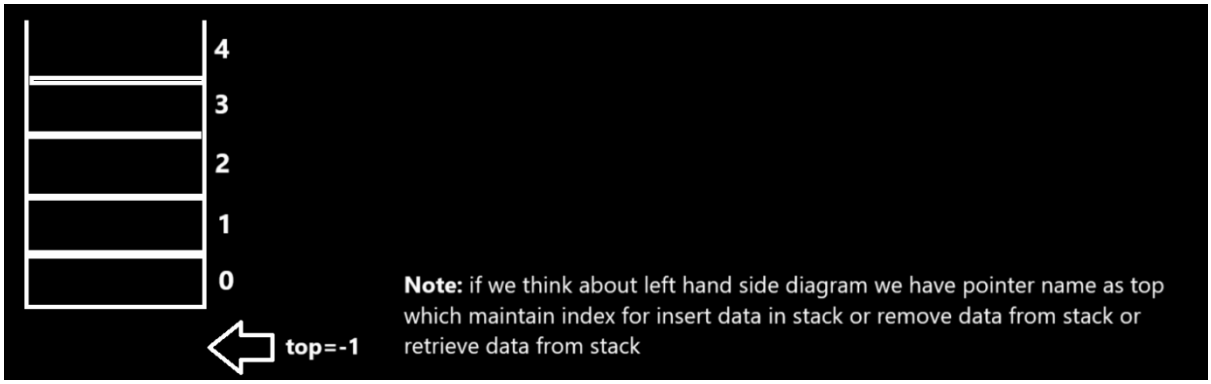
Stack and Queue

Q. What is stack?

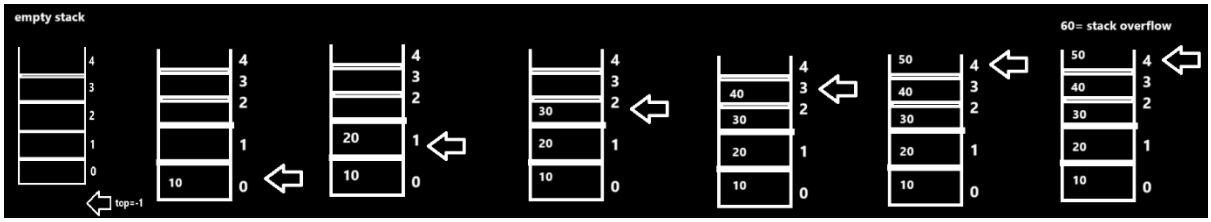
Stack is a data structure or it is single ended data structure which is used for manage the data in last in first out format means last inserted data return first and first inserted data return last called as stack.
if we think about stack we can insert data in stack using same end and retrieve data from stack from same end.

Following diagram shows the conceptual diagram of stack

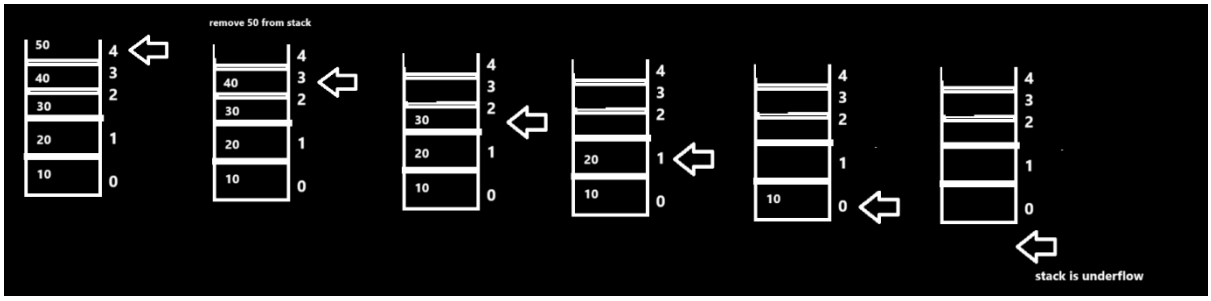


There are two ways to implement stack
1. Using array
2. Using linked list

If we want to work with stack we can perform three operations on stack
1. PUSH: push operation help us to insert data in stack and when insert data in stack then top of stack increase by 1 and initially top value is -1 but if we think about push operation of stack then there is possibility stack overflow.
Stack overflow means if we try to insert value in stack but there is no space for insert data in stack called as stack overflow show in following diagrams



2. POP: if we remove the data from stack called as pop operation and when we remove the data from stack then top decrease by 1 and in the case pop operation there is possibility of stack empty means when we have no element in stack for remove called as empty stack.



Now we want to design algorithm for push and pop

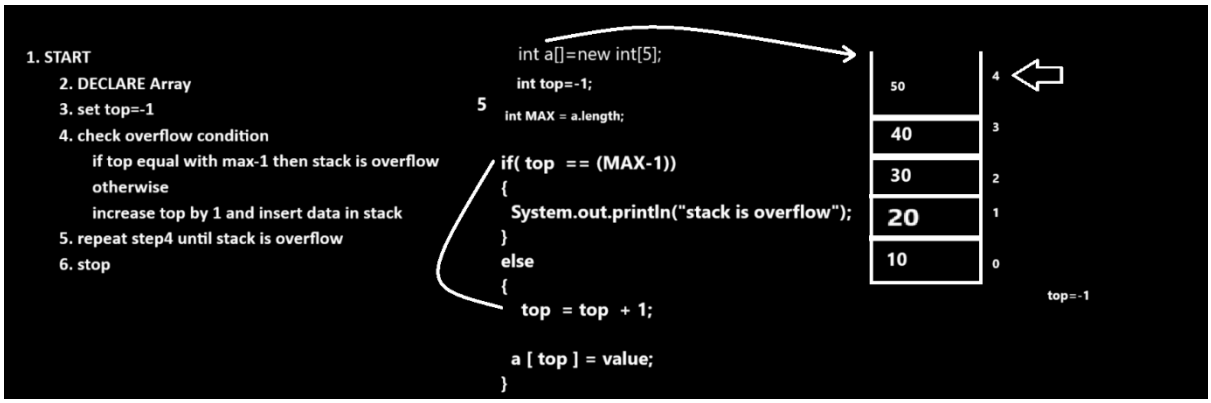
Algorithm for push and pop

PUSH Algorithm

---

1. **START**
2. **DECLARE Array**
3. **Set top=-1**
4. **Check overflow condition**
   - **if top equal with max-1 then stack is overflow**
   - **otherwise**
   - **increase top by 1 and insert data in stack**
5. **Repeat step4 until stack is overflow**
6. **Stop**

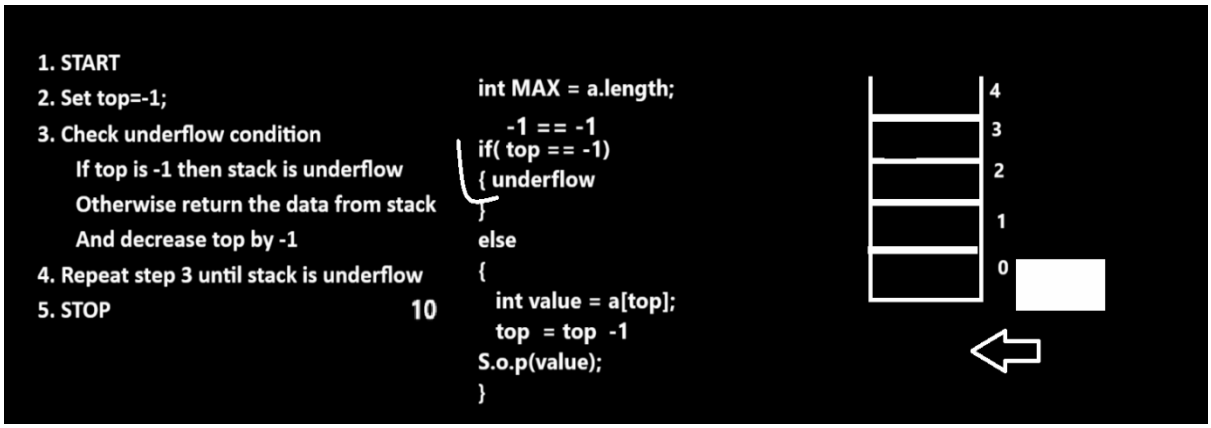
Example with source code



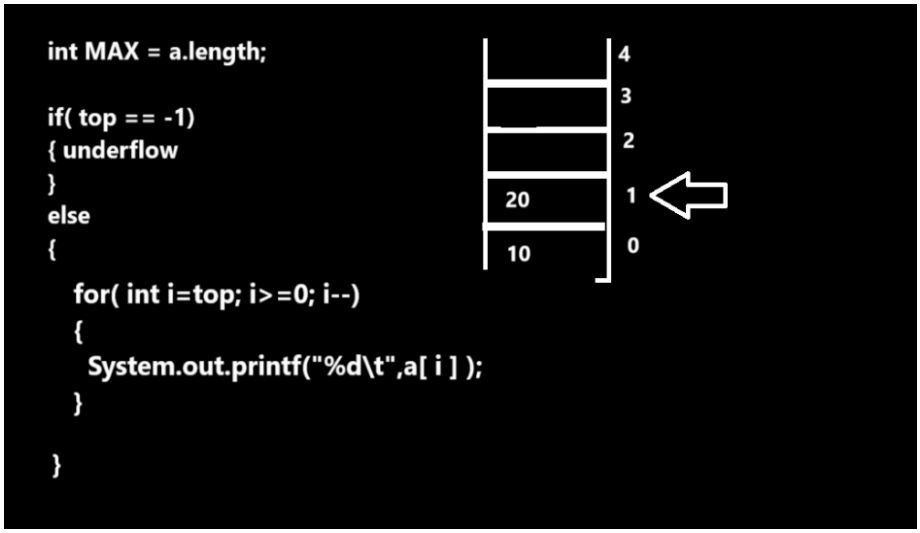
Algorithm for POP operation

---

1. START
2. Set top=-1;
3. Check underflow condition
If top is -1 then stack is underflow
Otherwise return the data from stack
And decrease top by -1
4. Repeat step 3 until stack is underflow
5. STOP



3. Display : if we want to fetch data then we required iterate loop from top to 0



Example with source code

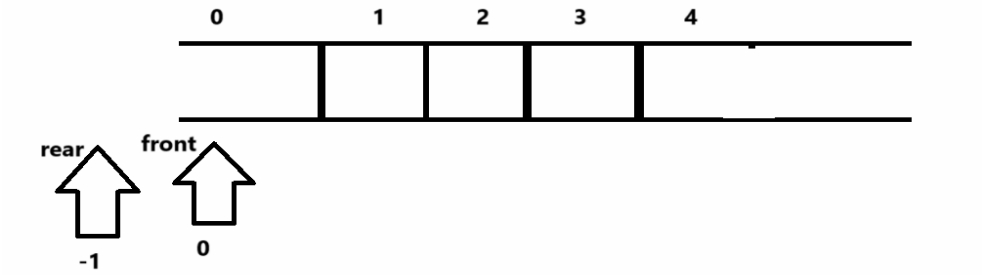
```
import java.util.*;
public class StackApp
{  static int top=-1;
  static int stack[]=new int[5];// array as stack
   public static void main(String x[])
        {  Scanner xyz  = new Scanner(System.in);
          do
          {
           System.out.println("1:PUSH");
                System.out.println("2:POP");
                System.out.println("3:DISPLAY");
                System.out.println("Enter your choice");
                int choice=xyz.nextInt();
                switch(choice)
                {
                  case 1:
                  System.out.println("Enter value in stack");
                  int value=xyz.nextInt();
                  push(value);//call push for push element in stack
                  break;
                  case 2:
                  pop();
                  break;
                  case 3:
                  display();
                  break;
                  default:
                   System.out.println("Wrong choice");
                }

          }while(true); // infinite loop
        }
        public static void push(int value)
        {
          int MAX=stack.length;
          if(top ==MAX-1)
```

```
    {
        System.out.println("Stack is overflow");
    }
    else
    {
     top = top+1;
          stack[top]=value;
    }
  }
  public static void pop()
  {
     if(top==-1)
          { System.out.println("Stack is underflow");
          }
    else{
       int value=stack[top];
            top=top-1;
            System.out.println("Removed data is  "+value);
    }
  }
  public static void display()
  {
    if(top==-1)
    { System.out.println("Stack is underflow");
    }
    else
    {
      for(int i=top;i>=0;i--)
           { System.out.printf("%d\n",stack[i]);
           }
    }
  }

}
```

Queue

Queue is data structure which is used for maintain data or store data in first in first out format
And it is two ended data structure and it has two pointers one is used for insert data known as rear and one is used for remove data known as front

Following diagram shows the queue



If we think about Queue we have two operations we can perform on Queue

1. Insert: Insert operation means we can store data in Queue and when we insert data in Queue then rear pointer of queue increase by 1 and initially rear pointer set by -1 and if we think about insert operation of Queue then there is possibility of Queue full means when we have no space for store data in queue then we can say queue is full.

2. Delete: delete operation means we can remove data from queue and when we remove the data from queue then front of queue increase by 1 and initially we initialize front=0 and in delete operation there is possibility of Queue underflow or empty means when we have no element for remove element from queue called as delete operation on queue.

Algorithm for insert, delete and display operation of Queue

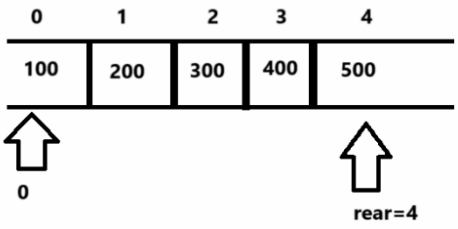
Algorithm of insert operation on Queue
1. START
2. Initialize rear=-1 and front=0
3. Check Queue is full condition
If rear ==MAX-1 then Queue is full
else or otherwise
rear=rear+1

4. Repeat step 3 until queue is full

5. STOP

1. Declare array for queue
2. Initialize rear=-1 and front=0
3. Check Queue is full condition
   If rear ==MAX-1 then Queue is full
   else or otherwise
      rear=rear+1
4. Repeat step 3 until queue is full
5. STOP

```
int queue[5]
int rear=-1,front=0
```



-1

5

```
int MAX=queue.length;

if( rear == (MAX-1))
{
   System.out.println("Queue is full");
}
else
{
   rear = rear + 1;
  queue[rear] = value;
}
```

Deletion Algorithm of Queue

---

1. START

2. SET rear=-1 and front=0

3. Check queue is underflow or empty condition

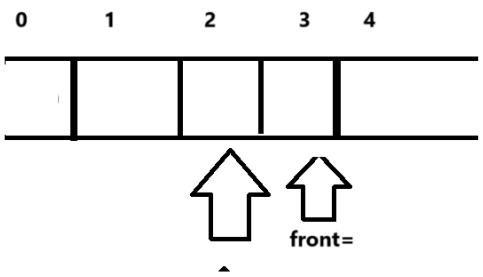
if((rear==-1 and front =0) ||(front==rear+1))

queue is empty

4. Otherwise remove data from queue and increase front by 1

5. STOP

1. START
2. SET rear=-1 and front=0
3. Check queue is underflow or empty condition
   if((rear==-1 and front =0) ||(front==rear+1))
   queue is empty
4. Otherwise remove data from queue and
   increase front by 1
5. STOP




```
if(   (rear==-1 && front=0) ||    (front==rear+1) )
{  System.out.println("Queue is empty");
}
else
{
  int value= queue[front];
   front=front+1;
 System.out.printf("%d\t",value);
}
```

rear=-1

100
200
300

Example with source code

```
import java.util.*;
public class QueueApp
{ static int queue[]=new int[5];
  static int front=0,rear=-1;
   public static void main(String x[])
   {
     do
          {
             Scanner xyz  = new Scanner(System.in);
                  System.out.println("1:INSERT");
                  System.out.println("2:DELETE");
                  System.out.println("3:DISPLAY");
                  System.out.println("Enter your choice");
                  int choice=xyz.nextInt();
                  switch(choice)
                  {
                    case 1:
                         System.out.println("Enter data in queue");
                         int data=xyz.nextInt();
                         insert(data);
                         break;
                         case 2:
                         delete();
                         break;
                         case 3:
                         display();
                         break;
                         default:
                         System.out.println("Wrong choice");
                  }
          }while(true);
   }
   public static void insert(int value)
   { int MAX=queue.length;
    if(rear==(MAX-1))
          { System.out.println("Queue is full");
```

```
        }
        else
        { rear=rear+1;
          queue[rear]=value;
        }
  }
  public static void delete()
  {
    if((rear==-1 &&front==0) ||(front==rear+1))
        { System.out.println("Queue is empty");
        }
        else
        {
          int value=queue[front];
              front=front+1;
              System.out.println("Deleted data is  "+value);
        }
  }
  public static void display()
  {  if((rear==-1 &&front==0) ||(front==rear+1))
        { System.out.println("Queue is empty");
        }
        else
        {for(int i=front; i<=rear; i++)
         {
          System.out.printf("%d\t",queue[i]);
         }
        }
  }
}
```

Time Complexity & Big-O notation

Q. What is time complexity?

Time complexity is mathematical function which always takes some input and generates some result
In terms of time complexity this function accept input as processor operation

Q. Why we need to understand the time complexity of code?

---

1. With the help of time complexity we can understand better code.
2. Now days top product base company asks you what time complexity of code is.

Q. Pre-requisite knowledge for time complexity?

---

1. Basic coding knowledge
2. Need to understand the mathematical function
3. Known about the resource utilization

Q. What is resource?

Resources is part of system which is responsible decide the execution and working of system called as resources.

There are two types of resources

---

1. RAM
2. Processor

If we think about time complexity we required to know the process utilization

Example: Suppose consider we have two developer names A and B and we give problem statement XYZ to developer A and B means we give same problem statement to both developer and both developer provide solution to us suppose consider developer A solution execution in 3 sec and developer B solution execute in 5 sec here we consider developer A give better solution
But the challenge is if suppose Developer A has modern machine and Developer b has old style machine so there is possibility developer A solution may be not better take less time because of machine configuration and Developer B solution may be better but take more time because of machine configuration
So we cannot judge performance application based on execution time of program so if we want to solve this problem we need to identity the process operation required in developer A solution and Developer B solution

Means we can say if we want to calculate the time complexity of any problem statement we need to know the process operation of the application or need to calculate the processor operation in application.

How to calculate the process operation

If we want to calculate the processor operation you have to standardize it.

Rules for calculate processor in code

1. Every Assignment operator in application consider as 1 individual processor operation
2. Every Arithmetic operation in application consider as 1 individual processor operation like as + , -, *, / etc
3. Every Comparison operator like as <,> , <=, >= etc consider as 1 individual processor operation
4. Every return statement consider as 1 individual processor operation

Q. Why processor operation calculation is important when we want to calculate time complexity?

Processor operator System independent operation.

Now we want to discuss about the one example

---

suppose consider we have one example

solution 1:

```
int addMul(int m)
{  m = m+1;
   return m*5;
}
```

if we want to calculate the processor operation of above code we can calculate like as m = m + 1 here + consider 1 operation , = consider as 1 processor operation means m = m+1 container two process operation 1+ 1

Again we have statement return m*5 here return required 1 processor operation and * multiplication required 1 process operation means we can say return m*5 contain two processor operation so m=m+1 and return m*5 required total 4 processor operation

Suppose consider some one write code like as

solution B –

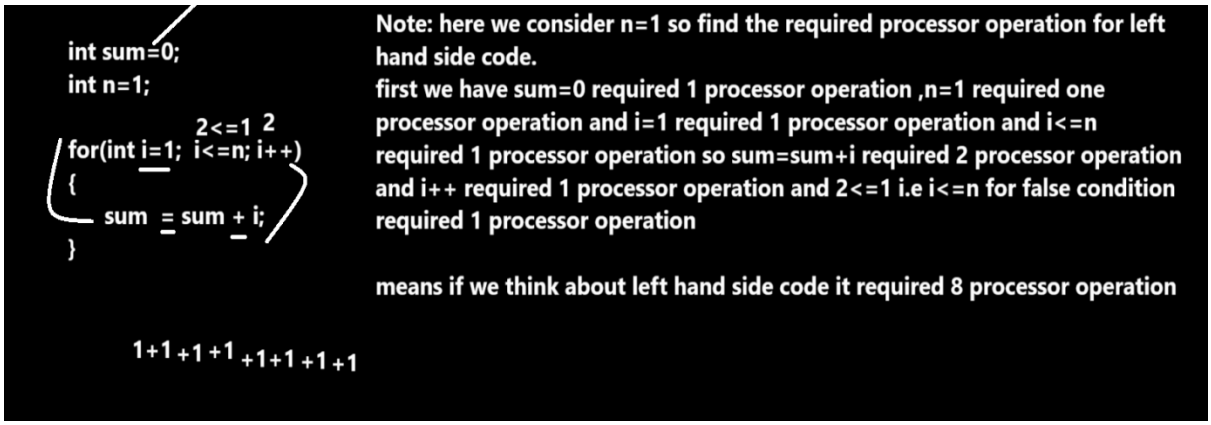
```
int addMul(int m)
{    return (m+1)*5;
}
```

if we think about above code we have statement return m+1*5 here return required 1 process operation , m+1 required 1 process operation and * multiplication required 1 processor operation so here we required 3 processor operation so when we compare both code then we can say solution B is better than A

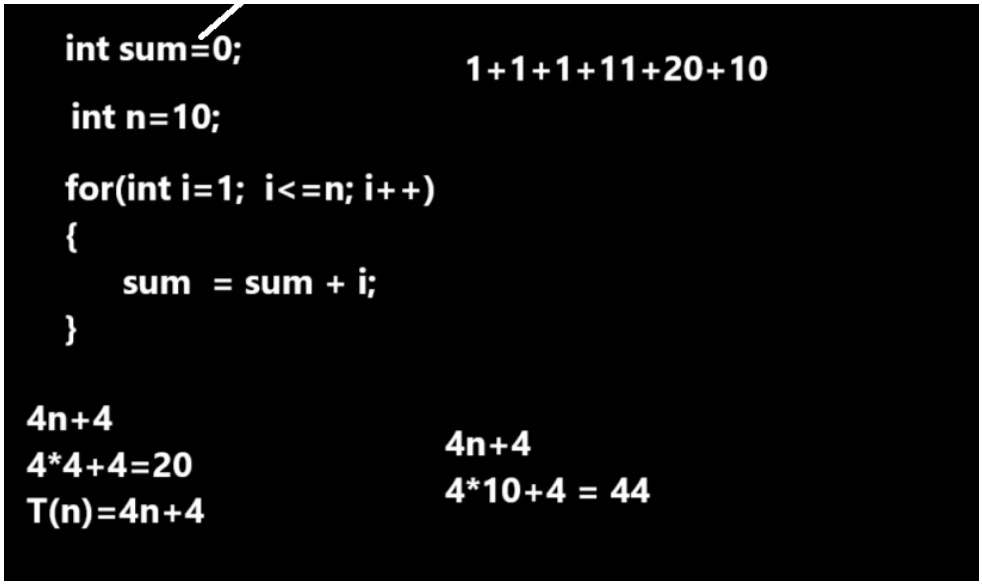
Note: if we think about both code in terms bigo notation we get same time complexity O(1) but we will discuss big o notation later in this chapter.

Now we want to discuss about one more example for calculating process operation

```
int sum=0;
for(int i=1; i<=n; i++)
{   sum = sum+i;
}
```



if we think about generalize terms for processor operation then you get result like as

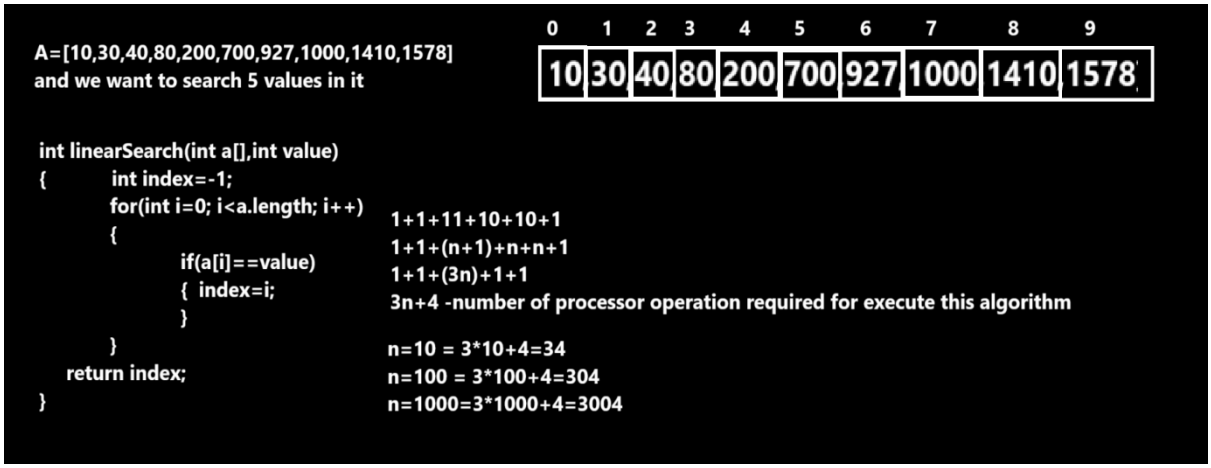


Now we want to discuss about two algorithms for comparison

1. Linear search

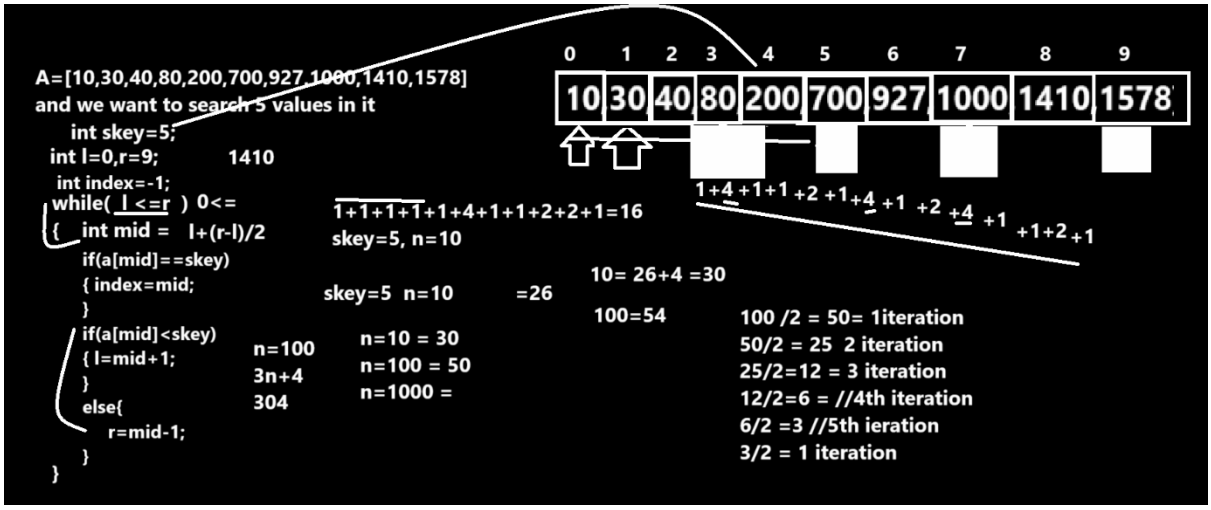
2. Binary search

Suppose consider we have following values in array and we want to search data in array



Above screen shot display the processor operation for linear search algorithm

Now we want to perform binary search operation



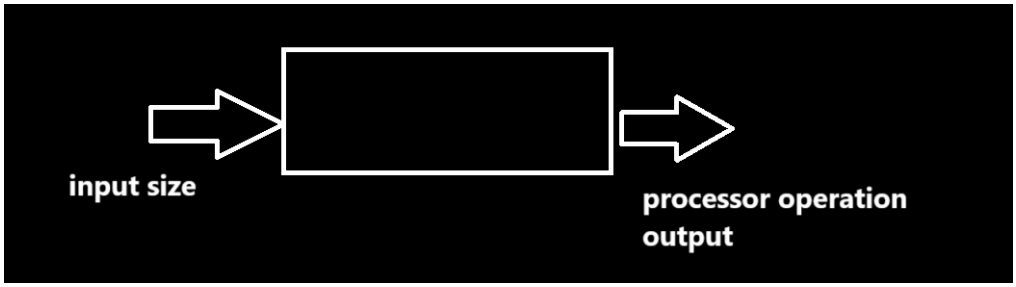
Q. What is time complexity?

Time complexity is mathematical function which takes some input and generates some output

If we think about this function in programming terms then input size is our input and processor operation is our output

So basically time complexity is denoted by T and input is denoted by n

so it represent like as t(n) if we think about previous mention input we can calculate processor but now we want to calculate this processor using mathematical functions or terms



time complexity logics

input size	Number of processor operation	Mathematical term
10	34	3*10+4 =3n+4
100	304	3*100+4= 3n+4
1000	3004	3*1000+4 =3n+4

means if we convert above logics in generalize format of time complexity according to mathematical function then it look like as

T(n)=3n+4

Now we want to discuss the Big O(notation) in time complexity

---

Wrapper classes

Q. What are wrapper classes?

Wrapper class is inbuilt API which help us to perform conversion in JAVA

Means using Wrapper classes we can perform conversion between primitive type to reference type and reference type to primitive type.

There are two types of conversion in JAVA?

---

1. **Primitive type casting :** primitive type casting means performing conversion between simple data types called primitive type casting.

There are two types of primitive type casting

a. Implicit type casting : implicit type casting means those conversions performed by the compiler internally called implicit type casting.

When we have larger data type at left hand side and smaller data type at right hand side then the compiler is able to perform conversion automatically called implicit conversion.

Example: long a;

int b=100;

a=b; // implicit conversion

```
public class ConApp
{ public static void main(String x[])
   {   int a=100;
       long b=a; //implicit conversion
       System.out.printf("B is %d\n",b);
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ConApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java ConApp
B is 100

C:\Program Files\Java\jdk1.8.0_291\bin>
```

b. **Explicit type casting :** explicit type casting means those conversions performed by developers are called explicit type casting.

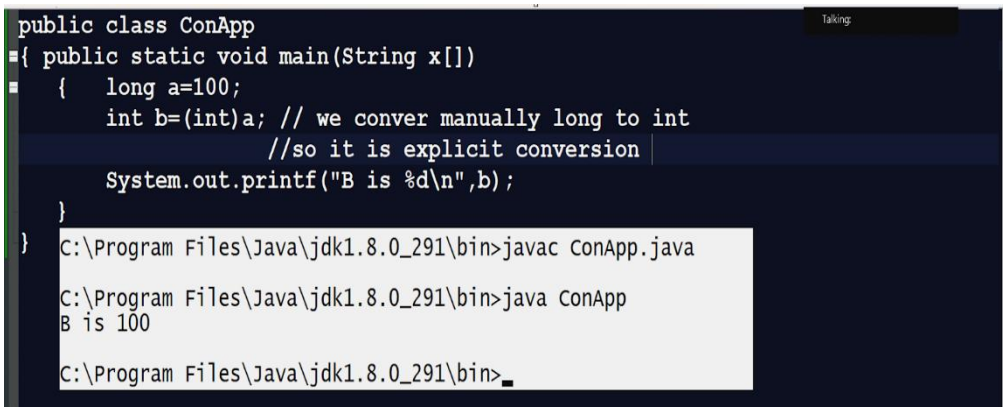
When we have a smaller type of data at left hand side and larger type of data at right hand side then compiler is unable to perform conversion automatically then developer has responsibility to perform conversion manually called as explicit conversion.

Example:

int a;

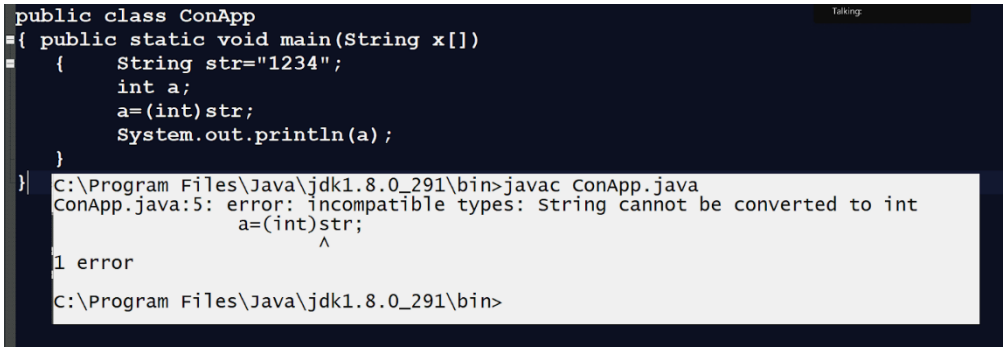
Long b=100;

a=(int)b;//explicit conversion



Note: implicit type casting and explicit type casting get failed when we try to convert referential value to primitive type of value and primitive type of value to referential type of value

Example:

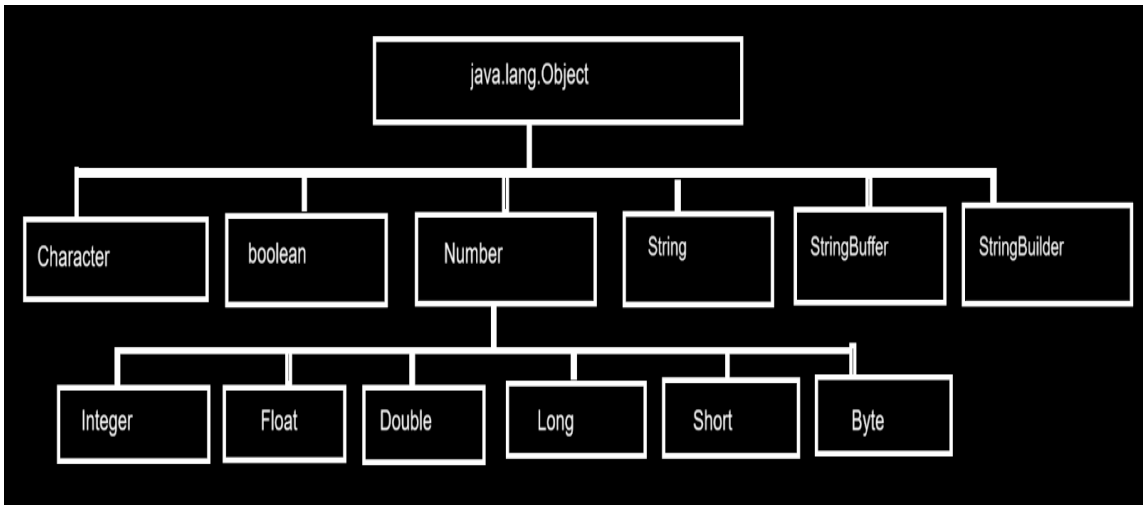


Note: if we think about above code we have reference variable String str="1234" and we try to convert this reference variable

to int and we cannot convert reference to primitive or primitive to reference.

If we want to solve this problem JAVA provides some inbuilt classes to us known as Wrapper classes

Hierarchy of Wrapper classes




```
public class ConApp
{ static Integer a;
  static int b;
  public static void main(String x[])
   {   System.out.println("A is "+a);
       System.out.println("B is "+b);
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ConApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java ConApp
A is null
B is 0

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Note: if we think about above code have statement Integer a; here a is reference variable so the default of a is null and b is primitive type of integer so its default value is 0

There are two types of conversion in Wrapper classes

---

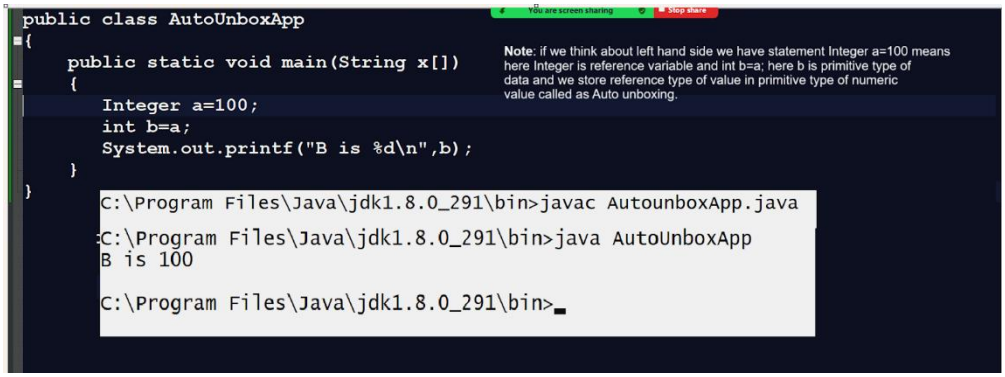
a. Autoboxing : autoboxing means converting primitive value into reference value directly called autoboxing**.**

```
public class AutoBoxApp
{
    public static void main(String x[])
    {
       int a=100;
       Integer b=a;//autoboxing
       System.out.println("B is  "+b);
    }
}
```

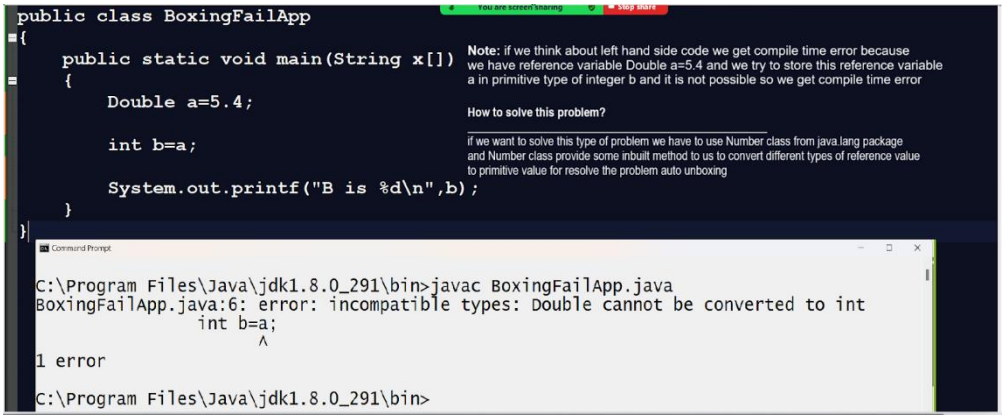
Note: if we think about left hand side we code have primitive type of data int a=100 and we have referential type of data name as Integer b and we have statement b=a; here means we store primitive value in to reference value called as autoboxing.

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac AutoBoxApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java AutoBoxApp
B is  100

C:\Program Files\Java\jdk1.8.0_291\bin>
```

b. **Autounboxing:** autounboxing means converting reference value to primitive value directly called autounboxing.



Note: if we think about Autoboxing and Autounboxing then it will fail when we try to convert different types of numeric reference to different types of primitive and different types of primitive to different types of reference shown in following code.



Now we want to think about Number class

Number: Number is abstract class from java.lang package and it contains some abstract methods which help us to perform conversion between Numeric type of reference to primitive type value

Methods of Number class

public abstract int intValue(): this method is used for convert referential Numeric value to primitive type of integer

```
public class BoxingFailApp
{
    public static void main(String x[])
    {
        Double a=5.4;
        int b=a.intValue(); //convert referencial double value to integer primitive
        System.out.printf("B is %d\n",b);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac BoxingFailApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java BoxingFailApp
B is 5

C:\Program Files\Java\jdk1.8.0_291\bin>
```

public abstract long longValue(): this method is used for convert referential numeric value to primitive type of long

public abstract float floatValue(): this method is used for convert referential numeric value to primitive type of float

public abstract double doubleValue(): this method is used for convert referential numeric value to double primitive type

public byte byteValue(): this method is used for converting numeric reference value to primitive byte type.

public short shortValue(): this method is used for converting the numeric reference value to primitive short type.

```
public class BoxingFailApp
{
    public static void main(String x[])
    {
        Double a=5.4;
        long l=a.longValue();//convert referencial double to lont value
        System.out.println("Long l "+l);
        Float f=5.6f;
        int d=f.intValue();//convert float to int
        System.out.println("Int D is "+d);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac BoxingFailApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java BoxingFailApp
Long l 5
Int D is 5

C:\Program Files\Java\jdk1.8.0_291\bin>
```

parseXXX(): this is the method present in every wrapper class and it is a static method present in every wrapper class and the goal of this method is convert string value to primitive type of value.

Syntax: int var=Integer.parseInt(String): convert string to integer

float var=Float.parseFloat(String): convert string to float

double var=Double.parseDouble(String): convert string to double

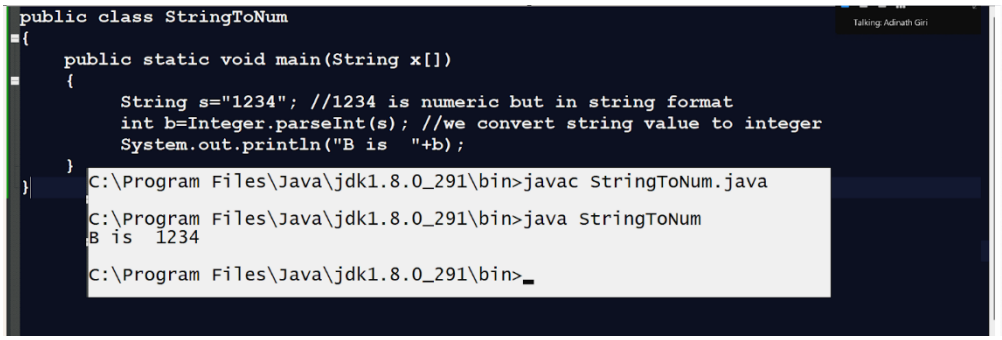
long var=Long.parseLong(String): convert string to long type

short var=Short.parseShort(String): convert string to short

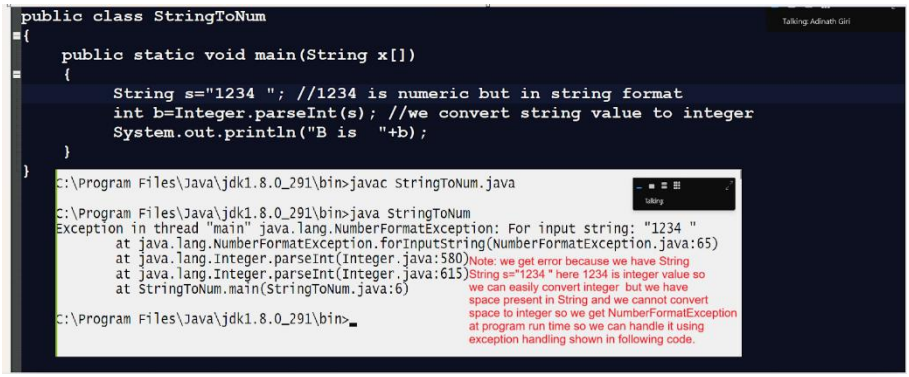
Etc

Note: this method may generate NumberFormatException to us at program run time.

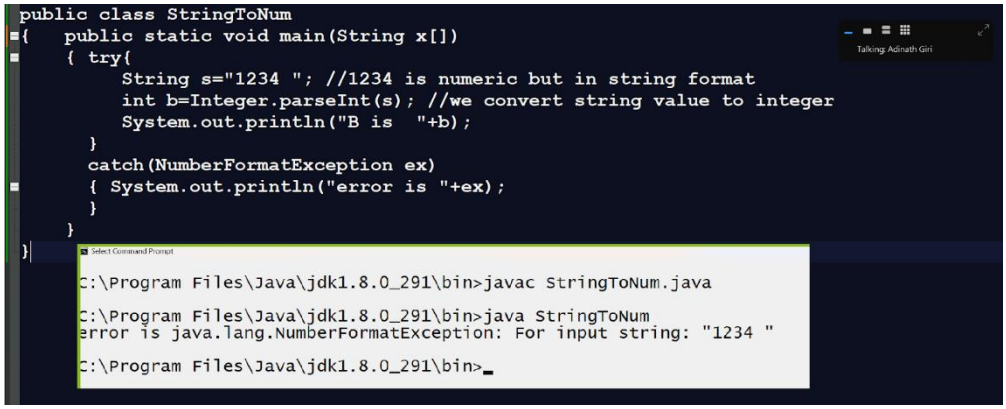
Because if we think about string then it may be possible it contains some non numeric values like space or any special character or alphabet then there is possible we get errors at program run time or NumberFormatException at program run time.



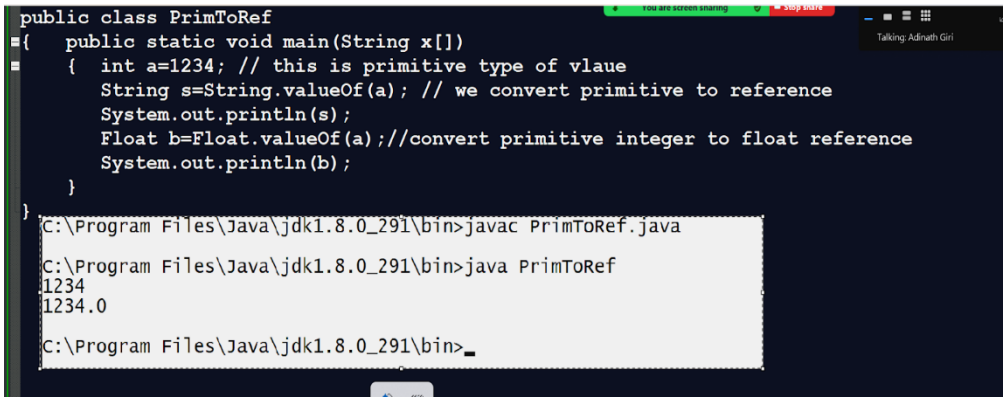
Note: possibility exception in given code



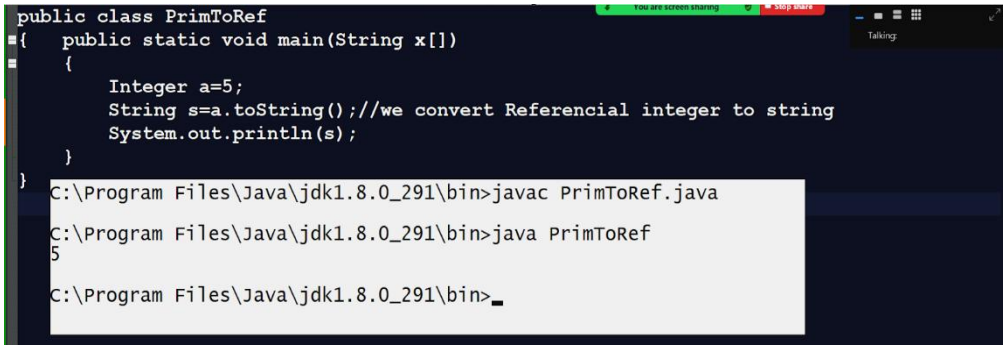
Example with source code



valueOf(): this method is used for converting any primitive value to reference variable means convert primitive type of numeric value to reference type or convert numeric value to string type of value also.



String toString(): convert any referential value to string type.



String Handling in JAVA?

Q. What is String in JAVA?

String is immutable objects in JAVA means once we initialize value then we cannot modify its later called as immutable

Note: in Java Every “” (double quote) is considered a string object.

If you want to create a string in java we have two ways.

a. Using initialization technique

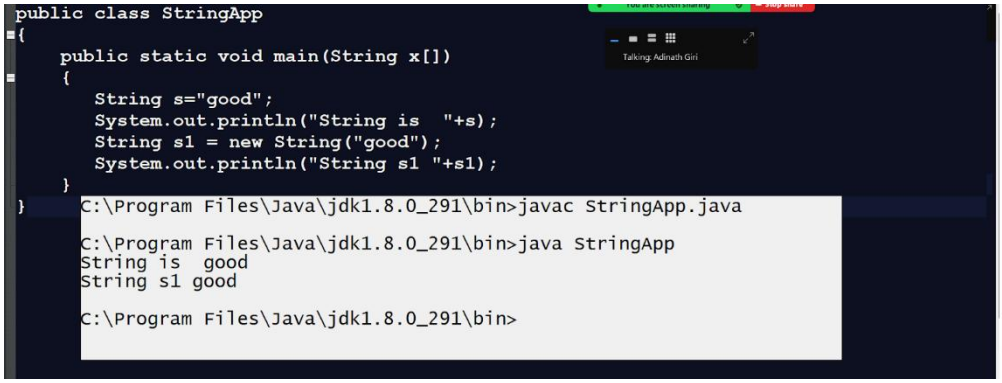
Syntax: String variable=”value”;

Example: String s=”good”;

b. Using new keyword

Syntax: String variable=new String(“value”);

Example: String s = new String(“good”);



Q. What is the difference between string creation using initialization and using a new keyword?

Using initialization approach: when we create a string using initialization technique then the string is created in string constant pool and when we have the same string value then internally JVM does not create two different string objects in memory just create a single string object and share its reference in different variables or address in different reference variable.

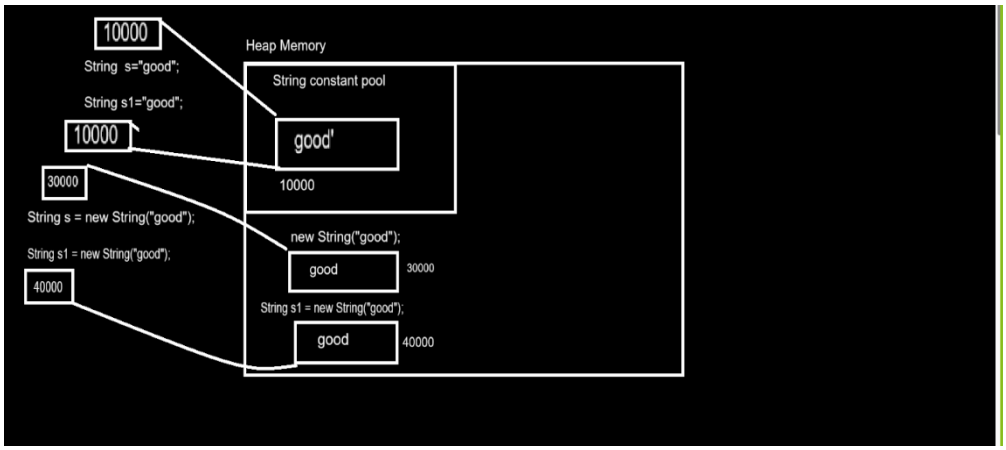
Using a new keyword approach: when we create a string using a new keyword then JVM creates a string object in the heap section of memory and when we have string with the same value or different value then JVM creates a new string object every time.

Q. What is the heap section?

Heap section is part of JVM memory and which is responsible for allocate memory at run time as well as destroy the memory at run time using garbage collection technique means Heap section can allocate memory of object or array at run time by using new keyword and it is part of JVM memory

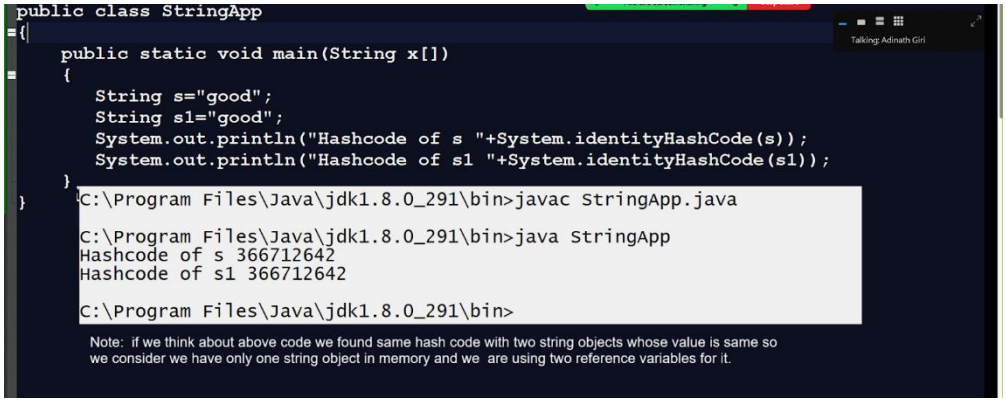
Q. What is a string constant pool?

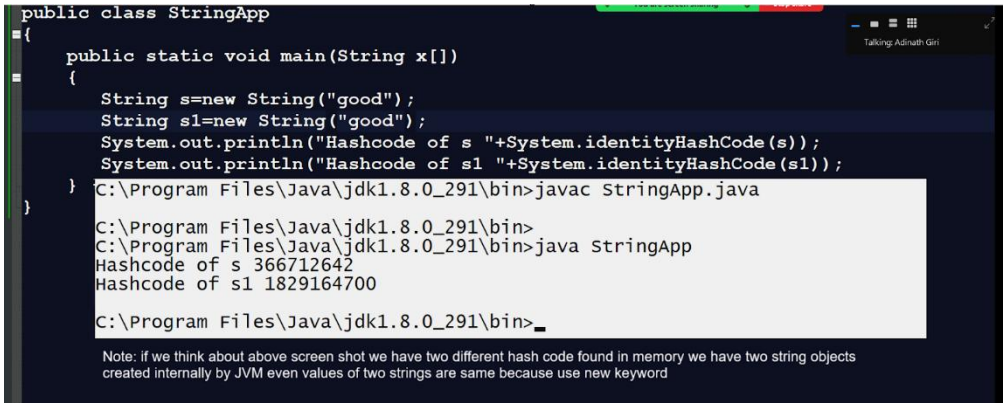
The Java String constant pool is an area in heap memory where Java stores string literals and when the same value string literals is found then not allowed to create a new string object just share its reference in different string variables.



How can we prove the above diagram?

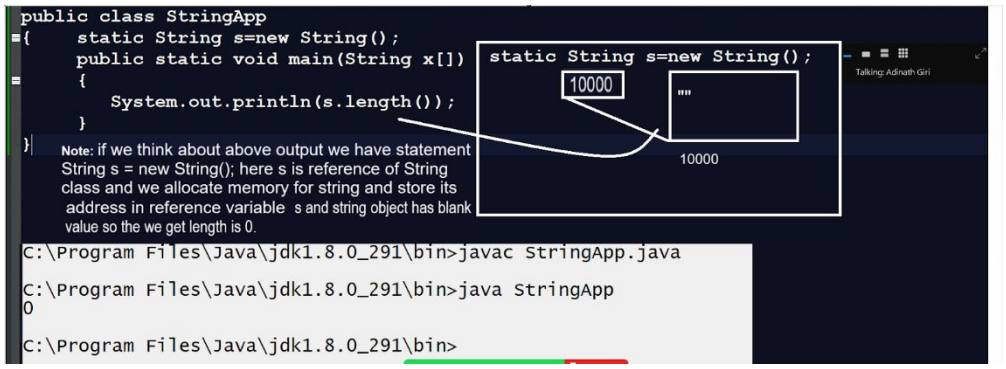
Note: you can check this by checking the hashcode of objects.



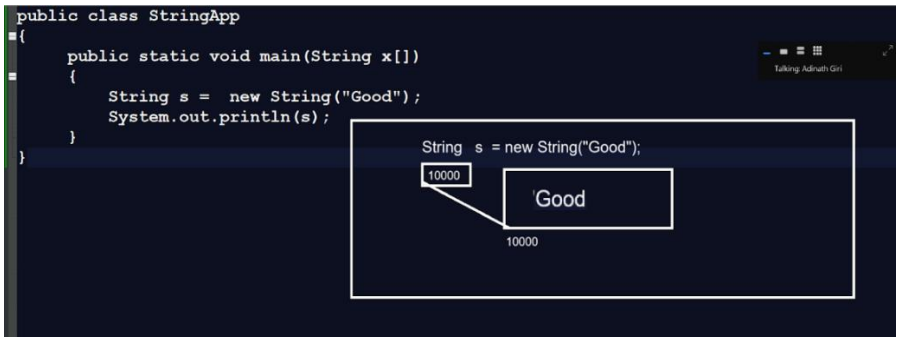


If we want to create a String object in java we have the following constructors.

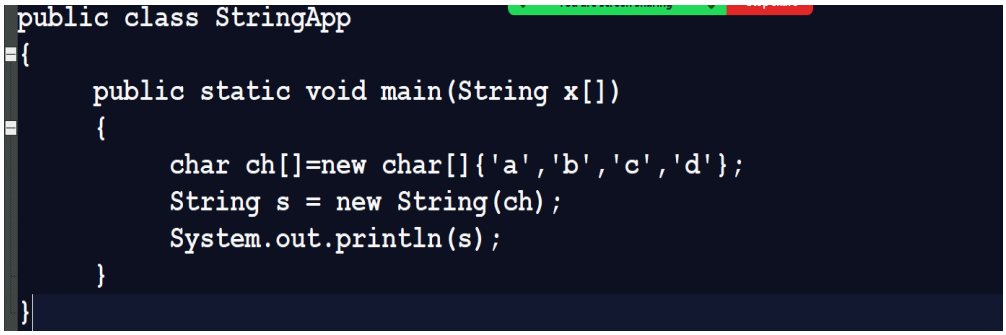
String(): this will create an empty string object in the heap section of memory.



String(String s) : this constructor can create string objects with some initialised value in memory.



String(char[]): this constructor helps us to create a string by using character array to convert a character array into string object.

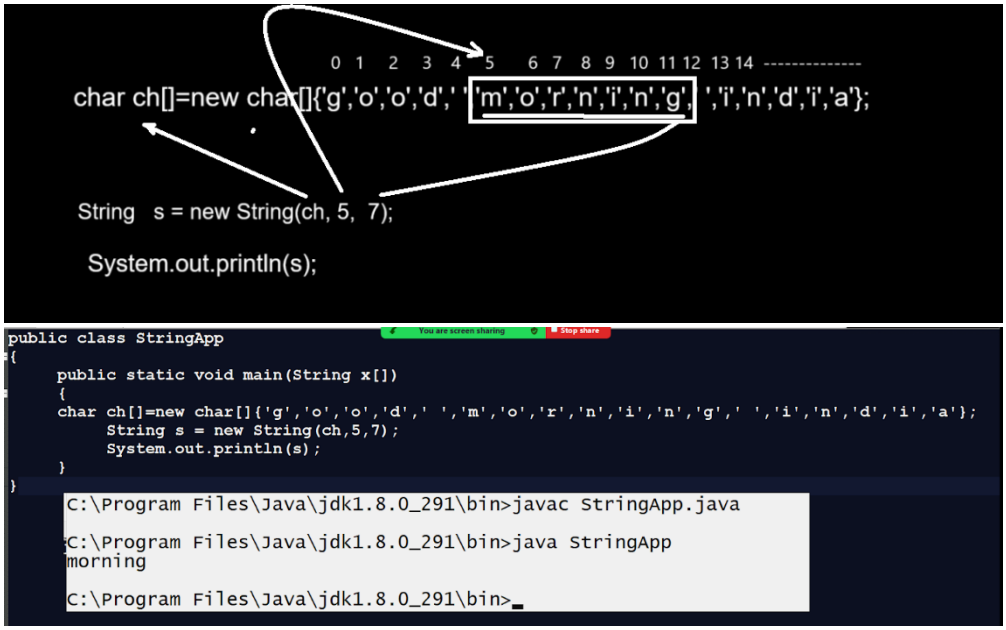


String(char[],int offset,int length): this constructor helps us to convert a specified portion of the character array into string format.

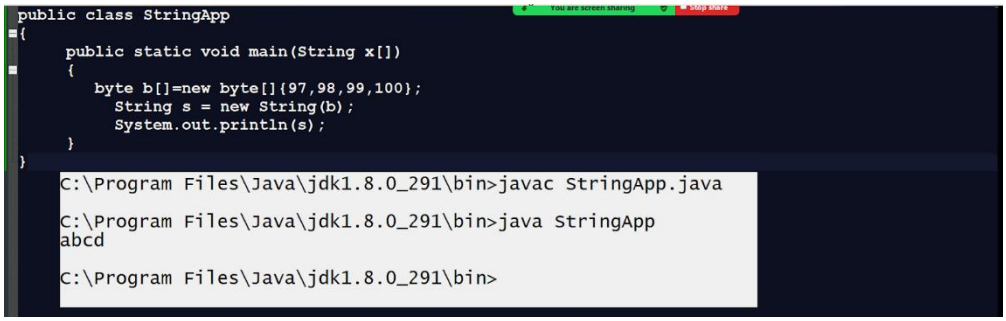
char []: this parameter accept character array as parameter

int offset: this parameter help us to set index from we want to extract data

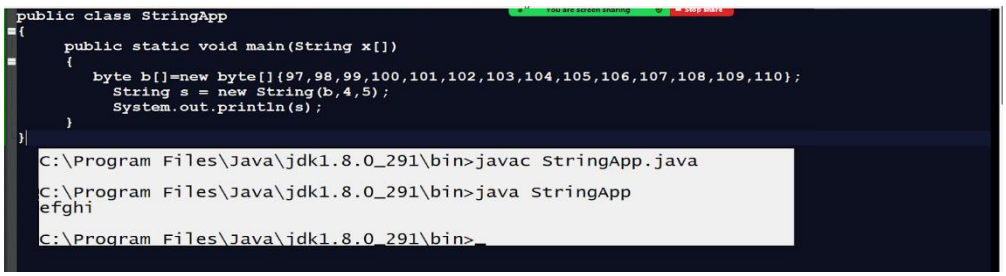
int length: this parameter helps us to decide the length of data which we want to extract.

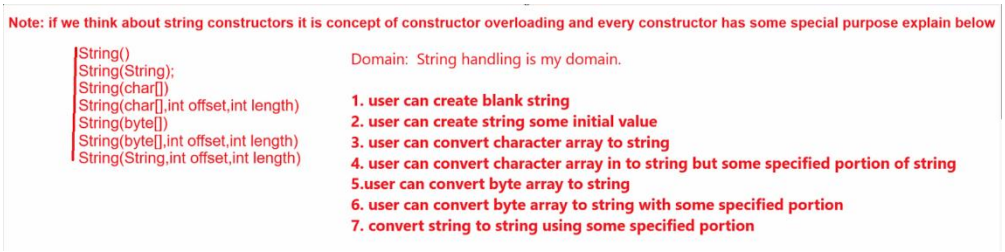


String(byte[]): this constructor helps us to convert a byte array into a string object.



String(byte[],int offset,int length): this constructor help us to extract some specified portion from byte array and convert in string format





Important points related with string

---

1. **String is thread safe :** means multiple threads cannot access String object simultaneously if one thread object uses string object then another thread cannot access within that period can access when previous thread finishes.

Because string class is final class so it is thread safe.

2. String class cannot inherit in another class because internally it is final class and final class cannot inherit in another class.

3. String class is immutable class means once we initialise some value in string object we cannot modify it later.

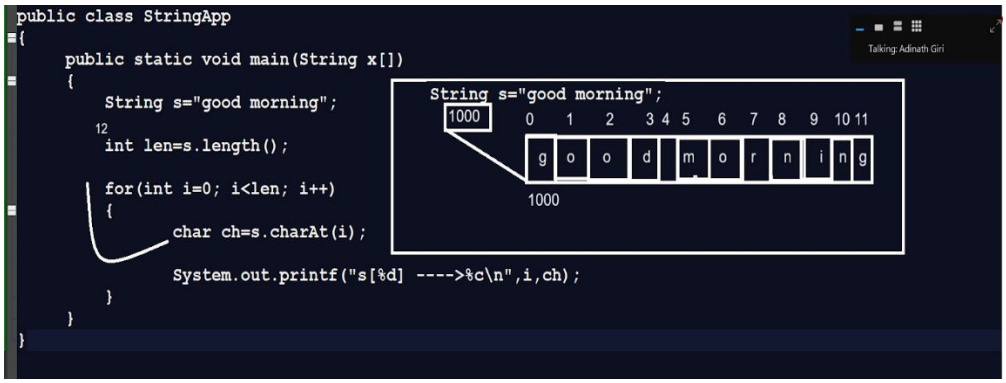
4. Not need to import package to use string class because it is member of java.lang and java.lang is default package so we not need to import

Methods of string class

---

int length(): this function helps us to calculate length of string.

char charAt(int index): this function is used for fetch characters using its index.



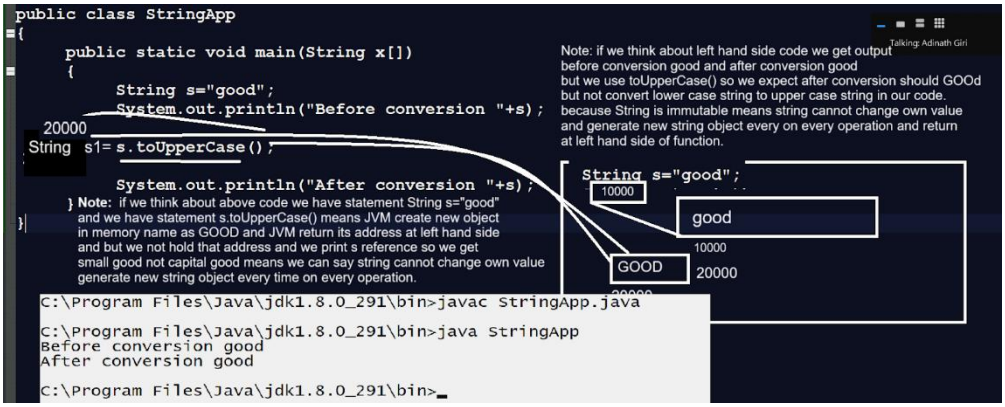
Example: WAP to calculate the length of string without using length() function

```
public class StringApp
{
     public static void main(String x[])
     {
          String s="good morning";
          s=s+"\0";

          int count=0;

          while( s.charAt(count)!='\0') //g!=0
          {
                ++count;//1
          }
          System.out.println("Length of string "+count);


     }
}
```

```
public class StringApp
{
     public static void main(String x[])
     {     int len=-1;
          String s="good";
          try{
              for(;;){
               s.charAt(++len);
              }
          }
          catch(Exception ex)
          { System.out.println("Length of string is "+len);
          }
     }
}
```

String toUpperCase(): this function can convert lower case string to upper case string.

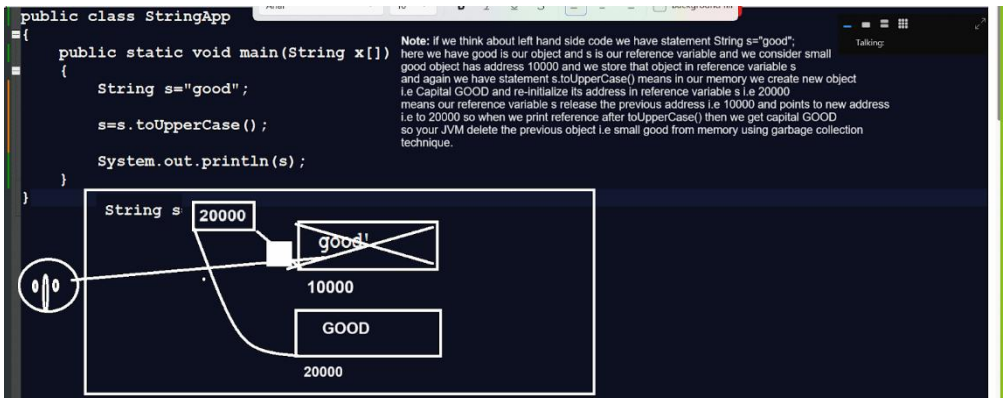
We solve above problem in next diagram



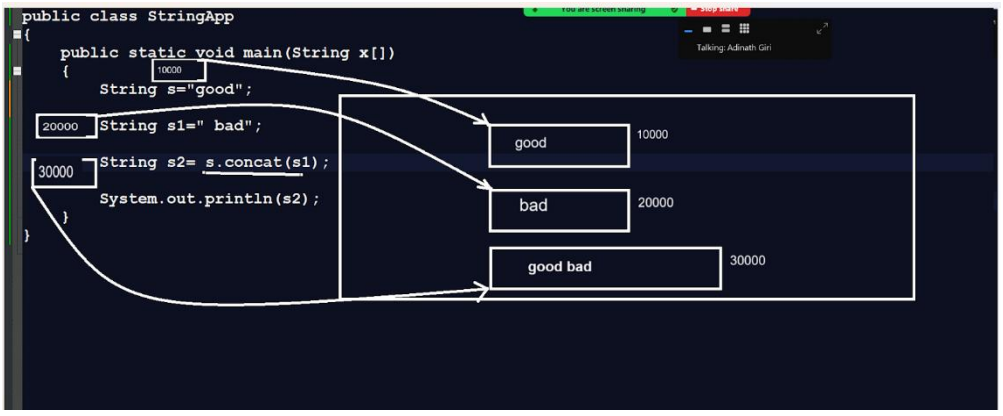
Example with source code

```
public class StringApp
{
    public static void main(String x[])
        {       String s="good";
System.out.println("Before conversion "+s+"\t"+System.identityHashCode(s));
String s1=s.toUpperCase();
 System.out.println("After conversion "+s1+"\t"+System.identityHashCode(s1));
        }
}
```

Example: Program address reinitialize in string variable

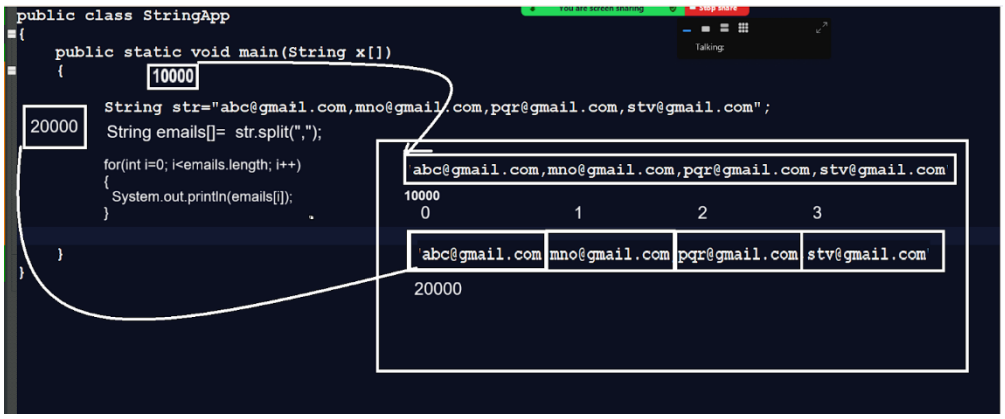


String concat(String): this method is used for combining the two strings and returning the third string as a new resultant at the left hand side means combining two string objects and generating the new third string object.

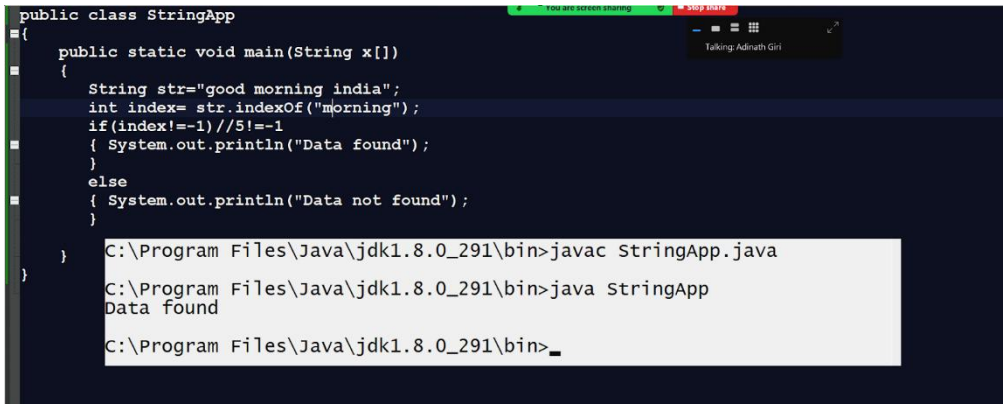


String trim(): this method is used for removing the white spaces at beginning and ending of string.

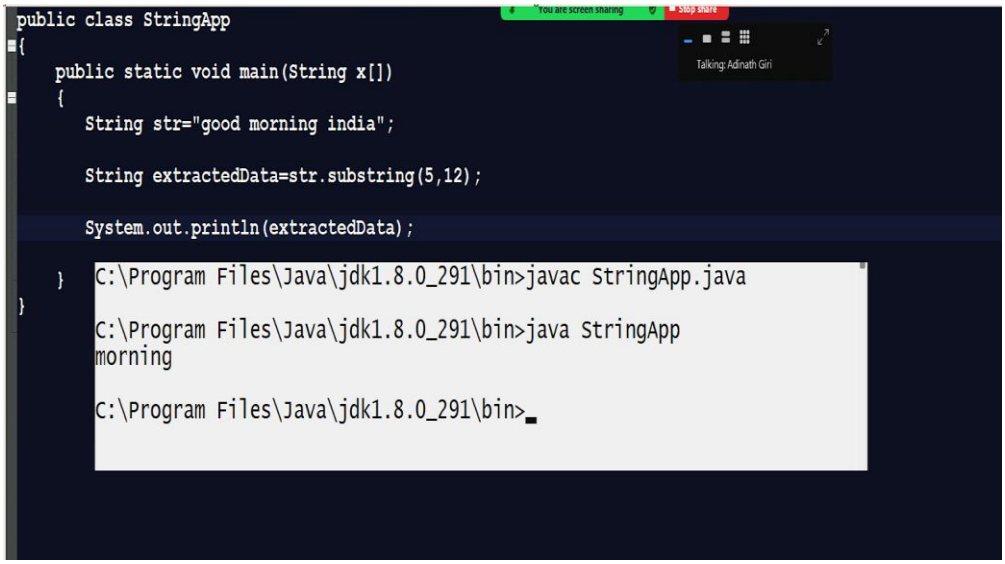
String [] split(String): this method can split the string using some specified character and return data in the form of an array.

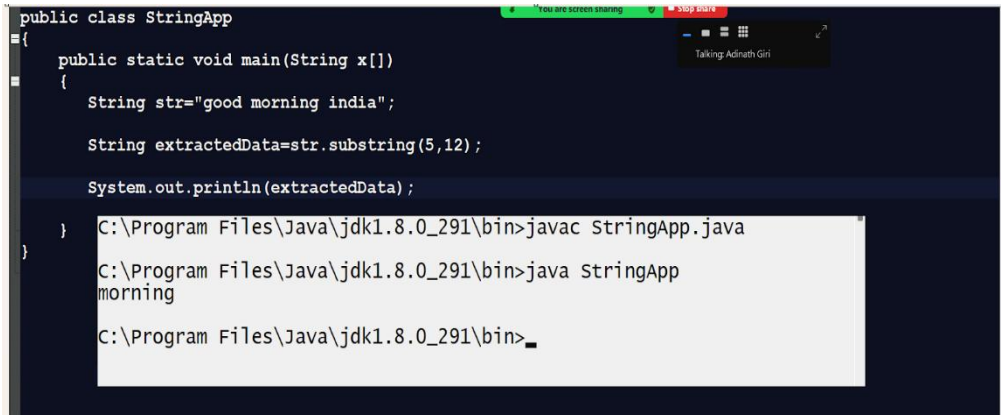


int indexOf(String): this method is used for search string or specified string data from string and return its index if string data found otherwise return -1

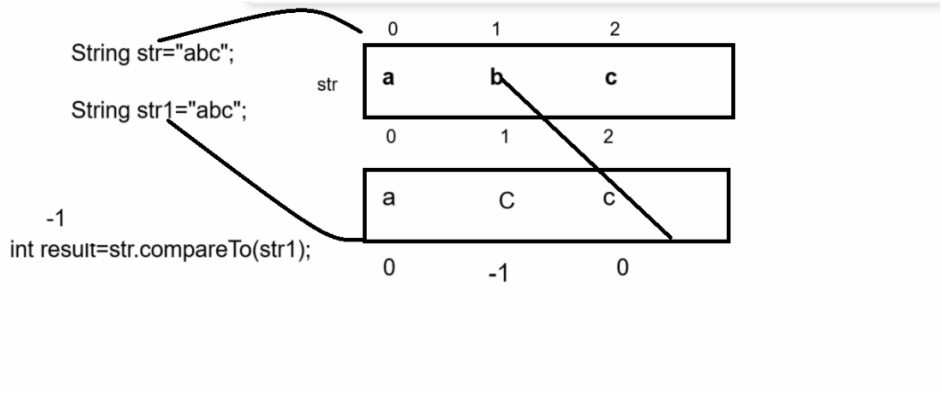


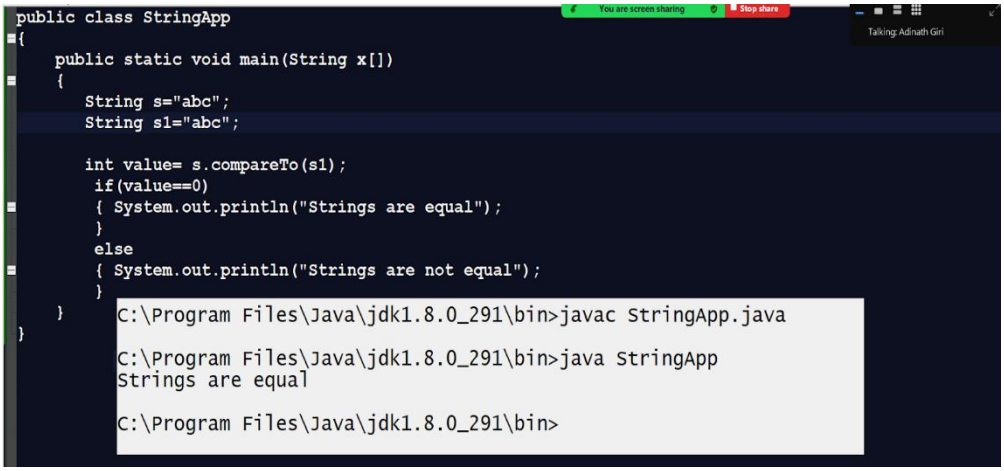
String subString(int startIndex,int endIndex): this method is used for extracting the some specified port of string between two indexes and storing its data at the left hand side.



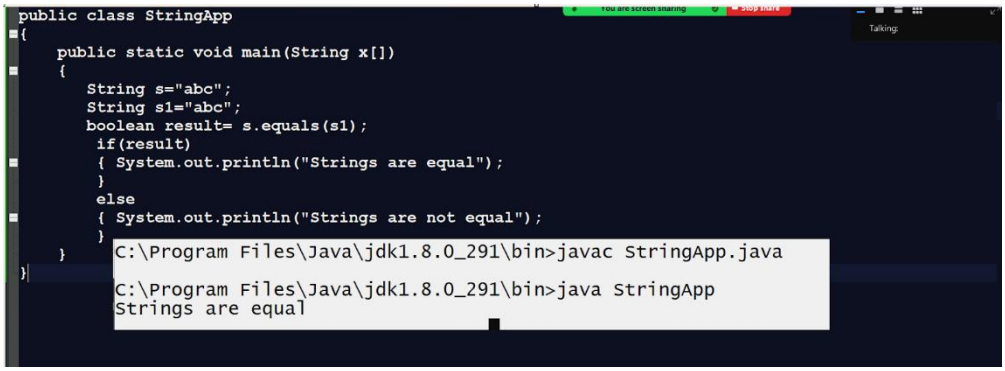


int compareTo(String): this method can compare two strings with each other by using lexical order means by using alphabetic comparison means compare alphabet of every index same alphabet found in two string return 0 if different alphabet found return first mismatch ascii code difference





boolean equals(Object) : this method can compare any kind of object using its internal content and satisfy java object comparison rules and return true if objects are equal otherwise false.



Q. What is the difference between compareTo() and equals()?

---

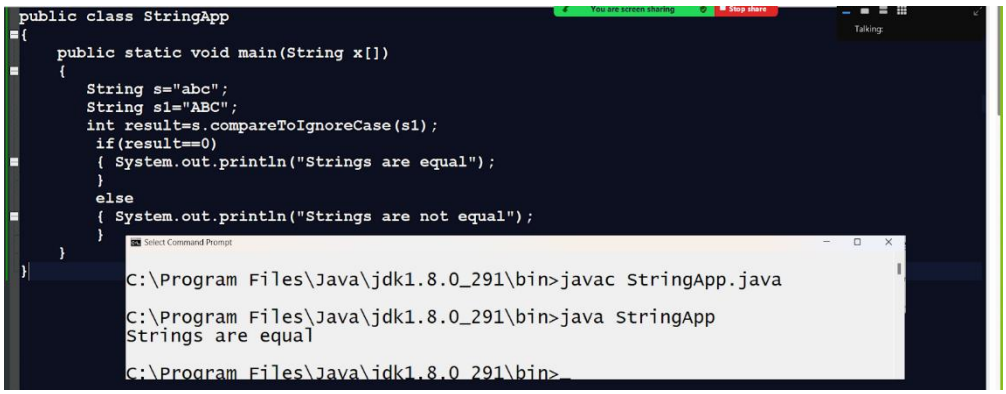
a. compareTo() method can compare only strings objects not other and equals() method can compare any kind of object in java

b. compareTo() method is originally member of string class and equals() method of java.lang.Object and Object is parent of every class in JAVA

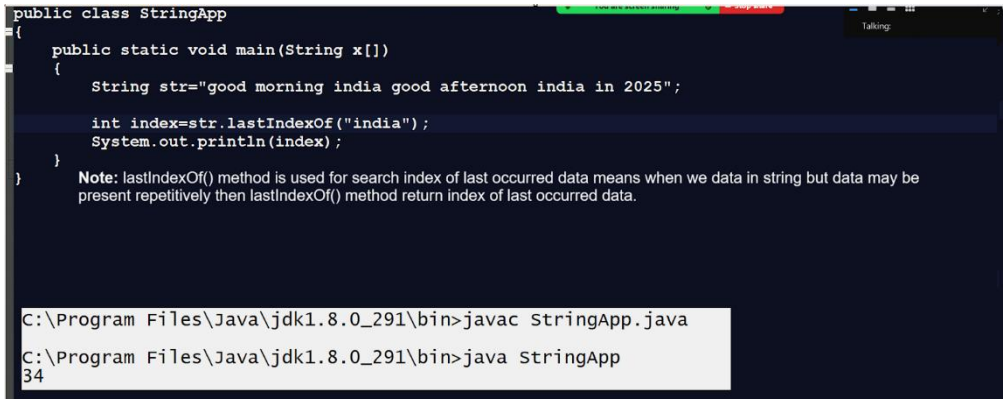
c. Return type of compareTo() method is int and return type of equals() method is boolean

d. compareTo() compares the string by using lexicol order and equals() method can compare string by using its hashcode.

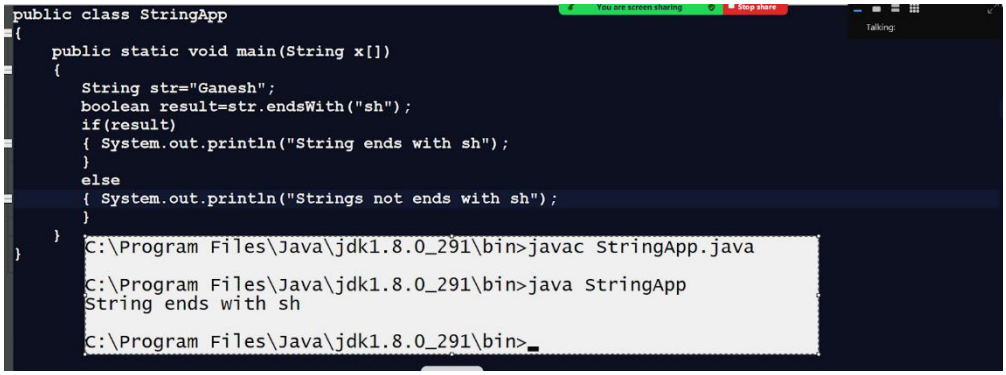
int compareToIgnoreCase(String): this method help us to compare two strings without checking its internal case



Example of lastIndexOf() method?



boolean endsWith(String): this method is used to check if a string ends with specified data or not if string ends with specified data return true otherwise return false.



boolean startsWith(String): this method checks if a string starts with specified character or not if start with specified character return true otherwise return false.

```
public class StringApp
{
    public static void main(String x[])
    {
       String str="Ganesh";
       boolean result=str.startsWith("G");
       if(result)
       { System.out.println("String ends with G");
       }
       else
       { System.out.println("Strings not ends with sh");
       }
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
String ends with G

C:\Program Files\Java\jdk1.8.0_291\bin>
```

String intern(): this method can shift string in string constant pool from heap and also return reference if same string available in string constant pool.

```
public class StringApp
{
    public static void main(String x[])
    {
        String s=new String("Good");
        String str=s.intern();
        System.out.println(str);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
Good

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Example: WAP to compare two strings in java without using

int compareTo()?

```
public class StringApp
{  public static void main(String x[])
    {    String str="abc";
         String str1="abc";
         boolean flag=true;
         if(str.length()==str1.length())
         {  for(int i=0; i<str.length();i++)
             {     if(str.charAt(i)!=str1.charAt(i))
                   {  flag=false;
                      break;
                   }
             }
             if(flag)
             { System.out.println("Strings are equal");
             }
             else{
                 System.out.println("Strings are not equal");
             }
         }
         else
         {
              System.out.println("Strings are not equal");
         }
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
Strings are  equal
```

Example: WAP to extract a digit from string and calculate its sum?

Input: abc123mno456stv

```
public class StringApp
{  public static void main(String x[])
        {   String str="abc1234mno567stv";
            int sum=0;
                for(int i=0; i<str.length();i++)
                {
                        char ch=str.charAt(i);
                        if(ch>=48 && ch<=57)
                        {  sum=sum+((int)ch-48);
                                System.out.printf("%d\t",((int)ch-48));
                        }
                }
                System.out.printf("\nSum of all values is %d\n",sum);
        }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
1       2       3       4       5       6       7
Sum of all values is 28

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Example: WAP to concat two string without using concat() function

a. **First approach using + (sign) :** In Java we can combine two strings by using + operator

```
public class ConcatApp
{
    public static void main(String x[])
    {
        String str="abc";
        String str1="mno";

        String str2 = str+str1;
        System.out.println("After concatination  "+str2);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ConcatApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java ConcatApp
After concatination  abcmno

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Note: we can convert any value or concat any value with string value.

b. We can take temporary string and initialize it using “” and travel the first string and concat every character in temp string and travel secon string and concat every character in temp string.

```
public class ConcatApp
{  public static void main(String x[])
    {   String str="abc";
        String str1="mno";
        String temp="";
         for(int i=0; i<str.length(); i++)
         { temp=temp+str.charAt(i);
         }
         for(int i=0;i<str1.length();i++)
         {   temp=temp+str1.charAt(i);
         }
         System.out.println(temp);
    }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac ConcatApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java ConcatApp
abcmno

C:\Program Files\Java\jdk1.8.0_291\bin>
```

WAP to check if strings end with specified data or not without using endsWith()?

```
import java.util.*;

public class CheckEndsWithApp

{  public static void main(String x[])

        {

           Scanner xyz  = new Scanner(System.in);

                System.out.println("enter original string");

                String str=xyz.nextLine();

                System.out.println("Check strings ends with");

                String str1=xyz.nextLine();

                boolean b= myEndsWith(str,str1);
```

```
            if(b){
                    System.out.println("Strings ends with  "+str1);
            }
            else
            {  System.out.println("Strings not ends with  "+str1);
            }
        }
        public static boolean myEndsWith(String original,String end)
        {  int index=original.lastIndexOf(end);
            if(index!=-1)
            { return true;
            }
            else{
              return false;
            }
        }
}
```

StringBuffer and StringBuilder class

StringBuffer and StringBuilder are the mutable classes of JAVA

Mutable means those values can change later or after initialization called mutable classes.

StringBuffer class

Important points related with StringBuffer

a. StringBuffer is mutable class of java

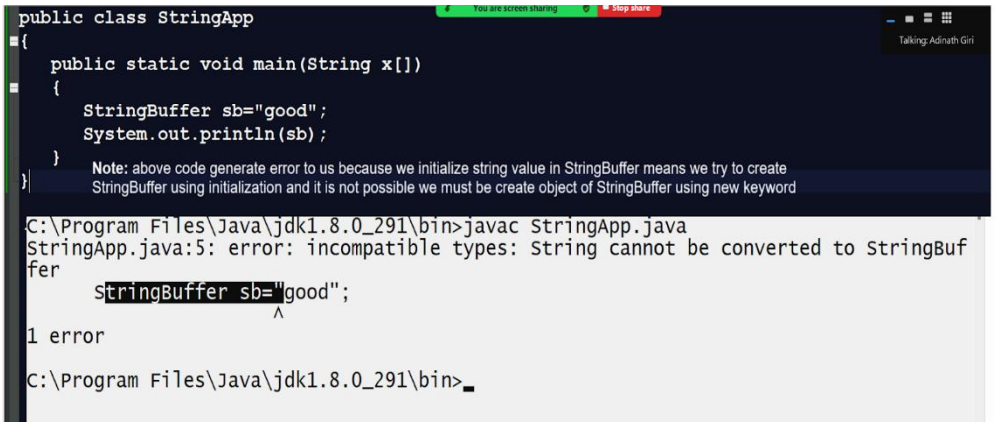
b. StringBuffer can create only its object in heap section means we can create object of StringBuffer by using only new keywords not using initialization

c. StringBuffer is a thread safe class that means an object of StringBuffer can be used by only one thread at time means StringBuffer object cannot be used by more than one thread simultaneously as well as StringBuffer methods are synchronized.

d. StringBuffer methods are synchronized so it is thread safe

e. StringBuffer have some additional methods as compare with string like as append(),insert(),delete() etc

Example of StringBuffer



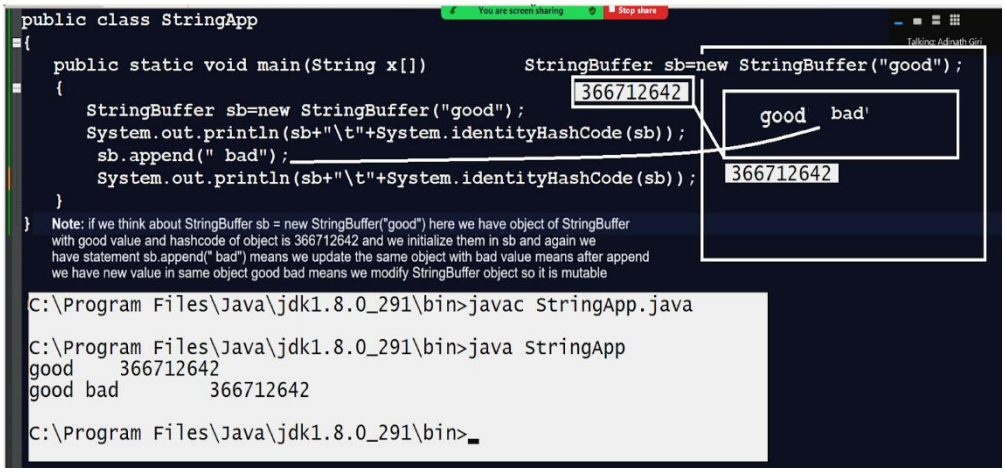
Example of StringBuffer using new keyword

Additional methods of StringBuffer and StringBuilder

---

void append(int), void append(float),void append(double),void append(long),void append(short) ,void append(Object) etc

This is the overloaded method with the StringBuffer object means using this method we can append any kind of data at the end of the StringBuffer object and not generate new object after append like as string modify existing object data or content



void insert(int index, int data), void insert(int index,float data), void insert(int index,double data) etc

This is also overloaded method with all data types in JAVA and using this method we can insert data on specified index in StringBuffer or StringBuilder object and shift previous data of that object from specified index in forward direction

void delete(int startIndex,int endIndex): this method is used for delete the data between two specified indexes and update StringBuffer or StringBuilder object

```
public class StringApp
{ public static void main(String x[])
   {
      StringBuffer sb=new StringBuffer("good morning india");
      System.out.println(sb+"\t"+System.identityHashCode(sb));
         sb.delete(5,12);
       System.out.println(sb+"\t"+System.identityHashCode(sb));
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
good morning india      366712642
good  india     366712642

C:\Program Files\Java\jdk1.8.0_291\bin>
```

void reverse(): this method is used for reverse the string value.

```
public class StringApp
{ public static void main(String x[])
   {
      StringBuffer sb=new StringBuffer("good morning india");
      System.out.println(sb+"\t"+System.identityHashCode(sb));
         sb.reverse();
       System.out.println(sb+"\t"+System.identityHashCode(sb));
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
good morning india      366712642
aidni gninrom doog      366712642

C:\Program Files\Java\jdk1.8.0_291\bin>
```

void replace(String oldstring,String newString): this method can replace the string object from old string to new string.

```
public class StringApp
{ public static void main(String x[])
   {
      StringBuffer sb=new StringBuffer("good morning india");
      System.out.println(sb+"\t"+System.identityHashCode(sb));
         sb.replace(5,12,"afternoon");
       System.out.println(sb+"\t"+System.identityHashCode(sb));
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
good morning india      366712642
good afternoon india    366712642

C:\Program Files\Java\jdk1.8.0_291\bin>
```

```
public class StringApp
{ public static void main(String x[])
   {
      StringBuilder sb=new StringBuilder("good morning india");
      System.out.println(sb+"\t"+System.identityHashCode(sb));
         sb.replace(5,12,"afternoon");
       System.out.println(sb+"\t"+System.identityHashCode(sb));
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac StringApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java StringApp
good morning india      366712642
good afternoon india    366712642

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Q. What is the difference between StringBuffer and StringBuilder?

---

a. StringBuffer object is thread safe and StringBuilder object is not thread safe

b. StringBuffer methods are synchronized and StringBuider methods are not synchronized.

c. StringBuffer is slower than StringBuilder

d. StringBuffer recommend when we want to thread safety but StringBuilder recommend in Asynchronous environment

e. StringBuffer present in JAVA from JDK 1.0 version and StringBuilder introduced in JDK 1.5 version of JAVA

Q. What is the difference between String and StringBuilder?

---

a. String is immutable class of Java and StringBuilder is mutable class of JAVA

b. String can create using initialization or using new keyword means can create using string constant pool or using heap but StringBuilder object must be create using new keyword means always create heap section of memory

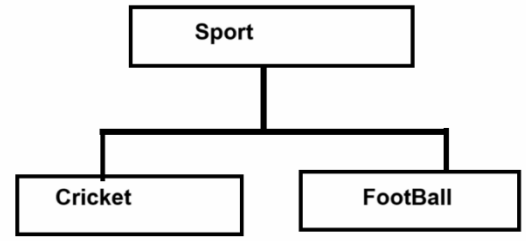
c. String is thread safe object and StringBuilder is not thread safe

d. StringBuilder has some additional method as compare with String like as append(),insert(),delete() etc

2. Referential type casting

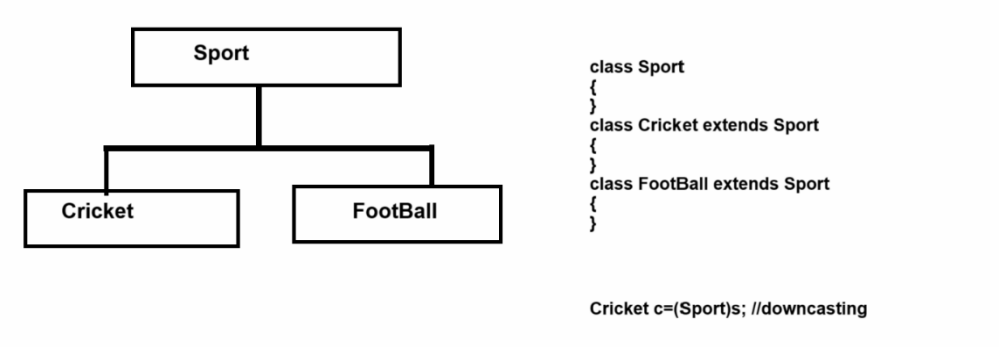
There are two types of referential type casting

a. Upcasting : when we convert a child object into parent object called as upcasting and normally in upcasting we create reference of parent and object of child class.

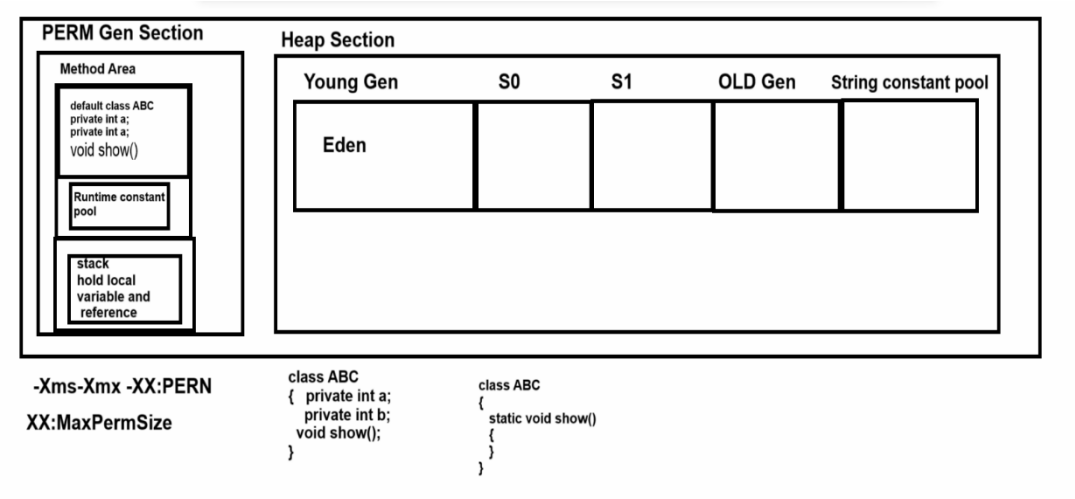



```
class Sport
{
}
class Cricket extends Sport
{
}
class FootBall extends Sport
{
}

Sport s = new Cricket(); //upcasting
```

b. **Downcasting**: Downcasting means when we convert parent object in to child object called as downcasting



JVM Memory Model - Memory Management In JAVA



Young Generation:

The young generation is the place where all the new objects are created. When young generation is filled ,garbage collection is performed is called as minor GC and two survivor memory spaces

Means JVM check if new created objects not longer use in application then JVM shift that objects in Survivor space and if required in future then again call in new heap and use it

Means shifting object from new heap section in to survivor space called as minor garbage collection and if object never required in application then JVM shift object from survivor space to Old heap and delete that object permanently from JVM memory called as major GC.

Old Generation Heap

Old Generation memory contains the object that is long-lived and survived after many rounds of Minor GC.Usually garbage collection is performed in old generation memory when it is full. Old generation garbage collection is called Major GC and usually takes a long time.

What is survivor space in JVM Heap Memory

Survivor space is initially empty and it is used for perform minor GC means if young generation is full then JVM shift unused object from heap to survivor space and survivor space hold unused objects and retrain object in new memory if they required in future if not required in future then shift in old memory and old memory can perform major garbage collection on it.

Permanent Generation

Permanent generation or perm Gen section is the application metadata required by the JVM to describe the classes and methods used in the application.

Here MetaData means classname, method name ,its access specifier, return type ,parameter list etc

Example: suppose JVM create object at run time but for object creation JVM need to know the class name, and for method calling JVM need know method name etc

Also permanent generation store static variables, static methods etc

Note: Permanent generation section is not part of heap memory that is created or populated by JVM at run time based on the classes used by the application.

Permanent generation also contains JAVA SE library classes and methods.

Method Area: Method area is part of space in the perm gen and used to store class structure (runtime constant or static variables and code for methods and constructor)

Memory Pool: Memory pool are created by JVM memory manage to create POOL immutable of objects

Example: String constant pool is best example of memory pool

Note: Memory pool can belong from heap section memory or may be from Permanent generation section it is dependent on JVM implementation.

Runtime constant pool: Runtime constant pool is pre-class runtime representation of the constant pool of class; it contains class run time constant i.e static constant or static method and runtime constant pool is part of method area.

Java Stack Memory

Java Stack memory is used for execution of thread. They contain method specific values that are short lived and reference to other objects that are getting referenced from the method means if we create any object in method or specified function then reference of that object stored in stack and object created in heap memory and reference hold address of object from heap memory.

Local variable and reference present in stack present whenever function in running when function close its block the JVM delete the all members from that function or stack

Normally stack memory under method area when we call function or start thread

Q.What is the difference between stack and heap memory?

Q. What is heap memory?

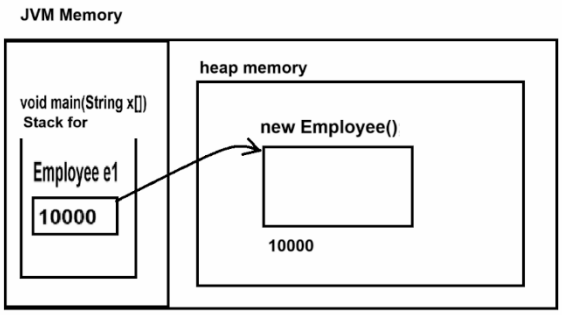
Heap is part of JVM memory which is used for creating objects at run time and performing minor GC and Major GC on object according requirement and access of object globally in application means provide access of object in multiple stack using reference as well as provide access of object run time memory pool and permanent generation section also.

Q. What is stack?

Stack is memory used for thread execution as well as created on function calling and maintain data in last in first out format and it contain local variable may be primitive or reference and short-lived variables means variable within stack removed from memory automatically when function block is closed

Following Example shows the difference between stack and heap?

```
class Employee
{  private int id;
   private String name;
   private int sal;
 //setter and getters here
}
public class  Test
{
   public static void main(String x[])
   {
        Employee e1 = new Employee();
         e1.setId(1);
         e1.setName("ABC");
         e1.setSal(10000);
   }
}
```



Q. What is garbage collection?

Garbage collection is automatic memory management technique of JVM which is used for release the object memory when object not use in future

There are two types of garbage collection

1. **Minor GC:** Minor GC means when new created object allocate memory eden or new generation of heap but if not longer use in future then JVM shift object in survivor space i.e on bench if object required in future then again shift in new generation heap or if not required then shift in old memory
2. **Major GC :** Major GC normally perform by Old heap memory means in old heap memory we have long lived objects but if object not in future means object address or reference not use in code anywhere in future then JVM delete the object permanently from heap section called as garbage collection

Note: developer can perform garbage collection manually for that develop required to override finalize() method of object class and call System.gc() method manually

MultiThreading

Note: Before starting Multithreading we should have to know

What is thread?

Thread is a sub part of the process or it is a light weight process.

Q. What is the process?

Process is a running program in ram called as process or program in execution called as process.

Q. What is multi threading?

Multi threading means executing more than one thread simultaneously or waiting for each other called multithreading or to utilize an idle period of time called multi threading.

Note: Threading is a concept of an operating system.

Java is a language which provides implementation to thread concepts.

How to implement thread using JAVA

If we want to implement thread using java we have two ways

1. **Using Thread class**
2. **Using Runnable interface**

How to implement thread using Thread class

Steps

---

1. **Add java.lang package in application**

Note: java.lang is a default package java means we do not need to import it.

2. **Create a user defined class and inherit the Thread class in it.**

Thread class is a member of java.lang package which contain methods or provide Thread implementation feature to us

```
class ABC extends Thread
{
}
```

Note: we inherit all Thread properties in class ABC means we can say ABC is also thread.

3. **Override run() method for writing thread logics**

---

run() method is used for writing the thread logics means when developer writing logic which work as thread or execute when thread start as well as JVM execute it simultaneously with other threads then we required logic in run() method

```
class ABC extends Thread
{  public void run()
   {
     write here your thread logics
   }
}
```

Note: run() is not a method of Thread class internally it is method of Runnable interface and Thread is a implementer class of Runnable interface

Runnable is functional interface as well as it is marker interface in JAVA

```
interface Runnable
{  public void run();
}
class Thread implements Runnable
{
     public void run(){
     }
}
class ABC extends Thread
{
   public void run()
   {
   }
}
```

Q. What is the functional interface in JAVA?

_____

Functional interface means interface containing only one abstract method called Functional interface.

Q. What is the marker interface in JAVA?

_____

Marker interfaces means interface may not contain any method or may contain method but java provides a special runtime environment to them for execution called as marker interface.

4. Create object of Thread child class and use Thread class method for manage the thread

void start(): this method is used for start the thread means when we call start() method then internally run() method get executed

Note: in threading we required to call run() method manually it is internally executed when we call start() method

public static void sleep(int milliseconds)throws InterruptedException : this method is used to hold the thread execution for a specified time period and again re-execute after some specified time period.

public void stop(): this method is used for terminate the thread or destroy the thread

public void join()throws InterruptedException : this method is used to hold the thread execution whenever the running thread is not completed.

public boolean isAlive(): this method is used to check if the thread is running or not if running return true otherwise return false.

public void setPriority(int priority): this method help us to set the priority of thread

public int getPriority(): this method returns the current priority of thread and every thread has default normal priority.

void setDaemon(boolean): this method can set thread as daemon thread when we pass true value in it

boolean isDaemon(): this method checks the working thread as daemon or not if working is daemon thread then returns true otherwise returns false.

Object class method for manage the thread

void wait(int milliseconds): this method can set conditional wait with thread

void wait(): this method can set unconditional wait with thread.

void notify(): this method call waits thread one by one single at time in Queue format means first in first out format.

void notifyAll(): this method calls all waited threads at time in the first out format.

Example with two threads and execute in waiting of each other

```
class F extends Thread
{
   public void run()
         {
           try{
              for(int i=1; i<=5; i++)
                    { System.out.println("First Thread is  "+i);
                      Thread.sleep(10000);
                    }
           }
           catch(InterruptedException ex)
           { System.out.println("Error is  "+ex);
           }
         }
```

```
}
class S extends Thread
{   public void run()
        {  try{
             for(int i=1; i<=50; i++)
                  { System.out.println("Second Thread is  "+i);
                    Thread.sleep(1000);
                  }
           }
           catch(InterruptedException ex)
           { System.out.println("Error is  "+ex);
           }
        }
}
public class ThreadProApp
{
   public static void main(String x[])
        {  F f1 = new F();
           f1.start();
          S s1 = new S();
           s1.start();
        }
}
```

Example: Suppose consider we have Two threads and first thread has 10 second waiting period and second thread has 1 second waiting period so by default JVM execute first thread and second in waiting of first and first in waiting of second but we want to execute second thread when first thread is finish even first thread waiting period.

Note: for implement above approach we can use join() method with first thread means call join() method with first thread after start() method of First thread

Example with source code

```
class F extends Thread
{
   public void run()
          {
            try{
               for(int i=1; i<=5; i++)
                     { System.out.println("First Thread is  "+i);
                       Thread.sleep(10000);
                     }
            }
            catch(InterruptedException ex)
            { System.out.println("Error is  "+ex);
            }
          }
}
class S extends Thread
{   public void run()
          {  try{
               for(int i=1; i<=50; i++)
                     { System.out.println("Second Thread is  "+i);
                       Thread.sleep(1000);
                     }
            }
            catch(InterruptedException ex)
            { System.out.println("Error is  "+ex);
            }
          }
}
```

```
public class ThreadProApp
{
   public static void main(String x[])throws InterruptedException
         {  F f1 = new F();
            f1.start();
                  f1.join();
           S s1 = new S();
            s1.start();
         }
}
```

Example: we want to execute the first thread 3 times and after terminate it.

Note we can use the stop() method to terminate or destroy the thread.

```
class F extends Thread
{
   public void run()
         {
           try{
              for(int i=1; i<=5; i++)
                    { System.out.println("First Thread is  "+i);
                          if(i==3)
                          {stop();
                          }
                      Thread.sleep(10000);
                    }
           }
           catch(InterruptedException ex)
           { System.out.println("Error is  "+ex);
           }
```

```
        }
}
class S extends Thread
{   public void run()
        {  try{
             for(int i=1; i<=50; i++)
                  { System.out.println("Second Thread is  "+i);
                    Thread.sleep(1000);
                  }
           }
           catch(InterruptedException ex)
           { System.out.println("Error is  "+ex);
           }
        }
}
public class ThreadProApp
{
   public static void main(String x[])throws InterruptedException
        {  F f1 = new F();
           f1.start();
                f1.join();
           S s1 = new S();
           s1.start();
        }
}
```

Synchronization and Asynchronization in Threading

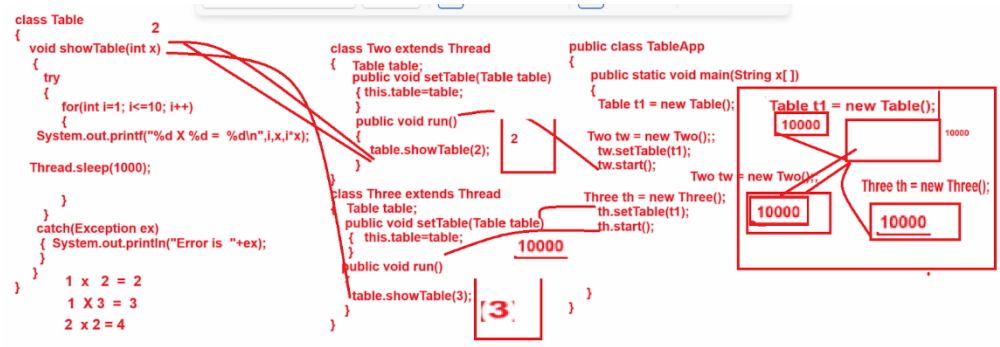
Synchronization means if two or more than two threads use the single resource sequentially one by one called as synchronization

Asynchronization

Asynchronization means if two or more than two threads use the single resource simultaneously called as asynchronization

Q. What is a Resource?

Resource means an object created in memory called a resource.



Example of asynchronization

```
class Table
{
  public void showTable(int x)
  {
    try{
            for(int i=1; i<=10; i++)
                { System.out.printf("%d X %d  = %d\n",i,x,i*x);
```

```
                    Thread.sleep(1000);
                }
            }
            catch(Exception ex)
            { System.out.println("Error is  "+ex);
            }
    }
}
class Two extends Thread
{
    Table table;
    void setTable(Table table)
    {  this.table=table;
    }
    public void run()
    {  table.showTable(2);
    }
}
class Three extends Thread
{
    Table table;
    void setTable(Table table)
    {  this.table=table;
    }
    public void run()
    {  table.showTable(3);
    }
}
public class SyncAsyncApp
```

```
{
   public static void main(String x[])
         {
            Table t1 = new Table();
                  Two tw =new Two();
                  tw.setTable(t1);
                  tw.start();
                  Three th = new Three();
                   th.setTable(t1);
                   th.start();
         }
}
```

Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>java SyncAsyncApp
1 X 2  = 2
1 X 3  = 3
2 X 3  = 6
2 X 2  = 4
3 X 3  = 9
3 X 2  = 6
4 X 3  = 12
4 X 2  = 8
5 X 2  = 10
5 X 3  = 15
6 X 3  = 18
6 X 2  = 12
7 X 2  = 14
7 X 3  = 21
8 X 2  = 16
8 X 3  = 24
9 X 3  = 27
9 X 2  = 18
10 X 3  = 30
10 X 2  = 20
```

If we want to mark your object or resource synchronized we have synchronized keyword

Example of synchronization

```
class Table
{
  public synchronized  void showTable(int x)
  {
    try{
          for(int i=1; i<=10; i++)
```

```
                { System.out.printf("%d X %d  = %d\n",i,x,i*x);
                        Thread.sleep(1000);
                }
            }
            catch(Exception ex)
            { System.out.println("Error is  "+ex);
            }
    }
}
class Two extends Thread
{
    Table table;
    void setTable(Table table)
    { this.table=table;
    }
    public void run()
    { table.showTable(2);
    }
}
class Three extends Thread
{
    Table table;
    void setTable(Table table)
    { this.table=table;
    }
    public void run()
    { table.showTable(3);
    }
}
```

```
public class SyncAsyncApp
{
   public static void main(String x[])
        {
           Table t1 = new Table();
                Two tw =new Two();
                tw.setTable(t1);
                tw.start();
                Three th = new Three();
                 th.setTable(t1);
                 th.start();
        }
}
```

wait(),notify() and notifyAll()

---

wait(): wait() method is used for hold thread execution for specified time period and it is method of object class internally

There are two types of wait method

Conditional wait : conditional wait means if a thread holds its execution for a specified time period and re-execute itself after some time period called conditional wait.

Syntax: void wait(int milliseconds): this is your conditional wait

Unconditional wait: unconditional wait means if thread hold its execution and not re-execute self after specified time period called as unconditional wait and if thread is in unconditional wait we required to provide notification to thread for re-execution purpose and for that we have two methods

Syntax: void wait(): unconditional wait means not specify time period for re-execution purpose.

Note: if we want to recall l unconditional waiting threads for re-execution we have notify() and notifyAll() methods.

void notify() : notify() method help us to call only one thread at time for re-execution purpose and call waited thread in FIFO order means using first in first out format

void notifyAll(): notifyAll() method helps to call all waiting threads at time.

Example with source code

```
import java.util.*;

class Table

{

  public synchronized  void showTable(int x)

  {

    try{

            for(int i=1; i<=10; i++)

                  {  System.out.printf("%d X %d  = %d\n",i,x,i*x);

                if(i==5)

                                    { wait(); //waiting for 1 minute

                                    }

                              Thread.sleep(1000);

                  }

          }

          catch(Exception ex)

          { System.out.println("Error is  "+ex);

          }

  }

  public synchronized void recallThread(){

            try

                  {
```

```
                notifyAll();
            }
            catch(Exception ex)
            {
                System.out.println("Error is  "+ex);
            }
    }
}
class Two extends Thread
{
    Table table;
    void setTable(Table table)
    {  this.table=table;
    }
    public void run()
    {  table.showTable(2);
    }
}
class Three extends Thread
{
    Table table;
    void setTable(Table table)
    {  this.table=table;
    }
    public void run()
    {  table.showTable(3);
    }
}
public class SyncAsyncApp
```

```
{
   public static void main(String x[])
         {
            Table t1 = new Table();
                  Two tw =new Two();
                  tw.setTable(t1);
                  tw.start();
                  Three th = new Three();
                   th.setTable(t1);
                   th.start();

                   do{
                           Scanner xyz  = new Scanner(System.in);
                           String msg=xyz.nextLine();
                           if(msg.equals("restart"))
                           {
                                    t1.recallThread();
                           }
                           if(msg.equals("stop"))
                           { System.exit(0);
                           }
                   }while(true); //infinite loop
         }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>java SyncAsyncApp
1 X 2  = 2
2 X 2  = 4
3 X 2  = 6
4 X 2  = 8
5 X 2  = 10
1 X 3  = 3
2 X 3  = 6
3 X 3  = 9
4 X 3  = 12
5 X 3  = 15
restart
6 X 3  = 18
7 X 3  = 21
8 X 3  = 24
9 X 3  = 27
10 X 3  = 30
6 X 2  = 12
7 X 2  = 14
8 X 2  = 16
9 X 2  = 18
10 X 2  = 20
```

Q. What is the difference between wait() and sleep() method?

1. The wait() method can work in synchronized block only and sleep() can work in synchronized as well as asynchronous block.
2. wait() is method of Object class and sleep() is static method of Thread class
3. wait() may be conditional or unconditional but sleep always works with conditional wait

Q. What is the difference between notify() and notifyAll()?

---

notify() method can call waited thread but call only one thread at time in FIFO order and notifyAll() can call waited thread but call all threads at time in FIFO order.

Q. Can we use notify() and notifyAll() in an asynchronous block?

No if we want to call notify() and notifyAll() must be call in synchronised block otherwise we get exception IllegalMonitorException

Q. What are the similarities between notify() and notifyAll() methods?

_____

1. Both are member of Object class
2. Both can work with only synchronised block
3. Both methods are used to call unconditional waiting thread.

How to create Thread using a Runnable interface

_____

Q. Why do we need to create a thread using Runnable interface?

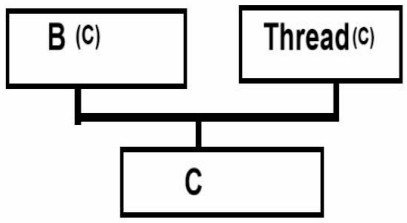
_____

Some times we have situations of Multiple inheritance in thread implementation then we cannot use Thread class for thread implementation because multiple inheritance situations cannot be handled by more than one classes just it is handled by interface.

Example: Suppose we have class C and C has parent class name as B and we want to create C class as thread class then we cannot inherit Thread class in C class.

If we want to create a Thread using a Runnable interface we have the following steps.

```
class B
{
}
class C extends B,Thread
{

}
```



Note: if we think about code implementation of thread we have two parent classes name as B and Thread and we try to inherit these parent classes in child class C But according to rule of JAVA we cannot inherit multiple parent classes in single child class so this situation say implement thread implementation by using multiple inheritance technique so we have option name as interface for that purpose java provide one interface to us known as Runnable interface for achieve multiple inheritance.

How to implement thread using Runnable interface

Steps to implement thread using Runnable interface

---

a. Add java.lang package in application

---

import java.lang.*;

b. Create class and implements Runnable interface in it

b. Override its run() method and write its logic

```
class ABC implements Runnable
{
  public void run()
  {
   try{
            for(int i=1; i<=5; i++)
                  {
                   System.out.printf("I = %d\n",i);
                   Thread.sleep(10000);
                  }
          }
          catch(Exception ex)
          { System.out.println("Error is  "+ex);
          }
  }
}
```

d. **Create object of Runnable implement class**

ABC a1 = new ABC();

e. **Create object of Thread class and pass Runnable implementer reference in it**

If we want to use thread using Runnable interface then we have to use Thread using the following constructor.

Thread(Runnable): if we think about this constructor it contain Runnable interface as parameter means here we can pass reference of Runnable using upcasting as well as we can pass reference any child of Runnable

Example with source code

```
class ABC implements Runnable
{
  public void run()
  {
    try{
            for(int i=1; i<=5; i++)
                  {
                    System.out.printf("I = %d\n",i);
                    Thread.sleep(10000);
                  }
          }
          catch(Exception ex)
          { System.out.println("Error is  "+ex);
```

```
        }
  }
}
public class TestRunApp
{
   public static void main(String x[])
        {
           ABC a = new ABC();
                Thread t = new Thread(a);
        }
}
```

f. Use thread method for work the thread

Using thread class reference we can use Thread class method for manage thread like as start() etc

Thread Priority

```
class ABC implements Runnable
{
  public void run()
  {
    try{
          for(int i=1; i<=5; i++)
                {
                  System.out.printf("I = %d\n",i);
                  Thread.sleep(10000);
                }
        }
```

```
            catch(Exception ex)
            { System.out.println("Error is  "+ex);
            }
   }
}
public class TestRunApp
{
    public static void main(String x[])
          {
             ABC a = new ABC();
                   Thread t = new Thread(a);
                   t.start();
          }
}
```

Example: Consider we have Two robots and we share some task them and we want to robots should perform task simultaneously in waiting of each other

```
class Box
{ private  int boxId;
  private String boxName;
  public Box(){
              boxId=1;
  }
  public void setBoxId(int id)
  { this.boxId=id;
  }
  public int getBoxId(){
     return boxId;
  }
  public void setBoxName(String boxName)
```

```
 {  this.boxName=boxName;
 }
 public String getBoxName()
 {  return boxName;
 }
 public synchronized void moveBox(String roboName)
 {try{
           int firstRoboBox=1,secondRoboBox=2;
    for(int i=1; i<=5; i++)
   {   if(roboName.equals("First"))
         {
         System.out.println(boxName+" "+(firstRoboBox)+" --->"+roboName);
                        firstRoboBox=firstRoboBox+2;
        }
                  else{
                        System.out.println(boxName+" "+(secondRoboBox)+" ---
>"+roboName);
             secondRoboBox=secondRoboBox+2;
                  }
       Thread.sleep(1000);
           }
 }
  catch(Exception ex)
  { System.out.println("Error is  "+ex);
  }
 }
}
class RoboFirst implements Runnable
{    Box box;
     private String roboName;
```

```
            int boxId=1;
    void setBox(Box box,String roboName)
            {  this.box=box;
               this.roboName=roboName;
            }
    public void run()
            {
               box.moveBox(roboName );
                        ++boxId;
            }
}
class RoboSecond implements Runnable
{ Box box;
     private String roboName;

    void setBox(Box box,String roboName)
            {  this.box=box;
               this.roboName=roboName;
            }
    public void run()
            {
               box.moveBox(roboName);
            }
}
public class RoboApp
{
  public static void main(String x[])
  {
     Box b = new Box();
```

```
            b.setBoxName("Box");
           RoboFirst rf=new RoboFirst();
           rf.setBox(b,"First");

           RoboSecond rs = new RoboSecond();
           rs.setBox(b,"Second");

            Thread first=new Thread(rf);
            first.start();

            Thread second = new Thread(rs);
             second.start();
  }
}
```

Thread Priority and daemon threads

Q. What is thread priority?

Thread priority is a concept to decide the execution or thread starting priority means when we have multiple threads and if we want to know when and which thread should be started then thread priority comes into picture.

There are three types of thread priority?

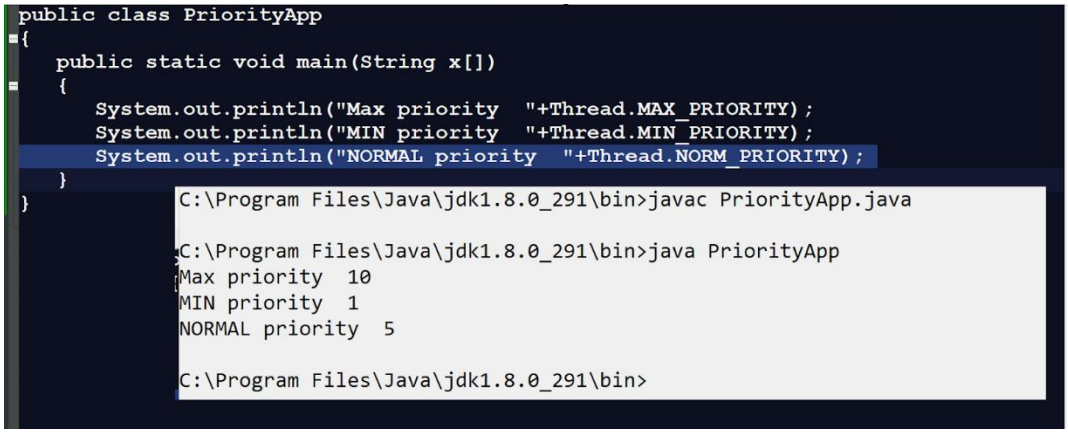
---

1. Max Priority
2. Min Priority
3. Norm Priority

If we want to work with thread priority we have some integer constant provided by Thread class and these constant values are public static in thread class

```
class Thread
{
  public static final int MAX_PRIORITY=10;
  public static final int MIN_PRIORITY=1;
  public static final int NORM_PRIORITY=5;
}
```

So you can access thread priority by using class name because all priority constant variables are static



If we want to set priority on thread we have following method

void setPriority(int priority): this function is used for set the thread priority

int getPriority(): this function is used for return priority of thread or current priority of thread.

Note: windows operating system is not support to thread priority of java properly

Example with source code

```
class A extends Thread
{
        public void run()
        {
          try
```

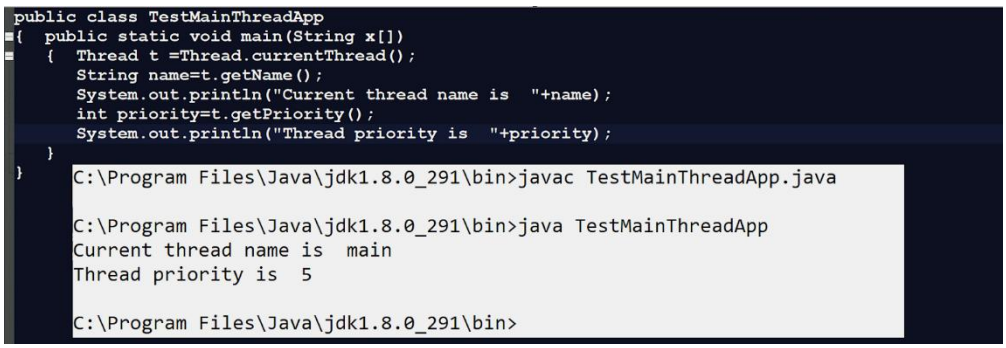
```
        {
                for(int i=1; i<=5; i++)
                { System.out.printf("Thread First = %d\n",i);
                        Thread.sleep(1000);
                }
        }
    catch(Exception ex)
        { System.out.println("Error is  "+ex);
        }
        }

}
class B extends Thread
{
         public void run()
        {
         try
        {
                for(int i=1; i<=5; i++)
                { System.out.printf("Thread Second = %d\n",i);
                        Thread.sleep(1000);
                }
       }
    catch(Exception ex)
        { System.out.println("Error is  "+ex);
        }
        }
}
```

```
public class PriorityApp
{
  public static void main(String x[])
  {
    A a1 = new A();
          B b1 =new B();
          b1.setPriority(Thread.MAX_PRIORITY);
          a1.setPriority(Thread.MIN_PRIORITY);
          a1.start();
          b1.start();

  }
}
```

Note: every thread has a default priority known as normal priority

Q. Can we create an application or java program without thread?

No, we cannot create any java program without thread because every java program contains the default thread name as main thread and it is started by JVM and the default priority of main thread is normal priority.



```
public class TestMainThreadApp
{  public static void main(String x[])
   {  Thread t =Thread.currentThread();
      String name=t.getName();
      System.out.println("Current thread name is  "+name);
      int priority=t.getPriority();
      System.out.println("Thread priority is  "+priority);
   }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac TestMainThreadApp.java

C:\Program Files\Java\jdk1.8.0_291\bin>java TestMainThreadApp
Current thread name is  main
Thread priority is  5

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Q. What is daemon thread in JAVA?

Daemon thread means thread which executes in the background to provide service to another thread called a daemon thread.

Means in short we can say background working thread known as daemon thread in JAVA.

Example: Garbage collector is daemon thread in JAVA

Q. Can a user own thread as daemon thread?

Yes user can create own thread as daemon thread and for that user has following methods

void setDaemon(boolean): this method helps us to set user thread as daemon thread when we pass true value in it

boolean isDaemon(): this method checks if the working thread is a daemon thread or not if the working thread is a daemon thread return true otherwise return false.

Example of daemon thread

```
class M extends Thread
{       public void run()
{
                        try{
                                for(int i=1; i<=5; i++)
                                { System.out.println("I is  "+i);
                                  Thread.sleep(1000);
                                }
                        }
                        catch(Exception ex)
                        { System.out.println("Error is  "+ex);
                        }
            }
}
public class TestMainThreadApp
```

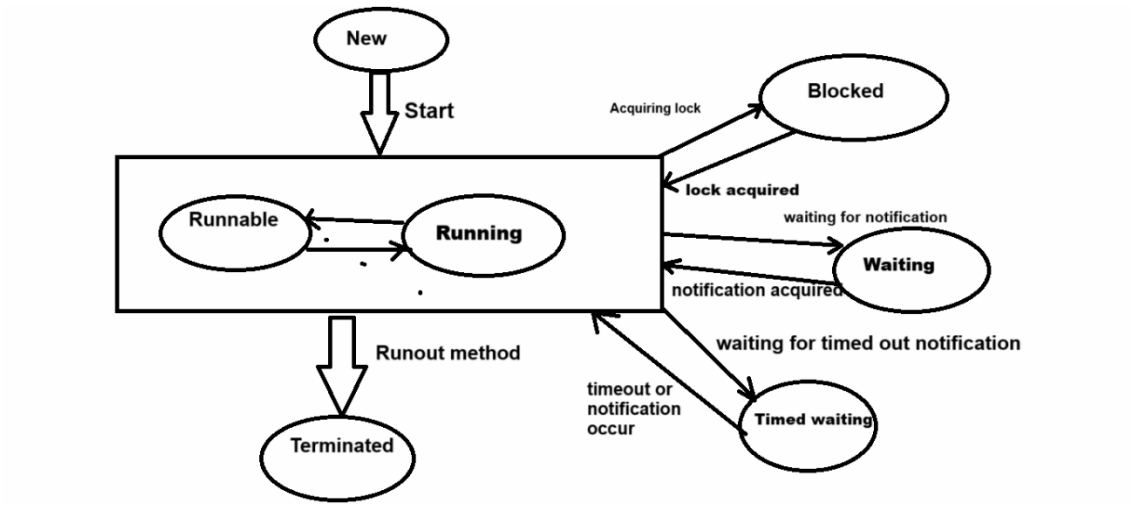
```
{  public static void main(String x[])
  {       M m1 = new M();
          m1.setDaemon(true);
          m1.start();
  }
}
```

Thread LifeCycle

Thread life cycle is the process of tracking thread creation to thread termination as well as intermediate activity or work done by thread called thread life cycle.

If we want to work with thread life cycle we have some important phases of thread life cycle

1. New State
2. RunnableState
3. Blocked State
4. Waiting State
5. Timed waiting state
6. Terminate state



Above diagram shows the thread life cycle

New State: when we create an object of thread class but thread not yet started called as new state of thread.

```
class A  extends Thread
{
  public void run()
  {
   //write here your logics
  }
}

A a1 = new A();  // new state
```

Runnabe state: if thread is ready to run is moved to runnable state or may running thread status is also known as runnable state and thread can move for run any instant of time is responsibility of thread scheduler means thread schedule is responsible for allocate cpu to thread for execution means in short we can say thread is in runnable state when we call a start() method or when run() is in execution

```
class A  extends Thread
{
  public void run()
  {
   //write here your logics
  }
}

A a1 = new A();  // new state
a1.start(); //runnable state
```

blocked : thread will be block state when it is trying to acquired lock but the current lock is acquired by the other thread. So thread will be move from block to runnable state.

Waiting state: thread will be in waiting state when its call wait() method or join method it will move from runnable state when other thread will notify or thread will be terminated.

Timed waiting: thread is time waiting when we call thread with timeout parameter and when specified time out then thread automatically move from waiting to runnable state e.g sleep() method of Thread class

Terminated state: terminate means when we destroy thread or stop the execution of thread called as terminated state

So termination happen in three condition

1. User can terminate thread manually by calling stop method
2. When runtime exception occur in thread execution then thread may be terminate
3. When thread execute its all logic then thread scheduler can terminate thread properly

//Remaining point we want to cover in JDK 1.8

a. Executor services

b. Thread pooling

c. Concurrency API

Collection

Q. What is a Collection?

Collection is a ready made implementation of data structure

Q.What is the purpose of Collection?

---

1. Ability to store any kind of data.
2. We can extends its size at run time as well as shrink its size at run time

Means we can store infinite data in collection

3. Provide ready made implementation of data structure

Note: if we think about array then we cannot modify its size at run time

Or we need to write manual logic to extend the array size.

When we use array we need to manual logic of data structure like as sorting ,searching etc

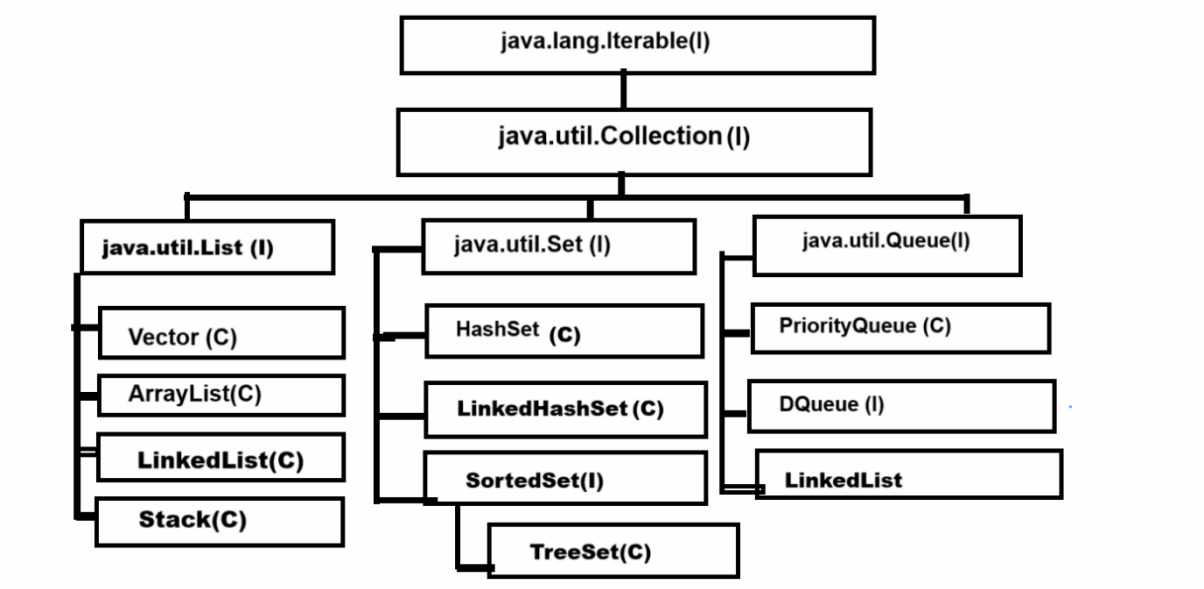
Note: when we create array of Object class then we can store any kind of data in array

```
package org.techhub;
public class TestcollApplication {
	public static void main(String[] args) {
		Object obj [] = new Object[5];
		obj[0]=100;
		obj[1]=false;
		obj[2]=5.4f;
		obj[3]=new java.util.Date();
		obj[4]="good";
		for(int i=0; i<obj.length;i++) {
			System.out.println(obj[i]);
		}  }
}
```

If we think about above code we have collection of different types of data using Object class array but there is limitation when we want to any data structure operation on above code we need to write manually like as searching, insertion of element , deletion of element etc

So Java Suggest we provide ready implementation of data structure to developer known as collection

If we want to work with Collection in JAVA we need to know the following classes and interface hierarchy



If we think about above diagram we have top most interface name as java.lang.Iterable

Q. Why is java.lang.Iterable the parent of all Collection?

The major goal of Iterable interface is to fetch data from collection and data fetching is common operation in all collections so Iterable is parent of all collection because Iterable interface contain iterator() method which return reference of Iterator interface and using Iterator interface we can fetch data from collection

```
interface Iterable
{  Iterator  iterator();
}
```

iterator() method return reference of Iterator interface and Iterator interface contain some inbuilt methods which is used for fetch data from collection like as

boolean hasNext() : this method check data present in collection or not if present return true otherwise return false

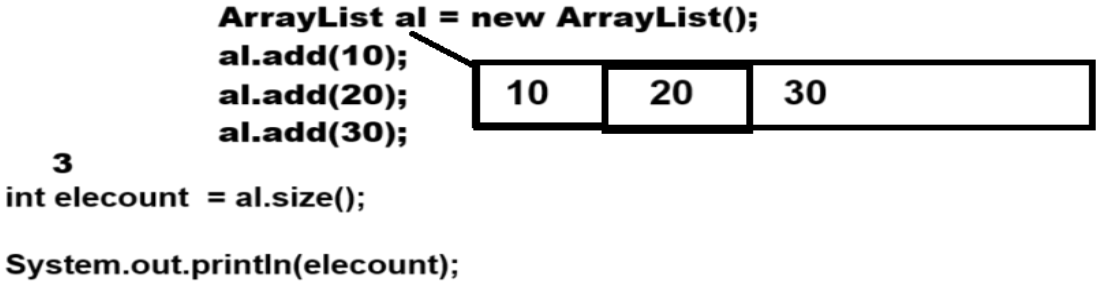
Object next() : this method can fetch data from collection and move cursor on next element

void remove() : this method can remove data from collection using iterator

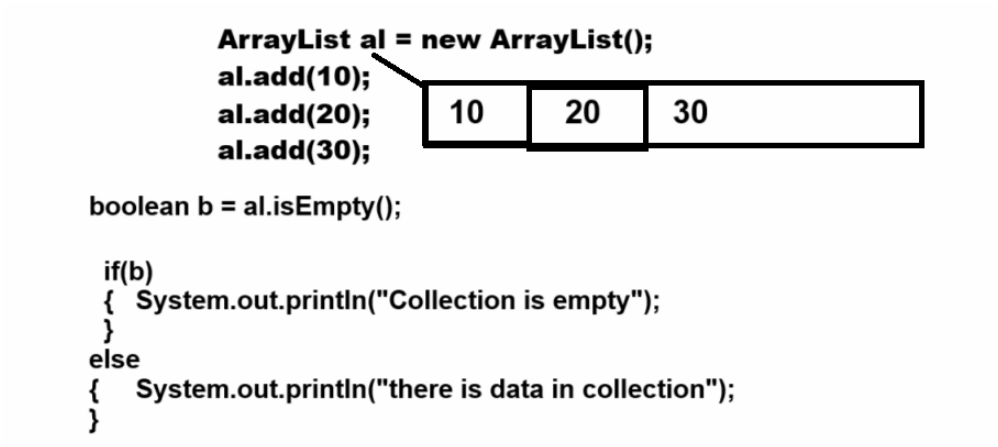
Collection interface

Collection interface contain common methods which is required for perform daily operation on collection like as adding element, removing element ,checking size etc

public abstract int size(): this method is used for return size of collection or return number of element present in collection

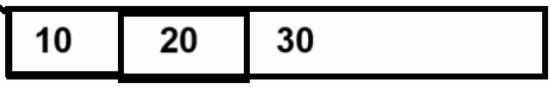


public abstract boolean isEmpty(): this method checks if the collection is empty or not if empty return true otherwise return false.



public abstract boolean contains(java.lang.Object): this method is used for checking data present in collection or not means normally we use this method for data searching purpose.

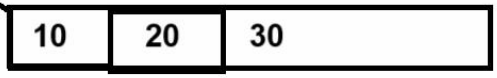
```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```




```
boolean  b = al.contains(20);

   if(b)
   { System.out.println("Data present");
   }
   else
   {
     System.out.println("Data not present");
   }
```

public abstract java.util.Iterator<E> iterator(): it helps us to fetch data from collection and it is a universal iterator.

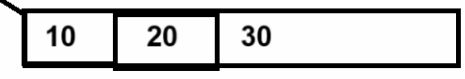
```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```




```
Iterator   i = al.iterator();

while (  i.hasNext() )
{
    Object   obj  = i.next();

 System.out.println(obj);

}
```

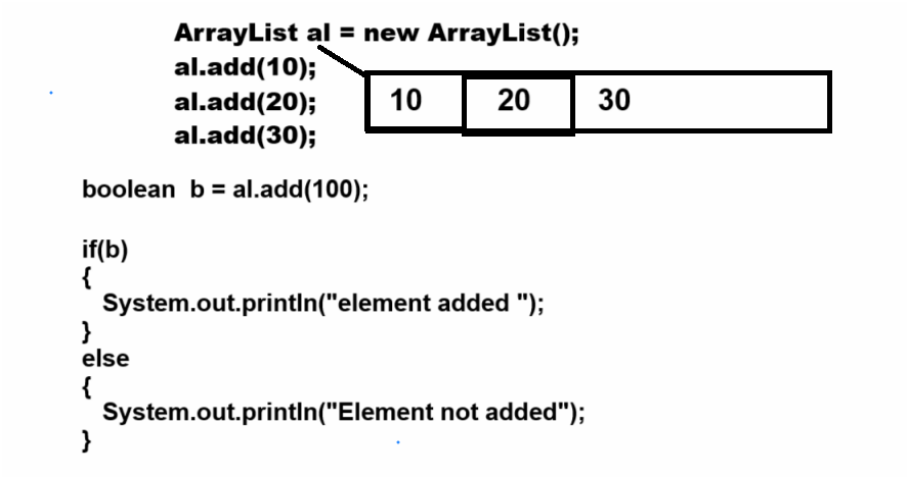
public abstract java.lang.Object[] toArray(): this method can convert any collection in object class array.

```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
```



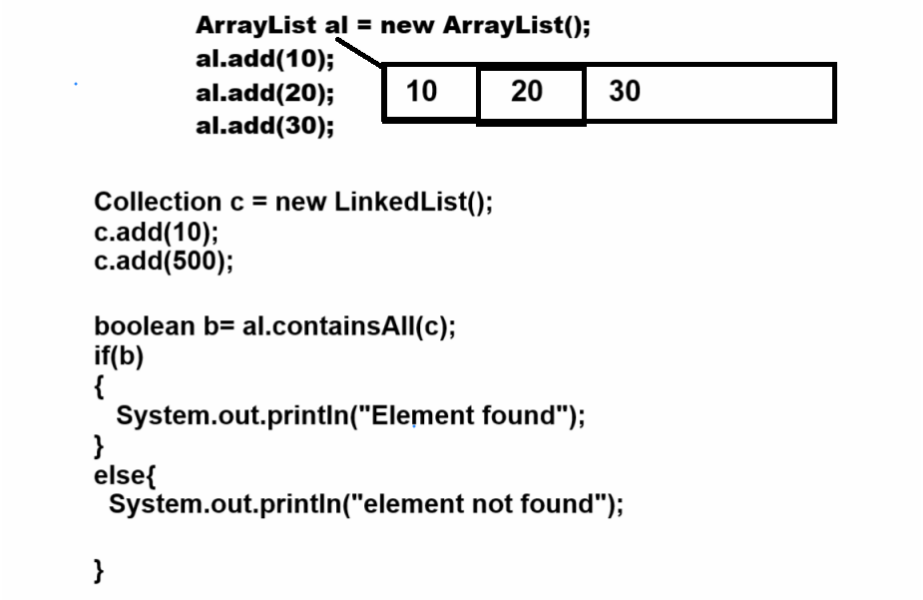

```
Object  obj []  = al.toArray();
for(int i=0; i<obj.length; i++)
{
  System.out.println(obj[i]);
}
```

public abstract boolean add(E) : this method can add a new element in collection and return true if element added otherwise return false.



public abstract boolean remove(java.lang.Object): remove element from collection and element removed successfully return true otherwise return false.

public abstract boolean containsAll(java.util.Collection<?>):



public abstract boolean addAll(java.util.Collection<? extends E>): this method helps us to add more than one element in collection at time.

public abstract boolean removeAll(java.util.Collection<?>): this method also helps us to remove more than one element at a time from collection.

There are three types of collection we have

List : List collection can store duplicate values or data and manage data by using indexing technique means we can say list collection is implementation of linear data structure internally.

Set : Set collection cannot allow duplicated data as well as can generate random data also provide data in sorted format using TreeSet collection etc

Means set collection is implementation of non linear and hash based data structure

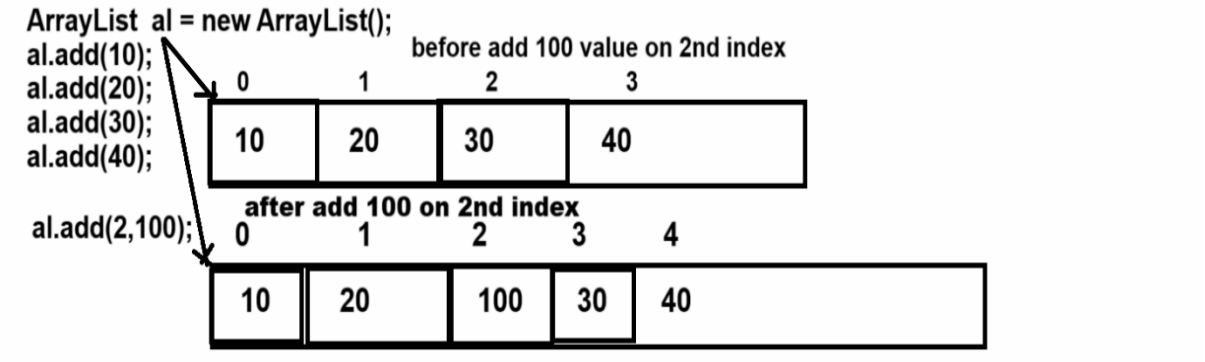
Queue : Queue in data structure allow to store duplicate values and can manage data using first in first out format and provide indexing to data management etc

Now we want to discuss about the List Collection

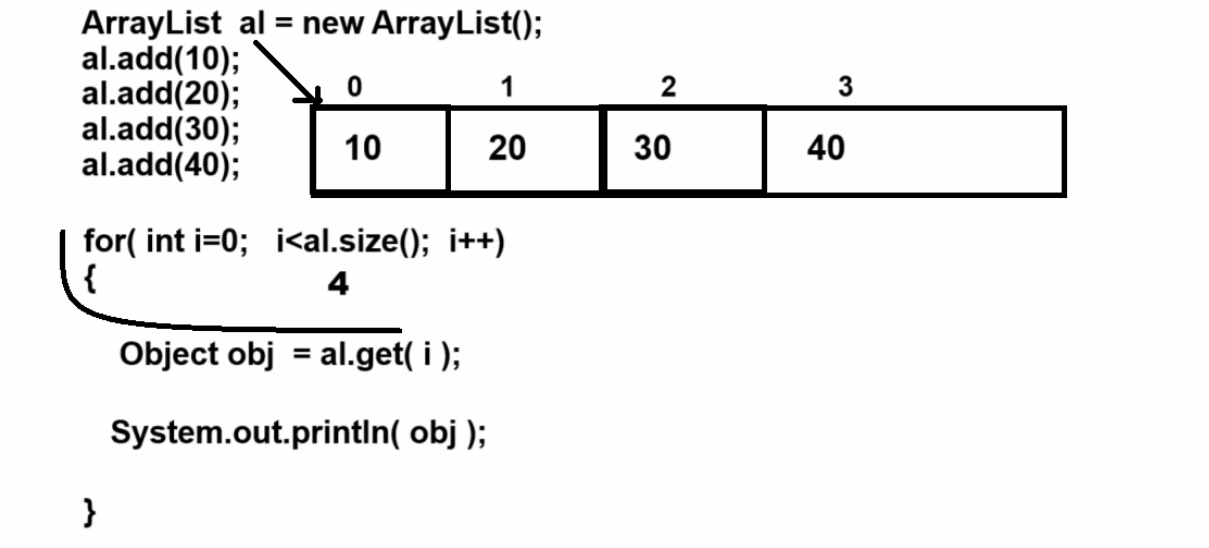
Methods of List Collection

List Collection contains all methods from Collection interface as well as List contains its own additional method also.

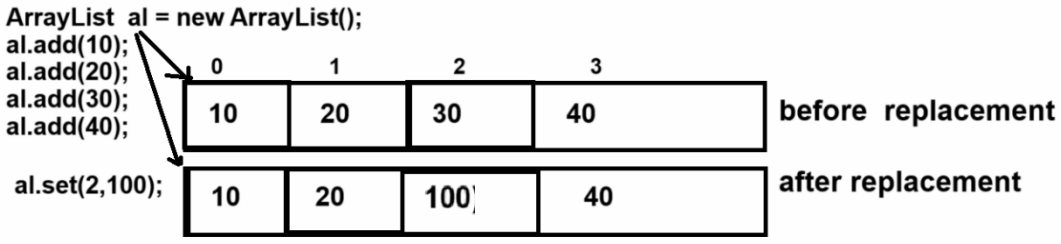
public abstract void add(int, E): this method is used for adding a new element on a specified index and moving previous values from that index onto the next index automatically.



public abstract E get(int): this method can return data from ArrayList collection using its index

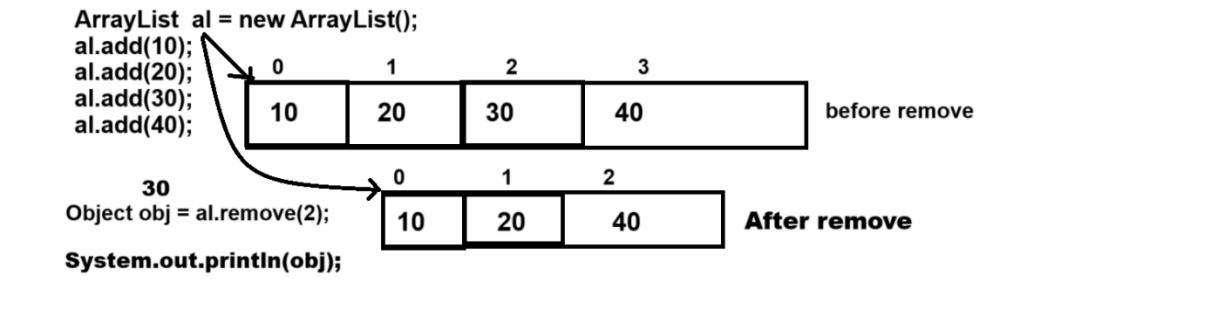


public abstract E set(int, E): this method is used to replace data on specified index.



Note: in above example set() method replace 100 value on index 2

public abstract E remove(int): this method is used for remove data using its index and return removed value as output

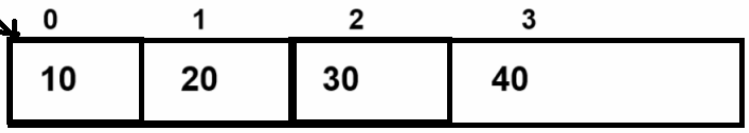


public abstract int indexOf(java.lang.Object): this method can return index of particular element if element or value not found in collection return -1

Note: this method can use for two purpose

a. Searching element from collection

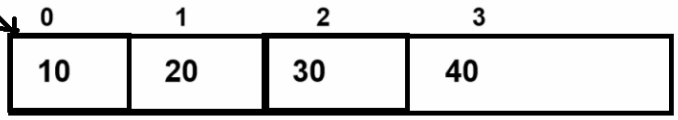
```
ArrayList  al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
al.add(40);
```




```
Scanner   xyz  = new Scanner(System.in);
System.out.println("enter value for search");
int value=xyz.nextInt(); //30

int index = al.indexOf(value);

 if(index!=-1)
 {  System.out.println("Data found");
 }
 else
 { System.out.println("Data not found");
 }
```

b. Before call remove method for avoid IndexOutOfBoundsException

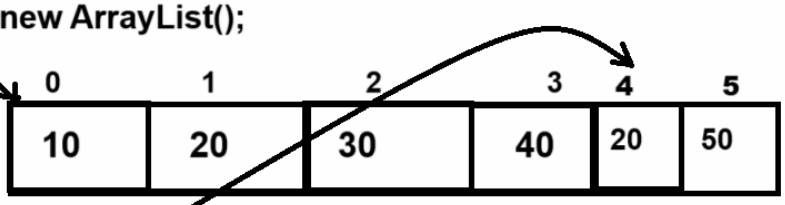
```
ArrayList  al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
al.add(40);
```




```
System.out.println("enter value for delete ");
int value = xyz.nextInt(); // 30
2 int index= al.indexOf(value); //30
if(index!=-1)
{
   al.remove(index);  //20
}
else
{
    System.out.println("value not found for remove or delete");
}
```

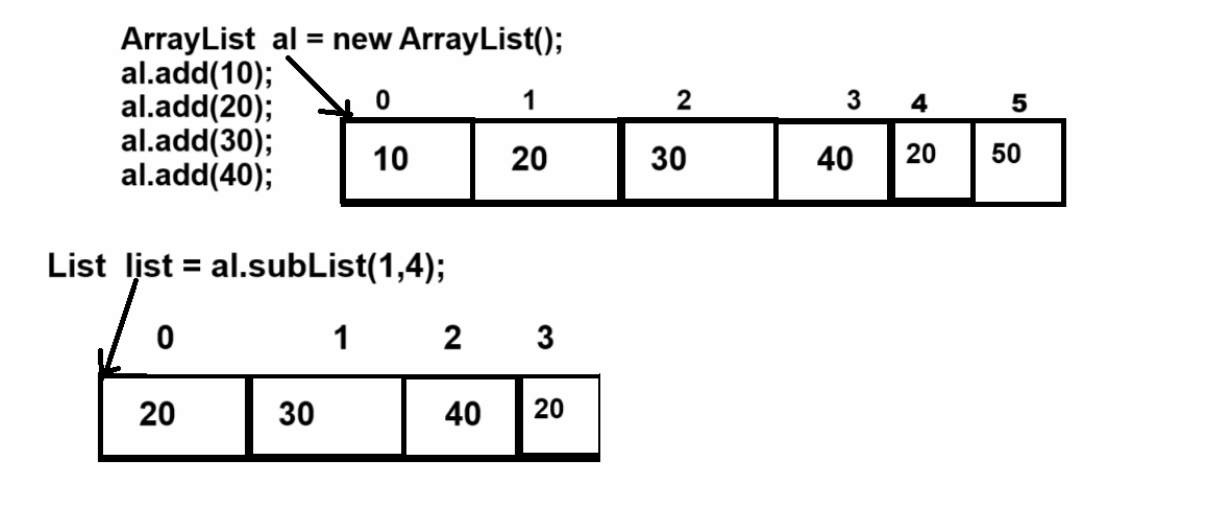
public abstract int lastIndexOf(java.lang.Object): this method return the last occurrence of index

```
ArrayList  al = new ArrayList();
al.add(10);
al.add(20);
al.add(30);
al.add(40);
```




```
 4
int index = al.lastIndexOf(20);
```

public abstract java.util.List<E> subList(int, int): this method helps us extract the some specified portion from List collection between two indexes.



Vector class

Vector is a dynamic array internally and it is known legacy collection

Q. What is a legacy collection?

Legacy collection means a classes which is part of previous version of JAVA before collection but later they are added as part of collection called as legacy collection

Vector was added as part of the collection in JDK 1.2 before that vector known as dynamic array in JAVA.

Important points related with vector

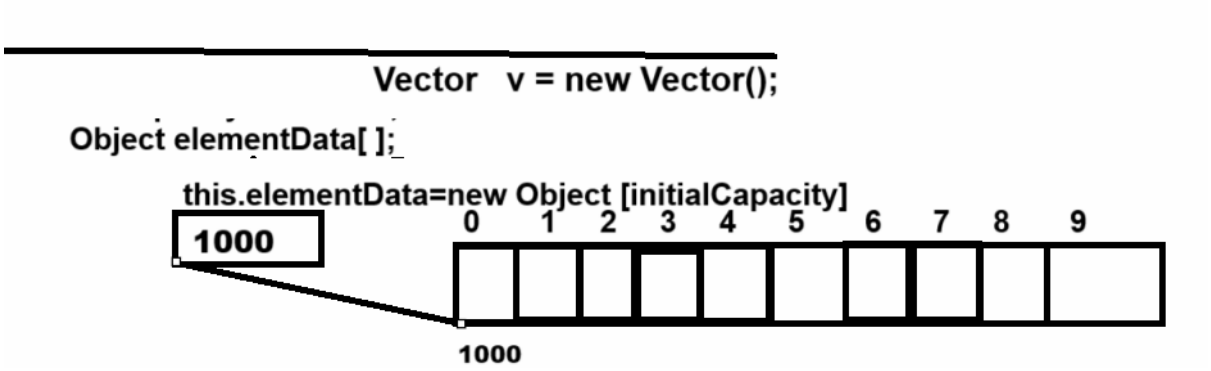
---

1. Vector is thread safe collection
2. Vector legacy collection
3. Default capacity of Vector is 10
4. Vector allocate memory double thant its current capacity so the threshold of Vector is 1.0 or 100%

If we want to work with Vector we have some constructor provided by Vector to us

1. **Vector():** When we use this constructor creating Vector object then internally we get array of Object class with initial capacity 10

   Means default capacity of Vector is 10.



If we want to cross verify the default capacity of a vector is 10 then we have a method name as int capacity() using this method we can check the capacity of the vector.

```
1  package org.techhub;
2  import java.util.*;
3  public class TestVectorApp {
4
5      public static void main(String[] args) {
6          Vector v = new Vector();
7          int capacity =v.capacity();
8          System.out.println("Initial Capacity of vector is "+capacity);
9      }
10 }
11
```

```
Initial Capacity of vector is 10
```

Important point: when we store the value more than the current capacity of a vector then the vector occupies double memory than its current capacity.

Example: if we create an object of Vector using default constructor then the default capacity of Vector is 10 and if we try to add the 11th element in vector then its new capacity is 20 and if we try to add 21th element then the new capacity is 40 and so on.

```java
package org.techhub;
import java.util.*;
public class TestVectorApp {

    public static void main(String[] args) {
         Vector v = new Vector();
         int capacity =v.capacity();
         System.out.println("Initial Capacity of vector is "+capacity);
         v.add(10);
         v.add(20);
         v.add(30);
         v.add(40);
         v.add(10);
         v.add(20);
         v.add(30);
         v.add(40);
         v.add(10);
         v.add(20);
         v.add(30);
         v.add(40);
         System.out.println("Size of vector is "+v.size());
         capacity =v.capacity();
         System.out.println("After Capacity cross "+capacity);

    }
}
```

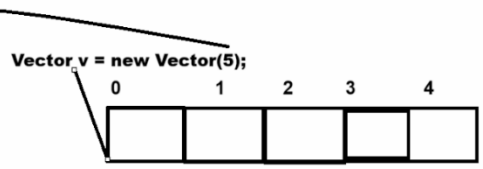
Output

```
Initial Capacity of vector is 10
Size of vector is 12
After Capacity cross 20
```

Note: user can set initial capacity of vector according to his requirement.

2. Vector(int initialCapacity) : this constructor can create a vector with initial capacity as per the user choice**.**

```java
class Vector
{  Object elementData[];
   Vector()
   {  this(10);
   }
   Vector(int initialCapacity)
   {
       this(initialCapacity,0);
   }
   Vector(int initialCapacity,int incrementalCapacity)
   {
        if(initialCapacity < 0)
        { throw new IllegalArgumentException("Illegal Argument exception  "+initialCapacity);
        }
       else{
        this.elementData=new Object[initialCapacity];

     }
   }
   Vector(Collection c)
   {
   }
}
```



Note: if we set initial capacity of vector according to user choice and if vector cross its current capacity then vector allocate memory double than its current capacity

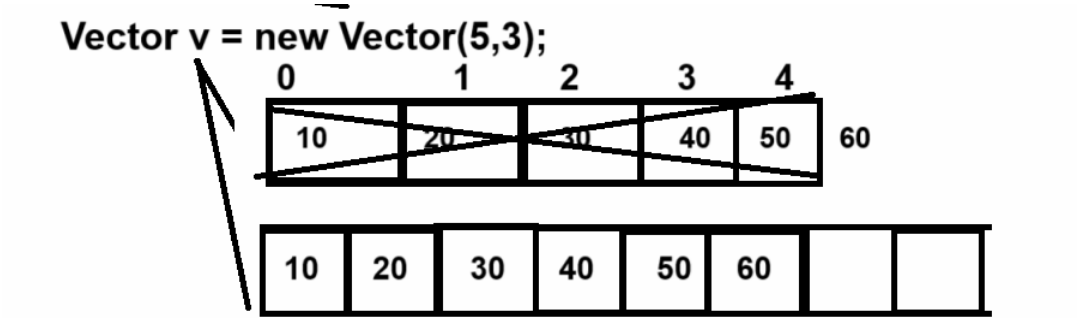
Note: if we want to decide the incremental capacity of Vector according to user choice not by default double then we have a constructor with two parameters.

3. Vector(int initialCapacity,int incrementalCapacity)

Parameter Details

int initialCapacity: this parameter is used for set initial capacity of vector.

int incrementalCapacity: this parameter is used for decide the incremental capacity of vector



Note: if we think about above constructor we create vector object using Vector v=new Vector(5,3) here we set initial capacity 5 and we set incremental capacity 3 means when we try to add 6th element in vector then vector increase its capacity by 3 blocks only

4. Vector(Collection) : this constructor helps us to copy data from another collection and store it in a Vector object.

```
package org.techhub;

import java.util.*;

public class TestVectorApplication {

	public static void main(String[] args) {

		ArrayList al = new ArrayList();

		al.add(10);

		al.add(20);

		al.add(30);

		al.add(40);

		Vector v = new Vector(al);

		System.out.println(v);

	}

}
```

Example by using Vector

_____

Case 1: Add New Element in Vector

Case 2: View All Elements

Case 3: Count number of element

Case 4: Search element by contains

Case 5: Search Element by index

Case 6: delete element by using index

Case 7: Fetch elements using get() method

Case 8: SubList

Case 9: Remove by element

Example with source code

_____

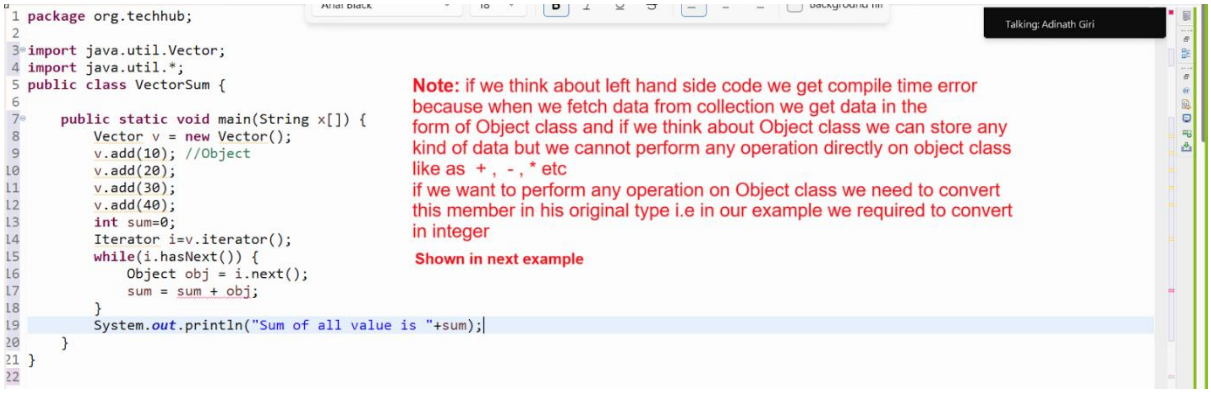
```
package org.techhub;
import java.util.*;
public class TestVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		do {
		Scanner xyz = new Scanner(System.in);
		System.out.println("1: Add Element");
		System.out.println("2: View All  ");
		System.out.println("3: Count number of element ");
		System.out.println("4:Search element by contains method");
		System.out.println("5:Search Element by index");
		System.out.println("6:Delete by using its index");
		System.out.println("7:Fetch elements by using get method");
		System.out.println("8: SubList");
```

```
System.out.println("9: Remove element by value");
System.out.println("Enter your choice");
        int choice = xyz.nextInt();
        switch (choice) {
        case 1:
                System.out.println("Enter data in vector");
                int value = xyz.nextInt();
                boolean b = v.add(value);
                if (b) {
                        System.out.println("element added");
                } else {
                        System.out.println("element not added");
                }
                break;
        case 2:
                Iterator i = v.iterator();
                while (i.hasNext()) {
                        Object obj = i.next();
                        System.out.println(obj);
                }
                break;
        case 3:
System.out.println("Number of element in vector " + v.size());
                break;
        case 4:
                System.out.println("Enter value for search ");
                value = xyz.nextInt();
                b = v.contains(value);
                if (b) {
```

```
                System.out.println("Value found");
            } else {
                System.out.println("Value not found");
            }
            break;
        case 5:
            System.out.println("Enter value for search ");
            value = xyz.nextInt();
            int index = v.indexOf(value);
            if (index != -1) {
                System.out.println("Data found");
            } else {
                System.out.println("Data not found");
            }
            break;
        case 6:
            System.out.println("Enter value for delete ");
            value = xyz.nextInt();
            index = v.indexOf(value);
            if (index != -1) {
                Object obj = v.remove(index);
                System.out.println("Data Deleted " + obj);

            } else {
                System.out.println("Data not found");
            }
            break;
        case 7:
            for (int k = 0; k < v.size(); k++) {
```

```
                    Object obj = v.get(k);
                    System.out.println(obj);
                }
                break;
            case 8:
            System.out.println("enter start index and end index");
                int startIndex = xyz.nextInt();
                int endIndex = xyz.nextInt();
                if (startIndex >= 0 && endIndex < v.size()) {
                    List list = v.subList(startIndex, endIndex);
                    for (Object obj : list) {
                        System.out.print(obj + "\t");
                    }
                }
                break;
            case 9:
                System.out.println("Enter value for delete ");
                value = xyz.nextInt();
                v.remove((Object)value);
                break;
            case 10:
                System.exit(0);
                break;
            default:
                System.out.println("Wrong choice");
            }
        } while (true);
    }
}
```

Example: WAP to create vector and store 5 values init and calculate its sum



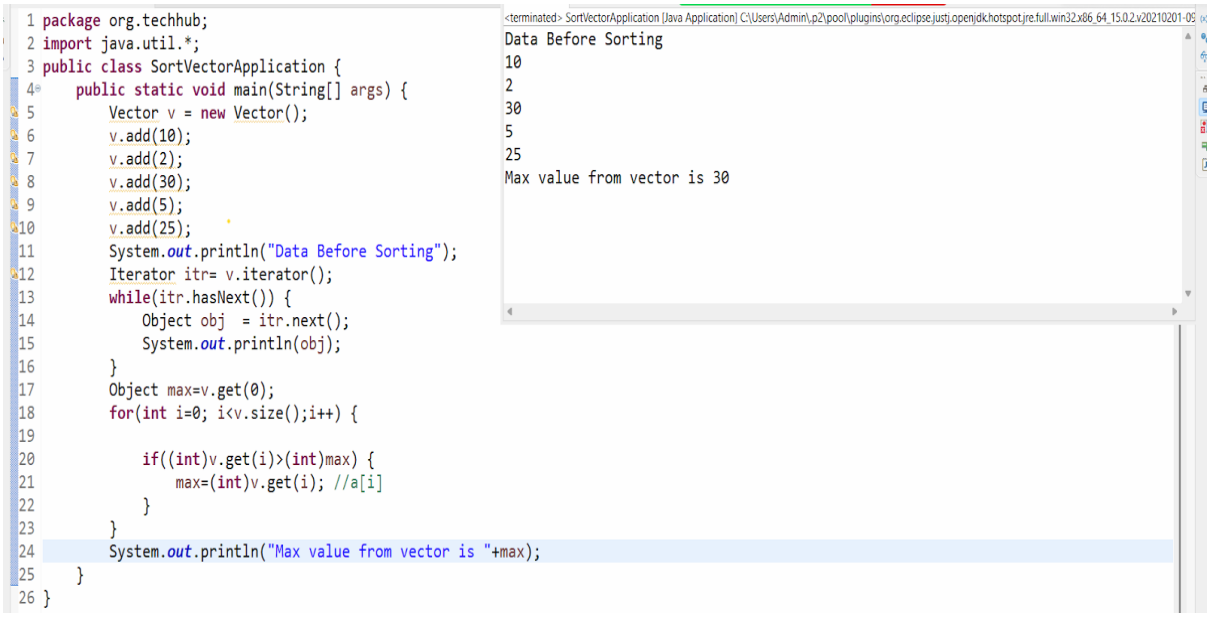

```
package org.techhub;

import java.util.Vector;
import java.util.*;
public class VectorSum {

    public static void main(String x[]) {
        Vector v = new Vector();
        v.add(10); //Object
        v.add(20);
        v.add(30);
        v.add(40);
        int sum=0;
        Iterator i=v.iterator();
        while(i.hasNext()) {
            Object obj = i.next();
            sum = sum + obj;
        }
        System.out.println("Sum of all value is "+sum);
    }
}
```

Example: WAP to create Vector and store 5 integer values in it and perform sorting operations on it.

```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		v.add(10);
		v.add(2);
		v.add(30);
		v.add(5);
		v.add(25);
		System.out.println("Data Before Sorting");
		Iterator itr= v.iterator();
		while(itr.hasNext()) {
			Object obj  = itr.next();
			System.out.println(obj);
		}
		//perform sorting
		for(int i=0; i<v.size();i++) {
			for(int j=(i+1); j<v.size();j++) {

				Object prev=v.get(i);
				Object next=v.get(j);
				if((int)prev>(int)next) {
					v.set(i, next);
					v.set(j, prev);
				}
			}
```

```
                }
                System.out.println("Data After Sorting");
                 itr= v.iterator();
                while(itr.hasNext()) {
                        Object obj  = itr.next();
                        System.out.println(obj);
                }
        }
}
```

Example: WAP to create Vector and store 5 values in it and find the max value from Vector without using Collections.max()



```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
        v.add(10);
        v.add(2);
        v.add(30);
        v.add(5);
        v.add(25);
        System.out.println("Data Before Sorting");
        Iterator itr= v.iterator();
        while(itr.hasNext()) {
            Object obj  = itr.next();
            System.out.println(obj);
        }
        Object max=v.get(0);
        for(int i=0; i<v.size();i++) {

            if((int)v.get(i)>(int)max) {
                max=(int)v.get(i); //a[i]
            }
        }
        System.out.println("Max value from vector is "+max);
    }
}
```

```
<terminated> SortVectorApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-05
Data Before Sorting
10
2
30
5
25
Max value from vector is 30
```

Example: WAP to store five words in Vector and create single string or line of statement using Vector

```java
package org.techhub;
import java.util.*;
public class SortVectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
         v.add("Good");
         v.add("bad");
         v.add("abc");
         v.add("mno");
         v.add("pqr");

         String str="";
         Iterator i=v.iterator();
         while(i.hasNext()) {
             Object obj = i.next();
             str=str+" "+(String)obj;//obj.toString()
         }
         System.out.println("String is  "+str);
    }
}
```

```
<terminated> SortVectorApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\java
String is   Good bad abc mno pqr
```

How to store user defined objects in Collection

Q. Why do we need to store user-defined objects in collections?

When we want to add more than one element of different type as a single value in a collection then we can store user defined objects in the collection.

Example: we want to create a collection of students with data id,name and per here id has integer data type, name has string data type and per has float data type.

If we want to solve above problem then we can create POJO class name as Student with field id, name and per and store data in Student object and store in collection i.e in Vector according to our example and when we want to fetch data from collection then we required convert that data in Student object from Object class.

Example with source code

```
package org.techhub;

public class Student {

        private int id;

        private String name;

        public int getId() {

                return id;

        }
```

```
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPer() {
		return per;
	}
	public void setPer(float per) {
		this.per = per;
	}
	private float per;
}
package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		 Student s1 = new Student();
		 s1.setId(1);
		 s1.setName("ABC");
		 s1.setPer(90.5f);
		 Student s2 = new Student();
		 s2.setId(2);
		 s2.setName("PQR");
```

```
        s2.setPer(70.5f);
        Student s3 = new Student();
        s3.setId(3);
        s3.setName("XYZ");
        s3.setPer(85.5f);

        v.add(s1);
        v.add(s2);
        v.add(s3);
        //Object - generalize format  - Student
        Iterator i=v.iterator();
        while(i.hasNext()) {
            Object obj = i.next();
            Student s=(Student)obj;
           System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
        }
    }
}
```

```
package org.techhub;

public class Student {
    private int id;
    private String name;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPer() {
        return per;
    }
    public void setPer(float per) {
        this.per = per;
    }
    private float per;
}
```

```
package org.techhub;
import java.util.*;
public class SortVectorApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
        Student s1 = new Student();
        s1.setId(1);
        s1.setName("ABC");
        s1.setPer(90.5f);

        Student s2 = new Student();
        s2.setId(2);
        s2.setName("PQR");
        s2.setPer(70.5f);

        Student s3 = new Student();
        s3.setId(3);
        s3.setName("XYZ");
        s3.setPer(85.5f);

        v.add(s1);
        v.add(s2);
        v.add(s3);
        //Object - generalize format  - Student
        Iterator i=v.iterator();
        while(i.hasNext()) {
            Object obj = i.next();
            Student s=(Student)obj;
           System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
        }
    }
}
```

Note: if we think about left hand side code we create object of the class first and we store data using setter method so it may be increase code size
we want to initialize at the time of object or may be later after object creation using a setter method
then we can overload the constructor in POJO class shown in next example.

Example : Source code with parameterized constructor

package org.techhub;

```
public class Student {
	private int id;
	private String name;
	public Student() {

	}
	public Student(String name,int id,float per) {
		this.name=name;
		this.id=id;
		this.per=per;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPer() {
		return per;
	}
	public void setPer(float per) {
		this.per = per;
	}
```

```
	private float per;
}
package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		 Student s1 = new Student("ABC",1,90.5f);

		 Student s2 = new Student("PQR",2,70.5f);

		 Student s3 = new Student("XYZ",3,85.0f);

		 v.add(s1);
		 v.add(s2);
		 v.add(s3);
		 //Object - generalize format  - Student
		 Iterator i=v.iterator();
		 while(i.hasNext()) {
			Object obj = i.next();
			Student s=(Student)obj;
	System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
		 }
	}
}
```

Or

```
package org.techhub;


public class Student {
	private int id;
	private String name;
	public Student() {
	}
	public Student(String name,int id,float per) {
		this.name=name;
		this.id=id;
		this.per=per;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPer() {
		return per;
	}
	public void setPer(float per) {
```

```
		this.per = per;
	}
	private float per;
}

package org.techhub;
import java.util.*;
public class SortVectorApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		 v.add(new Student("ABC",1,90.5f));
		 v.add(new Student("PQR",2,70.5f));
		 v.add(new Student("XYZ",3,85.0f));
		 Iterator i=v.iterator();
		 while(i.hasNext()) {
			 Object obj = i.next();
			 Student s=(Student)obj;
			System.out.println(s.getId()+"\t"+s.getName()+"\t"+s.getPer());
		 }
	}
}
```

Example: WAP to create Vector and store five book objects in vector and search book using id from collection.

```
package org.techhub;
import java.util.*;
class Book{
	private int id;
	private String name;
	public Book() {
```

```
	}
	public Book(String name,int id,float price) {
		this.name=name;
		this.id=id;
		this.price=price;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPrice() {
		return price;
	}
	public void setPrice(float price) {
		this.price = price;
	}
	private float price;


}
```

```
public class BookApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		Book b1 = new Book("ABC",1,100);
		Book b2 = new Book("MNO",2,200);
		Book b3 = new Book("PQR",3,300);
		Book b4 = new Book("STV",4,400);

		v.add(b1);
		v.add(b2);
		v.add(b3);
		v.add(b4);
		System.out.println("Display book data");
		Iterator itr=v.iterator();
		while(itr.hasNext()) {
			Object obj = itr.next();
			Book b =(Book)obj;
			System.out.println(b.getId()+"\t"+b.getName()+"\t"+b.getPrice());
		}
		Scanner xyz = new Scanner(System.in);
		System.out.println("Enter book id for search");
		int bid=xyz.nextInt();
		boolean flag=false;
		for(int i=0;i<v.size();i++) {

			Book b=(Book)v.get(i);
			if(b.getId()==bid) {
				flag=true;
				break;
```

```
                }
        }
        if(flag) {
                System.out.println("Book found");
        }
        else {
                System.out.println("Book not found");
        }
    }
}
```

Cursor in Collection

Cursor are the some iterators which help us to fetch data from collection

Types of Cursor in Collection

---

1. **For loop :** but normally we fetch using for loop from collection who maintain the index like as List Collection, Queue collection etc

```
package org.techhub;
import java.util.*;
class Book{
    private int id;
    private String name;
    public Book() {

    }
    public Book(String name,int id,float price) {
        this.name=name;
        this.id=id;
        this.price=price;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    private float price;
}
```

```
public class BookApplication {
    public static void main(String[] args) {
        Vector v = new Vector();
        Book b1 = new Book("ABC",1,100);
        Book b2 = new Book("MNO",2,200);
        Book b3 = new Book("PQR",3,300);
        Book b4 = new Book("STV",4,400);

        v.add(b1);
        v.add(b2);
        v.add(b3);
        v.add(b4);
        for(int i=0;i<v.size();i++)
        {
            Book b=(Book)v.get(i);
         System.out.println(b.getId()+"\t"+b.getName()+"\t"+b.getPrice());
        }
    }
}
```

Note: if we think about above code we fetch data from collection using for loop so it is not good approach because every time perform increment or decrement operation in ALU as well as check condition every time manually so when we have large data set in collection it will impact on perform of data fetching

2. **Enumeration :** Enumeration is a cursor in collection which helps us to fetch data from legacy collections only like as Vector etc

If we want to work with Enumeration or create reference of Enumeration we have elements() method of Collection

Syntax: Enumeration ref = collref.elements();

Note: Enumeration is read only cursor means using Enumeration we can fetch data from collection only not perform any other operation on Collection using Enumeration like as removing element or adding element etc

Methods of Enumeration

_____

boolean hasMoreElements(): this method can help us to check data present in collection or not if present return true otherwise return false.

Object nextElement(): this method can fetch data from collection and move cursor on next element

```
package org.techhub;
import java.util.*;
class Book{
	private int id;
	private String name;
	public Book() {

	}
	public Book(String name,int id,float price) {
		this.name=name;
		this.id=id;
		this.price=price;
	}
	public int getId() {
		return id;
```

```
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public float getPrice() {
		return price;
	}
	public void setPrice(float price) {
		this.price = price;
	}
	private float price;
}
public class BookApplication {
	public static void main(String[] args) {
		Vector v = new Vector();
		Book b1 = new Book("ABC",1,100);
		Book b2 = new Book("MNO",2,200);
		Book b3 = new Book("PQR",3,300);
		Book b4 = new Book("STV",4,400);

		v.add(b1);
		v.add(b2);
		v.add(b3);
```

```
            v.add(b4);
            Enumeration enm = v.elements();
            while(enm.hasMoreElements()) {
                  Object ele=enm.nextElement();
                  Book b=(Book)ele;
                  System.out.println(b.getId()+"\t"+b.getName()+"\t"+b.getPrice());
            }
      }
}
```

3. Iterator : Iterator helps us to fetch data from collection and provide some inbuilt methods to fetch data from collection.

boolean hasNext(): this method checks if an element is present in collection or not if present return true otherwise return false.

Object next(): this method fetches data from collection and moves the cursor on the next element.

void remove(): this method can remove data from collection at the time data traveling or traversing.

How does Iterator works internally with ArrayList,Vector,Stack and LinkedList?

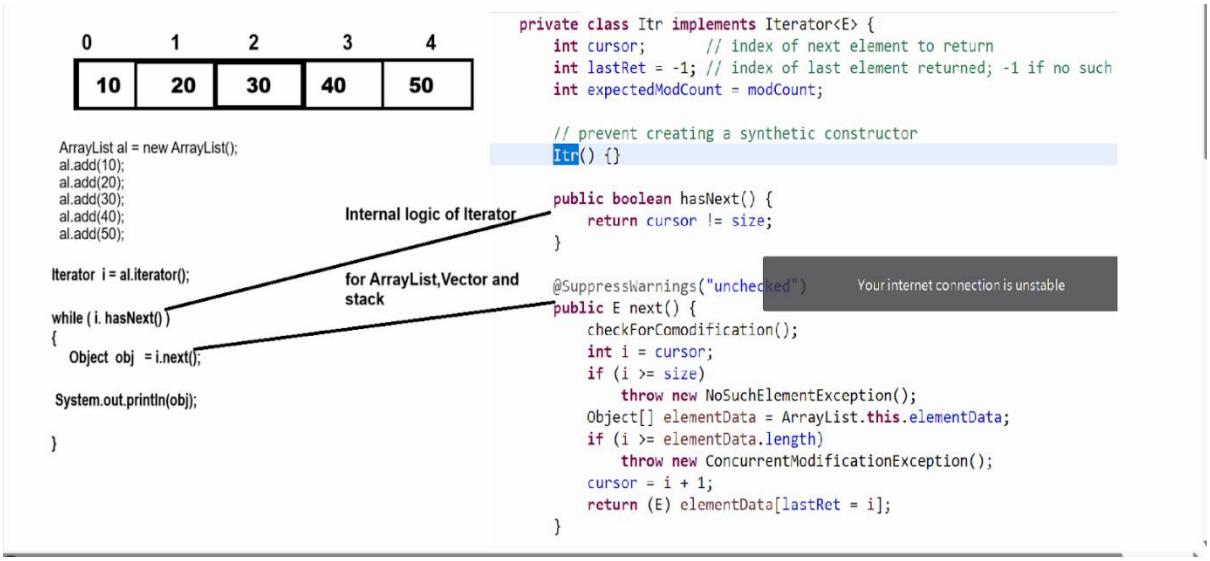
ArrayList,Vector and Stack are the dynamic arrays internally i.e Object [ ] to store elements.

So if we think about Iterator with these three classes

The iterator maintains the index to track the current element position.

next() method return elementData[index++] and hasNext() check index!=size

Show in following diagram



Internal working of Iterator for LinkedList

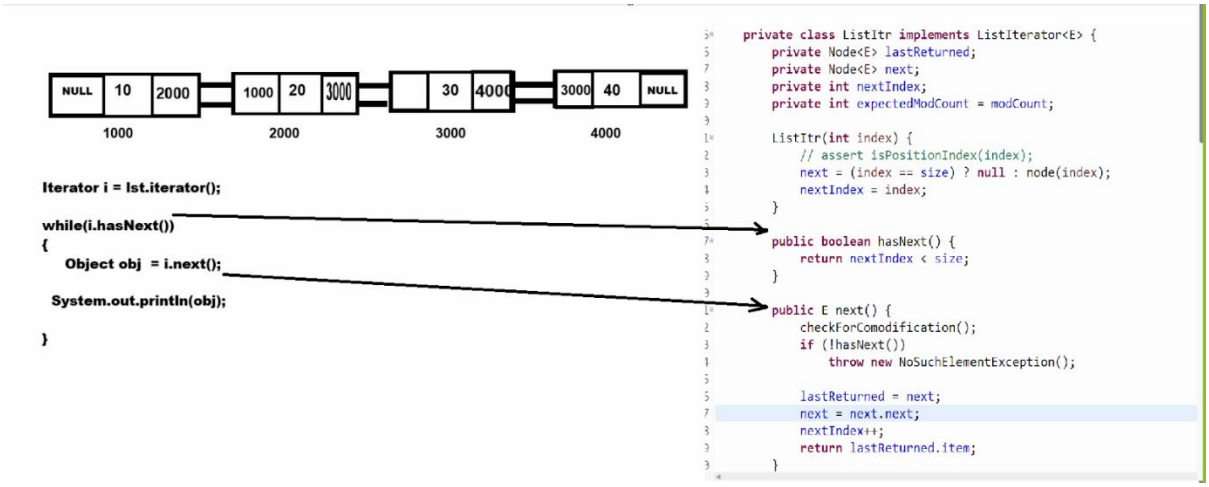
If we think about linked list in Java internally it is doubly LinkedList where each node holds references to its previous and next element as well as item or data for that purpose LinkedList has one inner class name as Node which contain three data i.e prev,next, E item

boolean hasNext(): method check the index<size in linked list for check data availability if index<size return true otherwise return false.

Object next(): this method can point one pointer on current node i.e lastReturn=next and move cursor on next node using ponter or reference of next node i.e next= next.next and increase index by 1

Index++ for move cursor on next node for comparing with size in hasNext() method and return data using lastReturn.item

Shown in following diagram



If we want to work with iterator we need to know two major important points

---

a. ModCount concept

b. Fail Fast Concept

ModCount concept in Java Collection

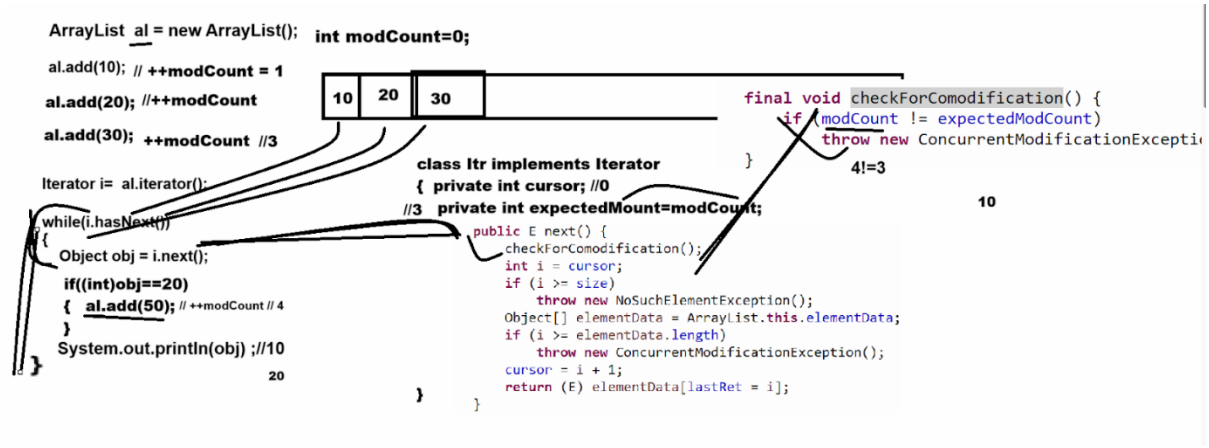
ModCount stands for modification count. It is an integer field to use track number of structural modifications made to collection. It plays a crucial role in ensuring fail-fast behaviour of iterators by detecting concurrent modification duration iteration

ModCount integer field present in collection classes like as ArrayList, LinkedList , HashMap or Vector etc

It keep track of structural modification to the collection such as

1. Adding elements
2. Removing elements
3. Clearing the collection

Note: structural modification are operations that alter the collection size or its internal structure



Example with source

```
package org.techhub;

import java.util.Vector;

import java.util.*;

public class VectorSum {

	public static void main(String x[]) {

		ArrayList v  = new ArrayList();

		 v.add(10);

		 v.add(20);

		 v.add(30);

		 v.add(40);

		 Iterator i = v.iterator();

		 while(i.hasNext()) {

			Object obj = i.next();

			if((int)obj==30) {

				i.remove();

			}

		 }

		 System.out.println(v);

	}

}
```

4. ListIterator : ListIterator is the child interface of Iterator and it is used for travel the collection in forward direction as well as in backward direction and ListIterator only works with List Collection, not other means it is not an universal cursor.

How to create reference of ListIterator

---

boolean hasNext(): check element present in collection or not in forward direction traveling

boolean hasPrevious(): this method check element present in collection or not in backward direction

Object next(): this method can fetch data from collection and move cursor on next element

Object previous(): this method can fetch data from collection and move cursor previous element

void add(Object): add new element in collection at the time of data adding

void remove(): this method can remove data from collection using ListIterator

void set(Object): this method can replace element in collection using ListIterator

Int previousIndex(): return the index of previous element

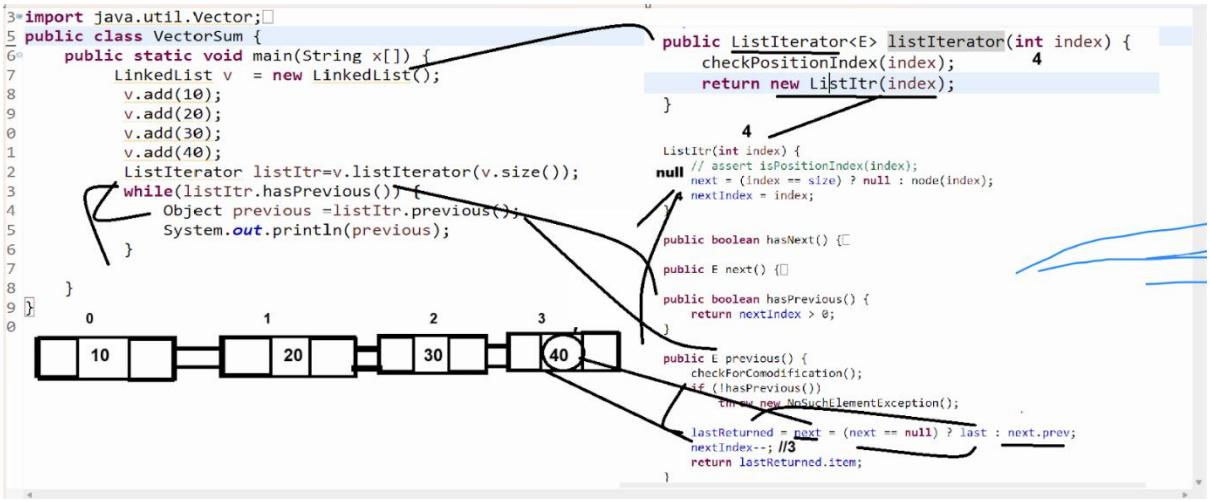
Int nextIndex(): return the index of next element

Example of ListIterator using Vector class

```
package org.techhub;

import java.util.Vector;

import java.util.*;

public class VectorSum {

        public static void main(String x[]) {

                Vector v  = new Vector();

                 v.add(10);

                 v.add(20);

                 v.add(30);

                 v.add(40);
```

```
        ListIterator listItr=v.listIterator(v.size());
        while(listItr.hasPrevious()) {
            Object previous =listItr.previous();
            System.out.println(previous);
        }

    }
}
```

Example of ListIterator using LinkedList



Q. What is the difference between Iterator and List Iterator?

---

a. Travelling capabilities /Traversal capabilities

Iterator: Travel collection or fetch data from collection using forward direction or in only one direction.

ListIterator: Travel collection or fetch data from collection using forward direction as well as using backward direction.

b. Element Modification:

Iterator: Iterator allow only removal operation at the time of travelling using remove() method

ListIterator: ListIterator can allo add(),remove(), set() -replace element in collection at the time of travelling.

c. **Index Access:**

Iterator: Iterator not provide method to access the index of collection at the time of traveling

ListIterator: ListIterator provides two methods to us: int previousIndex() , int nextIndex() for fetch index from collection at the time to traversing.

d. **Methods available**

Iterator: Iterator contain methods boolean hasNext(),Object next(), void remove()

ListIterator: ListIterator contain methods boolean hasNext(),boolean hasPrevious(),Object next(),Objext previous(), int previousIndex(),int nextIndex() , void add(E),void set(E),void remove() etc

e. **Performance Consideration**

1. For ArrayList both Iterator and ListIterator are the fast due to the index access
2. For LinkedList ListIterator is preferable because it avoids redundant travel when moving backward as LinkedList is optimized for bidirectional iteration.

How to create Fail safe collection

If we want to create fail safe collection we have CopyOnWriteArrayList class

Basically CopyOnWriteArrayList is member of java.util.concurrent package and design to handle the concurrent modification like as fail safe collection means not throws ConcurrentModification Exception

```
package org.techhub;

import java.util.Vector;

import java.util.concurrent.CopyOnWriteArrayList;

import java.util.*;

public class VectorSum {

        public static void main(String x[]) {

                CopyOnWriteArrayList v=new CopyOnWriteArrayList();
```

```
                v.add(10);
                v.add(20);
                v.add(30);
                v.add(40);
                Iterator i = v.iterator();
                System.out.println("Before Adding "+v);
                while(i.hasNext()) {
                        Object obj = i.next();
                        if((int)obj==30) {
                                v.add(50);
                        }
                }
                System.out.println("After adding  "+v);
        }
}
```

How CopyOnWriteArrayList works

---

1. **Snapshot mechanism :** when iterator is created it work on a snapshot (copy) of the underlying array at the time of iterator creation

2. **Structural modification:** operations like add(),remove() or set() create a new copy of the entire underlying array with the modification applied.

3. **Iterator behaviour :** Iterator operator on snapshot so they are unaffected by concurrent modification to the list during iteration.

5. **Foreach loop :** for each is an enhancement for loop introduced by java in JDK 1.5 version and it specially designed for fetch data from array or collection in forward directions only.

Important points related with for each

---

1. Fetch data only by using forward direction
2. Not need to manually
3. Not need to check condition direct fetch from 0 to n-1 index
4. By default increment by 1 internally
5. Cannot travel data using backward direction
6. Not need to use index for fetch data directly we get data from array or collection

Syntax: for(data type variable:array/collection)

```
{
  }
```

Example with source code

```
package org.techhub;
import java.util.Vector;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.*;
public class VectorSum {
	public static void main(String x[]) {
		ArrayList v=new  ArrayList();
		 v.add(10);
		 v.add(20);
		 v.add(30);
		 v.add(40);
		 for(Object obj:v) {
			System.out.println(obj);
		 }
	}
}
```

For each method from JDK 1.8 version of java

```
package org.techhub;

import java.util.Vector;

import java.util.concurrent.CopyOnWriteArrayList;

import java.util.*;

public class VectorSum {

	public static void main(String x[]) {

		ArrayList v=new  ArrayList();

		 v.add(10);

		 v.add(20);

		 v.add(30);

		 v.add(40);

		v.forEach((val)->System.out.println(val));

	}

}
```

ArrayList

ArrayList is a internally dynamic array of Object class and it is implementer class of List collection and it is asynchronous collection

Means not thread safe means multiple thread object can use ArrayList simultaneously

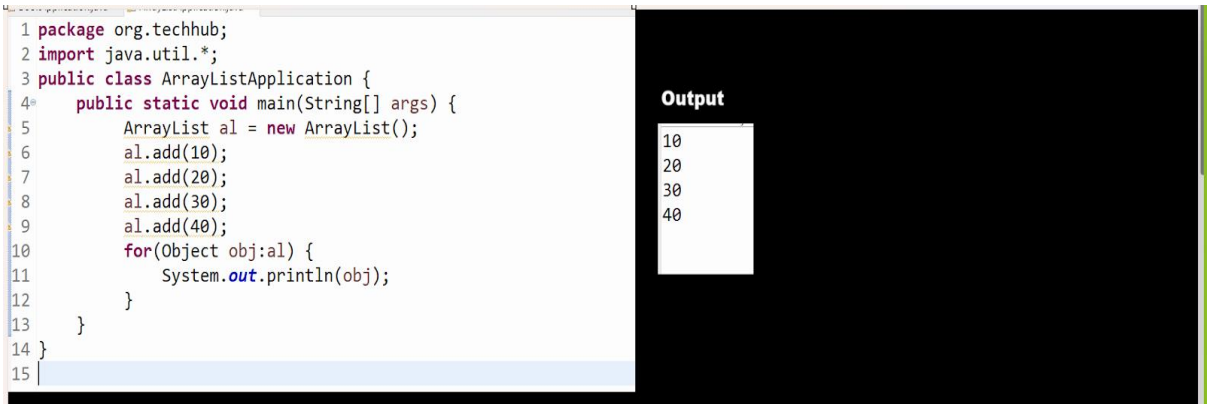
Constructor of ArrayList

ArrayList(): this constructor help us to create ArrayList with default size i.e 10

ArrayList(int initialCapacity): this constructor can create ArrayList object with initial capacity provided by user

ArrayList(Collection): This constructor can be used to copy data from another collection and store it in the ArrayList Collection.

Example using ArrayList



Q. What is the difference between ArrayList and Vector?

---

1. Vector is legacy collection and ArrayList is not legacy collection
2. Vector is thread safe or synchronized and ArrayList is not thread safe or non synchronized collection

Means performance wise ArrayList is faster than Vector but not thread safe

3. Vector allocates double memory than its current capacity when capacity crosses and ArrayList occupies half memory than its current capacity when capacity crosses.

Logic of ArrayList capacity increment

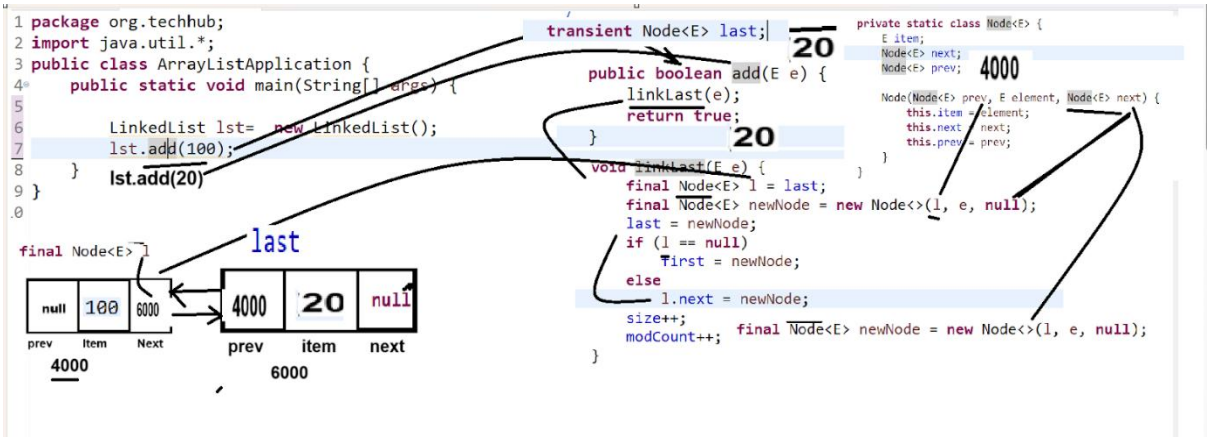
```
int newCapacity = size > currentCapacity ? currentCapacity+currentCapacity>>1 : currentCapacity;
                                          10+ 10>>1
                                          10+ 5
```

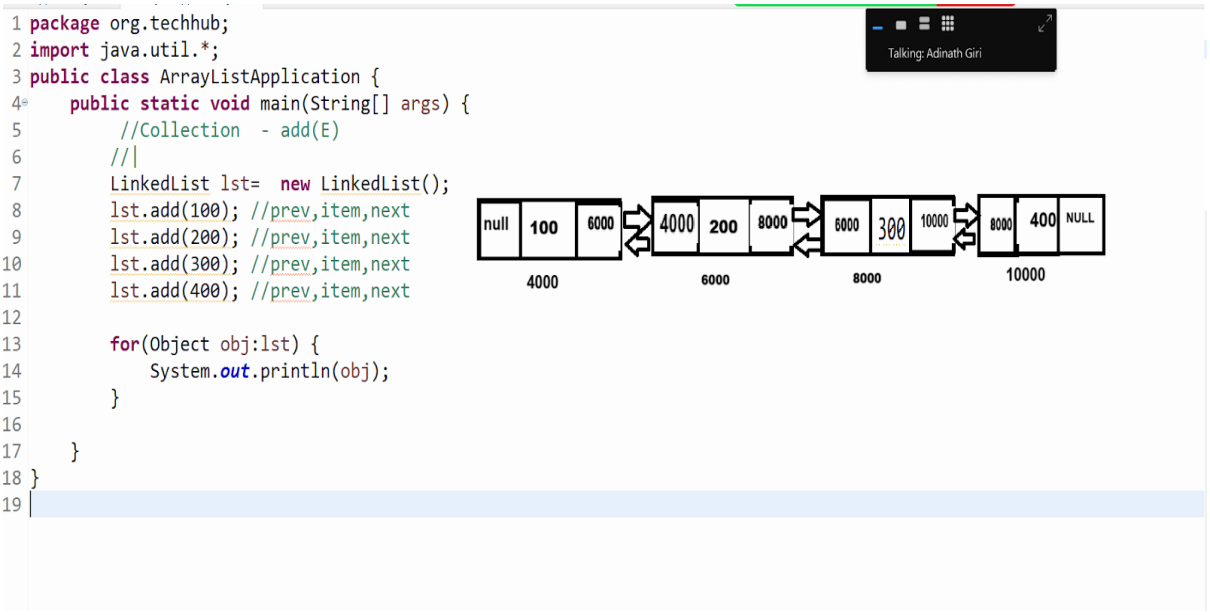
4. Threshold of Vector is 1.0 or 100% and Threshold of ArrayList is 0.5

LinkedList

If we think about LinkedList it is internally implementation of doubly LinkedList and doubly linked list is combination of node with three field

Prev,next and item and prev is reference of Node which is help us to hold address of previous node and next is a reference of Node which help us to store address of Next node and item is Object data type variable which represent internally by E generic notation which help us to store any kind of data or object in LinkedList





Q. What is the difference between ArrayList and LinkedList?

---

1. **Data Structure implementation differences**

ArrayList: ArrayList is internally dynamic array means ArrayList implemented by using array but resizable nature

Means when we add element in ArrayList it is add on specified location using index

LinkedList: LinkedList implemented by using doubly linked list internally means when we add data in linked list it is added as node with three field prev,item and next

2. Performance differences :

ArrayList and LinkedList has some permanence difference with operation like as fetching data or insert or deletion data as well as memory usage

Accessing or Fetching data

ArrayList : ArrayList provide constant time complexity (1) for data fetching by using indexing

LinkedList: Access Time is Linear O(n) because it need to travel list from beginning to end means we need to search address of next node for fetch data means LinkedList fetch data of current node and move to next node by address so we required to travel linked from beginning to ending

Insertion and deletion Operation

ArrayList: Insertion and deletion operation ArrayList required indexing shifting so need travel ArrayList for shift indexing for insertion and deletion so it may make some time so insertion and deletion operation slower by ArrayList than LinkedList means ArrayList O(n) time complexity for insertion and deletion

LinkedList : insertion and deletion operation perform faster by LinkedList because LinkedList only shift or manipulate address of node prev and next pointer without traveling so it is faster than ArrayList means LinkedList use O(1) for insertion and deletion

Memory usage:

ArrayList: ArrayList required less memory than LinkedList because ArrayList contain only data not address of another node they connected each other by using sequential address technique so easy for access via index

LinkedList: LinkedList required more memory than ArrayList because LinkedList contain three elements prev,next and node

3. Uses cases:

ArrayList: when we have list and we want to frequently search or fetching operation on collection then ArrayList is recommended but when have list and we want to perform continuous insertion or deletion then ArrayList is not recommended

LinkedList: When we required collection with frequently insertion or deletion then LinkedList is recommend and for fetching and searching LinkedList is not recommended

4. Interface Implementation Differences

ArrayList: ArrayList only implement List interface so it act as only list collection

LinkedList: LinkedList implements List interface as well as can implement DQueue interface means we can say we can use LinkedList as List as well as Queue

Q. What are the similarities between ArrayList and LinkedList?

1. **Interface implementation**: If we think About ArrayList and LinkedList both implements List interface so both classes contain common methods

2. Ordered collection : both classes provide data accessing as per the user data sequence means data fetching using ArrayList and LinkedList approach is same

Means both provide user sequence access

3. Duplicate elements : Both are implementer classes of List Collection and List collection allow duplicate elements so ArrayList and LinkedList also allow duplicate elements.

4. Allow Null Values: ArrayList and LinkedList both classes can hold null values.

5. Resizable : ArrayList is internally array but it is dynamic array so we can resize it as well as LinkedList is dynamic data structure so it is also resizable

6. Support iterators : ArrayList and LinkedList support Iterators for data fetching

7. **ThreadSafety :** both classes contain asynchronous method so they are not thread safe

Stack: Stack is last in first out data structure means first insert data remove last and last insert data remove first and stack has a single pointer known as top and when we insert

element in stack then top increases by 1 and when remove data from stack then top decreases by 1. Stack is a child of Vector

Methods of Stack collection

_____

E push(E): this method is used for push data or insert data in stack

E pop(): this method can remove data from stack and remove top most data from stack.

E peek(): this method can peek last index data means just we can view last index data but not remove using peek

boolean isEmpty(): this method can check elements present in collection or if present return true otherwise return false.

Note: when we fetch stack using iterator then it look like as Array Not stack so we required to fetch using ListIterator

Example: we want to perform following operation on Stack

Case 1: push

Case 2: pop

Case 3: display

Example with source code

```
package org.techhub;

import java.util.*;


public class ArrayListApplication {
	public static void main(String[] args) {
		Stack s = new Stack();
		do {
			Scanner xyz = new Scanner(System.in);
			System.out.println("1:INSERT");
			System.out.println("2:POP");
```

```
System.out.println("3:DISPLAY");
System.out.println("Enter your choice");
int choice = xyz.nextInt();
switch (choice) {
case 1:
        System.out.println("Enter value in stack");
        int value = xyz.nextInt();
        s.push(value);
        break;
case 2:
        boolean b = s.isEmpty();
        if (b) {
        System.out.println("There is no data in stack");
        } else {
System.out.println("Removed data is  " + s.pop());
        }
        break;
case 3:
        ListIterator li =s.listIterator(s.size());
        while(li.hasPrevious()) {
                Object obj  = li.previous();
                System.out.println(obj);
        }
        break;
case 4:
        System.exit(0);
        break;
default:
        System.out.println("Wrong choice");
```

```
                }
            } while (true);// infinite loop


        }
}
```

Example: we want to develop the application maintaining a question bank and every question contains the following data i.e qid , name , op1, op2, op3, op4 answer and we want to perform the following operation to perform the above task.

Case 1: Add New Question in Collection

Case 2: View All Question

Case 3: Search question by using question id

Case 4: remove question using question id

```
package org.techhub;
import java.util.*;
public class QuestionBank {
        public static void main(String[] args) {
                ArrayList al = new ArrayList();
                do {
                        Scanner xyz = new Scanner(System.in);
                        System.out.println("1:Add New Question");
                        System.out.println("2:View All Questions");
                        System.out.println("3:Search Question ");
                        System.out.println("4:Delete Question By Id");
                        System.out.println("Enter your choice");
                        int choice = xyz.nextInt();
                        switch (choice) {
                        case 1:
```

```
System.out.println("Enter qid name and all options and answer");
                int qid = xyz.nextInt();
                xyz.nextLine();
                String question = xyz.nextLine();
                String op1 = xyz.nextLine();
                String op2 = xyz.nextLine();
                String op3 = xyz.nextLine();
                String op4 = xyz.nextLine();
                String ans = xyz.nextLine();
Question q = new Question(qid, question, op1, op2, op3, op4, ans);
                al.add(q);
                break;
            case 2:
                Iterator i = al.iterator();
                while (i.hasNext()) {
                    Object obj = i.next();
                    Question ques = (Question) obj;
System.out.println(ques.getQid() + "\t" + ques.getName() + "\t" + ques.getOp1() + "\t"
                                + ques.getOp2() + "\t" + ques.getOp3() +
"\t" + ques.getOp4() + "\t" + ques.getAnser());
                }
                break;
            case 3:
                System.out.println("Enter question id for search question");
                int questionId = xyz.nextInt();
                i = al.iterator();
                boolean flag = false;
                while (i.hasNext()) {
                    Object obj = i.next();
                    Question ques = (Question) obj;
```

```
                        if (ques.getQid() == questionId) {
                                flag = true;
                                break;
                        }
                }
                if (flag) {
                        System.out.println("Question found in database or
collection");
                } else {
                System.out.println("Question not found");
                }
                break;
                case 4:
        System.out.println("Enter question id for search question");
                questionId = xyz.nextInt();
                i = al.iterator();
                flag = false;
                while (i.hasNext()) {
                        Object obj = i.next();
                        Question ques = (Question) obj;
                        if (ques.getQid() == questionId) {
                                int index = al.indexOf(ques);
                                if (index != -1) {
                                        al.remove(index);
                                        flag = true;
                                        break;
                                }
                        }
                }
                if (flag) {
```

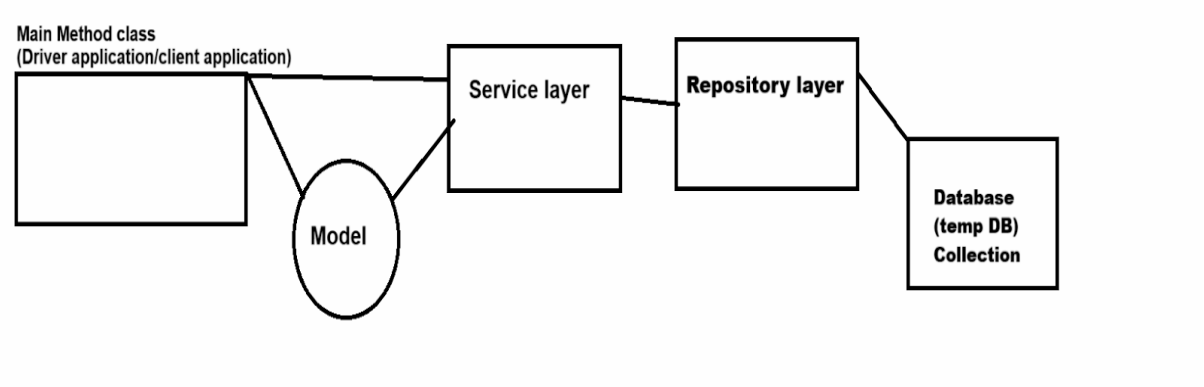
```
            System.out.println("Question removed from collection");
                                    } else {
                                            System.out.println("Question not found");
                                    }
                                    break;
                            default:
                                    System.out.println("Wrong choice");
                            }
                    } while (true);
            }
}
```

Store Management Application using Collection

1. Add Products in collection
2. View All products in collection
3. Search Product from collection
4. Delete Product from collection
5. Count total number of products from collection
6. Create Customer collection
7. Maintain purchase order of customer
8. View individual custom orders
9. View all customer reports
10. Delete customer and its order
11. Update customer orders

Etc

Coding layer



What is a client application?

Client application or driver class is a class which contain main methods where user can provide input and get results

And we required to object of service layer and model class in client application and using model class we can pass data from client application to service layer and service layer to repository

What is model class?

Model class is a class with setter and getter methods i.e POJO class which is used for store data or store different types of data

Means here model class work as container means it help us to store data and pass in different layer in application

What is the service layer?

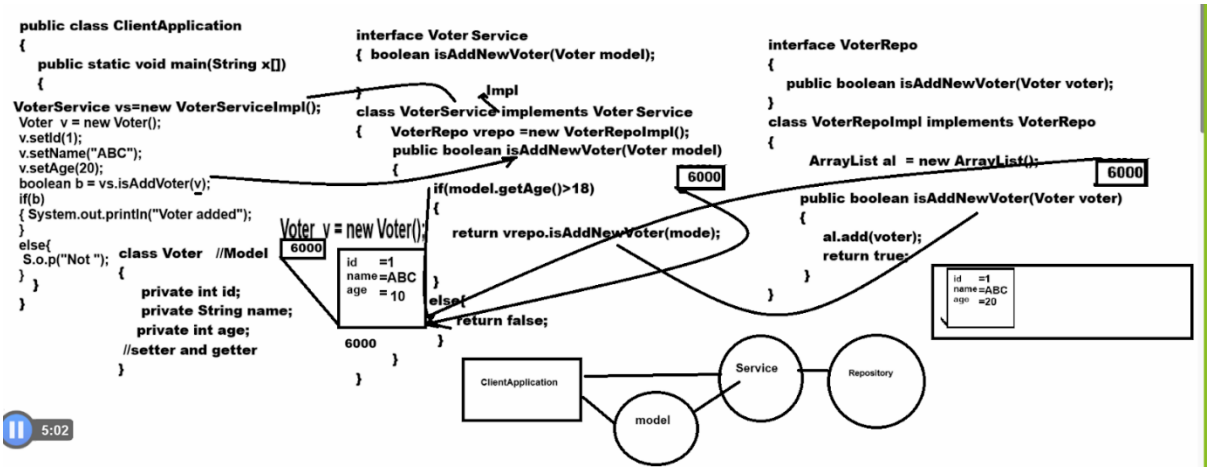
Service layer class which is used for write business logics in application

Note: business logic is not fix it is vary from requirement to requirement

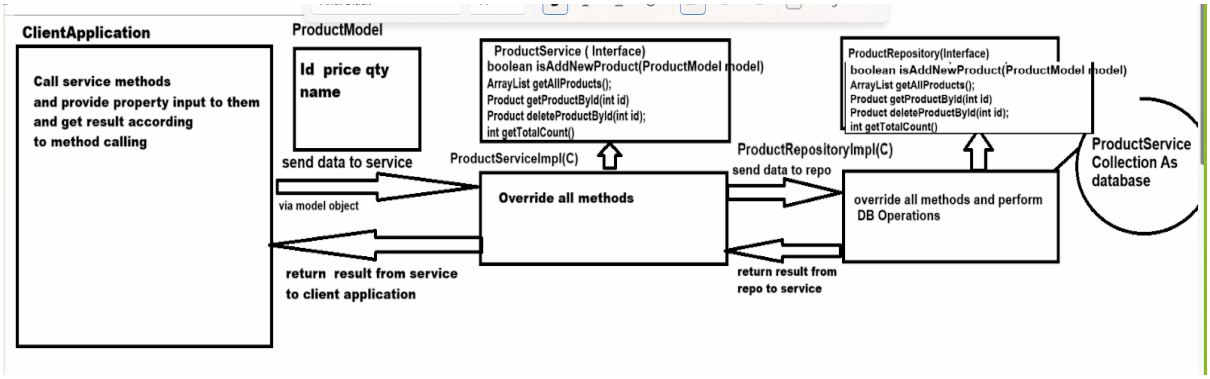
What is the repository layer?

Repository layer means class which contain database logics

Flow: we need to create service layer object in client application , model class object in class or catch model class reference return by service in client application , we need to create object of repository class in service layer means service method call from main method class i.e using client application and repository methods call from service classes



Product Model



public boolean isAddNewProduct(ProductModel): this method can accept product model class and store in database i.e in collection

Logic need to implement we add new products

---

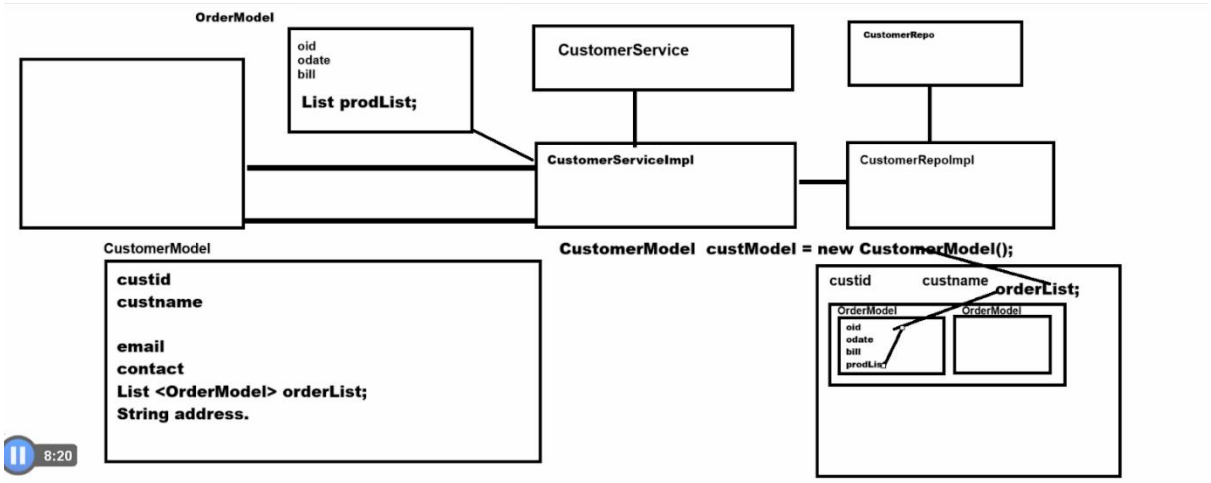
1. Product id should not duplicate
2. Product name should not be blank
3. Quantity must be know means quantity should not zero and price also should not zero

public ArrayList getAllProducts(): this method can return all products from database but if product not found in database then system should generate user define exceptions product not found

public Product getProductById(int id) : this method accept id from keyboard and return product data if id not found then generate user exception name as product not found exception

public Product deleteProductById(int id) : this method can accepted from keyboard and return product data and if id not found then return product not found exception

int getTotalCount(): this method can return total count of products

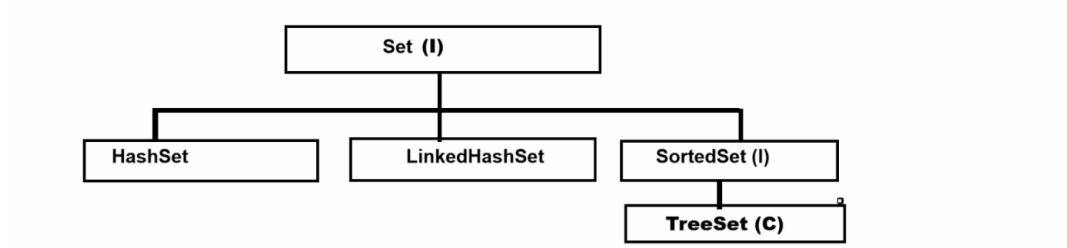


Note: remaining task should be complete by students

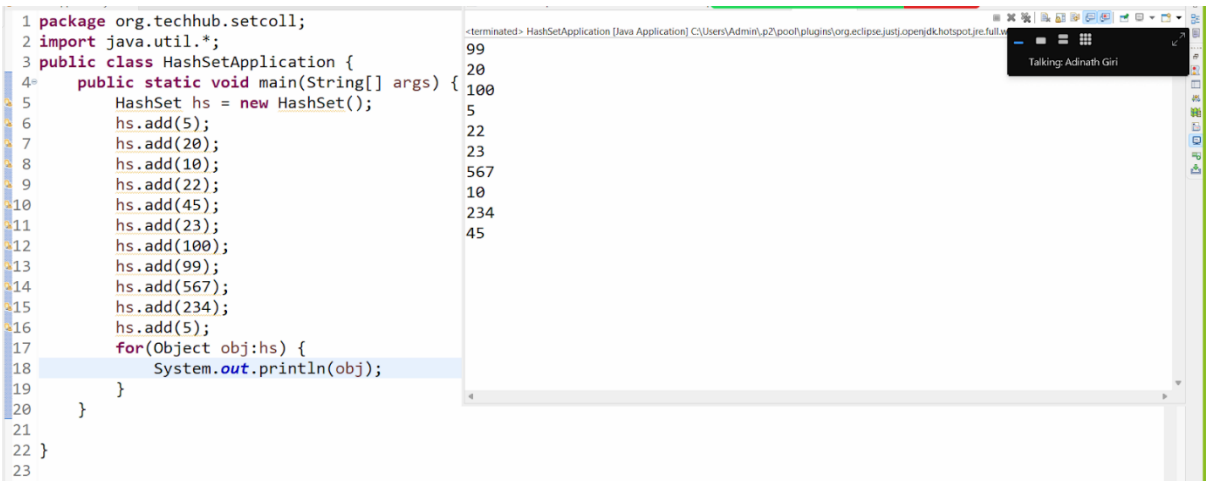
Set Collection

Set Collection does not allow duplicate elements and internally set collection internally works with data structure name as hash table .

If we think about set collection we have the following implementer classes.



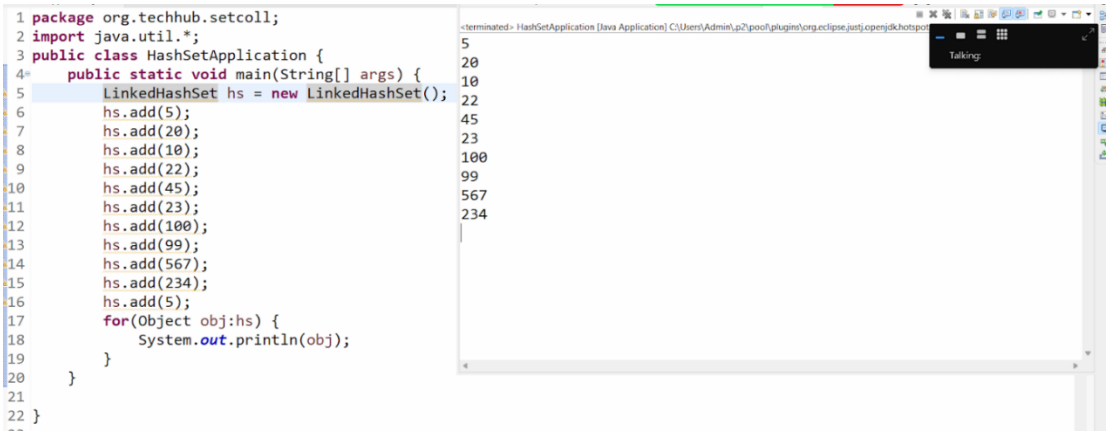
HashSet : HashSet can store unique element but generate random data means there is no sequence for data generation



```
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        HashSet hs = new HashSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        for(Object obj:hs) {
            System.out.println(obj);
        }
    }

}
```

```
99
20
100
5
22
23
567
10
234
45
```

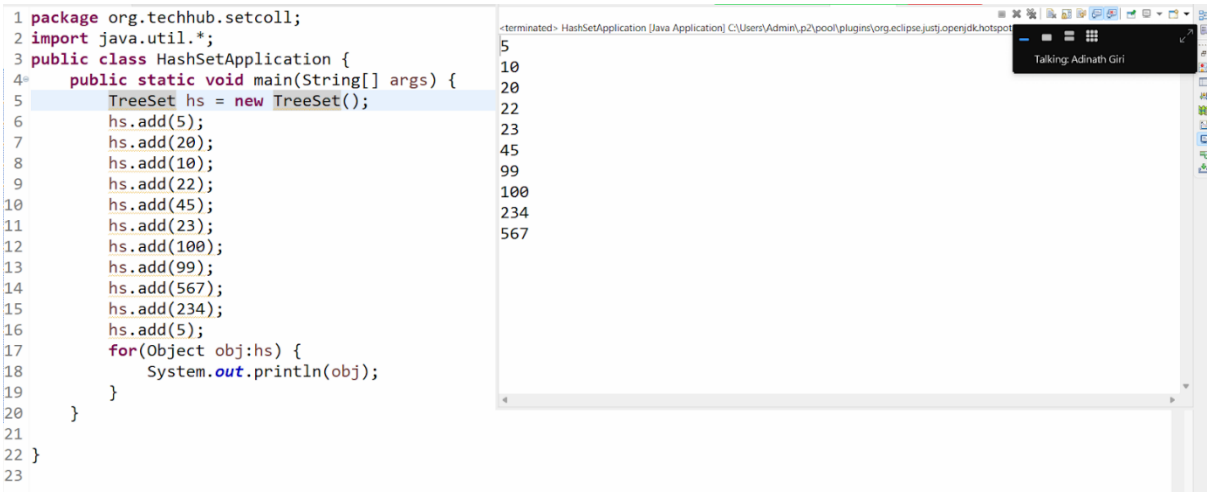
LinkedHashSet : LinkedHashSet can store unique element but store data sequence according to user sequence



```
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        LinkedHashSet hs = new LinkedHashSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        for(Object obj:hs) {
            System.out.println(obj);
        }
    }

}
```

```
5
20
10
22
45
23
100
99
567
234
```

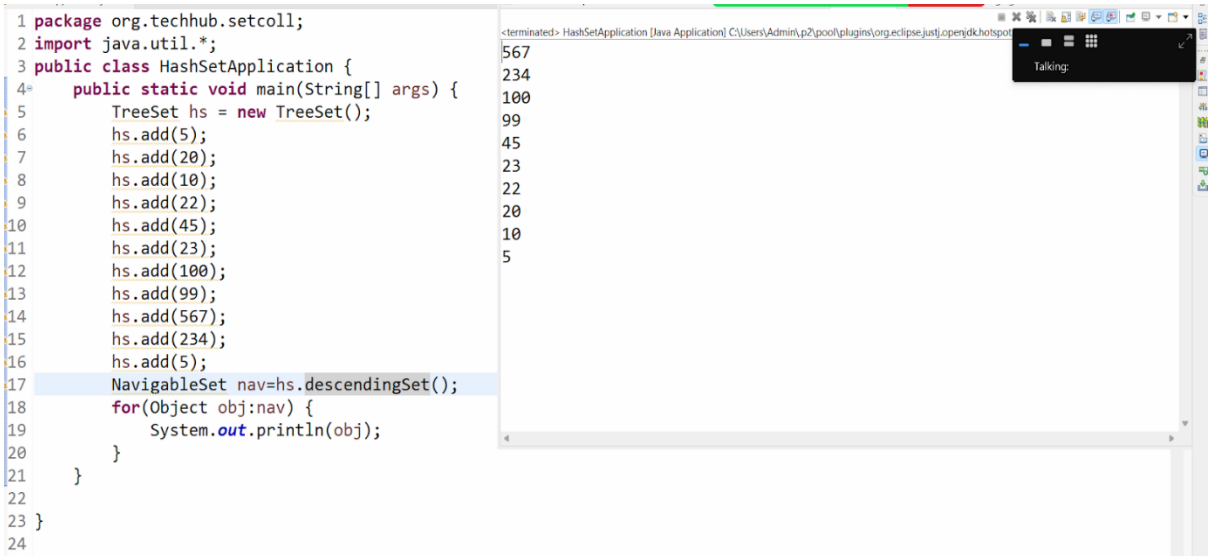
TreeSet : TreeSet Collection can store data in sorted format and arrange data in ascending order



```
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        TreeSet hs = new TreeSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        for(Object obj:hs) {
            System.out.println(obj);
        }
    }

}
```

```
5
10
20
22
23
45
99
100
234
567
```

Note: if we want to organize your data in descending order for TreeSet we have method name as descendingSet() and this method can generate objects of the NavigableSet interface and NavigableSet is the interface for fetching data in descending order.



```java
package org.techhub.setcoll;
import java.util.*;
public class HashSetApplication {
    public static void main(String[] args) {
        TreeSet hs = new TreeSet();
        hs.add(5);
        hs.add(20);
        hs.add(10);
        hs.add(22);
        hs.add(45);
        hs.add(23);
        hs.add(100);
        hs.add(99);
        hs.add(567);
        hs.add(234);
        hs.add(5);
        NavigableSet nav=hs.descendingSet();
        for(Object obj:nav) {
            System.out.println(obj);
        }
    }

}

```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
567
234
100
99
45
23
22
20
10
5
```

Talking:

HashSet class in depth working

HashSet works internally using Hashing technique

1. **Hashing :** In Hashing technique or hashing process

There are two things involved

a. Hash function : each element inserted into hashset is pass through the hash function which is calculate or computer unique integer called as hashcode and using hashcode decide element should place in hashtable or not

b. Hash code : the hashcode represents the element as an integer value which is then mapped with to an index in the internally array of the hashset i.e bucket

Q. What is a bucket?

---

a. Bucket is part of hashtable use by the hashset consists of an array of buckets each bucket can have one or more elements

b. The hashcode of the element is used to determine which bucket the element will go into.

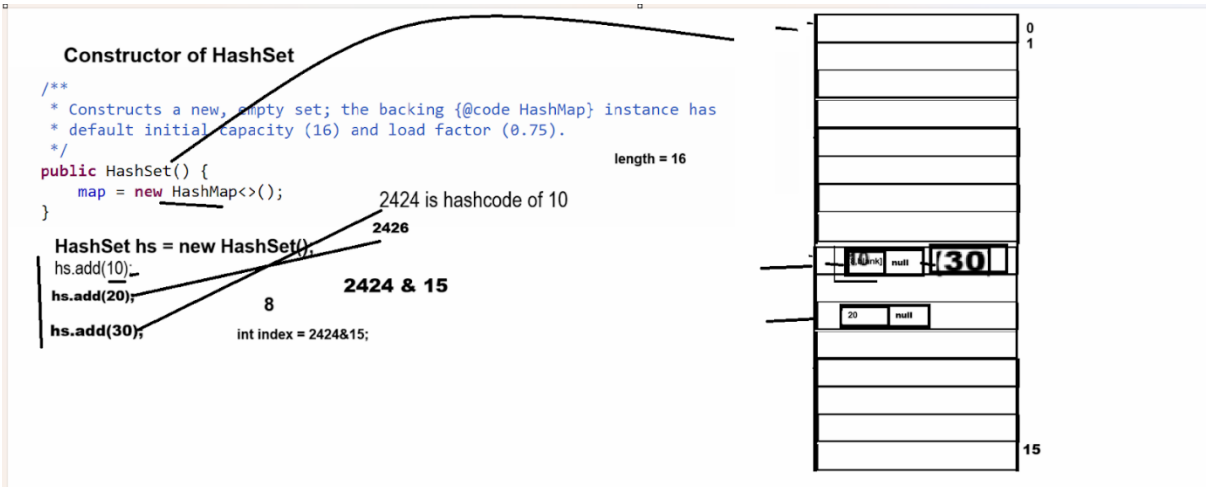
c. There is possibility if two element has same hashcode so their index may be same then there is possibility of data overriding for avoid this problem hashtable use one technique name as collision (where bucket holds linked list of element) or open addressing (where we try to store data in other bucket until bucket empty one found)

Constructor of HashSet

HashSet(): if we use default constructor of HashSet then internally this constructor use HashMap data structure and create one array of bucket with capacity 16 with load factor 0.75

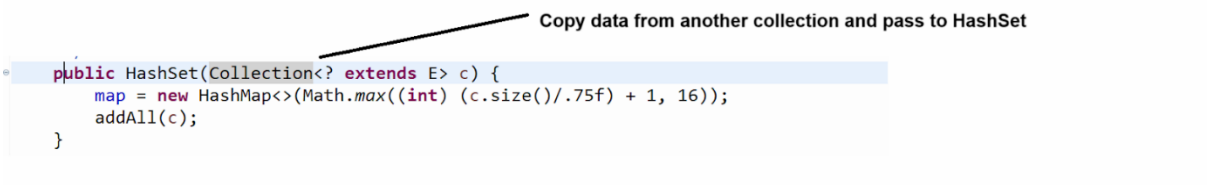
Q. What is the load factor?

Load factor is threshold which decide HashSet can increase capacity or not means here HashSet use default load factor 0.75 means up to 12 element HashSet capacity remain 16 i.e 0.75 but when try to insert 13th element i.e we try cross load factor or threshold set by hashset then hashset allocate memory double than its current capacity i.e 32



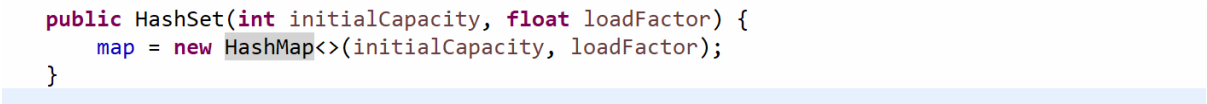
Constructor of HashSet

Syntax: HashSet(Collection): copy data from another collection and store in HashSet collection



Example: WAP to create ArrayList and remove duplicate element from ArrayList

public HashSet(int initialCapacity,float loadFactor): this constructor help us to set the user defined initial capacity and load factor



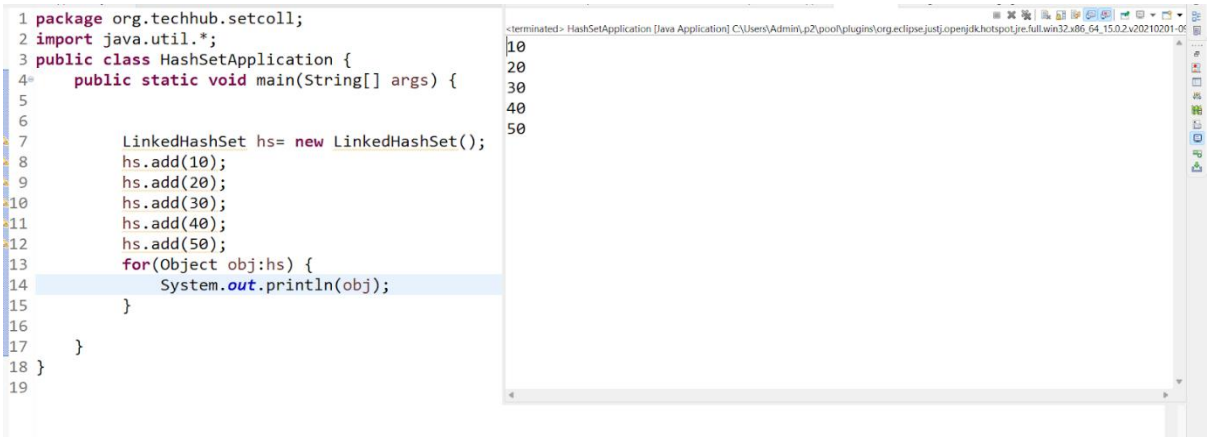
LinkedHashSet: LinkedHashSet is the child of HashSet and internally LinkedHashSet uses LinkedHashMap as data structure.

Constructor of LinkedHashSet

LinkedHashSet(): create internally bucket with default size 16 with load factor 0.75

LinkedHashSet(int initialCapacity,float loadFactor): create LinkedHashSet with initial capacity with default load factor

LinkedHashSet(Collection): this constructor help us copy data from another collection and use as parameter in LinkedHashSet



```
1 package org.techhub.setcoll;
2 import java.util.*;
3 public class HashSetApplication {
4     public static void main(String[] args) {
5
6
7         LinkedHashSet hs= new LinkedHashSet();
8         hs.add(10);
9         hs.add(20);
10        hs.add(30);
11        hs.add(40);
12        hs.add(50);
13        for(Object obj:hs) {
14            System.out.println(obj);
15        }
16
17    }
18 }
19
```

```
<terminated> HashSetApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0
10
20
30
40
50
```

TreeSet Collection

_____

TreeSet() : internally use TreeMap Data Structure

TreeSet(Comparator): this constructor help us to sort the data when user pass user defined object

TreeSet(Collection): this constructor help us to accept data from another collection and sort it

Etc

Collections class

_____

Collections is a utility class of JAVA which is used for perform regular data structure operation on Collection framework

Like as finding max element from collection ,finding element from collection, reverse the collection, sort the list collection, convert asynchronized collection to synchronized collection etc

Example: WAP to create ArrayList and perform operation on in it

Case 1: find the max value from array

Case 2: find the min value from array

Case 3: reverse the arraylist

Case 4: sort the ArrayList

Etc

```
package org.techhub.collectionsapp;
import java.util.*;
public class TestCollectionsApp {
	public static void main(String[] args) {
		ArrayList al = new ArrayList();
		al.add(5);
		al.add(10);
		al.add(100);
		al.add(20);
		al.add(50);
		al.add(90);
		al.add(40);

		Object maxVal = Collections.max(al);
		System.out.println("Max value from ArrayList is "+maxVal);
		Object minVal = Collections.min(al);
		System.out.println("Min value from ArrayList is "+minVal);
		Collections.sort(al);
		System.out.println(al);
		Collections.reverse(al);
		System.out.println(al);
	}
}
```

Example: WAP to create class name as Employee with field id, name and salary and store five employee objects in ArrayList and sort it

```
package org.techhub.collectionsapp;
import java.util.*;
class Employee{
	private int id;
	private String name;
		private int sal;
		public Employee() {
		}
		public Employee(String name,int id,int sal) {
			this.name=name;
			this.id=id;
			this.sal=sal;
		}
		public int getId() {
			return id;
		}

		public void setId(int id) {
			this.id = id;
		}
		public String getName() {
			return name;
		}
		public void setName(String name) {
			this.name = name;
		}
		public int getSal() {
```

```
                return sal;
        }
        public void setSal(int sal) {
                this.sal = sal;
        }
}
public class TestCollectionsApp {
        public static void main(String[] args) {
                ArrayList al = new ArrayList();
                Employee e1 = new Employee("ABC",3,10000);
                Employee e2 = new Employee("PQR",4,20000);
                Employee e3 = new Employee("STV",1,30000);
                Employee e4 = new Employee("XYZ",2,5000);
                Employee e5 = new Employee("SSSS",5,9000);
                al.add(e1);
                al.add(e2);
                al.add(e3);
                al.add(e4);
                al.add(e5);

                System.out.println("Display before sorting");
                for(Object obj:al) {
                        Employee e=(Employee)obj;
                        System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
                }
                Collections.sort(al);
                System.out.println("Display after sorting");
                for(Object obj:al) {
                        Employee e=(Employee)obj;
```

```
                System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
            }
        }
}
```

Output

Note: if we think about above output we have exception ClassCastExcetpion because we have collection with user defined object i.e Employee objects and which contain three types id,name and salary and we store all objects in ArrayList collection and we have Statement Collections.sort(al) here Collections.sort() method get confused which which field employee should sort means using id or name or salary so we get exception means Collections.sort() method by default sort data of collection when collection contain primitive type of data but when Collections contain user defined objects then Collections.sort() method cannot sort data and generate exception at run time so if we want to solve this problem we have two solutions

1. Comparable interface
2. Comparator interface

Q. What is a Comparable interface?

Comparable interface is a member of java.lang package and which is used for perform sorting with user defined objects using List collection or Set collection

Steps to work with Comparable interface

---

1. **Add java.lang package in application**

Note: we do not need to import the java.lang package because it is the default package of java.

2. **Create POJO class and implements Comparable interface in it**

```
class Employee implements Comparable{
	private int id;
	private String name;
		private int sal;
		public Employee() {
		}
		public Employee(String name,int id,int sal) {
			this.name=name;
			this.id=id;
			this.sal=sal;
		}
		public int getId() {
			return id;
		}

		public void setId(int id) {
			this.id = id;
		}
		public String getName() {
			return name;
		}
		public void setName(String name) {
			this.name = name;
		}
		public int getSal() {
			return sal;
```

```
	}
	public void setSal(int sal) {
		this.sal = sal;
	}
}
```

3. Override its compareTo() method and perform object comparison

---

compareTo() method compare current with other elements and perform following things and return some resultant value

1. If current object value is greater than parameter object then return 1
2. If current object value is less than parameter object then return -1
3. If current object is equal with parameter object value return 0

Example with source code

```
package org.techhub.collectionsapp;
import java.util.*;
class Employee implements Comparable{
	private int id;
	private String name;
		private int sal;
		public Employee() {
		}
		public Employee(String name,int id,int sal) {
			this.name=name;
			this.id=id;
			this.sal=sal;
		}
		public int getId() {
			return id;
		}
```

```
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getSal() {
		return sal;
	}
	public void setSal(int sal) {
		this.sal = sal;
	}
	public int compareTo(Object o) {
		Employee emp=(Employee)o;
		if(this.id>emp.id) {
			return 1;
		}
		else if(this.id<emp.id) {
			return -1;
		}
		else {
			return 0;
		}
	}
}
```

```
public class TestCollectionsApp {
	public static void main(String[] args) {
		ArrayList al = new ArrayList();
		Employee e1 = new Employee("ABC",3,10000);
		Employee e2 = new Employee("PQR",4,20000);
		Employee e3 = new Employee("STV",1,30000);
		Employee e4 = new Employee("XYZ",2,5000);
		Employee e5 = new Employee("SSSS",5,9000);

		al.add(e1);
		al.add(e2);
		al.add(e3);
		al.add(e4);
		al.add(e5);

		System.out.println("Display before sorting");
		for(Object obj:al) {
			Employee e=(Employee)obj;
			System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
		}
		Collections.sort(al);
		System.out.println("Display after sorting");
		for(Object obj:al) {
			Employee e=(Employee)obj;
			System.out.println(e.getId()+"\t"+e.getName()+"\t"+e.getSal());
		}
	}
}
```

Example: WAP to create class name as Player with field id,name and run and store 5 player objects in ArrayList and sort player data by using run.

```
package org.techhub.collectionsapp;
import java.util.*;
class Player implements Comparable{
	private int id;
	private String name;
	public Player() {
	}
	public Player(String name,int id,int run) {
		this.name=name;
		this.id=id;
		this.run=run;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getRun() {
		return run;
	}
```

```
	public void setRun(int run) {
		this.run = run;
	}
	private int run;
	@Override
	public int compareTo(Object o) {
		Player p1=(Player)o;
		 return this.run>p1.run?1:this.run<p1.run ?-1:0;
	}
}
public class PlayerApplication {

	public static void main(String[] args) {
		ArrayList al = new ArrayList();

		Player p1 = new Player("ABC",1,90000);
		Player p2 = new Player("PQR",2,3000);
		Player p3 = new Player("STV",4,12000);
		Player p4 = new Player("XYZ",5,7000);
		Player p5 = new Player("SSS",3,190000);

		al.add(p1);
		al.add(p2);
		al.add(p3);
		al.add(p4);
		al.add(p5);

		System.out.println("Display player record before sorting");
		for(Object obj:al) {
```

```
                Player p=(Player)obj;
                System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
            }
            Collections.sort(al);

            System.out.println("Display player record After sorting");
            for(Object obj:al) {
                Player p=(Player)obj;
                System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
            }
        }

}
```

Comparator interface

Comparator interface is used for perform sorting with a user defined objects but using Comparator we can perform sorting using multiple field

Steps to work with Comparator interface

---

1. **Add java.util package**
2. **Create seperate class for sorting and implements Comparator in it & override compare() method in every implementer class**

Example: suppose consider we want to sort player data by using id,by using name or by using run so we required to declare three class for sorting

a. SortById implements Comparator

```
package org.techhub.collectionsapp;

import java.util.Comparator;
```

```
public class SortPlayerById implements Comparator {

	@Override
	public int compare(Object o1, Object o2) {
		Player p1 = (Player) o1;
		Player p2 = (Player) o2;

		if (p1.getId() > p2.getId()) {
			return 1;
		} else if (p1.getId() < p2.getId()) {
			return -1;
		} else {

			return 0;
		}
	}
}
```

b. SortByName implements Comparator

```
 package org.techhub.collectionsapp;
import java.util.Comparator;
public class SortPlayerByName implements Comparator {

	@Override
	public int compare(Object o1, Object o2) {
		// TODO Auto-generated method stub
		Player p1=(Player)o1;
		Player p2=(Player)o2;
		return p1.getName().compareTo(p2.getName());
	}
```

}

c. SortByRun implements Comparator

```
package org.techhub.collectionsapp;
import java.util.Comparator;
public class SortPlayerByRun implements Comparator {

	@Override
	public int compare(Object o1, Object o2) {
		// TODO Auto-generated method stub
		Player p1 = (Player) o1;
		Player p2 = (Player) o2;
		if (p1.getRun() > p2.getRun()) {
			return 1;
		} else if (p1.getRun() < p2.getRun()) {
			return -1;
		} else {
			return 0;
		}
	}
}
```

Rules of compare() method

---

a. If first object is greater than second object then return 1

b. If first object is less than second object then return -1

c. If the first object is equal with the second object then return 0.

4. **Use Following method version of Collections.sort()**

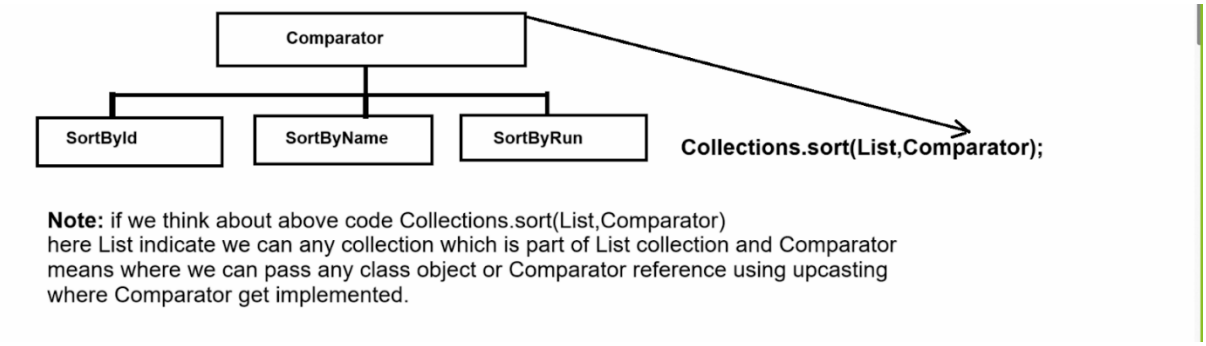
Note: when we use the Comparator we have to use Collection.sort()

Method using a following version

Note: Collections.sort() it is overloaded method of Collections class

Syntax1: Collections.sort(List): means sort primitive type of list collection or sort user defined object where Comparable get implements

Syntax: Collections.sort(List,Comparator): normally version use when we have Comparator or when we want to perform sorting with multiple field



Main class with Player POJO

```
package org.techhub.collectionsapp;
import java.util.*;
class Player{
	private int id;
	private String name;
	public Player() {
	}
	public Player(String name,int id,int run) {
		this.name=name;
		this.id=id;
		this.run=run;
```

```
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getRun() {
		return run;
	}
	public void setRun(int run) {
		this.run = run;
	}
	private int run;
}
public class PlayerApplication {

	public static void main(String[] args) {
		ArrayList ts = new ArrayList();
		Player p1 = new Player("XYZ",1,90000);
		Player p2 = new Player("PQR",2,3000);
		Player p3 = new Player("STV",4,12000);
		Player p4 = new Player("ABC",5,7000);
```

```
Player p5 = new Player("SSS",3,190000);
ts.add(p1);
ts.add(p2);
ts.add(p3);
ts.add(p4);
ts.add(p5);

System.out.println("Display Original Data");
for(Object obj:ts) {
	Player p=(Player)obj;
	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
}
System.out.println("Display player record sort by using id");
Comparator c=new SortPlayerById();
Collections.sort(ts,c);
for(Object obj:ts) {
	Player p=(Player)obj;
	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
}
System.out.println("Display player record sort by using name");
c=new SortPlayerByName();
Collections.sort(ts,c);
for(Object obj:ts) {
	Player p=(Player)obj;
	System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
}
System.out.println("Display player record sort by using run");
c=new SortPlayerByRun();
Collections.sort(ts,c);
```

```
		for(Object obj:ts) {
			Player p=(Player)obj;
			System.out.println(p.getId()+"\t"+p.getName()+"\t"+p.getRun());
		}
	}

}
```

How to convert asynchronous collection or map in to synchronous format

---

If we want to convert any asynchronous collection to synchronous collection we have some standard method provided by Collections class to us

List ref= Collections.synchronizedList(List): convert all asynchronous list implementer to synchronized object

Set ref=Collections.synchronizedSet(List)

Collection ref=Collections.synchronizedCollection(Collection)

Map ref=Collections.synchronizedMap(Map);

Etc

Example : we want to convert ArrayList object as synchronized object

```
package org.techhub.collectionsapp;
import java.util.*;
public class TestArrListApp {
	public static void main(String[] args) {
		List al = new ArrayList();
		al.add(100);
```

```
        al.add(200);
        al.add(300);
        List l=Collections.synchronizedList(al);
        System.out.println(l);
    }

}
```

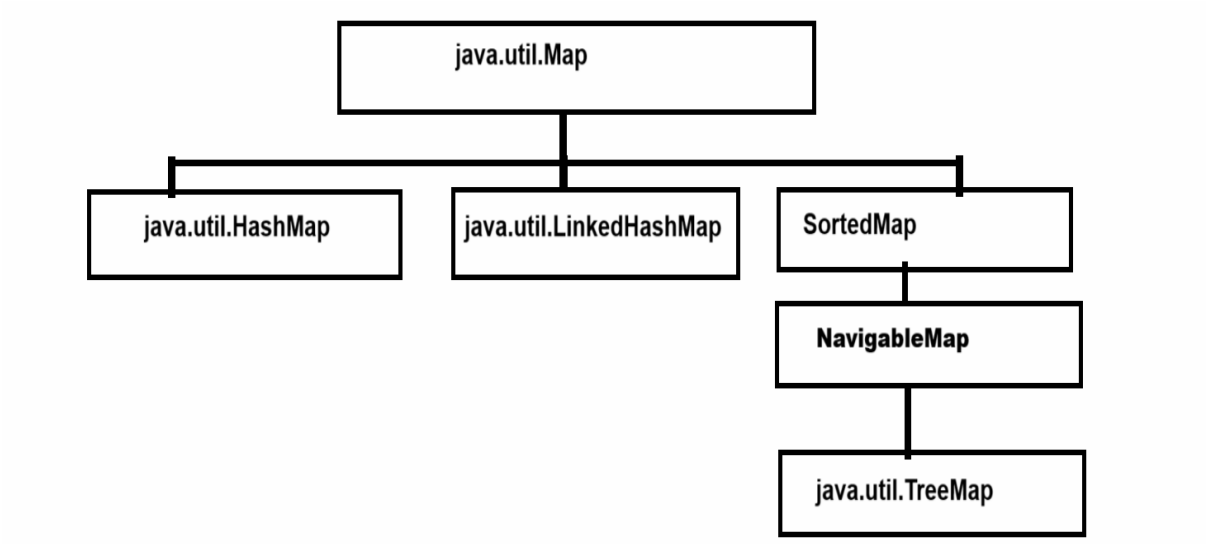
Map: Map is not part of collection but it is like as Collection and Map can store data in the form of key and value pair and key cannot be duplicated and value may be duplicated

Means we can say map is combination of set and list collection

Set work as key and list work as value

Note: Normally Map recommend when we have duplicated data but we want to maintain uniqueness of that data then map is recommended

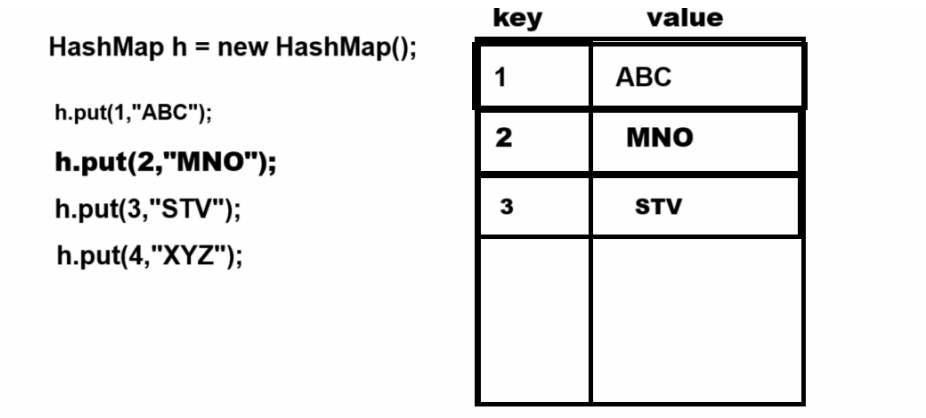
Hierarchy of Map



Methods of Map Collection

---

void put(Object key,Object value) : this method can store data in a map using key and value pairs.

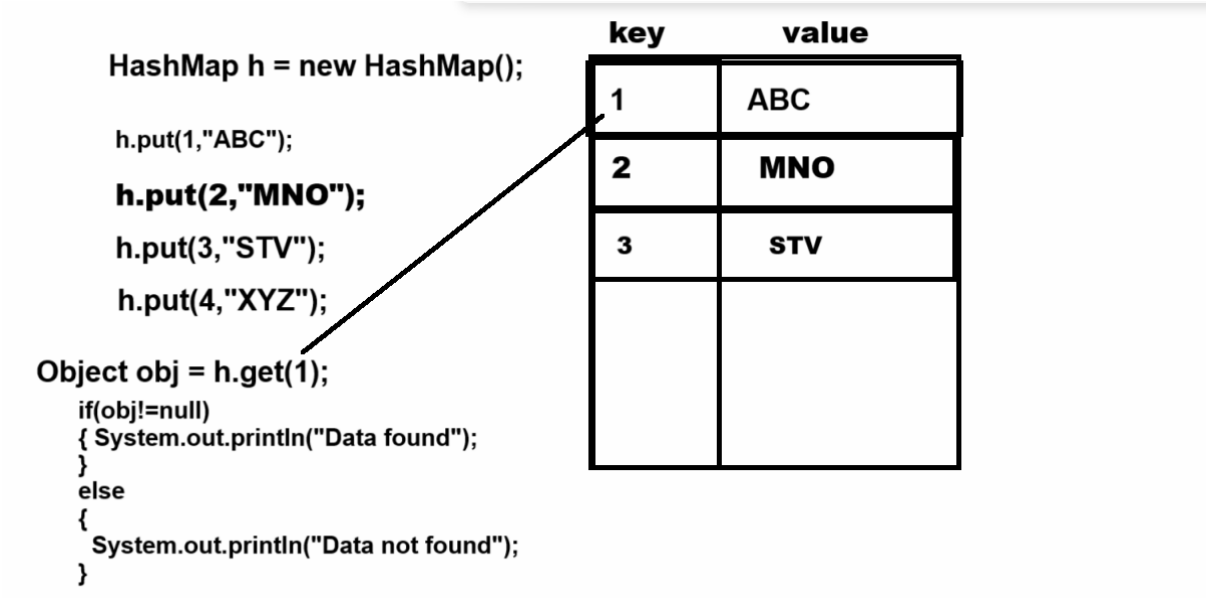


Object get(Object key) : this method can fetch from map using key and if key not found return null

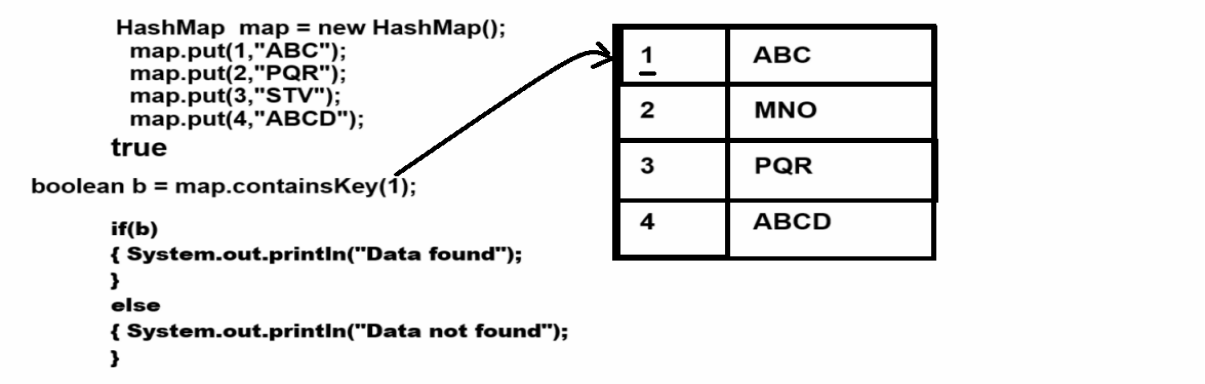
Normally we use this method in two cases

a. For fetching data

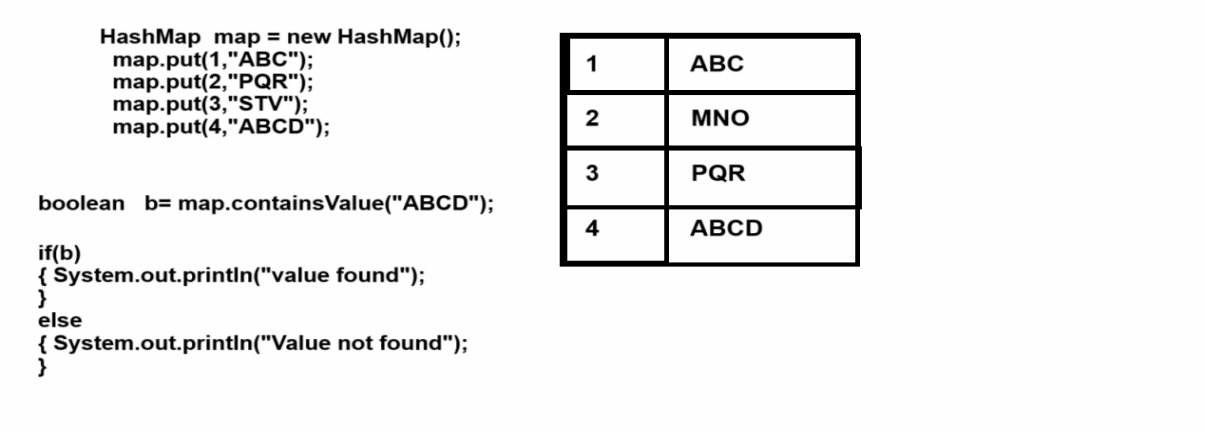
b. For search data : when the get() method returns a non null value means consider data is present and when get() method returns null then we consider data not present in the map.



boolean containsKey(Object key): this method helps us search data from map using key and if key found return true otherwise return false.

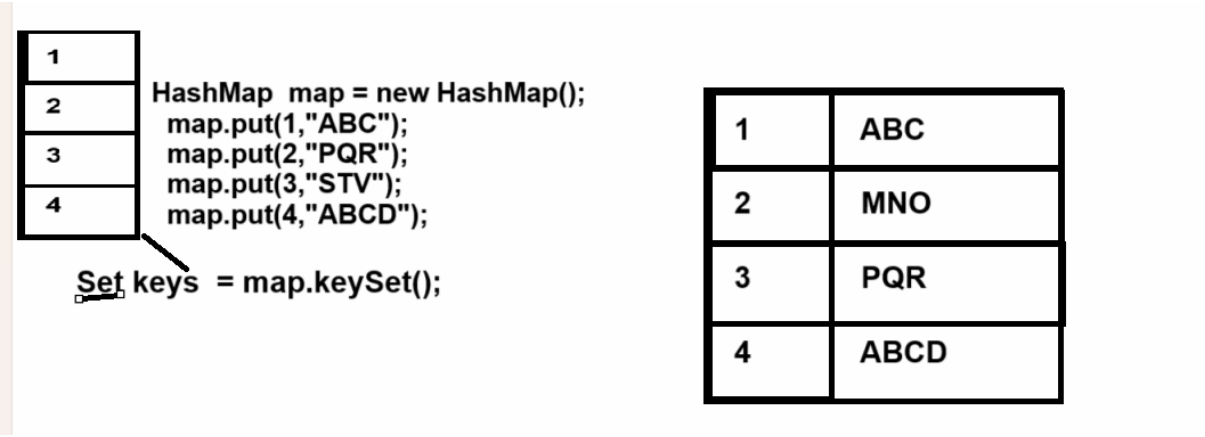


boolean containsValue(Object value) : this method help us to data present in map or not if present return true otherwise return false.

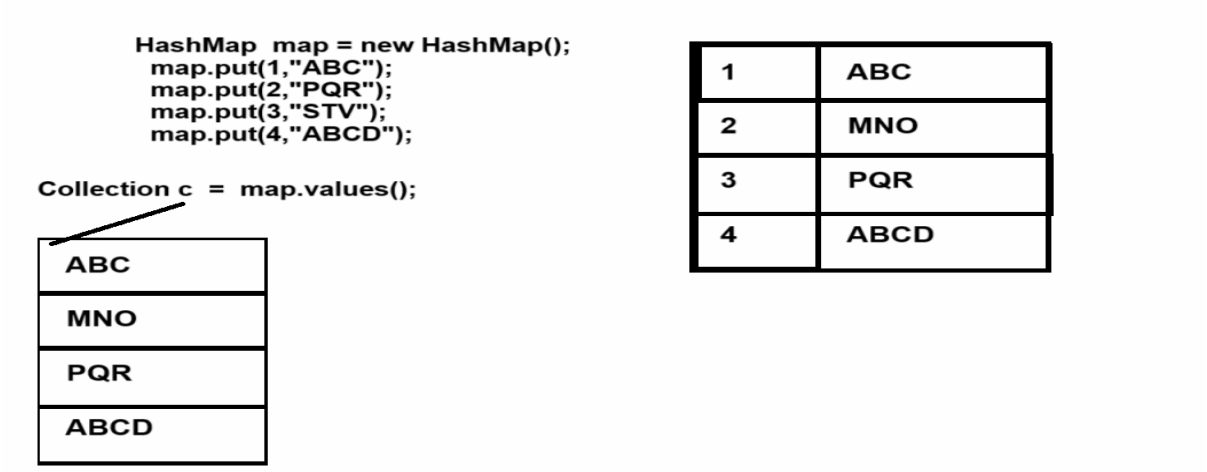


Object remove(Object key) : this method can remove data from map using its key and return remove value.

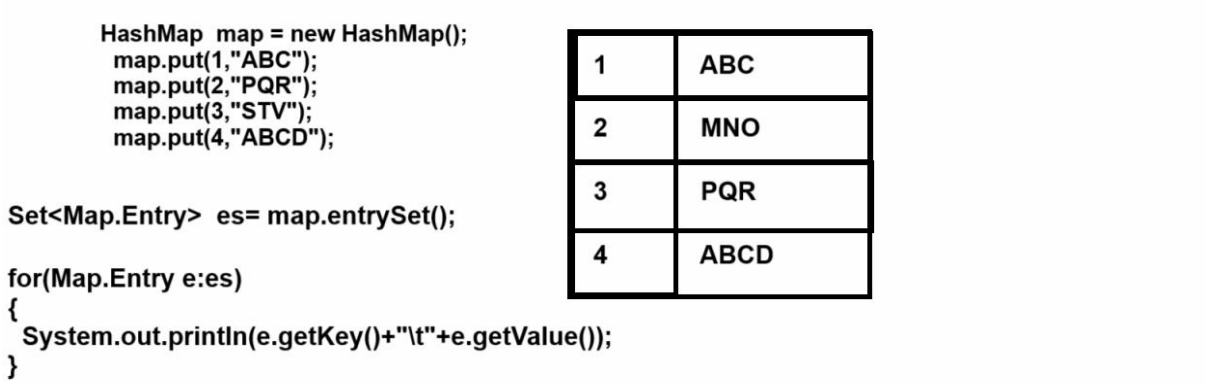
Set keySet() : this method can return all keys from map



Collection values() : this method can return all values from map as collection reference



Map.Entry entySet(): this method can return all data from map as Entry object means this method can return data in the form of key and value pair.

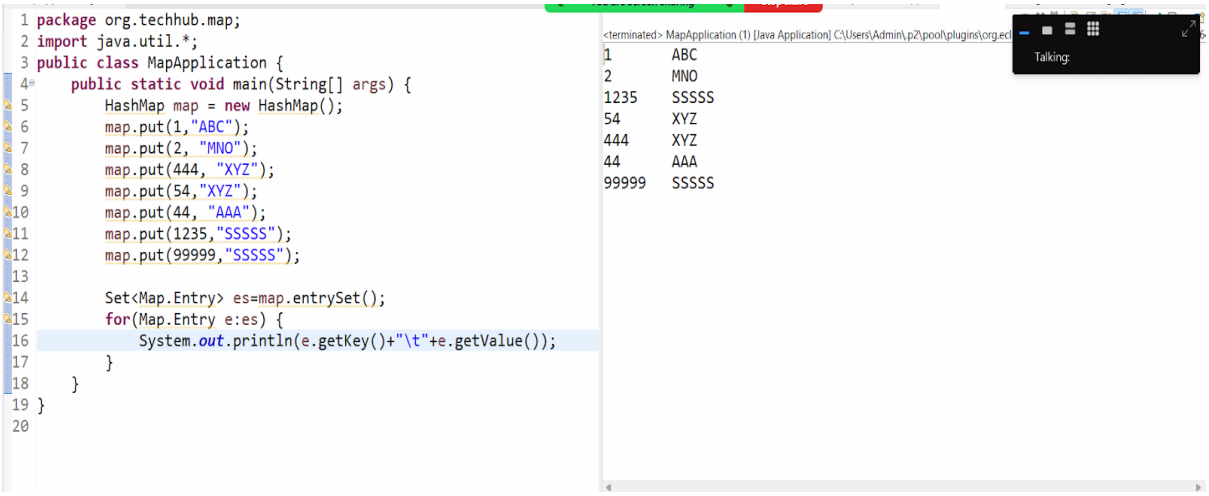


int size() : this method can return number of element present in map

boolean isEmpty() : this method can check data present in map or not if present return true otherwise return false.

HashMap

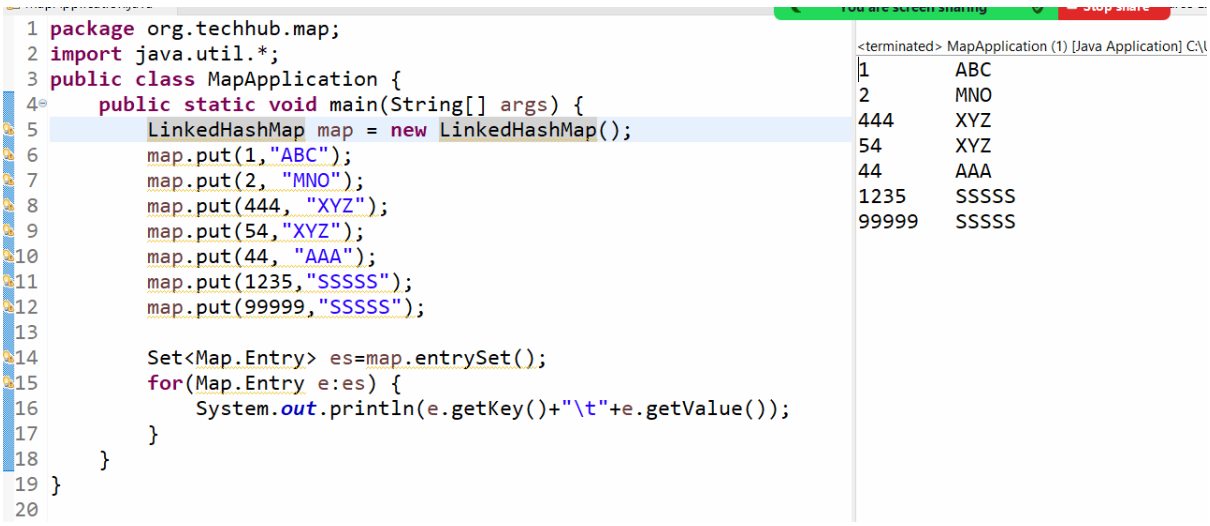
HashMap can generate random data or generate random keys



```
package org.techhub.map;
import java.util.*;
public class MapApplication {
    public static void main(String[] args) {
        HashMap map = new HashMap();
        map.put(1,"ABC");
        map.put(2, "MNO");
        map.put(444, "XYZ");
        map.put(54,"XYZ");
        map.put(44, "AAA");
        map.put(1235,"SSSSS");
        map.put(99999,"SSSSS");

        Set<Map.Entry> es=map.entrySet();
        for(Map.Entry e:es) {
            System.out.println(e.getKey()+"\t"+e.getValue());
        }
    }
}
```

```
<terminated> MapApplication (1) [Java Application] C:\Users\Admin\.p2\pool\plugins\org.ecl
1       ABC
2       MNO
1235    SSSSS
54      XYZ
444     XYZ
44      AAA
99999   SSSSS
```

LinkedHashMap: this map can arrange data in key and value pair but arrange all keys as per user sequence .



```
package org.techhub.map;
import java.util.*;
public class MapApplication {
    public static void main(String[] args) {
        LinkedHashMap map = new LinkedHashMap();
        map.put(1,"ABC");
        map.put(2, "MNO");
        map.put(444, "XYZ");
        map.put(54,"XYZ");
        map.put(44, "AAA");
        map.put(1235,"SSSSS");
        map.put(99999,"SSSSS");

        Set<Map.Entry> es=map.entrySet();
        for(Map.Entry e:es) {
            System.out.println(e.getKey()+"\t"+e.getValue());
        }
    }
}
```

```
<terminated> MapApplication (1) [Java Application] C:\(
1       ABC
2       MNO
444     XYZ
54      XYZ
44      AAA
1235    SSSSS
99999   SSSSS
```

TreeMap: TreeMap can store data in key and value pair but arrange all keys in ascending order.

```java
package org.techhub.map;
import java.util.*;
public class MapApplication {
    public static void main(String[] args) {
        TreeMap map = new TreeMap();
        map.put(1,"ABC");
        map.put(2, "MNO");
        map.put(444, "XYZ");
        map.put(54,"XYZ");
        map.put(44, "AAA");
        map.put(1235,"SSSSS");
        map.put(99999,"SSSSS");

        Set<Map.Entry> es=map.entrySet();
        for(Map.Entry e:es) {
            System.out.println(e.getKey()+"\t"+e.getValue());
        }
    }
}

```

```
<terminated> MapApplication (1) [Java
1       ABC
2       MNO
44      AAA
54      XYZ
444     XYZ
1235    SSSSS
99999   SSSSS
```

Example: we want to create a program to perform the following operation on map.

Case 1: store data in map

Case 2: View all data from map

Case 3: search data from map by using key

Case 4: delete data from map using key

Case 5: count total number of elements of map

Case 6: display only keys of map

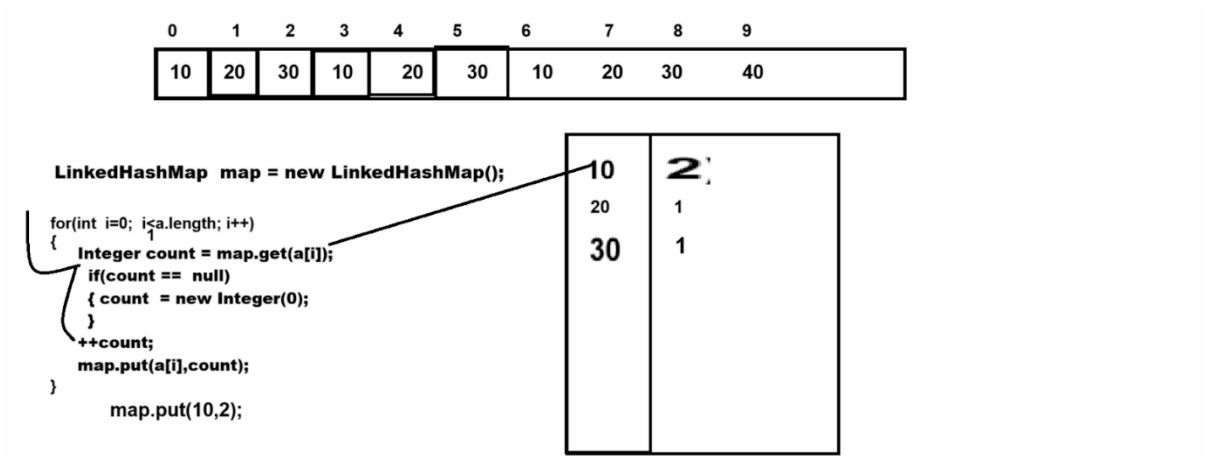
Example:

```
package org.techhub.map;
import java.util.*;
public class MapApplication {
	public static void main(String[] args) {
		LinkedHashMap map = new LinkedHashMap();
		do {
			Scanner xyz = new Scanner(System.in);
			System.out.println("1:Add New Data in Map");
			System.out.println("2:View All Data from map");
```

```
System.out.println("3: Search data from map using key");
System.out.println("4: Delete data from map using key");
System.out.println("5: Count total number of element ");
System.out.println("6: Display all keys");
System.out.println("Enter your choice");
int choice = xyz.nextInt();
switch (choice) {
case 1:
        xyz.nextLine();
System.out.println("Enter  name and id of student");
        String name = xyz.nextLine();
        int id = xyz.nextInt();
        map.put(id, name);
        break;
case 2:
        Set<Map.Entry> entrySet = map.entrySet();
        for (Map.Entry e : entrySet) {
System.out.println(e.getKey() + "\t" + e.getValue());
        }
        break;
case 3:
        System.out.println("Enter key for search");
        int key = xyz.nextInt();
        boolean b = map.containsKey(key);
        if (b) {
                System.out.println("Data found");
        } else {
                System.out.println("Data not found");
        }
```

```
break;
                        case 4:
                                System.out.println("Enter key for search");
                                key = xyz.nextInt();
                                b = map.containsKey(key);
                                if (b) {
                                        map.remove(key);
                                        System.out.println("Data found");
                                } else {
                                        System.out.println("Data not found");
                                }
                                break;
                        case 5:
        System.out.println("total element present in map " + map.size());
                                break;
                        case 6:
                                Set keys = map.keySet();
                                for (Object k : keys) {
                                        System.out.println(k);
                                }
                                break;
                        case 7:
                                System.exit(0);
                                break;
                        default:
                                System.out.println("Wrong choice");
                        }
                } while (true);
        }      }
```

Example: WAP to create array and store 10 values in it and find the occurrence of every element in array by using LinkedHashMap




```java
package org.techhub.map;
import java.util.*;
public class FindOccurenceApp {
    public static void main(String[] args) {
        Scanner xyz  = new Scanner(System.in);
        LinkedHashMap map = new LinkedHashMap();
        int a[]=new int[10];
        System.out.println("Enter values in array");
        for(int i=0; i<a.length;i++) {
            a[i]=xyz.nextInt();
        }
        for(int i=0;i<a.length;i++) {
            //Integer count=(Integer)map.get(a[i]);
            Object obj=map.get(a[i]);
            Integer count=(Integer)obj;
            if(count==null) {
                count = new Integer(0);
            }
            ++count;
            map.put(a[i],count);
        }
        System.out.println("Display occurence of every element");
        Set<Map.Entry> entrySet =map.entrySet();
        for(Map.Entry m:entrySet) {
            System.out.println(m.getKey()+"\t"+m.getValue());
        }
    }

}
```

```
Enter values in array
10
20
30
10
20
30
40
10
20
30
Display occurence of every element
10      3
20      3
30      3
40      1
```

Example: WAP to find the duplicate elements from array

```java
import java.util.*;

public class FindOccurenceApp {
    public static void main(String[] args) {
        Scanner xyz = new Scanner(System.in);
        LinkedHashMap map = new LinkedHashMap();
        int a[] = new int[10];
        System.out.println("Enter values in array");
        for (int i = 0; i < a.length; i++) {
            a[i] = xyz.nextInt();
        }
        for (int i = 0; i < a.length; i++) {
            // Integer count=(Integer)map.get(a[i]);
            Object obj = map.get(a[i]);
            Integer count = (Integer) obj;
            if (count == null) {
                count = new Integer(0);
            }
            ++count;
            map.put(a[i], count);
        }
        System.out.println("Display occurence of every element");
        Set<Map.Entry> entrySet = map.entrySet();
        for (Map.Entry m : entrySet) {
            if ((Integer) m.getValue() > 1) {
                System.out.println(m.getKey() + "\t" + m.getValue());
            }
        }
    }
}
```

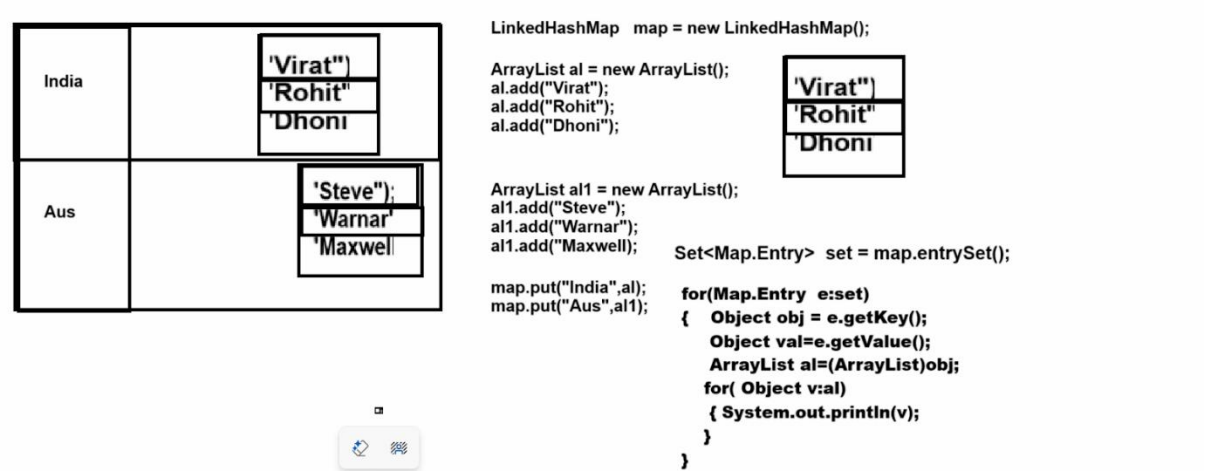
```
Enter values in array
10
20
30
40
50
10
20
30
10
20
Display occurence of every element
10      3
20      3
30      2
```

Example: WAP to find unique values from array

```java
import java.util.*;

public class FindOccurenceApp {
    public static void main(String[] args) {
        Scanner xyz = new Scanner(System.in);
        LinkedHashMap map = new LinkedHashMap();
        int a[] = new int[10];
        System.out.println("Enter values in array");
        for (int i = 0; i < a.length; i++) {
            a[i] = xyz.nextInt();
        }
        for (int i = 0; i < a.length; i++) {
            // Integer count=(Integer)map.get(a[i]);
            Object obj = map.get(a[i]);
            Integer count = (Integer) obj;
            if (count == null) {
                count = new Integer(0);
            }
            ++count;
            map.put(a[i], count);
        }
        System.out.println("Display occurence of every element");
        Set<Map.Entry> entrySet = map.entrySet();
        for (Map.Entry m : entrySet) {
            if ((Integer) m.getValue() ==1) {
                System.out.println(m.getKey() + "\t" + m.getValue());
            }
        }
    }

}
```

```
<terminated> FindOccurenceApp [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.j
Enter values in array
10
20
30
10
20
30
10
20
30
50
Display occurence of every element
50      1
```

Example: Implement following scenario using Map




```java
package org.techhub.map;
import java.util.*;
public class TourApplication {
    public static void main(String[] args)
    {     LinkedHashMap map = new LinkedHashMap();

          ArrayList al = new ArrayList();
          al.add("Virat");
          al.add("dhoni");
          al.add("rohit");

          ArrayList al1 = new ArrayList();
          al1.add("Steve");
          al1.add("Warnar");
          al1.add("Maxwell");

          map.put("India", al);
          map.put("Aus", al1);

          Set<Map.Entry> set=map.entrySet();
          for(Map.Entry m:set) {
              Object key=m.getKey();
              System.out.println("============"+key+"==================");
              Object value=m.getValue();
              ArrayList a=(ArrayList)value;
              for(Object val:a) {
                 System.out.println(val);
              }
          }
    }
}
```

```
<terminated> TourApplication [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspo
============India====================
Virat
dhoni
rohit
============Aus====================
Steve
Warnar
Maxwell
```

Source code above screenshot

```
package org.techhub.map;
import java.util.*;
public class TourApplication {
	public static void main(String[] args)
	{    LinkedHashMap map = new LinkedHashMap();

		ArrayList al = new ArrayList();
		al.add("Virat");
		al.add("dhoni");
		al.add("rohit");

		ArrayList al1 = new ArrayList();
		al1.add("Steve");
		al1.add("Warnar");
		al1.add("Maxwell");

		map.put("India", al);
		map.put("Aus", al1);

		Set<Map.Entry> set=map.entrySet();
		for(Map.Entry m:set) {
			Object key=m.getKey();

	  System.out.println("============="+key+"====================");
			Object value=m.getValue();
			ArrayList a=(ArrayList)value;
			for(Object val:a) {
				System.out.println(val);
			}
		}
```

```
	}
}
```

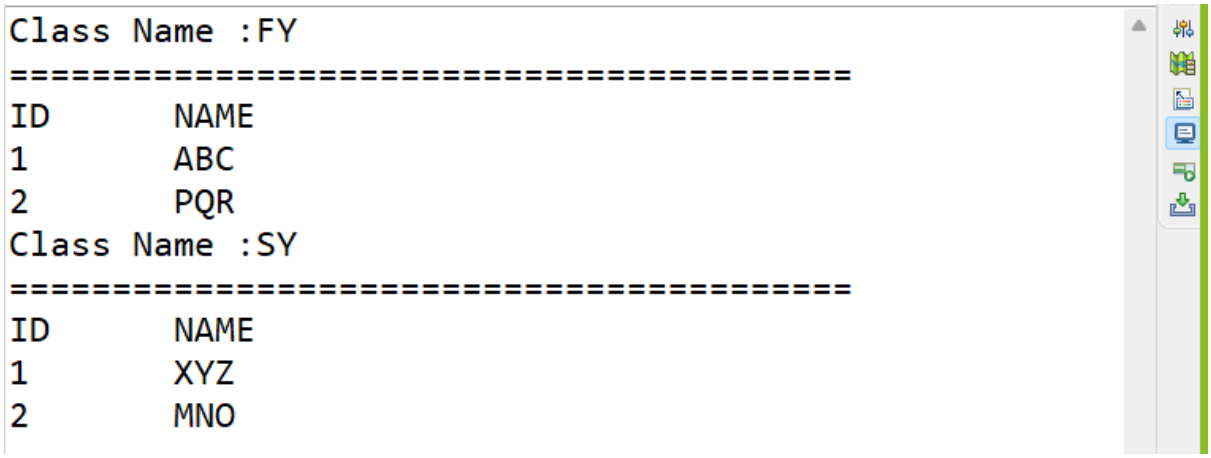
Example: MAp within collection with user defined objects

```
package org.techhub.map;
import java.util.*;
class Student {
	private int id;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	private String name;
	public Student() {

	}
	public Student(int id, String name) {
		this.id = id;
		this.name = name;
	}
```

```
}

public class DeptMapApplication {
	public static void main(String[] args) {
		LinkedHashMap dept = new LinkedHashMap();
			Student s1= new Student(1,"ABC");
			Student s2= new Student(2,"PQR");

			ArrayList first=new ArrayList();
			first.add(s1);
			first.add(s2);

			Student s3 = new Student(1,"XYZ");
			Student s4 = new Student(2,"MNO");

			ArrayList second = new ArrayList();
			second.add(s3);
			second.add(s4);
			dept.put("FY",first);
			dept.put("SY", second);

			Set<Map.Entry> entrySet=dept.entrySet();
			for(Map.Entry m:entrySet) {
				Object key= m.getKey();
				System.out.println("Class Name :"+key);

	System.out.println("=======================================");
				Object val=m.getValue();
				ArrayList a=(ArrayList)val;
				System.out.println("ID\tNAME");
```

```
                    for(Object o:a) {
                        Student s=(Student)o;
                        System.out.println(s.getId()+"\t"+s.getName());
                    }
                }
        }

}
```

Output



Generics

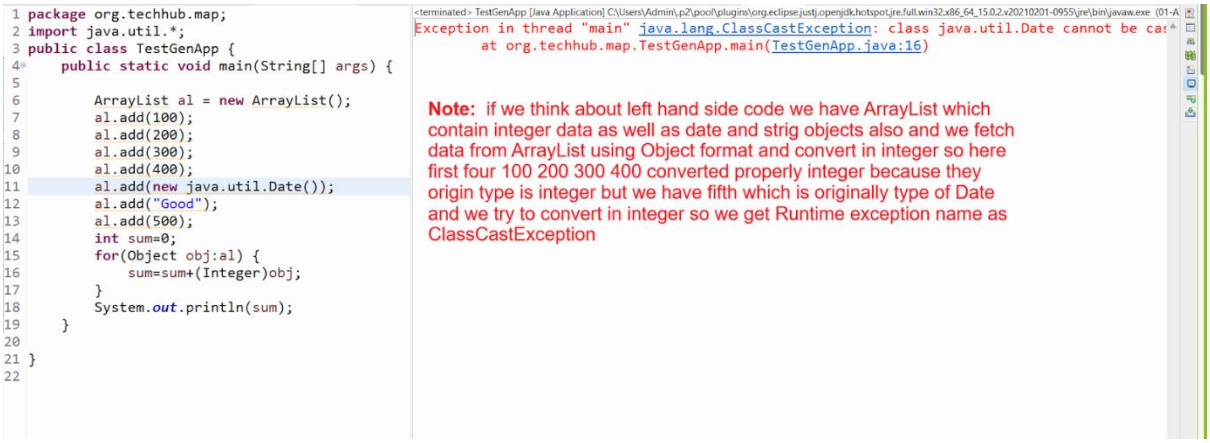
Generics is concept launch by JAVA in JDK 1.5 version and which is used for avoid ClassCastException at run time

Q. What is ClassCastException and when does it occur?

ClassCastException occur when we try to perform referential conversion

And it occur when we want to convert object in to specified class but if we different type of class at program runtime then casting not possible so JVM generate ClassCastException to us at run time

Suppose consider we are working with Collection framework and we try to store different types of data in collection then every element added in collection as Object type and when we try to retrieve data from collection then we get data in the form of Object class and we cannot perform direct operation on Object class then we need to cast in its original type then there is possibility of ClassCastException shown in following code



Note: if we want to solve above problem we have two solutions

1. **Use instanceof operator :** instanceof operator is used for check reference belongs from particular class or not if reference is belong from particular class return true otherwise return false.

Example with source code

```
package org.techhub.map;

import java.util.*;

public class TestGenApp {

        public static void main(String[] args) {


                ArrayList al = new ArrayList ();

                al.add(100);

                al.add(200);

                al.add(300);

                al.add(400);
```

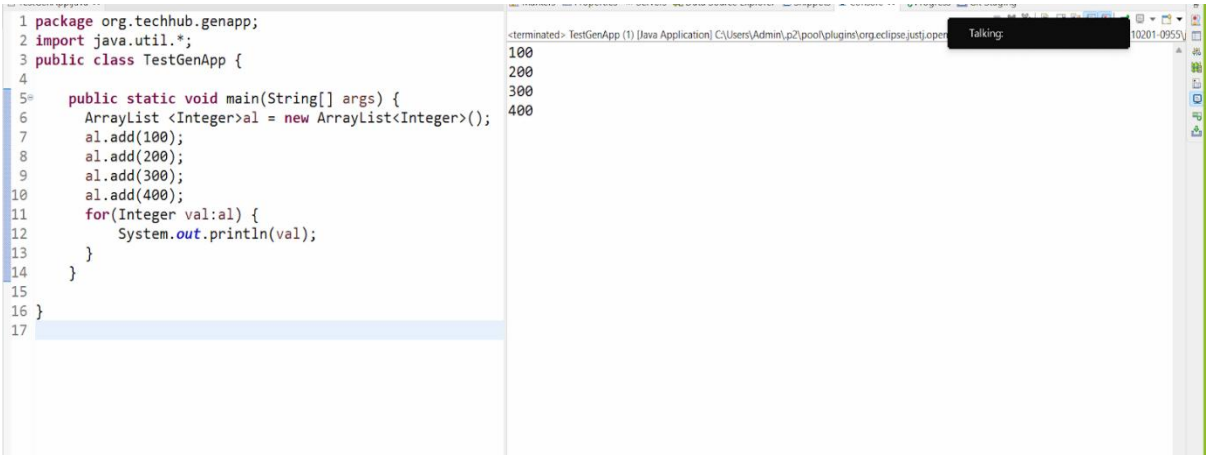
```
        al.add(new java.util.Date());
        al.add("Good");
        al.add(500);
        int sum=0;
        for(Integer  obj:al) {
            if(obj instanceof Integer) {
            sum=sum+(Integer)obj;
            }
        }
        System.out.println(sum);
    }
}
```

2. Use Generics : Generics is denoted by < >

Benefits of Generics

---

1. When we use generics with any Java Objects we not required to perform casting
2. We can customize our own classes and interfaces as generic classes and interfaces
3. Generic provide facility to us can work with more values at time by using wildcard generics and bounded generics



```
package org.techhub.genapp;
import java.util.*;
public class TestGenApp {

	public static void main(String[] args) {
	  ArrayList <Integer>al = new ArrayList<Integer>();
	  al.add(100);
	  al.add(200);
	  al.add(300);
	  al.add(400);
	  for(Integer val:al) {
		System.out.println(val);
	  }
	}
}
```

Note: if we think about above code we can say we have Collection with integer type means in Above ArrayList we cannot store data other than integer data type

Means we lost the main feature of collection to store different types of data

How to store different type of data in collection by using Generics

---

If we want to different type of data in collection using Generics we have to create POJO class and store all data in POJO class and use POJO class name as Data type with generics

```
package org.techhub.genapp;
import java.util.*;
class Employee {
	private int id;
	private String name;
	private int sal;
```

```
	public Employee() {
	}
	public Employee(int id, String name, int sal) {
		this.id = id;
		this.name = name;
		this.sal = sal;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getSal() {
		return sal;
	}
	public void setSal(int sal) {
		this.sal = sal;
	}
}
public class TestGenApp {
	public static void main(String[] args) {
```

```
		ArrayList<Employee> al = new ArrayList<Employee>();
		Employee emp1 = new Employee(1, "ABC", 10000);
		Employee emp2 = new Employee(2, "PQR", 20000);
		Employee emp3 = new Employee(3, "STV", 30000);
		al.add(emp1);
		al.add(emp2);
		al.add(emp3);

		Iterator<Employee> i = al.iterator();
		while (i.hasNext()) {
			Employee e = i.next();
	System.out.println(e.getId() + "\t" + e.getName() + "\t" + e.getSal());
		}
	}
}
```

How to create user defined class and user define interface as generics

---

If we want to create user defined class as Generics or interface as Generics class we have some inbuilt notation classes provided by java to us

E - E stands generic elements means we can say when we pass E as parameter in method or with class means method can accept any kind of data but we can restrict that method for work with particular type of data

T - T Stands generic type means when we have some data which may be change its type in implementer classes or some other place at run time then we can use T as Generic notations

K - K stands Generics key means when we want to create then we can use key as generic value

V - V stands for Generic value

Create user define class as Generic class

```
1 package org.techhub.genapp;
2 import java.util.*;
3 class Test<E>{
4
5     void add(E val) {
6         System.out.println(val);
7     }
8 }
9 public class TestApp {
10
11     public static void main(String[] args) {
12
13
14
15         Test <Integer>t=new Test<Integer>();
16         t.add(100);
17         t.add(200);
18         t.add(300);
19
20
21         Test <String>t1= new Test<String>();
22         t1.add("Good");
23
24     }
25 }
26
```

Talking:

Note: if we think about left hand side we have class
class Test<E> this indicate it is generic class
means we can use generic notation with Test class object at run time
can decide parameter type

void add(E val): this method indicate we can store any kind of data or accept kind of parameter but when we use generic notation with object then method can work according to object parameter

Test <Integer> t = new Test<Integer>() this method indicate t can work with integer type of data means when t call any method from Test class then method parameter is type of integer or method can return type of integer value

Test <String> t1 = new Test<String>() : here t1 indicate can work with string type of data means when we call any method using t1 then method can accept string type of parameter or method can return string type of parameter

User define interface with generics

```
1 package org.techhub.genapp;
2 interface Area<T>{
3       void setRadius(T r);
4 }
5 class Circle<E> implements Area<Integer>{
6     @Override
7     public void setRadius(Integer r) {
8         System.out.println("Integer r "+r);
9     }
10 }
11 class Cirm<E> implements Area<Double>{
12     @Override
13     public void setRadius(Double r) {
14         System.out.println("Double r  "+r);
15     }
16 }
17 public class InterfaceWithGenApp {
18
19     public static void main(String[] args) {
20       Circle <Integer> c = new Circle<Integer>();
21       c.setRadius(5);
22       Cirm <Double> cm = new Cirm<Double>();
23       cm.setRadius(10.6);
24     }
25
26 }
```

```
<terminated> InterfaceWithGenApp [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.jus
Integer r 5
Double r  10.6
```

Talking: Adinath Giri

Example with source code

```
package org.techhub.genapp;

interface Area<T>{

        void setRadius(T r);

}

class Circle<E> implements Area<Integer>{

        @Override
```

```
	public void setRadius(Integer r) {
		System.out.println("Integer r "+r);
	}
}
class Cirm<E> implements Area<Double>{
	@Override
	public void setRadius(Double r) {
		System.out.println("Double r  "+r);
	}
}
public class InterfaceWithGenApp {

	public static void main(String[] args) {
	  Circle <Integer> c = new Circle<Integer>();
	  c.setRadius(5);
	  Cirm <Double> cm = new Cirm<Double>();
	  cm.setRadius(10.6);
	}
}
```

WildCard Generics

Wildcard generics means generics work with any value with any data type and it is denoted by using ?

Normally wildcard generics use by developer when we pass collection as parameter value

There are two types of wild generics

---

1. **Unbounded wildcard generics :** unbounded generics means generics without any restrictions called as unbounded generis

2. **Example:**

```java
package org.techhub.genapp;
import java.util.*;
class ABC{
    void accept(List<?> list) {
        for(Object obj:list) {
            System.out.println(obj);
        }
    }
}
public class WildCardGenApp {
    public static void main(String[] args) {
        ABC a1 = new ABC();
        System.out.println("List with integer data type");
        List<Integer> list = Arrays.asList(10,20,30,40,50,60);
        a1.accept(list);
        System.out.println("List with float data type");
        List<Float> list1=Arrays.asList(5.4f,2.3f,5.5f);
        a1.accept(list1);
    }
}
```

```
<terminated> WildCardGenApp [Java Application] C:\Users\Admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20
List with integer data type
10
20
30
40
50
60
List with float data type
5.4
2.3
5.5
```

2. Bounded wildcard generics : bounded wildcard generics means we can give some restrictions with generics called as bounded generics

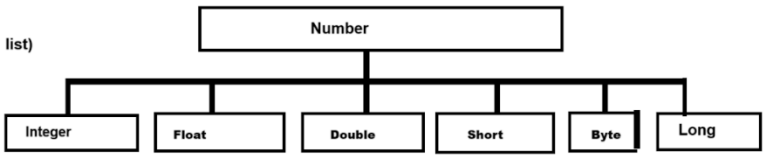
There are two types of bounded generics

---

a. Upper bound generics: upper bound generics can be specified by using < ? extends superclass name>

If we specify upper bound generics means we can use any child class name as replacement of ? mark

```java
class TestUpperBound
{
   void acceptNumberList(List <? extends Number> list)
   {   for(Number n: list)
       {
          System.out.println(n);
       }
   }
}

public class TestUpperBoundApp
{  public static void main(String x[])
   {
      TestUpperBound tub =new TestUpperBound();
      List<Integer> list = Arrays.asList(10,20,30,40,50);
      tub.acceptNumberList(list);

      List<Float> list1 = Arrays.asList(5.4f,2.5f,3.5f);
       tub.acceptNumberList(list1);
   }
}
```



b. Lower bound generics : lower bound generics denoted by using <? super chiclassname> means here ? can replace parent class reference

```
package org.techhub.genapp;
import java.util.*;
class A {
	void show() {
		System.out.println("I am A method");
	}
}
class B extends A {
	void display() {
		System.out.println("I am display method");
	}
}
class C {
	void accept(List<? super B> list) {
		for (Object a1 : list) {
			((A) a1).show();
		}
	}
}
public class LowerBoundApp {
	public static void main(String x[]) {
		C c1 = new C();
		A a1 = new A();
		A a2 = new A();
		List<A> list = new ArrayList<A>();
		list.add(a1);
		list.add(a2);
		c1.accept(list);   }
}
```

Object class and its method

_____

Q. What is an Object class?

_____

Object class is parent of all classes in JAVA and it is a member of java.lang package and java.lang is a default package java means no need to import it.

Q. Why is the Object class the parent of every class in JAVA?

_____

Object class contains some inbuilt methods which are required to perform daily or regular operation with objects so developers can override methods provided by Object class and perform operations.

Methods of Object class

_____

boolean equals(Object): this method help us compare two object with each other using its content and if objects are equal return true otherwise return false

int hashCode(): this method help us generate the user defined hashcode and internally equals() and hashcode() method has contract

String toString(): this method can convert user defined object in to string format

Object clone(): this method can perform object cloning with user defined data types.

Class getClass(): this method return reference of Class from java.lang

void wait(): this method can work with unconditional wait .

void wait(int): this method can work with conditional wait.

void notify():this method calls a waiting thread one by one and single thread at time.

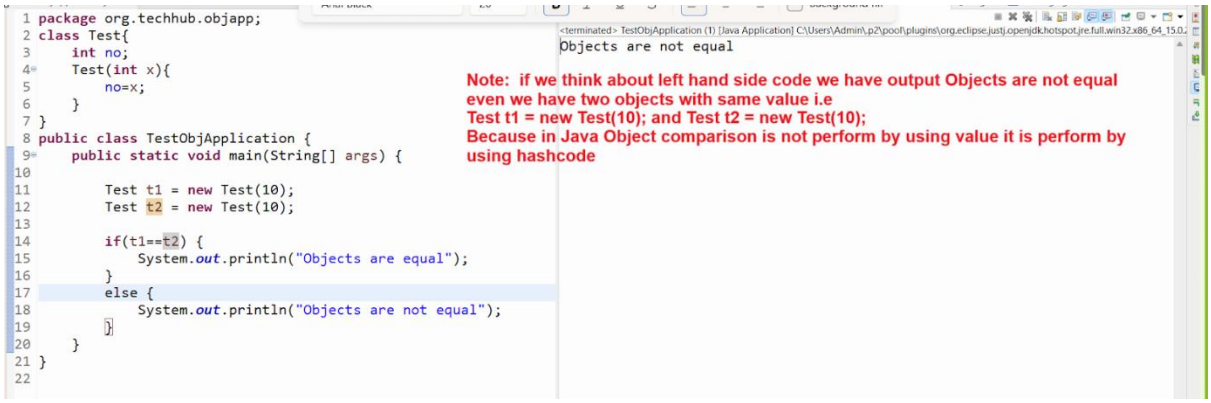
void notifyAll(): this method call a all waited thread at time

void finalize(): this method calls automatically when we call System.gc() method means normally this method helps us to perform garbage collection or resource cleaning purposes.

static{

} **:** static block is also part of Object class

Example

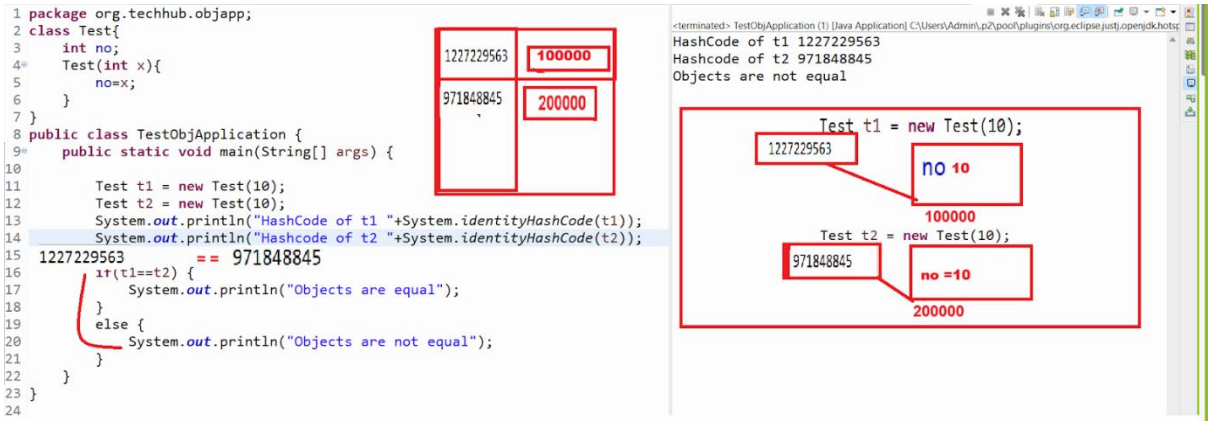


Q. What is hashcode?

Hashcode is unique integer number provide by JVM to every object in memory and it is not actual address it is associated internally with memory address and if we want to check the hashcode generated by JVM then we have method name as System.identityHashCode()

How to hashcode associated with a memory address?

Internally JVM use hashcode as key and memory address as value but not provided access memory address and memory address is unique so JVM generate hashcode unique for every object



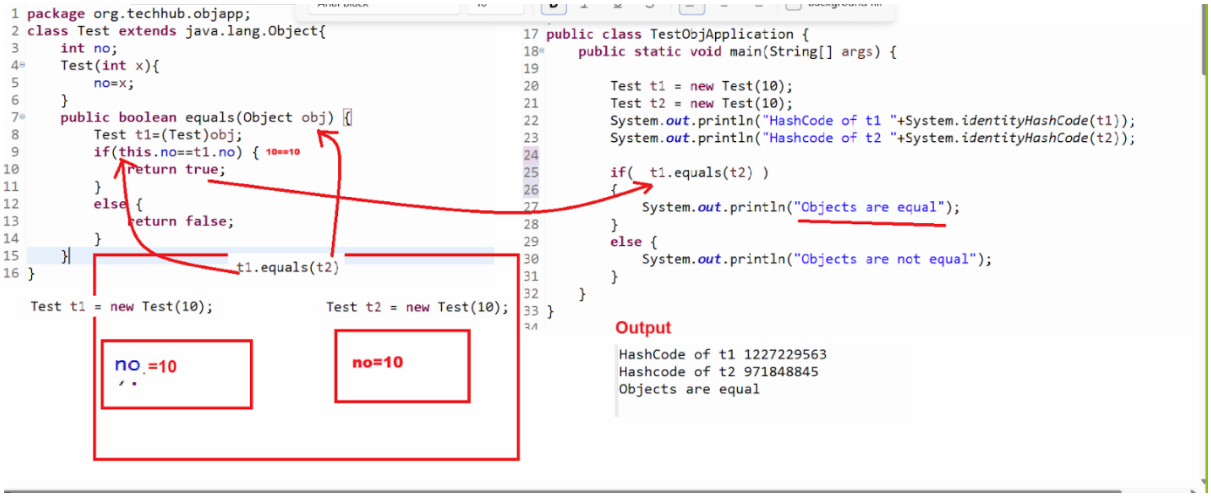
Note: if we think about above code we have reference t1 with hashcode

1227229563 and reference t2 with hashcode 971848845 means when JVM execute statement if(t1 == t2) means if(1227229563 ==971848845) here both hashcode are different so we get answer objects are not equal

Important points: JVM cannot generate same hashcode for two objects

Q. If JVM does not generate the same hashcode for two objects and if JVM considers two objects same when their hashcode code then how do they compare objects with each other in JAVA?

If we want to solve this problem java suggest us override two methods provided by Object class name as equals() and hashcode() and using equal method we can compare object content and if object contents are equal then return true otherwise return false and if two object contain same content then generate same hashcode for both object using hashcode() method provided by Object class and if objects are different then generate different hashcode for two objects using hashcode methods

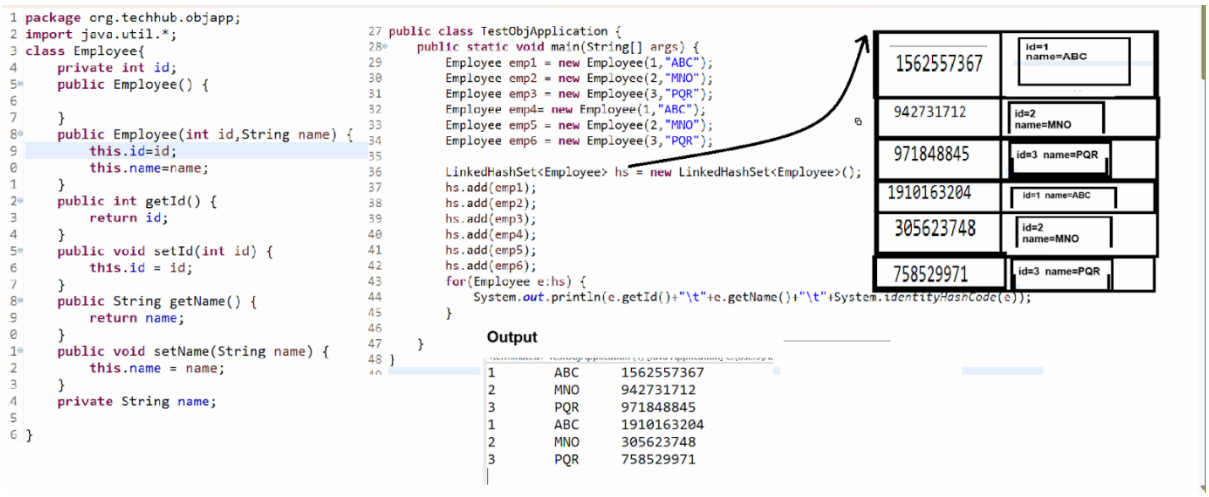


Note: if we think about above output we override only equals() method and we compare two objects and we get Objects are equals without overriding user defined hashCode method and here hashcode of two objects are different generated by JVM

Q. What is the purpose of a user-defined hashcode method or why does java suggest that users override their own hashcode method and generate the same hashcode when two objects are equal?

Note: yes we can compare two object with each other without using user defined hash code

But if we store your user-defined object in Set collection and not override hashcode() method with equals() method then set collection can store duplicate data.



```
package org.techhub.objapp;
import java.util.*;
class Employee{
    private int id;
    public Employee() {

    }
    public Employee(int id,String name) {
        this.id=id;
        this.name=name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    private String name;

}
```

```
public class TestObjApplication {
    public static void main(String[] args) {
        Employee emp1 = new Employee(1,"ABC");
        Employee emp2 = new Employee(2,"MNO");
        Employee emp3 = new Employee(3,"PQR");
        Employee emp4= new Employee(1,"ABC");
        Employee emp5 = new Employee(2,"MNO");
        Employee emp6 = new Employee(3,"PQR");

        LinkedHashSet<Employee> hs = new LinkedHashSet<Employee>();
        hs.add(emp1);
        hs.add(emp2);
        hs.add(emp3);
        hs.add(emp4);
        hs.add(emp5);
        hs.add(emp6);
        for(Employee e:hs) {
            System.out.println(e.getId()+"\t"+e.getName()+"\t"+System.identityHashCode(e));
        }

    }
}
```


Output

```
1       ABC     1562557367
2       MNO     942731712
3       PQR     971848845
1       ABC     1910163204
2       MNO     305623748
3       PQR     758529971
```

Note: if we think about above code we store 6 objects in LinkedHashSet with duplicated data and normally we said set not store duplicate data but in above set contain duplicated data

Q. How to store duplicate data?

When we store data in set the set collection compare the hashcode of data if set found unique hashcode of data then set not allow data but if we store user define objects in set collection and override equals() and hashcode() method in user defined class then set compare the hashcode for that object generated JVM and JVM not generate same hashcode for two different objects even object contain same data so set never find duplicates in this case so set allow duplicate data

So for avoid this problem Java Suggest us override equals() and hashcode() method in class or in user defined class and generate same user hashcode if two objects are equal so set collection use user hashcode for comparison if set found duplicate user hashcode then not store data otherwise store data shown in following code.

Example with source code

package org.techhub.objapp;

import java.util.*;

```
class Employee{
	private int id;
	public Employee() {

	}
	public Employee(int id,String name) {
		this.id=id;
		this.name=name;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	private String name;

	public boolean equals(Object obj) {
		Employee e=(Employee)obj;
		if(this.id==e.id && this.name.equals(e.name)) {
			return true;
		}
		else {
```

```
                return false;
            }
        }
        public int hashCode() {
            return id*10000;
        }
}
public class TestObjApplication {
        public static void main(String[] args) {
            Employee emp1 = new Employee(1,"ABC");
            Employee emp2 = new Employee(2,"MNO");
            Employee emp3 = new Employee(3,"PQR");
            Employee emp4= new Employee(1,"ABC");
            Employee emp5 = new Employee(2,"MNO");
            Employee emp6 = new Employee(3,"PQR");
            LinkedHashSet<Employee> hs = new LinkedHashSet<Employee>();
            hs.add(emp1);
            hs.add(emp2);
            hs.add(emp3);
            hs.add(emp4);
            hs.add(emp5);
            hs.add(emp6);
            for(Employee e:hs) {

        System.out.println(e.getId()+"\t"+e.getName()+"\t"+System.identityHashCode(e));
            }
        }
}
```

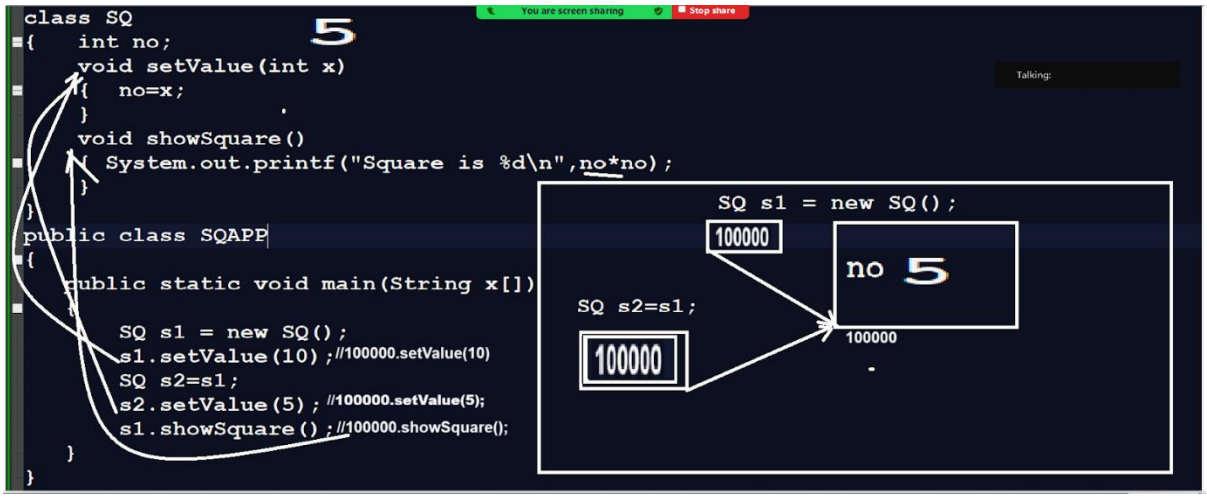
Object cloning concept using JAVA

Q. What is an Object clone?

If we create a duplicate copy of an object called an Object clone.

Q. Why do we need to perform object cloning?

Some time we apply more than one reference on single object and if we want to perform change on object using any one reference the object content may be lost or vanish or update if we want to solve this problem so java suggest we can use object cloning approach so we can persist the object previous content as well as we have one more copy of same object we can use as it is or we can change as per your requirement so original copy of object not hamper by modification



Note: if we think about above code we have main method with code

SQ s1 = new SQ() means we create object of SQ class and consider address of object is 100000 and store this address in s1 reference and we have statement s1.setValue(10) means 100000.setValue(10) means we call function using object address is 100000 and store 10 value in no variable which is present on 100000 address and we have one more statement SQ s2=s1 means we initialize s1 address in s2 reference means s1 and s2 points to same memory location i.e 100000 and we have statement s2.setValue(5) means 100000.setValue(5) so this 5 value get override in no which is present on 100000 location so

we lost the previous content of no on address 100000 and no is part of object so lost the object previous content and when we call statement s1.showSquare() means 100000.shwSquare() so we get answer Square is 25 i.e updated value

So conclusion is we lost the object historical data so if we want to solve this problem java suggest us use object cloning concept

How to implement the object cloning practically

If we want to implement object cloning practically we have clone() method provided by Object class

Steps to implement the object cloning

---

1. **Create user define class and implements Cloneable interface**

```
class SQ implements Cloneable
{   int no;
   void setValue(int x)
         {  no=x;
         }
         void showSquare()
         { System.out.printf("Square is %d\n",no*no);
         }
}
```

Important point:

Cloneable is a member of java.lang package and it is marker interface in JAVA

Q. What is the marker interface?

Marker interface means a interface which does not contain any method or state means blank interface but JVM provide special run time environment to them and marker interface provide some special information to class where they implement and JVM behave object of that class according to information provided by Marker interface

Example of marker interfaces

a. Cloneable b. Serializable etc

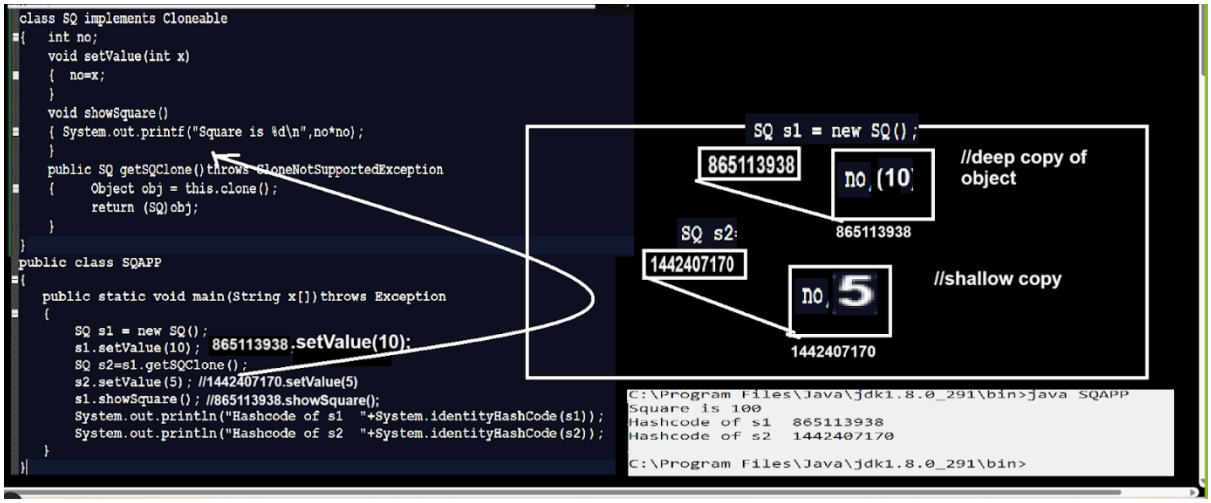
Example

```
Compiled from "Cloneable.java"
public interface java.lang.Cloneable {
}
```

Or

```
C:\Program Files\Java\jdk1.8.0_291\bin>javap java.io.Serializable
Compiled from "Serializable.java"
public interface java.io.Serializable {
}
```

2. Create a Factory method and call clone() in it and return reference to the new object created by the clone method .



Note: if we think about above code we create new object by using clone() method but we create object by using new then also we get same answer like as clone

Q. What is the difference between object creation using new and using clone()?

---

1. When we create object by using new keyword then constructor get executed and when we create object using clone then constructor not executed
2. When we use new then logic of constructor get executed if written by developer but by clone logic from constructor not executed written by developer
3. When we use new keyword then instance variable and static initialize according to default values provided by java means new construction of memory but when we create object by clone then existing object content copied in new object means there is no new construction of memory from initialization just constructor memory by using existing content

Example with source code

```
class SQ  implements Cloneable
{   int no;
  static int count;
   SQ(){
                ++count;
                System.out.println("Constructor call number of time  "+count);
        }
   void setValue(int x)
        {  no=x;
        }
        void showSquare()
        { System.out.printf("Square is %d\n",no*no);
        }
        public SQ getSQClone()throws CloneNotSupportedException
        {    Object obj = this.clone();
```

```
                return (SQ)obj;
        }
}
public class SQAPP
{
  public static void main(String x[])throws Exception
  {
     SQ s1 = new SQ();
          s1.setValue(10);
          SQ s2=new SQ();
     s2.setValue(5);
          s1.showSquare();
          System.out.println("Hashcode of s1  "+System.identityHashCode(s1));
          System.out.println("Hashcode of s2  "+System.identityHashCode(s2));
  }
}
```

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac SQAPP.java

C:\Program Files\Java\jdk1.8.0_291\bin>java SQAPP
Constructor call number of time  1
Constructor call number of time  2
Square is 100
Hashcode of s1  865113938
Hashcode of s2  1442407170

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Note: if we think about above code we execute constructor two times because we create two objects by using new keyword so constructor executed two times

Example object creation using clone method

```
class SQ  implements Cloneable
{   int no;
```

```
  static int count;
   SQ(){
                    ++count;
          System.out.println("Constructor call number of time  "+count);
          }
   void setValue(int x)
          {  no=x;
          }
          void showSquare()
          { System.out.printf("Square is %d\n",no*no);
          }
          public SQ getSQClone()throws CloneNotSupportedException
          {    Object obj = this.clone();
                  return (SQ)obj;
          }
}
public class SQAPP
{
  public static void main(String x[])throws Exception
  {
     SQ s1 = new SQ();
            s1.setValue(10);
            SQ s2=s1.getSQClone();
     s2.setValue(5);
            s1.showSquare();
            System.out.println("Hashcode of s1  "+System.identityHashCode(s1));
            System.out.println("Hashcode of s2  "+System.identityHashCode(s2));
  }
}
```

Output

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac SQAPP.java

C:\Program Files\Java\jdk1.8.0_291\bin>java SQAPP
Constructor call number of time  1
Square is 100
Hashcode of s1  865113938
Hashcode of s2  1442407170

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Static and instance block

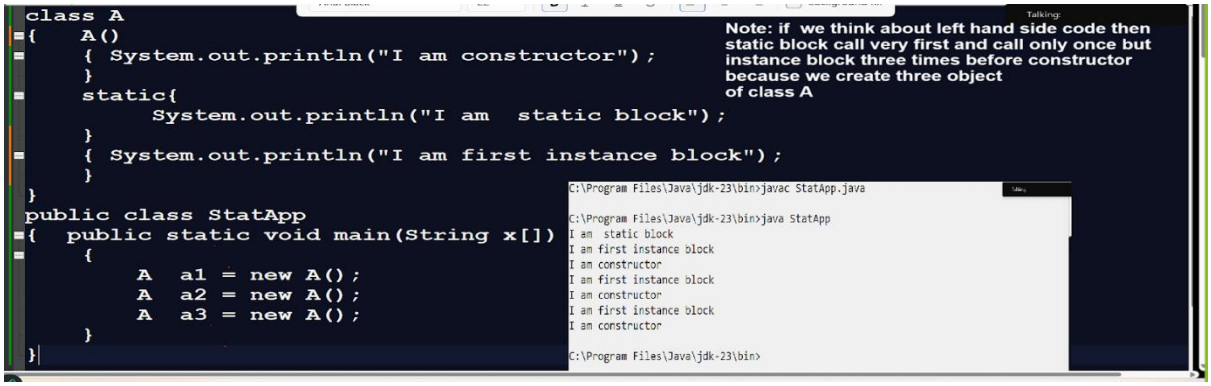
Static block is a member of object class and instance block is user defined block define by user in class

Q. What is the difference between instance block and static block?

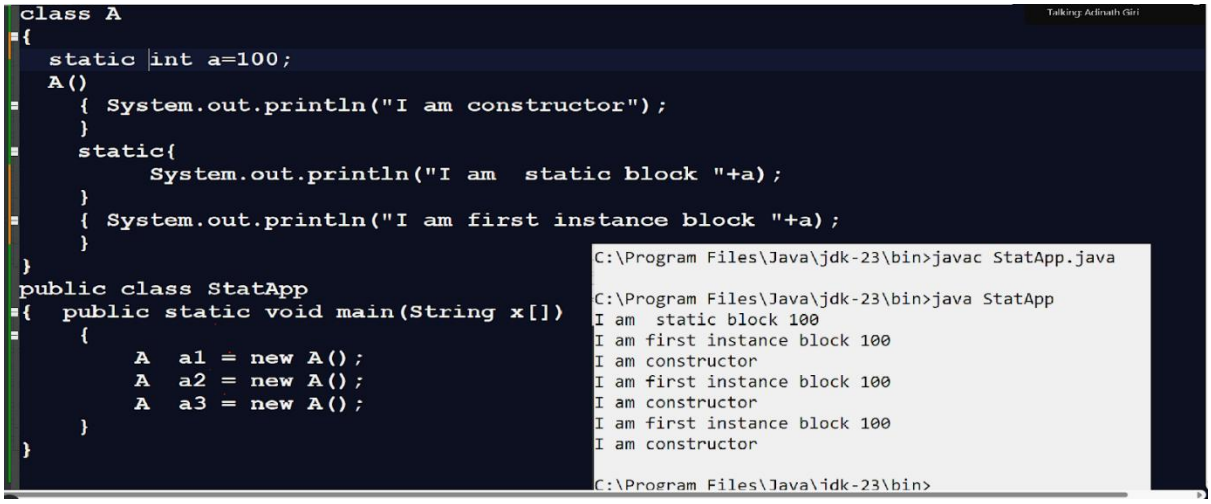
---

1. **Static blocks are executed very first in application even before the main method**

2. **Static blocks execute only once or the first time when we create an object of the class but instance block calls every time when we create an object of the class but before the constructor.**



3. Static block can use only static variable but instance block can use static and non static variable or instance variable

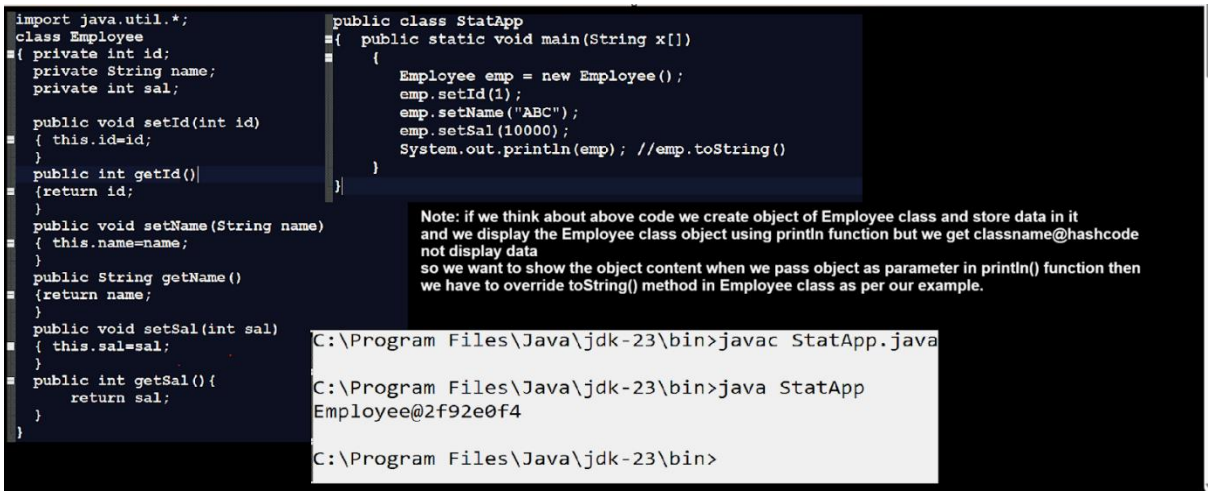


String toString(): this method is used for convert java object in to string format

Normally we use this method following purpose

1. When we want to convert an object to string format.
2. When we want display object content when we pass object as parameter in println method
3. When we store object in collection and display the object content when we print collection

Etc



Now we want to override the toString() method in Employee class.

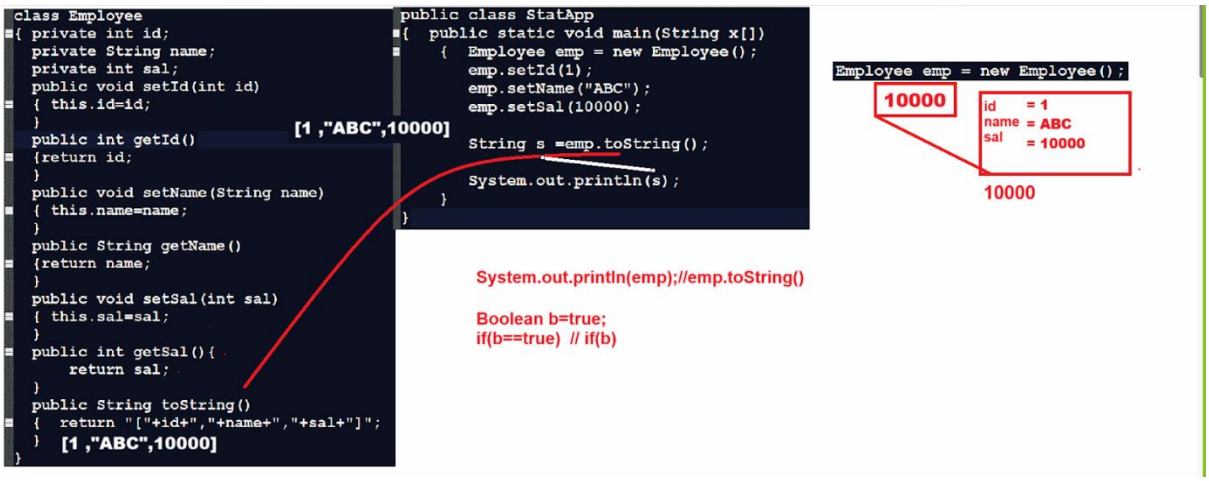
Example: WAP to create class name as Employee and store 3 employee object in ArrayList and display it without iteration

```
import java.util.*;
class Employee
{ private int id;
  private String name;
  private int sal;
  public void setId(int id)
  { this.id=id;
  }
```

```
 public int getId()
 {return id;
 }
 public void setName(String name)
 { this.name=name;
 }
 public String getName()
 {return name;
 }
 public void setSal(int sal)
 { this.sal=sal;
 }
 public int getSal(){
            return sal;
 }
 public String toString()
 {  return "["+id+","+name+","+sal+"]";
 }
}
public class StatApp
{  public static void main(String x[])
          {  Employee emp = new Employee();
             emp.setId(1);
             emp.setName("ABC");
             emp.setSal(10000);
             Employee emp1 = new Employee();
             emp1.setId(1);
             emp1.setName("ABC");
             emp1.setSal(10000);
```

```
        Employee emp2 = new Employee();
        emp2.setId(1);
        emp2.setName("ABC");
        emp2.setSal(10000);
        ArrayList al = new ArrayList();
        al.add(emp);
        al.add(emp1);
        al.add(emp2);

        System.out.println(al);
    }
}
```

Example: WAP to create employee class and convert Employee object in to string



void finalize(): finalize() method is used for resource cleaning purpose means this method call automatically when we perform garbage collection manually using System.gc() method

Means when we delete any object from the heap section and if we want to perform any operation at the time of object of deletion then we can override the finalize() method of Object class and write logic in it.

```
import java.util.*;

class Employee

{ private int id;

  private String name;
```

```
 private int sal;
 public void setId(int id)
 { this.id=id;
 }
 public int getId()
 {return id;
 }
 public void setName(String name)
 { this.name=name;
 }
 public String getName()
 {return name;
 }
 public void setSal(int sal)
 { this.sal=sal;
 }
 public int getSal(){
        return sal;
 }
 public String toString()
 {  return "["+id+","+name+","+sal+"]";
 }
 public void finalize()
 {  System.out.println("I am finalize method");
 }
}
public class StatApp
{  public static void main(String x[])
        {  Employee emp = new Employee();
```

```
            emp.setId(1);
            emp.setName("ABC");
            emp.setSal(10000);
            emp=null;
  System.gc(); //perform garbage collection
//means delete object from memory whose address was present in emp
//currently object not use by any other reference so JVM can delete //object
//from memory called as garbage collection

        }
}
```

Internal working of HashMap

When we create object of HashMap like as HashMap<String> map=new HashMap<String>() then JVM create 16 bucket of slot object and you can store 16 values in hashmap and load factor of HashMap is 75% of his capacity then it double the existing capacity.

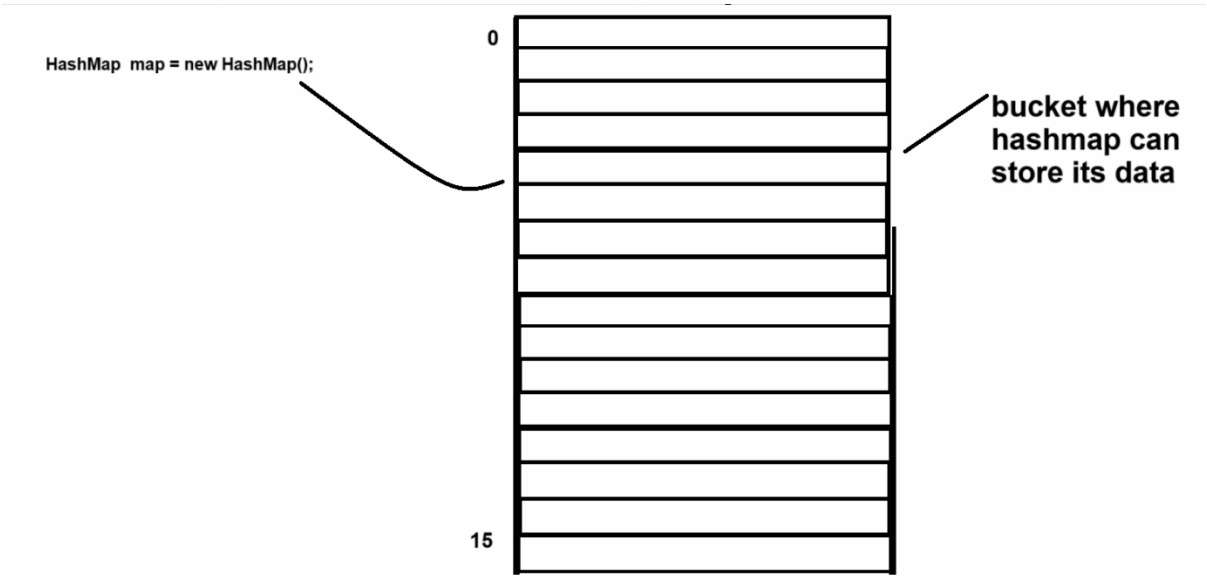
Means when we think about HashMap then internally the capacity 75% means 12 item and when we try to store 13 elements in hashmap then hashmap can increase its capacity from 16 to 32

Note: load factor 75% or 0.75 or ¾

Now we want to discuss about what is bucket and how data store in bucket

---

Bucket is internally array and bucket can store data in the form of LinkedList and LinkedList contain data in the form of node and address.



Suppose consider we are going to store data in map by using put() method like as

HashMap<String,String> map = new HashMap<String,String>();

map.put("FB","A");

Note: Here FB is key of String object

How to store data in HashMap internally

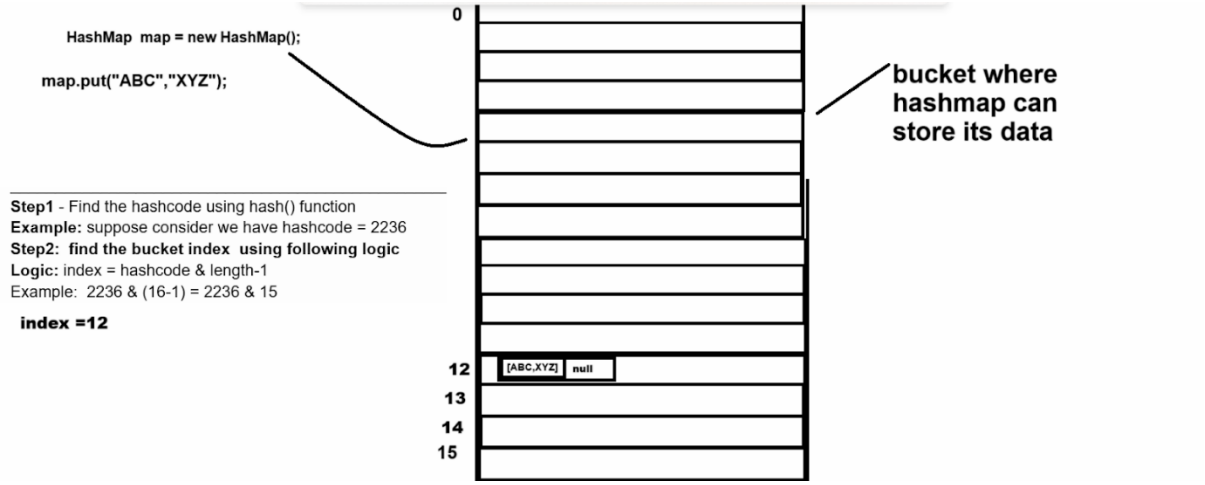
Step1 - Find the hashcode using hash() function

Example: suppose consider we have hashcode = 2236

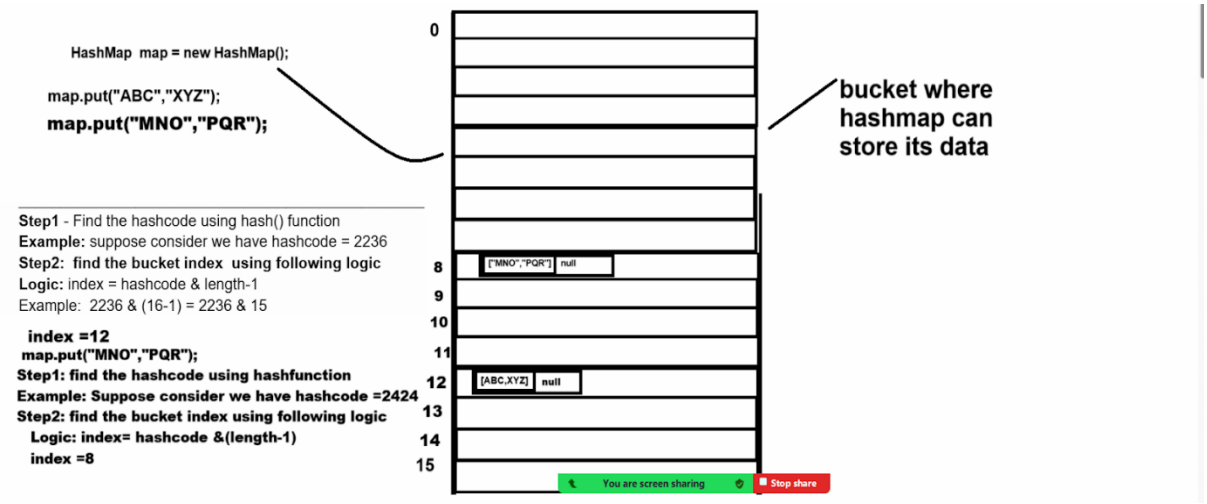
Step2: find the bucket index using following logic

Logic: index = hashcode & length-1

Example: 2236 & (16-1) = 2236 & 15



Now we want to add one more element



Now we want to discuss about the concept of hash collision

Q. What is hash collision?

Collision occur when two different keys produce the same hash code then we need to store two keys in same bucket the JVM compare newly generated key with previous key if newly generate key is same as previously key then override newly generated key on previous key and if newly generated key not same with previous generated key then JVM create new node and store in same index and attach new node with previous node

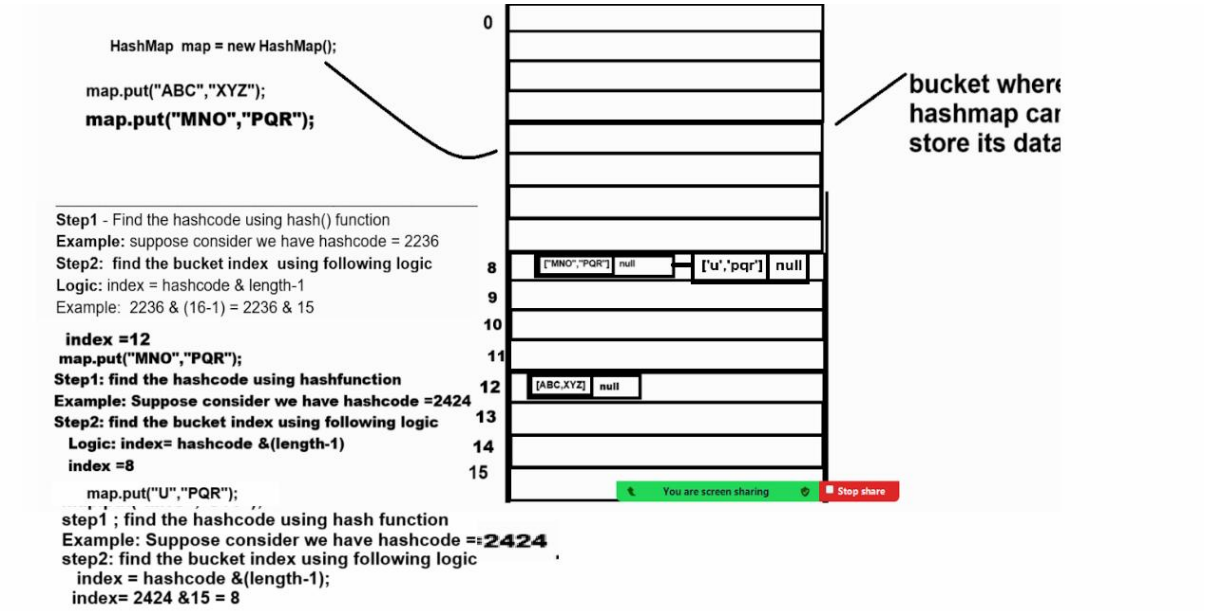
Suppose consider we are going add new data in HashMap

How get() method work internally

---

get() method of hashmap return the value using key very fast

map.get("u"): so the time complexity of get() method is O(1)



Steps

Step1: The JVM find the hashcode of u is 2424

Step2: find the bucked index: 8

Step3: go the bucket index 8

Step4: get the value by using key

Note: HashMap normally store only one node in bucket but if there is hashcode collision then there is possibility to store more than one node in HashMap but if we store more than one in single bucket of the HashMap then there may be possibility to degrade performance

Because we need to travel all node or linked list in that bucket and compare key in every node

So it may degrade the performance of retrieving data so

Java 8 enhance this HashMap

Means Java 8 convert your linked list from bucket in to tree structure after certain threshold and tree structure known as treefy structure and this treefy structure known as red black tree and binary search tree self balancing

Q. What is the red black tree structure?

Red black tree is self balanced tree provide efficient insert/deletion and retrieval operation

Properties of Red black tree

---

1. **Node color :** each node is either red or black
2. **Root Property:** Root is always black
3. **Red Not Property :** red node cannot have red childern means two red nodes can be adjacent
4. **Black depth property :** every path from a node to its descendent leave must have the same number of black nodes.
5. **Leaf node :** all leaf node are black and do not store any data they are placeholders and to maintain tree structure

Example: suppose we have data and we want to perform a red black tree using that data.

10 20 30 15 25 5 12 35 40 32 50

Case 1: Recolor node

Case 2: Red-Red violation

2.1 sibling or uncle is Red

Recolor the parent and the uncle to black and grandparent to red and repeat the fix-up process from the grand parent

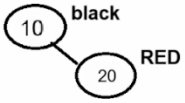
2.2 uncle is black or null so rotation to balance the tree recolor node accordingly

Case 1: Recolor node
Case 2: Red-Red violation
2.1 sibling or uncle is Red
Recolor the parent and the uncle to black and grandparent to red and repeat the fix-up process from the grand parent
2.2 uncle is black or null so rotation to balance the tree recolor node accordingly

10 20 30 15 25 5 12 35 40 32 50

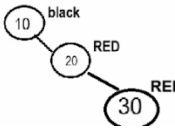
Suppose we consider we have 10 and we want to insert tree as new node then mark it as red

10 is our root node so it must be black



Note: when we insert second node then we can mark it as red

Now we want to insert 30 as new node



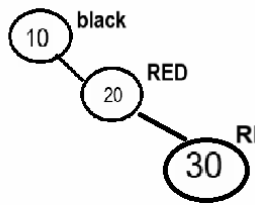
important point here RED RED conflict
when we have red -red conflict then there are two cases
case 1: check uncle node or sibling so uncle should be black or null so we required perform rotation means LL,LR ,RR, RL as well as perform recolor operation
case 2: uncle may be RED if uncle is RED so we required parent color as well as change the grand parent color also.

10 20 30 15 25 5 12 35 40 32 50

Suppose we consider we have 10 and we want to insert tree as new node then mark it as red

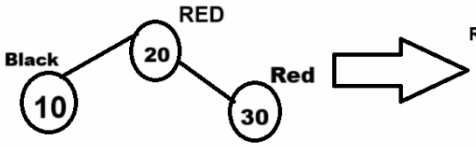
Note: when we insert second node then we can mark it as red

Now we want to insert 30 as new node

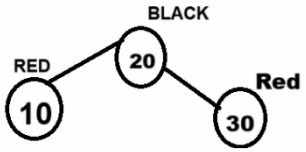


important point here RED RED conflict
when we have red -red conflict then there are two cases
case 1: check uncle node or sibling so uncle should be black or null so we required perform rotation means LL,LR ,RR, RL as well as perform recolor operation
case 2: uncle may be RED if uncle is RED so we required parent color as well as change the grand parent color also.

Note: here we required to perform rotation from right to left so your tree look like as after rotations



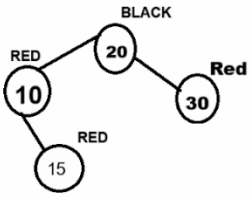
Recolor Tree after rotation



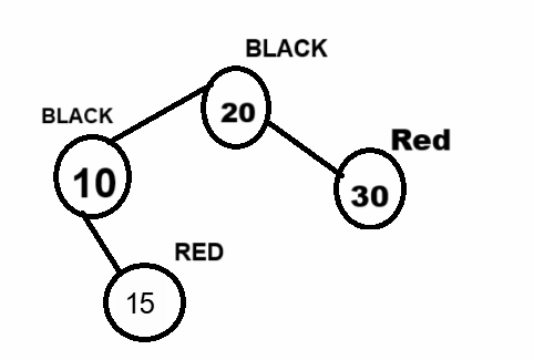
Case 1: Recolor node
Case 2: Red-Red violation
2.1 sibling or uncle is Red
Recolor the parent and the uncle to black and grandparent to red and repeat the fix-up process from the grand parent
2.2 uncle is black or null so rotation to balance the tree recolor node accordingly

10 20 30 15 25 5 12 35 40 32 50

Suppose we consider we have 10 and we want to insert tree as new node then mark it as red



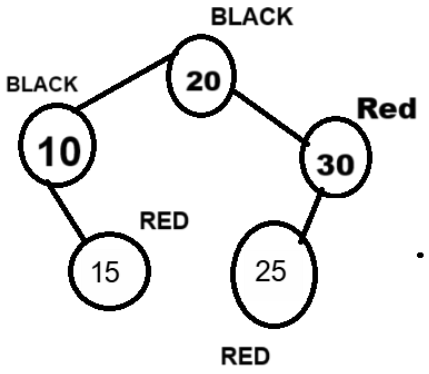
Note: when we insert 15 in Tree then 15 is less than 20 and greater than 10 so 15 should be right hand side of 10 and mark it as red so we have again red and red conflict as per the case 1 we required to find its uncle not so 30 is uncle of 15 and 30 is red not so here we not required rotation just need to perform recolor operation with parent node i.e 10 should be black



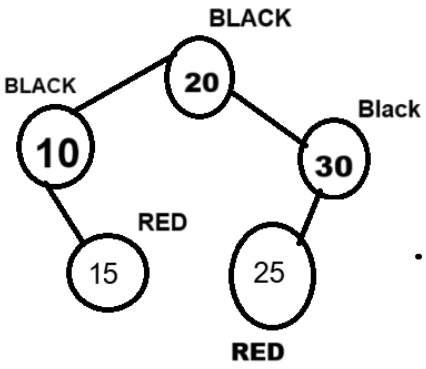
10 20 30 15 25 5 12 35 40 32 50

Suppose we consider we have 10 and we want to insert tree as new node then mark it as red

5

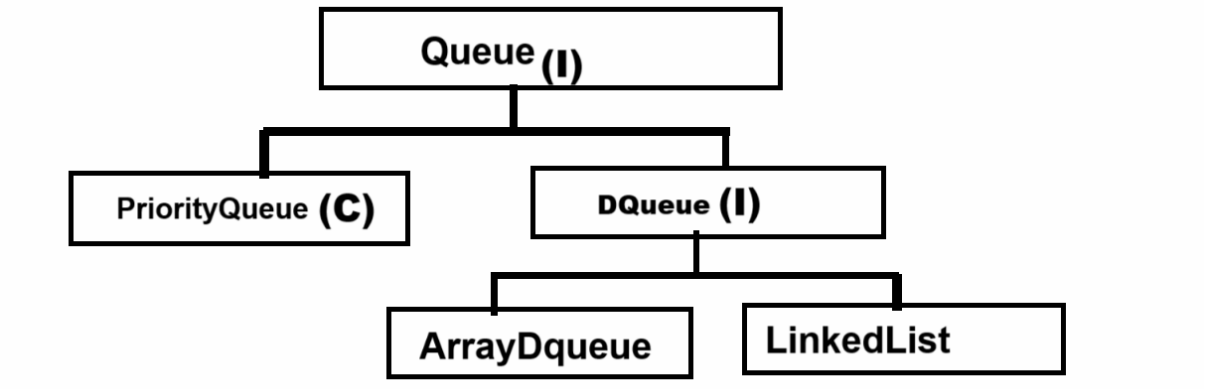


Note: we have again red and red conflict but 30 is not main root it is intermediate node so we required just recolor it



Queue

If we want to work with Queue we have to know the hierarchy of Queue

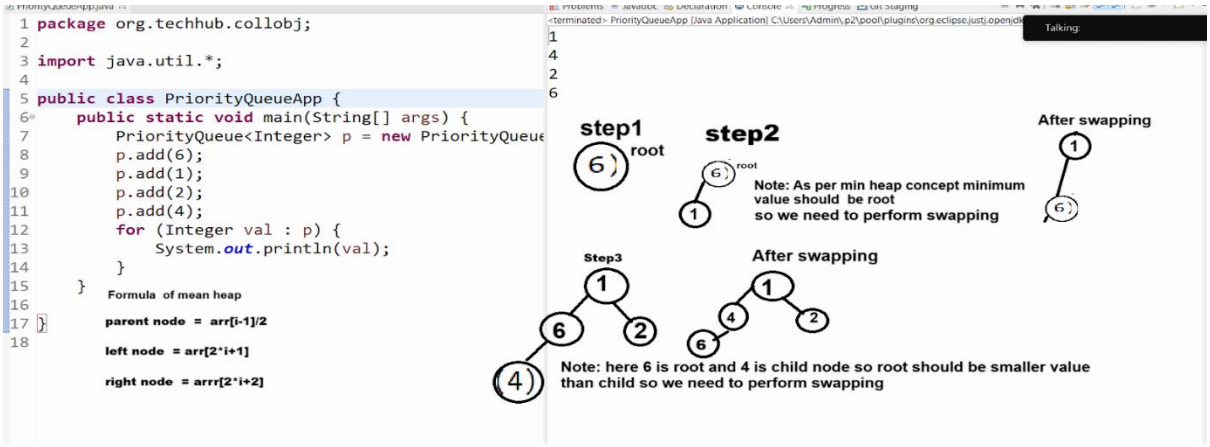


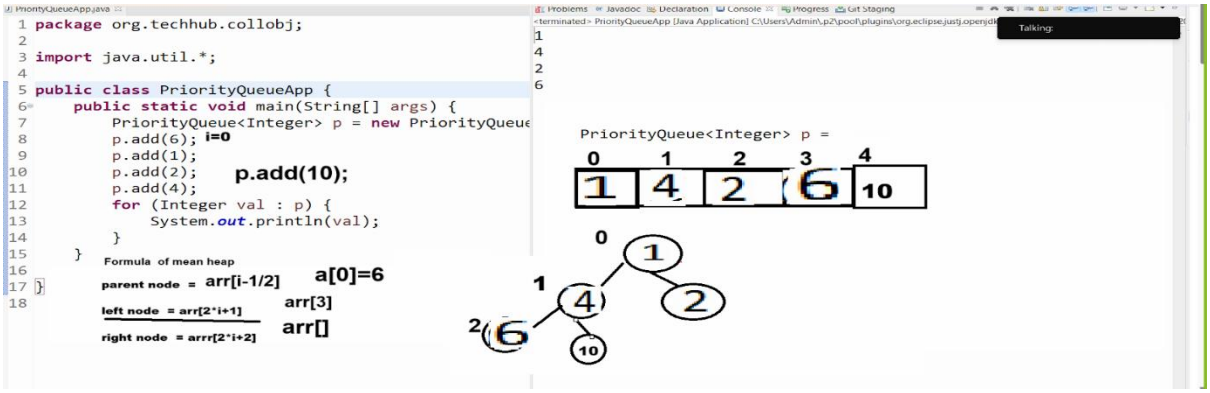
PriorityQueue : PriorityQueue of Java use Min Heap concept or Tree internally

Important points related with priority queue

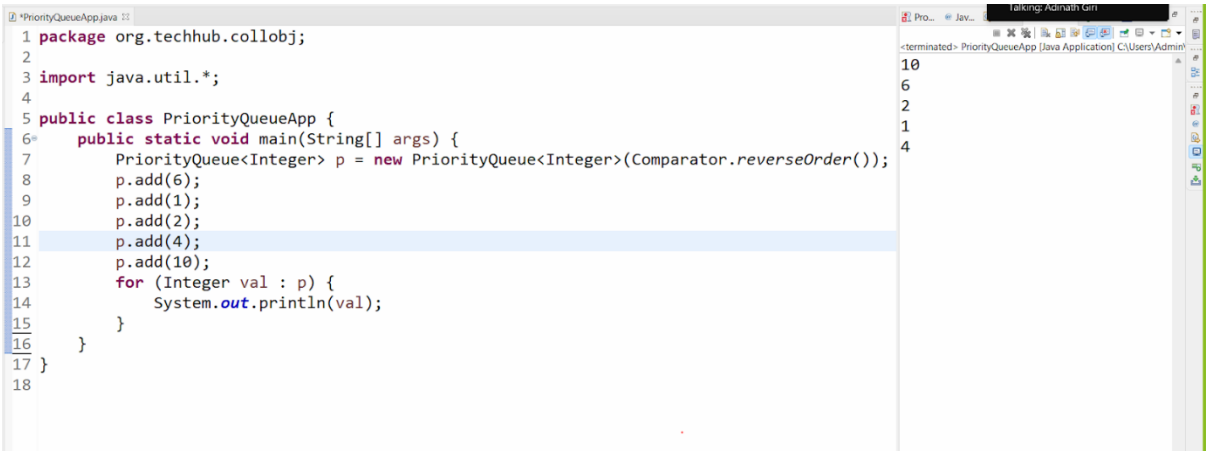
---

1. **Minimum value should be root node**
2. **Add new node from left hand side**

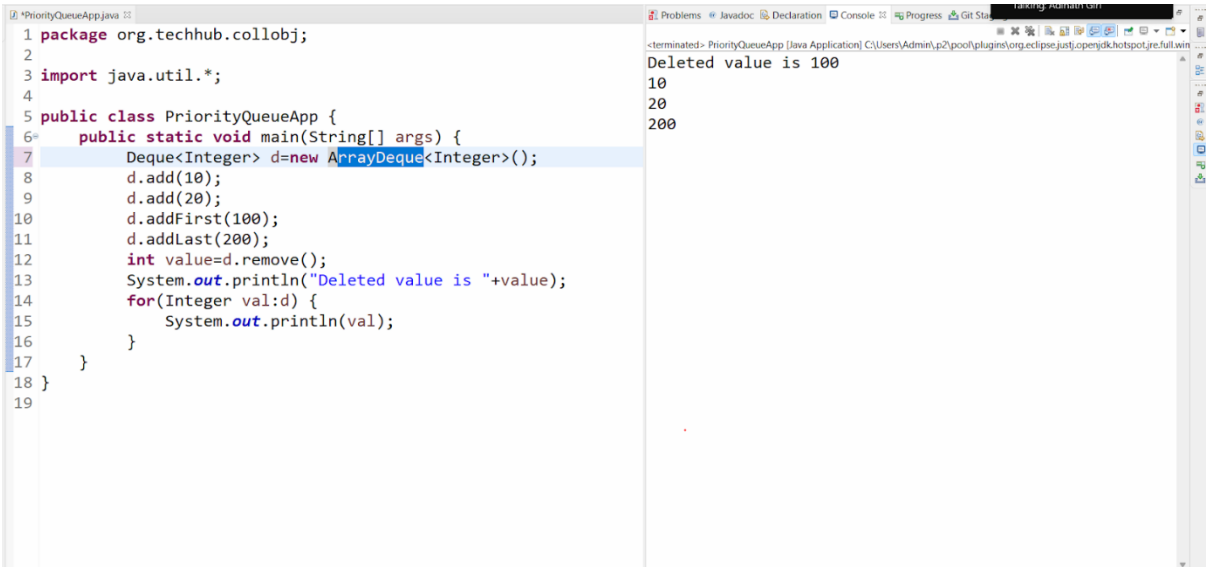




If we want to organize your priority Queue as Max Heap concept then we have to use Comparator interface and its reverseOrder() method means your maximum node should be root



DQueue : DQueue Stands for Double Ended Queue means we can insert or remove node from front as well as rear



```java
package org.techhub.collobj;

import java.util.*;

public class PriorityQueueApp {
    public static void main(String[] args) {
        Deque<Integer> d=new ArrayDeque<Integer>();
        d.add(10);
        d.add(20);
        d.addFirst(100);
        d.addLast(200);
        int value=d.remove();
        System.out.println("Deleted value is "+value);
        for(Integer val:d) {
            System.out.println(val);
        }
    }
}

```

```
Deleted value is 100
10
20
200
```

File Handling

IOStreams

Q. What are streams?

When we transfer data from one location to another location called streams.

Means we can send data from one computer to another computer in network or send data from one memory to another memory called as streams

If we think about iostream we can store our java program output in file permanently or read file from hard disk to our java program

If we want to work with streaming in JAVA we have one inbuilt package name as java.io

And this package provide some classes and interfaces also to work with files

When we want to send data to some location or get data from some location we need to know the path

If we want to work with path we have File class provided by java.io package

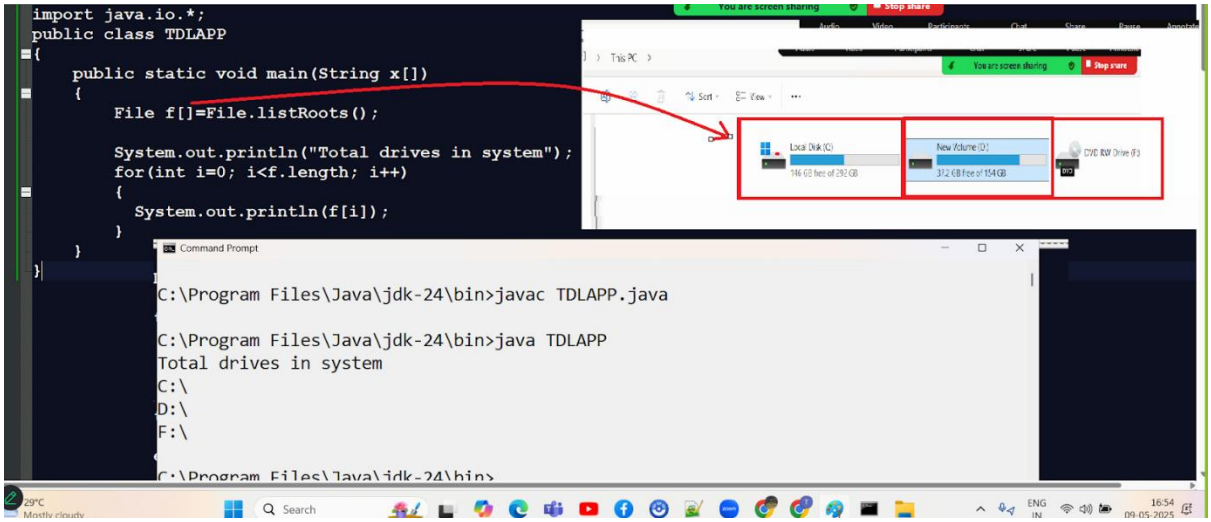
File class from java.io package

File class is used for access the drive list from computer, create folder , create file , delete folder , delete files, access the total space drive, free space of drive, list of files and list folders from specified path

How to access drive list or partition list from hard disk

If we want to check the number of partitions of your hard disk or number of drives in a hard disk then we have the listRoots() method of File class.

Syntax: static File [] listRoots(): this method helps us to return all drive lists from your hard disk.

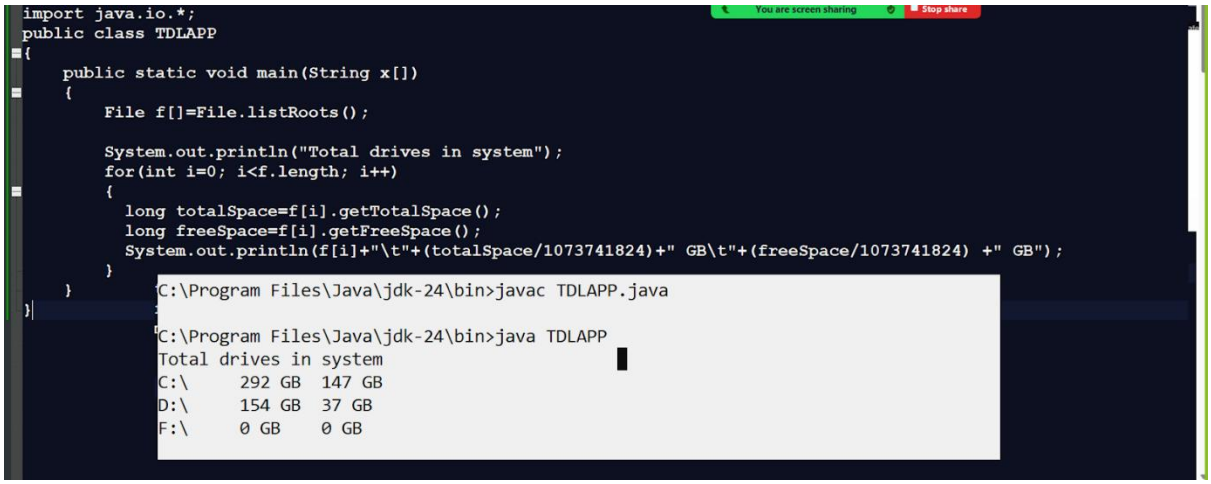


Access total space of drive and free space of drive

---

long getTotalSpace(): this method help us to return the total space of drive in bits format

long getFreeSpace(): this method help us to return the free space of drive in bits format

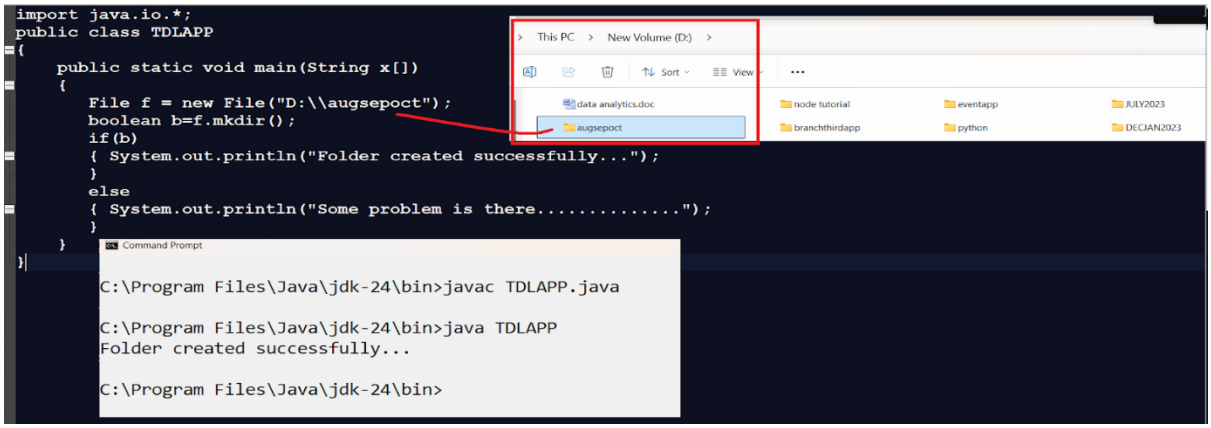


How to create a folder using file class?

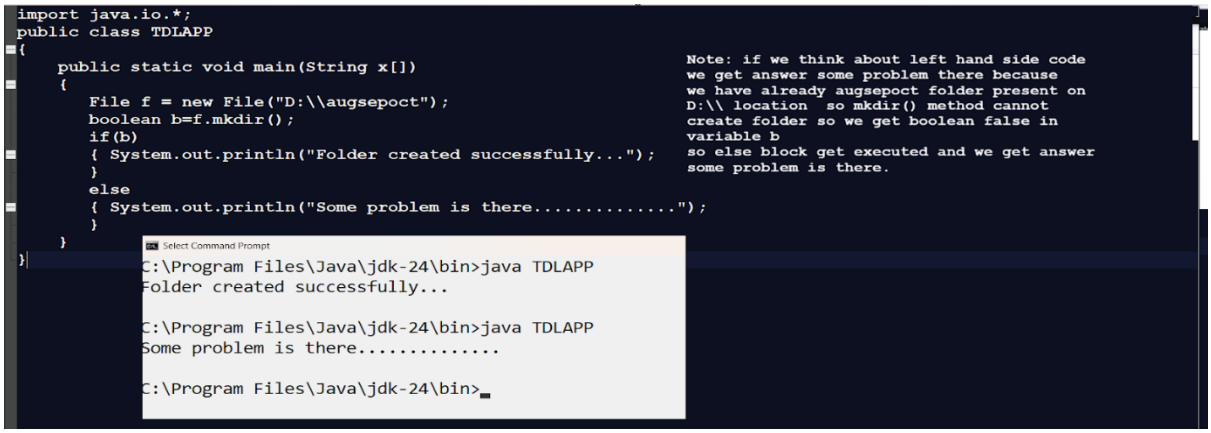
If we want to create folder using a File class we have method

boolean mkdir(): this method is used for creating a folder using File class and if the folder created successfully returns true otherwise returns false.

Example: we want to create folder on D drive name as augsepoct



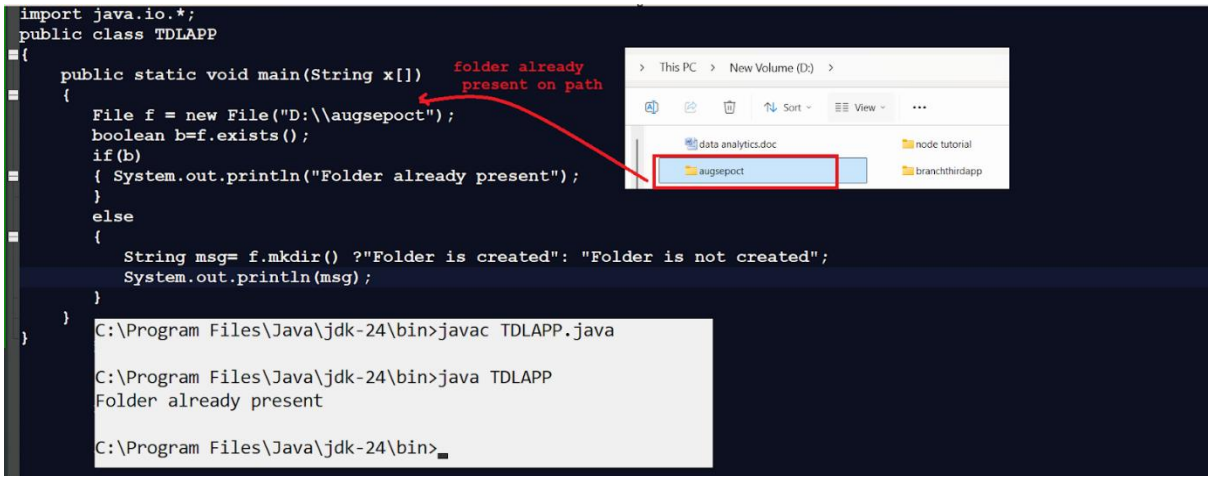
Now we want to execute same program and check the output



How to check folder or file exist on specified path

---

boolean exists(): this method can check if a folder or file is present or not on a specified path if present return true otherwise return false.

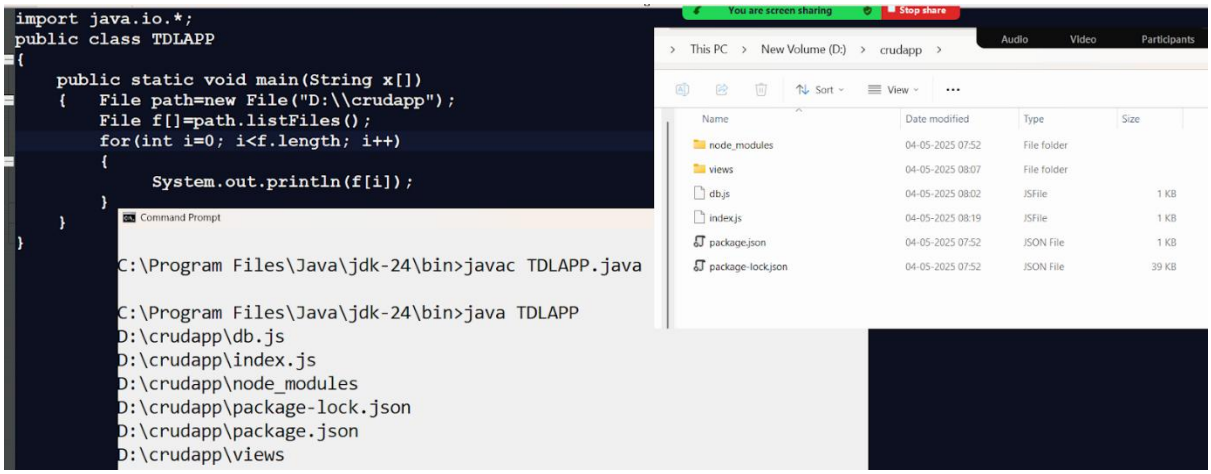


How to fetch all files and folder from particular path

If we want to fetch all files and folder from particular path we have method name as

listFiles()

Syntax: File ref[] listFiles(): this method is used for fetching all files and folders using some specified path.



If we think about above output we fetch all files and folders but we want to fetch only Files or folder from particular path.

boolean isFile(): this method checks if the path is file or not if file return return otherwise return false.

boolean isDirectory() : this method checks if the path is a folder or not if the path is a folder return true otherwise return false.

Example: Fetch only files from specified path

```
import java.io.*;
public class TDLAPP
{
    public static void main(String x[])
    {   File path=new File("D:\\crudapp");
        File f[]=path.listFiles();
        for(int i=0; i<f.length; i++)
        {
            if(f[i].isFile())
            {
               System.out.println(f[i]);
            }
        }
    }
}
```

```
C:\Program Files\Java\jdk-24\bin>javac TDLAPP.java

C:\Program Files\Java\jdk-24\bin>java TDLAPP
D:\crudapp\db.js
D:\crudapp\index.js
D:\crudapp\package-lock.json
D:\crudapp\package.json

C:\Program Files\Java\jdk-24\bin>
```

Example: Fetch only folder from specified path

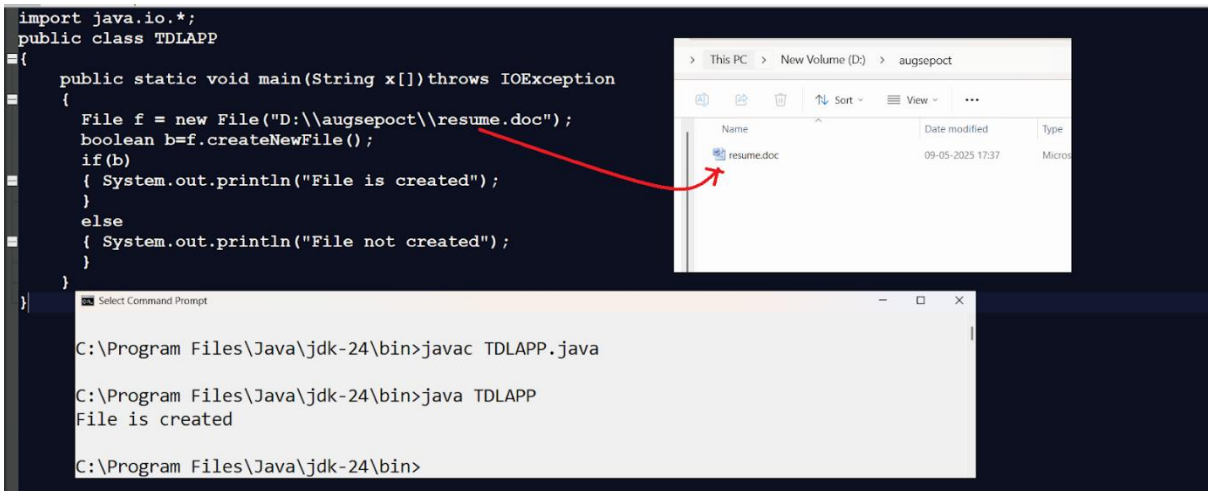
```
import java.io.*;
public class TDLAPP
{
    public static void main(String x[])
    {   File path=new File("D:\\crudapp");
        File f[]=path.listFiles();
        for(int i=0; i<f.length; i++)
        {
            if(f[i].isDirectory())
            {
               System.out.println(f[i]);
            }
        }
    }
}
```

```
C:\Program Files\Java\jdk-24\bin>javac TDLAPP.java

C:\Program Files\Java\jdk-24\bin>java TDLAPP
D:\crudapp\node_modules
D:\crudapp\views

C:\Program Files\Java\jdk-24\bin>
```

How to create file using a File class

If we want to create file using a File class we have method name as

boolean createNewFile(): this method can create a file and file is created return true otherwise return false.



Once we create file we can store data in file by using java program or read data from file using java program

How to store data in file using JAVA Program

If we want to store data in a file using Java Program we have two ways.

1. **Using Text format :** if we want to work with document file, text file etc format then we can store data in file using text format
2. **Using Byte Format :** if we want to work on image , audio , video file then we can use byte format.

How to work with Text format

If we want to work with text format in JAVA we have Reader and Writer classes

Means those classes name ends with Reader or Writer they support to text format

How to work with byte format

If we want to work with byte format in JAVA we have stream classes

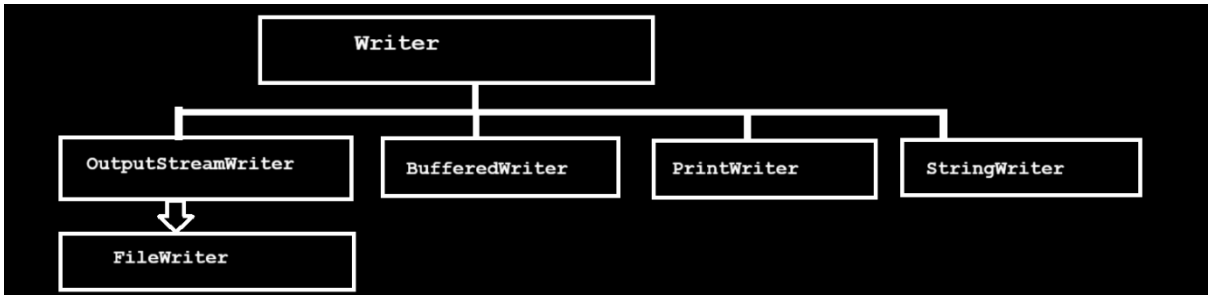
Means those classes ends with stream word they support to byte format

Like as FileOutputStream,FileInputStream etc

Now we want to discuss about the Writer classes

Writer classes help us to write data in file using a text format.

If we want to work with Writer classes we have following class Hierarchy



If we think about the above diagram, the Writer class is parent of all text data writing classes and it is an abstract class which contains some abstract methods which help us to write data in file and this file contains write() method and it is an overloaded method which helps us to write data in file.

Methods of Writer class

---

void write(char): this method can write single character at time in file

void write(char[]): this method can write character array in file

void write(String): this method can write string data in file

void write(char[],int offset, int length): this method can write specified length of data from character array in file

char[]: this parameter contain actual data

Int offset: this parameter decide from which index data want to write

Int length: this parameter decide how much data want to write in file'

void write(String,int offset,int length): this method can write specified length of data from string in file

String : this parameter contain actual data

Int offset: this parameter decide from which index data want to write

Int length: this parameter decide how much data want to write in file'

void close(): this method can close the file

FileWriter class

FileWriter class is used for write textual format data in file or text formatted data in file

Constructor of FileWriter class

---

1) **FileWriter(String path): this constructor accept the file path using string format and use write mode for writing data by default**

When we write file then we have two types of mode

1. Write Mode : write mode means if file is not exist then create new file and if file is exist then override previous data of file and add new data in file

2. Append Mode : Append mode means if file not exist then create new file add data in it and if file is exist then persist or save previous data of file and add new data at end of file called as append mode.

2. **FileWriter(File path): this constructor accept file path using File class reference and use write mode for writing data in file**

3. **FileWriter(String path, boolean mode):**

Parameter details

String path: this parameter is used for accept file path using a string format

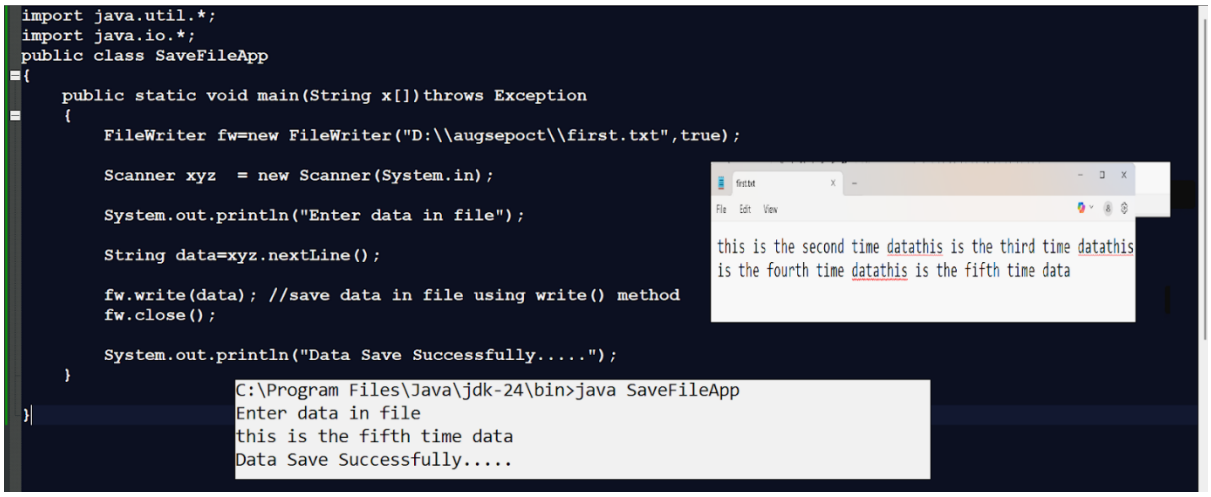
boolean mode: this parameter indicate file can use write mode or append mode means if we pass true then file can use append mode and if we pass false then file can use write mode.

4. **FileWriter(File path,boolean mode):**

Parameter details

File path: this parameter accept file path using file class reference
boolean mode: this parameter decide file can use write mode or append mode if we pass true then file can use append mode and if we pass false then file can use write mode.

Example: we want to create a file and store text data in it.



```
import java.util.*;
import java.io.*;
public class SaveFileApp
{
    public static void main(String x[])throws Exception
    {
        FileWriter fw=new FileWriter("D:\\augsepoct\\first.txt",true);

        Scanner xyz  = new Scanner(System.in);

        System.out.println("Enter data in file");

        String data=xyz.nextLine();

        fw.write(data); //save data in file using write() method
        fw.close();

        System.out.println("Data Save Successfully.....");
    }
}
```

```
C:\Program Files\Java\jdk-24\bin>java SaveFileApp
Enter data in file
this is the fifth time data
Data Save Successfully.....
```

```
first.txt
File  Edit  View
this is the second time datathis is the third time datathis
is the fourth time datathis is the fifth time data
```

Note: if we think about above code we write data using FileWriter class in file with append mode.

When we add data in file then new data added only on same line

Means using FileWriter class we cannot write data on new line in file

So if we want to write data on new line in file we have BufferedWriter class

Now we want to discuss about BufferedWriter class

Constructor of BufferedWriter class

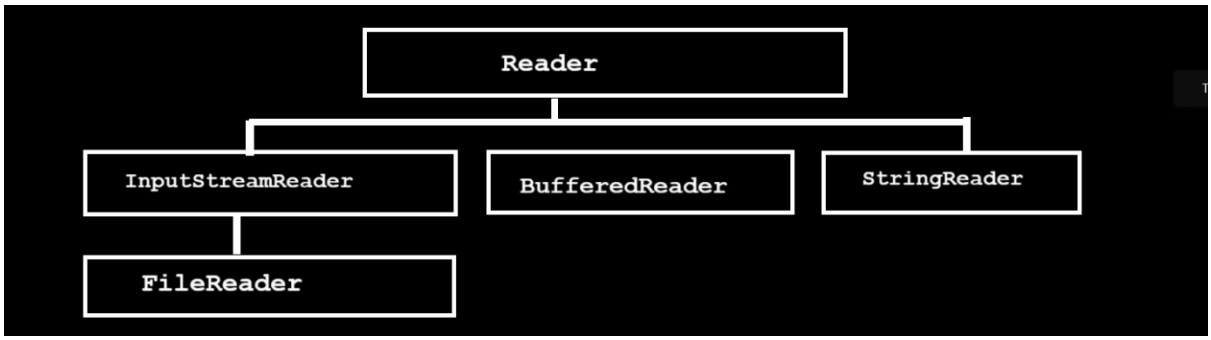
```
BufferedWriter(Writer); if we think about BufferedWriter constructor
it contain reference of Writer class and Writer is parent of all Writing
classes means we can pass reference of any child of Writer class in
BufferedWriter constructor
```

Note: BufferedWriter class contains newLine() method which is used for writing data on new lines in a file.

```java
import java.util.*;
import java.io.*;
public class SaveFileApp
{
    public static void main(String x[])throws Exception
    {
        FileWriter fw=new FileWriter("D:\\augsepoct\\third.txt",true);

        BufferedWriter bw = new BufferedWriter(fw);

        Scanner xyz  = new Scanner(System.in);

        System.out.println("Enter data in file");

        String data=xyz.nextLine();

        bw.write(data); //save data in file using write() method
        bw.newLine();

         bw.close();

        fw.close();

        System.out.println("Data Save Successfully.....");
    }

}
```

How to read textual file data

If we want to read file data using textual format we have following class hierarchy



Reader class

Reader class is abstract class and it contain some abstract methods which help us to read data from file

int read(): this method is used for reading a single single character and returning ascii code from file and return -1 when file is finished or when file has no data.

int read(char[]): this method is used for read file data and store in character array and return its length and file has no data or end then return -1

int read(char[],int offset,int length): this method can read specified length of file data from file.

Parameter details

---

char [ ]: this parameter contain actual data

int offset/index: this parameter decide from which index we want to read data from file and

index start from 0th location in file

int length: this parameter decides how much data you want to read from the file.

void mark(int index): set pointer from data want to read from file or set index location from data want to read from file

FileReader class

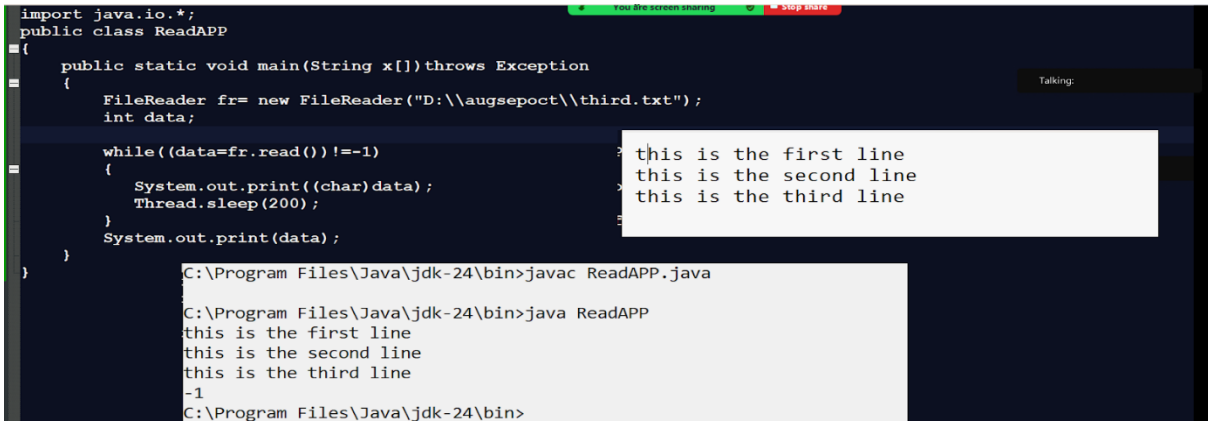
---

constructor of FileReader class

FileReader(String path): accept the file path using string format and use read mode by default

FileReader(File path): accept the file path using file class reference and use read mode by default.

Example using FileReader with read() method



Example using read(char[]);

```
import java.io.*;
public class ReadAPP
{  public static void main(String x[])throws Exception
        {   File f=new File("D:\\augsepoct\\third.txt");
                FileReader fr= new FileReader(f);
                int data;
                char ch[]=new char[(int)f.length()];
                 fr.read(ch);
                 for(char c:ch)
                 {  System.out.print(c);
                 }
        }
}
```

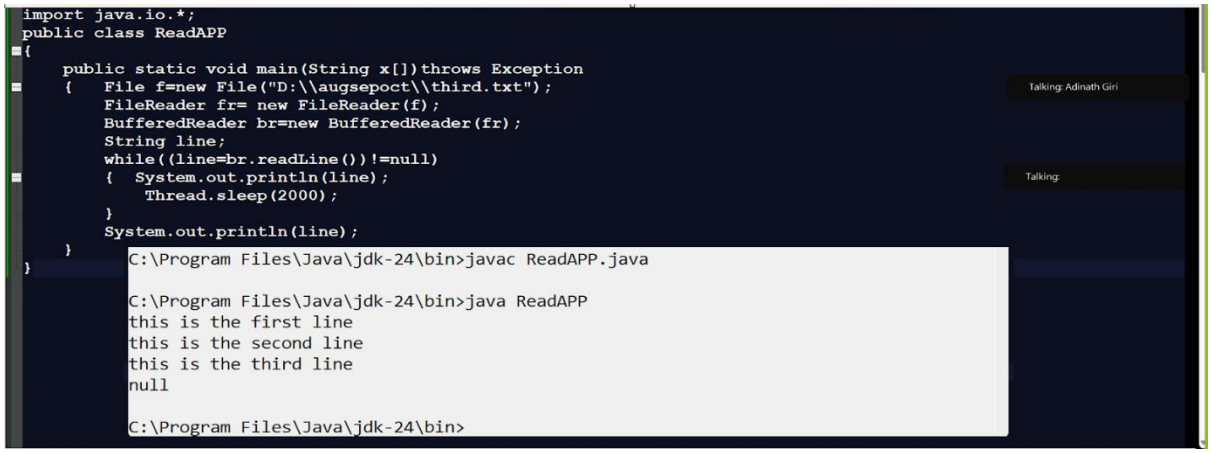
Note: if we use the FileReader class then we can read single single character data from file

But if we want to read line by line data from a file we have to use the BufferedReader class.

BufferedReader class uses readLine() method which is used for reading single single line data from file and returning null when file is finished.

Constructor of BufferedReader

BufferedReader(Reader): we can pass reference of any child of Reader or Reader itself



Stream classes

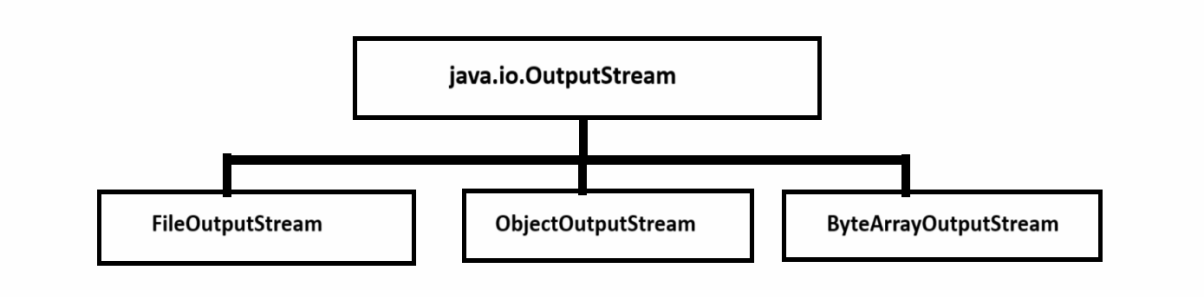
Stream classes are used for writing data in file using byte format.

If we want to work with Stream classes we have two types of stream class

OutputStream : OutputStream is a class which is parent of all stream classes and which is used for writing data in file.

InputStream : InputStream is a class which is parent of all stream classes and which is used for read byte format data from file.

Now we want to discuss about OutputStream class



If we think about the above diagram we have OutputStream is a parent of all stream classes and it contains some methods which help us to write data in file using byte format.

public abstract void write(int) throws java.io.IOException: this function is used for write single byte at time in file

public void write(byte[]) throws java.io.IOException: this function is used for write byte array data in file

public void write(byte[], int offset, int length) throws java.io.IOException: this function for write byte array with some specified size

public void flush() throws java.io.IOException: this function can flush the buffer

public void close() throws java.io.IOException: this function can close the file.

FileOutputStream class

This class is used for writing byte data in a file.

Constructor of FileOutputStream class

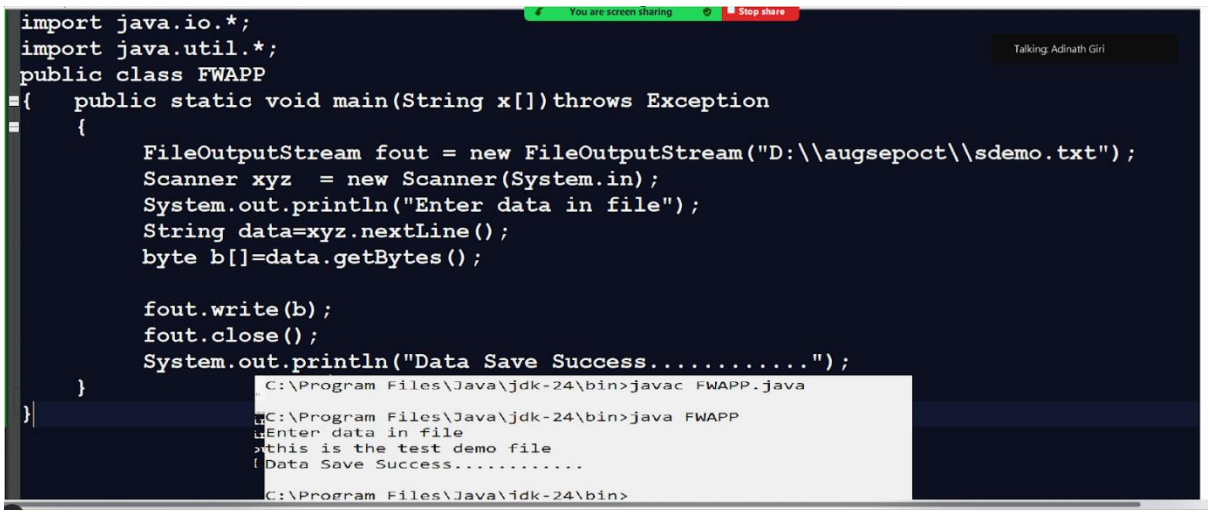
FileOutputStream(String path): this constructor can accept file paths using a string format and by default use write mode for writing data in file

FileOutputStream(File path): this constructor can accept file path using a file class reference for accept file path and use by default write mode for writing data in file

FileOutputStream(String path,boolean mode): accept the file path using string format and if we pass true then file can use append mode and if we pass false then file can use write mode.

FileOutputStream(File path,boolean mode): this constructor accepts file path reference using file class reference and if user passes true then file can append otherwise file can write mode.

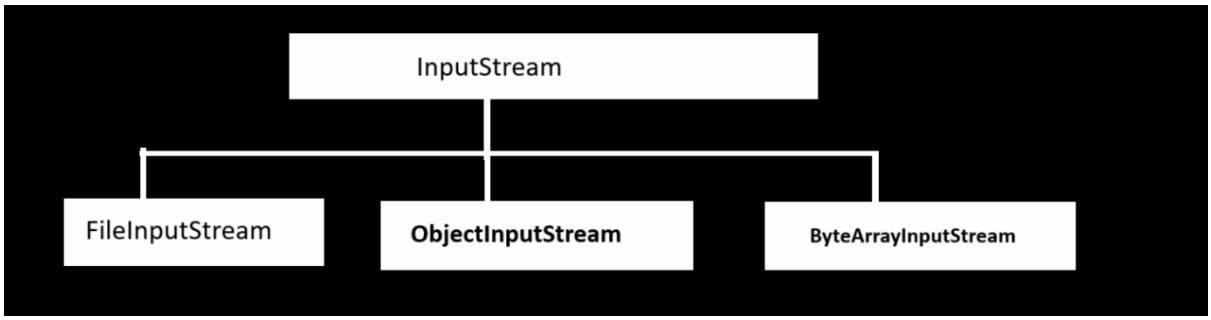
Example: WAP to input string data from keyboard and store in file using byte format.



InputStream classes

InputStream classes are used for reading data from files using byte format.

Hierarchy of InputStream classes



Note: here InputStream is an abstract class and it contains some abstract methods which help us to read data from file using byte format.

int read(): this method is used for reading single single byte data from file and return -1 when file is finished or when file has no data.

int read(byte[]): read complete data from file and store in byte array and return its length and if file has no data or finished then return -1

int read(byte[],int offset,int length): read the specified size of byte data from file

void mark(int index): this method can mark the index location and read data from specified index.

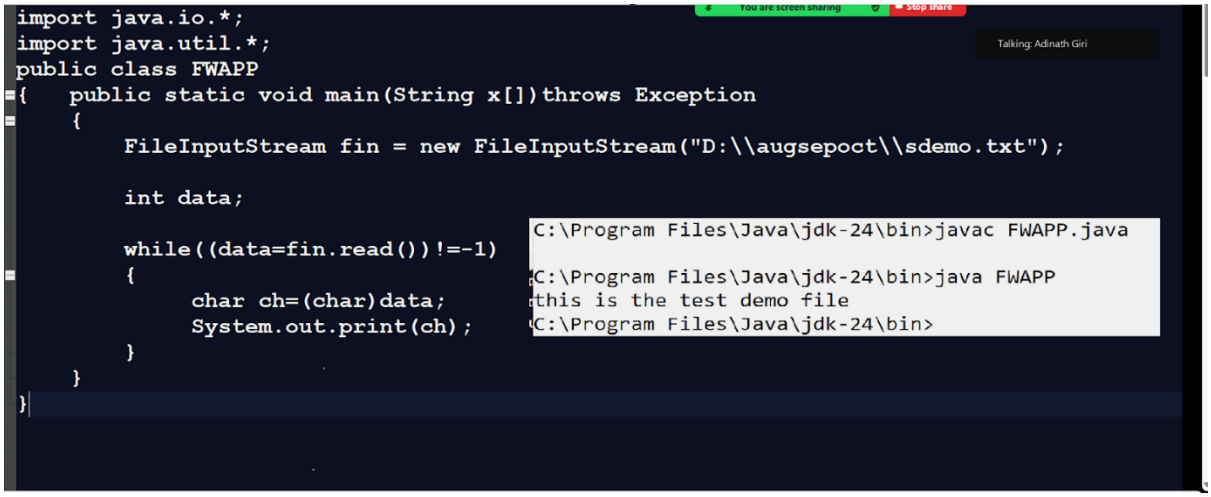
void close(): this method can close the reading object or file.

FileInputStream class

Constructor of FileInputStream class

FileInputStream(String path): this constructor accept file class reference using string format and use read mode by default

FileInputStream(File path): this constructor accepts file path using file class reference and uses read mode by default.



Example: we want to copy the image from particular location and paste on some other location

```
import java.io.*;
import java.util.*;
public class ImgCopyPasteApp
{   public static void main(String x[])throws Exception
        { FileInputStream fr = new FileInputStream("D:\\OCT2024\\er.jpg");
          int data;
          FileOutputStream fw =new FileOutputStream("D:\\augsepoct\\er.jpg");
          while((data=fr.read())!=-1)
          { fw.write(data);
          }
          fw.close();
          fr.close();
          System.out.println("SAVE");
        }
}
```

Serialization and Deserialization

Serialization : Serialization is process to store user define object in file called as serialization

When we store user define object in file then internally JVM encrypt the object content and store in file means when some open the file and check the file content then file content is not human readable format means we provide security to user file

As well as when we share in network from one machine to another machine in the form java object and if we serialize object then data send via network in encrypted format

Steps to work with Serialization

---

1. **Add java.io package in application**

Syntax: import java.io.*;

2. **Create POJO class and implement Serializable interface in it.**

Note: Serializable is marker interface in JAVA

Example with source code

```
import java.io.*;


class Employee implements Serializable
{
   private int id;
        private String name;
        private long sal;


        public void setId(int id)
        { this.id=id;
```

```
    }
    public int getId(){
      return id;
    }
    public void setName(String name)
    { this.name=name;
    }
    public String getName(){
      return name;
    }
    public void setSal(int sal)
    { this.sal=sal;
    }
    public int getSal()
    { return sal;
    }
}
```

3. Create Object of ObjectOutputStream class

Constructor of ObjectOutputStream

Syntax:

ObjectOutputStream(OutputStream): this constructor accepts reference of OutputStream means we can pass reference of any child of OutputStream.

Example

```
FileOutputStream fout=new FileOutputStream("D:\\augsepoct\\ser.txt");
ObjectOutputStream oout=new ObjectOutputStream(fout);
```

4. Call writeObject() method of OutputStream class

Syntax: void writeObject(Object): this method can save object of any class.

Example with source code

```
import java.io.*;


class Employee implements Serializable
{
   private int id;
        private String name;
        private long sal;


        public void setId(int id)
        { this.id=id;
        }
        public int getId(){
          return id;
        }
        public void setName(String name)
        { this.name=name;
        }
        public String getName(){
          return name;
        }
        public void setSal(int sal)
        { this.sal=sal;
        }
        public long getSal()
        { return sal;
        }
}
```

```
public class ObjSerApp
{
   public static void main(String x[])throws Exception
        {
                FileOutputStream fout=new FileOutputStream("D:\\augsepoct\\ser.txt");
                ObjectOutputStream oout=new ObjectOutputStream(fout);

                Employee emp = new Employee();
                emp.setId(1);
                emp.setName("ABC");
                emp.setSal(100000);

                oout.writeObject(emp); //write employee object in file and encrypt its data
                oout.close();
                System.out.println("Save");
        }
}
```

Deserialization

Deserialization means reading object data from a file called deserialization.

If we want to read object data from file we have class name as

ObjectInputStream and its one method name as readObject()

Constructor

ObjectInputStream(InputStream): this constructor accepts file path using inputstream reference.

```
import java.io.*;
class Employee implements Serializable
```

```
{          private int id;
           private String name;
           private long sal;
           public void setId(int id)
           { this.id=id;
           }
           public int getId(){
            return id;
           }
           public void setName(String name)
           { this.name=name;
           }
           public String getName(){
            return name;
           }
           public void setSal(int sal)
           { this.sal=sal;
           }
           public long getSal()
           { return sal;
           }
}
public class ObjDSerApp
{
   public static void main(String x[])throws Exception
           {         FileInputStream fin=new FileInputStream("D:\\augsepoct\\ser.txt");
                      ObjectInputStream ooin=new ObjectInputStream(fin);
                       Object obj=ooin.readObject();
                       if(obj!=null)
```

```
                {Employee emp=(Employee)obj;
                 System.out.println(emp.getId()+"\t"+emp.getName()+"\t"+emp.getSal());
                }
                else{
                        System.out.println("No Data");
                }
        }
}
```

transient: if we want to avoid some data store in file in serialization process then we can use the transient non access specifier with variable

.