

بسمه تعالی

فصل دوم

معرفی اصول شی گرائی برای مقابله با پیچیدگی نرم افزار

اهداف جلسه



- آشنائی با اصول شی گرائی
- درک نقش اصول شی گرائی در کنترل پیچیدگی
- سیستم‌های نرم‌افزاری
- درک مزایای مدل شی و کاربردهای آن

فهرست مطالب



- مقدمه
- روش‌های طراحی
- تجرید (Abstraction)
- محصورسازی (Encapsulation)
- واحدبندی (Modularity)
- سلسه مراتب (Hierarchy)
- مزایای مدل شی و کاربردهای آن

نرم افزار در روزهای اولیه

- هزینه اساسی طراحی مربوط به سخت افزار بود
- نقش نرم افزار، نقش ثانویه تلقی می شد (Afterthought)
- بیشتر نرم افزارها بوسیله یک نفر تولید و توسعه می شدند
- فرآیند طراحی غالباً در ذهن برنامه نویس انجام می شد
- زبان رایج: زبان ماشین سپس اسمبلی ابداع شد
- قابلیت سخت افزار بسیار محدود بود
 - برنامه ها کوچک و ساده بودند
 - مستندسازی چندانی مورد نیاز نبود

نرم افزار در عصر حاضر

- هزینه اساسی طراحی مربوط به نرم افزار است
- نقش نرم افزار، نقشی بسیار اساسی است
- بیشتر نرم افزارها بوسیله تیم های چند نفره تولید و توسعه می شوند
- فرآیند طراحی به صورت صریح و در خارج از ذهن برنامه نویس انجام می شود
- زبان های رایج: زبان های سطح بالا، ساخت یافته، و شی گرا

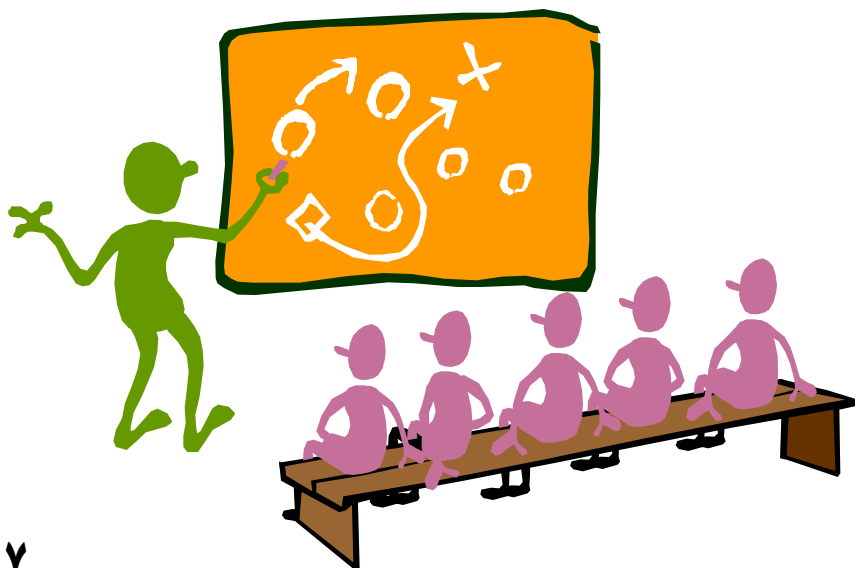


نرم افزار در عصر حاضر (ادامه)

- سخت افزارها سریعتر، ارزانتر و قابل اطمینان تر شدند
- اقتصادی شدن فرآیند خودکار سازی بسیاری از کاربردهای صنعتی و تجاری
- تقاضا برای نرم افزارهای پیچیده تر
- درک اهمیت مستندسازی سیستمها
- احساس نیاز به روشهای تحلیل و طراحی

روش‌های طراحی

- طراحی ساخت یافته (Structured Design)
- طراحی مبتنی بر داده‌ها (Data-Driven Design)
- طراحی شی‌گرائی (Object-Oriented Design)



اصول شی گزایی

شی گزایی

تجزید

محمصورسازی

وامد بندی

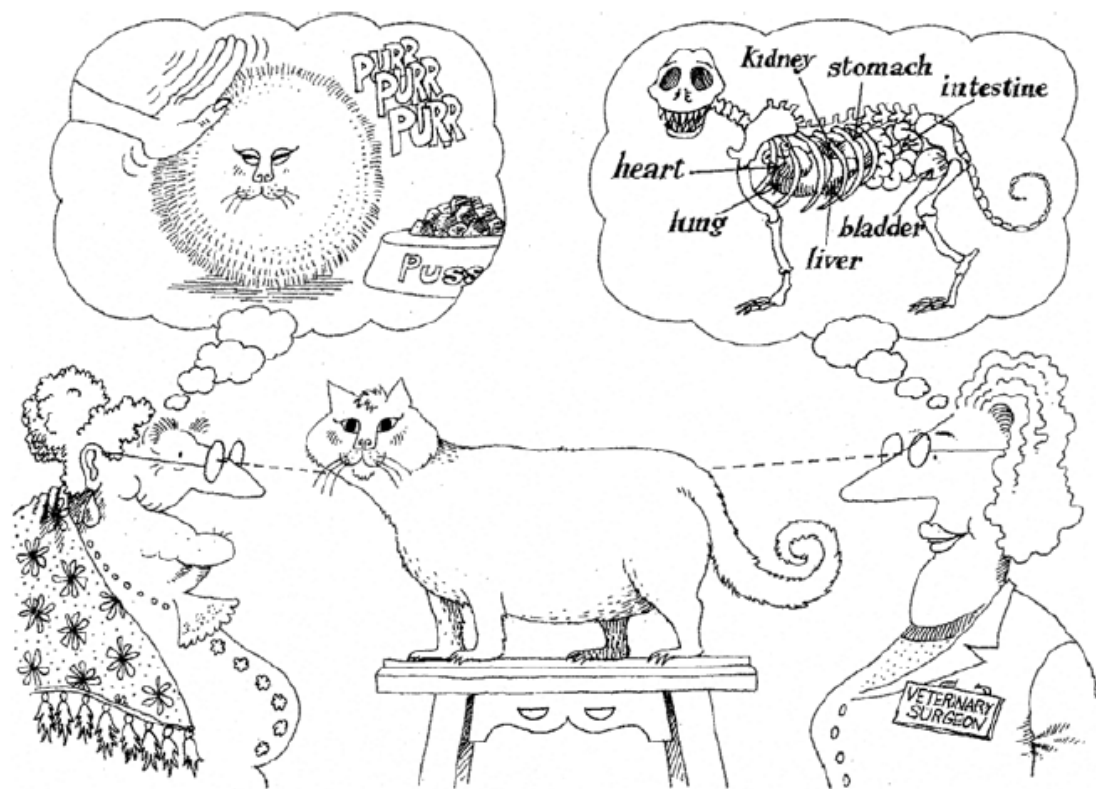
سلسله مراتب

تجرید (Abstraction)

- فرآیند **تمرکز** روی ویژگیها و رفتارهای اصلی یک پدیده و **نادیده گرفتن** ویژگیهای موقت و غیرمهم آن پدیده، از یک زاویه دید مشخص

تجريد (ادامه)

تمرکز روی ویژگیها و رفتارهای اصلی یک پدیده



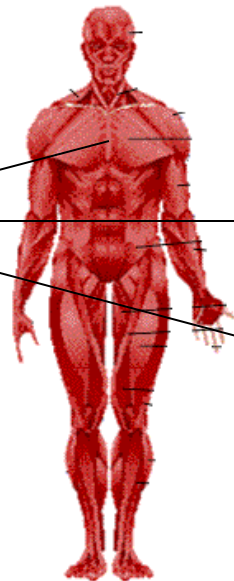
[Booch2007]

تجريد (ادامه)

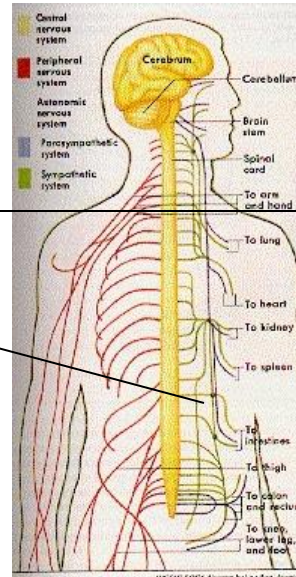
مثال‌هایی از تجريد



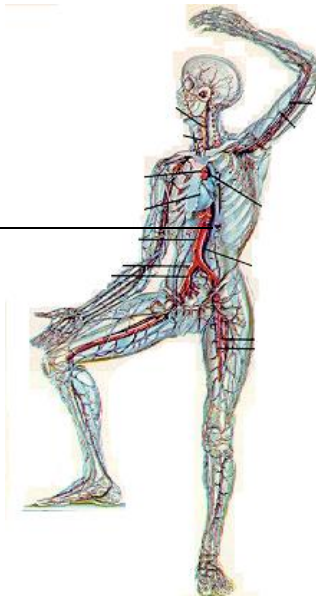
دیده‌های
گونه‌گون از
بدن انسان



عضلات بدن

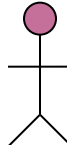


سیستم عصبی



سیستم گردش خون

تجريد (ادامه)

- ۲ بیان روابط میان اجزاء یک سیستم مکانیکی توسط یک معادله ریاضی
- ۳ استفاده از نماد  برای نمایش حضور موجودیت انسان در یک صحنه
- ۴ نمایش گرافیکی رفتار یک سیستم

نقش تجريد در کنترل پیچیدگی

■ یکی از ابزارهای اصلی کنترل و تسلط بر پیچیدگی

بوسیله تجريد

تنها ابعاد اساسی پدیده
مد نظر خواهد شد

جزئیات بی شماری که درباره یک پدیده مطرح است

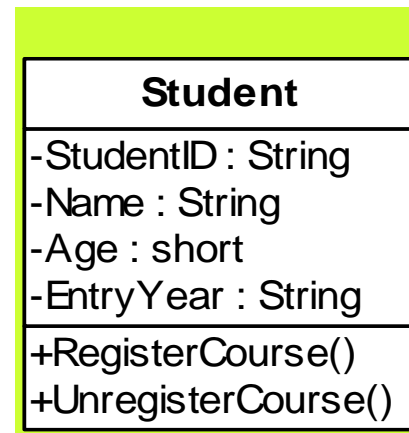
انواع تجرید

■ تجرید موجودیت

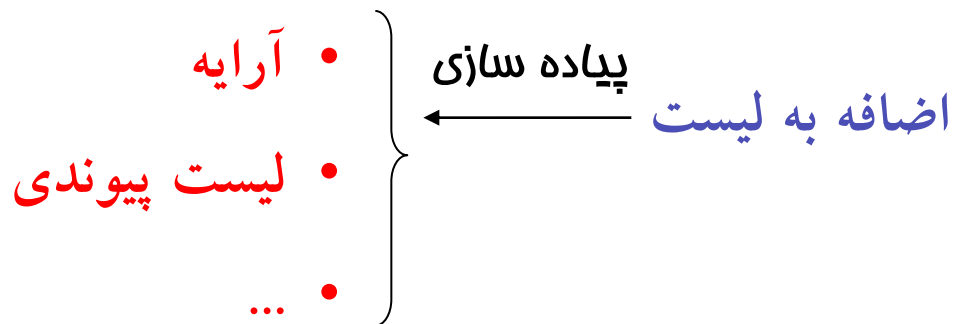
Real Object: Student



Abstraction: Student

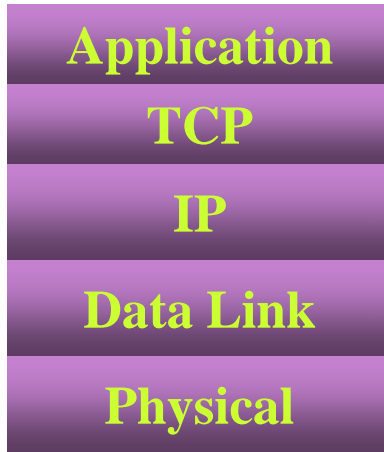


■ تجرید رفتار



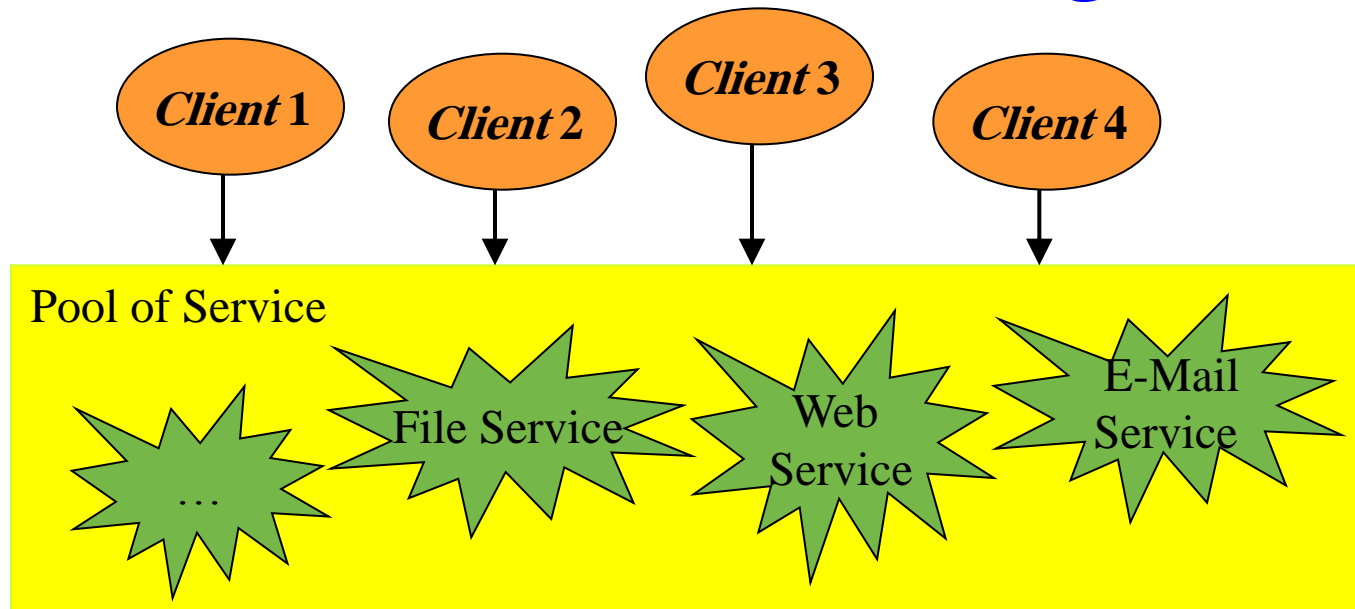
انواع تجرید (ادامه)

تجرید مجازی



پروتکل TCP/IP

تجرید تصادفی



ویژگی‌های تجرید

- برای یک شی تجریده‌های گوناگونی وجود دارد
- تجرید با نمود خارجی یک شی سر و کار دارد
- تجرید سطوحی دارد (میزان پرداختن به جزئیات)
- همه تجریدها دارای ویژگیهای ساکن و پویا هستند
- در شی‌گرایی مفهوم تجرید خود را در قالب نوع داده مجرد (Abstract Data Type) نشان می‌دهد

محصور سازی (Encapsulation)

■ عبارت از عدم پذیرش **تاثیرات ناخواسته** (Side Effects) و یا کنترل نشده و **محدود کردن** طرق دسترسی به / استفاده از یک شی

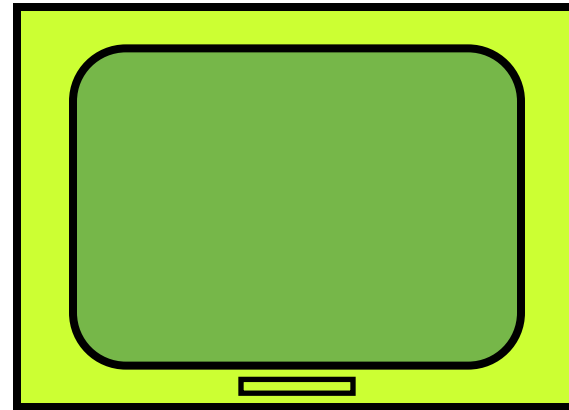
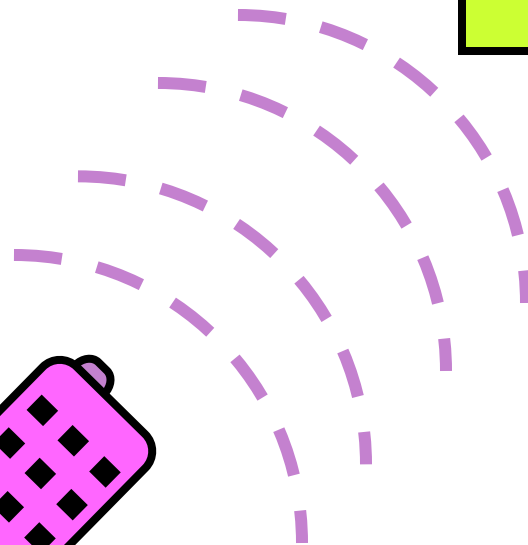
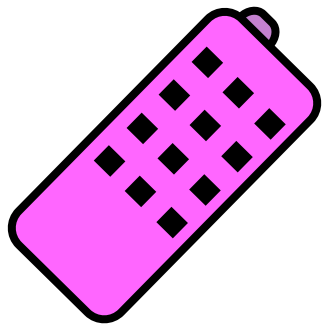
■ با توجه به این اصل، هر شی از دو مولفه زیر تشکیل می‌گردد

۱- واسط (Interface): توصیفی از سرویس‌هایی که این شی در اختیار Clientها قرار می‌دهد

۲- ساختار داخلی: داده‌ها + اعمال روی داده‌ها

محصول سازی (ادامه)

واسطه



شی

نقش محصور سازی در کنترل پیچیدگی

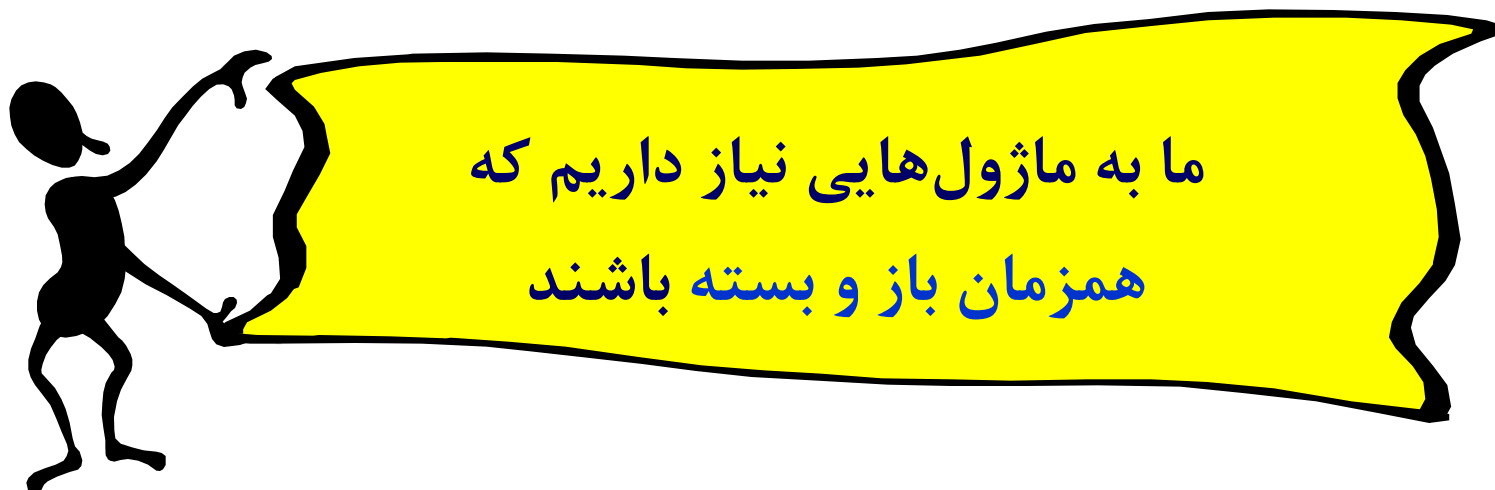
- کنترل و تسلط بر پیچیدگی بوسیله کنترل راه‌های دسترسی به یک شی که باعث جلوگیری از خرابکاری‌های احتمالی و محلی کردن گستره خطاها در خود شی
- ثبات واسط، امکان اعمال تغییر در پیاده‌سازی شی را فراهم می‌کند
- مفهوم یک واسط، چند پیاده‌سازی امکان استفاده مجدد را بالا می‌برد



محصولسازی – قاعده باز و بسته

■ قاعده باز و بسته

- هر ماژولی برای تعریف کننده آن باز و برای استفاده کننده بسته است
 - ماژول باز: ماژولی که برای اعمال تغییرات آماده باشد
 - ماژول بسته: ماژولی که امکان تغییرات در آن وجود ندارد



محصولسازی – ویژگیها

ارتباط بین اشیاء تنها از راه واسطها باشد

تجربید مکانیزم تعیین جزئیاتی است که باید پنهان شود، است،

اما محصولسازی، فرآیند پنهانسازی جزئیات و کنترل دسترسی به آن خواهد بود.

محصولسازی یک مفهوم نسبی است



واحدبندی (Modularity)

■ سیستم واحدبندی شده

■ به مجموعه‌ای از ماژول‌های (واحدها) منسجم و معنی‌دار که وابستگی بین آنها حداقل است تجزیه شده باشد

■ ماژول‌ها

■ مجموعه‌ای از عناصر با ارتباطات و وابستگی بالا

■ فایل‌ها در C و C++

■ مدارات مجتمع در سخت‌افزار

■ مولفه‌ها (Components) در استانداردهای COM، Java Beans و .NET

■ سرویس‌های وب در اینترنت (web Service)

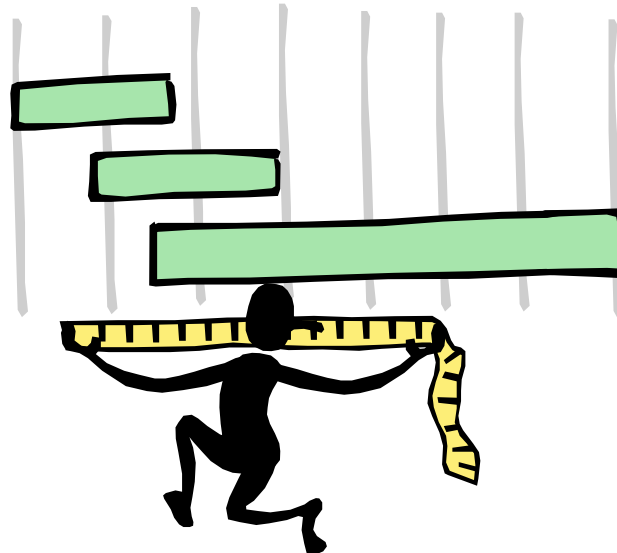
واحدبندی (ادامه)

■ انسجام (Cohesion)

■ درجه ارتباط عملکردهای عناصر داخلی یک ماژول

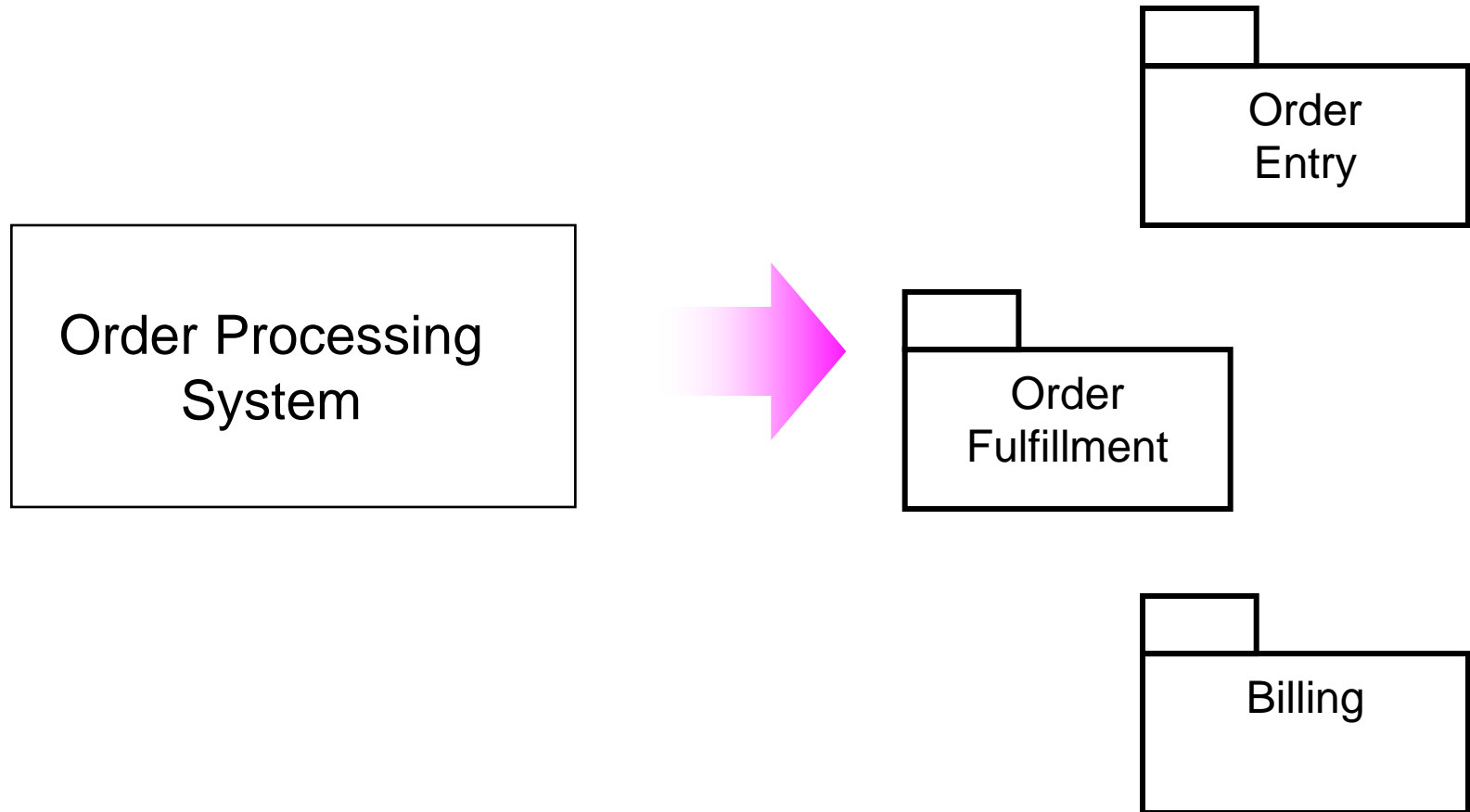
■ وابستگی (Coupling)

■ درجه ارتباط ماژول‌های گوناگون با یکدیگر

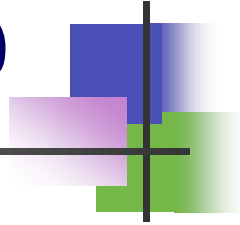


واحدبندی (ادامه)

مثال:



نقش واحدبندی در کنترل پیچیدگی



■ شکستن مساله به اجزائی کوچکتر یکی از راههای کارا برای مقابله با پیچیدگی

■ اگر مسئله P را به زیر مسئله‌های P1، P2 و P3 تقسیم نماییم آنگاه

$$C(P) > C(P1) + C(P2) + C(P3)$$

C: Complexity

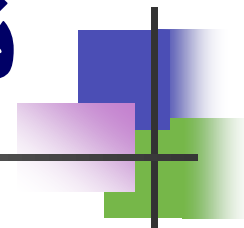
$$E(P) > E(P1) + E(P2) + E(P3)$$

E: Solving Energy

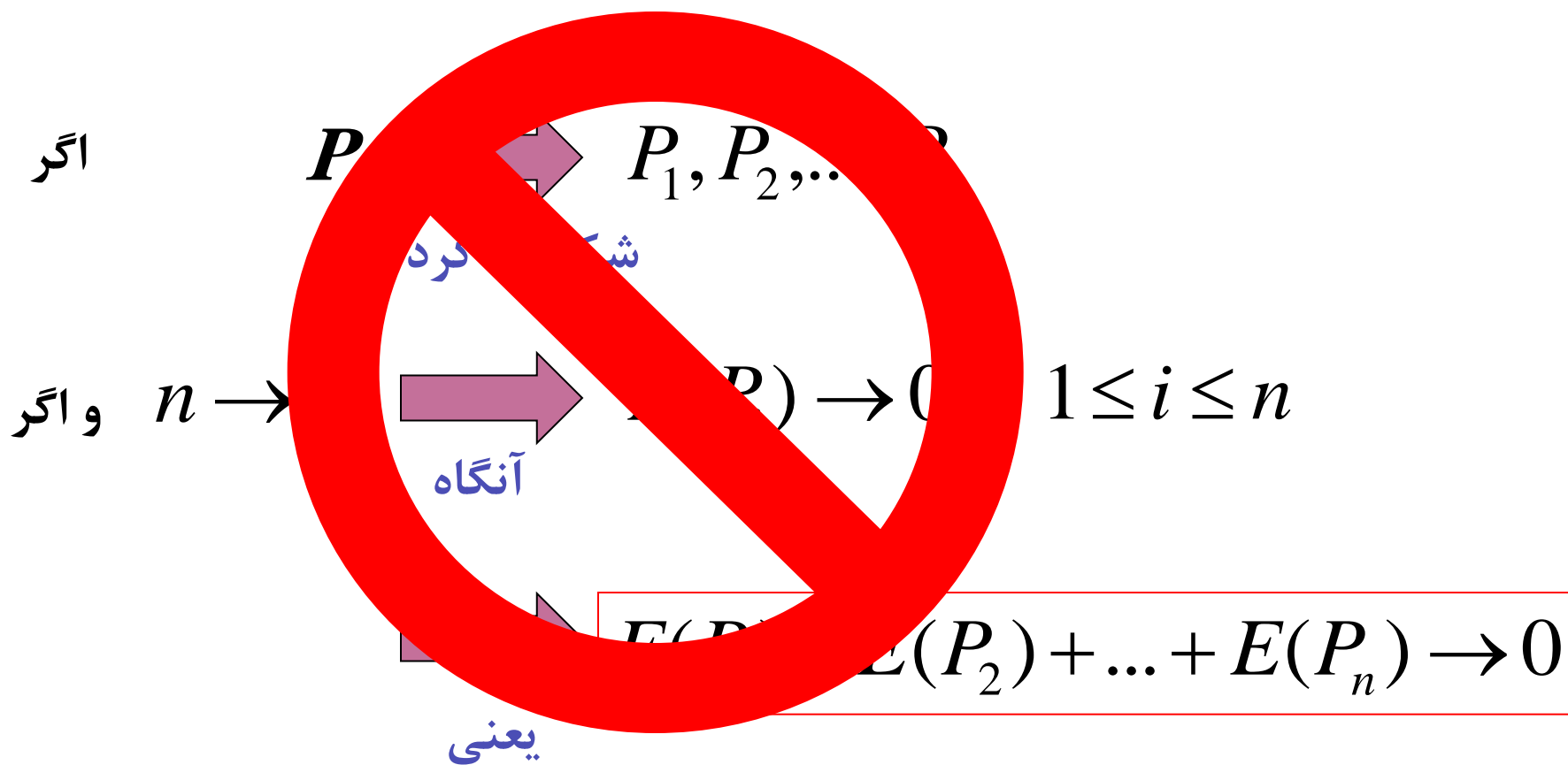
■ توجیه معادلات فوق

■ هنگام شکستن P وابستگی بین P1، P2 و P3 نادیده گرفته می‌شود

نقش واحدبندی در ... (ادامه)



بنابر استدلال قبل می توان نوشت:



نقش واحدبندی در ... (ادامه)

در روابط قبل برای سادگی، تلاش لازم برای یکپارچگی (*Integration*) راه حل ها با یکدیگر نادیده گرفته شده است، بنابراین باید نوشت:

$$E(P) > E(P_1) + E(P_2) + E(P_3) + I(E(P_1), E(P_2), E(P_3))$$

I: تلاش لازم برای یکپارچه سازی راه حل ها

توجه:

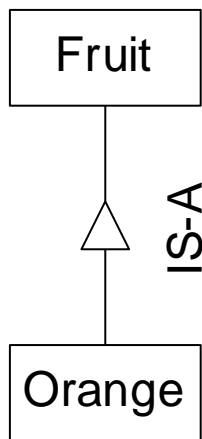
- هنگام شکستن *P*، روابط موجود بین زیر مسئله ها نادیده گرفته می شود
- دو مرحله شدن راه حل (شکستن سپس یکپارچگی)

واحدبندی – ویژگیها

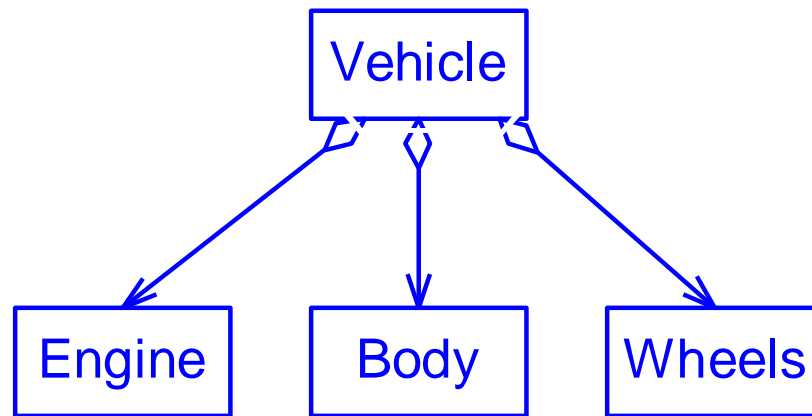
- اگر شرایط بیان شده در تعریف واحدبندی رعایت گردد آنگاه ماژول‌های بدست آمده **قابلیت استفاده مجدد بالایی** خواهند داشت
- تعداد زیرمساله‌ها نباید زیاد یا کم باشد
- تعیین **معیار شکستن یک مساله** مهمترین عامل برای موفقیت استفاده از این ویژگی است
- واحدها باید ویژگیهای ***Building Blocks*** را داشته باشند
 - استقلال (*Independent*)
 - واسط‌های خوش تعریف (*Well-defined Interfaces*)

سلسله مراتب (*Hierarchy*)

- عبارت از مرتب ساختن تجریدها در سطوح مختلف
- انواع سلسله مراتب
 - ساختار کلاس (IS-A)
 - ساختار شی (PART-OF)



Orange IS-A Fruit

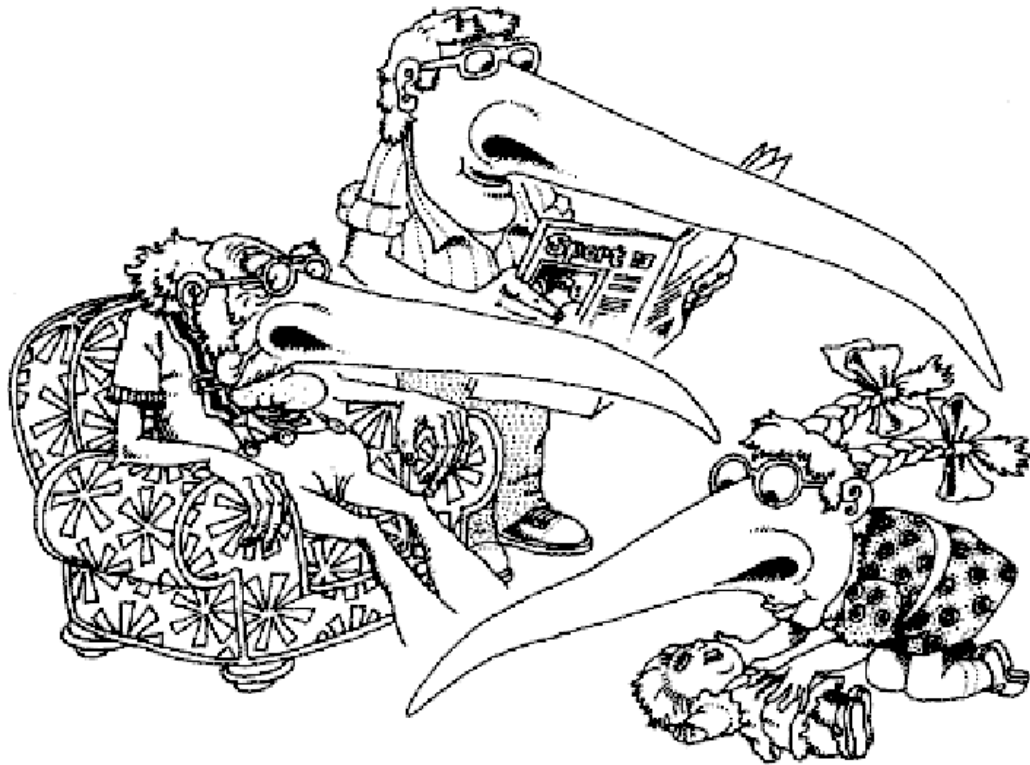


The Vehicle *HAS-AN* Engine

The Engine is *PART-OF* Vehicle

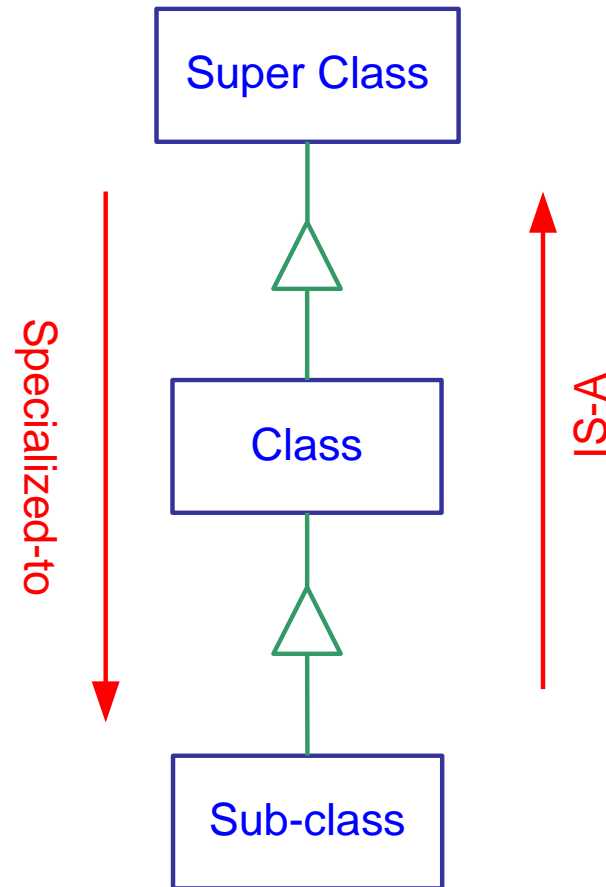
سلسله مراتب (ادامه)

وراثت (Inheritance) یکی از معروفترین انواع رابطه IS-A



[Booch2007]

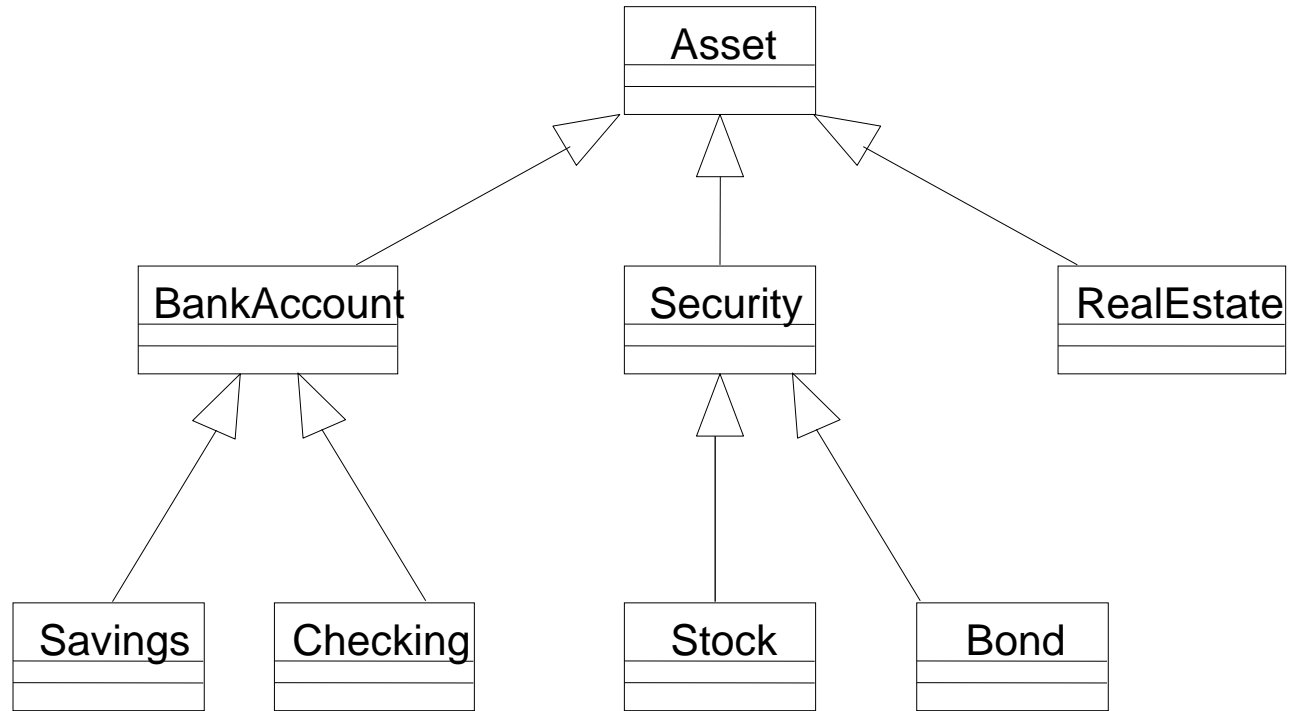
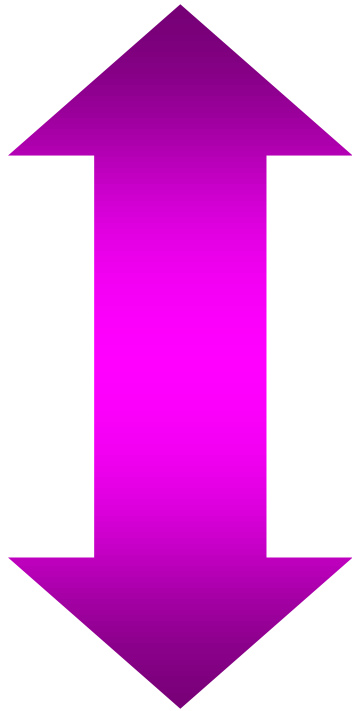
سلسله مراتب (ادامه)



سلسله مراتب (ادامه)

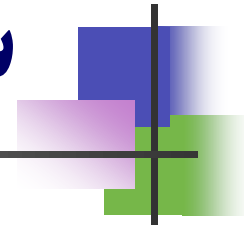
سطوح تجرید متفاوت در سطوح مختلف سلسله مراتب نمایان می گردد

افزایش تجرید



کاهش تجرید

سلسله مراتب (ادامه)



■ نقش سلسله مراتب در کنترل پیچیدگی

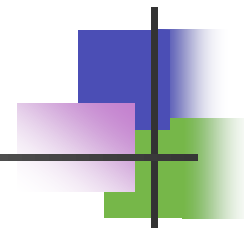
- با سازماندهی تجربیها در سلسله مراتب PART-OF و IS-A **درک ما نسبت به سیستم افزایش می یابد**
- **اهمیت سلسله مراتب PART-OF:** روابط موجود بین اشیاء و فعل و انفعالاتی که رخ می دهد را نمایان می سازد
- **اهمیت سلسله مراتب IS-A:** افزونگی موجود در سیستم را مدیریت می نماید (Economy of Expression)
- **استفاده از وراثت با محصورسازی تام** تعارض دارد زیرا مستلزم دسترسی مستقیم کلاس فرزند به بعضی از اعمال و داده های اختصاصی کلاس پدر است

مزایای مدل شی

- هدف نهائی تکنولوژی OO انجام فرآیند تولید نرم افزار به صورت مشابه فرآیند تولید سخت افزار (فرآیند استاندارد و سیستماتیک)
- قابلیت پشتیبانی از سیستم های توزیع شده (اشیاء یا مولفه ها روی سایت های گوناگون توزیع می شوند)
- ارائه مدل قویتری که پتانسیل مدیریت پیچیدگی کاربردهای امروزی را دارا باشد

مزایای مدل شی (ادامه)

- کاهش هزینه تولید و نگهداشت نرم افزار بوسیله در نظر گرفتن اشیاء بعنوان واحد مجتمع پذیر تفکیک نشدنی
- افزایش مقیاس پذیری و قابلیت توسعه سیستمها بوسیله محصورسازی
- استفاده مجدد بوسیله تکنولوژی مولفهها (COM، .NET، Java Beans) که بر مفاهیم مدل شی مبتنی است



پرسش و پاسخ