دانشگاه پیام نور

تحلیل و طراحی شیءگرا

رضا مغانی

امروزه کتاب‌خوانی و علم‌آموزی، نه تنها یک وظیفه‌ی ملی، که یک واجب دینی است.

مقام معظم رهبری

در عصر حاضر یکی از شاخصه‌های ارزیابی رشد، توسعه و پیشرفت فرهنگی هر کشوری میزان تولید کتاب، مطالعه و کتاب‌خوانی مردم آن مرز و بوم است. ایران اسلامی نیز از دیرباز تاکنون با داشتن تمدنی چندهزارساله و مراکز متعدد علمی، فرهنگی، کتابخانه‌های معتبر، علما و دانشمندان بزرگ با آثار ارزشمند تاریخی، سرآمد دولت‌ها و ملت‌های دیگر بوده و در عرصه‌ی فرهنگ و تمدن جهانی به‌سان خورشیدی تابناک همچنان می‌درخشد و با فرزندان نیک‌نهاد خویش هنرنمایی می‌کند. چه کسی است که در دنیا با دانشمندان فرزانه و نام‌آور ایرانی همچون ابوعلی سینا، ابوریحان بیرونی، فارابی، خوارزمی و ... همچنین شاعران برجسته‌ای نظیر فردوسی، سعدی، مولوی، حافظ و ... آشنا نباشد و در مقابل عظمت آنها سر تعظیم فرود نیاورد. تمامی این افتخارات ارزشمند، برگرفته از میزان عشق و علاقه فراوان ملت ما به فراگیری علم و دانش از طریق خواندن و مطالعه منابع و کتاب‌های گوناگون است. به شکرانه‌ی الهی، تاریخ و گذشته ما، همیشه درخشان و پربار است. ولی اکنون در این زمینه در چه جایگاهی قرار داریم؟ آمار و ارقام ارائه‌شده از سوی مجامع و سازمان‌های فرهنگی در مورد سرانه‌ی مطالعه‌ی هر ایرانی، برایمان چندان امیدوارکننده نمی‌باشد و رهبر معظم انقلاب اسلامی نیز از این وضعیت بارها اظهار گله و ناخشنودی نموده‌اند.

کتاب، دروازه‌ای به سوی گستره‌ی دانش و معرفت است و کتاب خوب، یکی از بهترین ابزارهای کمال بشری است. همه‌ی دستاوردهای بشر در سراسر عمر جهان، تا آنجا که قابل کتابت بوده است، در میان دست‌نوشته‌هایی است که انسان‌ها پدید آورده و می‌آورند. در این مجموعه‌ی بی‌نظیر، تعالیم الهی، درس‌های پیامبران به بشر، و همچنین علوم مختلفی است که سعادت بشر بدون آگاهی از آنها امکان‌پذیر نیست. کسی که با دنیای زیبا و زندگی‌بخش کتاب ارتباط ندارد بی‌شک از مهم‌ترین دستاورد انسانی و نیز از بیشترین معارف الهی و بشری محروم است. با این دیدگاه، به‌روشنی می‌توان ارزش و مفهوم رمزی عمیق در این حقیقت تاریخی را دریافت که اولین خطاب خداوند متعال به پیامبر گرامی اسلام(ص) این است که «بخوان!» و در اولین

سوره‌ای که بر آن فرستاده‌ی عظیم‌الشأن خداوند، فرود آمده، نام «قلم» به تجلیل یاد شده‌است: «إِقْرَأْ وَ رَبُّکَ الْاَکْرَمُ. اَلَّذی عَلَّمَ بِالْقَلَم» در اهمیت عنصر کتاب برای تکامل جامعه‌ی انسانی، همین بس که تمامی ادیان آسمانی و رجال بزرگ تاریخ بشری، از طریق کتاب جاودانه مانده‌اند.

دانشگاه پیام‌نور با گستره‌ی جغرافیایی ایران‌شمول خود با هدف آموزش برای همه، همه‌جا و همه‌وقت، به‌عنوان دانشگاهی کتاب‌محور در نظام آموزش عالی کشورمان، افتخار دارد جایگاه اندیشه‌سازی و خردورزی بخش عظیمی از جوانان جویای علم این مرز و بوم باشد. تلاش فراوانی در ایام طولانی فعالیت این دانشگاه انجام پذیرفته تا با بهره‌گیری از تجربه‌های گرانقدر استادان و صاحب‌نظران برجسته کشورمان، کتاب‌ها و منابع آموزشی درسی شاخص و خودآموز تولید شود. در آینده هم، این مهم با هدف ارتقای سطح علمی، روزآمدی و توجه بیشتر به نیازهای مخاطبان دانشگاه پیام‌نور با جدیت ادامه خواهد داشت. به‌طور قطع استفاده از نظرات استادان، صاحب‌نظران و دانشجویان محترم، ما را در انجام این وظیفه‌ی مهم و خطیر یاری‌رسان خواهد بود. پیشاپیش از تمامی عزیزانی که با نقد، تصحیح و پیشنهادهای خود ما را در انجام این وظیفه‌ی خطیر یاری می‌رسانند، سپاسگزاری می‌نماییم. لازم است از تمامی اندیشمندانی که تاکنون دانشگاه پیام‌نور را منزلگه اندیشه‌سازی خود دانسته و ما را در تولید کتاب و محتوای آموزشی درسی یاری نموده‌اند، صمیمانه قدردانی گردد. موفقیت و بهروزی تمامی دانشجویان و دانش‌پژوهان عزیز آرزوی همیشگی ما است.

دانشگاه پیام‌نور

فهرست مطالب

پیشگفتار

مجموعه پیش رو که براساس سرفصل مصوب وزارت علوم برای درس تحلیل و طراحی شیءگرا تدوین شده‌است، یک مجموعه کامل در مورد آموزش مفاهیم شیءگرا و مدل‌سازی و مستندسازی سیستم‌های نرم‌افزار با رویکرد شیءگراست. در این مجموعه سعی‌شده مطالب آموزشی، مبتنی بر نیازهای کاربردی دانشجویان رشته‌های مهندسی کامپیوتر و مهندسی فناوری اطلاعات در زمینه شناخت، طراحی، مدل‌سازی و مستندسازی سیستم‌های نرم‌افزار با رویکرد شیءگرا گردآوری شود. همچنین مثال‌های موجود در این مجموعه ساده و دارای جنبه کاربردی بوده و در راستای درک بهتر مفاهیم مطرح شده‌است. بررسی مفاهیم کلاسیک و ارائه تعریف‌های دقیق، معرفی ابزارها، شرح اصطلاحات و همچنین معرفی متدولوژی‌های تولید و توسعه نرم‌افزار با رویکرد شیءگرا، به‌صورت یکجا و با دید کاربردی از ویژگی‌های بارز این مجموعه می‌باشد.

محتوی این کتاب شامل ۷ فصل است که در ۳ بخش دسته‌بندی شده‌است. در بخش اول مفاهیم اساسی شیءگرا، سیستم‌های نرم‌افزاری شیءگرا و زبان‌های برنامه‌نویسی شیءگرا معرفی شده‌اند که درک آن برای شروع یادگیری تکنیک‌های تحلیل و طراحی با رویکرد شیءگرا بسیار ضروری است. همچنین در این بخش تفاوت رویکرد شیءگرا با روش‌های سنتی و نیز مزایای تحلیل و طراحی به روش شیءگرا بیان شده‌است.

بخش دوم به تشریح فرایندها و متدولوژی‌های تولید و توسعه سیستم‌های نرم‌افزاری شامل متدولوژی‌های سنتی و متدولوژی‌های شیءگرا می‌پردازد. همچنین مزایای تحلیل و طراحی با رویکرد شیءگرا و آشنایی با متدولوژی RUP و

متدولوژی‌های جدیدتر تحلیل و طراحی شیءگرا ازجمله AUP، SCRUMو XP در این بخش برای خواننده ارائه شده‌است.

در بخش سوم خواننده به‌صورت کاربردی با روش‌های مدل‌سازی و مستندسازی سیستم‌های نرم‌افزارهای شیءگرا آشنا خواهد شد. در این بخش زبان مدل‌سازی یکپارچه UML به‌عنوان رایج‌ترین ابزار مستندسازی و مدل‌سازی سیستم‌های نرم‌افزاری شیءگرا معرفی شده‌است و خواننده در قالب یک مثال کاربردی با نمودارهای زبان UML آشنا می‌شود. اهمیت مطالعه این فصل برای خواننده در درک مبتنی بر واقعیت‌بودن رویکرد شیءگراست. چرا که UML می‌تواند آنچه را که حاصل تحلیل و طراحی شیءگرا بوده به‌صورت نمودارها و ماژول‌های بصری در قالب انتزاع‌های ملموس از دنیای واقعی به خواننده ارائه نماید.

در پایان هر فصل، خلاصه مطالب به همراه خودآزمایی چندگزینه‌ای و خودآزمایی تشریحی اضافه شده‌است تا دانشجویان ضمن مرور مطالب اصلی هر فصل کیفیت یادگیری خود را به همان شیوه که در آزمون‌های پایان ترم دانشگاه پیام نور انجام می‌شود، ارزشیابی نمایند.

فهرست واژگان و همچنین ترجمه انگلیسی و فارسی آن‌ها نیز به‌عنوان ضمیمه در انتهای کتاب آمده است.

امیدوارم مطالب ارائه‌شده در این مجموعه برای دانشجویان دانشگاه پیام نور مناسب و مفید واقع گردد.

رضا مغانی

فصل اول

فناوری شیءگرایی

مقدمهٔ فصل اول

مطالب این فصل برای درک بهتر آنچه که در بخش‌های ۲ و ۳ مطرح می‌شود، ضروری است. قبل از معرفی روش‌ها و مدل‌های تحلیل و طراحی شیءگرا، لازم است خوانندگان محترم ابتدا با تعاریف موجود در فناوری شیءگرا مثل وراثت، چندریختی و مخفی‌سازی و نیز مفاهیم اساسی آن از جمله کلاس‌ها، اشیاء، صفات و رفتارها و روابط بین آن‌ها آشنا شوند.

شیءگرایی و مفاهیم آن

شیءگرایی (Object Oriented) اصطلاحی است که امروزه در صنعت نرم‌افزار بسیار رایج شده‌است و به‌عنوان یک رویکرد نوین در بین تولیدکنندگان سیستم‌های نرم‌افزاری مورد توجه قرار گرفته است. شرکت‌ها به سرعت تلاش می‌کنند تا خود را با این رویکرد جدید سازگار کنند و آن را در نرم‌افزارهای پیش روی خود و حتی در برنامه‌های موجود وارد نمایند. در حقیقت بیشتر برنامه‌ها امروزه با رویکرد شیءگرا توسعه می‌یابند.

رویکرد شیءگرا در توسعه نرم‌افزار اولین بار در اوخر دهه ۱۹۶۰ برای توسعه نرم‌افزار به کار گرفته شد و حدوداً ۲۰ سال طول کشید تا این رویکرد به‌طور گسترده مورد استفاده قرار گیرد. در طول دهه ۱۹۹۰ مهندسی نرم‌افزار شیءگرا الگوی منتخب بسیاری از تولیدکنندگان نرم‌افزار شد و تعداد فزاینده‌ای از مهندسان حرفه‌ای و طراحان سیستم‌های نرم‌افزار به آن روی آوردند. از آن زمان به بعد هر چه بیشتر می‌گذرد رویکرد شیءگرا بیشتر مورد استقبال قرار گرفته و با کمی جسارت می‌توان گفت این رویکرد در حال حاضر کاملاً جایگزین روش‌های کلاسیک توسعه نرم‌افزار شده‌است.

چرایی رویکرد شیءگرا

برای نگرش به مسأله‌هایی که قرار است توسط نرم‌افزارها حل شوند، راه‌های زیادی وجود دارد. یکی از پرکاربردترین این راه‌ها رویکرد شیءگرا است. به بیان دیگر رویکرد شیءگرا (O.O) یک راه متفاوت مشاهده برنامه‌ها است که شما با آن، یک برنامه را به قطعات بسیار کوچک یا اشیایی تقسیم می‌کنید، که تا اندازه‌ای مستقل از یکدیگر باشند. این قطعات به منزله بلوک‌های سازنده یک سازه کلی‌تر هستند. این سازه، برنامه یا سیستم نرم‌افزار مورد نظر شما است. یکی از امتیازات اساسی رویکرد شیءگرا این است که می‌توانید هر قطعه[1] را فقط یک بار بسازید و بارها و بارها از آن استفاده کنید. به عبارتی رویکرد شیءگرا منجر به استفادهٔ مجدد می‌شود و استفادهٔ مجدد از مؤلفه‌های برنامه منجر به توسعه سریع‌ترنرم‌افزارها و تولید برنامه‌هایی با کیفیت بالاتر می‌شود. همچنین نگهدرای نرم‌افزارهای شیءگرا آسان‌تر است زیرا ساختار آن‌ها ذاتاً فاقد پیوستگی است. این موضوع، به هنگام اعمال تغییرات، اثرات جانبی کمتری به وجود می‌آورد و برای مهندس نرم‌افزار و مشتری دردسر کمتری ایجاد می‌کند. به علاوه، تطبیق‌دادن و تغییردادن اندازهٔ سیستم‌های شیءگرا آسان‌تر است.

به لحاظ تکنیکی در رویکرد شیءگرا دامنه مسأله به‌عنوان مجموعه‌ای از اشیاء مشخص می‌شود که دارای صفات و رفتارهای (عملیات‌های) معین و معنادار است. این اشیاء با مجموعه‌ای از توابع موسوم به متد، عملیات یا سرویس، دستکاری می‌شوند و از طریق یک قرارداد پیام‌رسانی با هم ارتباط برقرار می‌کنند. برای طبقه‌بندی این اشیاء از مفهومی به نام کلاس استفاده می‌شود.

در رویکرد شیءگرا شما سیستم مورد نظر خود را منطبق بر همان چیزهایی که در دنیای واقعی وجود دارند، می‌سازید.

در واقع رمز محبوبیت رویکرد شیءگرا در بین تولیدکنندگان، برنامه‌نویسان، مدیران و کاربران سیستم‌های نرم‌افزاری، همین نکته است. چرا که مدل‌سازی سیستم با اشیائی که نمونه‌های آن در دنیای واقعی وجود دارد، از پیچیدگی آن می‌کاهد و سیستم را برای تمامی ذی‌نفعان آن قابل فهم می‌سازد.

---

1. Component

مفاهیم اساسی رویکرد شیءگرا

در بخش قبل اشاره شد که سیستم‌های شیءگرا در مواجهه با درخواست‌های تغییر، انعطاف‌پذیرند. برای درک انعطاف‌پذیری سیستم‌های شیءگرا، نیازمند شناخت قبلی نسبت به تعدادی از اصول اولیه رویکرد شیءگرا از جمله مخفی‌سازی[1]، وراثت[2] و چندریختی[3] خواهیم بود که این مفاهیم در ارتباط با دو مفهوم اصلی‌تر یعنی شیء[4] و کلاس[5] معنا پیدا می‌کنند. هر چند این موضوع در عمل و در زمان طراحی یک سیستم شیءگرا راحت‌تر قابل درک است.

شیء (Object)

اشیاء در اطراف ما قرار دارند. صندلی که روی آن می‌نشینید، کتابی که آن را می‌خوانید، خودکارتان، کارت شناسایی و یا هر مورد دیگری که دارای هویت منفرد و مستقل باشد و بتوان برای آن برخی ویژگی‌ها (صفات) و برخی رفتار (عملیات) را متصور شد.

در سیستم‌های نرم‌افزار، یک شیء معنای نزدیکی با این مفاهیم دارد. شیء، مؤلفه‌ای است که برخی اطلاعات و برخی رفتارها را در خود نگهداری (کپسوله) می‌کند. شیء، مفهومی است که برخی از چیزهای عینی در دنیای واقع را می‌توان با آن نمایش داد. به عبارت دیگر یک شیء، یک مصداق از یک تعریف کلی است که عینیت یافته و در دنیای واقع وجود دارد و می‌توان آن را آدرس‌دهی کرد. یعنی مکان استقرار آن در طبیعت قابل شناسایی است و اگر شیء مورد نظر متعلق به یک سیستم نرم‌افزاری باشد، مکان استقرار آن در حافظه‌های کامپیوتری قابل شناسایی خواهد بود. چند مثال از اشیاء مختلف در محیط واقعی به‌صورت زیر می‌باشند:

- حساب بانکی دانشگاه پیام نور
- خانه ای در خیابان حافظ
- یکی از کتب دانشمند شهیر ایرانی، خوارزمی
- دانشجوی رتبه اول کلاس شیءگرا
- گل قرمزی که روبه‌روی پنجره اتاق من است

---

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Object
5. Class

به‌عنوان مثال در یک سیستم نرم‌افزار مثل نرم‌افزار دستگاه خودپرداز بانک (ATM)، صفحه نمایش عابربانک[1]، ماژول کارت خوان[2]، کارت عابربانک شما[3]، حساب بانکی‌تان و رسیدی که پس از برداشت مبلغ ۵۰۰۰۰ تومان از حسابتان دریافت می‌کنید، همه نمونه‌هایی از اشیاء مرتبط با این سیستم هستند. هر شیء درون خودش مقداری از اطلاعات و رفتار را قرار داده است. به‌طور مثال حساب بانکی شما برخی اطلاعات را به این صورت با خود دارد: **شماره حساب، موجودی حساب، تاریخ آخرین تراکنش، نوع حساب (جاری-پس انداز)، تاریخ افتتاح و غیره**. همچنین این حساب دارای یک سری رفتار و عملکردهایی است که در ارتباط با اطلاعات خود یا در ارتباط با سایر اشیاء موجود در این سیستم از خود بروز می‌دهد. مثلاً حساب شما می‌داند که چگونه وقتی مبلغی را به حساب خود منتقل می‌کنید، موجودی‌اش را افزایش دهد، هنگامی‌که پولی برداشت می‌کنید مقدار موجودی خودش را کاهش دهد، هنگامی‌که در صفحه نمایش عابربانک، گزینه موجودی حساب را انتخاب می‌کنید، موجودی خود را در اختیار صفحه نمایش قرار داده تا به شما نشان داده شود و هنگامی‌که بخواهید بیشتر از موجودی آن مبلغی را برداشت نمایید می‌داند که چگونه رفتار کند. صفحه نمایش نیز می‌داند در صورت انتخاب هر یک از منوهای موجود چه رفتاری (عملیاتی) انجام دهد.بنابراین، هر شیء دارای دو بخش اصلی است: اطلاعات[4] و رفتار[5].

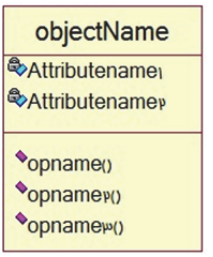
در توصیف‌های مختلف توسط نویسندگان از نام‌گذاری‌های دیگری نیز برای این دو بخش استفاده شده‌است، مثلاً برای اطلاعات از واژه صفات و ویژگی‌ها نیز استفاده شده‌است که همگی معادل واژه انگلیسیAttributes می‌باشند. برای رفتار اشیاء نیز استفاده از واژه‌های عملیات و متد نیز رایج است که این‌ها معادل واژه Methods و گاهی Operations می‌باشند. نمایش تصویری یک شیء به همراه صفات و عملیات آن در شکل ۱-۱ ارائه شده‌است. این نمادگذاری مطابق استاندارد UML[6] می‌باشد.

---

1. ATM Screen
2. Card Reader
3. ATM Card
4. Information
5. Behavior

۶. Unified Modeling Language یک زبان استاندارد برای مدل‌سازی سیستم‌های نرم‌افزاری است که در آن از نمادگذاری‌های بصری برای توصیف اشیاء و کلاس‌ها و روابط بین آن‌ها استفاده می‌شود. زبان UML در فصل چهارم همین کتاب معرفی شده‌است.

بخش‌های اطلاعاتی یا داده‌هایی که توسط اشیاء نگهداری می‌شوند، صفات آن می‌باشند. با وجود اینکه مقادیر صفات برای اشیاء تغییر خواهند کرد، خود صفات هرگز تغییر نمی‌کنند (ممکن است موجودی حساب شما هفته آینده افزایش یا کاهش یابد اما حساب شما همیشه دارای یک شماره حساب و یک فیلد برای نگهداری موجودی حساب، خواهد بود).



شکل ۱-۱. نمایش بصری مدل شیء در زبان UML

رفتارهای یک شیء به‌عنوان عملیات آن شناخته می‌شوند. چند نمونه عملیات مربوط به حساب بانکی شما عبارتند از: بازشدن حساب، بسته‌شدن حساب، نمایش موجودی، کسر از موجودی، افزایش موجودی، مسدود نمودن موجودی، نمایش نام صاحب حساب، نمایش شماره حساب و یا تغییر آدرس صاحب حساب.

در ساده‌ترین حالت، یک شیء، از روی یک کلاس ایجاد می‌شود. چون کلاس نمی‌تواند عملیاتی باشد، ولی نمونه‌های تولیدشده از آن قابلیت اجرایی خواهند داشت. در کامل‌ترین حالت یک شیء، موجودیتی است کاملاً مستقل با مسوولیت‌های شخصی خویش که در لحظهٔ تولید از کلاس مرجع خود بوجودآمده یا متولدشده و در پایان پس از انجام مسئولیت‌های خود می‌باید از بین برود.

کلاس (Class)

طرح کلی برای یک شیء را کلاس آن فراهم می‌کند. به عبارت دیگر، یک کلاس یک تعریف برای قالبی است که یک شیء می‌تواند داشته باشد و تعیین‌کننده اطلاعاتی است که یک شیء می‌تواند نگهداری کند و نشان‌دهنده رفتارهایی است که می‌تواند داشته باشد. در اصطلاح فنی هر شیء نهایتاً به یک کلاس نگاشت[1] می‌یابد. به عبارت دیگر

1. Map

شیء عینیت یافته کلاس است. یک نمونه واقعی و عینی از یک کلاس که قابل نگهداری و قابل آدرس‌دهی است. کلاس‌های مربوط به اشیاء ذکرشده در بخش قبل به ترتیب عبارتند از: ***حساب، خانه،کتاب، دانشجو و گل***

کلاس مربوط به یک کتاب نشان خواهد داد که هر کتاب چه تعداد صفحه و چه عنوانی دارد، نویسنده آن کیست و در چه سالی منتشر شده‌است. به‌طور مثال کتاب «الجبر و المقابله[۱]» یک نمونه[۲] از کلاس کتاب است که نویسنده آن خوارزمی[۳] بوده و در اوایل قرن سوم هجری قمری تالیف شده‌است. این نمونه موجود، یک شیء از کلاس کتاب است. می‌توان از یک کلاس تعداد نامحدودی نمونه (شیء) تولید کرد.

در زمان پیاده‌سازی سیستم‌های نرم‌افزار که با رویکرد شیءگرا طراحی شده‌اند هرکلاس به یک جدول در پایگاه داده نگاشت می‌شودو اشیاء تولیدشده از آن (نمونه‌های کلاس) به‌عنوان رکوردهای جدول مذکور نگهداری خواهد شد. البته این مطلب فقط در مورد کلاس‌های مربوط به موجودیت‌های ماندگار[۴] سیستم صادق است و نمونه‌های کلاس‌های مرزی[۵] و کلاس‌های کنترلی[۶] در پایگاه داده نگهداری نمی‌شوند بلکه به‌صورت موقت و در هر بار اجرای نرم‌افزار در حافظه اصلی[۷] ایجادشده و پس از بستن برنامه نابود می‌شوند.

مخفی‌سازی (Encapsulation)

در سیستم‌های شیءگرا مشخصات هر شیء یعنی اطلاعات و رفتار آن، در داخل شیء و به‌عنوان بخشی از تعریف شیء نگهداری می‌شود و به این ترتیب اطلاعات مرتبط به

---

۱. کتاب الجبر و المقابله، کهن‌ترین کتاب نوشته‌شده در جبر و نیز کهن‌ترین کتاب ریاضی است که از مسلمین به دست ما رسیده است. این کتاب در روزگار خلافت مأمون، یعنی حدود ۱۲۱۰ سال قمری و ۱۱۷۰ سال شمسی پیش از این تألیف شده‌است. این کتاب از سده ۱۲ میلادی، که اروپائیان با علم جبر آشنا شدند، تا سده ۱۶ میلادی (یعنی حدود ۴ قرن)، مبنای مطالعات علمی آنان در جبر بود. در اهمیت این کتاب همین بس، که امروزه این رشته مهم ریاضیات به نام همین کتاب یعنی جبر (در زبان‌های اروپایی به صورت‌هایی مانندAlgebra و مانند آن) خوانده می‌شود. این کتاب حل آنالیزی معادلات درجه اول و دوم را در بر دارد.

2. Instance

۳. ابوجعفر محمد بن موسی خوارزمی از دانشمندان بزرگ ریاضی و ستاره‌شناس ایرانی است که در سده دوم و سوم هجری می‌زیست.

4. Entity Classes

5. Boundary Classes

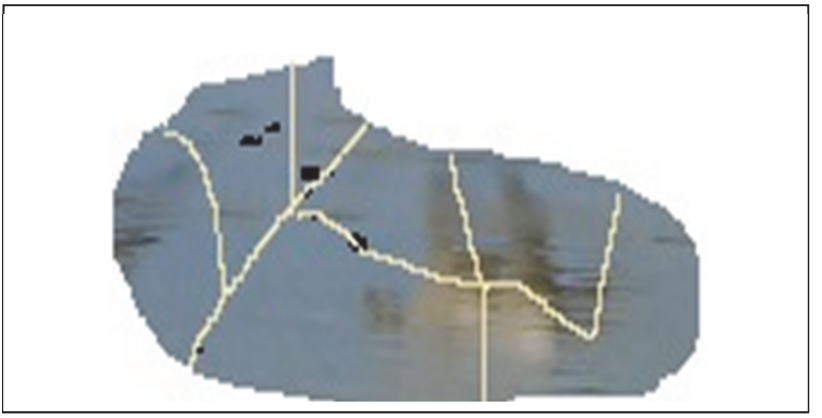
6. Control Classes

۷. منظور حافظه زمان اجراست که ممکن است RAMیا حافضه مجازی باشد.

هم در یک مکان و تحت یک نام دسته‌بندی می‌شوند. که به آن مخفی‌سازی می‌گویند. حاصل این کار طبقه‌بندی یا کلاس‌بندی مؤلفه‌های یک سیستم نرم‌افزاری است. مثلاً صفات یک حساب بانکی شامل: ***شماره حساب، موجودی، نام مشتری، آدرس مشتری، نوع حساب، نرخ بهره و تاریخ بازشدن حساب*** و نیز عملیات آن شامل: ***بازکردن حساب، بستن حساب، برداشت از حساب، تغییر نوع حساب و تغییر نام یا آدرس مشتری*** را می‌توان به‌صورت یکجا در یک شیء به نام حساب (Account) پنهان کرد. در نتیجه همه تغییرات مربوط به کلیه حساب‌ها می‌توانند به آسانی با اعمال آن تغییر، در کلاسِ حساب انجام شوند. (در نظر بگیرید هیأت مدیره تصمیم بگیرد نرخ بهره را برای کلیه حساب‌ها از ۱۵٪ به ۱۸٪ تغییر دهد. این خواسته فقط با تغییر فیلد مربوطه در کلاس Account انجام خواهد شد و به صورت خودکار، تغییر مذکور به کلیه زیرکلاس‌ها و اشیاء مربوط به این کلاس اعمال می‌شود.)

مخفی‌سازی به نوعی یک بسته‌بندی یا مرزبندی جدید برای مؤلفه‌های سیستم‌های نرم‌افزار فراهم می‌کند. حاصل این عمل محدودکردن تأثیرپذیری ناشی از اعمال تغییرات به سیستم است. به عبارت دیگر این امکان فراهم می‌شود که تغییرات را به‌صورت کنترل‌شده به سیستم اعمال کنیم و تغییرات فقط به بخش مورد نظر اعمال شوند و سایر بخش‌های سیستم تأثیر نپذیرند.

یک سیستم نرم‌افزار را به‌عنوان یک استخر بزرگ آب و درخواست‌های تغییر را به‌عنوان تکه سنگ‌هایی که به درون این استخر پرتاب می‌شوند تجسم کنید. هرگاه یک سنگ را درون این استخر بیاندازید، امواجی در همه جهت‌ها ایجاد می‌شوند و در سرتاسر استخر حرکت می‌کنند. آن‌ها به کرانه‌های استخر ضربه می‌زنند، طنین‌افکن می‌شوند و با دیگر امواج برخورد می‌کنند. گاهی ممکن است حتی مقداری از آب استخر به خارج بریزد. به عبارت دیگر برخورد سنگ با آب باعث ایجاد موج‌های کوچک زیادی شده‌است و تمام آب استخر را متلاطم نموده است. حال اگر داخل استخر را با موانعی به چند بخش تقسیم می‌کردیم به نحوی که مسیر حرکت امواج از هر بخش به بخش‌های دیگر مسدود می‌شد، قطعاً هنگامی‌که سنگی به درون استخر پرتاب می‌شد فقط آب‌های همان بخش از استخر متلاطم می‌شدند. امواج حاصل، امکان عبور از موانع به سایر بخش‌های استخر را نداشتند و سطح آب در بقیه بخش‌ها آرام می‌ماند به این ترتیب ما تغییرات را در استخر محدود کرده‌ایم. (شکل ۱-۲)



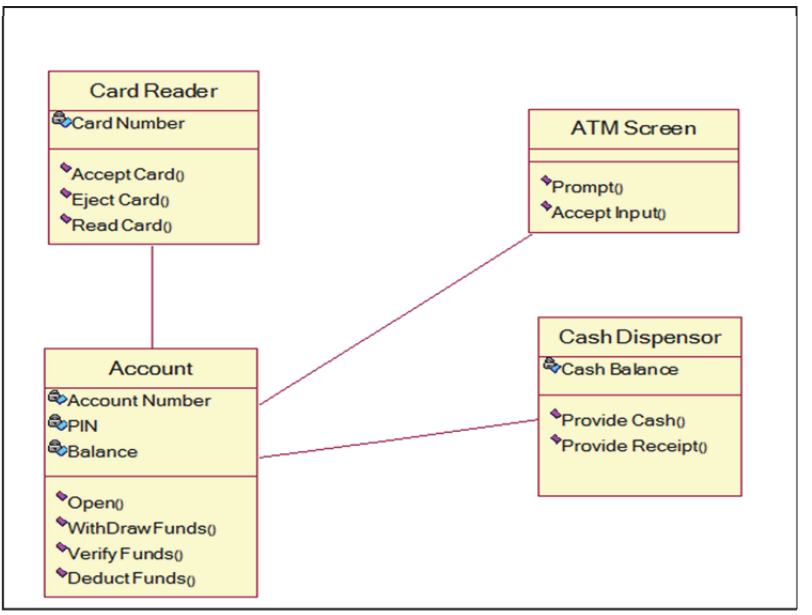
شکل ۱-۲. تقسیم استخر آب به چند بخش جهت محدودکردن تلاطم امواج

حال همین ایده مخفی‌سازی را در یک سیستم بانکی در نظر بگیرید. اخیراً مدیریت بانک تصمیم گرفته است که اگر مشتری در بانک یک حساب اعتباری دارد، بتوان از حساب اعتباری وی یک مبلغ اضافه برداشت کرده و به حساب جاری او منظور نمود. در یک سیستم نرم‌افزار غیرمخفی‌سازی‌شده (که احتمالاً خودمان طراح آن نبوده‌ایم)، کار را با یک روش غیرخودکار مرور می‌کنیم تا تجزیه و تحلیل مسئله آشکارتر شود.

روشن است که تمامی محل‌های عملیات برداشت از حساب در کد برنامه از قبل برای ما مشخص نیست. بنابراین باید تمام قسمت‌های کد برنامه را جست‌وجو کنیم و وقتی این عملیات را پیدا کردیم، باید یک سری تغییرات را ایجاد کنیم تا این درخواست جدید را یکپارچه کنیم. اگر واقعاً کار به درستی انجام شده‌باشد و ما خوشبین باشیم، احتمالاً ۸۰٪ موارد برداشت از حساب را پیدا کرده‌ایم. البته باید آن‌قدر حرفه‌ای باشیم که کارکرد سایر بخش‌های برنامه در نتیجه اعمال این تغییرات خراب نشود (تأثیر امواج به سایر بخش‌های استخر).

دوباره کار را با یک سیستم مخفی‌سازی‌شده از نو مرور می‌کنیم. برای شروع به مدل سیستم خود نگاه می‌کنیم و به آسانی جایی را که عملیات برداشت از حساب مخفی‌سازی شده‌بود پیدا می‌کنیم (متد withdrawFunds() در کلاس Account). کافی است عملیات تغییر یافته را در کلاس حساب جایگزین کنیم، و کار تمام می‌شود. شکل ۱-۳ را ملاحظه کنید.

نکته دیگر در بحث مخفی‌سازی، پنهان‌سازی اطلاعات[1] است که نتیجه آن ایجاد محدودیت در سطوح دسترسی به اشیاء و مشخصات آن‌ها است. پنهان‌سازی اطلاعات، توانایی پنهان‌کردن جزئیات مبهم یک شیء از دنیای خارج است. دنیای خارج برای یک شیء به معنی هرچیزی خارج از همان شیء، حتی بقیه سیستم است. به عبارت دیگر هر شیء (یا کلاس) می‌تواند مشخصات خود را از دیگر اشیاء (یا کلاس‌های) سیستم مخفی نماید. نتیجه این کار افزایش انعطاف‌پذیری[2] در سیستم‌های شیءگراست.



شکل ۱-۳. مخفی‌سازی(مدل کلاس برای سیستم بانکداری)

در اغلب زبان‌های برنامه‌نویسی شیءگرا[3] برای پنهان‌سازی اطلاعات مفهومی به نام قابلیت رؤیت[4] یا میدان دید معرفی شده‌است. در این زبان‌ها، عموماً چهار نوع قابلیت رؤیت برای کلاس‌ها، اشیاء و صفات و عملیات آن‌ها تعریف می‌شود:

Public. میدان دید عمومی است. این قابلیت پیشنهاد می‌کند که کلاس می‌تواند به‌وسیله همه کلاس‌های دیگر موجود در سیستم دیده شود.

---

1. Information Hiding
2. Flexibility

۳ زبان‌های برنامه‌نویسی شیءگرا در فصل دوم کتاب معرفی شده‌است.

4. Visibility