☆ Access Modifiers :

Private :	Public	Abstract
default	Static	Volatile
Protected	final	transient.

Private : when a class is Private we cannot access it
when a variable is Private it will not be inherited by
the child class. when a method is private it will
not be inherited by the child class.
Private members are not accessiable by the objects
Private class ~~cannot~~ is not accesible outside the
package it is also not accessable by the child class.

Default : when class is having default access modifier it is available within the package, through subclass/childclass. members are not accessable through object

Protected : When a member is protected it is inherited by the child class. we cannot access member through objects we can access class outside the package through child class.

Public : When members are public we can access them using objects. Public members can be inherited by the child class. class is accessible directly or by child class outside the package.

Static : when a class is static all member are static when a variab is static it will get memory when class loaded into memory. static variable shared among the objects. when method is static it get referred when class loaded in the memory. we can access static member using ~~classname~~
Classname. id;
classname. method();

Abstract : When a class is abstract we cannot create it's object when a method is abstract we cannot provide implemention, any class containing single abstract method must be declared as abstract class. abstract class can contain non-abstract meth any class inheriting or extending abstract class must overwrite all of it's abstract method else child class again become abstract class.

Scanned with CamScanner

Abstract and final can't be used together.

DATE
PAGE

Final : When a class is final we cannot creat its subclass When a method is final we cannot overridde it in child class. when variable is final we cannot change its value it works as constant.

Volatile: volatile can be modified by two Thread. volatile is also used to create Thread safe variable.

Transient : when a variable is transient we cannot searlized it

Q → can we apply abstract and final together on class.
A → Abstract is used to provide abstraction Through inheritance When final dosent allow to be inherit. when both apply together that class neither get inherited nor initialized

* Types of Variable:
1. local variable → declared inside any method. They must be initialized to before use.
2. Instance variable → class variable value is not shared among the objects. When class object is created they get memory.
3. Static variable → it will get memory when class loaded in memory. value of variable shared among all objects.

```
Class Area {
        Protected int r, w, h;
        Public void circle (int a) {
              r = a;
            System.out.println(3.14*r*r);
        }
        Public void rect (int a, int b) {
           w = a;
           h = b;
           System.out.println(w*h);
        }
  }
Class AreaDemo {
      Public static void main( String s[]) {
          area obj = new Area();
          obj. cirCle( 10);
          obj. rect(6, 7);
      }
  }
```

H.W
Take user input r, w, h and same code above.

```
import Java.util.Scanner;

Scanner sc = new Scanner(System.in);
  r = sc.nextInt();
```

if a class is static
make all members static.

DATE
PAGE

Static Datamembers :

```
class St {
    Public statice int n; int x;
     Void show() {
       system.out.println (n);
     }
    Public static void sm() {
        System.out.println(x);   → error use n.
    }
}
Class stDemo {
  Public static Void main(String args[]) {
    St.sm();
   system.out.println (St.n);
     St.n = 10;
   System.out.println(St.n);
   St obj = new St();
     obj.n = 1607;
   system.out.println(St.n);
   obj.show();
}
```

Abstract :

```
class Parent {
    void show() {
        system.out.println(" Demo inheritance");
    }
  }
class Child extends Parent {
    void m2() {
        system.out.println(" m2 of child");
    }
}
class DemoInherit {
    Public static void main(string args[]){
        child obj = new child();
        obj.show();
          obj.m2();
    }
  }
```

X

```
Abstract class Area {
    abstract void areaCircle(float);
    abstract void areaRect(float, float);
    abstract void areaTriangle(float, float);
}
```

DATE
PAGE

```
abstract clas Harshal esctends Area {
        Void areaCircle (float r){
                S.O.P (3.14*r*r);
        }
    }
abstract  class  Suryansh  esctends Harshal {
        Void  areaRet ( float l, float b){
                S.O.P ( l*b);
        }
  }
   class yukta estends Surya {
          Void  areaTriangle (float b, float h) {
                S.O.P ( 0.5* b *h);
          }
     }   =>        class Abs Demo

                        {
                        Public static  void main (String s[])
                       {
                          Yuchi obj =  new yucha ( );
                            obj .area circle (7);
                          obj.  arearectangle (10,7);
                          obj.  areatriangle  (7, 13);
                  }
              }
```

★ Final :

```
final class C1
 {
     void demo() {
         s.o.p("Hello world");
     }
 }
class C2 extends C1 {
     void child() {
         s.o.p("Hello from C2");
     }
 }


class finalDemo {
   public static void main(String args[]) {
          C2 obj = new C2();
          obj.demo();
          obj.child();
   }
 }
```

This program will genrati error.