Substring

- A part of string is called **substring**. In other words, substring is a subset of another string. In case of substring startIndex is inclusive and endIndex is exclusive.
- **public String substring(int startIndex):** This method returns new String object containing the substring of the given string from specified startIndex (inclusive).
- **public String substring(int startIndex, int endIndex):** This method returns new String object containing the substring of the given string from specified startIndex to endIndex.

```
public class TestSubstring{
 public static void main(String args[]){
   String s="Sachin Tendulkar";
   System.out.println(s.substring(6));//Tendulkar
   System.out.println(s.substring(0,6));//Sachin
 }
}
```

- O/P: Tendulkar Sachin

String Methods

toUpperCase() and toLowerCase()

The java string toUpperCase() method converts this string into uppercase letter and string toLowerCase() method into lowercase letter.

- String s="Sachin";
- System.out.println(s.toUpperCase());//SACHIN
- System.out.println(s.toLowerCase());//sachin
- System.out.println(s);//Sachin(no change in original)

O/P: SACHIN sachin Sachin

String trim() method

The string trim() method eliminates white spaces before and after string.

- String s=" Sachin ";
- System.out.println(s);// Sachin
- System.out.println(s.trim());//Sachin

O/P: Sachin

Sachin

startsWith() and endsWith() method

- String s="Sachin";
- System.out.println(s.startsWith("Sa"));//true
- System.out.println(s.endsWith("n"));//true

O/P: true

true

startsWith() and endsWith() method

- String s="Sachin";
- System.out.println(s.startsWith("Sa"));//true
- System.out.println(s.endsWith("n"));//true

O/P: true

true

charAt() method

- The string charAt() method returns a character at specified index.
- String s="Sachin";
- System.out.println(s.charAt(0));//S
- System.out.println(s.charAt(3));//h

- O/P: s

h

length() method

- The string length() method returns length of the string.
- String s="Sachin";
- System.out.println(s.length());//6

- O/P: 6

intern() method

- A pool of strings, initially empty, is maintained privately by the class String.
- When the intern method is invoked, if the pool already contains a string equal to this String object as determined by the equals(Object) method, then the string from the pool is returned.
- Otherwise, this String object is added to the pool and a reference to this String object is returned
- String s=new String("Sachin");
- String s2=s.intern();
- System.out.println(s2);//Sachin

- O/P: Sachin

valueOf() method

- The string valueOf() method coverts given type such as int, long, float, double, boolean, char and char array into string.

```
int a=10;
String s=String.valueOf(a);
System.out.println(s+10);
```

O/P: 1010

replace() method

- The string replace() method replaces all occurrence of first sequence of character with second sequence of character.

- String s1="Java is a programming language. Java is a platform. Java is an Island.";
- String replaceString=s1.replace("Java","Kava");//replaces all occurrences of "Java" to "Kava"
- System.out.println(replaceString);

O/P: Kava is a programming language. Kava is a platform. Kava is an Island.

Thank you