1. **OOP (*Object Oriented Programming*)**

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, dimana objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi.

1. ***Encapsulation***

Enkapsulasi (Pembungkusan), merupakan suatu metode “membungkus” *class*, atribut ataupun suatu *method* untuk dapat mengelola hak akses terhadap *class*, atribut, atau *method* tersebut. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci. Pola enkapsulasi paling sering yang digunakan oleh *class* adalah public, atribut adalah private, dan method adalah public. Penggunaan *public* pada *class* dan *method* agar *class* atau *method* tersebut dapat diakses oleh semuanya, sedangkan untuk atribut, biasanya diberi *private* supaya hanya kelas pada atribut itu saja yang dapat mengakses dan mengelola variable tersebut.

1. **Inheritance**

*Inheritance*merupakan konsep pewarisan, dimana sebuah kelas (super class) dapat menurunkan sifatnya (berupa atribut dan *method*) kepada anaknya (*sub class*). Untuk mewarisi sifat yang dimiliki super kelas digunakan keyword “*extends*”, sehingga *sub class* dapat menggunakan atribut ataupun *method*yang dimiliki *super class*, namun *super class* tidak dapat menggunakan atribut dan methode yang dimiliki sub kelasnya.

1. ***Polymorphism***

*Polymorphism* merupakan sebuah konsep di OOP dimana *Class-class*mempunyai fungsionalitas yang berbeda-beda satu sama lain namun disaat bersamaan menerapkan *Interface* yang sama. Pada *polymorphism*, suatu object dapat memiliki berbagai bentuk, sebagai objek dari *class* sendiri atau objek dari *super* *class*-nya. ***Polymorphism*** di implementasikan dalam OOP dengan bantuan fitur *Interface, Abstract Class* dan *Factory Pattern* sebagai pelengkap.