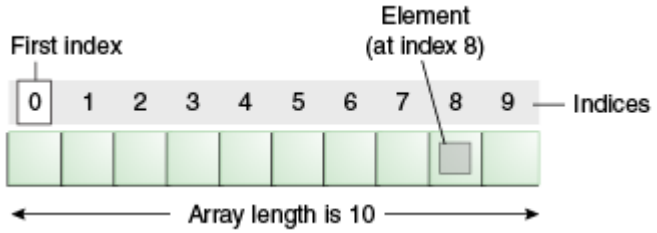
Array

- Array is a collection of similar type of elements that have contiguous memory location.
- **Java array** is an object the contains elements of similar data type.
- We can store only fixed set of elements in a java array.
- Array in java is index based, first element of the array is stored at 0 index.



- Disadvantage of Java Array: Size Limit
- There are two types of array.

    1) Single Dimensional Array

    2) Multidimensional Array

Array…

- **Declareation of an array:**
  - dataType[] arr; (or) dataType []arr; (or) dataType arr[];
- **Instantiation:** arr=new datatype[size];
- **Example of single dimensional 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;
//printing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

Array…

- You can declare, instantiate and initialize the java array together by:
  - int a[]={33,3,4,5};//declaration, instantiation and initialization
- **Example Passing Array to method**

```
class Testarray2{
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};
min(a);//passing array to method
}}
```

•;

Array…

- **Multidimensional array**
- **Declaration:**

  dataType[][] arrayRefVar; (or)
  dataType [][]arrayRefVar; (or)
  dataType arrayRefVar[][]; (or)
  dataType []arrayRefVar[];

- **instantiate Multidimensional**
  - int[][] arr=new int[3][3];//3 row and 3 column
- **initialize Multidimensional Array**

```
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;
```

Array...

- **Example of Multidimensional 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();
}
}}
```

Array...

- **What is the class name of java array?**
- In java, array is an object. For array object, an proxy class is created whose name can be obtained by getClass().getName() method on the object.

```
class Testarray4{
public static void main(String args[]){

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

Class c=arr.getClass();
String name=c.getName();

System.out.println(name);

}}
```

Object class

- The **Object class** is the parent class of all the classes in java by default.
- The Object class is beneficial if you want to refer any object whose type you don't know.
- There is getObject() method that returns an object but it can be of any type like Employee,Student etc.
- We can use Object class reference to refer that object.
- Example:

```
Object obj=getObject();//we don't know what object will be returned from this method
```

Methods of Object class

- public final Class getClass(): returns the Class class object of this object.
- public int hashCode(): returns the hashcode number for this object.
- public boolean equals(Object obj): compares the given object to this object.
- protected Object clone() throws CloneNotSupportedException: creates and returns the exact copy (clone) of this object.
- public String toString(): returns the string representation of this object.
- public final void notify(): wakes up single thread, waiting on this object's monitor.
- public final void notifyAll(): wakes up all the threads, waiting on this object's monitor.

Methods of Object class…

- public final void wait(long timeout)throws InterruptedException: causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method).
- public final void wait(long timeout,int nanos)throws InterruptedException: causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method).
- public final void wait()throws InterruptedException: causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method).
- protected void finalize()throws Throwable: is invoked by the garbage collector before object is being garbage collected.

Object Cloning

- **object cloning** is a way to create exact copy of an object.
- The clone() method of Object class is used to clone an object.
- The **java.lang.Cloneable interface** must be implemented by the class whose object clone we want to create.
- If we don't implement Cloneable interface, clone() method generates **CloneNotSupportedException**.
- The **clone() method** is defined in the Object class. Syntax of the clone() method is as follows:
  - protected Object clone() throws CloneNotSupportedException

Object Cloning…

```
class Student18 implements Cloneable{
int rollno;
String name;

Student18(int rollno,String name){
this.rollno=rollno;
this.name=name;
}
public Object clone()throws CloneNotSupportedException{
return super.clone();
}

public static void main(String args[]){
try{
Student18 s1=new Student18(101,"amit");

Student18 s2=(Student18)s1.clone();

System.out.println(s1.rollno+" "+s1.name);
System.out.println(s2.rollno+" "+s2.name);

}catch(CloneNotSupportedException c){}
}
}
```

Wrapper class

- **Wrapper class in java** provides the mechanism *to convert primitive into object and object into primitive*.
- The automatic conversion of primitive into object is known as autoboxing and vice-versa unboxing.
- J2SE 5.0 include **autoboxing** and **unboxing** feature.

Primitive Type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Wrapper class…

- **Wrapper class Example: Primitive to Wrapper**

```
public class WrapperExample1{
public static void main(String args[]){
//Converting int into Integer
int a=20;
Integer i=Integer.valueOf(a);//converting int into Integer

Integer j=a;//autoboxing, now compiler will write Integer
.valueOf(a) internally

System.out.println(a+" "+i+" "+j);
}}
```

Wrapper class…

- **Wrapper class Example: Wrapper to Primitive**

```
public class WrapperExample2{
public static void main(String args[]){
//Converting Integer to int
Integer a=new Integer(3);
int i=a.intValue();//converting Integer to int
int j=a;//unboxing, now compiler will write a.intValue
() internally

System.out.println(a+" "+i+" "+j);
}}
```

Strictfp Keyword

- strictfp keyword ensures that you will get the same result on every platform if you perform operations in the floating-point variable.
- The precision may differ from platform to platform that is why java programming language have provided the strictfp keyword.
- The strictfp keyword can be applied on methods, classes and interfaces.
- The strictfp keyword **cannot** be applied on abstract methods, variables or constructors.

javadoc tool

- We can create document api in java by the help of **javadoc** tool.
- In the java file, we must use the documentation comment /**... */ to post information for the class, method, constructor, fields etc.
- Example:

```
package com.abc;
/** This class is a user-
defined class that contains one methods cube.*/
public class M{
/** The cube method prints cube of the given number */
public static void  cube(int n){System.out.println(n*n*n);}
}
```

- To create the document API, you need to use the javadoc tool followed by java file name. Example:

```
javadoc M.java
```

Command Line Arguments

- The java command-line argument is an argument i.e. passed at the time of running the java program.
- The arguments passed from the console can be received in the java program and it can be used as an input.
- Example:

```
class A{
public static void main(String args[]){
for(int i=0;i<args.length;i++)
System.out.println(args[i]);
}
}
```

Compile: javac A.java

run by > java A Sourav Sudip Sukhendu

Output: Sourav
Sudip
Sukhendu

object vs class

Object	Class
Object is an **instance** of a class.	Class is a **blueprint or template** from which objects are created.
Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a **group of similar objects**.
Object is a **physical** entity.	Class is a **logical** entity.
Object is created through **new keyword** mainly e.g. Student s1=new Student();	Class is declared using **class keyword** e.g. class Student{}
Object is created **many times** as per requirement.	Class is declared **once**.
Object **allocates memory when it is created**.	Class **doesn't allocated memory when it is created**.
There are **many ways to create object** in java such as new keyword, newInstance() method, clone() method and factory method.	There is only **one way to define class** in java using class keyword.

Method overloading vs Method overriding

Method Overloading	Method Overriding
Method overloading is used to increase the readability of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.
Method overloading is performed within class.	Method overriding occurs in two classes that have IS-A (inheritance) relationship.
In case of method overloading, parameter must be different.	In case of method overriding, parameter must be same.
Method overloading is the example of compile time polymorphism.	Method overriding is the example of run time polymorphism.
In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in method overloading. But you must have to change the parameter.	Return type must be same or covariant in method overriding.