۴۰ ،۲ ،۲۱
chapter_08_Design Concept

فصل هشتم

■ **مفاهیم طراحی**

درس : طراحی نرم افزار پیشرفته

- مدرس : پرهام مرادی
http://eng.uok.ac.ir/moradi

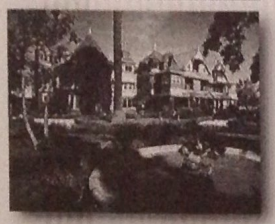
- تدریسیاران: سمانه حسین مردی – فاطمه پورباقی

نیمسال دوم ۹۴-۹۳
دانشکده ریاضی و علوم کامپیوتر
دانشگاه صنعتی امیرکبیر

جلسه هشتم

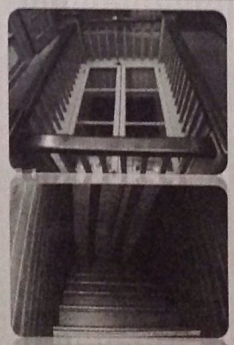
- طراحی
- فرآیند طراحی
- مفاهیم طراحی
- مدل طراحی

عمارت وینچستر



- این عمارت که در طی ۳۸ سال (۱۹۲۲ تا ۱۸۸۴) توسط بیوه ثروتمند آقای وینچستر رایفل ساخته شده است.
- در طراحی این ساختمان از هیچ معماری استفاده نشده است.
- خانم سارا وینچستر که ظاهراً یک آدم خرافاتی بوده، از طریق فالگیرها و پیش گوهائی که در اطرافش جمع شده بودند، به این اعتقاد عجیب رسیده بود که زندگی و مرگ وی بستگی به ادامه کارهای ساختمانی دارد که در خانه‌اش انجام می‌شود؛ به عبارت دیگر، وی تصور می‌کرد در صورت قطع شدن عملیات ساختمانی زندگی او نیز به پایان خواهد رسید.

عمارت وینچستر



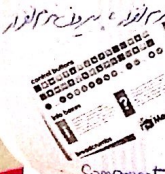
- ۳۸ سال از عمر خانم وینچستر و بسیاری از صنعت گران را به خود اختصاص داد
- این عمارت ۱۶۰ اتاقه برخلاف سایر خانه‌های هم عصر خود دارای سیستم‌های مدرن گرمائی (کانالهای بخار و هوای گرم) چراغهای گازی که با فشار یک دگمه روشن می‌شوند، سه عدد آسانسور قمال، و ۴۷ عدد بخاری است.
- جای جای خانه شامل عجایب زیادی چون درها و پنجره هائی که رو به دیوار باز می‌شوند، راه پله هائی که به هیچ جاختم نمی‌شوند، و تعداد بیشماری بخاری، راه پله، در و پنجره اضافی، انواع سیستمهای گرمائی و غیره‌است.

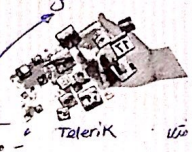
عمارت وینچستر

- عدم وجود راهبرد: دارای اتاقهای بیشمار – دارای سالنهای متعدد پذیرائی و رقص و غیره – بیشتر به یک هتل.
- عدم وجود نقشه فنی
- عدم توجه به نیازمندیهای واقعی: در انتخاب و ساخت بخشهای مختلف عمارت اعم از درها، پنجره‌ها، سقف، کف پوش و غیره به نیازمندیهای واقعی توجه نشده است
- زمان غیرمعقول: ۳۸ سال برای ساخت عمارت صرف شده‌است.
- هزینه غیرمعقول: ۵۵۵ میلیون دلار صرف ساخت این عمارت شده‌است.
- حضور نسلهای مختلفی از سبکها و سیستمها: در این عمارت شاهد انواع سیستمهای گرمائی(سه نوع بخاری معمولی، سیستم بخار، و سیستم هوای گرم) هستیم. علاوه بر آن عمارت فوق که تنها دارای دو طبقه‌است، دارای سه آسانسور مستقل است که راه اندازی آن بیشتر حالت تجملی و پرستیژ داشته‌است تا یک نیاز واقعی.
- غیرقابل توسعه: توسعه عمارت فوق بدون تخریب بسیاری از قسمت‌های اصلی آن مقدور نیست.
- بسیار پیچیده: تعدد راهروها، درها، پنجره‌ها، راه پله‌ها و اتاقها وضعیتی را بوجود آورد است که براحتی در آن گم میشوید. در واقع به همین دلیل این خانه از مدتها پیش به خانه ارواح جنگهای داخلی آمریکا معروف بوده‌است.

مراحل طراحی

- مرحله اول: طراحی معماری سیستم
- مرحله دوم: طراحی واسطهای کاربری
  - واسطها ارتباط نرم افزار را با کاربران ، سیستم های نرم افزاری دیگر و یا دستگاههای سخت افزاری را مشخص می کند
  - در طراحی واسطها ارتباط بین قطعات نرم افزاری نیز مشخص می شود.
- مرحله سوم: طراحی قطعات نرم افزاری
  - نرم افزار شامل مجموعه ای از قطعات نرم افزاری





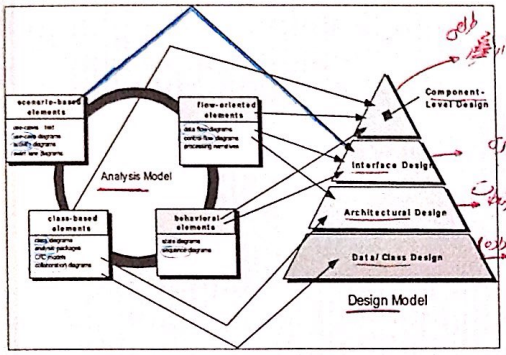
۱- طراحی

- یک بازنمایی یا یک مدلی از نرم افزار است که قرار است ساخته شود.
- طراحی، جزییات نرم افزار شامل ، معماری، ساختمان داده ها ، واسطهای کاربری و قطعات نرم افزاری را مشخص می کند
- در تحلیل (برخلاف طراحی) بیشتر تمرکز بر روی تشریح نیازمندیهای سیستم شامل رفتار سیستم (فعالیتها) و داده های سیستم است.
- امکان ارزیابی مدل طراحی (قبل از پیاده سازی و تست) وجود دارد.

ارزیابی طراحی

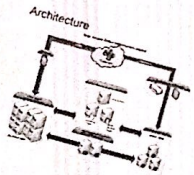
- بعد از اینکه مدل طراحی ایجاد شد باید ارزیابی شود
- ارزیابی طراحی توسط تیم نرم افزاری انجام می شود
- هدف از ارزیابی مشخص نمودن خطاها، ناسازگاریها، نقصها و یا راه حل های بهتر در مدل طراحی شده است.
- همچنین در ارزیابی مشخص می شود آیا طرح ارائه شده با محدودیتها، هزینه ها و زمانبندی مشخص شده قابل اجرا است یا خیر؟

ارتباط مدل تحلیل با مدل طراحی



ارتباط مدل تحلیل با مدل طراحی

- طراحی داده / کلاس (Data/Class Design)
  - کلاسهای مدل تحلیل به ساختمان داده های مورد نیاز سیستم و همچنین به کلاسهای واقعی تبدیل می شود
  - از ارتباطات بین اشیا و کلاسها (دیاگرام CRD) به عنوان پایه ای برای طراحی داده استفاده می شود.
  - جزییات بیشتر کلاسهای سیستم در بخش طراحی قطعات نرم افزاری ایجاد می شود.

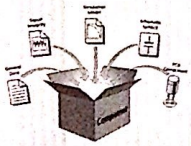


- طراحی معماری (Architectural Design)
  - ارتباط بین اجزاء سیستم مشخص می شود
  - سبک معماری (Style) برای سیستم انتخاب می شود.
  - محدودیتهایی که ممکن است بر پیاده سازی معماری مورد نظر تاثیر بگذارد مشخص می شود.

ارتباط مدل تحلیل با مدل طراحی- ادامه

- طراحی واسط ها(Interface Design)
  - ارتباط سیستم با افراد و دیگر سیستم ها و اجراء سخت افزاری مشخص می شود لذا برای طراحی واسطها، نیاز به سناریوها و مدل های رفتاری سیستم از بخش تحلیل وجود دارد.

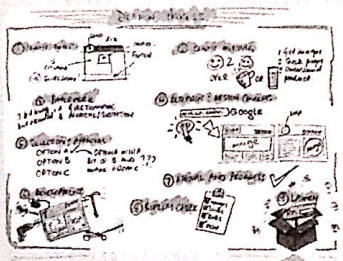
- طراحی قطعات نرم افزاری(Component Level Design)
  - بخشهای مختلف سیستم به قطعات نرم افزاری تبدیل می شود لذا نیاز به مدل کلاسهای سیستم ، مدل رفتاری و مدل جریان داده برای طراحی قطعات نرم افزاری وجود دارد.



۲- فرآیند طراحی

- یک فرآیند مرحله ای است که در آن نیازمندیهای سیستم که در مرحله تحلیل ایجاد شد به یک مدل طراحی تبدیل شده تا قابل پیاده سازی باشد (یک بلو پرینت از سیستم ایجاد می شود)

- در مراحل اولیه طراحی اجزای سیستم به صورت کلی (تجرید بالا) مشخص می شود و در مراحل بعدی جزییات فنی بیشتری از اجزای سیستم ارائه می شود.



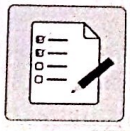
۲- فرآیند طراحی

کیفیت طراحی

- کیفیت طرح ایجاد شده در مراحل مختلف طراحی، توسط یک سری بازبینی فنی (Technical Review) مورد ارزیابی قرار می گیرد.

(جزییات بیشتر در مورد بازبینی فنی در فصل ۱۵ ارائه شده است)

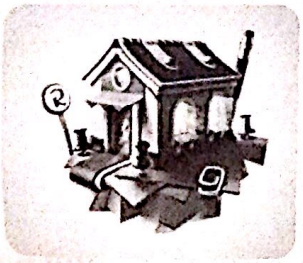
- سه ویژگی اصلی یک طراحی خوش تعریف شامل موارد زیراست:
  - طرح ارائه شده باید تمام نیازمندیهای مشخص شده در مرحله تحلیل را پیاده سازی کند
  - طرح ارائه شده باید برای برنامه نویسان ،افرادی که وظیفه تست سیستم را برعهده دارند و همچنین کسانی که وظیف پشتیبانی سیستم را برعهده دارند قابل فهم باشد
  - طرح ارائه شده باید یک دید کامل از نرم افزاری که قرار است ایجاد شود را ارائه بدهد.



۲- فرآیند طراحی

راهنمای کیفیت طراحی

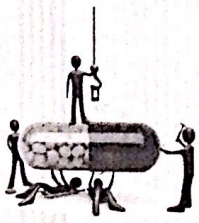
- ادامه: معیارهای طراحی
  - طراحی باید منجر به تولید واسطهایی با کمترین پیچیدگی شود – پیچیدگی ارتباطی بین قطعات نرم افزاری و همچنین پیچیدگی ارتباطی نرم افزار با دنیای خارج باید کم باشد
  - روش طراحی باید مرحله ای و تکراری باشد و بر اساس اطلاعات به دست آمده از مرحله تحلیل نیازمندیها باشد
  - طراحی باید به نحوی بازنمایی و یا نوشته شود که قابل فهم باشد و معنی آن برای افرادی که از آن استفاده می کنند (عموما برنامه نویسان و تست کنندگان) مبهم نباشد



فرآیند طراحی

راهنمای کیفیت طراحی

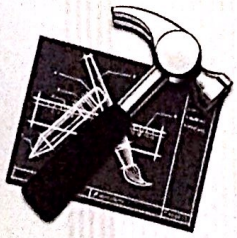
- برای ارزیابی مدل طراحی توسط اعضای تیم نرم افزاری، موارد ارائه شده در زیر می تواند به عنوان یک راهنما و معیار مورد استفاده قرار گیرد:
  - طراحی باید شامل یک معماری باشد که(۱) بر اساس الگوها و سبکهای شناخته شده معماری ایجاد شده باشد(۲) بر اساس روش تکاملی و مرحله ای ایجاد شده باشد – یعنی در هر مرحله جزییات بیشتری در معماری ارائه شود(۳) شامل مجموعه ای از قطعات نرم افزاری با طراحی خوب باشد
  - طراحی باید ماجولار باشد و سیستم به صورت منطقی به زیر سیستم ها تقسیم شود
  - طراحی باید شامل یک بازنمایی جدا از داده معماری ، واسطها و قطعات نرم افزاری باشد
  - طراحی باید منجر به تولید ساختمان داده های مناسب شود که کلاسهای آن بر اساس الگوهای شناخته شده طراحی شده باشد
  - طراحی باید منجر به تولید قطعات نرم افزاری شود که هر یک وظیفه خاصی را انجام دهند.



فرآیند طراحی

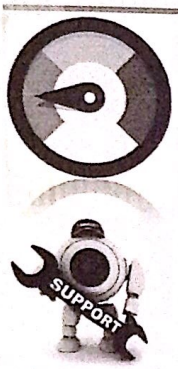
ویژگی های یک طراحی خوب

- بر اساس دیدگاه شرکت HP ویژگیهای یک طراحی خوب شامل موارد زیر است (به اختصار این ویژگیها FURPS نامیده می شود) :
- Functionality
  - این معیار بر اساس ویژگیهای نرم افزار و قابلیتهای آن قابل ارزیابی است. همچنین امنیت کل سیستم در ارزیابی این ویژگی در نظر گرفته می شود.
- Usability
  - براساس فاکتورهای انسانی قابل اندازه گیری است. سازگاری برنامه و مستندات نوشته شده برای آن میتواند در ارزیابی این ویژگی استفاده شود.
- Reliability
  - بر اساس تعداد خرابیهای نرم افزار و زمان رفع خرابی ارزیابی میشود. میانگین زمان رفع خرابی (MTTF)، تعداد خرابی و قابلیت رفع خرابی در ارزیابی این ویژگی استفاده می شود.



۲- فرآیند طراحی

ویژگی های یک طراحی خوب - ادامه



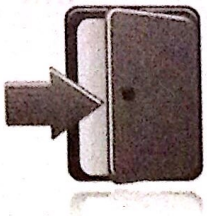
- **Performance**
  - این معیار بر اساس سرعت اجرای برنامه، زمان پاسخگویی به درخواستهای کاربر و میزان حافظه مصرفی قابل محاسبه و ارزیابی است.
- **Supportability**
  - بر اساس موارد زیر قابل ارزیابی است
    - Extensibility – قابلیت توسعه برنامه
    - Adaptability
    - Serviceability
    - Maintainability
    - Testability
    - Configurability

۳- مفاهیم مهم در طراحی

- Abstraction (تجرید، انتزاع) : در سطح داده ها، روال ها و کنترل ها
- Architecture (معماری): ساختار کلی نرم افزار
- Patterns (الگوها) : استفاده از راه حل های اثبات شده
- Separation of concerns(جداسازی مفاهیم): هر مساله پیچیده ای اگر به زیرمسائلش شکسته شود می تواند آسانتر حل شود.
- Modularity (ماژولاریتی): قسمت بندی داده ها و توابع
- Hiding (پنهان سازی): واسط های کنترل شده
- Independent Functionality(استقلال عملکرد): توابعی که مستقل هستند و وابستگی بسیار کمی دارند.
- Refinement (پالایش): ورود به جزئیات موارد انتزاعی مختلف
- Aspects : مکانیزمی است برای درک چگونگی تاثیر نیازمندیهای اصلی بر طراحی نرم افزار
- Refactoring : تکنیکی برای سازماندهی مجدد که طراحی را آسانتر می نماید
- OO design concepts (مفاهیم طراحی شی گرا): (به ضمیمه II کتاب مراجعه فرمایید)
- Design Classes(کلاس های طراحی): این کلاس ها، جزئیات موردنیاز جهت پیاده سازی کلاس های تحلیل را فراهم می آورند

۳- مفاهیم مهم در طراحی

Abstraction (تجرید، انتزاع)

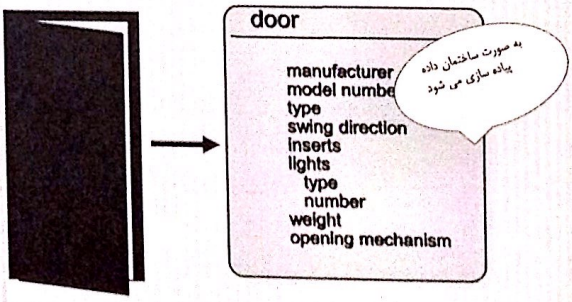


- تجرید :خلاصه گویی
- تجرید شامل دو نوع تجرید داده و تجرید رویه است.
- تجرید داده(Data Abstraction) :
  - استفاده از یک کلمه برای بازنمایی داده های مختلف
  - مثلا با شنیدن کلمه درب یا door بازنمایی کننده نوع در، جهت باز شدن، مکانیزم باز شدن و ... در ذهن بازنمایی می شود
- تجرید رویه (Procedural Abstraction)
  - استفاده از یک کلمه به جای مجموعه ای از اعمال
  - مثلا کلمه باز کردن در = منظور رفتن به سمت در = دستگیره را گرفتن = کشیدن در و ... می شود

۳- مفاهیم مهم در طراحی – ادامه

Abstraction (تجرید، انتزاع)

تجرید داده

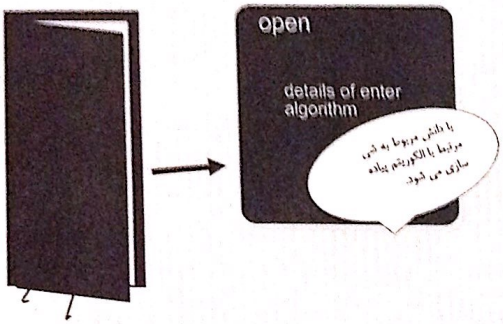


Important

۳- مفاهیم مهم در طراحی - ادامه

(تجرید/انتزاع) Abstraction

تجرید رویه



Important

۳- مفاهیم مهم در طراحی - ادامه

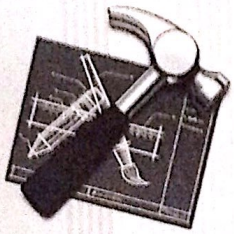
(معماری) Architecture

اجزای نرم افزار و نحوه ارتباط بین این اجزا را مشخص می کند
یک معماری باید شامل سه خصوصیت زیر باشد :

Structural properties
معماری باید اجزا سیستم را که شامل اشیا و ماجولها هستند را مشخص کند. همچنین باید نحوه ارتباط این ماجولها با هم مشخص شود

Extra-functional properties.
معماری باید نیازمندیهای سیستم شامل کارایی، قابلیت اعتماد امنیت، قابلیت تطبیق و خصوصیات دیگر را برآورده کند.

Families of related systems.
معماری باید بر اساس الگوهایی طراحی شود که این الگو ها در نرم افزارهای مشابه به کار رفته باشند.

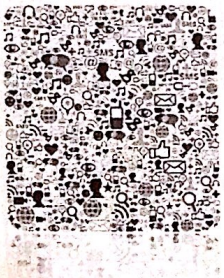


برای توصیف معماری یک سیستم می توان از زبان مخصوص به این کار Architectural Description Language (ADL) استفاده کرد.

Important

۳- مفاهیم مهم در طراحی - ادامه

(الگوها) Patterns



الگو یا Pattern
یک نام برای یک راه حل اثبات شده برای یک مشکل یا یک مساله نرم افزاری در یک موضوع مشخص است

مشخص کننده یک طرح یا یک راه حل (ساختار یک راه حل) برای یک مساله طراحی در یک حوزه مشخص است.

Important

۳- مفاهیم مهم در طراحی - ادامه

(الگوها) Patterns

یک قالب برای طراحی الگوها شامل موارد زیر است:

Pattern name
یک نام مختصر و مفید و غیر تکراری باید برای الگو در نظر گرفته شود

Intent
نحوه کار الگو را در این قسمت باید شرح داد

Also-known-as
نام های مشابه دیگری که برای الگو می توان در نظر گرفت در این بخش نوشته می شود. یا ممکن است این الگو با نامهای دیگری نیز شناخته شود.

Motivation
یک مثال از نحوه کاربرد الگو در این قسمت ارئه شود

Applicability
شرایطی را که الگو قابلیت استفاده را دارد در این بخش مشخص می شود. مثلا اینکه این الگو برای چه نوع نرم افزارهایی مناسب است و یا با در چه شرایطی باید استفاده شود

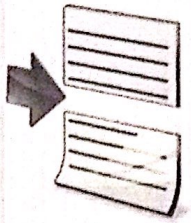
Structure
ساختار کلاسهای لازم برای پیاده سازی الگو را مشخص میکند

Consequences
نتیجه اجرای الگو برای مولد مشابه باید در این قسمت ذکر شود

Related patterns – الگوهای با مشخصات رفتاری مشابه یا این الگو مشخص می شود

Important

۳- مفاهیم مهم در طراحی - ادامه
(جداسازی مفاهیم) Separation of Concerns

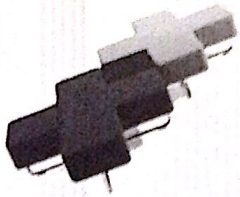


- یک مساله پیچیده وقتی به مسائل کوچکتر شکسته شود به راحتی قابل حل است.
- یک Concern ، یک ویژگی یا یک رفتار از سیستم است که بخشی از نیازمندیهای سیستم را برآورده می کند
- با شکستن و جدا سازی Concern ها به موارد کوچکتر، تلاش و زمان کمتری برای حل آن لازم خواهد بود.
- اگر یک مساله پیچیده p به دو مساله ساده تر p1 و p2 شکسته شود جمع میزان تلاش برای حل p1 و p2 کمتر از تلاش و زمان برای حل مساله p خواهد بود.

Important

۳- مفاهیم مهم در طراحی - ادامه
(ماژولاریتی) Modularity

- ماژولاریتی یک ویژگی نرم افزار است که باعث مدیریت بهتر تولید نرم افزار خواهد شد

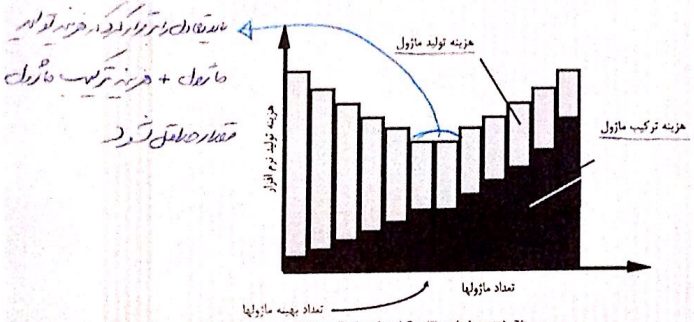


- Monolithic software
  - نرم افزاری که شامل یک ماژول بزرگ است
  - این نرم افزار شامل متغیرها و پیچیدگی زیادی خواهد بود و تقریبا فهم این نرم افزار برای برنامه نویسان غیر ممکن خواهد بود
- در بسیاری از موارد شکستن برنامه به تعدادی ماژول باعث فهم بهتر و در نتیجه هزینه تولید پایین تر خواهد شد.

Important

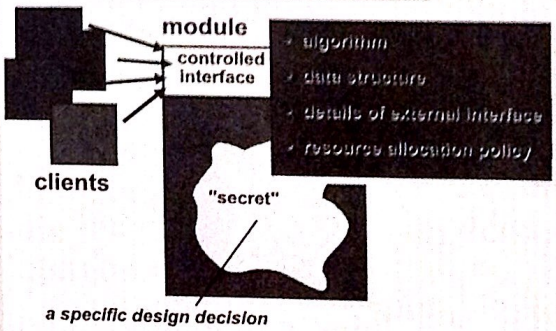
۳- مفاهیم مهم در طراحی - ادامه
(ماژولاریتی) Modularity
Trade-off

برای یک نرم افزار تعداد بهینه ماجولها باید چند مورد باشد؟



Important

۳- مفاهیم مهم در طراحی - ادامه
(پنهان سازی) Hiding



a specific design decision

Capsulation

دلایل نیاز به پنهان سازی اطلاعات؟

- کاهش اثرات جانبی
- کاهش تاثیر طراحی داخلی ماژولها
- تاکید بر ارتباطات بین ماژولها از طریق واسطها
- پرهیز از استفاده از داده عمومی
- استفاده از کپسوله سازی در طراحی ماژولها که یک خصوصیت و یا یک ویژگی طراحی خوب و با کیفیت است
- منجر به تولید نرم افزار با کیفیت خواهد شد.

با اندکی دقت در می یابیم :
پنهان سازی اطلاعات در روابط
انسانها نیز تاثیرات بسیار خوبی
خواهد داشت
یعنی پذیریم قرار نیست که ما در
جریان همه موارد اطرافمان قرار
داشته باشیم.
(نتیجه : کاهش اثرات جانبی!!!)

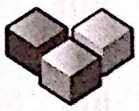
تصویر انجلینا جولی - خودش قبلا نقته

Important

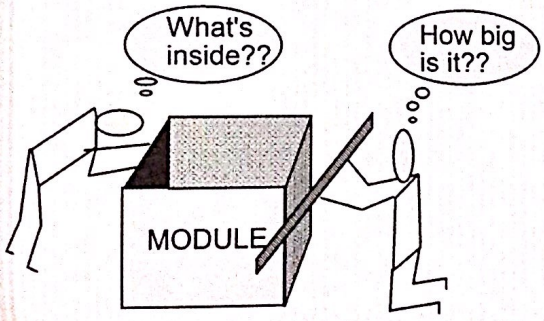
۳- مفاهیم مهم در طراحی – ادامه

Refinement

```
open
    walk to door;
    reach for knob;
    open door;
        repeat until door opens
        turn knob clockwise;
        if knob doesn't turn, then
            take key out;
            find correct key;
            insert in lock;
        endif
        pull/push door
        move out of way;
        end repeat
    walk through;
    close door.
```

اندازه گیری ماژول ها (۲ دیدگاه مختلف)

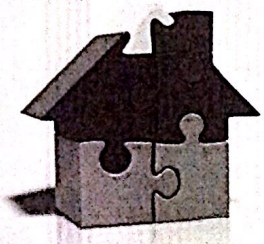


Important

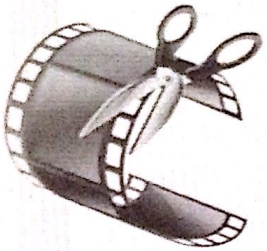
۳- مفاهیم مهم در طراحی – ادامه

Functional Independence (استقلال عملکرد):

- استقلال کارکردی با طراحی ماژولهایی که هر کدام یک وظیف مشخص انجام میدهند به دست می آید و تا جایی که امکان دارد باید ارتباط با سایر ماژولها کم شود و ارتباطات داخلی ماژول بیشتر شود
- *Cohesion*
  - میزان انسجام داخلی یک ماژول را نشان میدهد. ماژولی که انسجام بیشتری دارد بر روی یک وظیفه مشخص تمرکز دارد و ارتباطات کمتری با سایر ماژولها خواهد داشت. در بهترین حالت بهتر است یک ماژول تنها یک کار مشخص را انجام دهد.
- *Coupling*
  - میزان ارتباط یک ماژول با سایر ماژولها را نشان میدهد هر چه این ارتباط کمتر باشد استقلال ماژول بیشتر خواهد بود.
  - ارتباطات بین ماژولها بستگی به واسطهای ارتباطی و نوع داده ای دارد که بین آنها رد و بدل میشود.
  - در طراحی ماژولها بهتر است که انسجام داخلی (High Cohesion) بیشتر باشد و میزان ارتباطات آنها کمتر باشد (Low Coupling)



۳- مفاهیم مهم در طراحی – ادامه
Aspects



- دو نیازمندی A و B را درنظر بگیرید.
- نیازمندی A نیازمندی B را crosscut می کند اگر بتوان نرم افزار را به گونه ای تجزیه نمود که بدون وقوع A، نیازمندی B نتواند تکمیل شود.
- یک Aspect نمایشی است از مفهوم crosscut

۳- مفاهیم مهم در طراحی – ادامه
Aspects(مثال)

- Consider two requirements for the **SafeHomeAssured.com** WebApp.
- Requirement *A* is described via the use-case **Access camera surveillance via the Internet.** A design refinement would focus on those modules that would enable a registered user to access video from cameras placed throughout a space.
- Requirement *B* is a generic security requirement that states that *a registered user must be validated prior to using* **SafeHomeAssured.com**. This requirement is applicable for all functions that are available to registered *SafeHome* users.
- As design refinement occurs, *A** is a design representation for requirement *A* and *B** is a design representation for requirement *B*. Therefore, *A** and *B** are representations of concerns, and B* *cross-cuts* A*.
- An *aspect* is a representation of a cross-cutting concern. Therefore, the design representation, *B**, of the requirement, *a registered user must be validated prior to using* **SafeHomeAssured.com**, is an aspect of the *SafeHome* WebApp.

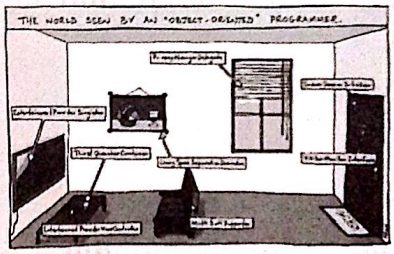
۳- مفاهیم مهم در طراحی – ادامه
Refactoring

- فرآیند تغییر در نرم افزار به نحوی که واسطها و مشخصات ظاهری نرم افزار تغییر نکند و تنها منجر به تغییرات داخلی در نرم افزار شود.

- هدف Refactoring بررسی موارد زیر است:
  - وجود افزونگی در کد و در داده
  - وجود عناصر غیر قابل استفاده
  - وجود الگوریتمهای غیر ضروری و ناکارامد در نرم افزار
  - وجود ساختمان داده های نامناسب و یا ساختار ضعیف
  - و هر نوع مشکلی که با برطرف کردن آن به طراحی بهتری برسیم

۳- مفاهیم مهم در طراحی – ادامه
OO Design Concepts(مفاهیم طراحی شی گرا)

- کلاس های طراحی
  - کلاس های موجودیت (Entity)
  - کلاس های مرزی (Boundary)
  - کلاس های کنترل(Controller)

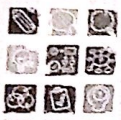


- Inheritance (وراثت):
  تمام مسئولیت های کلاس اصلی به تمام زیرکلاس ها به ارث می رسد.
- Messages(پیام رسانی):
  جهت تحریک رویدادهای مربوط به یک Object
- Polymorphism(چندریختی):
  با استفاده از این ویژگی، می توان طراحی انجام شده را به سهولت گسترش داد.

۳- مفاهیم مهم در طراحی

Design Classes (کلاس های طراحی)

- کلاس های تحلیل در حین طراحی Refine می شوند تا به کلاس های Entity تبدیل شوند
- کلاس های مرزی(Boundary) در حین طراحی توسعه می یابند تا واسط های کاربری را که کاربر رویت می نماید و در حین کار با نرم افزار با آن تعامل می کند را بسازند (مانند صفحات محاوره و یا گزارش های چاپی)
  - کلاس های مرزی با این هدف طراحی می شوند که روش های نمایش اشیاء Entity به کاربران را مدیریت نمایند
- کلاس های کنترل (Controller) جهت مدیریت موارد زیر طراحی می شوند
- ایجاد یا به روز رسانی اشیاء Entity
- نمونه سازی اشیاء Boundary به گونه ای که بتوانند اطلاعات موردنیازشان را از اشیاء Entity دریافت نمایند
- برقراری ارتباط های پیچیده بین مجموعه های مختلف اشیاء
- اعتبار سنجی ارتباطات داده ای بین اشیاء و یا بین کاربران و نرم افزار



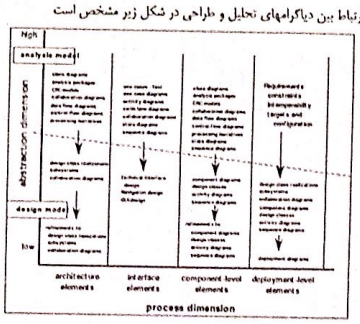
۴- مدل طراحی – ادامه

عناصر مدل طراحی

- عناصر داده ای
  - مدل داده-> ساختمان داده
  - مدل داده -> معماری پایگاه داده
- عناصر معماری
  - دامنه برنامه کاربردی
  - کلاس های تحلیل، ارتباط بین آنها، همکاری و رفتار آنها در طراحی تحقق می یابد
  - الگوها و سبک ها (فصل ۹ الی ۱۲)
- عناصر رابط (interface)
  - رابطهای کاربری
  - رابط های خارجی به سایر سیستم ها، دستگاهها، شبکه و یا سایر تولیدکنندگان و مصرف کنندگان اطلاعات
  - الگوها و سبک ها (فصل ۹ الی ۱۲)
  - رابط های داخلی بین کامپونت های مختلف طراحی
- عناصر کامپونت
- عناصر توسعه(Development)

۴- مدل طراحی

ارتباط بین دیاگرامهای تحلیل و طراحی در شکل زیر مشخص است



۴- مدل طراحی

عناصر مدل طراحی

عناصر معماری

- مدل معماری از ۳ منبع به دست می آید:
  - اطلاعات مربوط به دامنه برنامه ای که باید ساخته شود
  - عناصر خاصی از مدل نیازمندی مانند دیاگرام های جریان داده یا کلاس های تحلیل، روابط و همکاری های آنها در مساله موردنظر
  - الگوهای معماری (فصل ۱۲) و سبک هایی (فصل ۹) که به آنها دسترسی داریم.

۴- مدل طراحی

عناصر مدل طراحی

عناصر رابط

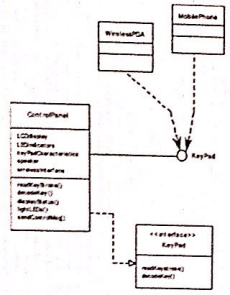
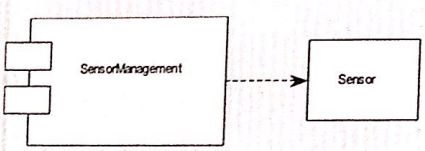


Figure 9.6 UML interface representation for ControlPanel

۴- مدل طراحی

عناصر مدل طراحی

عناصر کامپونت



۴- مدل طراحی

عناصر مدل طراحی

عناصر توزیع