بسمه تعالی

فصل دوم

معرفی اصول شئ گرائی برای مقابله با پیچیدگی نرم افزار

مدرس: فر یدون شمس

اهداف جلسه

- آشنائی با اصول شئ گرائی
- درک نقش اصول شئ گرائی در کنترل پیچیدگی سیستمهای نرم افزاری
- درک مزایای مدل شئ و کاربردهای آن

فهرست

- مقدمه
- روشهای طراحی
- تجرید Abstraction
- دربرگیری Encapsulation
- واحدبندی Modularity
- سلسه مراتب Hierarchy
- مزایای مدل شئ و کاربردهای آن

مقدمه

- نقش نرم افزار در روزهای اولیه عصر کامپیوتر
  - نقش ثانویه تلقی می شد(Afterthought)
  - هزینه اساسی طراحی یک سیستم کامپیوتری از آن سخت افزار بود
  - بیشتر نرم افزارها بوسیله یک نفر تولید و توسعه می شدند

مقدمه (ادامه)

- نقش نرم افزار در روزهای اولیه عصر کامپیوتر (ادامه)
  - فرایند طراحی به صورت ضمنی در ذهن برنامه نویس انجام می شد
  - زبان رایج: زبان ماشین سپس اسمبلی ابداع شد
  - قابلیت سخت افزار بسیار محدود بود برنامها کوچک و ساده بودند
  - ظاهرا نیازی به مستند سازی نبود

مقدمه (ادامه)

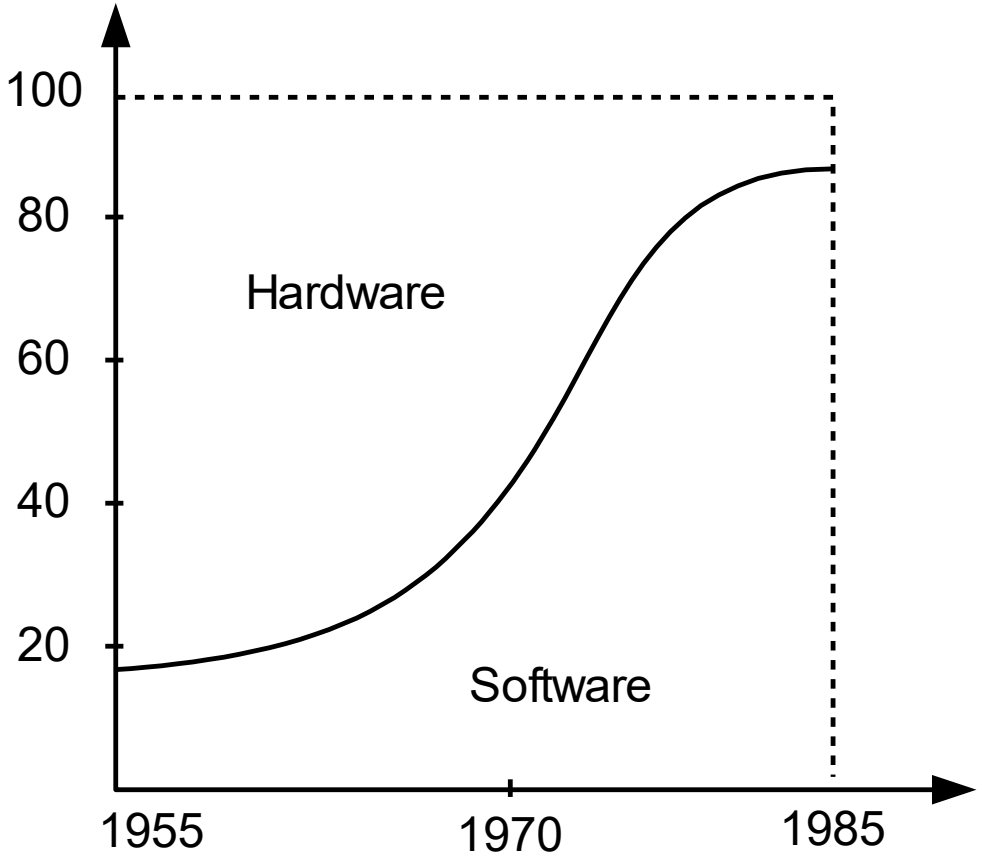
- نقش نرم افزار در روزهای کنونی
  - نقش بسیار اساسی
  - هزینه اساسی طراحی یک سیستم کامپیوتری از آن نرم افزار است
  - بیشتر نرم افزارها بوسیله تیمهای چند نفره تولید و توسعه می شوند
  - فرآیند طراحی به صورت صریح در خارج از ذهن برنامه نویس انجام می شود

مقدمه (ادامه)

- نقش نرم افزار در روزهای کنونی (ادامه)
  - سخت افزار سریعتر، ارزانتر، و قابل اطمینان تر
    اقتصادی شدن فرایند خودکار سازی بسیاری از کاربردهای صنعتی و تجاری
    تقاضا بر نرم افزارهای پیچیده تر

  - زبانهای رایج: زبانهای سطح بالا، ساخت یافته، و شئ گرا
  - درک اهمیت مستندسازی سیستمها
  - احساس نیاز به روشهای تحلیل و طراحی

مقدمه (ادامه)



درصد هزینه های نرم افزار در مقابل هزینه های سخت افزار از سالهای ۱۹۵۵ تا ۱۹۸۵

روشهای طراحی

- طراحی ساخت یافته Structured Design
- طراحی مبتنی بر داده ها Data-Driven Design
- طراحی شئ گرائی Object-Oriented Design

اصول شئ گرائی

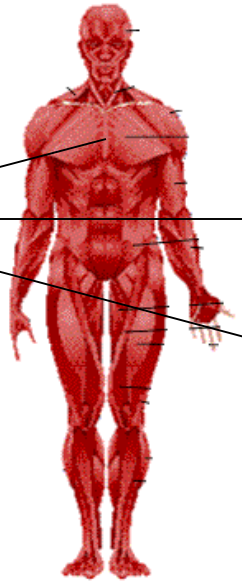
تجرید
(Abstraction)

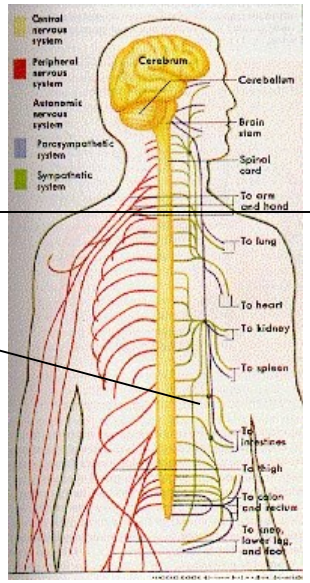
*"تجرید عبارتست از فرایند **متمرکز شدن** روی ویژگیها و رفتارهای اصلی یک پدیده، و **نادیده گرفتن** ویژگیهای موقت و غیر مهم آن پدیده، از یک زاویه دید مشخص"*

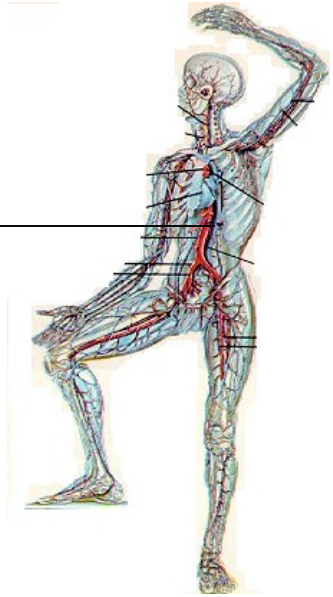
تجرید
(ادامه)

مثالهایی از تجرید

دیدهای گوناگون از بدن انسان



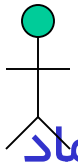




عضلات بدن سیستم عصبی سیستم گردش خون

تجرید
(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

٤. نمایش گرافیکی رفتار یکی سیستم

نقش تجرید در کنترل پیچیدگی:

- یکی از ابزارهای اصلی کنترل و تسلط بر پیچیدگی

بوسیله تجرید

جزئیات بی شماری که درباره یک پدیده مطرح است

تنها ابعاد اساسی پدیده مد نظر خواهد شد

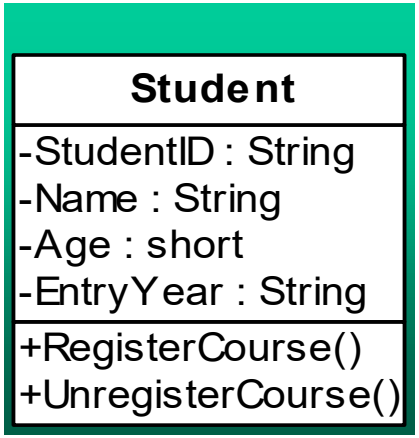
تجرید
(ادامه)

انواع تجرید:

- تجرید موجودیت(Entity Abstraction)

Real Object: Student

Abstraction: Student



تجرید
(ادامه)

- تجرید رفتار:

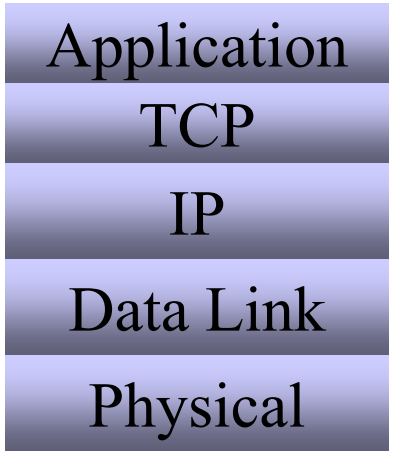
اضافه به لیست

پیاده سازی

- آرایه
- لیست پیوندی
- ...

- تجرید ماشین مجازی:

پروتکل TCP/IP



تجرید
(ادامه)

ویژگیهای تجرید:

- برای یک شئ تجریدهای گوناگونی وجود دارد
- تجرید با نمود خارجی یک شئ سر وکار دارد
- تجرید سطوحی دارد(میزان پرداختن به جزئیات)
- همه تجریدها دارای ویژگیهای ساکن و پویا هستند
- در شئ گرائی مفهوم تجرید خود را در قالب **نوع داده مجرد** (Abstract Data Type) نشان

محصور سازی (Encapsulation)

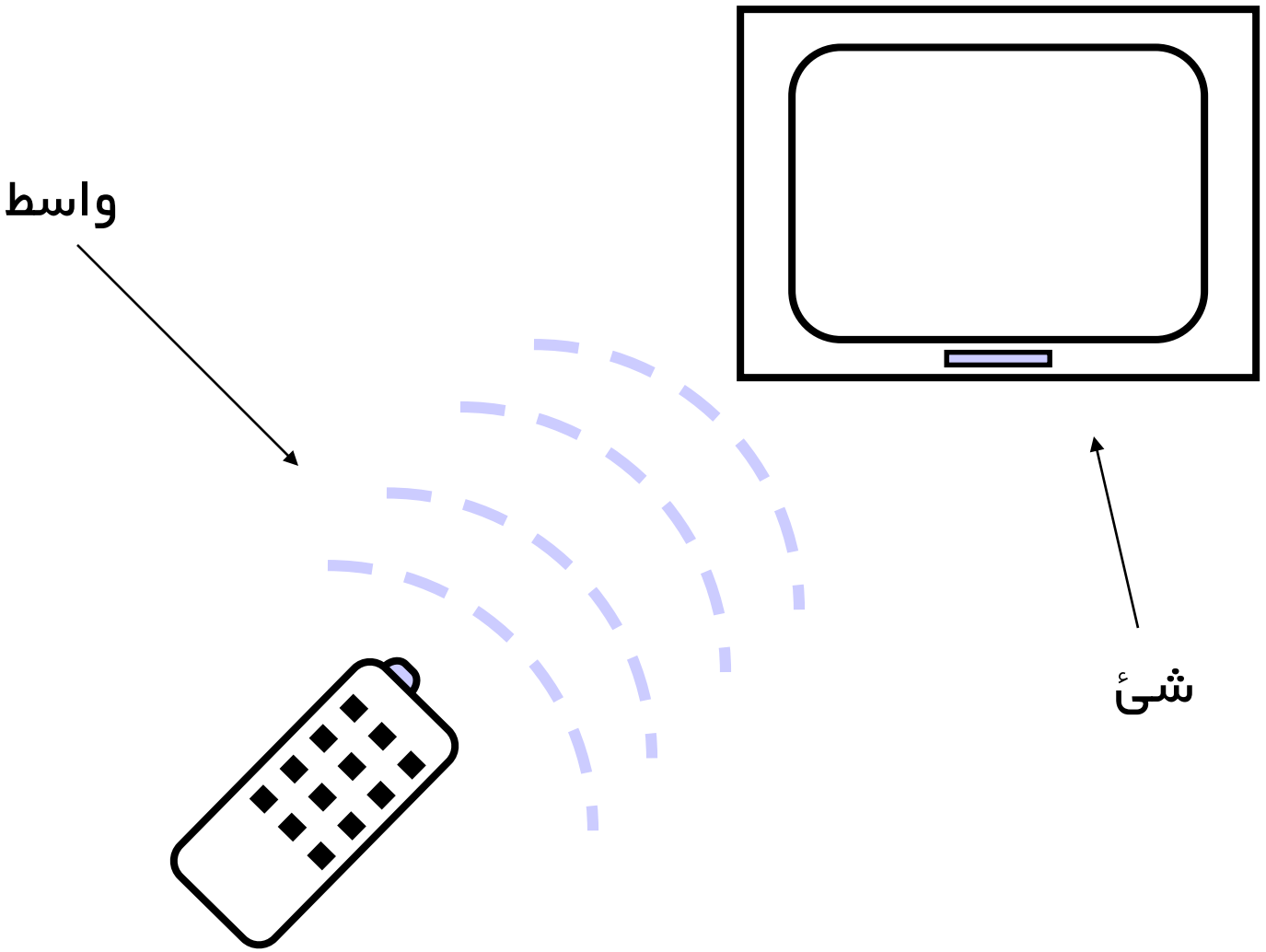
"محصور سازی عبارت از عدم پذیرش تاثیرات ناخواسته (Side Effects) و یا کنترل نشده و محدود کردن طرق دسترسی به یا استفاده از یک شئ"

با توجه به این اصل هر شئ از دو مولفه زیر تشکیل می گردد:

۱ – واسط(Interface): توصیفی از سرویسهایی که این شئ در اختیار Clientها قرار می دهد.

۲– ساختار داخلی: داده ها + اعمال روی داده ها

محصور سازی (ادامه)



محصور سازی (ادامه)

نقش محصور سازی در کنترل پیچیدگی:

- کنترل و تسلط بر پیچیدگی بوسیله کنترل راه های دسترسی به یک شئ که باعث جلوگیری از خرابکاریهای احتمالی و محلی کردن گستره خطاها در خود شئ
- با ثبات واسط یک شئ، می توان هر تغییری در پیاده سازی آن شئ انجام گیرد
- مفهوم یک واسط، چند پیاده سازی امکان استفاده مجدد را بالامی برد

محصور سازی (ادامه)

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

"ما به ماژولهایی نیاز داریم که همزمان باز و بسته باشند"

محصور سازی (ادامه)

ویژگیهای دربرگیری:

- ارتباط بین اشیاء تنها از راه واسطها باشد.
- تجرید مکانیزم تعیین جزئیاتی است که باید پنهان شود، است. اما محصورسازی، فرایند پنهان سازی جزئیات و کنترل دسترسی به آن خواهد بود.
- محصور سازی یک مفهوم نسبی.

واحد بندی (Modularity)

"سیستمی را واحد بندی شده می گویند که به مجموعه ای از ماژولهای(واحدها) منسجم و معنی دار که وابستگی بین آنها حد اقل است تجزیه شده باشد"

- ماژولها: واحد تشکیل دهنده ساختار فیزیکی سیستم نرم افزاری(شبیه مدارات مجتمع در سخت افزار)

مثالی از واحدها:

- فایلها در C و C++
- واحدها (Units) در Object Pascal
- مولفه ها(Components) در استانداردهای COM ، Java Beans و .NET

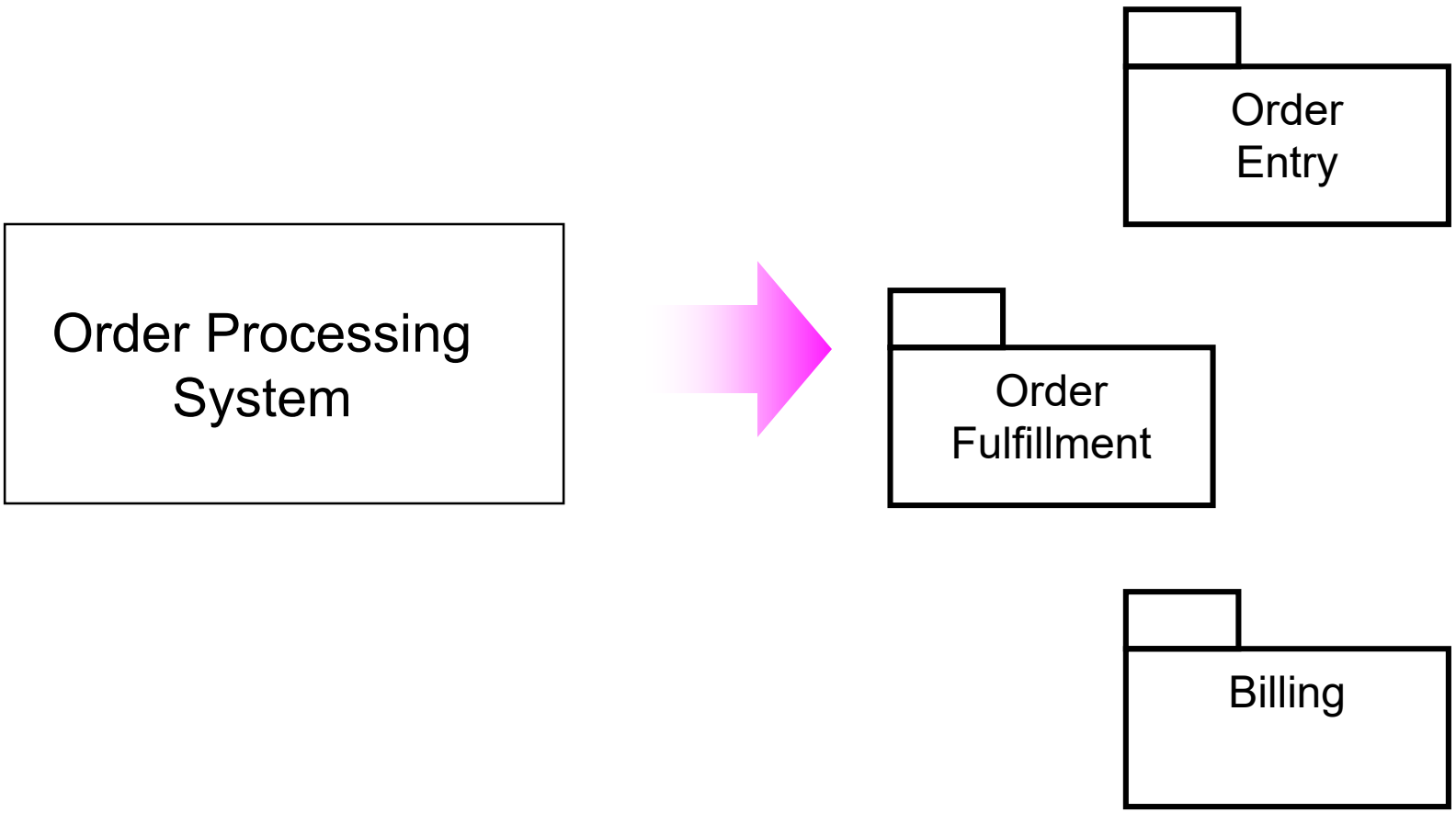
واحد بندی
(ادامه)

- انسجام(Cohesion): انسجام عبارتست از درجه ارتباط عملکردهای عناصر داخلی یک ماژول

- وابستگی(Coupling): وابستگی عبارتست از درجه ارتباط ماژولهای گوناگون با یکدیگر

واحد بندی
(ادامه)

مثال:



واحد بندی
(ادامه)

نقش واحد بندی در کنترل پیچیدگی:

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

مثال: اگر مسئله P را به زیر مسئله های P1، P2 و P3 تقسیم نماییم آنگاه

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

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

C: Complexity

E: Solving Energy

توجیه معادلات فوق: هنگام شکستن *P* وابستگی بین *P1*، *P2*، و *P3* نادیده گرفته می شود

واحد بندی
(ادامه)

بنابر استدلال قبل می توان نوشت:

اگر $P \Rightarrow P_1, P_2, \ldots, P_n$

شکسته گردد به

و اگر $n \to \infty \Rightarrow E(P_i) \to 0 \quad 1 \le i \le n$

آنگاه

$$\Rightarrow E(P_1) + E(P_2) + \ldots + E(P_n) \to 0$$

یعنی

واحد بندی
(ادامه)

در روابط قبل برای سادگی، تلاش لازم برای یکپارچگی(Integration) راه حلها 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 حلها

توجه:

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

واحد بندی
(ادامه)

ویژگیهای واحدبندی:

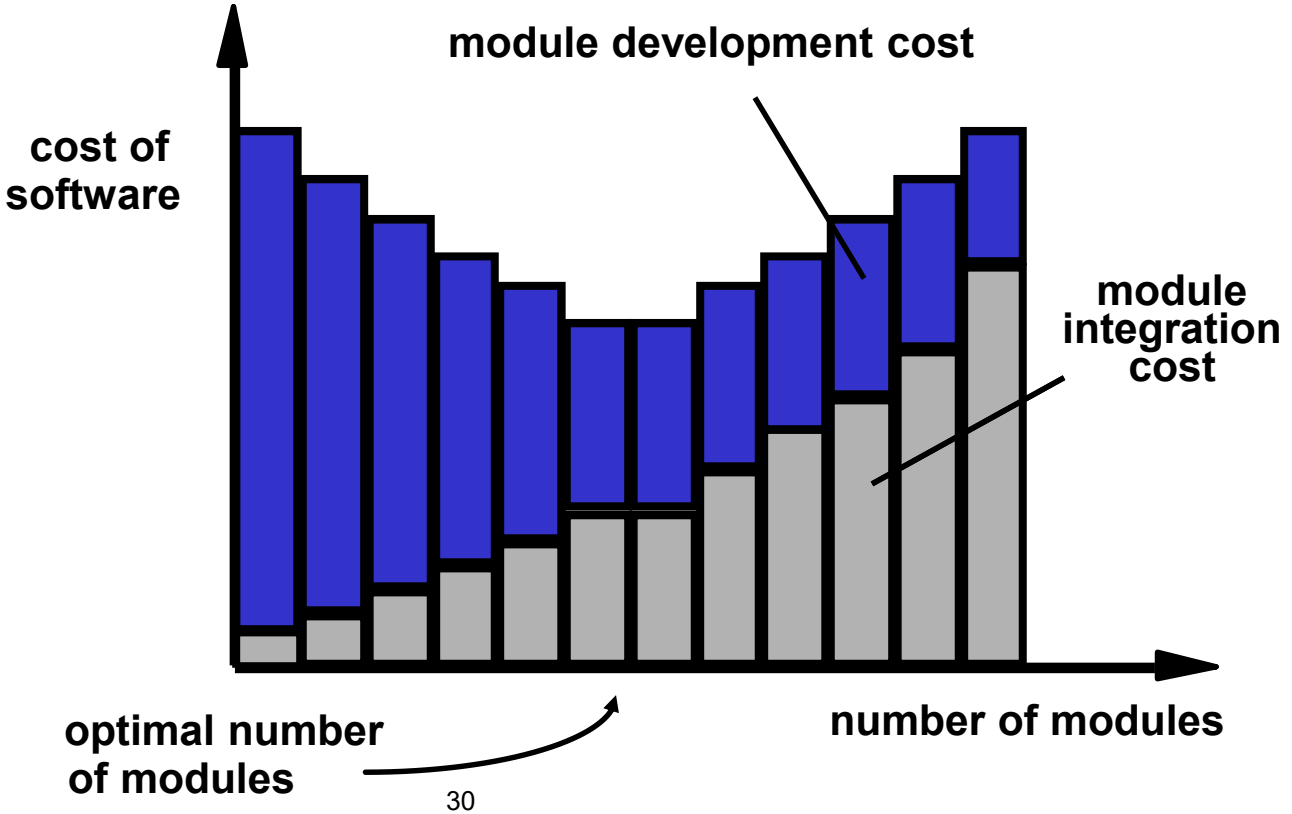
- اگر شرایط بیان شده در تعریف واحد بندی رعایت گردد آنگاه ماژولهای بدست آمده قابلیت استفاده مجدد بالایی خواهد داشت
- تعداد زیر مساله ها نباید زیاد یا کم باشد
- تعیین معیار شکستن یک مساله مهمترین عامل برای موفقیت استفاده از این ویژگی

واحد بندی
(ادامه)

- واحدهاباید ویژگیهای Building Blocks را داشته باشند:
  - استقلال(Independent)
  - واسطهای خوش تعریف(Well-defined Interfaces)

Modularity: Trade-offs

What is the "right" number of modules for a specific software design?



Cohesion
یکپارچگی

- Conventional view:
  - the “single-mindedness” of a module
- OO view:
  - *cohesion* implies that a component or class encapsulates only attributes and operations that are closely related to one another and to the class or component itself
- Levels of cohesion
  - Functional
  - Layer
  - Communicational
  - Sequential
  - Procedural
  - Temporal
  - utility

- دیدگاه سنتی
  - «وحدت رأی » مؤلفه
- دیدگاه شئ گرا
  - یکپارچگی بدان معناست که یک مؤلفه یا کلاس فقط صفات و عملیات هایی را در خود پنهان سازی می کند که رابطه ای تنگاتنگ با یکدیگر و با خود کلاس یا مؤلفه دارند .
- سطوحی از یکپارچگی
  - عملیاتی
  - لایه ای
  - ارتباطاتی
  - متوالی
  - رویه ای
  - زمانی
  - بهره مندی

Coupling
اتصال

- Conventional view:
  - The degree to which a component is connected to other components and to the external world
- OO view:
  - a qualitative measure of the degree to which classes are connected to one another
- Level of coupling
  - Content
  - Common
  - Control
  - Stamp
  - Data
  - Routine call
  - Type use
  - Inclusion or import
  - External

- دیدگاه سنتی
  - Coupling بیان کنندهٔ درجهٔ اتصال یک مؤلفه، به دیگر مؤلفه ها و جهان بیرون است .
- دیدگاه شئ گرا
  - Coupling یک میزان کیفی از درجهٔ اتصال کلاس ها به یکدیگر است .
- سطوح اتصال
  - اتصال محتوا
  - اتصال مشترک
  - اتصال کنترل
  - اتصال مهری
  - اتصال داده ای
  - اتصال فراخوانی روال ها
  - اتصال استفاده از نوع داده
  - اتصال واردات یا شمول
  - اتصال خارجی

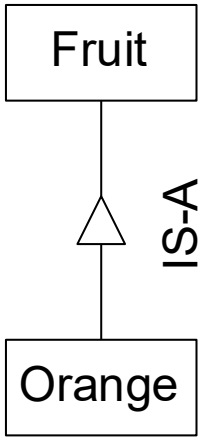
سلسله مراتب (Hierarchy)

"سلسله مراتب عبارت از مرتب ساختن تجریدها در سطوح مختلف"

انواع سلسله مراتب:

۱. ساختار کلاس(IS-A)

مثال: *Orange **IS-A** Fruit*



سلسله مراتب

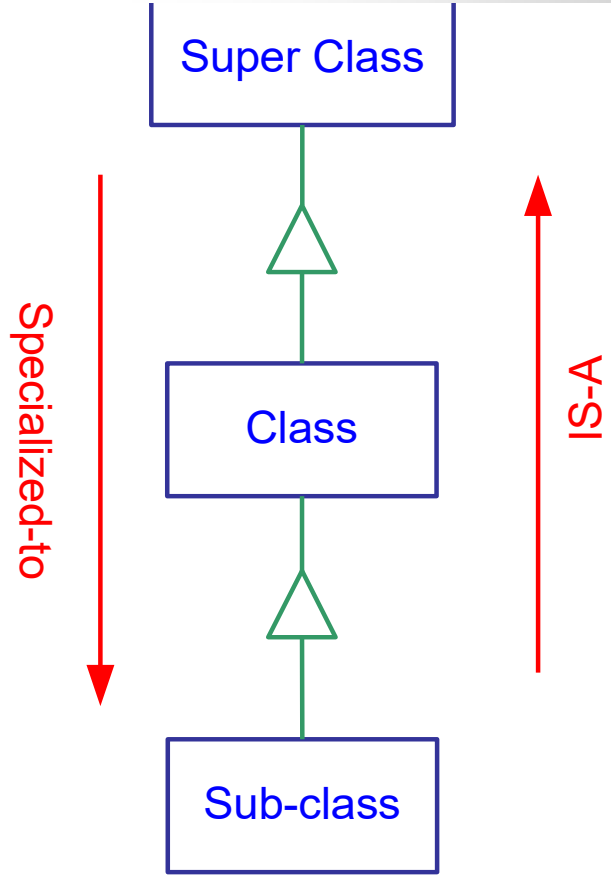
(ادامه)

"وراثت(Inheritance) یکی از معروفترین انواع

[Booch94]

سلسله مراتب

(ادامه)



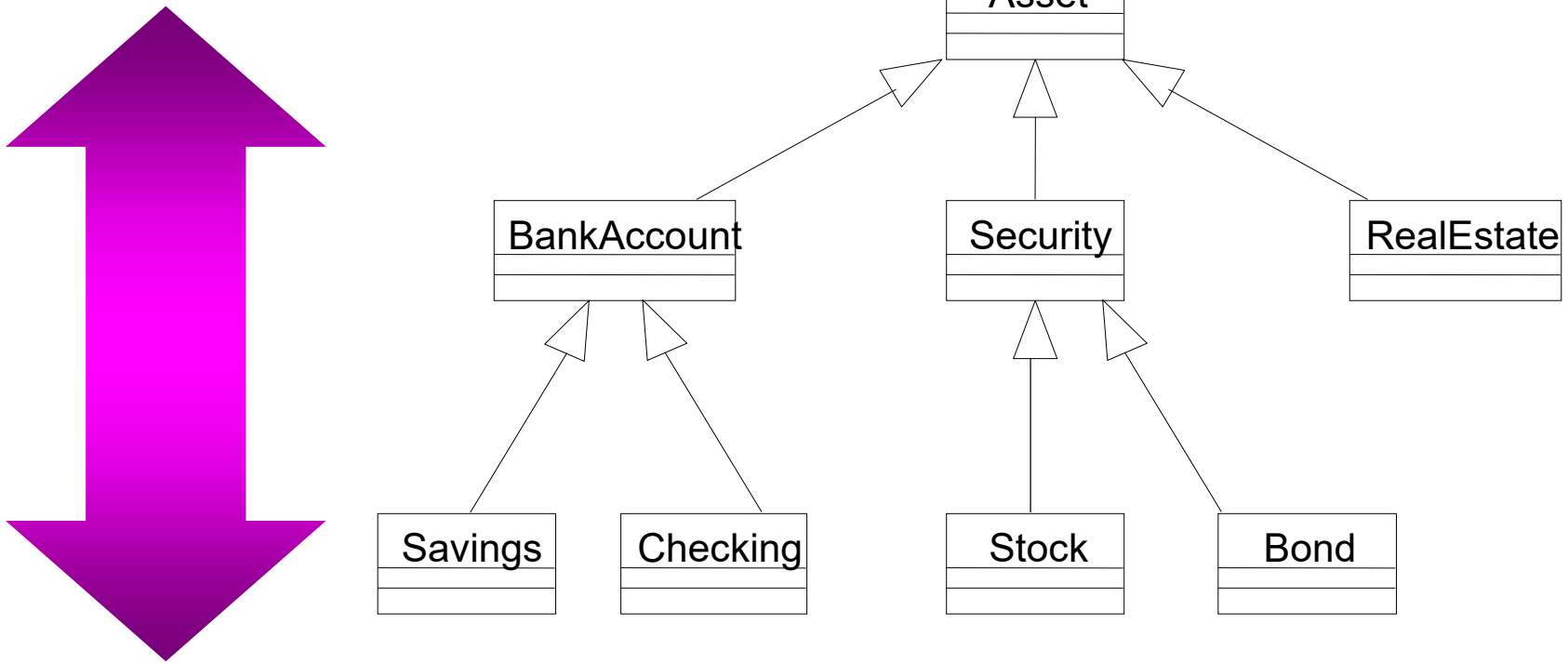
روابط وراثتی کلاس پدر، کلاس، و کلاس فرزند

سلسله مراتب

(ادامه)

"سطوح تجرید متفاوت در سطوح مختلف سلسه مراتب نمایان می گردد"

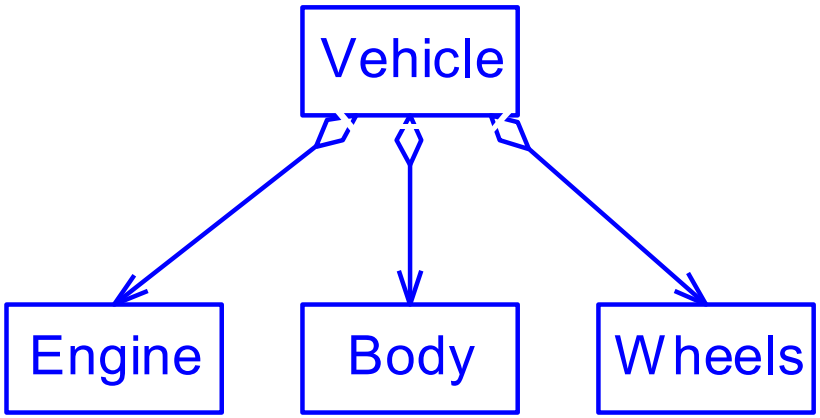
افزایش تجرید



کاهش تجرید

سلسله مراتب
(ادامه)

۲ ساختار شیء(PART-OF)



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

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

سلسله مراتب
(ادامه)

نقش سلسله مراتب در کنترل پیچیدگی:

- با سازماندهی تجریدها 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

سلسله مراتب
(ادامه)

ویژگیهای 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

مزایای مدل شیٔ

۱ هد جام فرایند تولید نرم افزار به صورت مشابه فرایند تولید سخت افزار(فرایند استاندارد و سیستماتیک)

۲ قابلیت پشتیبانی از سیستم های توزیع شده(اشیاء یا مولفه ها روی سایت های گوناگون توزیع می شوند)

۳ ارائه مدل قویتری که پتانسیل مدیریت پیچیدگی کاربردهای امروزی را دارا باشد

مزایای مدل شئ
(ادامه)

٤

٥ کاهش هزینه تولید و نگهداشت نرم افزار بوسیله در نظر گرفتن اشیاء بعنوان واحد مجتمع پذیر تفکیک نشدنی

٦ افزایش مقیاس پذیری و قابلیت توسعه سیستمها بوسیله محصورسازی

استفاده مجدد بوسیله تکنولوژی مولفه ها(COM، .NET، Java Beans) که بر مفاهیم مدل شئ مبتنی است