۱.توضیح دهید چگونه اصل "کپسوله‌سازی" در برنامه‌نویسی شیءگرا کمک به نگهداری کد می‌کند؟

باعث راحت تر شدن توسعه پذیری برنامه میشود . دسترسی به جزیات داخلی یک شی را محدود میکند این کار باعث میشود که از تغییر پیدا کردن ناخواسته جلوگیری کند (یا دولوپر بعدی که روی کد کار کند )

۲. تفاوت اصل "پلی‌مورفیسم" با (Overloading) و (Overriding) چیست ؟

تفاوت اصلی انهای در این است که پولیمورفسیم یک اصل کلی است که هم run time و هم compile time میتوان استفاده کرد ولی اوورلودینگ و اووررایدینگ دو روش خاص برای پیاده سازی پولیمورفیسم است

۳.تفاوت میان اصطلاحات "Interface" و "Abstract Class" چیست و در چه مواقعی از هرکدام استفاده می‌شود؟

در اینترفیس متد ها بدنه دندارند و کلاس هایی که از ان ارث بری کرده اند موظف به پیاده سازی هستند ولی ابسترکت متد ها برای کلاس های فرزند باز نویسی میشوند

کلاس ها فقط میتوانند از یک کلاس ابسترکت تارث بری کنند ولی میتوانند از چند اینتر فیس ارث بری کنند

۱. مواقعی که نیاز به وراثت چند گانه داریم باید از اینترفیس استفاده کنیم، به دلیل اینکه این امکان در کلاس‌های Abstract وجود ندارد.

۲. زمانی که بخواهیم تمام متدهای معرفی شده در کلاس پایه به طور کامل در کلاس مشتق شده پیاده شوند باید از اینترفیس استفاده کنیم.

۳. وقتی در پروژه‌های بزرگ با تغییرات زیادی مواجه هستیم، استفاده از کلاس Abstract توصیه می‌شود، زیرا با تغییر آن به طور خودکار تغییرات در کلاس های مشتق شده اعمال می‌شوند.

۴. با توجه به اینکه به غیر از اعلان متدها و خصوصیت‌ها امکان تعریف عناصر دیگری در اینترفیس ها وجود ندارد، در صورتیکه ملزم به استفاده از این عناصر باشیم، استفاده از کلاسهای Abstract ضروری می‌باشد.

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

۶. به طور کلی یک اینترفیس چارچوب و قابلیتهای یک کلاس را مشخص می کند و یک قرارداد است، ولی کلاس Abstract نوع کلاس را معین می‌کند. این تفاوت کمک بسیاری برای تشخیص زمان استفاده از این دو را به برنامه نویسان می دهد.

اصل DRY

به ما میگوید اگر یک تکه کد را بیشتر تز یک بار بخواهیم استفاده کنیم باید ان را یک متد کنیم