تفاوت پلی مورفیسم Overloadingو Overriding چیست:

ابتدا تعریف پلی مورفیسم :
یکی از مفاهیم اصلی شی گرایی است که موقعیت هایی را توصیف می کند که یک شی در اشکال مختلفی فراخوانده میشود.
در کل از طریق یک رابط به انواع اشیا مختلف دسترسی پیدا کردن.
دو نوع پلی مور فیسم داریم

1- پلی مورفیسم کامپایلی overloading(compile-time -polymorphism)

توابعی با نام یکسان در یک کلاس تعریف می شود اما تعداد یا نوع پارامتر های ورودی متفاوت هستند، درواقع کامپایلیر در هنگام کامپایل بر اساس تعداد ونوع پارامتر ها تصمیم میگیرد که کدام تابع را اجرا کند.مانند کد های زیر

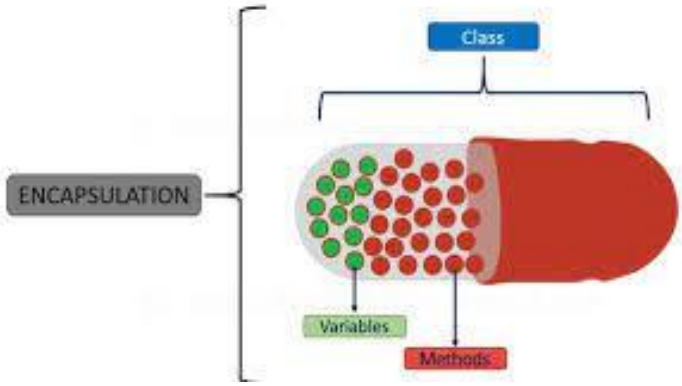
```
public void RegisterEmployee(string name, int employeeId, int salary)
    {
       //
    }
public void RegisterEmployee(string name, int employeeId, int salary, string
department)
    {
       //
    }
```

2-پلی مورفیسم زمان اجر (runtime polymorphism) Overriding

در این نوع، کلاس های مشتق شده می توانند تابع هایی که از کلاس پایه مشتق شده اند را با پیاده سازی های خود بازنویسی کنند.
این نوع در زمان اجرا برنامه متوجه تابع مناسب خود میشوند.

توضیح دهید چگونه اصل کپسوله سازی در برنامه شی گرا کمک به نگهداری میکند؟

برای پاسخ ابتدا باید بدانیم که کپسوله سازی چیست؟encapsulation
به معنی محصور کردن چیزی است یا از بین بردن دسترسی به قسمت های خصوصی کد
درکل یعنی اینکه هر آبجکت وضعیت خود را کنترل میکند
با توجه به تعریف بالا درصورت عدم استفاده از اصل کپسوله سازی موارد زیر ایجاد میشود
استدلال در کد بسیار دشوار میشود
بخش های غیر مرتبط کد به یکدیگر وابسته میشوند.



تفاوت اینتر فیس و ابسترکت چیست؟

Abstract :
پنهان کردن جزییات پیاده سازی درون چیزی
یا اینکه ما به نحوه عمل کردن یک تابع کاری نداریم

*ماهیت کلاس دارد.

*اعضای abstract دارد.

*اعضای غیر abstract دارد.

*از تمام امکانات کلاس عادی می تواند استفاده کند

*فقط برای وراثت استفاده میشود و از آن نمیتوان نمونه سازی کرد.

*یک کلاس abstract همان اینترفیس است اما بسیار گسترده تر و قدرتمند تر

*اصولا به کلاس های abstract کلاس های نیمه کاره گویند چون با فرزندانش تکمیل تر می شود.

Interface

*اینترفیس کلاس نیست و ماهیت کلاس ندارد!

*اینترفیس ها سازنده ندارند!

*اینترفیس ها فیلد ندارند!

*نمی توان از اینترفیس ها نمونه سازی کرد.

*در اینترفیس سطوح دسترسی معنا ندارند و همگی به صورت پیش فرض publicهستند.

*در صورتی که نخواهیم کلیه متدها در کلاس‌های فرزندان پیاده سازی شوند و تعدادی از آنها را در کلاس پدر کدنویسی کنیم، باید از کلاس Abstract استفاده کنیم.

*زمانی که بخواهیم تمام متدهای معرفی شده در کلاس Base Class به طور کامل در کلاس های فرزندان پیاده سازی شوند باید از Interface ها استفاده کنیم.