**Java Arrays**

Normally, an array is a collection of similar type of elements which has contiguous memory location.

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

**Advantages**

Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.

Random access: We can get any data located at an index position.

**Disadvantages**

Size Limit: We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically

**There are two types of array.**

1. Single Dimensional Array
2. Multidimensional Array

**Syntax to Declare an Array in Java**

dataType[] arr; (or)

dataType []arr; (or)

dataType arr[];

**Instantiation of an Array in Java**

datatype []arrayRefVar=new datatype[size];

**Example of Java Array**

//Java Program to illustrate how to declare, instantiate, initialize

//and traverse the Java array.

class Testarray{

public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//traversing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

**Declaration, Instantiation and Initialization of Java Array**

int a[]={33,3,4,5};//declaration, instantiation and initialization

**Let's see the simple example to print this array.**

class Testarray1{

public static void main(String args[]){

int a[]={33,3,4,5};//declaration, instantiation and initialization

//printing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

**For-each Loop for Java Array**

We can also print the Java array using for-each loop. The Java for-each loop prints the array elements one by one. It holds an array element in a variable, then executes the body of the loop.

for(data\_type variable:array){

//body of the loop

}

**Let us see the example of print the elements of Java array using the for-each loop.**

class Testarray1{

public static void main(String args[]){

int arr[]={33,3,4,5};

//printing array using for-each loop

for(int i:arr)

System.out.println(i);

}}

**Passing Array to a Method in Java**

class Testarray2{

//creating a method which receives an array as a parameter

static void min(int arr[]){

int min=arr[0];

for(int i=1;i<arr.length;i++)

if(min>arr[i])

min=arr[i];

System.out.println(min);

}

public static void main(String args[]){

int a[]={33,3,4,5};//declaring and initializing an array

min(a);//passing array to method

}}

**Multidimensional Array in Java**

**Syntax to Declare Multidimensional Array in Java**

dataType[][] arrayRefVar; (or)

dataType [][]arrayRefVar; (or)

dataType arrayRefVar[][]; (or)

dataType []arrayRefVar[];

**Example to instantiate Multidimensional Array in Java**

int[][] arr=new int[3][3];//3 row and 3 column

**Example to initialize Multidimensional Array in Java**

arr[0][0]=1;

arr[0][1]=2;

arr[0][2]=3;

arr[1][0]=4;

arr[1][1]=5;

arr[1][2]=6;

arr[2][0]=7;

arr[2][1]=8;

arr[2][2]=9;

**Example of Multidimensional Java Array**

class Testarray3{

public static void main(String args[]){

//declaring and initializing 2D array

int arr[][]={{1,2,3},{2,4,5},{4,4,5}};

//printing 2D array

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

System.out.print(arr[i][j]+" ");

}

System.out.println();

}

}}

**Java String**

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={‘A’,’B’,’C’};

String s=new String(ch);

**is same as:**

String s="ABC";

Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

**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.

**How to create a 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";

**2) By new keyword**

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

**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);

}}

**String Methods**

String charAt()

String compareTo()

String concat()

String contains()

String endsWith()

String equals()

equalsIgnoreCase()

String format()

String getBytes()

String getChars()

String indexOf()

String intern()

String isEmpty()

String join()

String lastIndexOf()

String length()

String replace()

String replaceAll()

String split()

String startsWith()

String substring()

String toCharArray()

String toLowerCase()

String toUpperCase()

String trim()

String valueOf()

**Few String methos Examples**

public class EqualsExample{

public static void main(String args[]){

String s1="java";

String s2="java";

String s3="JAVA";

String s4="python";

System.out.println(s1.equals(s2));//true because content and case is same

System.out.println(s1.equals(s3));//false because case is not same

System.out.println(s1.equals(s4));//false because content is not same

}}

public class LengthExample{

public static void main(String args[]){

String s1="java";

String s2="python";

System.out.println("string length is: "+s1.length());//10 is the length of javatpoint string

System.out.println("string length is: "+s2.length());//6 is the length of python string

}}

public class SplitExample{

public static void main(String args[]){

String s1="java string split method by";

String[] words=s1.split("\\s");//splits the string based on whitespace

//using java foreach loop to print elements of string array

for(String w:words){

System.out.println(w);

}

}}

public class StringTrimExample{

public static void main(String args[]){

String s1=" hello string ";

System.out.println(s1+"Variable");//without trim()

System.out.println(s1.trim()+"Variable ");//with trim()

}}

**Why String is Immutable?**

When we create a string in java like String s1="hello"; then an object will be created in string pool(hello) and s1 will be pointing to hello.Now if again we do String s2="hello"; then another object will not be created but s2 will point to hello because JVM will first check if the same object is present in string pool or not.If not present then only a new one is created else not.

Now if suppose java allows string mutable then if we change s1 to “hello world” then s2 value will also be “hello world” so java String is immutable.

String s1 = “hello” // String s1 = new String(“hello”);

String s2 = “hello”

String s3 = “hello”

String s4 = “hello”

String s5 = “hello”

String s6 = “hello”

class Testimmutablestring{

public static void main(String args[]){

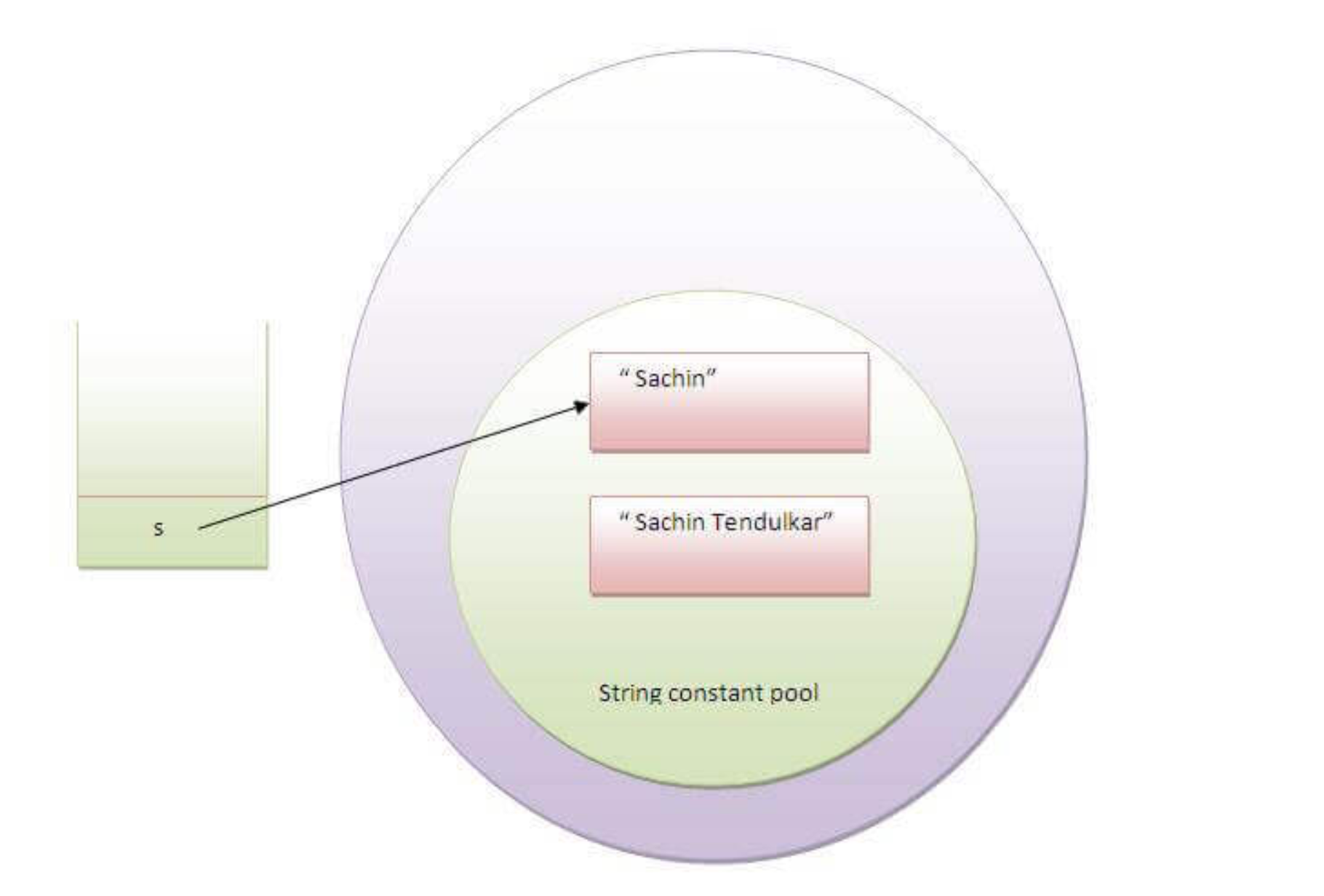
String s="Sachin";

s.concat(" Tendulkar");//concat() method appends the string at the end

System.out.println(s);//will print Sachin because strings are immutable objects

}

}



**Example to create Immutable class**

In this example, we have created a final class named Employee. It have one final datamember, a parameterized constructor and getter method.

public final class Employee{

final String pancardNumber;

public Employee(String pancardNumber){

this.pancardNumber=pancardNumber;

}

public String getPancardNumber(){

return pancardNumber;

}

}

**The above class is immutable because:**

The instance variable of the class is final i.e. we cannot change the value of it after creating an object.

The class is final so we cannot create the subclass.

There is no setter methods i.e. we have no option to change the value of the instance variable.

These points makes this class as immutable.

**Singleton class**

class Database {

private static Database dbObject;

private Database() {

}

public static Database getInstance() {

// create object if it's not already created

if(dbObject == null) {

dbObject = new Database();

}

// returns the singleton object

return dbObject;

}

public void getConnection() {

System.out.println("You are now connected to the database.");

}

}

class Main {

public static void main(String[] args) {

Database db1;

// refers to the only object of Database

db1= Database.getInstance();

db1.getConnection();

}

}