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



In above Multiple Inheritance Diagram, Below Combinations can be possible:-

if A,B,C are Classes then it will not be possible.

if A,B,C are Interfaces then it is possible.

IF A,B are interfaces & C is a Class then it is possible.

Ex:- Class C implements A, B also

A or B can be a class too ex:-

C extends A implements B

In Above Hybrid diagram below are possible combinations:-

if

A,B,C,D are all Classes then HYbrid Inheritance not possible.

Also B, C,D are Classes or B, C are Classes then HYbrid Inheritance not possible.

If D is a Class then A,B,C are Interfaces

ex:- D implements B,C where B extends A & C extends A Indirectly.

If All A,B,C, D are Interfaces then

D extends B,C where B extends A & C extends A Indirectly

Also Dis a Class & B or C can be a class Then A should be a Interface.

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.

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

Final Classes cannot extend the class.( Final Classes will not have child classes but it might have parent 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();

}

Wrapper Classes :- Wrapper Classes are introduced in java 1.5 . These are used in Collections mostly because Collections Doesn’t Support Primitive Datatypes(int ,boolean,float,long)

We can Convert data from primitive to its corresponding wrapper class type which is called Boxing or AutoBoxing.

we can create wrapper class datat in 3 ways:-

1) Integer i = 5;

2) Integer i1 = new Integer(10);

3) Integer i2 = Integer.valueOf(10 );

We can Convert data from wrapper class type to its primitive type which is called Unboxing

Integer i = 5;

int i2 = i.intValue();

int i3 = i;

Unary & Terminary Operators:-

Unary Operators:-

We have Pre& Post Increments and Pre & Post Decrement Operators in java.

int a;

a++ increases a value to 1 as post increment

which means int a = 10;

int b = a++;

here b value will be 10 but a value will be incremented to 1 on next usage;

syso(b) gives 10 &

syso(a) gives 11

++a increases a value to 1 as pre increment

which means int a = 10;

int b = ++a;

here b value will be 11 & a value will also be incremented to 11

syso(b) gives 11 &

syso(a) gives 11

for more information please refer to below link:-

<http://www.javawithus.com/tutorial/increment-and-decrement-operators>

Terniary Operator:-

if we want to represent if else condition in a single line we use ternirry operator(?:)

syntax:- ( < condition>) ? (statments if true) : (statments if false);

comparing the above ternary with with ifElse:-

If(condition){

statments if true

}

else{

statments if false

}

Is-A Has-A Uses-A relationships :-

Is - A relationship is represented by Inheritance(Single).

Ex:- Class B is-A Child of Class A

In Java Every Classs is-A Child of with java.lang.Object

Has -A relationship:-

if a Class is used as a varaible at Class Level in another class in another class

Class A

Class B{

A a1 = new A();

B Has-A relationship with class A.

Uses-A relationship:-

If a class is accessed(using object of the class) in a method of another class

Class A

Class B {

public void m1(){

A o1 = 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

If-else :-

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

}

else if(condition2){

Statement 2

}

.

else if(condition3){

Statement 3

}

.

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.

Loops:- We will use Loops if we want to execute some set of instructions/statement repeatedly based on some condition

1) While Loop: -

in while loop statement is executed until the condition is false

syntax:

< initialization of variable>

while(<condition>){

list of statements to be executed

increment or decrement counter

list of statements to be executed

}

2)Do While loop:-In Do while, the loop will be executed once without checking the condition, the condition will be checked at the end of first execution. It means loop will be repeated only if the condition is true. Also Do while should be terminated with SemiColumn(;)

syntax:-

< initialization of variable>

do{

list of statements to be executed

increment or decrement counter

list of statements to be executed

} while(<condition>)**;**

3) For Loop(formal):- for is another representation of while.

In formal for loop Initialization, Condition & Increment or Decrement will be present in Single Line.

syntax:-

for(initilization; <condition>; increment/decrement){

statements to be executed;

}

or

initilization

for (; <condition>; ){

statements to be executed;

i++

}

for loop execution:-

step 1 :- variable is initialized

step 2 :- condition is checked

Step 3 :- if condition is true, statements will be executed

Step 4 :- After loop is executed, the value(loop counter) will be incremented/decremented

Step 5:- In step 4 variable will hold the increment/decrement value. with this value, step 2,3& 4 are repeated untill step 2 fails.

forEach Loop:-

For Each Loop is used for executing array or collection of values

syntax:-

int a[] = {1,2,3,4,5};

for(int i: a){

syso(i);

}

for (int index =0; index<a.length; index++){

syso (a[index]);

In the above syntax i represents each element in the array a;

Note:- For Each loop is executed based on element of array where as For loop is executed based on Index of array

ex:- **for** ( **int** i : a) {

**if** (i % 10 == x) {

In the above syntax i represents each element in the array a;

Exception Handling:-

In Java, Exception is the super class for compile time & run time exceptions.

Exception is the subclass of Throwable class.

We have 2 types of exceptions

1) java.lang.Error

2) java.lang.Exception

1) java.lang.Error:- we(Program) cannot handle these errrors.

2) java.lang.Exception :- It is categorized in to 2 types.

a) Compile time Errors b) Run time Errors

a) Compile time Errors ( Java.lang.Exception & its subclasses except java.lang.RuntimeException and its Hierrarchy)

these are also called checked exceptions.

In Java we have to handle these exceptions at compile time.

b) Run time Errors (java.lang.RuntimeException and its Hierrarchy)

these are also callled as UnChecked Exceptions.

Note:- RuntimeException is the subclass of Exception.

We Handle the above both compile & runtime Exceptions using try, catch, finally Blocks and we use throw & throws

in try Block, we keep the problematic code & handle the exception using catch block.

if we write try block, it has to be followed by catch block.

Finally Block is the mandatory execution block. this block will be executed irrespective of exception occurred or not.

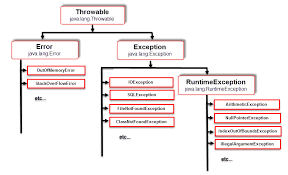
If exception occurs then Control comes to Catch Block & Finally Block will be executed.

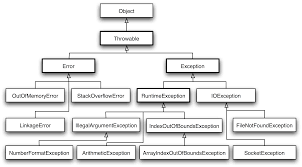
If exception Dosen't Occurs then Try & Finally Block will be executed.

Finally Block will be generally used for closing the resources, Database Connections, closing the file

Also Finally Should be Followed by Catch Block.







Throw & Throws are Keywords which is used for Throwing or rethrowing the exception from a method.

syntax:- throw new <exception class name>( );

throw is generally used inside a if condition of a method

throws is used for rethrowing the exception at method header level

syntax:- public void m1() throws < Exception class name> {

}

1)

public void m1(String s1) **throws Exception** {

if (!s1.equals("String"))

{

//throw new Exception();

**throw new Exception**("Both Strings are not equal");

//syso()

}

syso()

}

Note:- throw statement should not be followed by any statement.

2) public void m2(String s1) {

if (!s1.equals("String"))

{

//throw new RuntimeException();

**throw new RuntimeException**("Both Strings are not equal");

}

In above example 1 & 2 , m1 is throwing(code under the if block) a checked exception which we need to handle at compilation time.so we can rethrow the exception using throws keyword or we will handle the exception where m1 is called using try catch block.

m2 is throwing a RuntimeException & handling the runtime exceptions are optional.

**try with multiple Catch Blocks:-**

A try block can be followed with multiple catch blocks because the code inside the try block may throw different type of Exceptions.

if we write multiple catch blocks & if there is a parent child relation between the exception classes child catch block should come first.

if there is no parent child relationship we can write the catch block in any order.

if we write a parent exception in first Catch block, handling the child exceptions are not required.

because the child exceptions are already handled in prior Parent Class Exception.

In All the cases, Only 1 Catch Block will be executed

Throw is used inside the method where throws is used in the signature of the method.

User Defined Exception:-

It is a Normal Class & it should have behavior like below:-

Public class AException extends Exception{

}// It is a Checked Exception

Public class BException extends RuntimeException{

}//It is a Runtime Exception

**Marker Interface:-** Marker Interface is an Empty Interface. It means this interface does not have any variables or methods.

https://www.geeksforgeeks.org/marker-interface-java/

If any Class Implementing this marker Interface, then the Class will get some special behaviour.

Ex:- java.io. Serialiazable

java.lang.Cloneable

Class C1 Implements Serializable{

}

java.io. Serialiazable :- It gives the class(C1) object to be transferred across the network

Class C2 Implements Cloneable{

}

java.lang.Cloneable:- with Cloneable, we can create the duplicate of the Object in Class(c2).

Singleton:- It is a Design pattern to solve common problems & it is used for creating only one object for a class per JVM.

we can achieve this using Static Factory method & a Private Constructor.

Factory method is used to provide objects in a class. In java, we have both static & non static methods.

https://www.journaldev.com/1377/java-singleton-design-pattern-best-practices-examples

**String Immutability:-**

In java, **string objects are immutable**. Immutable simply means unmodifiable or unchangeable.

Once string object is created its data or state can't be changed but a new string object is created.

https://www.javatpoint.com/immutable-string

To overcome this String Immutability, Java Introduced String Buffer & String Builder.

**String Buffer :-** All the methods in this Class are Synchronized. i.e thread safe, it means two threads(2 flows) cannot call the methods of this class simultaneously. Because of this Synchronus feature it is little bit slow when compared to String Builder.

**StringBuilder :-** All the methods in this class are Not Synchronized. i.e Not Thread safe.

https://www.javatpoint.com/difference-between-stringbuffer-and-stringbuilder