|  |
| --- |
| **Encapsulation** |

|  |
| --- |
| **Encapsulation: Kod və məlumatların xaricdə müəyyən edilmiş digər kodlarla əldə edilməsinə mane olan qoruyucudur. Kod və məlumatları kənar müdaxilədən , həmdə sui istifadədən qoruyur.** OOP-nin ilk prinsipidir.  **Field** – class içərisində yer tutur, class ın özəlliklərini bildirmək üçün istifadə olunur. public string name; public int page;  **Encapsulation giriş gostəricilərindən(access specifiers) istifadə eləməklə həyata kecrilir.**  Enkapsulasiya giriş göstəricilərindən istifadə etməklə həyata keçirilir. Bir giriş göstəricisi bir sinif üzvünün əhatə dairəsini və görünürlüğünü təyin edir.  **1.Public**  **2Private**  **3Protected**  **4Internal**  **5 Protected internal** |

|  |
| --- |
| **Public**-Hər hansı bir dəyişənin digər obyekt və dəyisənlərə daxil olmasina icaze verir |
| **Private-**Bir sinif dəyişənlərinin və funksiyalarinin , digər funksiya və obyektlərdən gizlətməyə imkan verir. Hər hansısa bir metod ve ya class ın içində istifadə edilə bilər. |
| **Protected-**Kicik bir sinifin daha böyük siniflərə daxil olmağa imkan verir.Yalniz icinde olan sinifler ücün əlçatan olur. |
| **Internal**- Assembly içində olan bütün siniflərdə əlçatan olur.  **Protected Internal**-Daxili qorunur |

|  |
| --- |
| **Inheritance** |
| Bir obyekt basqa obyektin xüsusiyyətlərini qazanır.Baza tip üzvlərinin sahələrini və funksiyalarini götürərək baza tipində yaranır. Yaranmiş bir növə funksionallıq əlavə etmək lazım olduqda eynilik ən faydalıdır.  C # -də sahələri və metodları bir sinifdən digərinə miras almaq mümkündür. "Miras anlayışı" nı iki kateqoriyaya ayırırıq:  **Derived Class**- başqa bir sinifdən miras qalan sinif  **Base Class** (parent)- miras qalan sinif  **Value type**-Yaddaş yerində bir məlumat dəyərinə sahib olduqda bir dəyər növüdür.Bütün dəyər növləri SystemValue typdan , öz növbəsində System Objectdən qaynaqlanir.Value tipli dəyisəni bir metoddan digərinə ötürdükdə ,system basqa bir metoddan dəyişənin ayrı bir nüsxəsini yaradir. İki value type bir -birinə bərabər etdikdən sonra, birinin dəyərini dəyisdikdə digərinə adiyyatı olmur.  **Value type tipləri** : bool , byte,char ,decimal,double,enum, float,int,long,sbyte,short,struct,uint,ulong.  **Referance type**-Bir-birinə bərabər etdikdən sonra birini dəyisdikdə digəridə dəyişir daha az yer tutmağı üçün.  **Tipleri:** String  Arrays ,Class  Delegate | | |
| **Polymorphism** -Polimorfizm ümumi bir hərəkət sinfi üçün bir interfeysdən istifadə etməyə imkan verən bir xüsusiyyətdir. Xüsusi fəaliyyət şəraitin dəqiq təbiəti ilə müəyyən edilir.Gördüyü işlər dəyişir amma ozü dəyişmir. Polimorfizm statik (**static**)və dinamik(**dynamic**) ola bilər. Statik polimorfizmdə funksiyaya reaksiya tərtib zamanı təyin olunur. Dinamik polimorfizmdə iş vaxtında qərar verilir.  Statik Polimorfizm  Tərtib zamanı funksiyanın bir obyektlə əlaqələndirilməsi mexanizminə erkən bağlanma deyilir. Buna statik bağlama da deyilir. C # statik polimorfizmi tətbiq etmək üçün iki üsul təqdim edir. Onlar :  Function overloading- İşin həddindən artıq yüklənməsi  Operator overloading- Operatorun həddindən artıq yüklənməsi Dinamik Polimorfizm  C # interfeysin qismən sinif tətbiqini təmin etmək üçün istifadə olunan mücərrəd siniflər yaratmağa imkan verir. Tətbiq olunan bir sinif ondan miras aldıqda tətbiq tamamlanır. Abstrakt siniflər, əldə edilmiş sinif tərəfindən həyata keçirilən mücərrəd metodlardan ibarətdir. Törədilən siniflər daha çox ixtisaslaşmış funksionallığa malikdir.  . Abstrakt bir sinif nümunəsi yarada bilməzsiniz  . Abstrakt bir sinif xaricində mücərrəd bir metod elan edə bilməzsiniz | | |
| **Reusability-**Bir sinif yazıldıqdan və yaradıldıqdan sonra öz proqramlarında istifadə üçün digər proqramçılara paylana bilər. Bununla birlikdə, OOP-də miras, yenidən istifadə edilə bilmə fikrinin vacib bir genişləndirilməsini təmin edir. Bir proqramçı mövcud bir sinifdən istifadə edə bilər və dəyişdirmədən ona əlavə xüsusiyyətlər əlavə edə bilər. | | |