OOPS Concepts:-

1. Encapsulation :- Wrapping data members & its functions in to a single unit.

Creating a Class with Private variables & these Private Variables can be Accessed using Public setters & getters methods

Setter methods are used to set values for private variables(Class Level) of a class.

Seter method Return Type will be always void.

Getter Method return type will be always the return type of its variable

1. Abstraction :-

Hiding the Implementation of the functionality . It Tells what is does or how it functions & but it dosent tell how it was Implemented.

Ex;- clicking on login button(we don’t know how it is validating the Authentication)

Ex:- clicking on Submit button(we don’t know how all the data is saved & submitted in the backend)

This is Achieved by Interfaces & Abstract Classes.

100% Abstraction can be Achieved with Interface

100% Abstraction cannot be achieved with Abstract Class because it might/can contain defined methods because of which 100% Abstraction is not possible

|  |  |
| --- | --- |
| Abstract Class | Interface |
| It Contains Abstract Methods & Defined Methods | It contains only Abstract Methods & Constants(public static final) |
| Abstract Keyword has to applied at class level & Method Level for Abstract methods | Abstract Keyword is optional at Interface level & method level because JVM will anyways provide . |
| Abstract class can contain Constructor but it will be used by its subclasses | Interface cannot contain Constructor because it will give compilation error if we add Constructor |
| Class & Abstract Class will not participate in Multiple Inheritance | Interface can be used in multiple Inheritance. |
|  |  |
|  |  |
|  |  |

Similarity

object Cannot be created for Both Abstract class & Interface.

Abstract class objects can be created by using its sub classes.

Interface methods are accessed by using its sub class objects.

Inheritance :- Inheritang the Properties and behaviours(variables and methods) from Parent to its children

1. InHeritance :- Inheriting the variables & methods from SuperClass to its SubClasses(Parent to Child)

Private methods & private variables cannot be accessed/Visible in SubClasses.

Inheritance can be achieved through extends keyword between classes(A,B) & Interfaces(I1,I2)

Ex:- Class A extends B

Interface I1 extends I2

Class A can Implement Interface I1

Ex:- Class A Implements I1

Interface I1 cannot Implement or Extend Class A

Interface I1 implements A (Not Possible)

Single Inheritance:-

We can Create objects of SuperClass with SuperClass Reference so that we can access all the Public methods in Super Classs.

We can Create objects of SubClass with SubClass Reference so that we can access all the Public methods in Sub Classs & SuperClass

A a1 = new A();

If we Create objects of SubClass with SuperClass Reference then we can access all the Public methods of SuperClass But we will not be able to access methods from Sub Class.

In Hybrid Inheritance, A,B,C should be Interfaces because if A is Class B & C cannot implement class A.

If a Class A extends ClassB and implements InterfacesI1,I2 we should write the syntax as below:-

Class A extends B implements I1,I2



Polymorphism :-

Same Action Can be performed in multiple ways

We can achieve this feature in java using methods

1. **Compiled time Polymorphism** :- This can be Achieved by Method Over Loading Concept.

Method Over Loading Concept :- It is performed in a Class

Methods Name are Same but Signature should be different.

Method Signature:- Number of arguments, data types of Arguments, order of the Arguments

If two methods have same name with same signature, Access Specifiers & return types doesn’t make a difference.

1. **Run time Polymorphism**:- This can be Achieved by Method Over Riding Concept.

Method Over Riding:- It can be Achieved using Inheritance concept

Methods Name are Same & Signature are also Same . Along with these method return types & Access Specifiers should be also the same.

Constructor:-

Constructor is used for Allocating Memory of a Object for the Class.

Rules for Constructor:-

1. Constructor name should be exactly same as ClassName(CaseSensitive)
2. Constructor will not have return type(like void, int, String)

Default Constructor( 0 Arguments Constructor) is optional which is provided by JVM

Class can Have Default Constructor & OverLoaded Constructor.

For OverLoaded Constructor Signature should be different.( Number of arguments, data types of Arguments, order of the Arguments)

Static :-

Static is a Keyword which is used at Variable level & method level but not at the Class Level. Which means class can contain static variables Static methods & static blocks also.

JVM Allocates memory for static variables Static methods & static blocks at compile time so that we can access these without creating objects(using Class name, we can access these.)

If we try to access these with objects name it does not give any compile time or run time error. It just gives Warning.

Static methods cannot be Overridden in its subclasses because the methods are at class level but they can be accessed in Subclasses.

if we override superclass static method in subclass then, methods will be treated as normal methods of their own respective classes.

if we have a static method in super class & same method is added as instance method in a sub class it will give an compilation error & viceversa.

A Class can contain instance variables, static variables ,constants, instance methods, static methods, constructors, static blocks & instance blocks.

If a Class contain all the above details then it will be executed in the following order:-

1. All the Static Blocks in the order
2. When we create a object of the class all the instance blocks in the order are executed first
3. If constructor is overridden in class that will be executed.
4. Methods will be executed if they are called.

Final is a Keyword which is used at Variable level & method level & Class Level . In a Class if a variable is declared as Final the value will not be changed once it is assigned.

The order is also preserved.

Final Methods & cannot be Overridden in its subclasses but they can be accessed in Subclasses.

Final Classes cannot extend the class.

**Access Specifiers:-** Private , Default , Protected, Public

Access Specifiers are used at variable level, methods level, Class/Interface level & constructor level

If we don’t mention the access specifier it will be treated as **default** access specifier.

Note:- we cannot mention the access Specifier as default. It will give compilation error.

Private:- private variables are accessed in any of the methods of the class. But these are not accessed out of the class eventough classs is inherited.

Default:- Default variables are package level variables which means if a class has default variable, it can be accessed in all the classes of its package but It will not be accessed in subpackages.

Protected:- if a variable is created as protected it can be accessed anywhere in package , other packages & also in its sub packages too.

Note:- if we want to access protected variable in a class in sub packages, then Subpackage class must extend class in the Parent package.

Example:-

package com.a

class A1 protected x1;

package com.a.b

class B1 extends A1 protected x1 is accessable.

Public:- we can access public variables in any class of any package through out the application without inheritance by using the class object.

ClassA m1 to m5

Class B m4, m5, m6

Class C m2,m10

Class D

B o1 = new C

B o2 = new A

I m1 to m10

A m1 to m10 with dummy methods

C

C extends A

A Implements Interface

class A{

A(){

Syso DC

}

main(){

new A();

}

**Conditional Clauses:-**

If() {

}

If Clause will have a Syntax as above. It has condition between () which means block of statement/s between{} will be executed if the condition is true.

If {} - Incorrect Syntax

If (){} – Correct Syntax

IfElse :-

Else Block is followed by if Condition. There should not be any statement between if block & else block.

Else block will be executed when condition in if clause is false.

If(condition){

]

else{

}

NestedifElse:-

The syntax will be as follows:-

If(condition1){

Statement 1

}

elseif(condition2){

Statement 2

}

.

.

.

else{

}

the last else block is optional in the above sysntax.

It checks the conditions in sequence from top to bottom till the condition is true. If none of the conditions are true else block will be executed.