اصول برنامه نویسی شی گرا – به زبان ساده

زمان مطالعه: ۳۲ دقیقه

برنامه نویسی یکی از مهم‌ترین مهارت‌های روز دنیا به حساب می‌آید و برنامه نویسی شی گرا (Object Oriented Programming)یا به اختصار «OOP» هم یکی از مهم‌ترین مقوله‌های برنامه نویسی است. در این مقاله به بررسی اصول برنامه نویسی شی گرا (Object Oriented Programming Principles) پرداخته شده است. هدف اصلی شی گرایی پیاده‌سازی موجودیت‌های دنیای واقعی در برنامه نویسی است. اصول برنامه نویسی شی گرا روش‌ها و متغیرهایی را برای استفاده مجدد از کدها در برنامه، همراه با سازگاری کامل ایجاد می‌کنند. اصول برنامه نویسی شی گرا شامل چهار اصل است که در این مطلب به طور جامع به آن‌ها پرداخته می‌شود.

فهرست مطالب این نوشته

پیش از پرداختن به هر یک از اصول برنامه نویسی شی شی گرا بهتر است ابتدا مفهوم برنامه نویسی شی گرا به طور خلاصه و به بیان ساده شرح داده شود.

برنامه نویسی شی گرا چیست؟

برنامه نویسی شی گرا روشی برای توسعه برنامه نویسی بر اساس «شی (Object)» است. شی گرایی نوعی از برنامه نویسی به حساب می‌آید که در طراحی نرم افزار به وسیله آن، به جای استفاده از توابع و منطق برنامه نویسی از داده‌ها یا اشیا استفاده می‌شود. در دنیا همه چیز را می‌توان به عنوان یک شی در نظر گرفت. در برنامه نویسی شی گرا می‌توان اشیا را بر اساس خصوصیت‌ها و رفتار آن‌ها تعریف کرد. این رویکرد از برنامه نویسی برای برنامه‌های بزرگ، پیچیده و فعال در بروزرسانی و «نگهداری (Maintain)» آن‌ها مناسب است. این برنامه‌ها شامل روش‌هایی برای ساخت و طراحی ازجمله برنامه‌ها و اپلیکیشن‌های موبایل هستند، برای مثال می‌توان از برنامه نویسی شی گرا در تولید نرم افزارهای شبیه‌سازی سیستم استفاده کرد.

سازماندهی یک برنامه به صورت شی گرا، آن را برای توسعه با روش‌های گروهی و مشارکتی نیز سودمند می‌کند. از مزایای کلی برنامه نویسی شی گرا می‌توان به «قابلیت استفاده مجدد از کدها(Code Reusability)»، «مقیاس‌پذیری (Scalability)» و «کارایی (Efficiency)» بالا اشاره کرد. اولین مرحله استفاده از شی گرایی جمع‌آوری همه اشیا مورد نیاز برنامه و ارتباطات میان آن‌ها است. نمونه‌هایی از یک شی می‌تواند شامل برخی موجودیت‌های فیزیکی ازجمله یک انسان با ویژگی‌هایی مانند نام و آدرس یا برنامه‌های رایانه‌ای کوچک مانند ویجت‌ها باشد.

مثالی ساده از شی گرایی در دنیای واقعی

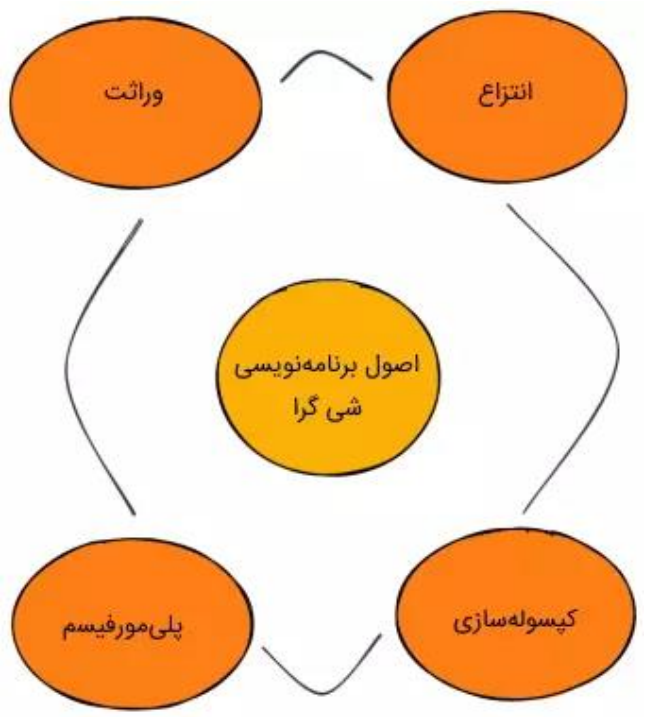
به عنوان مثالی از برنامه نویسی شی گرا می‌توان یک تلویزیون را در نظر گرفت، این تلویزیون یک شی است و «ویژگی‌هایی (Property)» مربوط به آن ازجمله ارتفاع، رنگ، نوع هوشمند یا CRT و سایر موارد وجود دارند. رفتارهایی که یک تلویزیون می‌تواند انجام دهد شامل عوض شدن کانال آن، تغییر میزان صدای آن، روشن و خاموش کردن آن و بسیاری از رفتارهای دیگر است. عملیات رفتار، بر ویژگی‌ها تأثیر می‌گذارد، برای مثال تغییر کانال که یک رفتار است بر ویژگی آن تأثیر خواهد گذاشت. بنابراین، موارد مشابهی در برنامه نویسی شی گرا اتفاق می‌افتد و هر شی دارای برخی ویژگی‌ها و رفتارهای مرتبط به آن است که در کلاس (Class) تعریف می‌شوند.

به طور خلاصه، شی گرایی روشی برای برنامه نویسی است که در توسعه نرم افزار استفاده می‌شود و اصول خاصی دارد که به توسعه آن کمک می‌کند. روش‌های توسعه نرم افزار بسیاری وجود دارند، اما می‌توان گفت که شی گرایی معروف‌ترین روش به حساب می‌آید. در ادامه این مقاله، به بررسی اصول برنامه نویسی شی گرا پرداخته می‌شود.

اصول برنامه نویسی شی گرا

شی گرایی در برنامه نویسی روشی برای طراحی برنامه‌ها با استفاده از کلاس و اشیا به حساب می‌آید. با پیاده‌سازی شی گرایی می‌توان اشیا را در برنامه سازماندهی کرد. این روش به عنوان رویکردی جهت کنترل داده‌ها در کدها نیز مورد استفاده قرار می‌گیرد. شی گرایی در برنامه نویسی حوزه‌ای است که علاوه بر دو مفهوم اساسی «کلاس» و «شی»، چهار اصل و خصوصیت کلیدی زیر را نیز شامل می‌شود. چهار اصل برنامه نویسی شی گرا عبارت‌اند از:

- «وراثت» یا «ارث بری» (Inheritance)
- «تجرید» یا «انتزاع» (Abstraction)
- «پلی مورفیسم» یا «چند ریختی» یا «چند شکلی» (Polymorphism)
- «کپسوله سازی» (Encapsulation)



در این رویکرد، برنامه نویسان، نوع داده، ساختار نوع داده و عملیاتی را تعریف می‌کنند که روی ساختار داده‌ها انجام می‌شوند. بنابراین، علاوه بر کلاس‌ها و اشیا، برای بهره‌مندی از مزیت‌های شی‌گرایی در برنامه نویسی، لازم است چهار اصل فوق نیز در برنامه پیاده‌سازی و برقرار شوند.

برای درک درست اصول برنامه نویسی شی گرا، باید با اصول و مبانی برنامه نویسی شی گرا و مؤلفه‌های اصلی آن آشنایی داشت؛ به همین دلیل در انتهای مقاله به این موارد اشاره شده است. در صورت عدم آشنایی با آن‌ها و نیاز به مرور، پیشنهاد می‌شود، ابتدا به بخش مؤلفه‌های برنامه نویسی شی گرا مراجعه و سپس این بخش مطالعه شود.

در بخش بعدی از این مقاله، انواع اصول برنامه نویسی شی گرا را بررسی می‌کنیم.

مانند بسیاری از رویکردهای موجود در زمینه مهندسی‌های دیگر مانند مهندسی‌های الکترونیک که دستگاهی می‌سازند، مهندسین خودرو که وسایل نقلیه ایجاد می‌کنند و سایر موارد، مهندسی نرم افزار و برنامه نویسی نیز برای رویکردهای خود دارای اصولی است. پیش از شرح اصول برنامه نویسی شی گرا، در خصوص مفاهیم برنامه نویسی، دو عنصر مهم زیر وجود دارند که ابتدا به شرح آن دو پرداخته شده است:

- داده‌ها
- عملیات انجام شده روی داده‌ها یا همان توابع

به طور کلی در توسعه نرم افزار، هدف اصلی آن کار بر روی داده‌ها است. اگر هر تابعی بر روی داده‌ها اجرا شود، ممکن است این احتمال وجود داشته باشد که در آینده نیز همان عملیات را بتوان به همان روش یا به روشی اصلاح شده روی داده‌ها انجام داد. بنابراین روشی برای این استفاده مجدد به وجود آمده است که از اصول برنامه نویسی شی گرا به حساب می‌آید، در بخش‌های بعدی این اصول شرح داده می‌شوند. ابتدا به شرح اصل «تجرید» یا «انتزاع (Abstraction) «پرداخته شده است.

انتزاع یا Abstraction در برنامه نویسی شی گرا چیست ؟

انتزاع در شی گرایی فرایندی است که بر اساس آن تنها اطلاعات مورد نیاز نمایش داده می‌شوند و اطلاعات غیرضروری پنهان خواهند شد. انتزاع یا همان «Abstraction» به معنی مخفی کردن پیاده‌سازی‌های داخلی و نشان دادن ویژگی‌های مورد نیاز یا مجموعه‌ای از خدمات ارائه شده است. انتزاع یکی از اصول بسیار حیاتی برنامه نویسی شی گرا به حساب می‌آید. می‌توان گفت که هدف اصلی انتزاع، «پنهان کردن» داده‌ها است. انتزاع به معنی انتخاب داده‌های مناسب از میان گروه زیادی از داده‌ها، برای نشان دادن اطلاعات مورد نیاز جهت کمک به کاهش پیچیدگی و چالش‌های برنامه است.

کلاس‌ها و متدها هم می‌توانند انتزاعی باشند. کلاس انتزاعی نوعی کلاس است که یک یا چند متد انتزاعی در آن تعریف می‌شود. متد انتزاعی دارای تعریف متدها است و پیاده‌سازی متدها شامل آن نمی‌شود. در ادامه مثال‌هایی برای درک بهتر انتزاع در شی گرایی ارائه شده است.

مثال های انتزاع در برنامه نویسی شی گرا

در این بخش مثال استفاده از دستگاه‌های ATM بانک ارائه شده است. اگر قصد برداشت پول از این دستگاه وجود داشته باشد، به راحتی می‌توان با استفاده از جزئیات موجود در ویژگی‌های ارائه شده آن‌ها، برداشت پول را انجام داد. به عنوان یک کاربر، هدف استفاده از ATM فقط برداشت پول، انتقال پول، مشاهده موجودی پول و سایر خدمات این‌چنینی است و کاربر علاقه‌ای به دانستن جزئیات برنامه‌ها و پیاده‌سازی‌های داخلی دستگاه ATM مانند جزئیات کارت، اعتبارسنجی و سایر موارد ندارد. دستگاه‌های ATM فقط یک «رابط کاربری گرافیکی (Graphical User Interface | GUI) «ساده برای تعامل کاربران با آن ارائه می‌دهند و نیازی نیست که کاربر از همه پیاده‌سازی‌های داخلی دستگاه آگاه باشد.

بنابراین می‌توان گفت که در کلاس دستگاهATM ، اصل شی گرایی انتزاع رعایت شده است. در هر زبان برنامه نویسی از سینتکس خاصی برای پیاده‌سازی انتزاع استفاده می‌شود. برای مثال یکی از معروف‌ترین زبان‌های برنامه نویسی که از اصول شی گرایی پشتیبانی می‌کند، زبان «جاوا (Java) «است و از کلمه کلیدی abstract برای نشان دادن کلاس‌های انتزاعی خود استفاده می‌کند. همچنین می‌توان با استفاده از رابط یا اینترفیس در جاوا به آن دسترسی پیدا کرد.

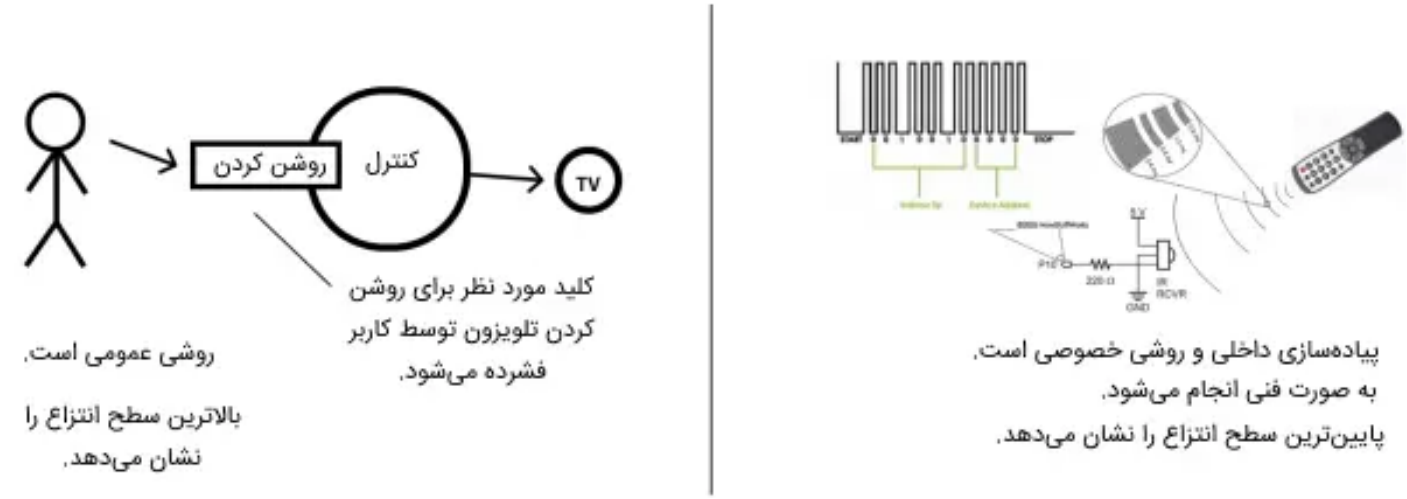
در زبان برنامه نویسیC++ ، دسترسی به این اصول به وسیله گروه‌بندی برخی ویژگی‌ها و استفاده از «تعیین‌کننده سطح دسترسی (Access Specifier) «امکان‌پذیر است و می‌توان مشخص کرد که دسترسی به چه بخش‌هایی از داده‌ها در خارج از کلاس امکان‌پذیر خواهد بود. تعیین کننده‌های سطح دسترسی دارای کلمات کلیدی هستند که تعیین می‌کنند از کجا می‌توان به ویژگی‌ها و متدهای کلاس دسترسی داشت.

برای مثال با استفاده از کلمه کلیدی private دسترسی به داده‌هایی که دارای اصل انتزاع هستند، فقط به داخل کلاس محدود می‌شود و با استفاده از کلمه کلیدی public از هر جایی از برنامه می‌توان به داده‌ها دسترسی داشت. همچنین انواع کلمات کلیدی دیگری نیز در زبان‌های برنامه نویسی متفاوت برای نشان دادن انتزاع وجود دارند. در ادامه شبه‌کدی برای شرح بیشتر انتزاع در برنامه نویسی شی گرا ارائه شده است:

```
class Television{
void changeChannel(){
//Implementation
}
void adjustVolume(){
//Implementation
}
void otherFeatues(){
//Implementation
}
}
```

مثال شبه‌کد فوق، بیان می‌کند که کلاس تلویزیون دارای ویژگی‌هایی ازجمله تغییر کانال، تنظیم صدا و سایر موارد است. بنابراین یک کاربر تنها باید از آن توابع برای رفع نیازهای خود در رابطه با تلویزیون استفاده کند. نیازی نیست که کاربر بداند پیاده‌سازی برنامه داخلی تغییر کانال یا تنظیم صدا به چه صورتی انجام می‌شود. تنها متدهایی برای کاربر ارائه می‌شوند

که جهت استفاده از وظایف ایجاد شده‌اند. در تصویر زیر روشی انتزاعی برای استفاده از کنترل تلویزیون جهت روشن کردن آن ارائه شده است:



به عنوان مثالی از روش‌های برنامه نویسی می‌توان به توابع کتابخانه «Math» در پایتون یا همان کتابخانه ریاضی پرداخت، این کتابخانه دارای توابعی ازجمله pow() و min() و max() و سایر موارد است که برنامه نویسان فقط از آن‌ها استفاده می‌کنند و نیازی نیست که از روش کارکرد و پیاده‌سازی کدهای داخلی آن‌ها با جزئیات اطلاعی داشته باشند. بنابراین استفاده از توابع آماده در برنامه نویسی را هم می‌توان نوعی انتزاع در نظر گرفت. در ادامه این بخش برخی از ویژگی‌ها و مزایای مهم Abstraction در برنامه نویسی شی گرا شرح داده شده‌اند.

مزایای انتزاع در برنامه نویسی شی گرا چه هستند؟

انتزاع یا Abstraction در برنامه نویسی شی گرا به دلیل داشتن ویژگی‌های بسیاری در بخش‌های متفاوتی از برنامه‌ها مورد استفاده قرار می‌گیرند. در این بخش از مقاله، به بررسی و شرح برخی از این ویژگی‌ها پرداخته شده است و این ویژگی‌ها در ادامه فهرست شده‌اند:

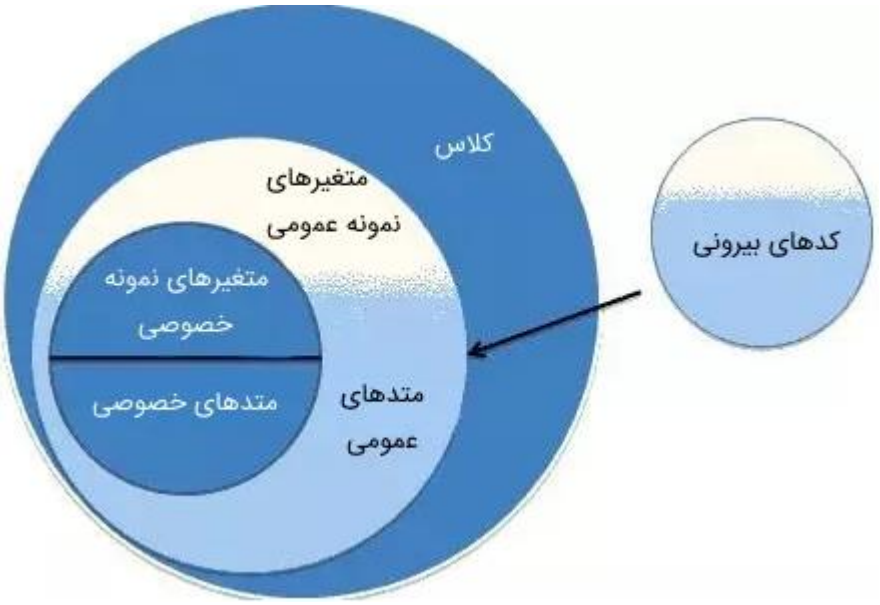
- **امنیت** :با استفاده از روش Abstraction کاربران بیرون از کلاس اطلاعی از روش پیاده‌سازی داخلی کلاس ندارند و این موضوع باعث می‌شود که امنیت برنامه‌ها افزایش پیدا کند. بنابراین با استفاده از این روش کاربران غیرمجاز به داده‌ها دسترسی نخواهند داشت.
- **تقویت آسان برنامه** :فرض می‌شود که قرار است در آینده منطق پیاده‌سازی داخل کلاس تغییر پیدا کند، برای این تقویت و افزایش کارایی برنامه نیازی نیست که همه بخش‌های منطقی خارجی کلاس تغییر پیدا کنند که در دسترس کاربران هستند. فقط باید در متدها تغییراتی ایجاد شود و تأثیری روی عملکرد نخواهد داشت.
- **بهبود سهولت در برنامه نویسی** :این روش به کاربران امکانی را می‌دهد تا از ویژگی‌های برنامه بدون دانستن نوع پیاده‌سازی آن‌ها استفاده کنند، بنابراین، برنامه نویسی با این روش برای افراد ساده‌تر می‌شود.
- **بهبود قابلیت نگهداری** :بدون تأثیری روی عملکرد کاربر، با استفاده از انتزاع اعمال تغییرات راحت‌تر می‌شود و با تغییر یک مورد خاص نیازی نیست که همه موارد وابسته به آن هم تغییر داده شوند و این اصلاحات به صورت خودکار اعمال می‌شوند. بنابراین قابلیت نگهداری کدها در این روش بهبود خواهند یافت.

در بخش بعدی از این مطلب، اصول کپسوله سازی یا همان Encapsulation را یاد می‌گیریم.

کپسوله سازی یا Encapsulation در برنامه نویسی شی گرا چیست ؟

کپسوله سازی یا همان «Encapsulation» یکی از اصول مهم برنامه نویسی شی گرا به حساب می‌آید و به معنی قرار دادن داده‌ها و ویژگی‌ها یا متدها و اعضای داده‌ها در یک واحد است. در کلاس‌های یک برنامه شی گرا، داده و ویژگی‌هایی وجود دارند و عملیاتی روی داده‌ها انجام می‌شود. بر اساس این اصل شی گرایی، داده‌ها در یک واحد ادغام شده‌اند. کپسوله سازی باعث افزایش امنیت برنامه می‌شود؛ زیرا هر چیزی مربوط به یک کار واحد در آن گروه‌بندی شده است و دسترسی به داده‌ها بر اساس نیاز انجام خواهد شد.

این فرآیند با متصل کردن داده‌ها و کدها به یکدیگر و ایجاد یک واحد منفرد، امنیت آن‌ها را تأمین و از سوءاستفاده و دخالت عوامل خارجی جلوگیری می‌کند. کپسوله سازی باعث می‌شود متدهای موجود در هر کلاس فقط به داده‌های همان کلاس دسترسی داشته باشند و متدهای کلاس‌های دیگر به داده‌های کلاس مورد نظر دسترسی ندارند. داده‌های کلاس کپسوله شده از دسترس دیگر کلاس‌ها مخفی می‌شوند و آن‌ها به این داده‌ها دسترسی نخواهند داشت. مفهوم کپسوله سازی به عنوان روش «پنهان‌سازی داده» (Data Hiding) نیز شناخته می‌شود. پنهان‌سازی داده نیز یعنی مخفی کردن داده‌های کلاس و دنیای بیرون از کلاس نیز دسترسی محدودی به این داده‌ها داشته باشند.



برای مثال در کپسوله سازی به زبان C++ استفاده از کلمه کلیدی تعیین‌کننده سطح دسترسی private ، داده‌ها را فقط برای کلاس خودشان قابل دسترسی و تغییر می‌کنند و کاربران در خارج از کلاس امکان دسترسی به آن را ندارند. در ادامه این مطلب برای درک بهتر روش کپسوله سازی در برنامه نویسی شی گرا، مثال‌هایی برای آن ارائه می‌شود.

مثال های کپسوله سازی در برنامه نویسی شی گرا

برای مثال در یک داروخانه، قرص‌های کپسولی وجود دارند که برای درمان بیماری‌های مختلف افراد مورد استفاده قرار می‌گیرند. یک کپسول برای درمان تب در نظر گرفته می‌شود، در این کپسول انواع ترکیبات مختلف برای درمان تب در یک گروه قرار گرفته‌اند. می‌توان گفت که گروه‌بندی ترکیبات متفاوت برای درمان تب در یک کپسول نشان‌دهنده کپسوله سازی است. در این مثال، تب به عنوان یک داده در نظر گرفته می‌شود و کپسول عملیاتی است که روی داده انجام گرفته است. به طور مشابه در برنامه نویسی، زمانی که در یک برنامه چندین عملیات مربوط به یک داده خاص وجود داشته باشند، گروه‌بندی آن‌ها در یک واحد نشان‌دهنده روش کپسوله‌سازی است.

در مثالی دیگر، ساخت یک خودرو در نظر گرفته می‌شود و همه وسایل مورد نیاز برای ساخت خودرو ازجمله موتور، جعبه دنده، چرخ‌ها و سایر موارد به کامل موجود هستند. همه این وسایل در یک بدنه قرار می‌گیرند تا خودرو ساخته شود. هیچ‌وقت در طراحی یک خودرو این‌طور نیست که چرخ‌ها به ماشین متصل نباشند یا موتور از ماشین خارج شود، همه وسیله‌ها در یک بدنه قرار می‌گیرند. بنابراین، در اصل کپسوله سازی برنامه نویسی شی گرا نیز همه عملیاتی که به یک داده خاص ارتباط دارند، در یک کپسول گروه‌بندی می‌شوند.

به عنوان مثالی از مبحث برنامه نویسی، می‌توان ساختمان داده «لیست پیوندی» (Linked List) را در نظر گرفت. در این مثال یک گره در ساختمان داده لیست پیوندی ایجاد می‌شود که شامل داده‌ها است و این گره با کلاس لیست پیوندی ارتباط دارد. عملیاتی که روی داده‌ها انجام می‌شود ازجمله درج، حذف، جستجو و سایر موارد در یک کلاس واحد گروه‌بندی می‌شوند و این کار مثالی برای کپسوله سازی به حساب می‌آید. در ادامه شبه‌کدی برای درک بهتر این مثال ارائه شده است:

```
class LinkedList{
private class Node{
data;
next pointer;
}
public createNode(){}
public addNode(){}
public deleteNode(){}
.
.
}
```

در شبه کدهای فوق، ویژگی کپسوله سازی مشاهده می‌شود و هر عملیات مرتبط با لیست پیوندی به عنوان کلاس private یا همان خصوصی تعریف شده است. همچنین عملیاتی که با نوع public یا همان عمومی تعریف شده‌اند برای همه بخش‌ها قابل دسترسی هستند.

پیاده‌سازی روش کپسوله سازی در هر زبان برنامه نویسی با روش خاصی انجام می‌شود، برای مثال در زبان جاوا برای پیاده‌سازی این قابلیت از متدهای تنظیم کننده (Setter) و گیرنده (Getter) استفاده شده است و در زبان C++ از متدهای گیرنده و تنظیم‌کننده با نام‌های به ترتیب «Mutator» و «Accessor» استفاده می‌شود. اگر مؤلفه‌ای از پنهان‌سازی داده‌ها و انتزاع استفاده کند به عنوان یک «مؤلفه کپسوله شده» (Encapsulated Component) شناخته خواهد شد. در بخش بعدی از مطلب اصول برنامه نویسی شی گرا به بررسی ویژگی‌های کپسوله سازی می‌پردازیم.

مزایای کپسوله سازی در برنامه نویسی شی گرا چیست ؟

کپسوله سازی که یکی از اصول مهم شی گرایی به حساب می‌آید، دارای ویژگی‌ها و مزایایی است که استفاده از آن را برای برنامه سودمند می‌کند. از این‌رو، در این بخش به بررسی برخی از ویژگی‌های کپسوله سازی پرداخته شده است:

- با استفاده از روش کپسوله سازی می‌توان امنیت بسیار بالایی را برای برنامه‌ها و داده‌ها فراهم کرد.
- کپسوله سازی داده‌ها و عملیات آن‌ها را در یک واحد و در کنار یکدیگر گروه‌بندی می‌کند.
- این روش یک لایه امنیتی اضافه به داده‌ها می‌افزاید و به کاربر امکان دسترسی فقط به لایه‌های مجاز را می‌دهد.

در بخش بعدی از مطلب «اصول برنامه نویسی شی گرا» این موضوع را یاد می‌گیریم که «وراثت» یا «ارث بری» (Inheritance) چیست.

وراثت یا Inheritance در برنامه نویسی شی گرا چیست ؟

وراثت یکی از اصول برنامه نویسی شی گرا است که با استفاده از آن می‌توان ویژگی‌های موجود در یک کلاس را در کلاسی جدید استفاده کرد. فرض می‌شود کلاسی وجود دارد و به عنوان کلاس «والد» (Parent) در نظر گرفته می‌شود و این کلاس دارای چند متد مرتبط با کلاس‌های دیگر است. سپس، کلاس جدیدی به عنوان کلاس «فرزند» (Child) اعلان می‌شود که متدهایی مرتبط به خودش دارد. بنابراین، زمانی که کلاس فرزند از کلاس والد ارث بری می‌کند، کلاس فرزند می‌تواند از متدهای موجود در کلاس والد در کنار متدهایی استفاده کند که خودش در کلاس دارد. به این رویکرد ارث بری یا وراثت یا همان Inheritance در برنامه نویسی شی گرا گفته می‌شود.

وراثت روشی است که با استفاده از آن یک شی خصوصیت‌های شی دیگری را کسب می‌کند یا به ارث می‌برد. وراثت از طبقه‌بندی سلسله‌مراتبی (Hierarchical Classification) نیز پشتیبانی می‌کند. ایده دسته‌بندی سلسله‌مراتبی (طبقه‌بندی بالا به پایین) به این صورت است که کلاس‌های جدیدی مبتنی بر کلاس‌های فعلی ایجاد می‌شوند. در ادامه این مطلب، برای درک بهتر وراثت در برنامه نویسی شی گرا، مثال‌هایی از این اصل مهم شی گرایی ارائه شده است.

مثال های وراثت در برنامه نویسی شی گرا

همیشه بروزرسانی‌هایی وجود دارند که روی سخت افزار و نرم افزار سیستم‌ها اعمال می‌شوند. برای مثال در سال‌های ابتدایی استفاده از تلفن‌های همراه، از آن‌ها فقط برای صحبت کردن استفاده می‌شد و سپس با پیشرفت و بروزرسانی‌های به وجود آمده در آن‌ها قابلیت‌هایی ازجمله دسترسی به اینترنت، دوربین و سایر موارد اضافه شد. بنابراین، این انقلاب به وجود آمده در صنعت ساخت تلفن‌های همراه با استفاده از اضافه کردن برخی ویژگی‌ها به تلفن‌های همراه قدیمی ایجاد شده است و عملکردهای امروزی آن‌ها از اول ایجاد نشده‌اند، از این‌رو می‌توان روند پیشرفت تلفن‌های همراه را به عنوان مثالی برای وراثت در شی گرایی در نظر گرفت.

به عنوان مثالی دیگر می‌توان به بررسی انواع حیوانان پرداخت. مثلاً یک مرغ عضوی از دسته‌بندی پرندگان به حساب می‌آید. پرندگان نیز بخشی از کلاس تخمگذاران هستند. تخمگذاران هم زیرکلاس حیوانات محسوب می‌شوند. از دسته‌بندی سلسله‌مراتبی که همان دسته‌بندی بالا به پایین است در صورتی استفاده می‌شود که مثلاً قصد توصیف دسته خاصی از حیوانات مثل پستانداران یا تخمگذاران وجود داشته باشد. در این دسته‌بندی سلسله‌مراتبی، دسته یا همان کلاس تخمگذاران دارای خصوصیت‌های جزئی‌تری مثل نوک، خونسردی، خونگرمی و سایر موارد هستند. در واقع تخمگذاران کلاس فرزندی برای کلاس حیوانات به حساب می‌آیند و دسته حیوانات نیز کلاس والدی برای دسته تخمگذاران محسوب می‌شوند.

کلاس فرزند کلاسی است که خصوصیت‌های کلاس والد را به ارث می‌برد. به کلاس فرزند، زیرکلاس (Sub Class) یا «کلاس مشتق شده» (Derrived Class) هم می‌گویند. یک کلاس والد، «سوپرکلاس» (Super Class)، «کلاسی پایه» (Base Class) یا کلاس والد (Parental Class) است که زیرکلاس خصوصیت‌های آن را به ارث می‌برد. در ادامه شبه‌کدی از پیاده‌سازی وراثت در شی گرایی ارائه شده است:

```
class Base{
data;
feature1(){}
feature2(){}
}
class Derived extends Base{
feature3(){}
}
```

در قطعه کد فوق، یک کلاس پایه وجود دارد که دارای داده‌ها و ویژگی‌هایی است. سپس کلاس مشتق شده یا همان فرزند کلاس پایه را پیاده‌سازی می‌کند و در کنار ویژگی که خودش دارد از ویژگی‌ها و داده‌های آن نیز استفاده می‌کند. بنابراین به طور کلی می‌توان گفت که کلاس مشتق شده دارای تمام ویژگی‌هایی است که در کلاس پایه قرار دارند و این نشان‌دهنده ارث بری است.

در این مثال feature1() و feature2() ویژگی‌های کلاس پایه و feature3 ویژگی کلاس مشتق شده به شمار می‌روند. زبان‌های برنامه نویسی مختلف روش‌های متفاوتی برای پیاده‌سازی وراثت دارند، برای مثال در زبان جاوا از کلمه کلیدی extends ، زبان C++ از «دو نقطه» (:) و زبان «پایتون» (Python) از پرانتز برای ارث بری استفاده می‌کنند. در بخش بعدی از این مطلب، به شرح ویژگی‌های وراثت در شی گرایی پرداخته شده است.

مزایای وراثت در برنامه نویسی شی گرا چیست ؟

بزرگترین و مهم‌ترین ویژگی وراثت در برنامه نویسی شی گرا، قابلیت استفاده مجدد از کدها است؛ به این دلیل که نیازی نیست کدهای نوشته شده در کلاس والد، مجدداً در کلاس فرزند نوشته شوند و می‌توانند از ویژگی‌های نوشته شده قبلی استفاده کنند و به صورت خودکار در کلاس مشتق شده استفاده شوند. در ادامه برخی دیگر از ویژگی‌های این اصل شی گرایی ارائه شده‌اند:

- باعث کاهش «افزونگی» (Redundancy) در کدها می‌شود.
- باعث کاهش سایز «کدهای منبع» (Source Code) و بهبود خوانایی کدها می‌شود.
- کدها را با استفاده از تقسیم آن‌ها به گروه‌های کلاس‌های والد و فرزند ساده‌تر می‌کند.
- پشتیبانی از قابلیت «توسعه‌پذیری» (Extensibility) به وسیله «رونویسی» (Override) عملکردهای کلاس پایه توسط کلاس‌های فرزند انجام می‌شود.
- به دلیل اینکه با قابلیت استفاده مجدد از کدها، کدهای کلاس پایه همیشه در حال تست و اشکال‌زدایی هستند، قابلیت اطمینان در برنامه‌ها افزایش می‌یابد.

روش وراثت در شی گرایی دارای معایب معدودی نیز است که در ادامه به بررسی برخی از آن‌ها می‌پردازیم:

- به دلیل این‌که در روش وراثت کلاس فرزند و والد وابستگی و «اتصال» (Coupling) زیادی با یکدیگر دارند، اگر کدهای کلاس والد تغییر پیدا کنند، روی کلاس فرزند نیز تأثیر خواهند گذاشت.
- در یک کلاس سلسله‌مراتبی بسیاری از اعضای داده‌ها بدون استفاده می‌مانند و حافظه اختصاص داده شده به آن‌ها استفاده نمی‌شود. بنابراین، اگر وراثت به درستی پیاده‌سازی نشود، این موضوع، روی کارایی برنامه تأثیر می‌گذارد.

در بخش بعدی از تعریف و شرح وراثت که یکی از اصول برنامه نویسی شی گرا است، انواع مختلف وراثت را یاد می‌گیریم.

انواع مختلف وراثت چیست ؟

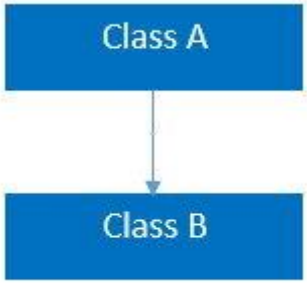
رویکرد وراثت در برنامه نویسی شی گرا دارای انواع مختلفی است که در زبان‌های برنامه نویسی استفاده می‌شوند. در ادامه انواع وراثت فهرست شده‌اند و پس از آن هر یک از آن‌ها بررسی خواهند شد.

- «منفرد» (Single Level)
- «چند سطحی» (Multilevel)
- «چندتایی» (Multiple)
- «چندمسیره» (Multipath)
- «سلسله‌مراتبی» (Hierarchical)
- «ترکیبی» (Hybrid)

در بخش‌های بعدی به بررسی و شرح هر کدام از این انواع وراثت پرداخته شده است. ابتدا خواهیم دید که وارثت منفرد چیست و چه عملکردهایی دارد.

وراثت منفرد در برنامه نویسی شی گرا چیست ؟

در این نوع وراثت یک کلاس مشتق شده، ویژگی‌ها و رفتارهایی را از یک کلاس پایه به ارث می‌برد، به همین دلیل به آن وراثت منفرد گفته می‌شود. در مثال زیر، کلاس A والد یا پایه و کلاس B فرزند یا مشتق شده است و کلاس B ویژگی‌ها و رفتارهای کلاس والد A را به ارث می‌برد.

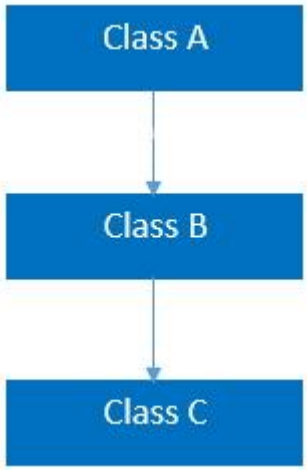


شبه‌کدی برای برنامه نویسی وراثت منفرد به صورت زیر نوشته می‌شود:

```
//Base Class
class A
{
public void fooA()
{
//TO DO:
}
}

//Derived Class
class B : A
{
public void fooB()
{
//TO DO:
}
}
```

وراثت چند سطحی در برنامه نویسی شی گرا چیست ؟

در این نوع از رویکرد وراثت، یک کلاس از کلاس دیگری ارث بری می‌کند که آن کلاس دیگر نیز مجدداً از یک کلاس متفاوت دیگر ارث می‌برد. یعنی کلاس سطح آخر بیش از یک کلاس والد دارد؛ به همین دلیل به آن وراثت چند سطحی می‌گویند. در مثال زیر، کلاس C ویژگی‌ها و رفتارهای کلاس B را به ارث می‌برد و کلاس B نیز ویژگی‌ها و رفتارهای کلاس A را به ارث خواهد برد. بنابراین، در این مثال، کلاس A والد کلاس B و کلاس B والد کلاس C به شمار می‌روند، پس می‌توان گفت که کلاس C ویژگی‌ها و رفتارهای کلاس A به همراه کلاس B را نیز به ارث برده و یک وراثت چند سطحی به وجود آمده است.



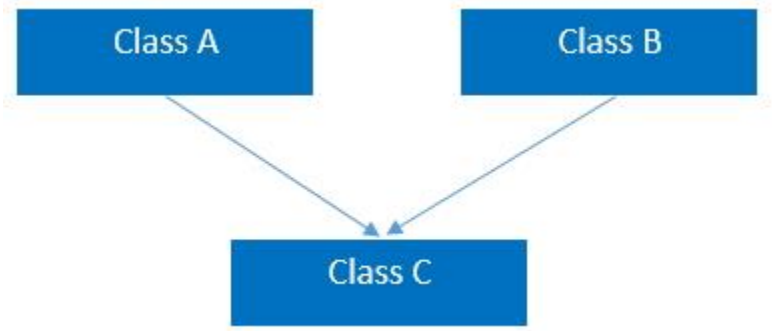
شبه‌کد برنامه نویسی وراثت چند سطحی به صورت زیر نوشته می‌شود:

```
//Base Class
class A
{
public void fooA()
```

```
{
//TO DO:
}
}

//Derived Class
class B : A
{
public void fooB()
{
//TO DO:
}
}

//Derived Class
class C : B
{
public void fooC()
{
//TO DO:
}
}
```

وراثت چندتایی در برنامه نویسی شی گرا چیست ؟

در این نوع از وراثت، کلاس والد شامل بیش از یک کلاس پایه می‌شود، ارث بری چندتایی توسط برخی از زبان‌های فریم ورک .NET. از جمله C#، F# و سایر موارد و زبان برنامه نویسی جاوا پشتیبانی نمی‌شود. در مثال زیر، کلاسC ، ویژگی‌ها و رفتارهای کلاس B و A را در یک سطح به ارث می‌برد. بنابراین کلاس‌های A و B هر دو والد کلاس C هستند.



در ادامه، شبه‌کد برنامه نویسی وراثت چندتایی به صورت زیر نوشته شده است:

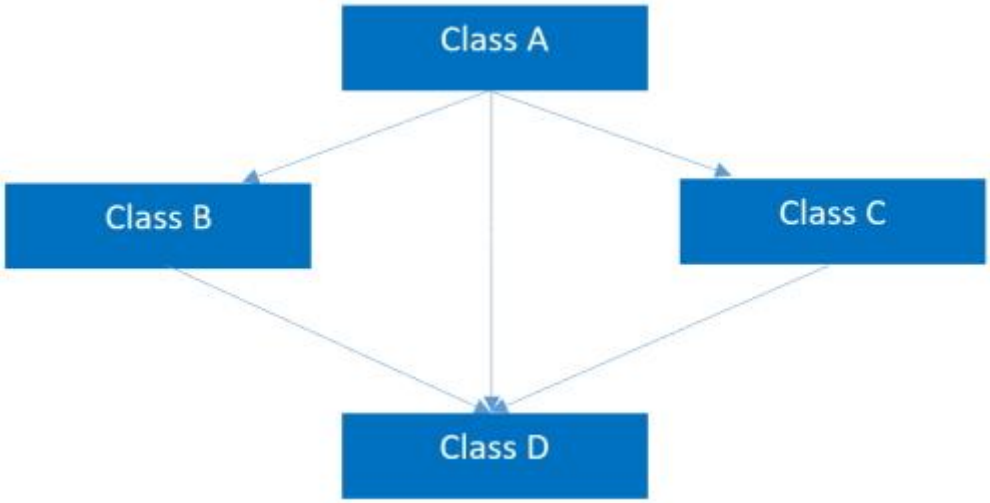
```
//Base Class
class A
{
public void fooA()
{
//TO DO:
}
}

//Base Class
class B
{
public void fooB()
{
//TO DO:
}
}
```

```
//Derived Class
class C : A, B
{
public void fooC()
{
//TO DO:
}
}
```

بخش بعدی از مقاله «اصول برنامه نویسی شی گرا» به بررسی ادامه انواع وراثت به عنوان یکی از اصول برنامه نویسی شی گرا اختصاص داده شده است.

وراثت چندمسیره در برنامه نویسی شی گرا چیست ؟

در این رویکرد از وراثت، کلاس مشتق شده با استفاده از کلاس‌های مشتق شده دیگر ایجاد می‌شود و آن کلاس‌های مشتق شده دیگر دارای یک کلاس والد یکسان هستند. این نوع از وراثت نیز توسط زبان‌های .NET. ازجمله C#، F# و سایر موارد پشتیبانی نمی‌شود. در مثال زیر، کلاسD ، ویژگی‌ها و رفتارهای کلاس C و B و همچنین کلاس A را به ارث می‌برد. هر دو کلاس B و C در یک سطح از کلاس A ارث بری می‌کنند. بنابراین، کلاس A والد هر دو کلاس B و C و همچنین D به حساب می‌آید. از این‌رو، می‌توان گفت در این مثال ارث بری چندمسیره انجام شده است.



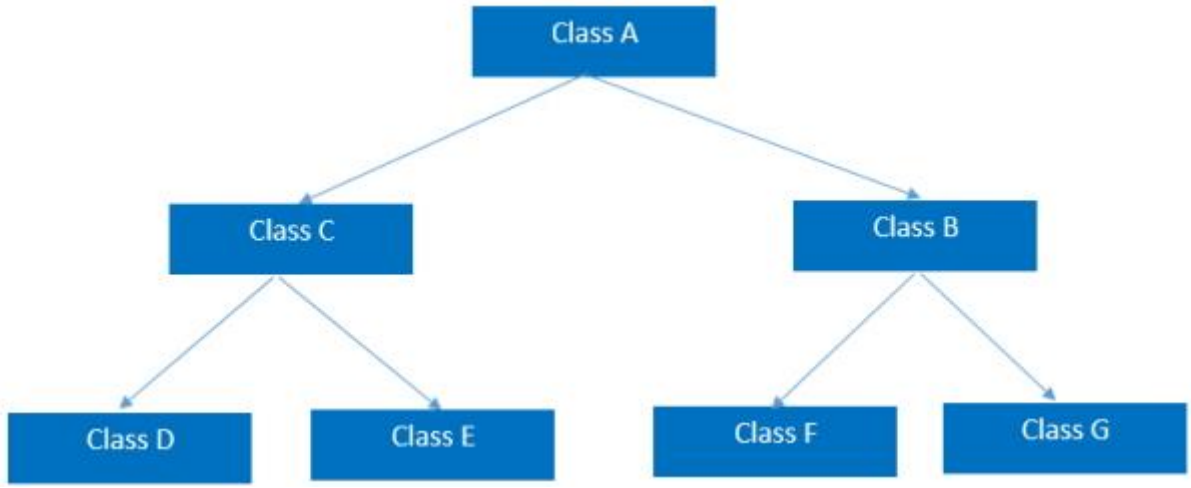
شبه‌کد برنامه نویسی وراثت چندمسیره به صورت زیر نوشته می‌شود:

```
//Base Class
class A
{
public void fooA()
{
//TO DO:
}
}

//Derived Class
class B : A
{
public void fooB()
{
//TO DO:
}
}

//Derived Class
```

```
class C : A
{
public void fooC()
{
//TO DO:
}
}

//Derived Class
class D : B, A, C
{
public void fooD()
{
//TO DO:
}
}
```

وراثت سلسله مراتبی در برنامه نویسی شی گرا چیست ؟

در این نوع از وراثت، یک کلاس والد دارای چند کلاس مشتق شده یا همان زیر کلاس است، به عبارت دیگر چندین کلاس فرزند یک کلاس والد دارند. همچنین کلاس‌های فرزند نیز به عنوان کلاس والد دارای کلاس‌های فرزند دیگری هستند. به عنوان مثال، در تصویر زیر کلاس والد A دارای دو فرزند کلاس B و کلاس C است. همچنین، کلاس B و C نیز هر کدام دارای دو کلاس فرزند دیگر هستند. کلاس‌های D و E فرزندان کلاس C و کلاس‌های G و F فرزندان کلاس B به حساب می‌آیند.



در ادامه، شبه‌کدهایی برای نشان دادن روش نوشتن برنامه نویسی شی گرا وراثت سلسله‌مراتبی نمایش داده شده‌اند:

```
//Base Class
class A
{
public void fooA()
{
//TO DO:
}
}

//Derived Class
class B : A
{
public void fooB()
{
//TO DO:
}
}
```

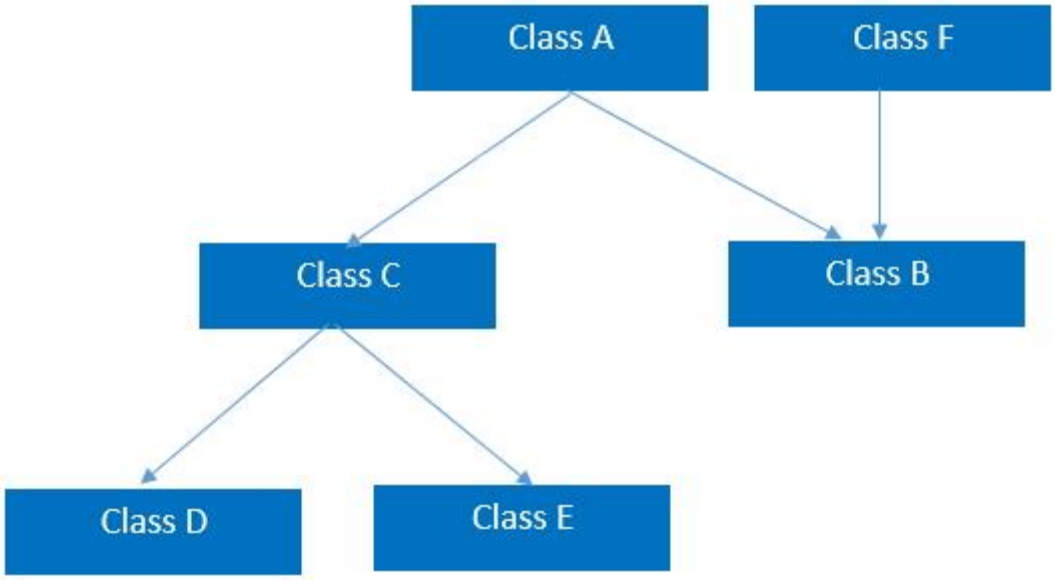
```
//Derived Class
class C : A
{
public void fooC()
{
//TO DO:
}
}

//Derived Class
class D : C
{
public void fooD()
{
//TO DO:
}
}

//Derived Class
class E : C
{
public void fooE()
{
//TO DO:
}
}

//Derived Class
class F : B
{
public void fooF()
{
//TO DO:
}
}

//Derived Class
class G :B
{
public void fooG()
{
//TO DO:
}
}
```

وراثت ترکیبی در برنامه نویسی شی گرا چیست ؟

این نوع از وراثت در برنامه نویسی شی گرا از ترکیب بیش از یک وراثت به وجود می‌آید. از این رو، این وراثت می‌تواند ترکیبی از انواع وراثت زیر باشد:

- وراثت چند سطحی و چندتایی
- ارث بری سلسله‌مراتبی و چند سطحی
- وراثت سلسله‌مراتبی و چندمسیره یا سلسله‌مراتبی
- ارث بری چند سطحی و چندتایی

برخی از زبان‌های .NET. ازجمله C#، F# و سایر موارد از ترکیب وراثت چندتایی و چندمسیره پشتیبانی نمی‌کنند. بنابراین، وراثت ترکیبی توسط این زبان‌های برنامه نویسی پشتیبانی نمی‌شود. به عنوان مثال در تصویر زیر یک وراثت ترکیبی از ترکیب وراثت سلسله‌مراتبی و چندتایی ارائه شده است.



شبه‌کد نشان‌دهنده وراثت ترکیبی مثال فوق در ادامه ارائه شده است:

```
//Base Class
class A
{
public void fooA()
{
//TO DO:
}
}

//Base Class
class F
{
public void fooF()
{
//TO DO:
}
}

//Derived Class
class B : A, F
{
public void fooB()
{
//TO DO:
}
}

//Derived Class
class C : A
{
public void fooC()
{
//TO DO:
```

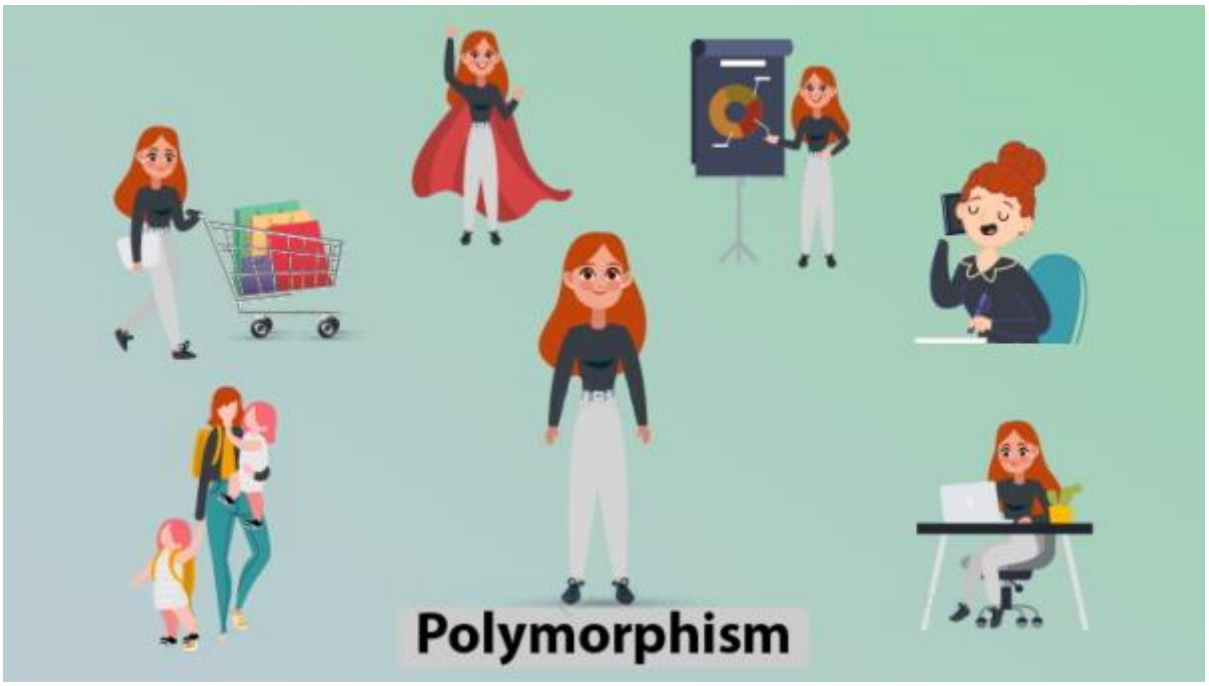
```
}
}

//Derived Class
class D : C
{
public void fooD()
{
//TO DO:
}
}

//Derived Class
class E : C
{
public void fooE()
{
//TO DO:
}
}
```

در بخش بعدی از این مطلب به بررسی یکی دیگر از اصول برنامه نویسی شی گرا، یعنی پلی مورفیسم یا چند ریختی پرداخته می‌شود.

پلی مورفیسم یا چند ریختی در برنامه نویسی شی گرا چیست ؟

پلی مورفیسم، چند ریختی یا چند شکلی اساسی و ضروری‌ترین اصل برنامه نویسی شی گرا به حساب می‌آید. کلمه «Polymorphism»از ترکیب دو کلمه «Poly» به معنی «زیاد» و کلمه «Morph» به معنی شکل یا فرم ایجاد شده است. از این‌رو پلی مورفیسم را می‌توان چند ریختی یا چند شکلی نیز نامید. در برنامه نویسی شی گرا، هر شی یا متدی که دارای بیش از یک نام باشد با این مفهوم ارتباط دارد و این موضوع چیزی جز چند شکلی نیست. پلی مورفیسم به وجود شکل‌های بسیار در برنامه گفته می‌شود یا می‌توان گفت پلی مورفیسم پردازشی به حساب می‌آید که یک عمل واحد را با روش‌های گوناگونی انجام می‌دهد.

پلی مورفیسم در برنامه نویسی جهت بالا بردن خوانایی و استفاده مجدد از داده‌ها به کار گرفته می‌شود. همان‌طور که در دنیای واقعی یک شخص می‌تواند طبق وظایفش در موقعیت‌های گوناگون، جایگاه متفاوتی داشته باشد. مانند جایگاه یک زن در برابر فرزندش به عنوان مادر و در محیط کار به عنوان کارمند، در پلی مورفیسم نیز یک وظیفه طبق جایگاه و رفتار آن با روش‌های مختلف تعریف می‌شود. چندریختی زمانی اتفاق می‌افتد که تعداد زیادی کلاس از طریق ارث بری با یکدیگر مرتبط هستند.



زمانی که هر شی بتواند مرجع شی والد خود را حفظ کند، پلی مورفیسم اتفاق می‌افتد، برای درک بهتر این موضوع در ادامه مثال‌هایی برای پلی مورفیسم در برنامه نویسی شی گرا ارائه شده‌اند.

مثال های پلی مورفیسم در برنامه نویسی شی گرا

دستگاه تلویزیون توسط شرکت‌های مختلفی تولید می‌شود، اما زمانی که فردی قصد گفتن نام این دستگاه را دارد، با نام برند آن، تلویزیون را خطاب نمی‌کند. به راحتی به آن تلویزیون گفته می‌شود و این تلویزیون می‌تواند دارای برندهای متفاوتی باشد. بنابراین می‌توان گفت که این موضوع نشان‌دهنده چند ریختی است. همچنین می‌توان این مفهوم را «عمومیت بخشی (Generalization) «نیز نامید؛ زیرا یک نام کلی وجود دارد که می‌تواند برای همه زیرکلاس‌هایی اعمال شود که از آن مشتق شده‌اند. در ادامه شبه‌کد مرتبط با این مثال ارائه شده است:

```
class Television{
public void TVOn(){}
public void TVOff(){}
}
class TV extends Television{
public void surfInternet(){}
}
```

```
Television TV = new SamsungTV(); //Yes, It is absolutely correct.
```

در قطعه کد فوق، کلاس TV کلاس Television را پیاده‌سازی می‌کند و متد TVOn() و TVOff() در کلاس TV نیز در دسترس هستند. اکنون یک شیء از مرجع کلاس Television و کلاس TV ایجاد شده است. اگر TV.TVOn() فراخوانی شود، به خوبی اجرا می‌شود. بنابراین این مسیری برای رسیدن به پلی مورفیسم است.
به دلیل وجود رویکرد پلی مورفیسم در این مثال می‌توان TV را همان Television در نظر گرفت؛ زیرا Television دارای همه متدهای مرتبط به آن است و زمانی که Television.on() فراخوانی شود، سپس TV نیز روشن خواهد شد. متدها می‌توانند انواع مختلفی داشته باشند که باعث می‌شود انواع پلی مورفیسم ایجاد شوند. در ادامه مطلب «اصول برنامه نویسی شی گرا» به بررسی انواع پلی مورفیسم پرداخته شده است.

انواع پلی مورفیسم در برنامه نویسی شی گرا چیست ؟

پلی مورفیسم در برنامه نویسی شی گرا شامل دو نوع متفاوت زیر است:

- چند ریختی ایستا یا ثابت (Static Polymorphism) یا چندریختی زمان کامپایل (Compile Time Polymorphism)
- چند ریختی پویا (Dynamic Polymorphism) یا چند ریختی زمان اجرا (Runtime Polymorphism)

هر کدام از روش‌های فوق، با استفاده از رویکردهای مخصوصی پلی مورفیسم را انجام می‌دهند که در ادامه نمایش داده می‌شوند:

- متد اضافه بار یا سربارگذاری (Overloading) که مختص به پلی مورفیسم زمان کامپایل است.
- متد رونویسی کردن (Overriding) که در پلی مورفیسم زمان اجرا استفاده می‌شود.

در بخش بعدی از این مطلب، هر یک از این انواع پلی مورفیسم را به همراه متد مختص به آن یاد می‌گیریم.

چند ریختی زمان کامپایل چیست؟

پلی مورفیسم زمان کامپایل به عنوان پلی مورفیسم ایستا (ثابت) نیز شناخته می‌شود. حل کردن این نوع از چند ریختی در زمان کامپایل از طریق متد اضافه بار انجام می‌شود. متد Overloading به عنوان فرایندی تعریف شده است که می‌تواند چندین متد و داده را با نام یکسان در یک کلاس ایجاد کند و هر کدام از این متدها کار متفاوتی را انجام دهند. روش Overloadingزمانی اتفاق می‌افتد که بیش از یک متد با نام یکسان در کلاس پایه وجود داشته باشد. در روش چند ریختی زمان کامپایل، فراخوانی متد در هنگام کامپایل انجام می‌گیرد. در ادامه مثالی برای درک بهتر این مفهوم ارائه شده است.

```
class Base{
public void method1(int x){ }
}
class Derived extends base{
//Overloaded Method
public void method1(float y){ }
}
```

در کدهای فوق، method1 وجود دارد که از آن دو بار و با پارامترهای مختلف استفاده شده است. این متد در اولین استفاده مقدار عدد صحیح و در دومین استفاده مقدار عدد اعشاری را به عنوان پارامتر دریافت می‌کند. به این دلیل به این رویکرد مقیدسازی ایستا (Static Binding) نیز گفته می‌شود و در طول زمان کامپایل مشخص است که کدام متد باید فراخوانی شود. در بخش بعدی از این مطلب به بررسی چند ریختی زمان اجرا پرداخته شده است.

چند ریختی زمان اجرا چیست؟

پلی مورفیسم زمان اجرا با نام مقیدسازی پویا (Dynamic Binding) نیز معرفی شده است. این نوع از پلی مورفیسم جهت فراخوانی یک متد رونویسی شده (Overridden) استفاده می‌شود که به صورت پویا به جای زمان کامپایل، در زمان اجرا انجام می‌شود. متد رونویسی فرآیندی است که در آن کلاس فرزند همان متدی را دارد که در کلاس والد وجود داشت. این

متد زمانی انجام می‌شود که یک کلاس فرزند متدی با نام، پارامترها و نوع بازگشتی والد یا والد یکسان داشته باشد. سپس تابعی که این مقادیر یکسان را دارد، تابع موجود در کلاس والد را رونویسی می‌کند.
به عبارت ساده‌تر، اگر کلاس فرزند تعریف خود را برای متدی ارائه دهد که از پیش در کلاس والد موجود است، سپس می‌توان گفت که آن تابع در کلاس پایه یا والد رونویسی شده است. باید به این نکته توجه داشت که چند شکلی زمان اجرا تنها از طریق توابع قابل دستیابی است و از طریق اعضای داده (Data Member) قابل دسترسی نیست. در برنامه نویسی شی گرا همه اعضایی (Member) که با نوع اصلی تعریف می‌شوند، اعضای داده به حساب می‌آیند.
متد Overriding با استفاده از متغیر مرجع (Reference Variable) کلاس پایه انجام می‌شود. اینکه کدام متد باید فراخوانی شود، بر اساس شیء مشخص شده است که توسط متغیر مرجع به آن ارجاع داده می‌شود. این موضوع «Upcasting» به حساب می‌آید Upcasting .زمانی انجام می‌شود که متغیر مرجع کلاس والد به شی کلاس فرزند اشاره کند. برای درک بهتر موضوع، قطعه کد مثال Upcasting در ادامه ارائه شده است.

```
class A{}
class B extends A{}
A a=new B(); //upcasting
```

در ادامه مثالی برای درک بهتر متد Overriding ارائه شده است:

```
class Base{
public void method1(int x){ } // Shadowed Method
}
class Derived extends base
public void method1(int y){ }//Overriden Method
}
```

در کدهای مثال فوق، method1 در کلاس مشتق شده رونویسی می‌شود و زمانی که شی کلاس مشتق شده ایجاد شده شود، method1 فراخوانی و سپس کلاس مشتق شده method1 فراخوانی می‌شود. بنابراین، کلاس پایه» method1 پنهان (Shadowed) «خواهد شد. به این روش مقیدسازی پویا نیز گفته می‌شود؛ زیرا زمانی که شئی حافظه را در حین اجرا برنامه دریافت می‌کند، بر اساس شی، متد رونویسی شده فراخوانی می‌شود. بنابراین این نوع از پلی مورفیسم در حین اجرای برنامه رخ می‌دهد. در بخش بعدی به صورت خلاصه برخی از مزایای اصول برنامه نویسی شی گرا بررسی شده‌اند.

مزایای اصول برنامه نویسی شی گرا چیست ؟

برنامه نویسی شی گرا یکی از روش‌های برنامه نویسی بسیار پرکاربرد و پرطرفدار به حساب می‌آید و دلیل این محبوبیت، مزایا و ویژگی‌های بسیاری است که این رویکرد در نوشتن برنامه‌ها دارد. در این بخش به برخی از مزایای اصلی اصول برنامه نویسی شی گرا ارائه شده‌اند:

- **«پیمانه‌بندی» (Modularity):** اصل کپسوله‌سازی این امکان را به اشیا می‌دهد که بدون نیاز به ارتباط با بیرون از بخش خود فعالیت داشته باشند، همچنین این اصل «عیب‌یابی (Troubleshooting) «و توسعه مشارکتی را در برنامه ساده می‌کند.
- **استفاده مجدد از کدها:** با استفاده از اصل ارث بری در برنامه نویسی شی گرا کدها می‌توانند مجدداً مورد استفاده قرار بگیرند و دیگر نیازی نیست کدهای یکسان در برنامه نویسی نوشته شوند.
- **بهره‌وری:** برنامه نویسان می‌توانند با کتابخانه‌های مختلف و کدهای قابل استفاده مجدد سریع‌تر برنامه‌های جدید بسازند و بهره‌وری برنامه‌ها را افزایش دهند.
- **بروزرسانی و مقیاس‌بندی آسان:** برنامه نویس‌ها می‌توانند با استفاده از اصول برنامه نویسی شی گرا، توابع و عملکردهای برنامه را به صورت مستقل پیاده‌سازی کنند.
- **«واسط‌ها» (Interface):** به دلیل وجود روش‌های ارسال پیام و واسط‌ها که برای ارتباط بین اشیا استفاده می‌شود، توضیح سیستم‌های خارجی با استفاده از روش شی گرایی ساده است.
- **امنیت:** استفاده از کپسوله‌سازی و انتزاع، کدهای پیچیده را پنهان می‌کند، نگهداری نرم افزار ساده‌تر می‌شود. همچنین برای مثال می‌توان امنیت پروتکل‌های اینترنت را در نظر گرفت.
- **انعطاف‌پذیری:** پلی مورفیسم به توابع این امکان را می‌دهد که با کلاسی سازگار شوند که در آن قرار گرفته‌اند. همچنین اشیا گوناگون نیز می‌توانند از واسط در شی گرایی عبور کنند.

در بخش بعدی از مقاله «اصول برنامه نویسی شی گرا» به مسئله محبوبیت برنامه نویسی شی گرا در مقایسه با دیگر نوع‌های برنامه نویسی پرداخته شده است.

چرا اصول برنامه نویسی شی گرا بسیار مهم هستند؟

اصول برنامه نویسی شی گرا با اشیا زندگی واقعی ارتباط دارند. در این نوع برنامه نویسی هر عملی که قرار است به صورت تابعی انجام شود، با استفاده از کلاس‌ها و اشیا انجام می‌شود. ساختار کدها با برنامه نویسی به روش شی گرا سازمان‌یافته‌تر است؛ زیرا نیازی نیست کدهایی که چند بار در یک برنامه پیاده‌سازی می‌شوند را چند بار نوشت و فقط یک بار کافی است. می‌توان کلاس‌هایی ایجاد کرد که تابع را تعریف می‌کنند و با ایجاد یک شی از آن‌ها، تابع فراخوانی می‌شود. از طرف دیگر شی گرای دارای اصلی به نام «وراثت» است که قابلیت استفاده مجدد از کدها را افزایش می‌دهد و بروزرسانی کدها را بسیار ساده می‌کند. همچنین «انتزاع» و «کپسوله‌سازی» به افزایش امنیت داده‌ها کمک می‌کنند. برخی از پرکاربردترین و مهم‌ترین زبان‌های برنامه نویسی که در آن‌ها از شی گرایی و اکثر اصول برنامه نویسی شی گرا پشتیبانی می‌شود، شامل «جاوا» (Java)، «C++»، «جاوا اسکریپت» (JavaScript)، پرل (Perl)، PHP و C# هستند. همچنین برخی از زبان‌های برنامه نویسی که به طور کامل شی گرا به حساب می‌آیند، شامل پایتون (Python)، روبی (Ruby)،«اسکالا» (Scala) و سایر موارد می‌شوند.

بخش بعدی از مقاله «اصول برنامه نویسی شی گرا» به بررسی چگونگی توسعه یک نرم افزار با استفاده از شی گرایی اختصاص دارد.

چگونه یک نرم افزار با استفاده از شی گرایی توسعه پیدا می کند؟

شی گرایی یک رویکرد مهندسی است که برای توسعه نرم افزار ایجاد شده است. به عبارت دیگر اشیا و کلاس‌ها دارای یک طراحی اولیه هستند، سپس بر اساس این طراحی پیاده‌سازی آن‌ها و این رویکرد توسعه نرم افزار دقیقاً مانند یک طراحی مهندسی انجام می‌شود. برای مثال، همان‌طور که ساختار طراحی ساخت یک ساختمان شامل مراحلی ازجمله برنامه‌ریزی، طراحی نقشه، نهایی شدن نقشه ساختمان و شروع ساخت و ساز ساختمان می‌شود، در برنامه نویسی شی گرا نیز استفاده از مفاهیم کلاس‌ها و اشیا پله‌های توسعه نرم افزار را ایجاد می‌کنند و ابتدا باید طرح کلی برنامه و اشیا و کلاس‌های آن تعریف شوند و در روند ایجاد برنامه از آن‌ها استفاده شود.

در روش‌های برنامه نویسی شی گرا وظایف برای توسعه می‌توانند به راحتی بین توسعه دهندگان تقسیم شوند، همچنین می‌توان از کدها و وظایف قبلی به راحتی در کدهای فعلی استفاده کرد و نرم افزار را توسعه داد. در بخش بعدی به علت پیدایش و ایجاد برنامه نویسی شی گرا پرداخته شده است.

چرا برنامه نویسی شی گرا به وجود آمده است؟

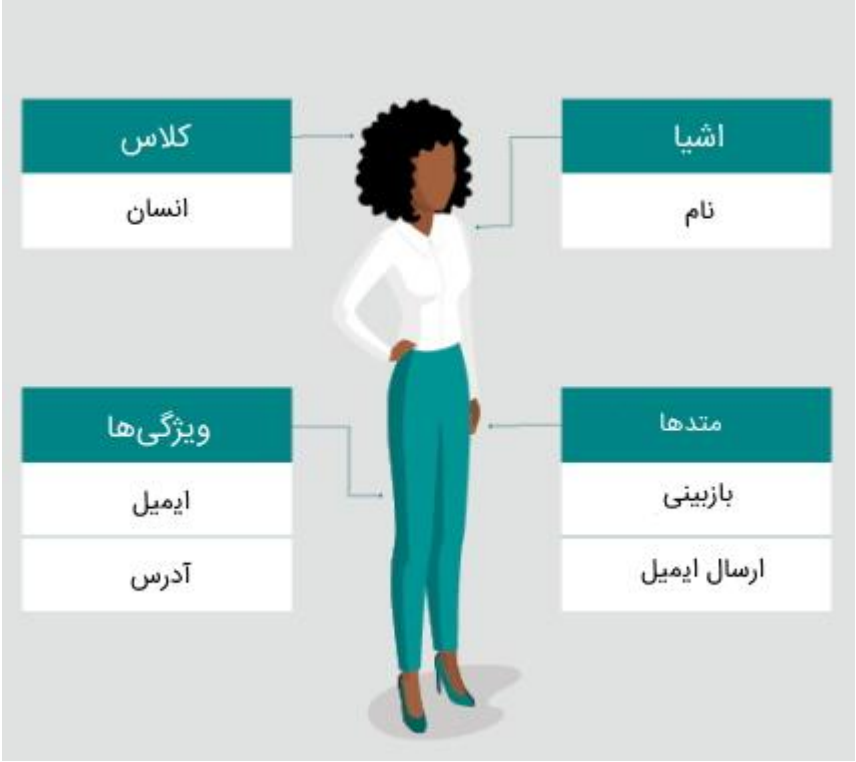
برنامه نویسی شی گرا توسط «Alan Kay» در طول سال‌های ۱۳۳۹ تا ۱۳۴۶ شمسی (۱۹۶۰ تا ۱۹۶۷ میلادی) معرفی شده است. در آن سال‌ها این نوع برنامه نویسی محبوبیت زیادی میان برنامه نویسان نداشت؛ زیرا در آن سال‌ها کامپیوترها توانایی اجرا و ایجاد برنامه‌ها با مقیاس بزرگ را نداشتند و فقط از آن‌ها برای پیاده‌سازی برخی وظایف خاص و ساده استفاده می‌شد. اما امروزه وظایف نرم افزاری بسیار محبوب هستند و اندازه نرم افزارها هم افزایش پیدا کرده‌اند. بنابراین، مدیریت رویکردهای «برنامه نویسی رویه‌ای» (Procedural Programming) یا «برنامه نویسی تابعی» (Functional Programming) به دلیل وجود حجم بالای کدها بسیار دشوار خواهد بود.

همچنین زمانی که در کدهای طولانی نوشته شده با این رویکردها خطایی رخ دهد، «اشکال‌زدایی» (Debugging) آن‌ها به سختی انجام می‌شود. در نهایت، برنامه نویسی شی گرا مشکلاتی از این قبیل را در برنامه نویسی با استفاده از اصول مخصوص به خودش حل کرد. در این نوع از برنامه نویسی وظایف با توجه به عملکرد کلاس تعریف می‌شوند، همچنین هر کلاسی را می‌توان به راحتی مدیریت و در صورت بروز هر گونه خطا، آن را اشکال‌زدایی کرد. به عنوان مثال اگر نرم افزاری برای بانک طراحی شود، بانک دارای چندین بخش و عملیات مرتبط با آن است. برای نمونه بخش‌هایی مانند بخش وام، دارایی، بیمه و سایر موارد در این نرم افزار وجود دارند.

مدیریت همه بخش‌ها با استفاده از برنامه نویسی تابعی بسیار دشوار خواهد بود؛ زیرا هر بخش دارای توابع مختلفی است و در صورت بروز خطا، پیدا کردن بخش دارای خطا دشوار خواهد بود و همه نرم افزار را تحت تأثیر قرار می‌دهد. با استفاده از اصول برنامه نویسی شی گرا همه این موارد با کمک کلاس‌ها و اشیا مدیریت می‌شوند. می‌توان در این مثال بخش وام، بیمه و هر بخش دیگری را به عنوان یک کلاس در نظر گرفت و از این موضوع مطمئن بود که اگر خطایی مثلاً برای بخش وام ایجاد شد، کلاس‌های دیگر تحت تأثیر آن قرار نمی‌گیرند. در بخش بعدی از این مقاله، به بررسی مؤلفه‌های اصول برنامه نویسی شی گرا پرداخته شده است.

مؤلفه های برنامه نویسی شی گرا

اصول برنامه نویسی شی گرا برای طراحی این نوع از برنامه نویسی دارای مؤلفه‌هایی است که باید درک شوند. در این بخش به طور خلاصه به بررسی این مؤلفه‌های شی گرایی پرداخته شده است. به عنوان اولین مؤلفه، بخش بعدی به بررسی و شرح شی اختصاص دارد.



شی چیست؟

اشیا موجودیت‌هایی هستند که باعث می‌شوند کلاس‌ها در برنامه پیاده‌سازی شوند. اشیا ویژگی‌ها و رفتارهای کلاس‌ها را برای پیاده‌سازی ایجاد می‌کنند. به عبارت دیگر، اشیا نمونه‌هایی از کلاس هستند که با استفاده از تعریف داده‌ها ایجاد شده‌اند. آن‌ها می‌توانند به صورت اشیا دنیای واقعی یا موجودیت‌های انتزاعی در نظر گرفته شوند. کلاس مانند یک نقشه ساخت است که اشیای مختلفی را می‌توان بر اساس آن ایجاد کرد.

برای مثال، یک خودرو شیئی همراه با ویژگی‌هایی مانند رنگ، مدل، برند، نام، نوع سوخت و سایر موارد است و همچنین رفتارهایی مانند حرکت کردن دارد. بنابراین، این ویژگی‌ها و اشیا باعث به وجود آمدن شی خودرو می‌شوند. در بخش بعدی از این مقاله به بررسی مؤلفه کلاس در برنامه نویسی شی گرا پرداخته شده است.

کلاس در شی گرایی چیست؟

کلاس‌ها انواع داده‌ای هستند که «توسط کاربر تعریف می‌شوند (User Defined) «و به عنوان طرح اولیه برای اشیا، «ویژگی‌ها (Attribute) «و «متدها (Method) «مورد استفاده قرار می‌گیرند. به عبارت دیگر کلاس می‌تواند به عنوان نقشه ساختی برای اشیا در نظر گرفته شود. کلاس بیان می‌کند که اشیا چه وظایفی را می‌توانند انجام دهند و شامل همه ویژگی‌ها، رفتارهایی می‌شود که یک مدل می‌تواند استفاده کند. می‌توان گفت که یک کلاس شامل تعریف اشیا می‌شود. برای مثال، رنگ خودرو، نوع موتور خودرو و سایر موارد مثال‌هایی از کلاس هستند. طبق تعاریف بیان شده، می‌توان هر تعداد کلاسی که مورد نیاز برنامه است را تولید کرد و محدودیتی در ایجاد کلاس‌ها برای یک برنامه وجود ندارد. در بخش بعدی از مقاله «اصول برنامه نویسی شی گرا» به بررسی «متُدها» پرداخته شده است.

متد در شی گرایی چیست؟

»متدها (Method) «توابعی هستند که داخل یک کلاس تعریف می‌شوند و رفتارهای اشیا را توصیف می‌کنند. استفاده از هر متد موجود در تعریف کلاس، با ارجاع به شی نمونه شروع می‌شود. به علاوه «زیر روال‌های (Subroutine) «موجود در اشیا، «متدهای نمونه (Instance Method) «نامیده می‌شوند. برنامه نویس‌ها از متدها به منظور استفاده مجدد از کدها یا حفظ عملکرد کپسوله‌سازی درون یک شی در یک زمان استفاده می‌کنند. همچنین متدها می‌توانند حالت کلاس‌ها را تغییر

دهند. برای مثال، متد راندن یک خودرو، حالت خودرو را از پارک شده به در حال حرکت تغییر می‌دهد. در بخش بعدی از مقاله «اصول برنامه نویسی شی گرا» به تعریف نمونه در این نوع برنامه نویسی پرداخته شده است.

نمونه در شی گرایی چیست؟

نمونه در برنامه نویسی شی گرا به عنوان اعضای کلاس در نظر گرفته می‌شود. آن‌ها برخی از مقادیر مرتبط با اشیای کلاس را نگهداری می‌کنند. همچنین می‌توان به شئی نیز نمونه گفت که توسط کلاس ایجاد شده است. در برخی از منابع نمونه و شی را با یکدیگر هم‌معنی نیز در نظر می‌گیرند. در بخش بعدی از مقاله «اصول برنامه نویسی شی گرا» تعریف »ویژگی « (Attribute)در این نوع برنامه نویسی را یاد می‌گیریم.

ویژگی یا Attribute در برنامه نویسی شی گرا چیست؟

ویژگی‌ها در کلاس‌ها تعریف می‌شوند و وضعیت اشیا را نشان می‌دهند. اشیا دارای داده‌های ذخیره شده در بخش ویژگی‌ها هستند. ویژگی‌های هر کلاس فقط متعلق به آن کلاس است. ویژگی‌ها خصوصیات کلاس را نشان می‌دهند و وجه تمایز بین کلاس‌ها هستند. آن‌ها نباید با رفتار کلاس اشتباه گرفته شوند، رفتارها وظایفی است که اشیا کلاس آن‌ها را انجام می‌دهند. برای درک بهتر مؤلفه‌های ارائه شده در این بخش مثالی در ادامه شرح داده شده است.

مثال مؤلفه های اصول برنامه نویسی شی گرا

بر اساس رویکرد برنامه نویسی شی گرا در مثال مرتبط با بررسی ساختارها و بخش‌های مختلف یک تلویزیون، بخش‌های متفاوت و عملکرد آن به صورت زیر تعریف شده‌اند:

- نام شیء به عنوان «تلویزیون» در نظر گرفته می‌شود.
- ویژگی‌های تلویزیون ازجمله کانال، صدا، خاموش و روشن کردن و سایر موارد تعریف می‌شوند.
- ایجاد توابعی که عملکرد تلویزیون را به عنوان رفتارها مدیریت می‌کنند، می‌توان برای مثال به تنظیم کردن صدای تلویزیون یا سایر موارد مشابه اشاره کرد.
- در نهایت همه ویژگی‌ها در یک برنامه پیاده‌سازی می‌شوند.

در این مثال یک شیء با نام تلویزیون تعریف شده که از نوع کلاس تلویزیون است. سپس «ویژگی‌های (Property) «تلویزیون ازجمله انواع کانال‌ها، میزان صدای مختلف و سایر موارد در آن تعریف شده‌اند که به عنوان نمونه‌های کلاس در نظر گرفته می‌شوند. می‌توان توابعی ایجاد کرد که این ویژگی‌ها را تغییر بدهند و این توابع به عنوان متدهای تلویزیون شناخته شده‌اند. در نهایت به صورت کلی زمانی که این رویکرد در برنامه‌ای پیاده‌سازی شود، یک شی از کلاس تلویزیون ایجاد خواهد شد تا در برنامه اجرا شود. در ادامه شبه کدی (Pseudocode) برای درک بهتر این مثال ارائه شده است.

```
class Television{
int channel; //Instance
int volume; //Instance
boolean on = false;
```

```
.
.
void changeChannel(){} //Methods
void adjustVolume(){} //Methods
void switchOnOff(){} //Methods
.
.
}

Television tv = new Television(); //Television Object
```

در قطعه کدهای فوق، تلویزیون به عنوان کلاسی تعریف شده است که شامل نمونه‌هایی می‌شود که ویژگی‌های کلاس هستند. متدها به تغییر رفتار کلاس تلویزیون کمک می‌کنند. در نهایت یک شی از تلویزیون به نام `tv` ایجاد شده است که همه ویژگی‌ها و رفتارهای کلاس را دربرمی‌گیرد. (لینک برگشت)

جمع‌بندی

اصول برنامه نویسی شی گرا پرکاربردترین الگوی برنامه نویسی به حساب می‌آید و به توسعه برنامه‌های کاربردی در مقیاس بزرگ و کوچک کمک می‌کنند تا توسعه دهندگان بتوانند با استفاده از آن‌ها برنامه‌های کارآمدی بسازند. این اصول برنامه نویسی به تقویت و امنیت اشیا برنامه با استفاده از اصول قدرتمند انتزاع و کپسوله سازی کمک می‌کند. معروف‌ترین زبان‌های برنامه نویسی شی گرا شامل پایتون، جاوا، روبی و سایر موارد می‌شوند. در این مطلب سعی شد به طور کامل و همراه با مثال‌های کاربردی و شبه‌کدها انواع اصول برنامه نویسی شی گرا توصیف و مورد بررسی قرار بگیرد.