Objektově orientovaný návrh

- pojem "návrh"
  - návrh architektury systému viz probráno
  - podrobný návrh systému - každá součást systému je popsána natolik podrobně, aby to postačovalo pro kódování

- vstupem jsou analytické třídy představující role - budou pokryty jednou nebo více třídami návrhu (analytická třída se v návrhu může stát jednou třídou, částí třídy, agregovanou třídou, skupinou spřízněných tříd, asociací, naopak asociace třídou apod.)
- vytvoření návrhových tříd (design classes)
- definice operací
- definice asociací, agregací a kompozic

- pro analytické třídy budeme vytvářet jednu nebo více návrhových tříd
  - návrh hraničních tříd
    - jedna hraniční třída bude odpovídat jednomu oknu nebo formuláři
    - jedna třída pro každé API nebo protokol

  - entitní (datové) třídy
    - často pasivní a perzistentní, vybereme pro ně způsob implementace: v souboru, v relační databázi (z perzistentních tříd vytvoříme tabulky atd.)
    - nejsou-li třídy perzistentní, pak budou implementovány v paměti

  - řídící třídy - obsahují aplikační logiku

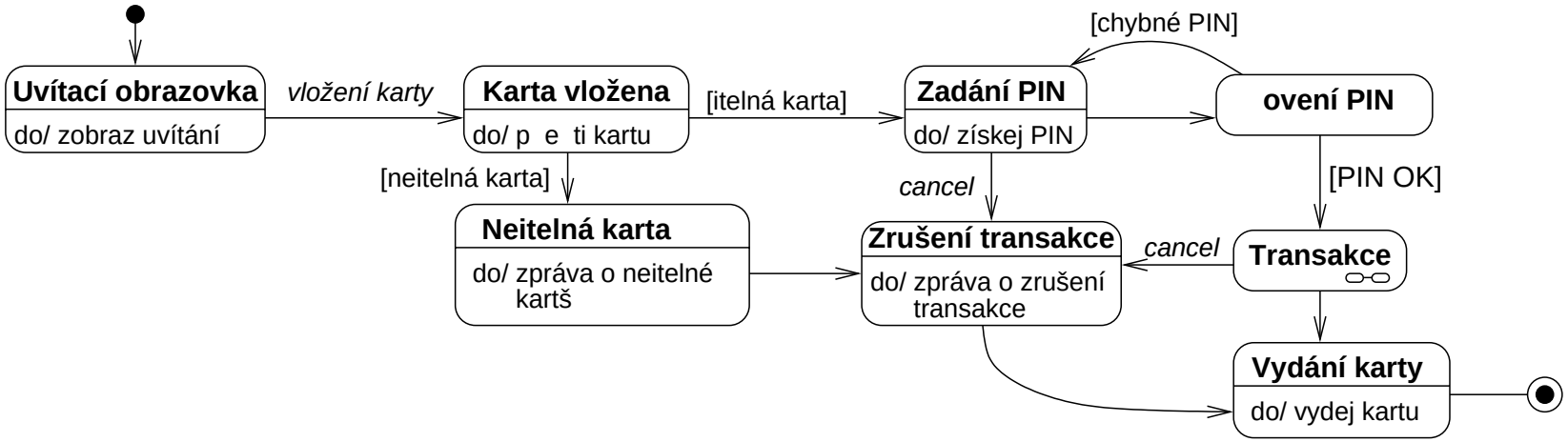
Definice operací tříd:

- standardní konstruktory se předpokládají => z návrhu je vynecháváme
- operace, které čtou a nastavují veřejné atributy, se (většinou) předpokládají

- hledáme operace - prvním vodítkem již vytvořený seznam sloves
- další způsob - získat operace z popisu interakce objektů
  - nakreslíme diagramy spolupráce nebo sekvenční diagramy (resp. upravíme diagramy získané ve fázi analýzy, doplníme zasílané zprávy)
  - z nich zjistíme, jaké stimuly musí být objekt schopen přijmout, navrhneme odpovídající operace

- další možnosti, co může být operace:
  - inicializace nově vytvořené instance včetně propojení s asociovanými objekty
  - případné vytvoření kopie instance (pokud je zapotřebí)
  - případný test na ekvivalenci instancí (pokud je zapotřebí)

- operace popíšeme: název, parametry, návratová hodnota, krátký popis, viditelnost
- pokud je algoritmus složitý,
  - popíšeme metodu (= implementaci operace)
  - nakreslíme stavový diagram objektu nebo operace; např. stavový diagram pro bankomat:



- základní kritérium: každá metoda by měla dělat jednu věc dobře

Definice atributů tříd

- vycházíme z logických atributů (produkt analýzy) popisujících, co je potřebné pro uchování stavu objektu
- další možnost - jaké atributy jsou třeba pro implementaci operací

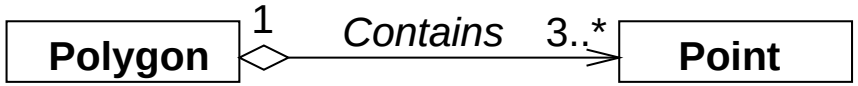
- atributy v návrhu
  - mají být "jednoduché" (int, boolean, float, ...)
  - nebo mají mít sémantiku hodnoty (tj. být nezměnitelné, např. String)
  - jinak použijeme asociaci

- atributy popíšeme: jméno, typ, počáteční hodnota, viditelnost
  - snažíme se o skrývání informací - soukromé atributy (viditelné pouze pro potomky = protected "#")

- ověříme, že všechny atributy jsou potřebné
- pokud zjistíme, že se atributy a operace dělí do dvou tříd, které spolu příliš nesouvisejí, pak se pravděpodobně jedná o dvě různé třídy – měli bychom třídu rozdělit

Definice asociací, agregací a kompozic

- stejně jako v analýze, máme ale podrobnější informace o chování
- navíc můžeme přidat tzv. průchodnost (navigability) - od kterého ke kterému objektu budeme chtít snadno přecházet - označuje se šipkou v asociačním vztahu



- pokud třída obsahuje více než cca 10 atributů, 10 asociací nebo 20 operací, není dobře navržená a potřebuje rozdělit

Definice zobecnění (hierarchie dědičnosti)

- stejně jako v analýze, tj. společné vlastnosti vyjmeme do nadtříd
- pokud má některá třída sjednocení atributů svých podtříd (místo průniku), zřejmě spolu podtřídy ve skutečnosti nemají nic společného
- hierarchie by měla být vyvážená, tj. neměly by být třídy, pro které je hierarchie neobvykle plochá nebo naopak hluboká

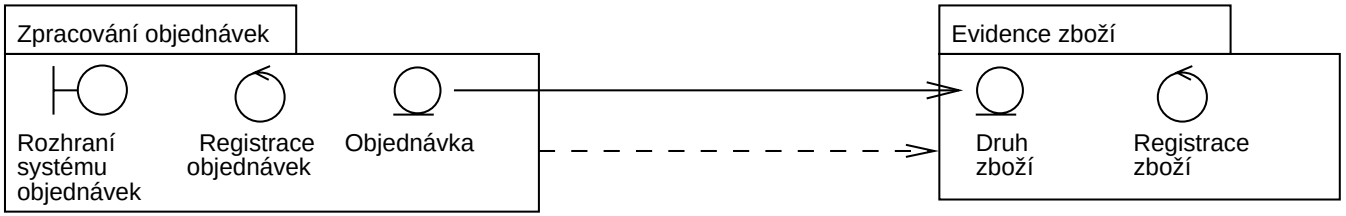
Kontrola modelu

- ověřit realizace případů užití, v návrhu nám nesmí chybět chování potřebné pro některý případ užití

Volitelně vytváření balíčků

- pro lepší srozumitelnost (strukturování),
- pro izolaci částí, které se budou častěji měnit (případně obojí)

- do balíčku vkládáme funkčně příbuzné třídy (změna chování jedné třídy způsobí změnu chování druhé)
- třídy uvnitř balíčku veřejné (public) i soukromé (private)
  - veřejná třída může mít asociace s libovolnou jinou třídou; tvoří rozhraní balíčku
  - soukromá třída může být asociována pouze se třídami uvnitř stejného balíčku

- pokud má třída v jednom balíčku asociaci se třídou v jiném balíčku, pak balíčky na sobě navzájem závisejí - modeluje se vztahem závislosti (přerušovaná šipka)

- v jednom diagramu můžeme zakreslit i závislosti mezi třídami apod.