- ## java oop concept explain basics

without oop concept there have to codie lotof coding in a single file but using oop we can reduce the maintain cost, save

time so using objects we can devide the program into sub objects and finaly can be used in the main program.So therefore we can add new functions

for the system easily.

class is like a blueprint and objects are like the houses of this blueprint.

Car car1=new car()

car.color="red"

car.model="abcc"

System.out.println()

(inside the System class there is a variable called out and inside the out variable there is a method called println())

Person.exit() //call the method exit() inside the Person class.

- ## Constructors

```
class Cat{

    int num;

    int num1;


    public Cat(){

    }//this is default constructor it should not to be implement
```

```
        public Cat(){
        num=5;
        num1=7;
        System.out.println("abccccc");

        public Cat(int x){
        num=x;
        }
}

class Car{

        Cat c1=new Car();
        System.out.println(c1.num);
}
//output is
abccccc
5
//constructors do not have an return type even void it starts when the program starting.
```

*Method Overloading

```
class Cat{

method1(5);
method1(5,2);
}
```

void method1(int x){

}

void method1(int x,int y){

}

method overloading is creating two or many methods with the same name in one class by changing the method signature.

overloaded method can have different return types and they should have different parameters.

public static void method(){

System.out.println();

}

public static String method(){

return "hii";

}

//above method are not overloaded because when call the method it can be identified which should be called.

- **Inheritence**

A------->B

this means A class is inherited by the B class

A is sub class and B is super class

class B{

name;

age;

method1{

}

method2{

}

}

class A extends B{

color;

method3{

}

method4{

}

}

//class A have the all the parameters and methods of the B and if we dont want to inherit a parameter or method we have to use the private keyword.

A b= new A();

when we create a object callinng the constructor of the B class All clasess are loaded which have relations with the B classes.all methods and parameters

include in this two classes are loaded into the object.

- ## Access Modifiers

public = everyone can visible.

protected = tamantai, subclass(package or other package), package

default = tamanta, package

private = tamanta

if class or variables are protected they can be arranged by the same class,other subclasses of same package or different packages and also

visible for the existing packages.

- ## Overriding

This is the method which change the method body of a subclass which inherited from a super class.

ex->

```
class Monkey{
void climb(){
//khkhkhk
}
}

class A extends Monkey{
void climb(){
//hjkhkjhkj
}
}
```

Method overriding and overloading are called polymorphism.

- ## Super Keyword

Call the Super class constructor from the Sub class.

Subclass inherits all the *members* (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

```
Class Monkey{
        Monkey(){
        //ghghghgghghgh
        }
}

Class Man extends Monkey{
        Man(){
                Super();
        }
```

Man a=new Man ()

When we create an object using Man sub class using Man () constructor subclass constructer called.

If we want to call the super class constructor from the sub class we call the super () keyword inside the subclass

Constructor. As an example in Monkey class contractor have some features which should include in the Man constructor but when inheriting constructors are not inherited. So we have to add super () keyword to call the Monkey () constructor. If there are parameters in the monkey () constructor we should have to pass them.

Call a super class variable or method from the sub class.

If we want to call the Man class climb with all the functions of the Monkey class climb () method. So we want to firstly add the functions of the Monkey class climb () method.

```
Class Monkey {
        Monkey () {
        //ghghghgghghgh
```

```
        }
void climb () {
Ghgjghgjhgjh
}
}

Class Man extends Monkey {
        Man () {
                Super();
        }

Void climb () {
Super.climb ();
Gfghfghfghfghfhg
}
}
```

- **Casting**

```
class A{
}
Class B extends A{
}
```

- **Upcasting**

```
int a=5;
long b= 60;
```

b = a;

We can add a small data value into large data value variable.

A a= new A ();

B b= new B ();

A is super class and B is sub class here,

A a = new B ();

Here sub class object is add to super class variable.

- ## Downcasting

  Super class variable ekakta dapu sub class object ekak newatha sub class variable ekakta gnna method eka.

  A a = new B();

  B b = a; //compile error

  Here 'a' variable have an object of sub class B so we can assign a to a new variable of b.

  But here comes a compiler error because the interpreter only consider the data type of the 'a' it is an object of A so super class variable cannot assign into sub class object.(But a have the value of sub class)

  Here we have to use downcasting.

  B b = (B) a;

- ## Polymorphism

  Class A{

  void print(){

  System.out.println("A");

  }

  }

  Class B extends A{

void print(){

System.out.println("B");

}

}

B b = new B();

b.print() //B

A a=new A();

a.print(); //A

A a1 = new B(); here super class reference point to the sub class object.

a1.print(); //B in compile time there should be a method in super class(A) as print()

that is why a1.print() means compiler consider is there a method in A super class like print() if there is a method compile successfully and runtime run the print() method of the sub class B.

This is called polymorphism it includes upcasting, Inheritance, overriding

Polymorphism is the method of call a sub class method which is overriding from the super class

- ## Abstract classes

Abstract class Vehicle{

Abstract void park();
void method(){
//asasasacc
}

class Car extends Vehicle{

}

Normal method -> can
Object hadana->No
Sub class object->yes
Constructor-> No

Vehicle v= new Vehicle() // cant
(Therefore no constructor)

Vehicle v1 = new Car();
(Can reference a sub class object)

- ## Interface

Interface is used to do multiple inheritance

```
Class A {
}

Interface B{
        Public abstract void method();
        Public static final int a==4;
}

Interface D {
}

Class C extends A implements B,D{
}
B b =new C();
C c = new C();
D d=new C();
```

Like abstract classes interface can't create object. They can only reference to sub class method like abstract classes. So interface can't create constructors.

Can't have normal methods.

Should have final static constant variables.

- ## Uses of an interface

```
Interface CommonPerson{

        Public abstract void getName();
}

Class Person implements Commonperson{

        Private String name;
        String mobile;
        String password;
        Int age;

        Public void getName(){
                return name;
        }
}


Class Temp{

        Void testMethod(CommonPerson p){
                p.getName();
        }

        }
```

- ## Encapsulation

```
        Binding data with methods.
        Class Student
        {

                Private int rollNo;
                Private String name;

                Public void setRollno(int r)
                {
```

```
        Rollno=r;
        }
    }

    Public class EncapsulationDemo
{
    Public static void main(String[] args)
    {
    Student s1=new Student();
```