به نام پروردگار دانایی

طراحی سیستم‌های شی‌گرا

برنامه‌نویسی شی‌گرا

درس اول: مقدمه‌ای بر اصول شی‌گرایی

سید کاوه احمدی

OOP Buzzwords

responsibility-driven design

inheritance

encapsulation

iterators

overriding

coupling

cohesion

javadoc

interface

collection classes

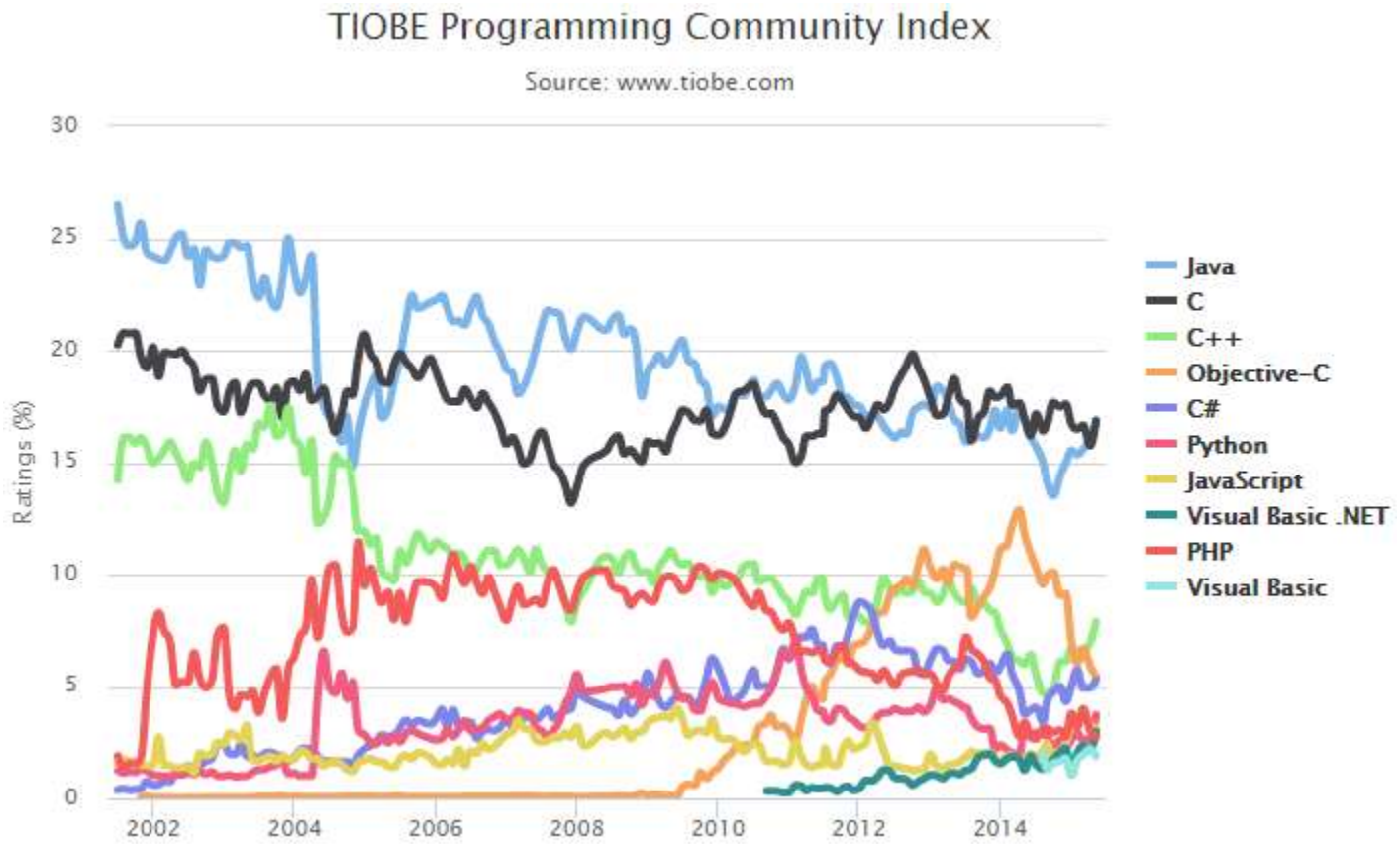
mutator methods

polymorphic method calls

چرا جاوا

- آشنایی با یک خانواده جدید از زبان‌های برنامه‌نویسی.
- جاوا همه جا اجرا می‌شود!
- زبان اول در کارهای آکادمیک و پروژه‌های `Enterprise`.
- رعایت صحیح کلیه اصول شی‌گرایی
- اندروید!

چرا جاوا



- TIOBE programming community index is a measure of popularity of programming languages
- TIOBE stands for "The Importance of Being Earnest" which is taken from the name of a comedy play written by Oscar Wilde at the end of the nineteenth century.
- The index is calculated from the number of search engine results for queries containing the name of the language

چرا جاوا: BlueJ

- یک محیط توسعه مجتمع (IDE: Integrated Development Environment)

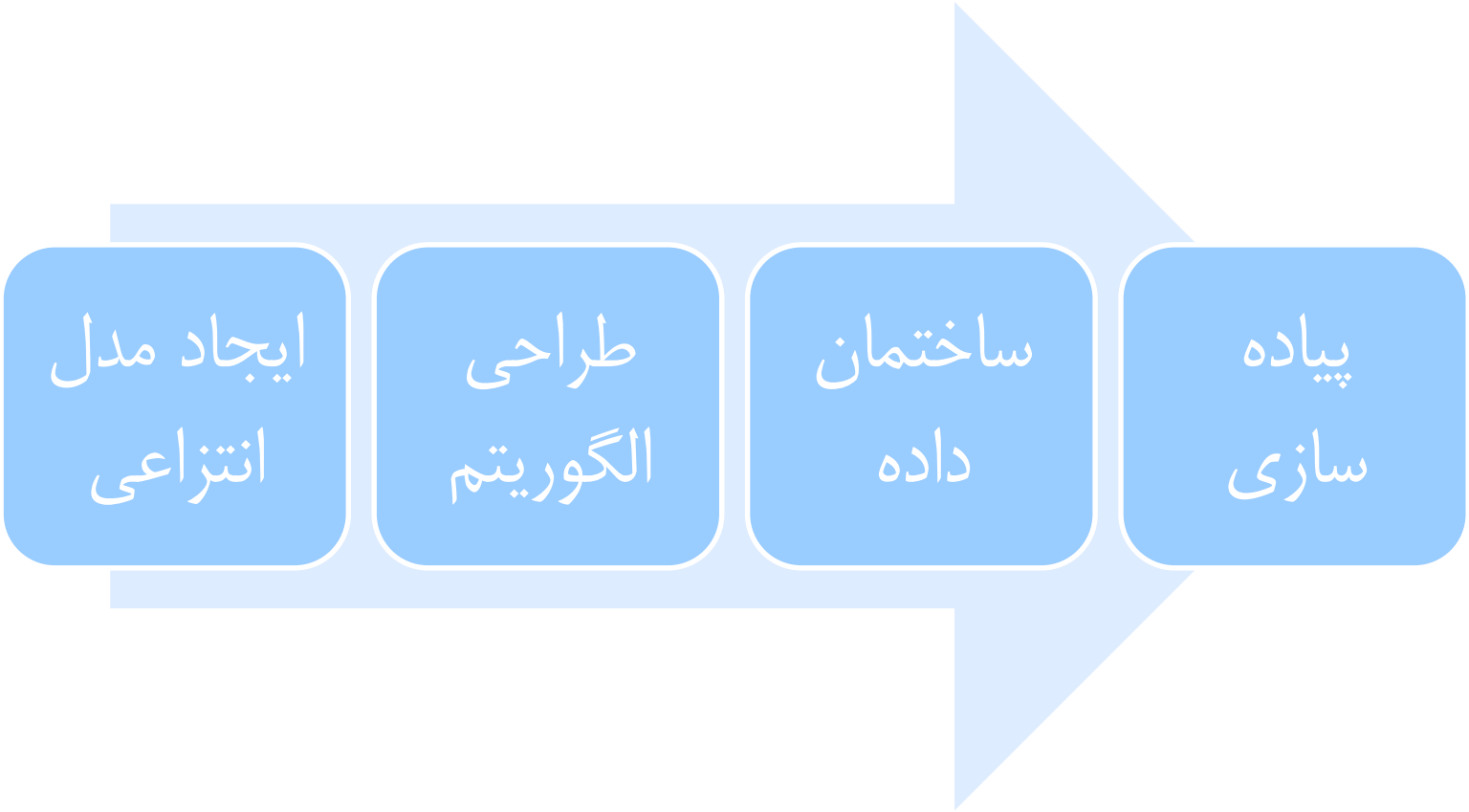
  برای یادگیری برنامه‌نویسی جاوا

  - هدف از توسعه این IDE آموزش برنامه‌نویسی شی‌گرا است
  - در طول درس از BlueJ استفاده خواهیم کرد.

- IDEهای مناسب برای کاربردهای حرفه‌ای:
  - **NetBeans**
  - **JetBrains IntelliJ IDEA**
  - **Eclipse**

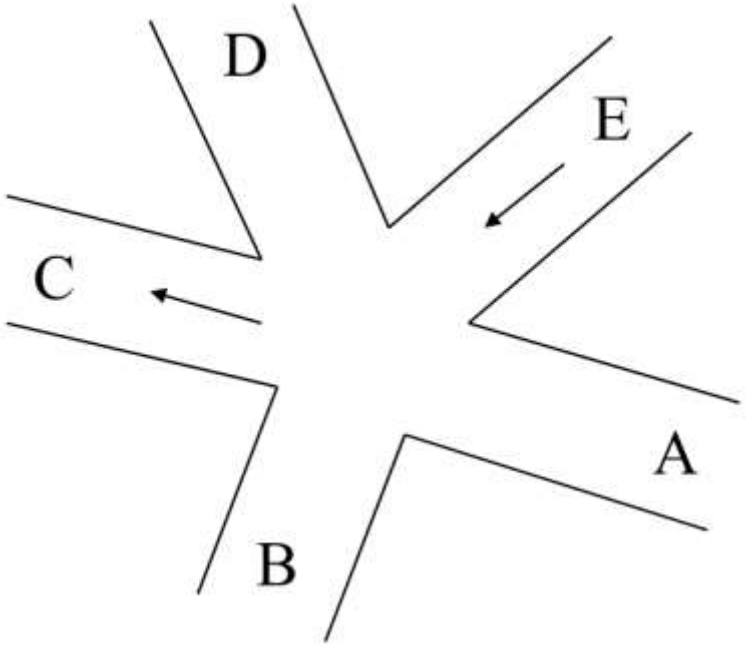
- یک کارشناس نرم‌افزار باید چه چیزهایی بداند؟

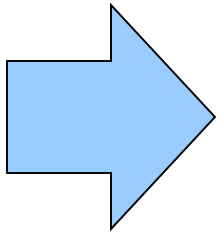
مراحل حل مسئله

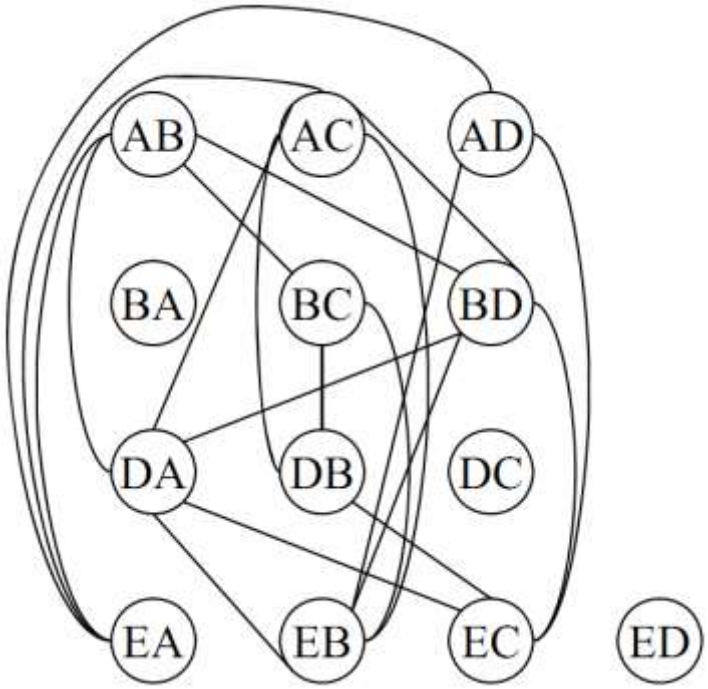


مراحل حل مسئله

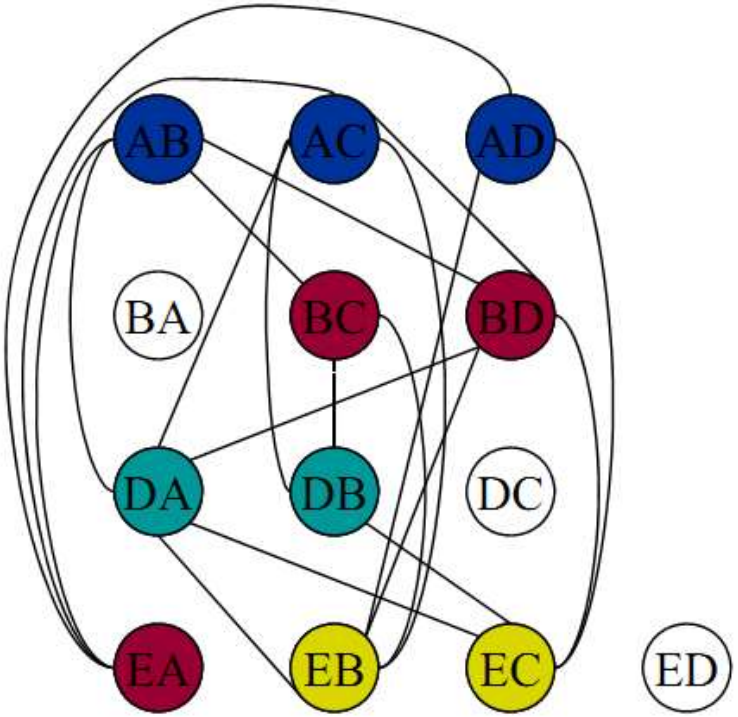
- هدف:
  - کمترین تعداد زمان‌های چراغ
  - عبور بیشترین ترافیک در هر زمان







مراحل حل مسئله



- **انتزاع:** در نظر نگرفتن جزئیات غیر ضروری
  - عرض خیابان‌ها
  - میزان ترافیک خیابان‌ها
- **مدل:**
  - مثلا AD به معنی گردش از A به D است
  - رسم یک یال بین گردش‌هایی که همزمان میسر نیست
- **الگوریتم:** این مسئله با الگوریتم رنگ آمیزی گراف‌ها قابل حل است.
  - رئوس مجاور نباید همرنگ باشند
- **ساختمان داده:**
  - چگونه گراف را پیاده‌سازی کرده آنرا رنگ آمیزی کنیم؟

[محمد قدسی – دانشگاه شریف]

- کمی درباره توسعه نرم‌افزار

کمی درباره توسعه نرم‌افزار

- یک توسعه دهنده نرم‌افزار با تجربه چه چیزهایی می‌داند؟
  - مشخصات فنی برخی زبان‌های برنامه‌نویسی
  - مهمتر: قواعد عمومی که در عمده زبان‌های برنامه‌نویسی مشترک است.
  - مهارت‌ها:
    - طراحی
    - اشکال زدایی
    - آزمون
    - `Refactoring`
    - مستندسازی
  - آموزش و یادگیری این مهارت‌ها مشکل است.
  - باید با تجربه به دست بیاید

یادگیری در تمام عمر

- توسعه دهنده نرم‌افزار خوب هیچگاه از یادگیری دست بر نمی‌دارند. بدون توجه به اینکه چه مدت است این کار را انجام می‌دهند.
  - یادگرفتن روش‌های جدید فکر کردن و نگرش به مسئله
- هیچ راه کوتاهتری برای رسیدن به این هدف در هیچ جایی وجود ندارد.
  - هرچه بیشتر تلاش کنید، بیشتر نتیجه می‌گیرید.

- اهمیت طراحی در توسعه نرم‌افزار
- اهمیت شی‌گرایی در طراحی و توسعه نرم‌افزار
- اصول شی‌گرایی تعریف شده برای نیل به طراحی مناسب
- ارتباط این اصول با برنامه‌نویسی شی‌گرا

انواع نرم‌افزار

- برنامه (Program Software)
  - مجموعه‌ای از دستورات/توابع/کلاس‌ها که برای انجام عملیاتی خاص نوشته می‌شود.
    - برنامه‌ای که اعداد اول را باز می‌گرداند.
    - برنامه‌ای که کوتاه‌ترین مسیر را در یک گراف پیدا می‌کند.
    - برنامه جستجو در ...

انواع نرم‌افزار

- نرم‌افزار کاربردی و تجاری (Application Software)
  - نرم‌افزاری شامل چندین برنامه که برای پاسخگویی به یک نیاز خاص نوشته می‌شود.
    - عمدتا مبتنی بر پایگاه داده
    - نرم‌افزار حسابداری، نرم‌افزار تحلیل ریسک، فتوشاپ، بازی‌ها و...

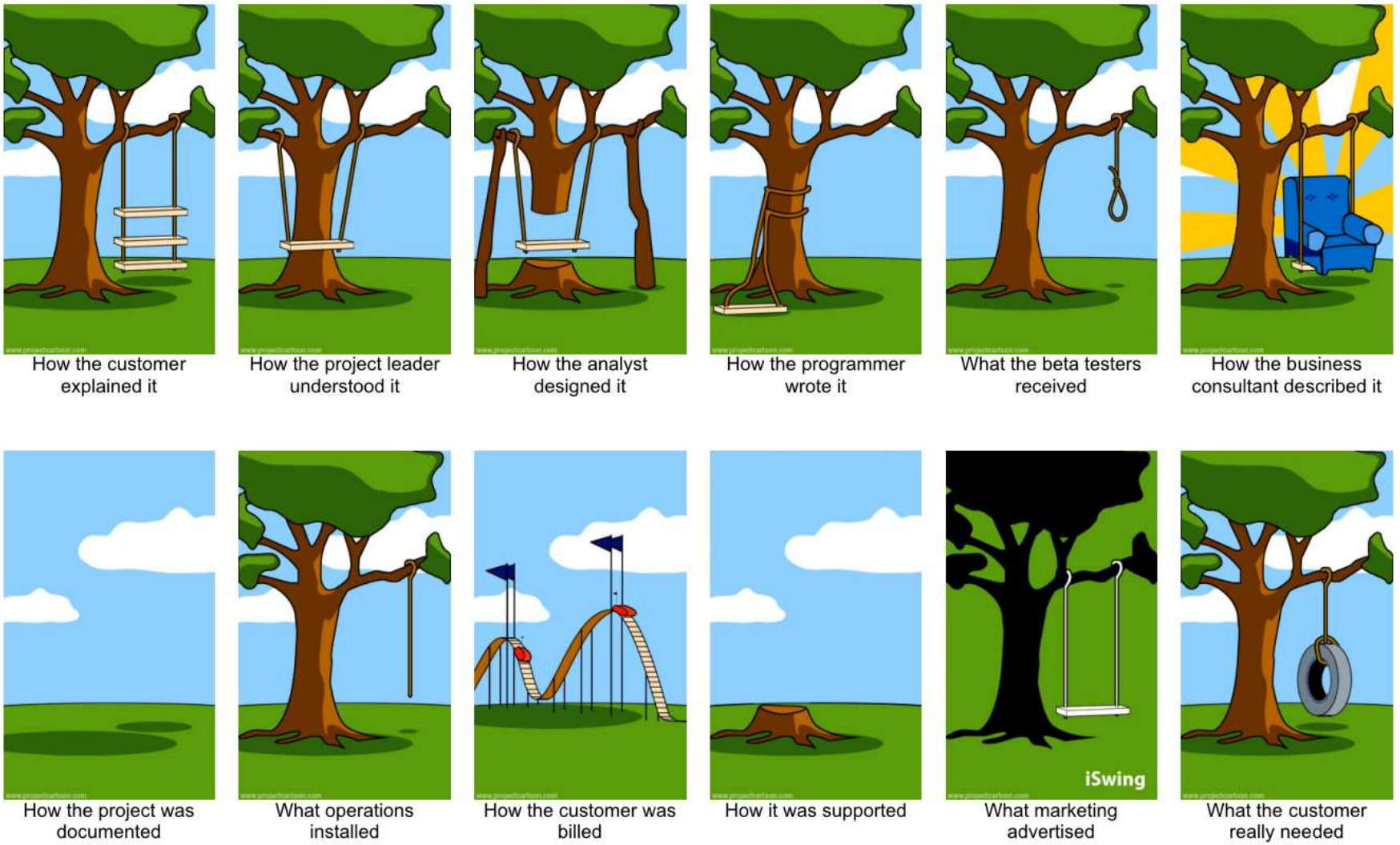
انواع نرم‌افزار

- نرم‌افزار سیستمی (System Software)
  - نرم‌افزارهایی که برای پنهان کردن پیچیدگی سخت‌افزار به کار گرفته می‌شود.
    - سیستم عامل، درایورهای سخت‌افزارها و...

بحران نرم افزار

- هزینه‌های هنگفت تولید نرم افزار
- عدم تحویل به موقع
- عدم تامین نیازمندی‌های کاربر
- کیفیت پایین و نامطمئن
- سختی نگهداری بعلت کیفیت پایین طراحی

توسعه برنامه کاربردی در عمل!



چه زمانی باید فاتحه مهندسی نرم افزار را خواند؟

- آخ...! یادم رفت!
- اِ... اینجوری هم می‌شد!
- عجب...! پس این را هم می‌خواستید؟!
- ببخشید! به جا نمی‌آورم! شما...؟! (در مواجه با یکی از اشخاص کلیدی مشتری)
- ای تف به این روزگار...! توی پروژه قبلی هم همین اشتباه رو کرده بودیم!
- الان که ۱۰۰ درصد را تحویل گرفته‌اید، خواهش می‌کنم حداقل پیش پرداخت پروژه را بدهید!
- تا اونجا که یادم می‌آید، این تیکه کد را دو سه بار دیگه هم Copy-Paste کرده بودم!
- رو سیستم خودمون که درست کار می‌کنه!
- مشکل از شبکه است!
- قبلا این مساله را به ما گفته بودین؟!
- این موضوع در RFP ذکر نشده بود!
- ببخشید حجم ریالی قرارداد شما با این درخواست مطابق نیست!

- [http://weblog.radmanitd.com/index.php/archives/2916]

چه زمانی باید فاتحه مهندسی نرم افزار را خواند؟

این برنامه دیگه به درد نمی‌خوره

باید از اول بنویسیمش!

چرا نوشتن نرم افزارهای تجاری مشکل است؟

- پیچیدگی مساله
  - اگر موضوعات ساده باشند که برای حل آنها به نرم افزار فکر نمی‌کنیم!
  - نیازمندی‌های گوناگون و متضاد
  - مکانیزه کردن تعاملات انسانی

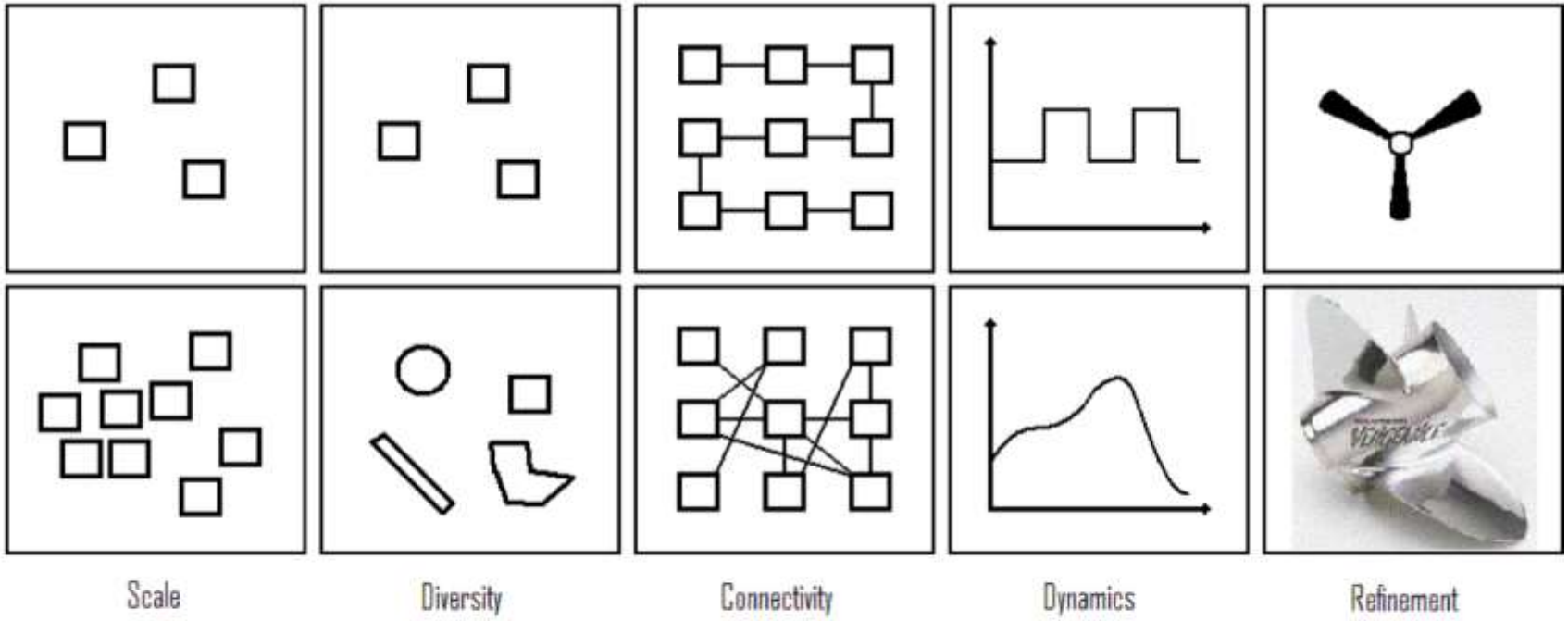
چرا نوشتن نرم افزارهای تجاری مشکل است؟

- تغییر رخ می‌دهد
  - تغییر ناشی از رفع ایرادات نرم افزار
  - تغییر به منظور بهبود عملکرد
  - افزودن امکانات جدید
  - تغییر نیازمندی‌ها
  - تغییر قواعد کسب و کار
  - تغییر قوانین حاکم بر فضای مسئله

چرا نوشتن نرم افزارهای تجاری مشکل است؟

- پیچیدگی فرآیند تولید
  - ماهیت مسائل ذاتا پیچیده است. ماهیت نرم افزار نیز همینطور!
  - ذهن انسان پیچیده است و پیچیدگی را دوست دارد!
  - استاندارد نبودن فرآیندهای مهندسی نرم افزار
    - فعالیت‌های مهندسی نرم افزار مستلزم طراحی‌های نوآورانه است و به شکل روتین قابل انجام نیست.
    - مهندسی نرم افزار هنوز در حال بکارگیری به روش غیر اصولی (non-systematic) است.
    - نسبت به سایر تخصص‌های مهندسی سازماندهی کمتری دارد
    - نسبت به برخی از رشته‌های مهندسی، استانداردهای کمتری دارد
    - بستن طراحی و کامل شدن آن تا انتهای پروژه به طول می‌انجامد
    - دستاورد اصلی آن مهیتی غیرقابل لمس دارد (محصول انتزاعی است و نه فیزیکی)

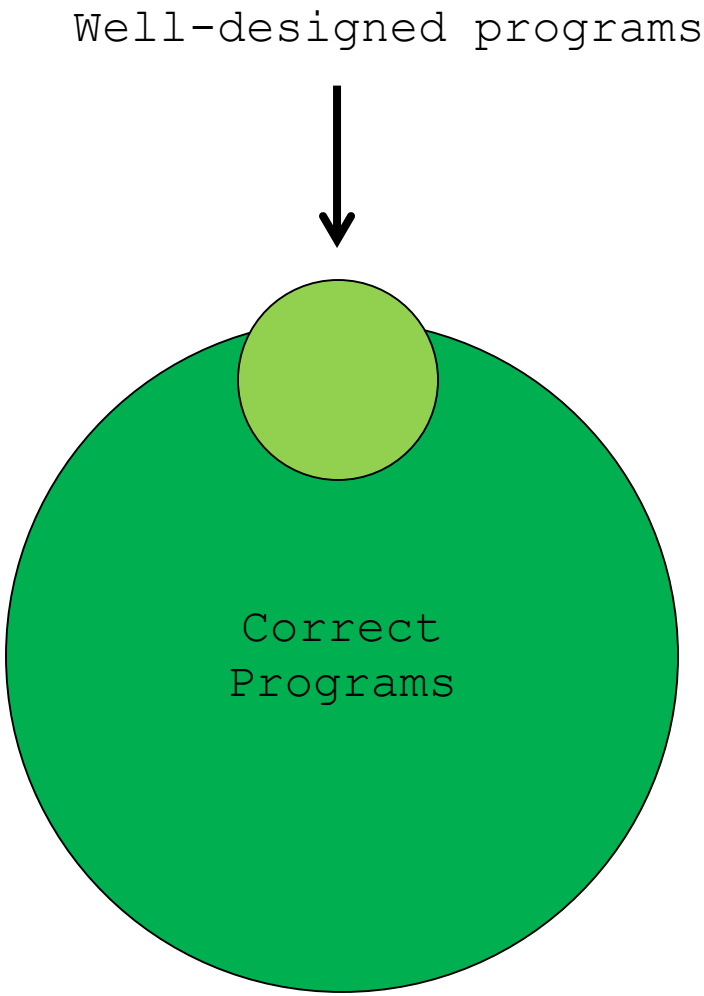
چرا نوشتن نرم افزارهای تجاری مشکل است؟



چرا نوشتن نرم افزارهای تجاری مشکل است؟

- نمی‌توان پیچیدگی را از بین برد بلکه باید آنرا کنترل کرد.
  - پیچیدگی نرم افزار با پیچیدگی سیستم‌های طبیعی و محصولات فیزیکی ساخت دست بشر متفاوت است و یک خاصیت ذاتی سیستم‌های نرم افزاری بزرگ می‌باشد.

اهمیت طراحی خوب در نرم‌افزار



- کدامیک؟
  1. یک برنامه که درست کار می‌کند؟
  2. یک برنامه که به خوبی طراحی شده است؟

- بینهایت برنامه وجود دارد که کارهای مورد انتظارشان را به خوبی انجام می‌دهند
- تعداد محدودی به خوبی طراحی شده‌اند

اهمیت طراحی خوب در نرم‌افزار

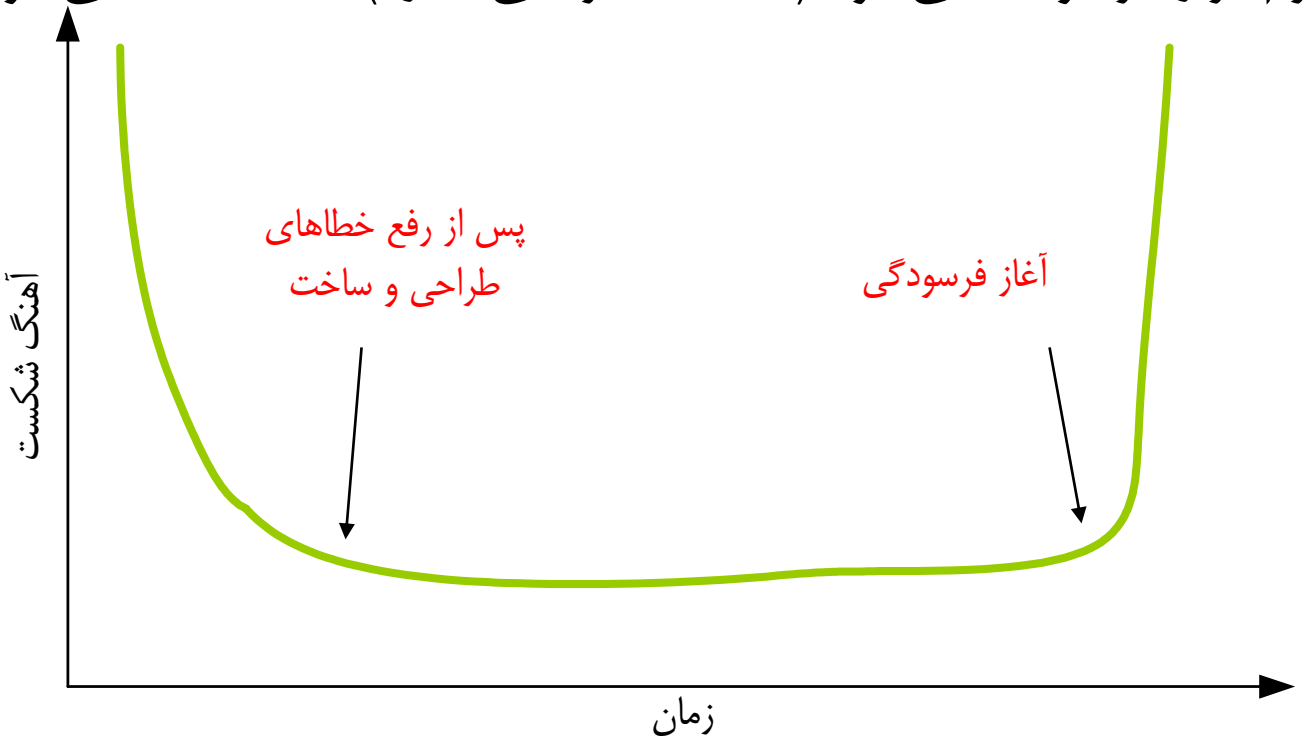
- با یک طراحی خوب
  - نوشتن یک برنامه با طراحی خوب راحت‌تر است.
  - توسعه، به‌روز رسانی و تغییر ساده‌تر است.

اهمیت طراحی خوب در نرم‌افزار

- جلوگیری از فاسد شدن سریع نرم افزار
- کنترل پیچیدگی

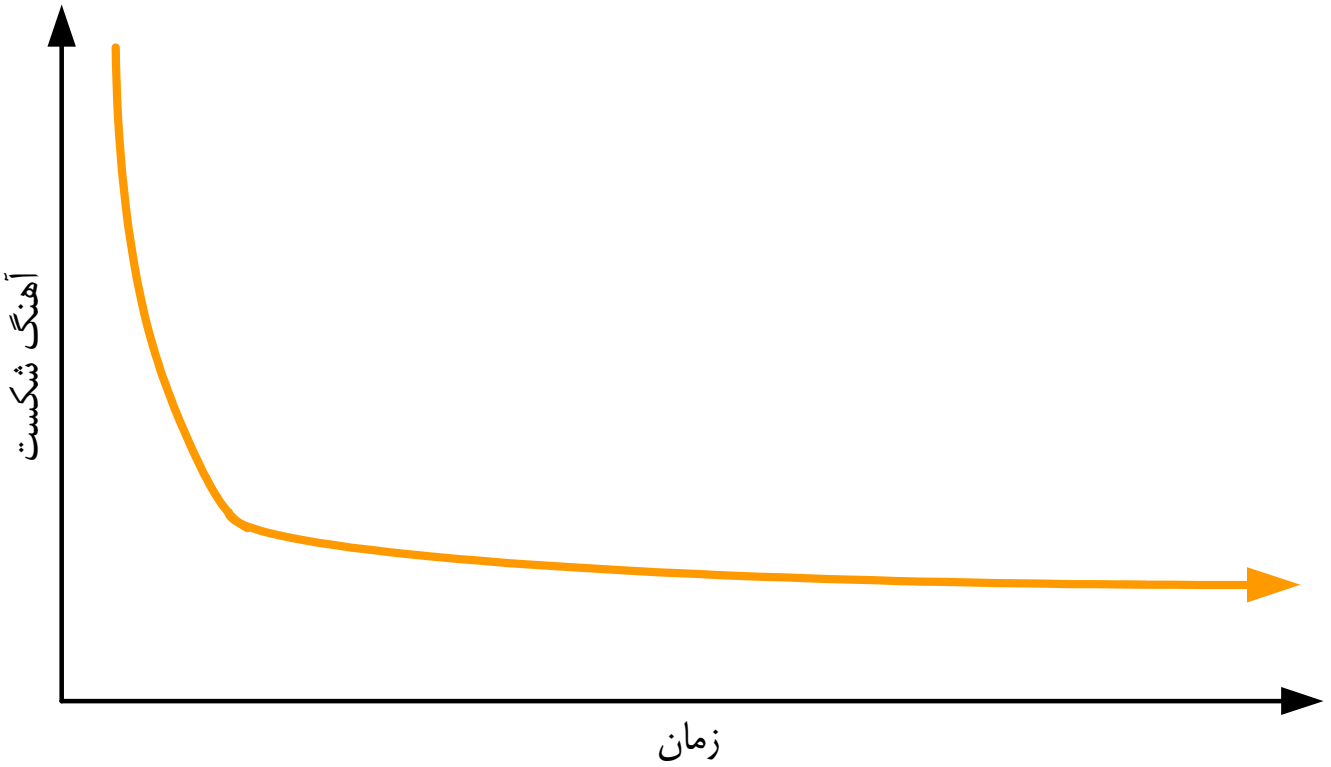
اهمیت طراحی خوب در نرم‌افزار

- در نرم‌افزار فقط یک چیز است که تغییر نمی‌کند: **تغییر**
- بنابراین نرم‌افزار فرسوده نمی‌شود (ماهیت فیزیکی ندارد) بلکه فاسد می‌شود.



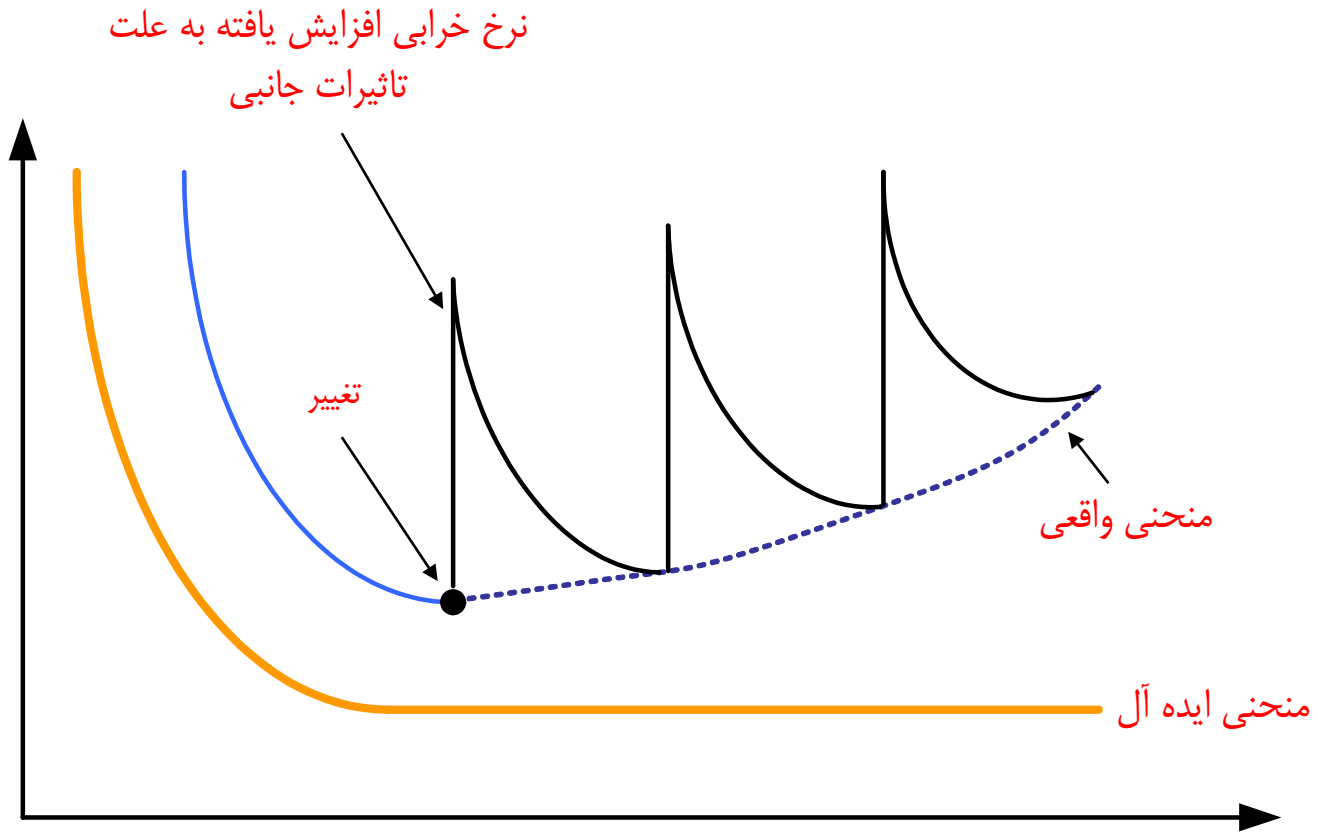
منحنی نرخ خرابی سخت افزار نسبت به زمان

اهمیت طراحی خوب در نرم‌افزار



منحنی ایده‌آل آهنگ شکست نرم افزار نسبت به زمان

اهمیت طراحی خوب در نرم‌افزار



منحنی نرخ خرابی واقعی نرم افزار نسبت به زمان

- Simplicity is prerequisite for reliability.
- Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.

Edsger W. Dijkstra

First Law of Software Quality

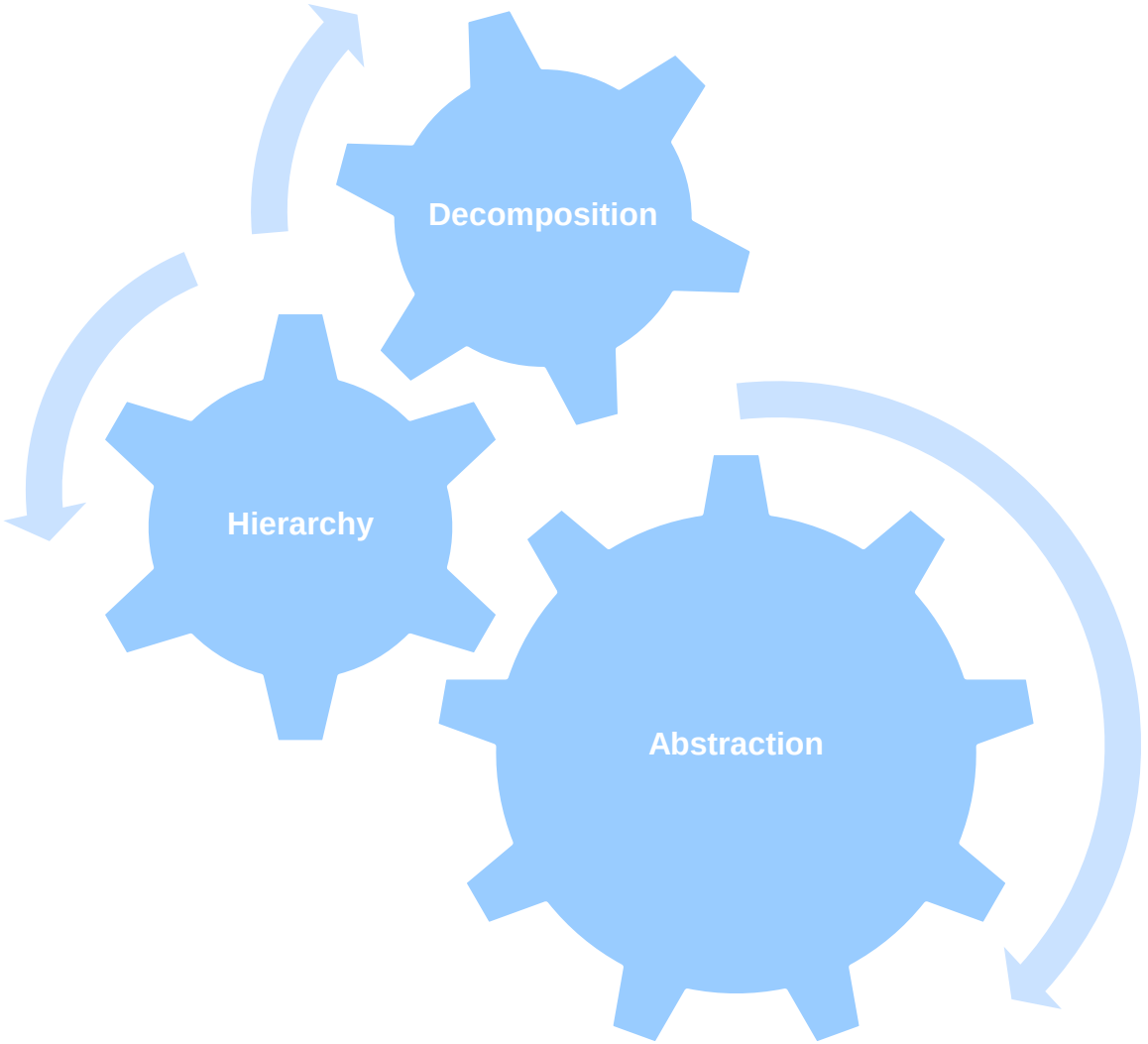
$$e = mc^2$$

$$\text{errors} = (\text{more code})^2$$

چرا طراحی شی‌گرا؟

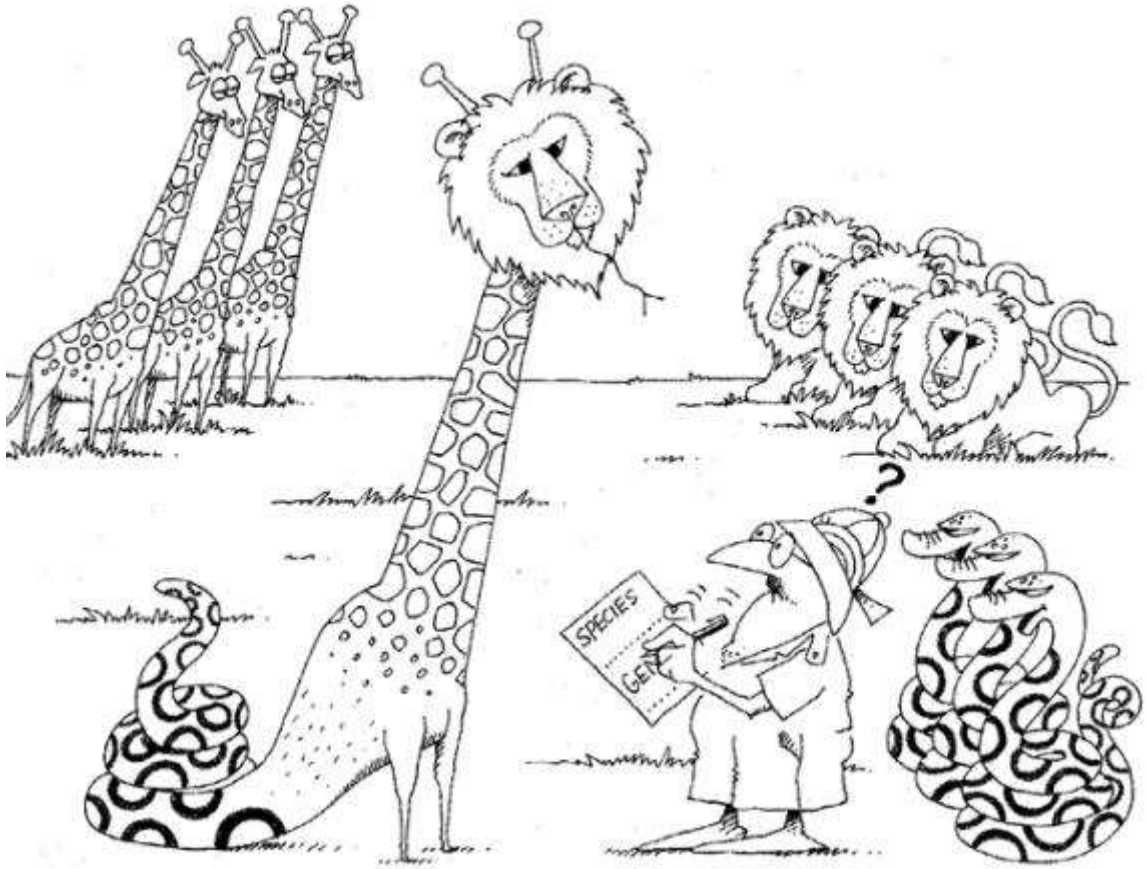
- در اغلب سیستم‌های پیچیده، پیچیدگی به صورت سلسله مراتب (Hierarchy) ظاهر می‌شود.
  - سلسله مراتب‌های Is-A یا Has-A
  - در سیستمی که از چند زیرسیستم تشکیل می‌گردد، ارتباط بین اجزای درونی هر زیر سیستم (Intra-component Linkage) قوی‌تر از ارتباط بین زیرسیستم‌ها (Inter-components Linkage) است.
- سیستم‌های سلسله مراتبی معمولا از تعداد کمی از زیر سیستم‌های مشخص و متفاوت تشکیل می‌شوند که این زیرسیستم‌ها به صورت‌های گوناگون و ترتیب‌های مختلف ظاهر می‌شوند.
- معمولا سیستم‌های پیچیده که به صورت محکم و استوار عمل می‌کنند حاصل تکامل سیستم‌های ساده‌ای هستند که به درستی عمل می‌کردند.

Bring Order to Chaos



Bring Order to Chaos

- Classification



چرا طراحی شی‌گرا؟

- ماهیت شی‌گرایی به نحوی است که می‌تواند با توجه به ویژگی‌های سیستم‌های پیچیده، آنرا کنترل کند.
  - این ماهیت دلیل اصلی انتخاب رویکرد شی‌گرا است.
  - برخی به اشتباه فکر می‌کنند چون انسان با اشیا سروکار دارد و طراحی با این مفاهیم آشنا برایش ساده‌تر است پارادایم شی‌گرا شکل گرفته است!

چرا طراحی شی‌گرا؟

- طراحی شی‌گرا، طراحی خوب را تشویق می‌کند.
  - اگر تمام اصول و قواعد رعایت شود و فقط با طراحی کلاس خود را فریب ندهیم، طراحی مناسبی خواهیم داشت.
- طراحی شی‌گرا تنها راه رسیدن به یک طراحی خوب نیست.
  - اما کماکان از روش‌های شناخته شده‌ی دیگر بهتر است.

چرا طراحی شی‌گرا؟

- اشیا انتزاعی از موجودیت‌های واقعی سیستم هستند.
- اشیا مستقل از هم هستند.
  - می‌توانند به صورت مجرد طراحی و پیاده‌سازی شوند.
  - به این شکل می‌توان به شکاندن پیچیدگی مسئله کمک کرد.
- اطلاعات اشیا محصور شده است.
  - تغییرات از بیرون را نمی‌پذیرد.
  - با دریافت پیغام‌ها خودش وضعیت خودش را تغییر می‌دهد.

چرا طراحی شی‌گرا؟

- به دلیل ویژگی‌های ذکر شده در اسلاید قبل، در صورت طراحی درست، تغییر در یک بخش از نرم‌افزار منجر به تغییر در سایر بخش‌ها نخواهد شد.
  - نگهداری ساده‌تر
  - تغییر کم هزینه‌تر
  - بالا رفتن طول عمر نرم‌افزار

چرا طراحی شی‌گرا؟

- قابلیت استفاده مجدد از کد

  - همانطور که ذکر شد، سیستم‌های پیچیده از زیر سیستم‌های مشخص و متفاوت تشکیل شده که این زیر سیستم‌ها به صورت‌های گوناگون در نقاط مختلف ظاهر می‌شوند.
    - در بحث برنامه‌نویسی شی‌گرا تحت عنوان چند ریختی آنرا خواهیم دید.
  - ضمن آنکه ایده‌آل ما این است که بتوانیم همانند سخت افزار با مونتاژ کردن قطعه کدهای مختلف کنار هم (اینجا اشیا) برنامه‌های جدید بسازیم

چرا طراحی شی گرا؟

*OO is about **not** writing code*

The more you can reuse code,
the less you have to write

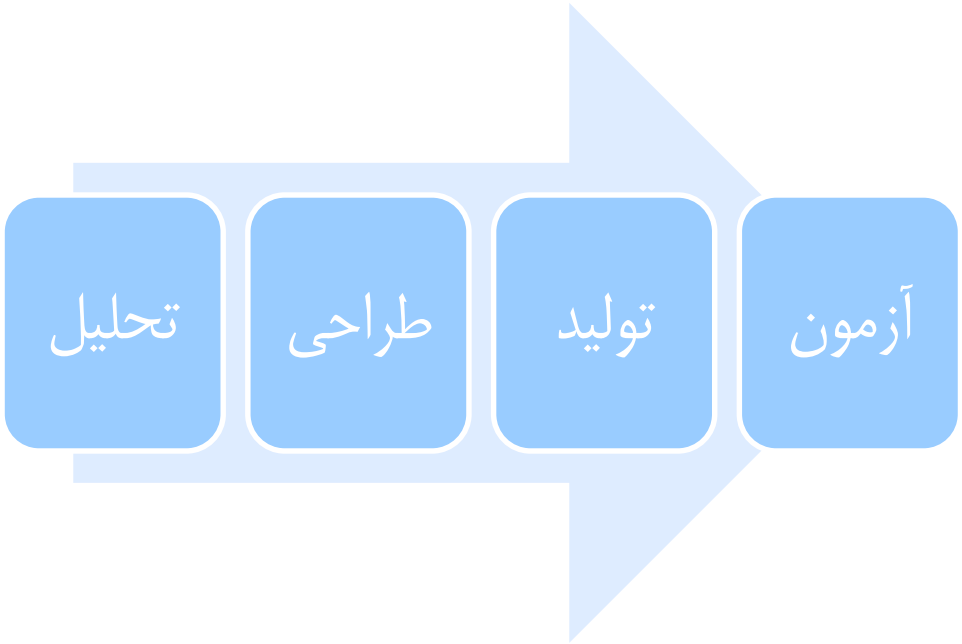
چرا طراحی شی‌گرا؟

- کدام کدها می‌تواند مورد استفاده مجدد قرار گیرد؟
  - کتابخانه کلاس‌ها: مثلا جاوا دارای هزاران از آن است.
  - کدهایی که توسط دیگران نوشته شده است.
  - کدهای خودتان

برقراری موازنه در طراحی

- در طراحی نرم‌افزار بین چند هدف باید توازن برقرار ساخت. می‌خواهیم سیستمی طراحی کنیم که:
  - ساده باشد
  - پیاده سازی آن آسان باشد
  - نگهداری و توسعه آن آسان باشد
  - اجرای آن سریع باشد
  - به حافظه کمی نیاز داشته باشد
  - درک آن برای سایر برنامه‌نویسان ساده باشد
  - استفاده از آن ساده باشد
  - به درستی کار کند
  - قابل اجرا روی پلت فرم‌های دیگر باشد
  - تمام امکانات مورد نظر همه کاربران را داشته باشد
  - ...

توسعه برنامه کاربردی



- با نگرش شی‌گرا:
  - تحلیل شی‌گرا (OOA)
  - طراحی شی گرا (OOD)
  - برنامه‌نویسی شی‌گرا (OOP)

– اصول اساسی شی‌گرایی

اصول اساسی شی گرایی

- تجرید (Abstraction)
- واحدبندی (Modularity)
- محصور سازی (Encapsulation)
- سلسله مراتب (Hierarchy)

مفاهیم کلیدی در برنامه‌نویسی شی‌گرا

- اشیا و کلاس‌ها (Objects & Classes)
- محصور سازی (Encapsulation)
- وراثت (Inheritance)
- چند ریختی (Polymorphism)

تجرید

- فرآیند متمرکز شدن روی ویژگی‌های اصلی یک پدیده از یک زاویه دید مشخص.
  - تمرکز روی بخش‌هایی که برای ناظر مهم است نه اینکه واقعا مهم باشد.
- نادیده گرفتن ویژگی‌های موقت و غیر مهم آن پدیده.
  - نادیده گرفتن جزئیات برای متمرکز شدن توجه بر سطح بالاتری از یک مسئله.
- نادیده گرفتن جزئیات در زمان مناسب
  - فکر کردن به آنچه یک متد/شی/کلاس انجام می‌دهد، نه چگونگی انجام آن.
  - برخورد کردن با اجزای سیستم به عنوان یک `black box`

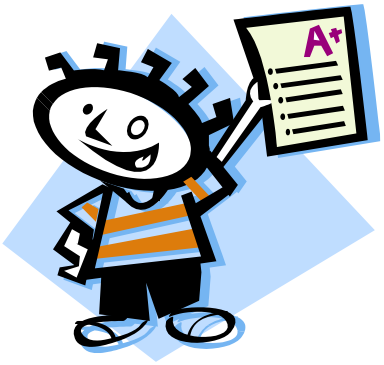
تجرید

- **یک مثال:** بدن انسان
  - سیستم گردش خون یک تجرید از بدن انسان است.
    - نگاه به بدن انسان از یک زاویه دید مشخص
    - اسکلت بندی بدن انسان مهم است و در تجرید گردش خون آنرا در نظر می‌گیریم اما با جزئیات آن کاری نداریم (در اینجا برای ما یک `black box` است).

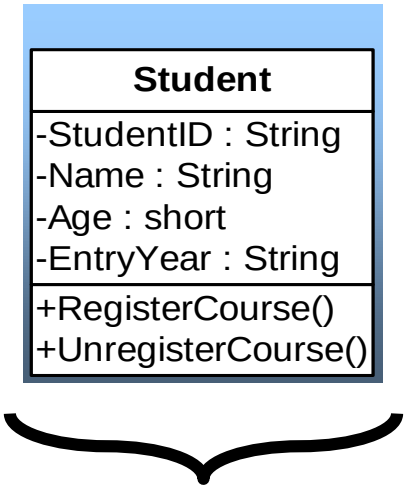
انواع تجرید

- تجرید موجودیت (Entity Abstraction)

Real Object: Student



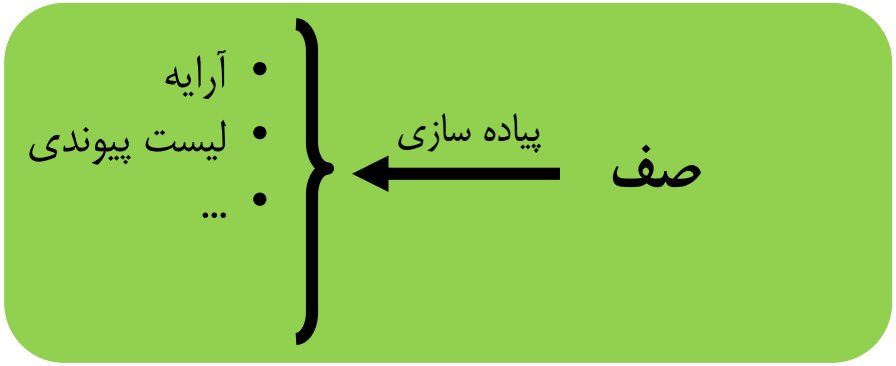
Abstraction: Student



بخشی از ویژگی‌ها و رفتارها از بیرون دیده می‌شود. نحوه پیاده‌سازی رفتارها را نمی‌دانیم.

انواع تجرید

- تجرید رفتار (Behavioral Abstraction)
  - به نحوه‌ی پیاده‌سازی صف کاری ندارم. برایمان اهمیتی هم ندارد. مهم این است که صف باشد!



ویژگی‌های تجرید

- تجرید با نمود خارجی یک شی سروکار دارد. به بیان دیگر تجرید آن بخش‌هایی را نگاه می‌کند که یک ناظر بیرونی آنرا می‌بیند.
  - می‌داند سیستم چه کاری انجام می‌دهد اما نمی‌داند چگونه
  - می‌داند سیستم از چه بخش‌هایی تشکیل شده.
  - رفتار از بیرون سیستم – مثال استفاده از اتومبیل
- تجرید داری سطوح مختلفی است. (میزان پرداختن به جزئیات)
  - سطح تجرید بالاتر: جزئیات کمتر – وسعت دید بیشتر
  - مثال هواپیما و ارتفاع

تجرید و پیچیدگی

- یکی از ابزارهای اصلی کنترل و تسلط بر پیچیدگی
  - جزئیات بی شماری درباره یک پدیده مطرح است
  - تنها ابعاد اساسی پدیده مد نظر قرار خواهد گرفت
  - به اجزای مختلف به صورت `black box` نگاه می‌شوند که با استفاده از واسطهای تعریف شده با هم تعامل دارند

محصور سازی

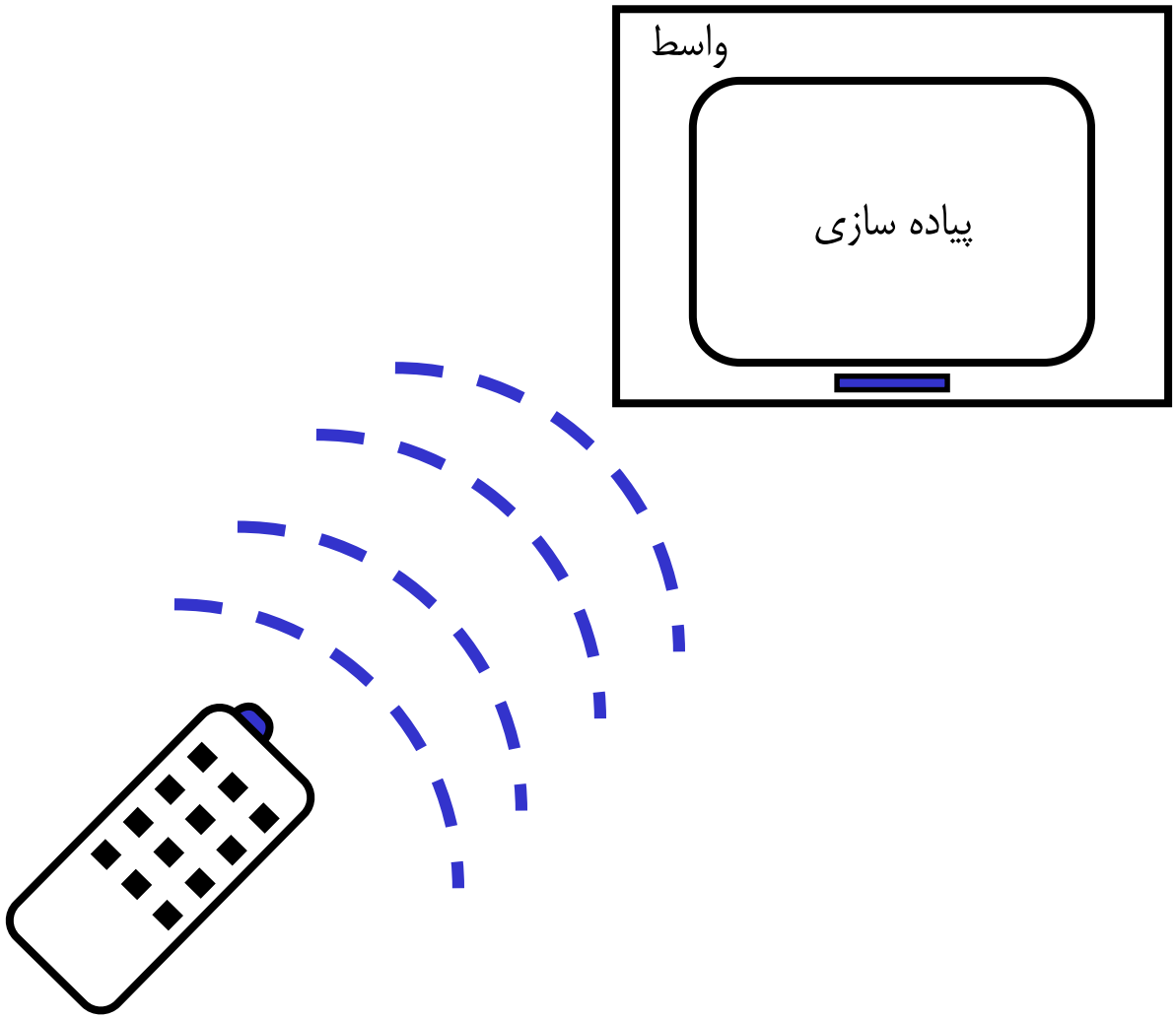
- محصور سازی عبارتست از محدود کردن روش‌های دسترسی یا استفاده از یک شی به طوری که از تاثیرات ناخواسته (`Side Effects`) یا کنترل نشده جلوگیری شود.
- اصل محصور سازی بر پایه اصل مخفی کردن اطلاعات است.
  - این اصل می‌گوید هر شی باید آن میزان از اطلاعات را در اختیار داشته باشد که به آن نیاز دارد.

محصور سازی

- از نظر محصور سازی هر شی به دو بخش اساسی تقسیم می‌شود:
  - واسطها (Interface)
  - ساختار داخلی
- سایر اشیا نمی‌دانند واسطهای شی چگونه پیاده‌سازی شده و کارهای مورد نظر را چگونه انجام می‌دهد.
- هر شی برای تعامل با سایر اشیا از واسطهای آن استفاده می‌کند.

محصور سازی

- ارتباط بین اشیا تنها از راه واسط‌ها



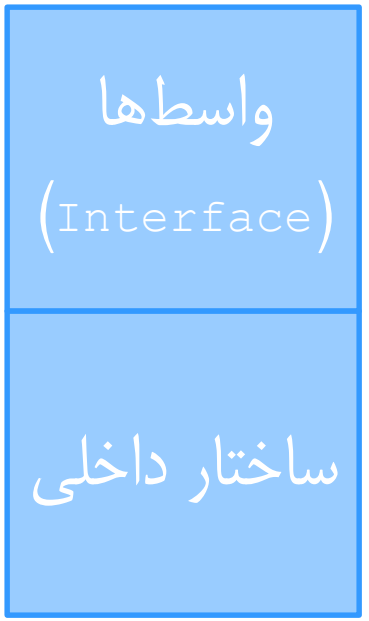
محصور سازی و پیچیدگی

- هر ماژول برای تعریف کننده آن باز و برای استفاده کننده آن بسته است.
  - جلوگیری از خرابکاری‌های احتمالی و محلی کردن گستره خطاها در شی
- با ثبات داشتن واسط یک شئ، می‌توان هر تغییری در پیاده سازی آن شئ انجام گیرد
  - هر ماژول برای توسعه باز و برای تغییر بسته است.
    - با تغییر واسطها، اشیای استفاده کننده از کلاس نیز برای سازگاری از واسطهای جدید نیاز به تغییر پیدا می‌کنند.
      - تغییر در بخش منجر به تغییر در سایر بخش‌ها خواهد شد!

تجرید و محصور سازی

- تجرید و محصور سازی دو مفهوم مکمل هستند
  - تجرید: رفتارهای قابل مشاهده یک شی (نمود خارجی)
  - محصور سازی: پیاده سازی رفتارها (نمود داخلی)
- تجرید مکانیزم تعیین جزئیاتی است که باید پنهان شود.
- محصورسازی، فرایند پنهان سازی جزئیات و کنترل دسترسی به آن است.
- محصور سازی یک مفهوم نسبی.

تجرید و محصور سازی



- برای تعامل با سایر اشیا
- اشیا دیگر دسترسی دارند
- **تجرید با آن درگیر**

- پیاده‌سازی رابط‌ها
- اشیا دیگر دسترسی ندارند
- **محصور سازی با آن درگیر**

یک واسط، چند پیاده سازی:
ویژگی‌ای که به یک رابط امکان می‌دهد تا **برای گروهی از عملیات** مورد استفاده قرار گیرند.

چندریختی (Polymorphism)

مفهوم یک واسط، چند پیاده سازی امکان استفاده مجدد را بالامی برد

واحدبندی

- پروسه تقسیم‌بندی یک مسئله به قسمت‌های خوش تعریف است به نحوی که بتوانند به طور جداگانه تولید و آزمایش شوند و به روش‌های خوش تعریفی بتوانند تعامل برقرار کنند.
- سیستمی را واحدبندی شده می‌گویند که به مجموعه‌ای از واحدهای منسجم و معنی‌دار که وابستگی بین آنها حداقل است تجزیه شده باشد.
- ماژول‌ها واحد تشکیل دهنده ساختار فیزیکی سیستم نرم افزاری هستند.

واحدبندی

- واحدها باید ویژگی‌های Building Blocks را داشته باشند:
  - استقلال (Independence)
  - واسطهای خوش تعریف (Well-defined Interfaces)
- یک مجموعه از تجریدهای مرتبط
- واحدبندی قابلیت استفاده مجدد را بالا می‌برد.

واحدبندی

- انسجام (Cohesion): درجه ارتباط عملکردهای عناصر داخلی یک ماژول دارای است.
- وابستگی (Coupling): درجه ارتباط واحدهای گوناگون به یکدیگر.
- درون سیستمی قوی – برون سیستمی ضعیف

واحدبندی

می‌خواهید برای تعطیلات به اسپانیا بروید

- می‌توانید مسئله را به چند واحد تقسیم کنید:
  - رفتن به فرودگاه
  - پرواز به اسپانیا
  - رفتن از فرودگاه در اسپانیا به هتل
- هر واحد می‌تواند به صورت جداگانه حل شود. این کار مسائل را ساده‌تر می‌کند.
  - پیدا کردن یک تاکسی تلفنی برای رفتن به فرودگاه، یک آژانس هواپیمایی برای رفتن به اسپانیا و یک اتوبوس شاتل برای رفتن به هتل

واحدبندی و پیچیدگی

- شکستن مساله به اجزائی کوچکتر یکی از راههای کارا برای مقابله با پیچیدگی است.
- مثال: اگر مسئله P را به زیر مسئله‌های P_1، P_2 و P_3 تقسیم کنیم آنگاه
  - $C(P) > C(P_1) + C(P_2) + C(P_3)$
  - $E(P) > E(P_1) + E(P_2) + E(P_3)$
  - C: Complexity
  - E: Solving Energy

واحدبندی و استفاده مجدد

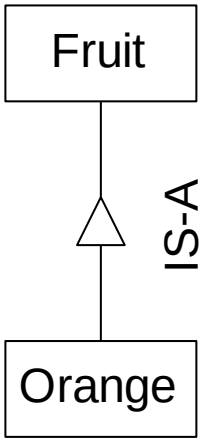
- اگر شرایط بیان شده در تعریف واحد بندی رعایت گردد، ماژول‌های با قابلیت استفاده مجدد بالایی می‌توان ایجاد کرد.
  - استفاده مجدد به معنی `Copy` و `Paste` نیست!
- تعیین معیار شکستن یک مساله مهمترین عامل برای موفقیت استفاده از این ویژگی است.

سلسله مراتب

- سلسله مراتب عبارت از مرتب ساختن تجریدها در سطوح مختلف.
- برای یک سیستم می‌توان سلسله مراتب مختلفی را در نظر گرفت. سلسله مراتب درک ما را از سیستم بالا می‌برد.
- انواع سلسله مراتب :
  - سلسله مراتب ساختار کلاس (IS-A)
  - سلسله مراتب ساختار شی (PART-OF)

سلسله مراتب IS-A

- سلسله مراتب ساختار کلاس.
- رابطه‌ی زیر نوع بین انواع مختلف اشیا را نشان می‌دهد.
- نام دیگر این نوع سلسله مراتب تخصیص / تعمیم (Generalization/Specialization) می‌باشد.



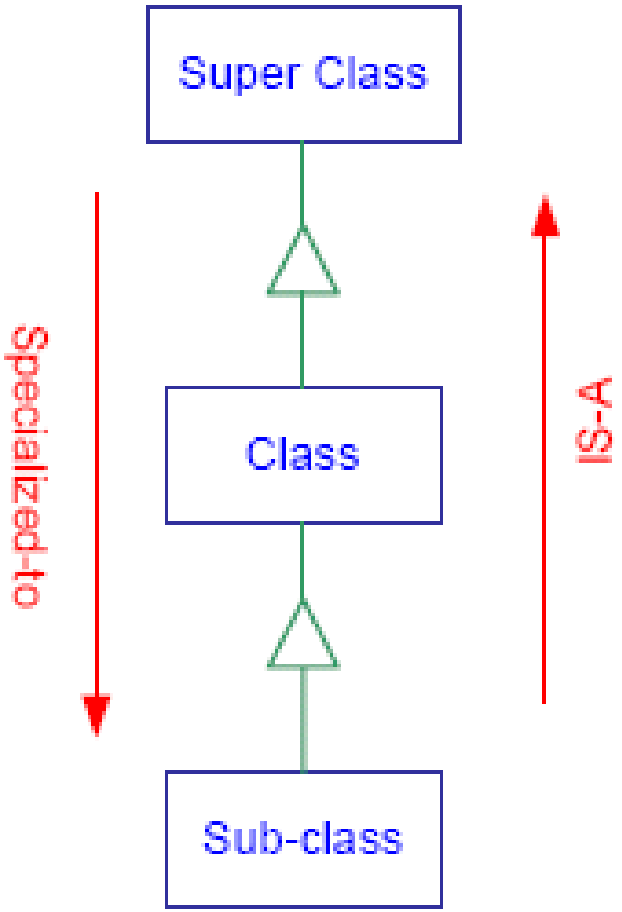
Orange IS-A Fruit

وراثت (Inheritance)

- وراثت مهمترین شکل سلسله مراتب IS-A است.

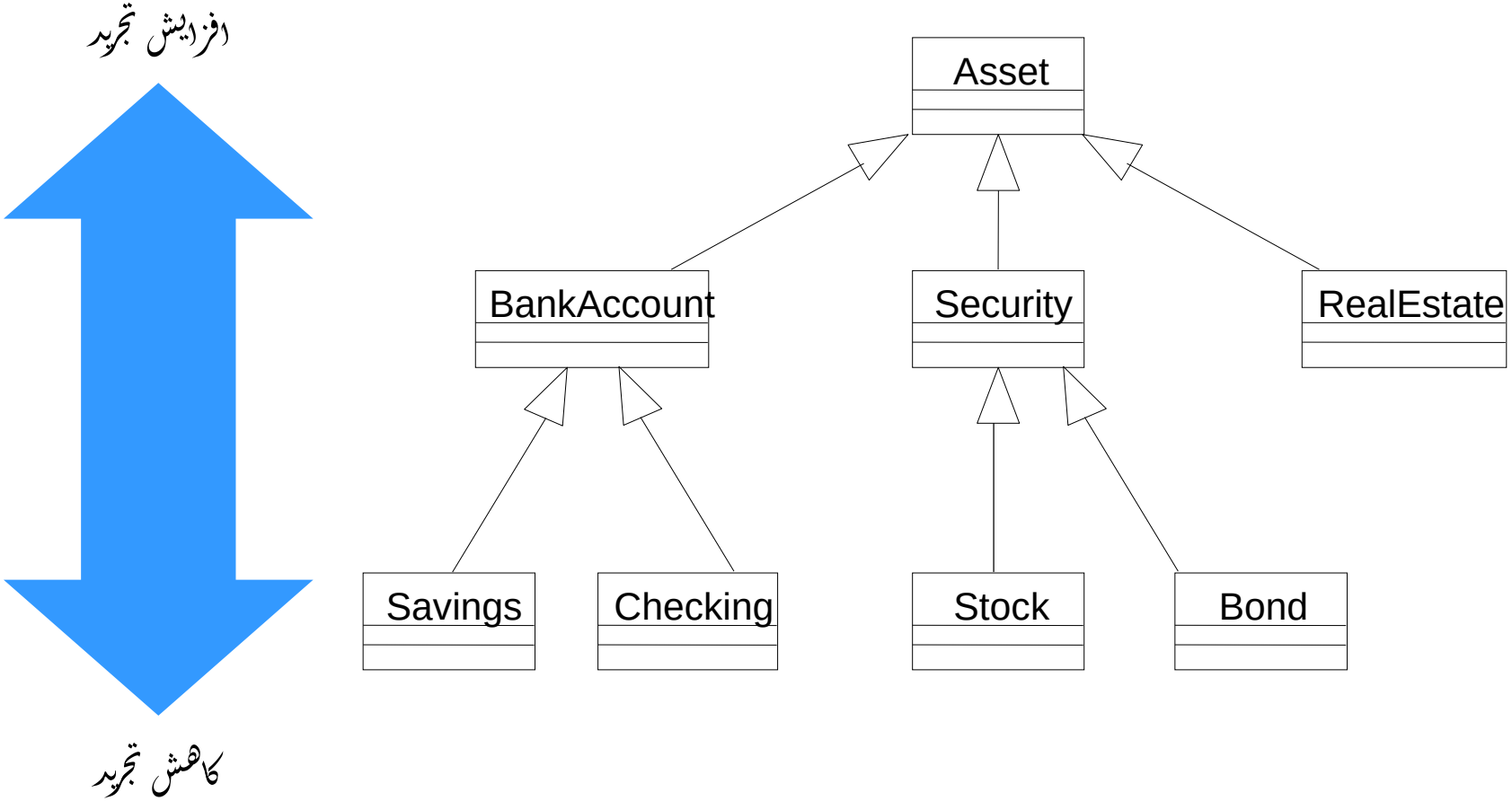
- وراثت عبارت است از رابطه بین چند کلاس که در آن یک کلاس در ساختار ، رفتار یا هر دو با یک کلاس (Single Inheritance) یا چند کلاس (Multiple Inheritance) دیگر شرکت دارد.
- کلاس فرزند یک تخصیص از کلاس عمومی‌تر (کلاس پدر) را نمایش می دهد.
- Type → Subtypes (Person / Student)
- کلاس‌های فرزند با اضافه کردن متدها و خصوصیات جدید، کلاس پدر را گسترش می‌دهند
- استفاده مجدد از کدها

وراثت



وراثت

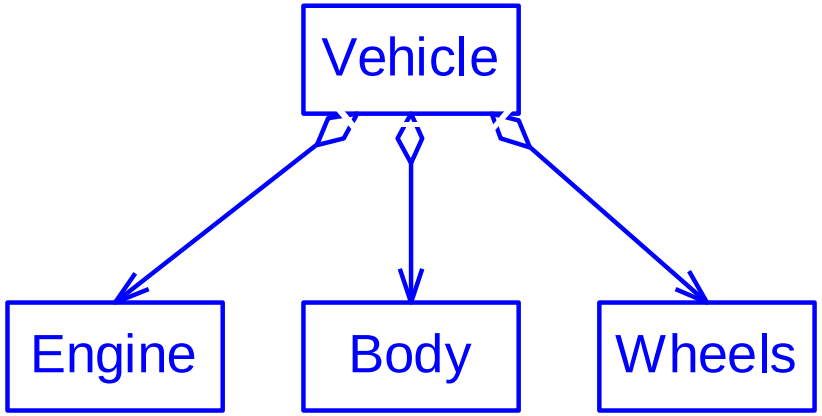
- سطوح تجرید متفاوت در سطوح مختلف سلسه مراتب نمایان می‌گردد.



سلسله مراتب PART-OF

- این سلسله مراتب اجزا تشکیل دهنده اشیا را نشان می‌دهد.
  - یک کلاس از یک یا چند کلاس دیگر تشکیل می‌شود.
- نام دیگر این نوع سلسله مراتب رابطه کل / جز (Whole/Part) یا رابطه تجمعی (Aggregation Relationship) است.

سلسله مراتب PART-OF



The Vehicle ***HAS-An*** Engine

The Engine is ***PART-OF*** Vehicle

سلسله مراتب و پیچیدگی

- با ساماندهی تجریدها در سلسله مراتب PART-OF و IS-A درک ما نسبت به سیستم افزایش می‌یابد.
- سلسله مراتب PART-OF روابط موجود بین اشیا و فعل و انفعالاتی که رخ می‌دهد را نمایان می‌سازد.
- سلسله مراتب IS-A افزونگی موجود در سیستم را مدیریت می‌کند (Economy of Expression)

محصور سازی و سلسله مراتب

استفاده از **وراثت** با **محصورسازی تام** تعارض دارد زیرا مستلزم دسترسی مستقیم کلاس فرزند به بعضی از اعمال و داده های اختصاصی کلاس پدر است.

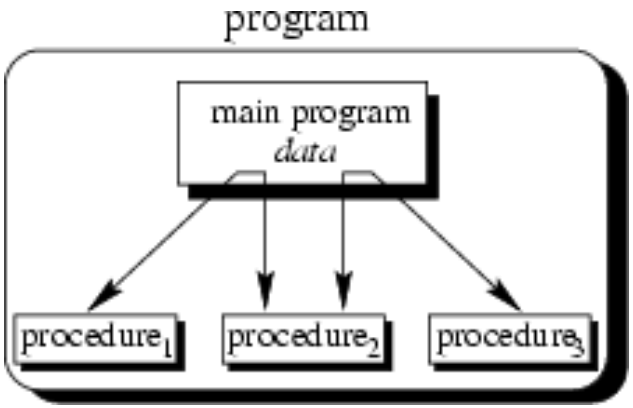
یک زبان برنامه‌سازی شی‌گرا

- بطور خلاصه می‌توان گفت که یک نرم‌افزار از مجموعه‌ای از داده‌ها و الگوریتم‌ها تشکیل شده است.
- یک زبان شی‌گرا باید
  - باید از اشیا به عنوان تجرید الگوریتم + تجرید داده حمایت کند.
  - هر شی دارای نوع باشد (از مفهوم کلاس پشتیبانی کند).
  - بتوان یک رابطه زیرنوعی را بین اشیا ایجاد کرد. (سلسله مراتب)

پارادایم‌های برنامه‌نویسی

- عمومی تر
  - **دستوری** (ساخت یافته – رویه‌ای)**:** این را انجام بده بعد آنرا
    - `Fortran, Algol, Pascal, Basic, C,…`
  - **شی‌گرا:** ارسال پیغام بین اشیا
    - `Simula, Smalltalk, C++, Java, C#,…`
- کمتر مورد استفاده
  - **تابعی:** ارزیابی یک الگو و ارسال آن
    - `Haskell, ML, Lisp, Scheme`
  - **منطقی:** بر پایه اظهارات منطقی و پاسخ به آنها
    - `Prolog`

پارادایم برنامه‌نویسی دستوری

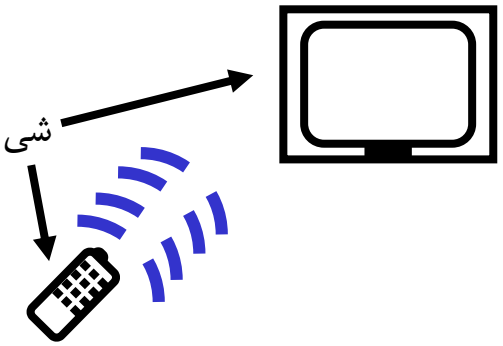


- قدیمی‌ترین و سرراست‌ترین روش برنامه‌نویسی
  - تمرکز بر عملکرد سیستم (قرار است چه کاری انجام شود)
- برنامه مجموعه‌ای از دستورالعمل‌ها است
- توابع تا حدودی اجازه واحدبندی، محصورسازی و استفاده مجدد کدها را فراهم می‌آورد.

پارادایم برنامه‌نویسی دستوری

- این خط کد را انجام بده.
- سپس این خط کد را.
- سپس این خط...

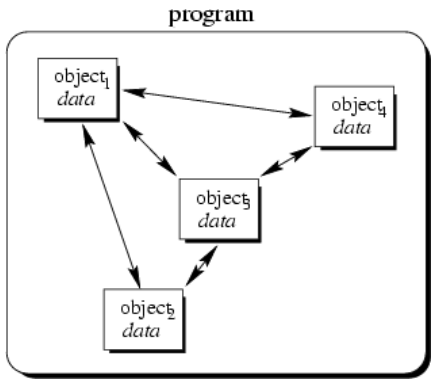
پارادایم برنامه‌نویسی شی‌گرا



- پارادایم رایج‌تر برنامه‌نویسی
- به جای تمرکز برآنچه سیستم باید انجام دهد،

تمرکز روی:

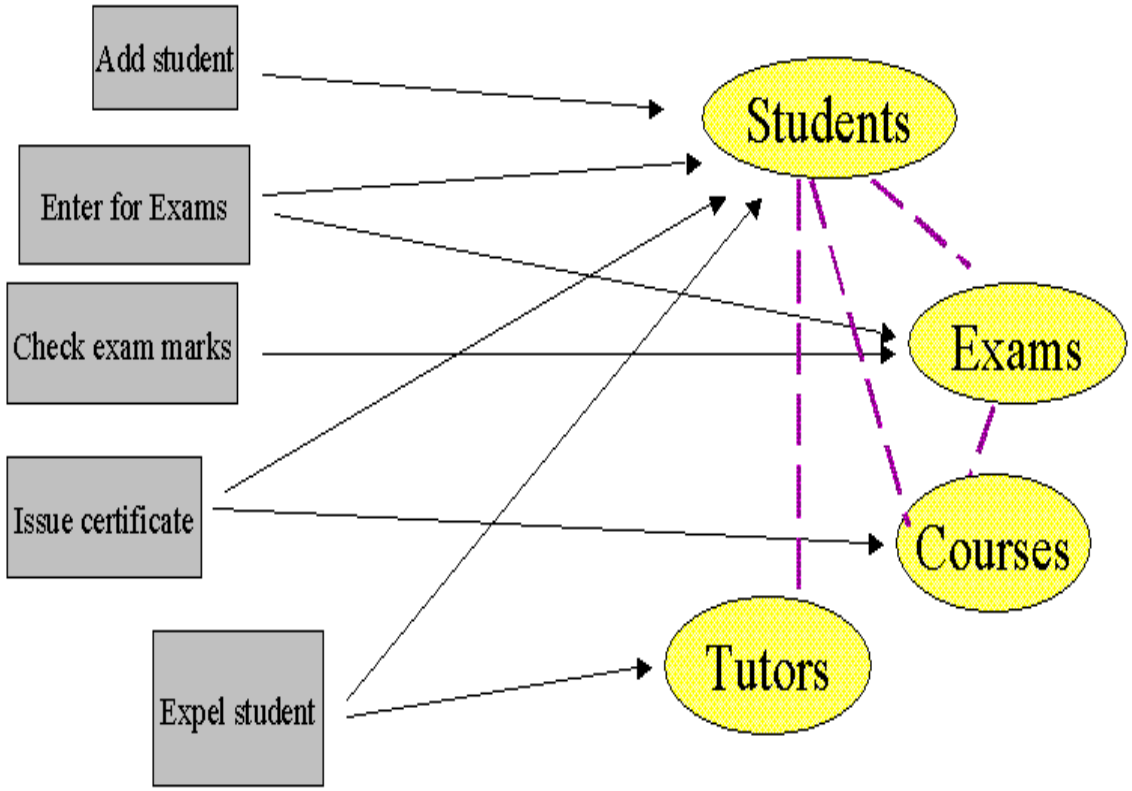
- سیستم شامل چه اشیایی است
- چگونه با هم تعامل دارند (پیغام)



پارادایم برنامه‌نویسی شی‌گرا

- کلید برق پیغامی را به الکتریسیته ارسال می‌کند (شی فعال – تاثیرگذاری روی سایر اشیا با ارسال پیام).
- الکتریسیته پیغامی را به لامپ می‌فرستد (شی عامل – تاثیرگذاری روی سایر اشیا و تاثیر پذیری از آنها).
- لامپ روشن می‌شود (شی غیر فعال – تاثیرپذیر از سایر اشیا)!
- برنامه‌نویسی دستوری به عنوان بخشی از برنامه‌نویسی شی‌گرا مورد نیاز است.

تفاوت رویکرد شی‌گرا و ساخت‌یافته



مقایسه فلسفه پارادایم‌ها

- ساخت‌یافته
  - درک مسئله
  - شکستن مسئله به چند زیر مسئله
  - نوشتن الگوریتم‌هایی برای حل زیر مسئله‌ها
  - نتایج: راه حل یکپارچه، غیر قابل استفاده مجدد
- شی‌گرا
  - درک کامل دامنه مسئله (Problem domain)
  - نوشتن یک سیستم که دامنه را مدل‌سازی می‌کند
  - نتایج: یک مجموعه از اجزا که می‌تواند در شکل‌های مختلف مورد استفاده قرار گیرد. (در این دامنه یا دامنه‌های دیگر)