String?

- 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','t','p','o','i','n','t'};
- String s=**new** String(ch); is same as:
- String s="javatpoint";

- **Java String** class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.
- The java String is immutable i.e. it cannot be changed.
- Whenever we change any string, a new instance is created.
- For mutable string, we can use StringBuffer and StringBuilder classes.

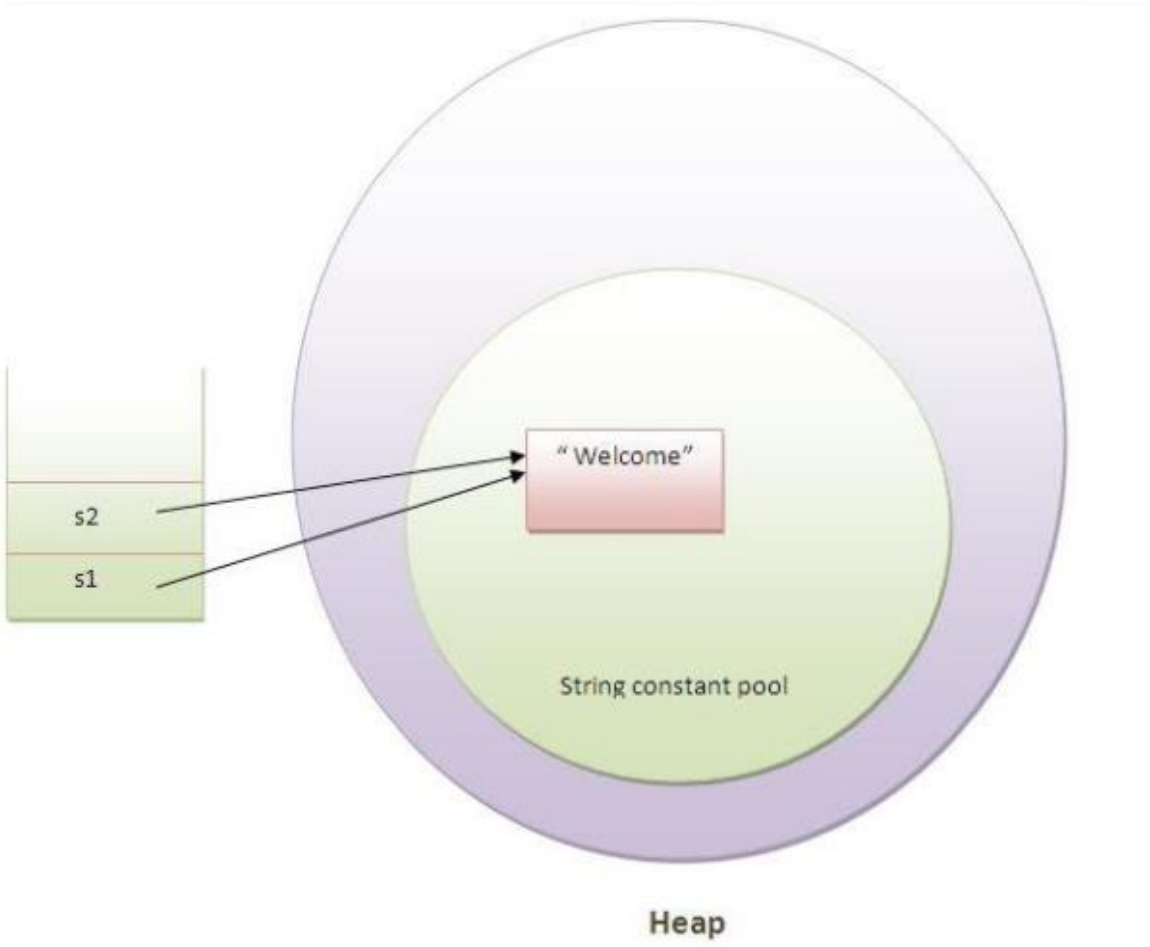
- The java.lang.String class is used to create string object.
- **How to create String object?**
- There are **two** ways to create String object:

1. By string literal
2. 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

String objects are stored in a special memory area known as string constant pool.



Why java uses concept of string literal?

- To make Java more memory efficient (because no new objects are created if it exists already in string constant pool).

2) 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 heap(non pool).

Java String Example

```
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);
}}
```

java
Strings
example

Java String class 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 formatted string
4	static String format(Locale l, String format, Object... args)	returns formatted string with given locale
5	String substring(int beginIndex)	returns substring for given begin index
6	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index
7	boolean contains(CharSequence s)	returns true or false after matching the sequence of char value

8	static String join(CharSequence delimiter, CharSequence... elements)	returns a joined string
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	returns a joined string
10	boolean equals(Object another)	checks the equality of string with object
11	boolean isEmpty()	checks if string is empty
12	String concat(String str)	concatinates specified string
13	String replace(char old, char new)	replaces all occurrences of specified char value
14	String replace(CharSequence old, CharSequence new)	replaces all occurrences of specified CharSequence

15	static String equalsIgnoreCase(String another)	compares another string. It doesn't check case.
16	String[] split(String regex)	returns splitted string matching regex
17	String[] split(String regex, int limit)	returns splitted string matching regex and limit
18	String intern()	returns interned string
19	int indexOf(int ch)	returns specified char value index
20	int indexOf(int ch, int fromIndex)	returns specified char value index starting with given index
21	int indexOf(String substring)	returns specified substring index
22	int indexOf(String substring, int fromIndex)	returns specified substring index starting with given index

23	String toLowerCase()	returns string in lowercase.
24	String toLowerCase(Locale l)	returns string in lowercase using specified locale.
25	String toUpperCase()	returns string in uppercase.
26	String toUpperCase(Locale l)	returns string in uppercase using specified locale.
27	String trim()	removes beginning and ending spaces of this string.
28	static String valueOf(int value)	converts given type into string. It is overloaded.

Immutable String in Java

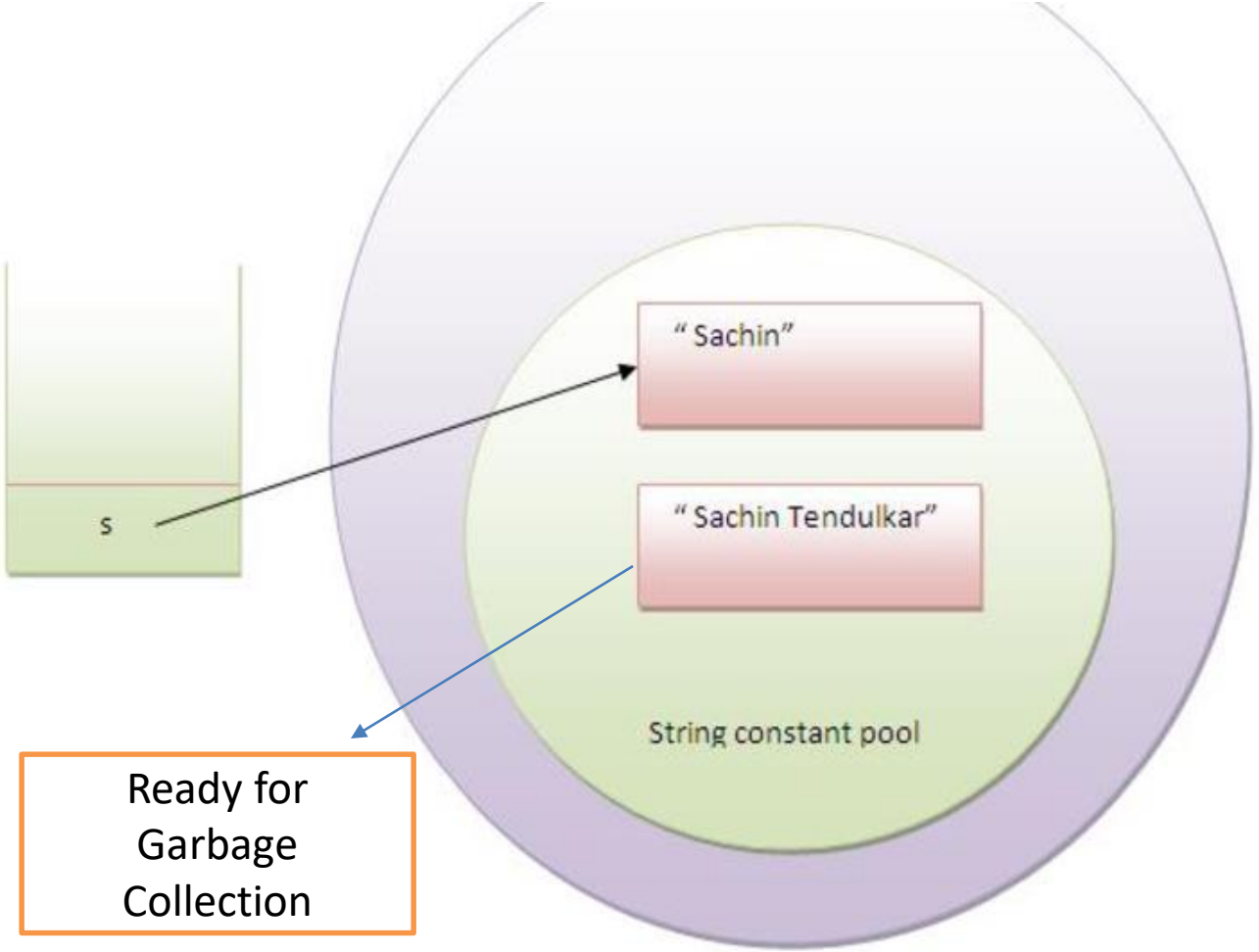
- In java, **string objects are immutable**. Immutable simply means unmodifiable or unchangeable.
- Once string object is created its data or state can't be changed but a new string object is created.

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 s
trings are immutable objects
 }
}
```

Sachin

Here Sachin is not changed but a new object is created with sachintendulkar. That is why string is known as immutable.



Example

```
class Testimmutablestring1{
 public static void main(String args[]){
   String s="Sachin";
   s=s.concat(" Tendulkar");
   System.out.println(s);
 }
}
```

Output:
Sachin Tendulkar

Example

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

    // create strings
    String first = "Java";
    String second = "Python";
    String third = "JavaScript";

    // print strings
    System.out.println(first);   // print Java
    System.out.println(second);  // print Python
    System.out.println(third);   // print JavaScript
  }
}
```

length of a String using length()

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

    // create a string
    String greet = "Hello! World";
    System.out.println("String: " + greet);

    // get the length of greet
    int length = greet.length();

System.out.println("Length: " + length);
  }
}
```

String: Hello! World
Length: 12

Java String compare

- We can compare string in java on the basis of content and reference.
- It is used in **authentication** (by equals() method),
- **sorting** (by compareTo() method),
- **reference matching** (by == operator) etc.
- There are three ways to compare string in java:

  By equals() method

  By = = operator

  By compareTo() method

1) String compare by equals() method

```
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
 }
}
```

Example2

```
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

 }
}
```

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 nonpool)
 }
}
```

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

Example

```
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 )

 }
}
```

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:

1. By + (string concatenation) operator
2. By concat() method

1) String Concatenation by + (string concatenation) operator

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

Example

```
class TestStringConcatenation2{
 public static void main(String args[]){
   String s=50+30+"Sachin"+40+40;
   System.out.println(s);//80Sachin4040
 }
}
```

2) String 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
  }
}
```