برنامه نویسی شی گرا چیست؟ – توضیح به زبان ساده

زمان مطالعه: ۱۷ دقیقه

به‌عنوان یک توسعه‌دهنده نرم‌افزار یا فردی که به‌دنبال یادگیری مهارت برنامه‌نویسی است، یکی از مباحثی که زیاد با آن رو به رو می‌شوید، مدل برنامه‌نویسی شی‌گرا است. در این شیوه رایج از برنامه‌نویسی، کدهای برنامه را به‌جای اینکه به‌صورت توابع یا دیگر ساختارهای منطقی بنویسند، در قالب اشیا بیان می‌کنند. **یک شی می‌تواند به‌صورت فیلد داده‌ای یا مؤلفه‌ای تعریف شود که خصوصیات و متدهای مخصوص به خود را دارد**. برنامه‌نویسی شی‌گرا که به OOP هم معروف است، به‌طور گسترده‌ای در توسعه نرم‌افزارها به‌کار می‌رود و همچنین، یکی از مهارت‌های مورد نیازی است که برای ورود به حرفه برنامه‌نویسی از سوی کارفرمایان مطرح می‌شود.

برنامه‌نویسی شی‌گرا مانند برنامه‌نویسی رویه‌ای، تابعی، دستوری و ... یکی از شیوه‌هایی است که برای توسعه نرم‌افزارها مورد استفاده قرار می‌گیرد. به‌طور مثال در برنامه‌نویسی رویه‌ای، برنامه در قالب «رویه‌هایی» (Procedures) نوشته می‌شود که یکدیگر را فراخوانی می‌کنند. در برنامه‌نویسی تابعی نیز برنامه در قالب ترکیب توابع توسعه داده می‌شود. برنامه‌نویسی شی گرا که در این مطلب مورد بحث قرار گرفته است، یکی از شیوه‌های توسعه برنامه‌های مدرن به‌شمار می‌رود و مزایای گوناگونی را در اختیار ما قرار می‌دهد. در این مطلب، مفاهیم شی‌گرایی نظیر شی، کلاس، متدها و ویژگی‌ها، و همچنین اصول آن همچون انتزاع، چندریختی، کپسوله‌سازی و وراثت را به زبان ساده و مثال‌های متعدد و واقعی به شما توضیح داده‌ایم.

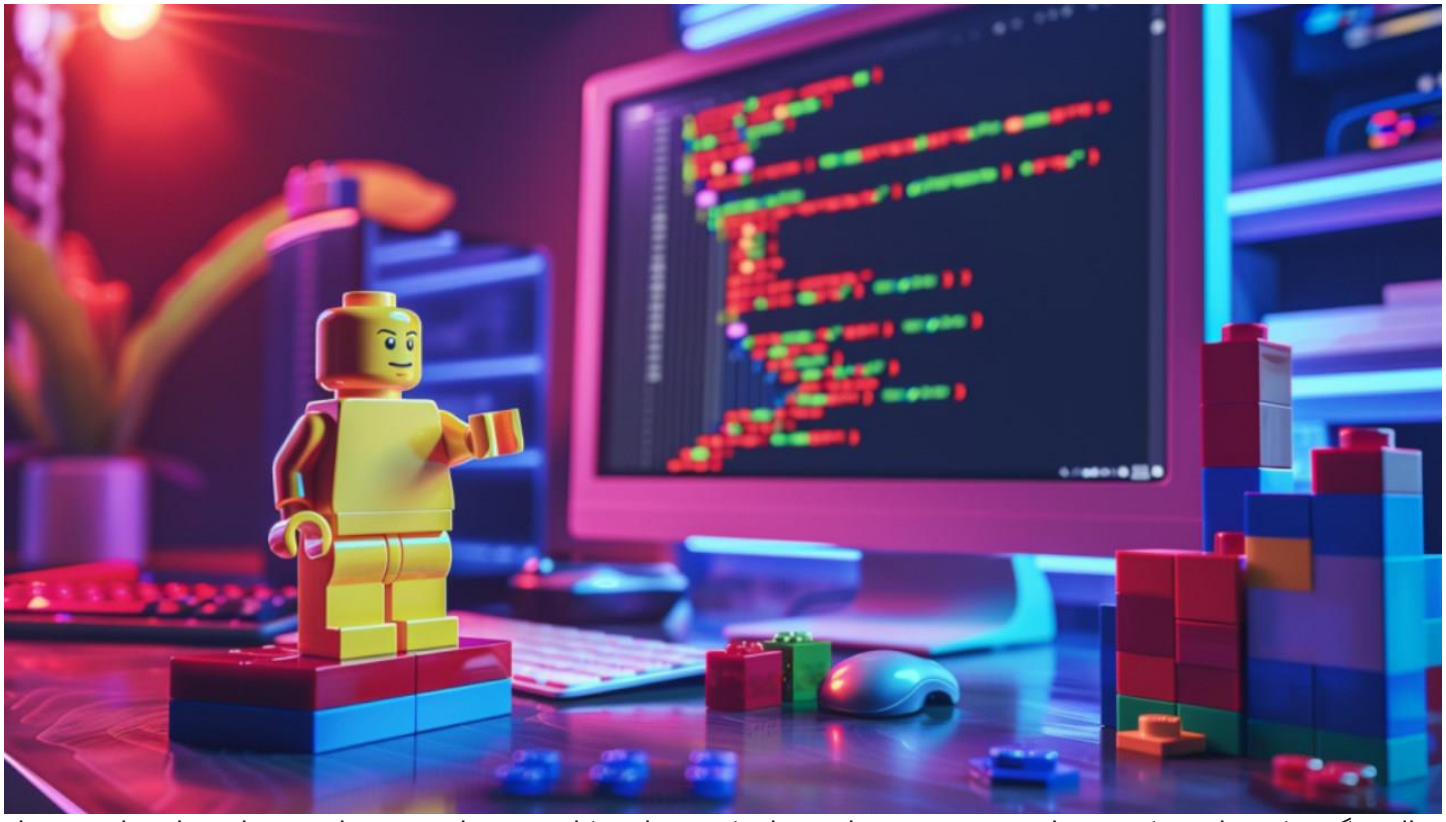
فهرست مطالب این نوشته

مفهوم برنامه‌نویسی شی‌گرا که نخستین بار توسط زبان برنامه‌نویسی سیمولا در اختیار برنامه‌نویسان قرار گرفت، شامل اجزایی همچون کلاس‌ها، اشیا، متدها و ویژگی‌هایی است که در کنار هم، ساختار این نوع برنامه‌نویسی را شکل می‌دهند. برنامه‌نویسی OOP همچنین، بر مبنای اصولی مانند کپسوله‌سازی، انتزاع، وراثت و چندریختی استوار است. بیشتر زبان‌هایی که امروز با آن‌ها آشنایی داریم از برنامه‌نویسی شی‌گرا در کنار برنامه‌نویسی رویه‌ای پشتیبانی می‌کنند اما در این میان برخی از زبان‌ها مانند روبی، اسکالا و غیره هماهنگی بیشتری با OOP دارند و با همه چیز مانند یک شی رفتار می‌کنند.

برنامه نویسی شی گرا چیست؟

برنامه‌نویسی شی‌گرا (Object-Oriented Programming | OOP)، یکی از مدل‌های رایج برای برنامه‌نویسی و توسعه اپلیکیشن‌های گوناگون محسوب می‌شود. در این روش، سورس‌کد برنامه را در قالب قطعه کدهایی کوچک یا همان «اشیا» (Objects) می‌نویسند. شما می‌توانید هر یک از این مؤلفه‌ها را به دفعات در طول برنامه خود استفاده کنید.

بازی خانه‌سازی یا لِگو را تصور کنید که در آن، قطعات کوچک و ساده‌ای را به هم چسبانده و اجسام معنادار، بزرگ‌تر و جالب‌تری می‌سازیم. در برنامه‌نویسی شی‌گرا نیز با چنین فرایندی رو به رو هستیم و با استفاده از اشیا می‌توانیم برنامه‌های پیچیده و حرفه‌ای را ایجاد کنیم. برای درک بهتر، شی را می‌توانید معادل چیزهایی در دنیای واقعی مانند یک «سگ»، «ماشین»، «درخت» یا هر مؤلفه دیگری تصور کنید که ضمن داشتن یک سری خصوصیات، می‌توانند اَعمال یا رفتارهایی هم داشته باشند. به‌طور مثال، یک «سگ» ضمن داشتن ویژگی «نژاد» و بسیاری خصوصیات دیگر، رفتارهایی مانند «دویدن» و رفتارهای متعدد دیگری را می‌تواند داشته باشد.



مثال دیگری که برای درک بهتر این مفهوم می‌توانیم بیان کنیم، بازی کامپیوتری است. در این نوع‌بازی‌ها به‌طور معمول چندین کاراکتر یا شخصیت وجود دارند که هر یک قابلیت‌های و توانایی‌های خود را دارد. به‌طور مثال، یک کاراکتر سیاه‌پوست که سلاح گرم دارد و دیگری، کاراکتری سفید پوست که از شمشیر برای جنگیدن استفاده می‌کند. این شخصیت‌ها را می‌توانیم معادل همان اشیا در برنامه‌نویسی شی‌گرا نظر بگیریم که هر کدام، خصوصیاتی مانند نام، سن، رنگ پوست و همچنین رفتارهایی همچون تیراندازی کردن، به‌کارگیری شمشیر و غیره را دارند.
با بیان این مثال‌ها، اکنون می‌دانیم که رویکرد برنامه‌نویسی شی‌گرا، کدهای برنامه‌مان را به‌شکلی نظم می‌دهد که در آن هر شی، ویژگی‌های خود را دارد و اعمال مشخصی را می‌تواند انجام دهد. این رویکرد باعث می‌شود تا سایر توسعه‌دهندگان، کدهای ما را راحت‌تر بفهمند و فرایند تغییر و به‌روزرسانی سورس‌کدها نیز ساده‌تر شود.

OOP، یکی از روش‌های مهم در برنامه‌نویسی و توسعه نرم‌افزارها به‌شمار می‌رود که قابلیت‌هایی نظیر آنچه در زیر آورده‌ایم را در اختیار شما قرار می‌دهد.

- امکان‌پذیر کردن مشارکت و کار تیمی در توسعه برنامه‌ها
- افزایش قابلیت نگهداری کدها و توسعه‌پذیری
- مدیریت پیچیدگی کدها و انتزاع
- چندریختی و انعطاف‌پذیری
- وراثت و قابلیت استفاده مجدد از کدها
- کپسوله‌سازی و محافظت از داده‌ها

نحوه کار برنامه‌نویسی شی گرا چیست؟

برای اینکه بتوانیم یک شی از نوع مورد نظر خود بسازیم نیاز به یک الگو داریم و در برنامه‌نویسی شی‌گرا به این الگو، Class گفته می‌شود. در واقع، این کلاس است که مشخص می‌کند یک شی، چه نوع خصوصیات و رفتارهایی داشته باشد. به‌طور مثال، اگر کلاس «وسیله نقلیه چهار چرخ» داشته باشیم، آنگاه بیشتر خودروهای اطراف ما می‌توانند شیئی از این کلاس باشند.

مثال واقعی برنامه نویسی شی گرا

تعدادی ماشین اسباب‌بازی را در نظر بگیرید. هر یک از این ماشین‌ها یک شی یا Object محسوب می‌شوند و هر یک از این اشیا بر مبنای الگویی ساخته شده‌اند که خصوصیات و رفتار ماشین‌ها را مشخص می‌کنند. این الگوها، در واقع همان کلاس‌های ما هستند که در اینجا نقش کارخانه ماشین‌های اسباب‌بازی را دارند.

به بیان دقیق‌تر، یک Class تعیین می‌کند که ماشین اسباب‌بازی ما، «۴ عدد چرخ» و ویژگی «رنگ بدنه» را به‌عنوان خصوصیات خود داشته باشد و از سویی دیگر به‌عنوان قابلیت‌هایش می‌تواند «به سمت جلو یا عقب حرکت کند». حال اگر ما یک شی

از این کلاس ایجاد کنیم، طبیعتاً ماشینی خواهد بود که ۴ عدد چرخ دارد و دارای رنگ بدنه‌ای است که ما تعیین کرده‌ایم و در نهایت، این شی می‌تواند به سمت جلو یا عقب حرکت کند.

مزیت برنامه نویسی شی گرا چیست؟

استفاده از رویکرد برنامه‌نویسی شی‌گرا مزایای زیادی را برای برنامه‌نویس به‌دنبال خواهد داشت که در ادامه به چند مورد اشاره کرده‌ایم.

- **قابلیت استفاده دوباره (Reusability):** «برنامه‌نویسی شی‌گرا به شما امکان می‌دهد تا اشیا را در بخش‌های گوناگون پروژه خود، به‌طور مجدد و به دفعات استفاده کنید. بدین‌ترتیب با پرهیز از نوشتن مجدد خیلی از کدها، در زمان و انرژی شما صرفه‌جویی می‌شود.
- **بهره‌وری (Efficiency):** «مدل برنامه‌نویسی شی‌گرا با سازمان‌دهی بهتر کدها، بازدهی برنامه ساخته شده را افزایش می‌دهد.
- **قابلیت نگهداری (Maintainability):** «کدهایی که با روش برنامه‌نویسی شی‌گرا نوشته می‌شوند را به طور معمول راحت‌تر می‌توانید تغییر دهید یا به‌روزرسانی کنید.

مفاهیم اصلی برنامه نویسی شی گرا چیست؟

برنامه‌نویسی شی‌گرا، دارای چندین مفهوم پایه‌ای است که در ادامه فهرست کرده‌ایم.

- Class-ها و Object ها
- انتزاع
- کپسوله‌سازی
- وراثت
- چندریختی

هر یک از این مفاهیم را در ادامه و به زبانی ساده توضیح داده‌ایم.

کلاس و شی در برنامه نویسی شی گرا

Class-ها را می‌توانید مانند الگوهایی برای ایجاد یک شی تجسم کنید که شامل اطلاعاتی در مورد نحوه تولید آن‌ها هستند.

انتزاع

مفهوم «انتزاع» (Abstraction) در برنامه‌نویسی، به توسعه‌دهنده کمک می‌کند تا به‌جای تلاش برای درک جزئیات پیچیده، توجه خود را روی انجام کارهای اصلی و مهم‌تر بگذارد و از این طریق، به‌شکل خیلی ساده‌تری بتواند کدها را درک کرده و از آن‌ها استفاده کند.

برای درک بهتر مفهوم «انتزاع» یک مثال واقعی می‌زنیم. فرض کنید با ریموت کنترل تلویزیون، صدای آن را کم می‌کنید. در این حال، کار مورد نظر شما انجام می‌شود و لزومی هم ندارد که از جزئیات کار صورت گرفته یا ساز و کار درونی ریموت کنترل اطلاعی داشته باشید.

کپسوله سازی

برای درک بهتر «کپسوله‌سازی» (Encapsulation) یک مثال می‌زنیم. فرض کنید که جواهرات ارزشمند خود را درون جعبه‌ای قرار داده و درب آن را قفل می‌کنید. به‌طوری‌که تنها افراد خاصی می‌توانند به آن‌ها دسترسی داشته باشند. در برنامه‌نویسی نیز به‌کمک کپسوله‌سازی، جلوی دسترسی‌های غیر مجاز به داده‌ها را می‌گیریم. با این کار، ضمن افزایش امنیت، خوانایی کدها نیز بیشتر می‌شود.

وراثت

همان‌گونه که در دنیای واقعی، برخی صفات از «والدین» (Parents) به «فرزندان» (Children) منتقل می‌شوند، در برنامه‌نویسی شی‌گرا نیز کلاس‌ها می‌توانند خصوصیات و متدهای دیگر کلاس‌ها را به ارث ببرند. به این مفهوم، ارث‌بری یا «وراثت» (Inheritance) گفته می‌شود.

چندریختی

چندشکلی یا «چندریختی» در برنامه‌نویسی شی‌گرا به این معنی است که اشیا در شرایط گوناگون می‌توانند، رفتار متفاوتی را از خود نشان دهند.

شی در برنامه نویسی شی گرا چیست؟

»شی (Object) «یا آبجکت به هر مؤلفه‌ای گفته می‌شود که ۲ مورد زیر را در خود داشته باشد.

- **ویژگی‌ها :**خصوصیات شی را بیان می‌کنند.
- **متدها :**بیان‌گر رفتار شی هستند یا کارهایی که شی می‌تواند انجام دهد.

به‌طور مثال، می‌توانیم یک اتومبیل در دنیای واقعی را یک شی در نظر بگیریم. این شی ویژگی‌هایی مانند رنگ بدنه، اتومات یا دستی بودن گیربکس، حداکثر سرعت و غیره را دارد و در کنار آن، رفتارها یا «متدهایی (Methods) «مانند روشن شدن، تغییر دنده، شتاب گرفتن و غیره را فراهم می‌کند.

کلاس در برنامه نویسی شی گرا چیست؟

همان‌طور که پیش‌تر هم توضیح دادیم، «کلاس» (Class) در برنامه‌نویسی شی‌گرا مانند یک طرح کلی یا یک الگو عمل کرده و شی از روی آن ساخته می‌شود. به بیان دقیق‌تر، یک کلاس بیان می‌کند که یک شی چه نوع خصوصیات و متدهایی داشته باشد.
برای درک بهتر، کلاس دیگری به‌نام «گربه» را تصور کنید. این کلاس خصوصیاتی مانند سن، نژاد، نام و غیره دارد که ویژگی‌های آن محسوب می‌شوند. از سوی دیگر متدهایی نظیر صدای میو، خوابیدن، غذا خوردن و غیره را در بر می‌گیرد.

اکنون می‌خواهیم یک شی جدید از این کلاس ایجاد کنیم. خوب است بدانید که به این فرایند، نمونه‌سازی می‌گویند و ما در حقیقت یک «نمونه» (Instance) از کلاس مورد نظر ساخته‌ایم. نمونه جدید از کلاس «گربه» می‌تواند خصوصیات مخصوص به خود را داشته باشد و رفتارهای مشخص شده را انجام دهد.

مثال کلاس و شی در برنامه نویسی شی گرا چیست؟

اکنون، می‌خواهیم ۲ مفهوم کلاس و شی را با مثالی جالب برایتان توضیح دهیم. همچنین کدهای پیاده‌سازی این مثال، در زبان برنامه‌نویسی پایتون را نیز آورده‌ایم.
فرض کنید یک «عصای جادویی (Magic Wand) «شبیه به آنچه که در فیلم‌های هری‌پاتر دیده‌اید به شما داده‌اند و این وسیله با قدرت جادویی خود می‌تواند کارهای خارق‌العاده‌ای را انجام دهد. اکنون می‌خواهیم از این مثال برای توضیح مفاهیم کلاس و اشیا در برنامه‌نویسی شی‌گرا استفاده کنیم. شی یا آبجکت را می‌توانیم به‌عنوان یک عصای جادویی در نظر بگیریم که خصوصیات و قابلیت‌هایی دارد. به‌طور مثال، این عصا «۱۰۰ سانتی‌متر طول دارد» و می‌تواند «چیزهای گوناگونی را ناپدید کند». موارد گفته شده در واقع، خصوصیت و رفتار شی عصا محسوب می‌شوند.
حالا یک مدرسه جادویی مانند مدرسه هری‌پاتر را تصور کنید. در این مدرسه، کارگاهی به‌نام Wand وجود دارد که تمام عصاهای جادویی در آن ساخته می‌شوند. می‌توانیم کارگاه Wand را به عنوان یک Class در نظر بگیریم که نحوه ساخت این عصاهای جادویی را در اختیار دارد .بنابراین، اکنون می‌دانیم که یکClass ، الگو یا ارائه دهنده قوانینی برای ساخت اشیا است .در مثالی که بیان کردیم، کلاس «Wand» در واقع، قوانین یا دستورات مربوط به ساخت آبجکت‌ها یا عصاهای جادویی را بیان می‌کند و می‌دانیم که هر عصا، ویژگی‌ها و عملکردهای مخصوص به خود را دارد.

پیاده سازی شی و کلاس

در این قسمت، با استفاده از کلاس Wandیک شی به‌نام my-wandمی‌سازیم. همان‌طور که گفته شد، اشیا و کلاس‌های مورد بحث در اینجا، در واقع همان عصاهای جادویی و کارگاه ساخت عصا هستند-Class .ها نحوه ساخت عصاهای جادویی را مشخص می‌کنند وObject ، بیان‌گر یک عصای واقعی است که خصوصیات و توانمندی‌های مخصوص به خود را دارد. بدیهی است که می‌توان اشیا یا عصاهای مختلفی را با این کلاس تولید کرد که هر کدام، ویژگی‌های مخصوص به خود (مثل رنگ، اندازه و غیره) و همچنین عملکردهای مشخص یا همان متدها را دارند.

کدهای این بخش به زبان پایتون را در ادامه آورده‌ایم.

```
class Wand:
    def __init__(self, color, length):
        self.color = color
        self.length = length

    def cast_spell(self):
        print("The magic wand works".)

 #Create a wand object
my_wand = Wand("Purple", 10)
 #Use the wand's method
my_wand.cast_spell()
```

خروجی این کدها پس از اجرا به‌صورت زیر خواهد بود.

```
The magic wand works.
```

کدهای این برنامه را در ادامه توضیح داده‌ایم.

- **خطوط شماره ۱ تا ۷ :**این خطوط، شامل تعریف و بدنه کلاس Wand هستند و تعریف ۲ خصوصیت و یک رفتار این کلاس را در بر می‌گیرند.
- **خطوط شماره ۳ و ۴ :**در این خطوط، ۲ ویژگی عصا رنگ و طول یعنی color و length تعریف شده‌اند.
- **خط شماره ۶ :**یک متد برای این کلاس تعریف کرده‌ایم که cast_spell نام دارد و در هنگام اجرا، رشته The magic wand works. را در خروجی چاپ می‌کند.

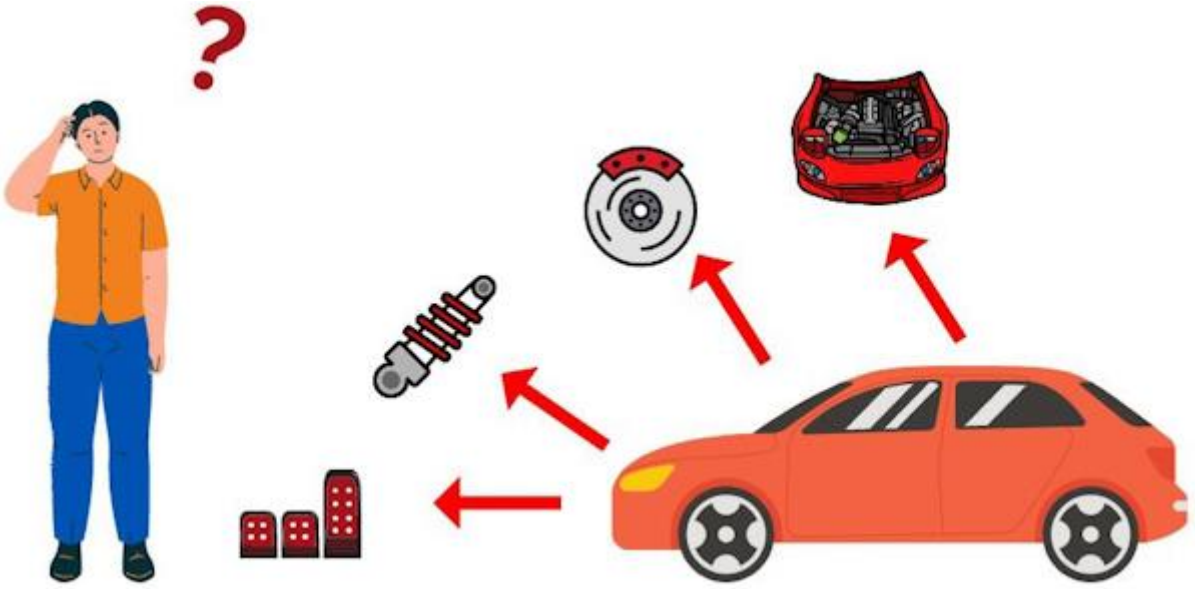
اکنون کلاس ما که کارگاه ساخت عصای جادویی است ساخته شد.

- **خط شماره ۱۰ :**یک نمونه یا Instance از کلاس Wand می‌سازیم و ۲ ویژگی رنگ و طول را برای عصای جدید مشخص می‌کنیم.
- **خط شماره ۱۲ :**اکنون که شی خود یعنی my_wand را از کلاس Wand ساخته‌ایم، می‌توانیم متد cast_spell() را از آن صدا بزنیم تا رفتار جادویی تعیین شده - یا همان چاپ رشته - انجام شود.

انتزاع در برنامه نویسی شی گرا چیست؟

به زبان ساده، «انتزاع داده‌ها» (Data Abstraction) را می‌توان مخفی‌سازی جزئیات غیرضروری از کاربر دانست .به‌صورتی‌که تنها اطلاعات مهم برای او نمایش داده شود و از این طریق، از پیچیدگی‌های سیستم دور بماند و روی قسمت‌های مهم‌تر تمرکز کند.

دستگاه ATM را می‌توان یک مثال واقعی از انتزاع در نظر گرفت. به‌طور مثال، برای دریافت وجه نقد به دستگاه ATM مراجعه می‌کنید و کارهایی مانند ارائه کارت خود به دستگاه، وارد کردن رمز عبور، وارد کردن مبلغ درخواستی و غیره را انجام می‌دهید. سپس دستگاهATM ، مبلغ مورد نظر را به شما تحویل می‌دهد یا اینکه در صورت بروز مشکل در هر یک از مراحل، این مورد را با پیامی به شما اطلاع می‌دهد. بدین‌ترتیب، جزئیات غیرضروری این فرایند نظیر ارتباط دستگاه ATM با شبکه بانکی، بررسی رمز عبور و معتبر بودن کارت، بررسی اطلاعات مربوط به حساب بانکی و غیره از شما پنهان شده و تنها با عملیات اصلی و ضروری سر و کار دارید.
به‌عنوان مثالی دیگر، اگر بخواهیم انتزاع داده‌ها را در مورد اتومبیل‌ها بیان کنیم، پنهان‌سازی جزئیات داخلی خودرو از جمله موتور خودرو، جعبه‌دنده، لنت‌ها و غیره، این مفهوم را به‌خوبی نشان می‌دهد. به بیان دیگر، انتزاع داده‌ها در مثال خودرو یعنی اینکه ما بدون اطلاع ازجزئیات فنی آن، از خودرو استفاده می‌کنیم.



پیاده سازی انتزاع

کلاس Carدر مثال زیر، مواردی مانند موتور خودرو و جزئیات سایر قسمت‌های آن را انتزاع می‌کند. یعنی کاربران می‌توانند بدون آگاهی از جزئیات فنی، خودرو را استارت بزنند و روشن کنند.

```
class Car:
    def __init__(self, brand, color):
        self.brand = brand
        self.color = color

    def start(self):
        print("The " + self.color + " " + self.brand + " car starts.")

    def drive(self):
        print("The " + self.color + " " + self.brand + " car is driving.")

my_car = Car("Tesla", "red")
my_car.start()
my_car.drive()
```

خروجی این کدها پس از اجرا به‌صورت زیر خواهد بود.

```
The red Tesla car starts.
The red Tesla car is driving.
```

در این خروجی، ابتدا پیام «خودروی قرمزرنگ تسلا روشن شده است» به اطلاع کاربر می‌رسد و سپس، پیام «خودروی قرمزرنگ تسلا در حال رانندگی است» را چاپ می‌کنیم.
توضیح کدهای این برنامه را در ادامه آورده‌ایم.

- **خطوط شماره ۳ و ۴ :**در این خطوط، ۲ ویژگی colorو brandرا برای کلاس Carتعریف کرده‌ایم که به ترتیب، رنگ و برند خودرو را نشان می‌دهند.
- **خط شماره ۶ :**این خط شامل تعریف متد startاست که عمل روشن کردن خودرو را با چاپ رشته‌ای به اطلاع کاربر می‌رساند.
- **خط شماره ۹ :**متد driveرا در این خط تعریف کرده‌ایم که رانندگی و حرکت کردن را با چاپ یک پیام به کاربر اطلاع می‌دهد.
- **خط شماره ۱۲ :**با نمونه‌سازی از کلاس Carیک شی به‌نام my_carساخته‌ایم که بیان‌گر یک خودروی Teslaقرمزرنگ است.

کپسوله سازی در برنامه نویسی شی گرا چیست؟

مفهوم «کپسوله‌سازی» (Encapsulation) به این معنی است که داده‌ها و متدها در محفظه‌ای امن قرار گرفته‌اند و تنها به‌واسطه یک اینترفیس یا رابط عمومی می‌توان به آن‌ها دسترسی داشت.

پیاده سازی کپسوله سازی

کلاس Car در کدهای برنامه زیر، ویژگی‌هایی نظیر برند، رنگ بدنه و میزان سوخت خودرو را محصور یا به اصطلاح کپسوله می‌کند. در این حالت، با اینکه می‌توان عمل سوخت‌گیری و رانندگی را انجام داد اما اطلاع از میزان سوخت، به متدهای تعریف شده محدود شده است.

```
class Car:
    def __init__(self, brand, color):
        # Initialize car with brand, color, and full fuel tank (fuel_level = 100)
        self.brand = brand
        self.color = color
        self.fuel_level = 100

    def refuel(self):
        # Refuel the car, setting the fuel level back to 100
        self.fuel_level = 100
        print("The " + self.color + " " + self.brand + " car has been refueled.")

    def drive(self):
        # Check if the car has fuel
        if self.fuel_level > 0:
            # If fuel level is greater than 0, drive the car and reduce fuel level by 10
            self.fuel_level -= 10
            print("The " + self.color + " " + self.brand + " car is driving.")
        else:
            # If out of fuel, print a message indicating so
            print("The car is out of fuel!")

# Create a Car object
my_car = Car("Toyota", "blue")

# Refuel the car and drive it
my_car.refuel()
my_car.drive()
```

اجرای این برنامه، خروجی زیر را به‌دنبال خواهد داشت.

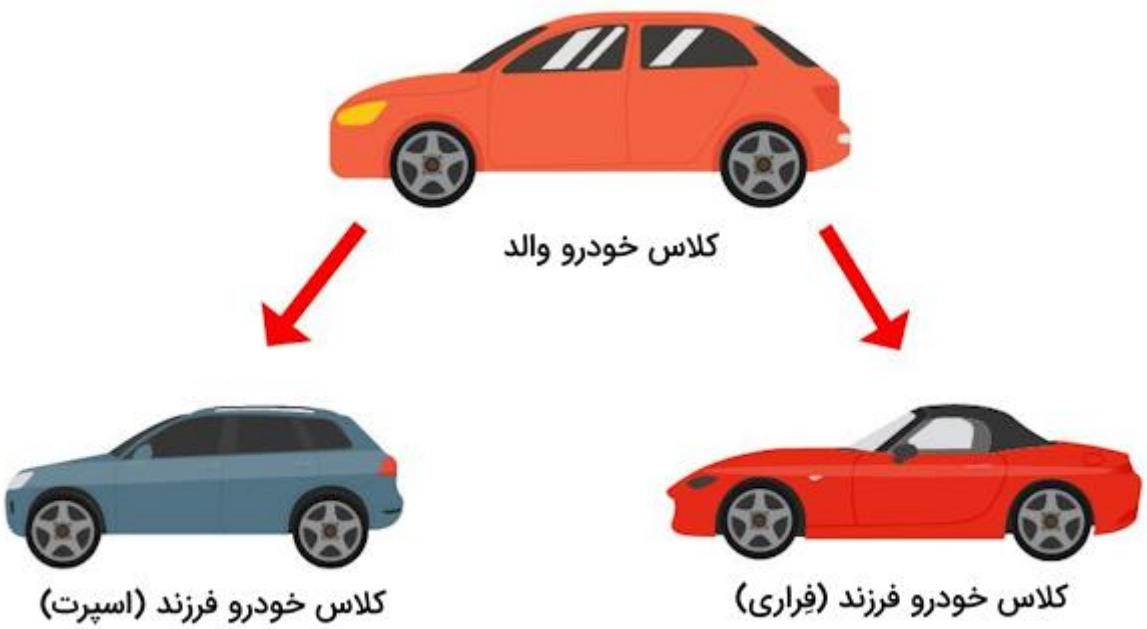
```
The blue Toyota car has been refueled.
The blue Toyota car is driving.
```

در این خروجی‌ها ابتدا عمل سوخت‌گیری با پیام «خودروی تویوتای آبی‌رنگ سوخت‌گیری شد» به اطلاع کاربر می‌رسد و سپس پیام «خودروی تویوتای آبی‌رنگ در حال رانندگی است» چاپ می‌شود.
توضیح مربوط به این کدها را در ادامه آورده‌ایم.

- **خطوط شماره ۴ تا ۶ :** در این خطوط، خصوصیات برند، رنگ و مخزن سوخت پُر (با مقدار ۱۰۰) برای کلاس Car تعریف شده است.
- **خطوط شماره ۸ تا ۱۱ :** این خطوط بدنه متد refuel را نشان می‌دهد. با توجه به عملکرد این متد که سوخت‌گیری خودرو و پر شدن باک آن با مقدار ۱۰۰ است، مقدار ویژگی fuel_level را با ۱۰۰ مقداردهی کرده و سپس این موضوع را با چاپ پیام در خروجی به اطلاع کاربر می‌رساند.
- **خطوط شماره ۱۳ تا ۲۱ :** این خطوط، بیان‌گر بدنه متد drive هستند که هنگام فراخوانی، میزان سوخت فعلی خودرو را بررسی می‌کند. اگر میزان سوخت بیش از صفر باشد آنگاه، ۱۰ واحد از آن کم کرده و انجام عمل رانندگی را با چاپ پیغامی در خروجی به اطلاع کاربر می‌رساند. اما اگر سوختی باقی نمانده باشد آنگاه، برای کاربر چاپ می‌کند سوخت خودرو تمام شده است.
- **خط شماره ۲۴ :** یک نمونه با نام my_car از کلاس Car ساخته و ویژگی‌های برند تویوتا و رنگ آبی را برای آن تعیین می‌کند.
- **خطوط شماره ۲۷ و ۲۸ :** در این خطوط، متدهای refuel و drive از شی my_car فراخوانی می‌شوند.

وراثت در برنامه نویسی شی گرا

می‌دانیم که وراثت در معنای کلی، به انتقال یک سری صفات از نسلی به نسل دیگر اشاره دارد. در برنامه‌نویسی شی‌گرا نیز این مفهوم وجود دارد و ما می‌توانیم کلاس‌های جدیدی را ایجاد کنیم که دارای ویژگی‌ها و متدهایی از یک کلاس اصلی یا والد به هستند. در این حالت، کلاس والد به‌عنوان «Superclass» یا کلاس اصلی شناخته می‌شود و به کلاسی که از والد، خصوصیات یا متدهایی را به ارث می‌برد نیز «Subclasses» یا زیرکلاس یا کلاس فرعی می‌گویند. به فرایند که طی آن یک کلاس جدید ساخته می‌شود و در واقع، خصوصیات یا متدهایی را از کلاس موجود به ارث می‌برد، Specialization می‌گویند.
برای نمونه، اجازه دهید همان مثال خودرو را بیان کنیم. قصد داریم تا خودروهای منحصر به فردی ایجاد کنیم که خصوصیات و رفتارهای مشترک و مشابهی را از کلاس اصلی به ارث می‌برند. از این طریق می‌توانیم انواع گوناگونی از خودروها را با ویژگی‌های مخصوص به هریک، تعریف کنیم.



پیاده سازی وراثت

در این مثال، یک کلاس اصلی به‌نام Car، به‌عنوان کلاس والد داریم. قرار است تا کلاس‌های SportsCar و SUV به‌عنوان کلاس‌های فرزند، خصوصیاتی را از این کلاس والد به ارث ببرند. هر کدام از این زیرکلاس‌ها در عین حال که خصوصیات و متدهای مخصوص به خود را می‌توانند داشته باشند، متد start و خصوصیات برند و رنگ را نیز از کلاس والد به ارث می‌برند.

```
class Car:
    def __init__(self, brand, color):
        self.brand = brand
        self.color = color

    def start(self):
        print("The " + self.color + " " + self.brand + " car starts.")

class SportsCar(Car):
    def accelerate(self):
        print("The " + self.color + " " + self.brand + " sports car accelerates quickly.")

class SUV(Car):
    def off_road(self):
        print("The " + self.color + " " + self.brand + " SUV tackles rough terrains with ease.")

my_sports_car = SportsCar("Ferrari", "red")
my_sports_car.start()
my_sports_car.accelerate()
```

```
my_suv = SUV("Jeep", "black")
my_suv.start()
my_suv.off_road()
```

خروجی این کدها، به‌صورت زیر خواهد بود.

```
The red Ferrari car starts.
The red Ferrari sports car accelerates quickly.
The black Jeep car starts.
The black Jeep SUV tackles rough terrains with ease.
```

توضیحات مربوط به کدهای این برنامه را در ادامه آورده‌ایم.

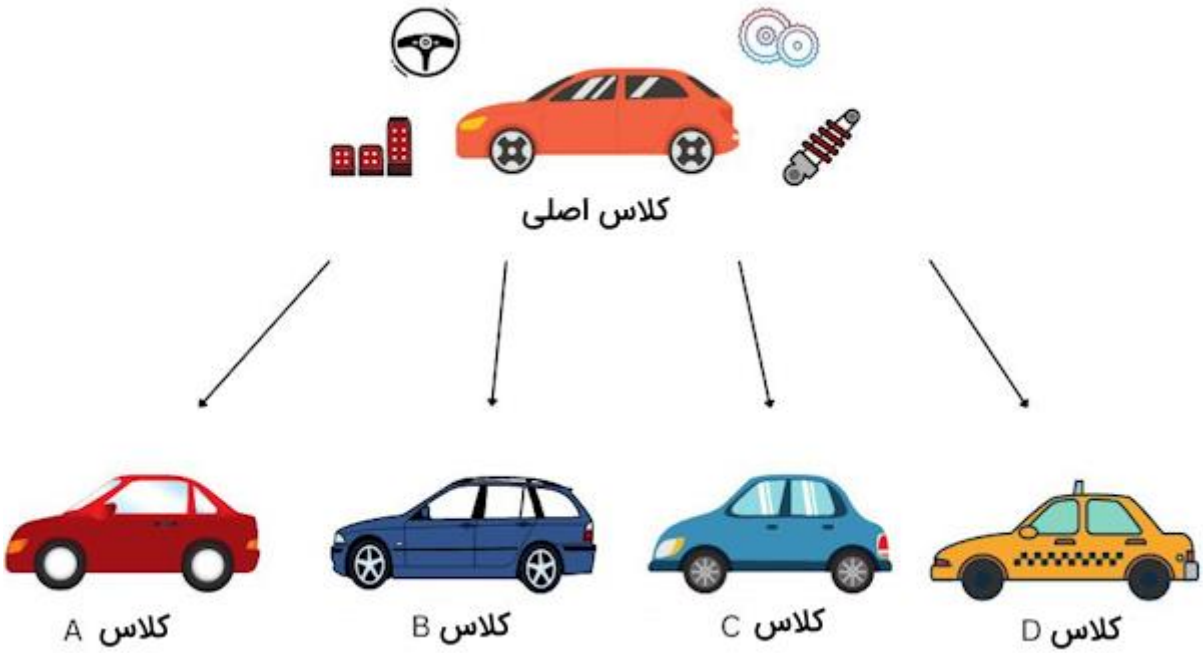
- **خطوط شماره ۱ تا ۷ :** در این خطوط، بدنه کلاس اصلی یا والد یعنی Car تعریف شده است. این کلاس علاوه بر ۲ خصوصیت برند و رنگ، دارای متدی به‌نام start است که در صورت فراخوانی، رشته‌ای را برای کاربر چاپ می‌کند.
- **خطوط شماره ۹ تا ۱۱ :** در این خطوط، بدنه کلاس فرزند SportsCar تعریف شده است. متد این کلاس، accelerate نام دارد و در صورت فراخوانی، افزایش سرعت خودروی اسپورت را با پیامی به اطلاع کاربر می‌رساند.
- **خطوط شماره ۱۳ تا ۱۵ :** بدنه کلاس فرزند SUV را در این خطوط مشاهده می‌کنید که دارای متدی به‌نام off_road است و با فراخوانی آن، در قالب پیامی به کاربر می‌گوید که این خودرو به‌سادگی، ناهمواری‌های زمین را پشت سر می‌گذارد.

توجه داشته باشید که هر دوی این خودروها، متد start را از کلاس Car به ارث برده‌اند.

- **خطوط شماره ۱۷ تا ۱۹ :** در این خطوط، نمونه‌سازی از کلاس SportsCar انجام شده است و متدهای start و accelerate از شی جدید یعنی my_sports_car یا همان فِراری قرمزرنگ، اجرا می‌شوند. توجه داشته باشید که متد start، از والد این شی به ارث رسیده است.
- **خطوط شماره ۲۱ تا ۲۳ :** در این خطوط نیز، یک شی جدید یعنی «جیپ مشکی‌رنگ» از کلاس SUV به‌نام my_suv ایجاد می‌شود و پس از اجرای متد start که از کلاس والد یعنی Car به ارث رسیده، متد off_road را از آن اجرا کرده‌ایم. هر کدام از این متدها پیامی را چاپ می‌کنند و برای این کار از خصوصیات به ارث رسیده استفاده می‌کنند.

چندریختی در برنامه نویسی شی گرا

یکی از موارد جالبی که در رویکرد برنامه‌نویسی شی‌گرا شاهد آن هستیم، پلی‌مورفیسم یا «چندریختی» (Polymorphism) است که طبق آن، اشیا می‌توانند شکل‌های گوناگونی به‌خود بگیرند. به بیان دیگر، اگر چندین کلاس با کلاس والد یکسانی داشته باشیم، آن‌گاه اشیای آن‌ها می‌توانند به‌جای یکدیگر استفاده شوند. اگر دوباره به مثال خودروها برگردیم، چندریختی را می‌توانیم به این صورت بیان کنیم که ما از طریق رابط یا اینترفیسی یکسان، آبجکت‌های مختلفی که از خودروها ساخته‌ایم را به‌شکلی در نظر می‌گیریم که گویا همگی از یک نوع هستند.
تصویر زیر این مورد را به‌خوبی نشان می‌دهد.



پیاده سازی چندریختی

در کدهای برنامه‌ای که در این قسمت آورده‌ایم، کلاس `Car` را مشاهده می‌کنید که اینترفیسی را با متد `drive` تعریف می‌کند. کلاس‌های `Sedan` و `Convertible` از این کلاس ارث‌بری کرده و پیاده‌سازی خود را از متد `drive` انجام داده‌اند. در ادامه آن، هر ۲ نمونه ایجاد شده از خودروها را در یک ساختمان داده لیست قرار داده‌ایم و با پیمایش روی آن‌ها، متد `drive` موجود در هر یک را صدا می‌زنیم.

```
class Car:
    def __init__(self, brand, color):
        self.brand = brand
        self.color = color

    def drive(self):
        # Raise NotImplementedError if drive method is not implemented in subclass
        raise NotImplementedError(
            "drive method must be implemented in each car subclass")


class Sedan(Car):
    def drive(self):
        # Override drive method for Sedan subclass
        print("The", self.color, self.brand, "sedan is driving smoothly.")


class Convertible(Car):
    def drive(self):
        # Override drive method for Convertible subclass
        print("The", self.color, self.brand,
              "convertible is driving with the top down.")


# Create instances of Sedan and Convertible
my_sedan = Sedan("Honda", "silver")
my_convertible = Convertible("BMW", "blue")

cars = [my_sedan, my_convertible]

# Iterate through cars list and call drive method for each car
for car in cars:
    car.drive()
```

اجرای این کدها به‌صورت زیر خواهد بود.

```
('The', 'silver', 'Honda', 'sedan is driving smoothly.')
('The', 'blue', 'BMW', 'convertible is driving with the top down.')
```

توضیح کدهای این برنامه را در ادامه آورده‌ایم.

- **خطوط شماره ۱ تا ۹ :** در این خطوط، بدنه کلاس Car تعریف شده است. این کلاس، ۲ خصوصیت «برند» و «رنگ» و همچنین متدی با نام drive را در بر می‌گیرد. کلاس Car را به‌عنوان کلاس والد در نظر می‌گیریم.
- **خطوط شماره ۱۲ تا ۱۵ :** در این خطوط، کلاس Sedan را به‌صورت زیرکلاس یا کلاس فرزند Car تعریف کرده‌ایم. پارامتر ارسالی به آن در واقع، نام کلاسی است که مواردی را از آن به ارث می‌برد. متد drive را در این کلاس، بازنویسی کرده‌ایم. این متد هنگام فراخوانی، رشته‌ای را برای کاربر چاپ می‌کند که محتوای خصوصیات به ارث برده از والد خود است.
- **خطوط شماره ۱۸ تا ۲۲ :** در این خطوط، کلاس Convertible را نیز به‌صورت کلاس فرزند Car تعریف کرده‌ایم. و پارامتر Car کلاس والد آن را مشخص می‌کند. متد drive را در این کلاس فرزند بازنویسی کرده‌ایم. این متد هنگام فراخوانی، رشته‌ای را برای کاربر چاپ می‌کند.
- **خطوط شماره ۲۶ و ۲۷ :** از ۲ کلاس Sedan و Convertible نمونه‌سازی کرده‌ایم و به‌ترتیب در my_sedan و my_convertible قرار داده‌ایم.
- **خط شماره ۲۹ :** لیستی به‌نام cars شامل اشیای my_sedan و my_convertible ساخته‌ایم.
- **خطوط شماره ۳۲ و ۳۳ :** روی لیست cars پیمایش می‌کنیم و متد drive را از آن‌ها فراخوانی می‌کنیم.

جمع‌بندی

در این مطلب از مجله فرادرس، با یکی از مهم‌ترین رویکردهای برنامه‌نویسی یعنی برنامه‌نویسی شی‌گرا آشنا شدیم و برای درک بهتر این مبحث، مثال‌های واقعی متعددی را بیان کردیم. با یادگیری برنامه‌نویسی شی‌گرا و انجام تمرین‌های عملی منظم می‌توانید به‌سادگی بر این شکل از برنامه‌نویسی مسلط شوید.

دانستن اینکه برنامه نویسی شی گرا چیست، از این بابت حائز اهمیت است که مزایای متعددی را برای شما فراهم می‌کند. ضمن اینکه یکی از مهارت‌های مهمی به‌شمار می‌رود که کارفرمایان، هنگام جذب برنامه‌نویس نرم‌افزار، به‌دنبال آن هستند. رویکرد برنامه‌نویسی شی‌گرا به شما کمک می‌کند تا سیستم‌های بزرگ را در قالب اشیا بیان کنید. این طراحی ماژولار باعث می‌شود تا مدیریت کدها یعنی درک، نگهداری و تغییر آن‌ها بسیار آسان‌تر شود.

منبع