*Whenever we are writing our own classes, we have to provide some information about our class to JVM like whether this class can be accessible from anywhere or not, whether child class creation is possible or not , whether object creation is possible or not, etc.*

*We can specify this information by using appropriate modifier.*

***The only allowed modifiers for Top-level classes/ non nested classes are:-***

1. ***Public***

* *If a class is declared as public , we can* ***access that class from anywhere****. If any class is not public and if using that class in any other class then we will get compile error.*

1. ***Default***

* *If a class is declared as default then we can access that class* ***only in current package****. Which means we cannot access that class from any other package. Thus default access is also known as package level access.*

1. ***Final***

* *It is a modifier applicable for classes, methods, variables.*
* ***(Method)****Whatever methods parent have by default available to child through inheritance. If the child is not satisfied with parent method implementation then child is allowed to redefine that method based on its requirement.*
* *If the parent class method is declared as final then we cannot override that method in child class because its implementation is final.*
* ***(Class) If a class Is declared as final then we cannot extend that*** *class. We cant create child class for that class****.Inheritace is not possible for final classes***
* *Every method present in final class is final by default, But every variable present in a final class need not be final variable.*
* *Major benefit is to provide uniqieness to method /class /variable. To acheieve security and unique implementation is advantage of final*

1. ***Abstract***

* *It is a modifier applicable for classes and method, but not for variables.*
* ***(Method)*** *Eventhough we don’t know about implementation still we can decalre a method with abstract modifier , because abstract methods only declartion is avilable but not implementation. Hence abstract method declaration should end with ;*
* *Child class is responsible to provide implementation for parent class abstract methods.*
* *Thus if any class contains abstract method , then that class should be declared as abstract.*
* *By declaring abstract method in parent class we can provide guidelines to child classes such as which methods compulsorily the child class has to implement.*
* *Abstrat method never talks about implementation, if any modifier talks about implementation then it forms illegal combination with abstract modifiers.*
* *These combinations are not allowed with abstract 🡪 final, native, synchronized, static, private, strictfp.*
* ***(Class)*** *For any java class if we don’t want to create an object because of partial implementation. Such type of class we should declare with abstract modifier. i.e for abstract classes instantiation is not possible****.***
* ***Abstract classes can have concrete methods.***
* *If a class contains atleast one abstract method, then compulsorily we should declare class as abstract or else we will get Compile time Error.*

***Reason:- If a class contains atleast one abstract method then implementation is not complete and hence it is not recommended to create object.To restrict object creation compulsory we should declare class as abstract.***

* *Even though class doent contain any abstract method still we can declare class as abstract if we don’t want object creation. Because abstract class can contain 0 abbstract methods also****. HttpServlet 🡪 abstract class but no abstractmethods.***
* *Every adapter class is recommended to declare as abstract but it doesn’t contain any abstract method****.***
* *If we are extending abstract class then for each and every abstract method of parent class we should provide implementation otherwise we have to declare child class as abstract****.***
* *In this case next level child class is responsible to provide implementation****.***
* *Abstract methods compulsory we need to override in child class to provide implementation****.*** *Whereas we cant override final methods. Hence final abstract combination is illegal combination for methods.*
* *For final classes we cant create child class, whereas abstract class we should creae child class to provide implemenatation. Hence final abstract combination is illegal at class-level.*
* *Abstract class can contain final method, whereas final class cannot contain abstract method.*

1. ***Strictfp***

* *The strictfp modifier in Java ensures* ***consistent*** *and* ***platform-independent floating-point calculations****. It makes sure that* ***float and double operations*** *follow* ***strict rules*** *and give the same result no matter what platform or JVM the code is running on.Applies to*
* *Classes, interfaces, or methods*
* *Not to variables or fields*

***Effect***

* *Forces all float and double operations in the scope to use strict 32- or 64-bit precision, disallowing extended precision on some platforms.*
* *Eliminates small platform-specific variations in rounding or intermediate precision.*

***Use Cases***

* *Financial systems (currency calculations)*
* *Scientific simulations where bit-exact reproducibility matters*
* *Distributed computing where different machines must agree on results*

***But for inner class/nested classed modifiers allowed are:-***

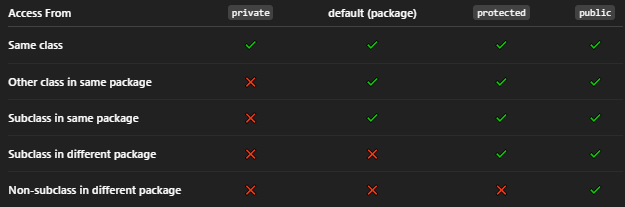
*Above 5 + private, protected, static*

*--------------------------------------------------------------------------------------------------------------------------------------*

***Member-Level Modifiers:-***

***Access Modifiers: Definitions***

* ***private*** *Accessible only within the same class.*
* *(****default****) package-private  
  (no keyword) Accessible to all classes in the same package, but not from outside packages.*
* ***protected***
  + *Accessible to all classes in the same package (like default), and*
  + *Accessible to subclasses (even if they’re in a different package).*
* ***public*** *Accessible from anywhere.*



***Private<default<protected<public***

***Final :-***

1. ***Instance Variables:***

* *If value of variable varies from object to object such object are called instance variables.*
* *For every object a separate copy of instance variables will be created****.***
* *For instance variables we are not required to perform intialization explicitly. JVM will always provide default values.*
* *If instance variable declared as final then compulsory we need to perform initialization explicitly, whther we are using or not and JVM wont provide default values.*

***Rule:-***

*For final variable , compulsory we should perform initialization before constructor completion.*

1. ***Static variables:-***

* *If value of variable is not varied from object to object, such type of variables are not recommended to declare as instance variables.*
* *We need to declare those variables at class-level by using* ***static modifier.***
* *In case of instance variables , for every obejct a separate copy will be created but in case of static variables a single copy will be created at class-level and shared by every object of that class.*
* *For static variables , it is not required to perform initialization explicitly. JVM will always provide default values.*
* *If static variable is declared as final then compulsorily we need to perform initilaization exxplicitly otherwise we will get compile time error.*

***Rule:-***

*For final static variables compulsory we should perform initialization before class loading**completion. 1)while declaration only 2) inside static block.*

*(****Note: For local variables JVM wont provide any default values, need to initialize explicitly***

***Eventhough local variable if final , before using only we need to perform initialization, The only applicable modifier for local variable is final)***

***Static:-***

1. *We can declare static methods, variables, nested inner class but not class(top-level class)*
2. *In case of instance variables a separate copy will be created but in case of static variable a single copy will be created and shared by every object of that class.*
3. *We* ***cannot access instance variables from static areas(methods/blocks) but can access from non-static areas directly****.*
4. *We can access static variables* ***from both static and non-static areas directly.***

***Synchronized:-***

1. *It is modifier applicable for methods and blocks but not for classes and variables.*
2. *Synchronized method should compulsorily contain implementation, whereas astract method doesn’t contain any implementation. Hence abstract and synchronized cant be used together.*

