Diagramy tried (Class diagrams)

UML (Unified Modeling Language)
- **koniec 80-tych, začiatok 90-tych rokov – boom OO metód, každá pozostáva z notácie a postupu a každá mala vlastnú notáciu**
- **používa sa na:**
  - **dokumentáciu**
  - **komunikáciu**

Diagramy tried:
- **srdce OO metód, má aj najviac features**

znázorňuje typy objektov a ich statické vzťahy
- **asociácie**
- **podtypy**

Fig.4.1

Perspektívy – čo vlastne kreslíme:
- **konceptuálny model (nezávislá od softvéru)**
- **špecifikácia [softvéru] – typy (interfejsy, nie implementácie)**
- **implementačný model – triedy**

keď kreslíš, vyber si perspektívu
keď čítaš, ujasni si perspektívu

Fig.4.1

Asociácie
- **konceptuálny model: znamená to existenciu vzťahu (Order-Customer)**
  - **role, násobnosti**
- **špecifikácia: responsibilities - že sa dajú zistiť vzťahujúce sa entity a že ich treba niekde updatovať**
  Vzťah Customer-Order: že viem od Customera zistiť, aké dal Ordery. Podobne od Orderu viem zistiť, ktorý Customer ho dal.
- **implementačný model – asociácia znamená, že existujú prepájacie dátové štruktúry**
- **navigovateľnosť – týka sa špec. a impl. modelu. Hovorí, že zodpovednosť(špec), resp. dátová štruktúra(impl) je na jednej strane.**
  (čo keď je neuvedená ? undecided/bidirectional)

Atribúty – podobné asociáciám
- **Customer: Name**
  - **koncept: že zákazníci majú mená**
  - **špecif: že sa dá meno zistiť a nastaviť**
  - **impl: že je tam skutočne ten atribút**
- **môže byť: viditeľnosť meno : typ = default hodnota**
- **na rozdiel od asoc – jedna hodnota, neuvádza sa povinnosť, plus (pri šp/impl): navig. jedným smerom, objekt má svoju vlastnú kópiu**

Operácie
- **špecif: public metódy typu (normálne sa manipulačné metódy k atribútom neuvádzajú)**
- **impl: všetky (ak treba)**

visibility(+#-) name (paramlist) : returntype
+ latest_amount_of (Phenomenon_type type) : Quantity
- **pri konc.modeli nešpecifikovať rozhranie – skôr hlavné zodpovednosti (účel triedy) – vziať use case a pozrieť sa, ako by ho mohli triedy implementovať.**

(prídeme k tomu pri analýze)
- **operácie vs. metódy**

Generalizácia
- **konceptuálne: keď všetky inštancie podtypu sú zároveň inštanciami nadtypu, t.j. všetko čo povieme o nadtype (asoc, attr, oper) platí aj pre podtyp**
- **špecifikácia: interfejs podtypu zahŕňa všetko z interfejsu nadtypu, resp. tiež princíp zameniteľnosti**
- **implementácia: dedičnosť v programovacích jazykoch**

Key point – interface inheritance vs. implementation inheritance (in-inher sa dá urobiť aj inak ako cez dedenie)

Constraints – ohraničenia (nevyjadriteľné gr.jazykom)
- **používajú sa vtedy, keď grafické prvky nestačia**
- **{} – aj neformálny NL, aj formálny**

Agregácia a kompozícia - fig 5.3

Rozhrania a abstraktné triedy
- **super vec: oddelenie interfejsu od implementácie**
- **podčiarknutie faktu, že ide o interfejs**
- **{abstract}, <<interface>>**
- **fig. 5-7 (windows)**
- **(v implementačnom modeli rozlišujeme generalizáciu a refinement)**

{ordered}, {bag}, {ordered bag}, {hierarchy}, {dag}
(polygon – points)

Parametrizované triedy
- **fig. 5-15, 5-16, 5-17.**

Tipy:
- **nepoužívaj všetku notáciu**
- **ujasni si perspektívu (pri práci s programom sú najdôležitejšie špecifikačné d.)**
- **sústreď sa na niekoľko kľúčových diagramov (nekresli všetky triedy)**

Package Diagrams

Cieľ: zachytiť štruktúru na vyššej úrovni
Dá sa použiť všade, najčastejšie na grupovanie tried

Obr 7.1

Pojmy: package, dependency

Interaction Diagrams

(doteraz sme mali statickú stránku, teraz dynamickú)

ID znázorňujú, ako objekty spolupracujú (interagujú) pri realizácii 1 funkcie. (Pri OO je kód realizujúci 1 funkciu často roztratený po mnohých objektoch-triedach.)

Zachytiť chceme scenár:
Oder Entry window: Order.prepare (na už vyplnený Order)
Order: * OrderLine.prepare
OrderLine:
- **StockItem.check (či je vôbec na sklade)**

- **ak áno, vezmi a zisti, či netreba objednať**
- **ak nie (...)**

Obr. 6.1 aj s aktiváciami a návratmi

Obr 6.2, 6.3

Collaboration Diagrams (obr 6.4)

Stavové diagramy

<u>Znázorňujú jeden objekt a jeho stavy (počas vykonávania viacerých funkcií)</u>

Príklad s telefónom

Pojmy:

- **stav**
- **event – udalosť, kt.spôsobuje zmenu stavu**
- **guard**
- **action – akcia, ktorá sa má pritom vykonať**
- **do, entry, exit / activity – čo sa má pritom robiť**

obr 8.1

stav Cancel

obr 8.3, 8.5