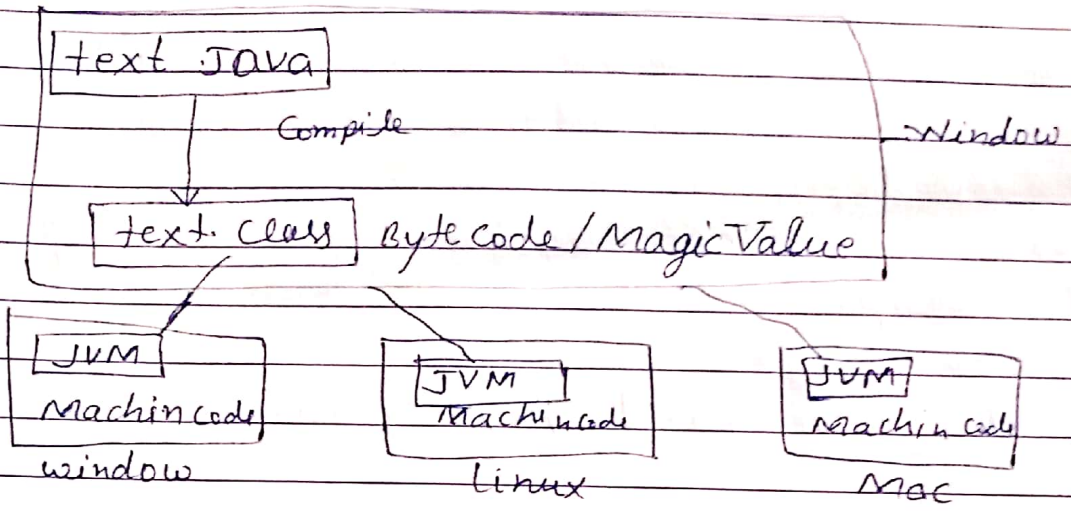
* Platindependent :-



When we compile any Java program then generate byte code (magic value) in one os and this bytecode can be run any other os with JVM.

JVM understand byte code then produce machine code based on current os
WORA:- Write one Run Anywhere

* Architecture Nuetral :-

Our Java program will not be effected when try change any hardware Configration.

* OOP :-

JAVA is a object oriented programing language in this case we break our complex application into class and object.

CLS — JAVA 1.0 — CL2

* History of JAVA:-

⇒ JAVA developed by James Gosling (1994) and team
⇒ James Gosling considends as father of JAVA.
⇒ JAVA developed under Sun MicroSystem.
⇒ But Now Sun micro System is a part of oracle Corp.

* How to run first JAVA program:-

⇒ install JdK
⇒ Copy path till bin
(C:\Program files (x86)\JAVA\Jdk1.8.0_251\bin)
⇒ Open run window and cmd the click on OK.
⇒ type cd and paste, copied path with space
⇒ type notepad with file Name
for ex:

notepad Lab.java

Write code inside Lab.java file

```
Class Lab
 {
   public static void main (String args[])
    {
     System.out.println("Welcome");
    }
 }
```

* How To compile:-

Javac fileName
for ex:- javac Lab.java
Java fileName

How to Run

Java className
for ex:- Java Lab

* Data type and Variable

```
class Lab2
{
  public static void main(String args[])
  {
    System.out.println(
      byte a = 10;
      short b = 1234;
      int c = 12345;
      long d = 123456,7;

      float e = 12.23f;
      char y = 'z';
    System.out.println(a);
      System.out.println(b);
      System.out.println(c);
      system.out.println(e);
      System.out.println(y);
  }
}
```

* Type of variable :-

=) If we declared any variable inside a class without using static keyword known as instance variable. diffult value 0

⇒ if we declare any variable inside class with static keyword known as static variable.
diffult value 0

=) If we declare any variable inside members of a class known as local variable diffult value is depend on user

eg

```
class Lab
{
  int a= 10;  ---> instance
  static int b = 20; ---> static
  System.out.println (String args[])
  {
    int c =30;  ---> local
  }
}
```

⇒ We can not access instance variable inside static context.

forcg

```
class Lab3
{
  int a = 10;
  static b = 20;
public static void main (String args[])
```

```
s
    int c = 30;
// System.out.println(a); ⇒ Error
   System.out.println(b);
    System.out.println(c);
  }
}
```

* Zero is a default value of instance and static variable.

*⇒ developer are responsible to initialized local variable before using.

⇒ priority of local variable is always high.

* Class:-

⇒ class is a logical discription.

⇒ class is a blue print.

⇒ in Java we can create class by using 'class' keyword.

* Object:-

⇒ object is a physical representation of class

⇒ object has state, behaviour and identity.

⇒ object is a runtime entity.

* How to create object in Java:-

className var = new className()

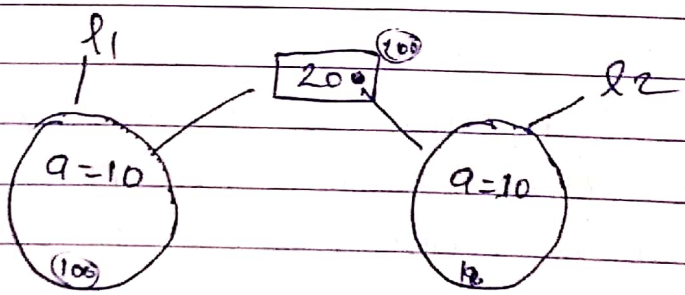
for ex:

Lab l = new Lab();

new Keyword is used to create object and return its address.

eg

```
class Lab;
 {
 int a=10;
 static int b=20;
public static void main(String args[])
 {
   Lab l1 = new Lab();
   Lab l2 = new Lab();
System.out.println(l1.a); //10
 System.out.println(l1.b); //20
     l1.a =100;
     l2.b = 200;
 System.out.println(l2.a); //10
 System.out.println(l2.b); //200
 }
}
```



How many way to access instances variable.

```
Class Lab
 {
  int a=10;
```

```
public static void main (String args[])
  {
    Lab l = new Lab();
    System.out.println (l.a); //10
    System.out.println (new Lab().a); //10
    //System.out.println (Lab.a);

    Lab l1 = null;
    //System.out.println (l1.a);
  }
}
```

How may way to access static variable:

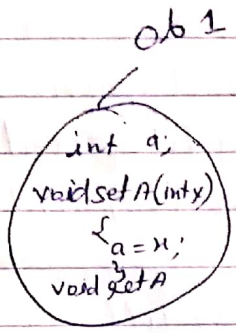
```
class Lab
  {
    static int = 10;
public static void main (String args[])
  {
    Lab l = new lab();
    System.out.println(l.a); //10
    System.out.println (new Lab().a); //10
    System.out.println (Lab.a); //10

    Lab l1 = null;
    System.out.println (l1.a); //10
  }
}
```

* Method in JAVA

method discript funcilaty of an object

```
class MyMethod
 {
  int a;
  void setA (int x)
    {
      a=x;
     }
     void getA()
     {
System.out.println(a);
  }
System public static void main(String args[])
     {
       MyMethod obj = new MyMethod();
         obj.setA(10);
          obj.getA();
MyMethod obj2 = new MyMethod();
  obj2.setA(100);
   obj2.getA();
  }
}
```



Sum of two no. using method in JAVA

```
class A
 {
  int a, b, c;
   void setValue(int x, int y)
     {
       a=x;
       b=y;
     }
```

```
int sumValue()
    {
      c = a+b;
      return c;
    }
  }
class SumLab
 public static void main (String args[])
   {
     A ob = new A();
     ob.setValue (10, 20);
int   t = ob.sumValue();
   System.out.println (t);
    }
}
```

* Constructor

⇒ Name of constructor is always same as class name.

⇒ Constructor automatically called at the time of Object Creation.

for ex.

```
class Lab
  {
    Lab()
     {
       System.out.println ("Constructor...");
     }
public static void main (String args[])
    {
```

Constructor have no returntype value

```
Lab L1 = new Lab();
         new Lab();
  }
}
```

⇒ Constructor have no return type even void but if we specify return type with constructor it considered as normal method.

gmf (*****)

⇒ Constructor is special type of method which is used to intialized of an object.

Type of Constructor

(1) Deffult constructor :-

⇒ When Developer not inserted any constructor ineside Java program then Java compiler insested on constructor known as default constructor जब इस किसी भी प्रकार के constructor Java में ~~कि~~ ~~कोई~~

(2) Non parameteries Constructor :-
Which have no agrament. as Lab()

⇒ A constructor which have no parameter.

eg
```
class Lab
{
    Lab()
    {
        System.out.println("Constructor");
    }
    public static void main(String args[])
```

```
{
 Lab l = new Lab();
}
}
```

Parameturies Constructor.

```
class Student
  {
   int sid;
   String sname;
   Student (int id, String name)
    {
     sid = sid;
     sname = name;
    }
   void showValue
    {
   System.out.println("Student ID:"+sid);
   System.out.println(Student Name:"+Sname);
    }
  }
class Lab
  {
   public static void main (String args[])
    {
   Student st1 = new Student(101,"Raj");
    st1.showValue();
  }
}
```

* Copy Constructor

```
class Student
  {
   int id;
   String sname
```

```
Student (int id, String name)
    {
     sid = id;
     sname = name;
    }
    Student (Student tobj)
       {
        sid = tobj.sid;
        sname = tobj.name;
       }
    void showvalue()
     {
      System.out.println("Student ID:"+sid);
      System.out.println("Student Name:"+name);
     }
  }
  class Lab
   {
  public static void main (String args[])
     {
      Student st1 = new student (101,"Raj");
        st1.showValue();
     }
  }
```

gmp

This Key word :-

This Keyword points the current object different

eg

```
class Lab
  {
```

```
int a=100
void show()
{
  int a=10;
System.out.println(a); //10
System.out.println(this.a);//1000
}
public static void main (String args[])
{
  Lab l=new Lab();
    l.show();
}
}
```

Eg

```
class Lab
{
  int a;
    lab (int a)
      {
      this.a=a;
      }
    void show()
      {
      System.out.println(a);
      }
  public static void main(String args[])
    {
      Lab l = new Lab (10);
        l.show();
    }
}
```

Method overloading

Creating a multiple method with same name but which have different number of argument or different type of argument.

for ex:-

```
Class Sum
  {
    void sumValue (int a, int b)
    {
     System.out.println (a+b);
    }
     void sumValue (int a, int b, int c)
    {
     System.out.println (a+b+c);
    }
  }
Class Lab
  {
    public static void main (String args[])
    {
     Sum obj = new Sum();
      obj.sumValue (10, 20);
      Obj. sumValue (10, 20, 30);
    }
  }
```

eg

```
class Sum
  {
    void sumValue (int a, int b)
    {
```

```
System.out.println(a+b);
  }
  void sumValue(float a, float b)
    {
      System.out.println(a+b);
    }
}
class Lab
    {
public static void main(String args[])
  {
    Sum obj = new Sum();
       obj.sumValue(10,20)
       obj.sumValue(30.20);
  }
}
```

* Inheritence :-

⇒ Creating a new class by using the property of existing class, such type of concept known as inheritence.

⇒ Newly created class known as child / sub / Derived class

⇒ Existing class known as parents / super / Base class

⇒ If child class wats to accessed the property of super class then child class must used 'extends' keyword.

⇒ Advantage :- reusibility

Syntax:-

```
class Parents
  {
  // body of parents class
  }
  class child extends parents
```

for example ①

```
class Person
 {
   int id;
   String name;
   void setId (int id)
    {
     this.id = id;
    }
    void getId ()
     {
    System.out.println ("ID = " + id);
     }
     void setName (String name);
      {
        this.name = name;
       }
       void getName ()
        {
System.document.out.println ("Name = " + name);
        }
  }
```

② class Teacher extends Person

```
    {
    }
```

class Student entends ~~Student~~ person

```
    { }
```

```
class Lab
  { public static void main (String args[])
    {
    Student st = new Student ();
      st.setId (101);
```

```
st.getName("Ankit");
 st.getId();
 st.getName();

Teacher t = new Teacher();
      t.setId(1001);
      t.getName("ALOK");
       t.setId();
       t.getName();
    }
  }
```

example ②

```
class A
    {
    void show()
        {
System.out.println("Show() in class A");
        }
    }
class B extends A
    {
    void display()
    {
System.out.println("display() in class B");
        }
    }
class Lab
    {
    public static void main(String args[])
    {
```

```
A ob1 = new A ();
    ob1. show ();

B ob2 = new B ();
    ob2 = display ();
        }
        }
```

In above example.

Case 1 -

```
A ob1 = new A ();
    ob1 .showA ();
```

By using parent object we can say parent specific member but it we try to call child specific members it way produce error.

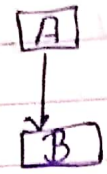
Case 2

```
B ob2 = new B ();
    ob2.show ();
    ob2.display ();
```

By using child object we can say child specific member and also parents specific members

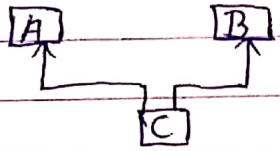
Type of inheritence

1. Single inheritence



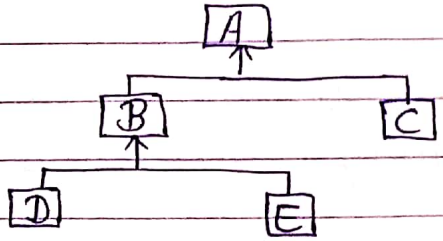
② Multilevel inheritence —

③ Multiple inheritence —

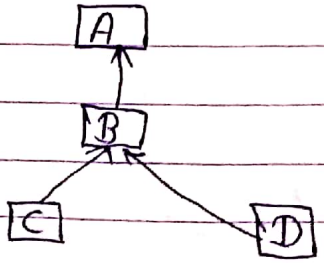


(Not support the JAVA)

④ Hierarchal inheritence —



⑤ My bvid —



Qu Why JAVA doesn't support multiple inheritence?

To avoid ambiguity error java does not support multiple inheritance through class. But through the interface, multiple inheritance is possible in Java. No Java dosen't support multiple inheritance directly because it leads to overiding of methods when both extended class have a same method name

What is JAVA ?

* Simple:-

JAVA is a simple become it not supported complex topic of other language like C & C++

* Secure:-

JAVA program within a JVM which protech unaouthorised access.

* Robust:-

⇒ No pointer support.
⇒ Strong Memory Mng.
⇒ Support Exception Handling.
⇒ Type checking.

Constructor calling inheretence

```
class A
 {
   A()
     {
     System.out.println("A() constructor");
     }
   }
  class B extends A
    {
      B(){
          System.out.println("B() constructor");
        }
      }
     class Lab
        {
         public static void main(String args[])
          {
            B ob = new B();
          }
        }
```

* Method overriding :-

Creating a method inside child class which is alleady available in a super class with a same signature in a super class.

for Example

```
class A
{
 void show(){
  System.out.println("show() in class A");
 }
}
class B  extends A
{   void show(){
      System.out.println("show() in class B");
    }
}
class Lab
 {
  public static void main(String args[])
  {
    A ob = new B();
      ob.show();
  }
 }
```

Compile time polymorphism / early / static

```
child c = new child();
```

In this case method invocation decided at compile time which is based on reference type.

Run time polymorphism / late dynamic / dynamic

```
Parents p = new child();
```

In this case method invocation decided at

Run time which is based on object.

* Array In JAVA.

Array is a object in Java

```
import java.util.*;
class MyArray
 {
    int a[]=new int[5];
 void setValue()
  {
  System.out.println("Enter fiveElement");
 {cannersc = new Scanner (System.in);
  for(int i=0; i<=4; i++)
    {
     a[i]= sc.next();
    }
}
  void getValue():
   {
    System.out.println("Five Element Are");
     for(int i=0; i<=4; i++)
       {
         System.out.println(a[i]);
       }
    }
   }
   class Lab
    {
      public static void main(String args[])
```

length → variable
length(); → function

```
{
MyArray m = new MyArray();
 m.setValue();
 m.getValue();
 }
}
```

* Variable length Argument (Var-Args)

In the case of variable length Argument, function can takes number of argument by using elips (...)
for ex:-

```
class A
{
 void sum(int...n){
    int s=0;
  for(int i=0; i<n.length; i++)
    {
     s=s+n[i];
    }
  System.out.println("Sum"+s);
 }
}
class Lab
{
 public static void main(String args[])
  {
    A ob = new A();
    ob.sum();
    ob.sum(10,20);
    ob.sum(10,20,30);
 }}
```

Package

How to set parament JAVA path

- copy path till bin
  (C:\Program files (x86)\Java\jdk1.8.0_251\bin)
⇒ right click on my computer
⇒ go to properties
⇒ click on advance ~~stu~~ system setting
⇒ click on enviorement variable
⇒ find path variable inside system variable
⇒ select path click on edit button
⇒ paste, copied path but use semicolon begineng of path name.
⇒ OK OK OK

Package

⇒ Package is a workspace where classes, inner classes and enterface etc are available.
⇒ if we want creates package in Java we must used 'Package' Keyword
⇒ package Keyword is always first statement of any Java Program
⇒ by using package keyword we specify folder name (folder structure), where we want to generate our class file.

Note:-
-d specify where to place generated class files

for example

```
package com.asu.pl;
public class Lab
  {
    public static void main(String args[])
      {
        System.out.println("Welcome");
      }
  }
```

* How to compile

Javac -d . filename

for ex:- JAVAC -d . Lab.java

How to Run

Java fully Qualified Name

for ex:- Java com.asu.pl.Lab

* Access Modifiers :-

Which describe scope of a members of a program

* Modifiers :-

Public
Private
protected

* Scope :-

Public
private
Protected
default

public :-

If we declare any class as public then class name must be same as file name.

One program can be contains only one class as public
We can not declared any class as private and protected.
for ex:-
Lab.java
public class Lab { } ⇒ valid
public class Test{ } ⇒ invalid
ex:-

```
package com.asu.P1;
  public Class A
        {
          public void readme()
            {
              Soplan ("Reading.....");
            }
        }
```

B.JAVA

```
Package com.asu.p1;
  emport com.asu.P1*;

      public class B
          {
            public static void main(starting args[])
              {
                A ob = new A();
```

```
    ob.readme();
  }
}
```

C.JAVA

```
package com.asu.p2;
import com.asu.p1*;
    public class c
      {
        psvm (String args[])
          {
            A ob = new A();
            ob.readme();
          }
      }
```

Private :-

① private member can be access in which will decales

for ex:- A.JAVA

```
package com.asu.p1;
    public class A
      {
        private void readme()
          {
            sopln ("Reading...");
          }
      }
```

B.JAVA

Some as previes code.

C.JAVA

some as previes Code

Default:-

① If we declared any member without any access modifiers (Public, Private, Protected) then the scope of member considered as default scope.

② default members can be accessed within a same folder.

A.JAVA

```
package com.asu.p1;
        public class A
            {
              Void readme()
                {
                  sopln("Reading....");
                }
            }
```

B.JAVA

Same as peavies code.

C.JAVA

same as peavies code.

Qu① What is deffrence between default and protacted scope.

A.JAVA

```
package com.asu.p1;
    public class A
    {
      protected void readme()
      {
        SOPln("Reading.....");
      }
    }
```

B.JAVA

same as peavies code

CJAVA

```
package com.P2;
import com.asu.P1;
public class C extends A
{
  public static void main(String args[])
  {
    C ob = new C();
    ob.readme();
  }
}
```

* Abstract Keyword :-

⇒ Abstract is a Keyword which is applicable with class and method.

⇒ Abstract class can not be instantiated (if we decleared any class abstract Keyword)

⇒ If any class don't wants to provide body at its method then method must be declared as abstract.

If any class holds abstract method then class must be declared as abstract.
for ex:-

```
abstract class A {
  void show () {
    System.out.println ("show() in class A");
  }
```

```
abstract display ();
}
class B extends A {
        display () {
    System.out.println ("display () in class B");
    }
}
class Lab {
    public static void main (String args[])
    {
        B ob = new B ();
        ob.show ();
        ob.display ();
    }
}
```

Qu② What is different between Abstract and enterface.?

Interface :

⇒ Interface is a fully abstract.
⇒ with the help of interface keyword we can creats interface in JAVA.
⇒ If we declare any method inside interface then public and astraict inside interface keyword automatically added at the begining of method signature.

for ex:-

```
Interface A {
        void show();
// Public abstract void show()
}
```

⇒ In interface we can not specify abstract keyword monually with any method

⇒ enterface connot be entanticated

Note

Any class can be emplements more then one. enterface.

for ex:-

```
enterface A {
        void show(); // Public abstract void main
}
Class B emplements A {
        public void show()
        {
```

```
        soln (" show() in class B");
    }
}
class Lab
    {
     psvm (String [] args)
      {
       A ab = new B();
          ab.show();
      }
     }
```

ex: -

```
interface A {
    void show(); //public abstract void show();
    }
interface X {
        void display();
        }
class B implements A, X {
        public void show()
          {
          soln ("show() in class B");
          }
    public void display()
      {
      soln("display() in class B");
      }
            }
class Lab
      {
       psvm (String [] args)
         {
```

```
A ab = new B();
   ab = show();
X  ob1 = new B();
   ob1.display();
   }
}
```

for ex

```
interface A {
        void show();
        }
interface X extends A {
                void display();
                }
class B implement X {
        public void show
                {

                Some code

        }

class Lab
    {
      psvm (String [] args)
            {
            X ob1 = new B();
              ob1.display();
              ob1.show();
            }
    }
```

final Keyword

19/08/2019

* String :-

⇒ string is a final class.
⇒ string is immutable class.

⇒ We can creates string object in two way

① With new Keyword
② without new Keyword.

for ex:-

```
class StrLab{

    public static void main(String args[])
    {
    String str1 = "Java-Study";
    String str2 = new String("Java-Study");
    System.out.println(str1);
    System.out.println(str2);
    }
}
```

* Method in string class

eg①

```
class StrLab{
public static void main(String args[])
{
String str = "Java-Study";
```

```
System.out.println(str.length());
System.out.println(str.toUpperCase());
System.out.println(str.toLowerCase());
System.out.println(str.indexOf('a'));
System.out.println(str.lastIndexOf('a'));
}
}
```

eg②

```
class StrLab
{
public static void main(String args[])
{
String str = "Java-Study";
System.out.println(str.startsWith("Java-"));
System.out.println(str.endsWith("Study-"));
System.out.println(str.replace("Java","Python"));

String str2 = "-Java";
System.out.println(str.concat(str2));
}
}
```

* trim() :- trim is use without space

```
class StrLab
{
public static void main(String args[])
{
String str = "Java-Study";

System.out.println("["+str+"]");
System.out.println("["+str+"]");
System.out.println(str.charAt(3));
}
}
```

* Split() :-

```
import java.util.*;
class StrLab
 {
   public static void main(String args[])
    {
 Scanner sc = new Scanner(System.in);
  String str = sc.nextLine();
   String word[] = str.split(" ");
for(int i=0; i<word.length; i++){
    System.out.println(word[i]);
   }
  }
}
```

=> length & length() (method) different

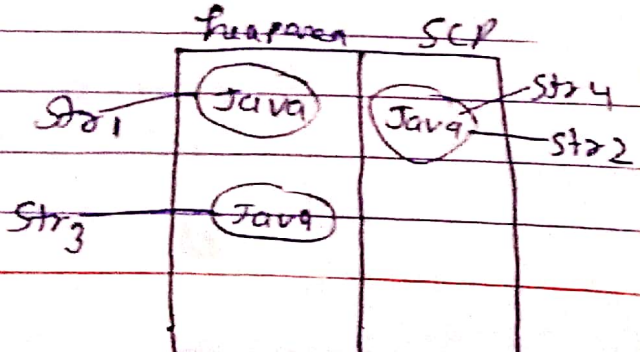
Q. Difference between == and equals() method

eg

```
class StrLab
{
public static void main(String args[])
{
String str1 = new String("Java");
String str2 = "Java";
String str3 = new String("Java");
String str4 = "Java";

System.out.println(str1 == str2);
System.out.println(str2 == str4);
System.out.println(str1.equals(str2));
System.out.println(str2.equals(str4));
}
}
```

two String address to compare



Qu String is a immutable ?

eg

```
class Lab
 {
  public static void main(String[] arg)
    {
     String = "Java";
     str = str + " study";
  System.out.println (str);
    }
 }
```

Reverse String

```
import java.util.*;
 class Lab
  {
   public static void main(String[] arg)
     {
   System.out.println("Enter The String");
  Scanner sc=new Scanner(System.in);
  String str = sc.nextLine();
```

This is a pen
neW a si siht

```
for(i=str.length()-1; i>=0; i--){
    System.out.println(str.charAt(i));
    }
}
```

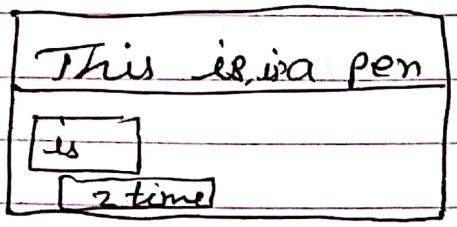
* String change in word by word *

```
import java.util.*;
class Lab
{
public static void main(String args[]
{
System.out.println("Enter The String");
Scanner sc = new Scanner(System.in);
String str = sc.nextLine();
String word[] = str.split(" ");
for(String w:word)
{
for(int i=w.length()-1; i>=0; i--)
System.out.print(w.charAt(i));
System.out.print(" ");
}
}
```

This is a pen

~~Task~~

Qu ①



Exception

Exception is an unexpected event which occure during the execution of program and interrupt the normal flow of a program

Handle the exception known as Exception Handling.

for eg:-

```
import java.util.*;
class ExpDemo
{
public static void main (String args[])
{
System.out.println("Enter The Number");
Scanner sc = new Scanner(System.in);
int data =sc.nextInt();
System.out.println(" Welcome..in);
intdata =sc.nextInt();
System.out.println("Welcome 1");
System.out.println("Welcome 2");
System.out.println(10/data);
System.out.println("Welcome 3);
}
}
```

* Try Keyword:-

Try Keyword used where exception might be arrise

Catch

⇒ we can not used catch without try.
⇒ Catch handels the generated exception and contians Exception handler code.

for ex

```
import java.util.*;
class ExpDemo
 {
  public static void main (String args[])
{
System.out.println("Enter the Number");
Scanner sc = new Scanner (System.in);
int data = sc.nextInt();
System.out.println("Welcome 1");
    try
      {
        system.out.println(10/data);
      }
    catch (ArithmeticException e){
      System.out.println("Exception Handle");
    }
  System.out.println("Welcome 2");
 }
}
```

Q40 NullPointer Exception

Ans if we trying to perform any operation on null we will get NullPointeException

for ex

```
String str1 =null;
System.out.println(str1.length());
```