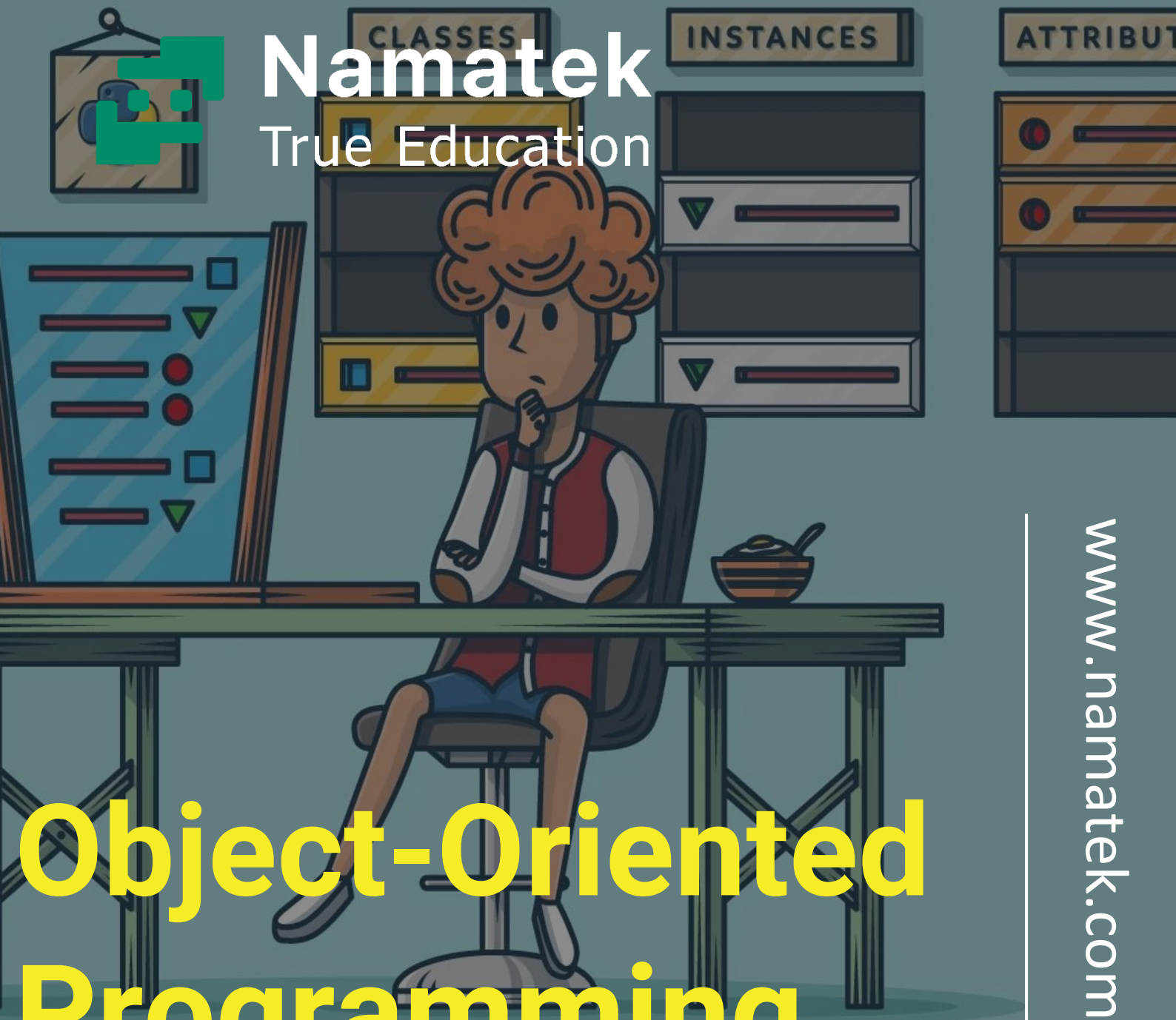
Namatek
True Education



Object-Oriented Programming

برنامه نویسی شیء گرا

www.namatek.com

فهرست مطالب

برنامه نویسی شیء گرا همانگونه که از نام آن پیدا است، به برنامه نویسی‌های مدل سازی‌ای اطلاق می‌شود که از اشیا برای ساختار کد استفاده می‌کنند. هدف استفاده از این نوع برنامه نویسی‌ها، ترکیب مواردی که در دنیای واقعی وجود دارد با مفاهیمی همچون وراثت، کپسولاسیون و چند شکلی است. در این مقاله به بررسی برنامه نویسی شیء گرا، ساختار، مفاهیم، اصول، برترین زبان‌های برنامه نویسی شیء گرا، نکات مهم در برنامه نویسی شیء گرا و مزایا و معایب آن خواهیم پرداخت.

برنامه نویسی شیء گرا چیست؟

برنامه نویسی شیء گرا یا OOP که مخفف عبارت Object – Oriented Programming است، به عنوان یک پارادایم برنامه نویسی و نه یک زبان خاص تعریف می‌شود. این نوع برنامه، یک ابزار قدرتمند است که به توسعه دهندگان این امکان را می‌دهد تا برنامه‌هایی را با مدل سازی اشیای واقعی ایجاد کنند. با استفاده از اشیا، کلاس‌ها و ساختارهای داده، توسعه دهندگان می‌توانند برنامه‌های کاربردی قوی، کارآمد و کاربردی ایجاد کنند. برنامه

نویسی شیء گرا، یک الگوی برنامه نویسی در علوم کامپیوتر است که طراحی نرم افزار را حول داده‌ها یا اشیا به جای توابع و منطق سازماندهی می‌کند. یک شی را می‌توان به عنوان یک فیلد داده تعریف کرد که دارای ویژگی‌ها و رفتار منحصر به فردی است. برنامه نویسی شیء گرا در مورد مدل سازی یک سیستم به عنوان مجموعه ای از اشیا است که در آن، هر شیء نمایانگر جنبه خاصی از سیستم است. اشیا، شامل توابع یا متدها و داده‌ها است. شیء، یک رابط عمومی برای کد دیگری را فراهم می‌کند که می‌خواهد از آن استفاده کند؛ اما حالت خصوصی و داخلی خود را حفظ کند. نیازی نیست که دیگر بخش‌های سیستم به هر آنچه که داخل شیء می‌گذرد، اهمیت دهند.

ساختار

برنامه نویسی شیء گرا شامل ساختارهای مختلفی است که به عنوان بلوک‌های سازنده آن شناخته می‌شوند و عبارت اند از:

- **کلاس:** کلاس نوعی داده است که چارچوبی را به منظور ایجاد اشیا فراهم می‌کند. کاربر می‌تواند یک کلاس به منظور ایجاد چندین شیء بدون نیاز به نوشتن کدهای اضافی تعریف کند.
- **Object:** در برنامه نویسی شیء گرا، یک شیء نشان دهنده یک نمونه یا ایجاد یک کلاس است. آبجکت‌ها، داده‌های خاصی مانند ویژگی‌ها و رفتارها را به منظور پیاده سازی کد تعریف می‌کنند.
- **روش:** روش یا متد، تابعی است که وظیفه یا عملی را انجام می‌دهد. به عنوان مثال، یک متد ممکن است اطلاعات مربوط به داده‌های یک شیء را برگرداند.

- **ویژگی:** ویژگی، قادر است ساختار اطلاعات یک شیء را ذخیره و وضعیت آن را تعریف کند. کاربر می‌تواند یک ویژگی را به عنوان بخشی از یک کلاس تعریف کند.

مفاهیم

مفاهیم برنامه نویسی شیء گرا به صورت زیر هستند:

1) کلاس

کلاس، یک نوع داده تعریف شده توسط کاربر است که از اعضای داده و توابع عضو تشکیل شده است. با ایجاد نمونه ای از کلاس می‌توان به این اعضا و توابع دسترسی پیدا کرد و از آنها استفاده کرد. اساساً یک کلاس به عنوان یک طرح اولیه برای اشیا عمل می‌کند و مجموعه ای از ویژگی‌ها و روش‌های مشترک، برای همه اشیایی که از آن نوع است را نشان می‌دهد. به عنوان مثال، کلاسی به نام Car را در نظر بگیرید. در حالی که بسیاری از خودروها دارای نام‌ها و مارک‌های متفاوتی هستند؛ اما همه آنها دارای ویژگی‌های مشترکی هستند، مانند داشتن چهار چرخ، محدودیت در سرعت و محدوده مسافتی که پیموده‌اند. در اینجا خودرو کلاس است و چرخ‌ها،

محدودیت سرعت و مسافت پیموده شده توسط آنها از جمله ویژگی‌های آن هستند.

2) شیء

شیء، یک واحد اساسی از برنامه نویسی شیء گرا است که موجودیت دنیای واقعی را نشان می‌دهد. یک شیء نمونه ای از یک کلاس است. در حالی که تعریف یک کلاس، حافظه ای را به خود تخصیص نمی‌دهد، نمونه سازی یک شیء این کار را خواهد کرد.یک شیء دارای هویت، حالت و رفتار است.

این امر شامل داده‌ها و کد به منظور ایجاد تغییراتی در آن داده‌ها است.

اشیاء می‌توانند بدون نیاز به دانستن جزئیات داده یا کد با یکدیگر تعامل داشته باشند. کافی است، نوع پیام‌های پذیرفته شده و نوع پاسخ‌های برگشتی آنها را بدانید.

3) اتصال پویا

اتصال پویا که به عنوان اتصال دیرهنگام نیز شناخته می‌شود، به فرآیندی اشاره دارد که در آن کدی که باید در پاسخ به فراخوانی تابع اجرا شود در زمان اجرا تعیین خواهد شد و به این معنا است: کد دقیقی که در ارتباط با فراخوانی رویه معینی است تا زمانی که برنامه اجرا نشود، مشخص نخواهد شد. یکی از مزایای مهم وراثت این است که یک کلاس مشتق شده از تمامی اعضای کلاس پایه خود را به ارث می‌برد. تا زمانی که کلاس مشتق شده، هیچ یک از اعضای عمومی کلاس پایه که در آن قرار گرفته را پنهان نکند، یک شیء از کلاس مشتق شده را می‌توان در هر جایی که شیء کلاس پایه در آن مورد انتظار است، استفاده کرد. این قابلیت به چند شکلی زیر گروهی هم معروف است.

4) ارسال پیام

ارسال پیام یک روش ارتباطی در برنامه نویسی شیء گرا و همچنین برنامه نویسی موازی است. در برنامه نویسی شیء گرا، اشیا با ارسال و دریافت اطلاعات به یکدیگر با هم ارتباط برقرار می‌کنند. پیام برای یک شیء، درخواستی برای اجرای یک رویه معین است که تابعی را در شیء دریافت کننده، فراخوانی می‌کند تا نتایج مورد نظر را ایجاد کند. ارسال پیام، شامل مشخص کردن نام شیء، نام تابع و اطلاعاتی است که باید ارسال شوند تا تعامل و هماهنگی بین اشیاء مختلف را تسهیل کنند.

اصول برنامه نویسی شیء گرا

در دنیای برنامه نویسی شیء گرا، همه چیز تقریباً یک شیء در نظر گرفته می‌شود. اشیا حاوی داده‌هایی هستند که به عنوان ویژگی‌ها و متدها نیز در نظر گرفته می‌شوند. این نوع برنامه نویسی به اشیا اجازه می‌دهد تا با استفاده از ٤ اصل با یکدیگر در تعامل باشند که عبارت اند از:

- کپسوله سازی
- وراثت

- چند شکلی
- انتزاع

این ٤ اصل، اشیا را قادر می‌سازند تا برای ایجاد برنامه ای قدرتمند با یکدیگر ارتباط برقرار کنند و همکاری کنند. در ادامه به بررسی هر یک از این اصول خواهیم پرداخت.

1) کپسوله سازی

کپسولاسیون یکی از اصول در برنامه نویسی شیء گرا است که به دنبال پنهان کردن جزئیات اجرای اشیا از دنیای خارج است. این اصل، بیان می‌کند که تمامی اطلاعات مهم در داخل شیء موجود است. فقط داده‌های انتخاب شده به صورت خارجی در دسترس قرار خواهد گرفت. عملکرد و وضعیت درونی هر شیء به صورت خصوصی در کلاس مشخص شده، ذخیره می‌شود، در حالی که اشیاء دیگر به آن دسترسی ندارند یا توانایی ایجاد تغییرات در آن را ندارند. در عوض، آنها فقط می‌توانند با چند تابع یا روش عمومی تعامل داشته باشند. این شکل از پنهان کردن داده‌ها، امنیت در برنامه و کنترل تغییرات در وضعیت شیء را امکان پذیر می‌کند، خطر خطاها را کاهش می‌دهد و برنامه را قابل درک می‌کند.

2) وراثت

وراثت یکی دیگر از اصول برنامه نویسی شیء گرا است که به توسعه دهندگان کمک می‌کند تا کلاس‌های جدیدی را بر اساس کلاس‌های موجود (والد) ایجاد کنند، با این شرط که ویژگی‌ها و روش‌های آنها را نادیده بگیرند یا افزایش دهند.

این امر به ویژه در برنامه‌هایی که حاوی هزاران خط کد است، مفید خواهند بود؛ زیرا تعمیر و نگهداری را ساده تر می‌کند و از تکرار کد جلوگیری می‌کند. با استفاده از منطق کلاس والد در کلاس فرزند، توسعه دهندگان می‌توانند اشیایی را ایجاد کنند که کد یا منطق را به اشتراک بگذارد و در عین حال متفاوت نیز باشد. این امر، پیچیدگی‌های کد را کاهش می‌دهد و نیاز به ایجاد یک شیء جدید برای هر شیء مورد استفاده در برنامه را از بین خواهد برد.

3) چند شکلی

چند شکلی، اصلی است که وراثت را با اجازه دادن به اشیا از کلاس‌های مختلف برای انجام اقداماتی با نام یکسان و با استفاده از کدهای مختلف تکمیل می‌کند.به عنوان مثال، روش نمایش اطلاعات می‌تواند برای نمایش داده‌های متنوع در مورد اشیایی همچون ماشین، هواپیما یا کشتی استفاده شود. علاوه بر این، پلی مورفیسم به ایجاد برنامه‌های انعطاف پذیرتر و مدولار کمک می‌کند. به صورت کلی، این اصل می‌تواند فرآیند توسعه را ساده کند؛ زیرا امکان ایجاد روش‌ها و توابع مشترک برای چندین نوع اشیا را فراهم می‌کند.

4) انتزاع

انتزاع یکی دیگر از اصول است که به کاربر کمک می‌کند تا روی عناصر ضروری یک سیستم تمرکز کند و جزئیاتی که اهمیت کمتری دارد و هیچ تاثیری بر ویژگی‌های کلیدی آن سیستم ندارند را نادیده بگیرید. این امر، به کاربر کمک خواهد کرد تا برنامه‌های قابل فهم تری بسازد. انتزاع را می‌توان به عنوان گسترش کپسولاسیون نیز در نظر گرفت. به عنوان مثال،

برنامه‌هایی را در نظر بگیرید که حاوی هزاران خط کد هستند. از طریق اصل انتزاع، هر شیء فقط مکانیسم خاصی را برای استفاده نشان می‌دهد. بنابراین، کد داخل تا حد زیادی مستقل از اشیاء دیگر می‌شود. برای مثال، در برنامه‌ای که اطلاعات مربوط به فیلم‌ها را ذخیره می‌کند، می‌توان یک کلاس فیلم ایجاد کرد که دسترسی به جزییات اساسی مانند عنوان، سال انتشار و ژانر را فراهم می‌کند، در حالی که اطلاعات کم اهمیت‌تر مانند عکس‌ها را پنهان می‌کند.

برترین زبان‌های برنامه نویسی شیء گرا

زبان های برنامه نویسی زیادی وجود دارند که از برنامه نویسی شیء گرا، به صورت کامل یا جزئی پشتیبانی می‌کنند. از محبوب ترین و پرکاربردترین زبان‌های برنامه نویسی شیء گرا می‌توان به موارد زیر اشاره کرد:

- جاوا (JAVA)
- پایتون (Python)
- سی پلاس پلاس (C++)
- روبی (Ruby)

- سی شارپ (C#)
- جاوا اسکریپت (Java Script)

جاوا یک زبان همه منظوره، سطح بالا و کامپایل شده است و به گونه ای طراحی شده است که قابل حمل، قوی و ایمن باشد و از اصل یک بار بنویس، هر جا اجرا کن، پیروی می‌کند. پایتون یک زبان همه منظوره، سطح بالا و تفسیر شده است که به دلیل خوانایی، سادگی و تطبیق پذیری بالایی که دارد، شناخته شده است. C++ یک زبان کامپایل شده و همه منظوره، سطح پایین و توسعه یافته از زبان برنامه نویسی C با ویژگی‌های اضافه شده برای برنامه نویسی شیء گرا است. روبی یک زبان همه منظوره، سطح بالا و تفسیر شده است که تحت تأثیر Perl ، Smalltalkو Lisp قرار گرفته است.

نکات مهم در استفاده از برنامه نویسی شیء گرا

به منظور توسعه موفقیت آمیز برنامه‌ها با استفاده از این نوع برنامه نویسی، رعایت توصیه‌های زیر بسیار مهم خواهد بود:

- هنگام طراحی کد شیء گرا، سادگی و کارایی را به عنوان هدف خود در نظر بگیرید.

- اصول SOLID را رعایت کنید تا از بروز هرگونه مشکلی جلوگیری کنید، کد را انعطاف پذیر و قابل نگهداری کنید و بتوانید به راحتی آن را تغییر دهید.
- کلاس‌های مورد نیاز و روابط بین آنها را به منظور کسب اطمینان از عملکرد مؤثر برنامه شناسایی کنید.
- سعی کنید از تقلید خودداری کنید و این درک را به دست آورید که چه زمانی آن را اعمال کنید و چه زمانی به جای آن از ترکیب شیء استفاده کنید.
- از انتزاع به منظور ساده سازی در درک برنامه نویسی‌ها استفاده کنید.
- از استانداردها و قراردادهای برنامه نویسی پیروی کنید تا از وضوح و قابلیت نگهداری کد مطمئن شوید.

مزایا و معایب برنامه نویسی شیء گرا

مزایا و معایب استفاده از برنامه نویسی شیء گرا را در ادامه بررسی خواهیم کرد.

مزایای استفاده از برنامه نویسی شیء گرا

به دلیل مزایای متعددی که این نوع برنامه نویسی دارد، علی رغم ظهور مدل‌های مختلف برنامه نویسی، هنوز هم به عنوان سنگ بنای توسعه نرم افزار باقی مانده است. این مزایا را در ادامه بررسی خواهیم کرد.

1) قابلیت استفاده مجدد

یکی از مزایای کلیدی این نوع برنامه نویسی امکان استفاده مجدد کد از طریق وراثت است. وراثت به یک کلاس اجازه می‌دهد تا ویژگی‌ها و متدها را از کلاس دیگر به ارث ببرد و نیاز به کدهای اضافی را از بین ببرد.

این امر، از بروز مشکلات مربوط به تکرار کد جلوگیری می‌کند و از ثبات در سراسر برنامه اطمینان حاصل می‌کند. با تعریف کلاس‌ها، توسعه دهندگان می‌توانند هر چند بار که نیاز است از بخش‌های کد در یک برنامه به صورت مجدد استفاده کنند. کلاس‌های فرزند خصوصیات و رفتارهای کلاس‌های والد را به ارث می‌برند و اصلاح روش‌ها و مقادیر به ارث رسیده را آسان می‌کنند.

2) افزایش بهره وری در توسعه نرم افزار

برنامه نویسی شیء گرا به شکل قابل توجهی بهره وری را افزایش می‌دهد و به توسعه دهندگان این امکان را می‌دهد که برنامه‌هایی را با استفاده از ماژول‌های از پیش نوشته شده و به هم پیوسته ایجاد کنند، بدون این که نیازی به کار از ابتدا داشته باشند. این رویکرد ماژولار اجازه می‌دهد تا نرم افزار به مشکلات قابل مدیریت و گسسته تقسیم شود.

توانایی تقسیم کار و تمرکز بر ماژول‌های مبتنی بر شیء، کارایی را افزایش خواهد داد. علاوه بر این، برنامه نویسی شیء گرا، قابل گسترش است و اجازه می‌دهد تا ویژگی‌ها و رفتارهای جدیدی به اشیا اضافه شود. این مدولار بودن، توسعه پذیری و قابلیت استفاده مجدد از آن، به افزایش بهره وری در مقایسه با تکنیک‌های برنامه نویسی رویه ای سنتی ختم خواهد شد.

3) ساده شدن عیب یابی

این نوع برنامه نویسی با ارائه مکان‌هایی واضح در کد به منظور شناسایی مشکلات، عیب یابی را ساده می‌کند. هنگامی که یک خطا رخ دهد، معمولاً به شیء یا روش خاصی که سبب ایجاد مشکل می‌شود، اشاره خواهد کرد و نیاز به بازرسی از مناطق نامرتبط کد را کاهش خواهد داد. کپسوله سازی، یک اصل اساسی در برنامه نویسی شیء گرا است و تضمین می‌کند که اشیاء خودکفا هستند، که این موضوع به جداسازی مسائل کمک می‌کند. این خودکنترلی به ویژه برای مهندسان و توسعه دهندگان DevOps مفید است؛ زیرا به آنها این امکان را می‌دهد تا روی چندین پروژه به صورت همزمان کار کنند، در عین حال از تکرار کدها جلوگیری کنند و از مدیریت خطاهای قوی اطمینان لازم را به دست آورند.

4) افزایش امنیت

به منظور حفظ امنیت در برنامه و ارائه داده‌های حیاتی برای مشاهده، برنامه نویسی شیء گرا، داده‌های محدود را از طریق مکانیسم‌های مخفی کردن و انتزاع داده‌ها، فیلتر می‌کند. مفهوم انتزاع داده در برنامه نویسی

شیء گرا این امکان را به کاربر می‌دهد تا تنها مقدار کمی از داده‌ها را به کاربر نمایش دهد که یکی از نقاط قوت برنامه نویسی شیء گرا است.
هنگامی که تنها اطلاعات لازم در دسترس باشد، سایر اطلاعات قابل دسترسی نخواهند بود. در نتیجه، حفظ امنیت امکان پذیر می‌شود. سایر مزایای برنامه نویسی شیء گرا در ایده انتزاعی به منظور پنهان کردن پیچیدگی از سایر کاربران و نمایش اطلاعات عنصر بر اساس نیازها استفاده می‌شود.

مزایای یادگیری برنامه نویسی شیء گرا

یادگیری برنامه نویسی شیء گرا، می‌تواند برای برنامه نویسانی که می‌خواهند مهارت‌ها و دانش خود را در توسعه سیستم ارتقا دهند، بسیار مفید باشد. این برنامه نویسی، می‌تواند به آنها کمک کند تا همانند اشیا فکر کنند که اغلب با موجودیت‌ها و مفاهیم دنیای واقعی که آنها سعی در مدل سازی و حل آنها دارند، مرتبط است. علاوه بر این، برنامه نویسی شیء گرا می‌تواند آنها را قادر به نوشتن کدهای قابل استفاده مجدد و ماژولار بیشتری کند که می‌تواند زمان و هزینه توسعه را کاهش دهد و در عین حال کیفیت و قابلیت نگهداری را افزایش دهد. به علاوه، درک زبان‌ها و چارچوب‌های برنامه نویسی شیء گرا می‌تواند قابلیت استخدام و تطبیق پذیری برنامه نویس را در بازار افزایش دهد. در نهایت، یادگیری برنامه نویسی شیء گرا می‌تواند برنامه نویسان را قادر به اعمال الگوها و اصول طراحی کند که می‌تواند مهارت‌های حل مسئله و طراحی را در آنها افزایش دهد و در عین حال کدنویسی آنها را قوی تر و مقیاس پذیرتر کند.

معایب استفاده از برنامه نویسی شیء گرا

یکی از این عیوب، احتمال افزایش پیچیدگی و اندازه کد است، به خصوص زمانی که سیستم، شامل سطوح مختلف وراثت، پلی مورفیسم و اتصال پویا باشد. این ویژگی‌ها می‌توانند درک، اشکال زدایی و آزمایش کد را سخت تر کنند و خطاها و اشکالاتی را معرفی کنند که شناسایی و رفع آنها دشوار است. یکی دیگر از اشکالات این نوع برنامه نویسی این است که می‌تواند حافظه و منابع CPU بیشتری را نسبت به پارادایم‌های دیگر مانند برنامه نویسی رویه‌ای یا عملکردی مصرف کند و به این دلیل است که اشیا هم داده‌ها و هم روش‌ها را ذخیره می‌کنند و برای ایجاد و دستکاری به فضا و زمان بیشتری نیاز دارند.