Day -107

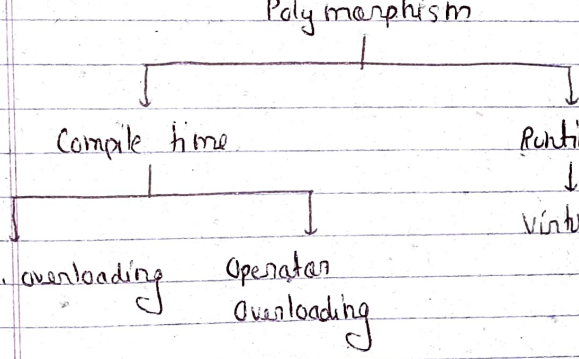
OOPs - 6

* Polymorphism:

=> It means many forms.

=> For ex: A Human can be papa, chacha, bhai etc.

=>



=> Runtime errors are those that comes before compiling the program & errors are missing comma, etc.

=> Compile time errors are like :—

```
int i= 10;   int x= i/0;
                  ↳ This is runtime error.
```

* Function overloading:

=> When we have same name functions, but with diff. parameter.

=>

```
class Area {
    public:
    int calcArea(int r){
        return 3.14*r*r;
    }
```

```
int calcArea(int l, int b){
    return l x b;
}
}
```

=> Here, the name of fun, is same but have diff. parameters
=> So, a same fun. has diff. forms that's why it is polymorphism.

* Operator Overloading:

=> Ex: int a = 5, int b = 10;
cout << a + b;

string str1 = "Rohit";
string str2 = "Negi";
cout << str1 + str2;

=> Here, operation is same but working differently according to the data types.

=> Now, to understand opr. overloading —
Suppose, we want to add —

```
  3 + i2
+ 4 + i6
-------
  7 + i8
```

=> So,

```
class Complex{
    int real, img;
    public:
    Complex(int real, int img){
        this -> real = real;
    }   this -> img = img;
```

```
void display(){
    cout << real << " + i " << img;
}

Complex operator + (Complex &C){
    Complex ans;
    ans.real = real + C.real;
    ans.img = img + C.img;
    return ans;
}
```

* Virtual Function:

```
Class Animal{
    public:
    void speak(){
        cout << "HuHu";
    }
};

class Dog: public Animal{
    public:
    void speak(){
        cout << "Bark";
    }
};

int main(){
    Animal *p;
    p = new Dog();
    p -> speak(); ⟶ It will print HuHu.
}
```

Date : / /
Page:

=> 'HuHu' is printed because 'p' decided to print 'HuHu' at compile time.

=> But we want to print ~~it in~~ 'Bank'.

=> So for this we use virtual keyword.

=> We will write virtual before void in Animal class.

=> Then it will decide at compile time.