# +- Návrhové vzory I

# Skrývání implementace

## Zástupce (Proxy)

* Nahradí objekt jiným zástupným objektem
* Vizuálně stejný, ale můžeme zasahovat do jeho chování
* Umožní zachytávat a převádět volání na jiná
* Implementován rozhraním nebo abstraktní třídou
* Vzdálený
  + Zastupuje objekt, který se nachází jinde
  + Zprostředkovává komunikaci mezi zastupovaným vzdáleným objektem a objekty z okolí zástupce
  + Musí být stále připraven na možnost selhání
  + Bankomat komunikující s bankou
* Ochranný
  + Kontrola přístupu k datům na základě práv
* Virtual
  + Používá se, když potřebujeme odložit vytváření objektu (velký obrázek je nahrazen virtual proxy, zatímco se načítá)

## Příkaz (Command)

* Zabalí metodu do objektu
* Reakce na určitou akci
* Oddělení spouštěcí akce od akce činící



## Iterátor (Iterator)

* Samostatný objekt umožňující lineární procházení kolekcemi bez znalosti jejich vnitřní implementace
* V C# na kolekce implementující IEnumerator lze použít foreach = **Implicitní iterátor**
* **Explicitní iterátor** si musíme sami napsat, instance iterátoru na požádání vrací odkaz na další prvek v kolekci, dokud nedojede k prvku poslednímu
* **Externí iterátor**: průchod kolekcí řídí klient s využitím iterátoru; složitější, ale mocnější řešení
* **Interní iterátor**: průchod kolekcí řídí iterátor sám; klient specifikuje, co provádět s prvky kolekce (např pomocí LINQ + lambda) Obsah obrázku text

  Popis byl vytvořen automaticky

## Stav (State)

* Řeší velké rozdíly v chování objektu v určitých stavech
* Objekt má konečný počet definovaných stavů a vždy se může nacházet v jednom z nich
* Animace v Unity

## Šablonová metoda (Template Method)

* Definuje „kostru“ algoritmu
* Neobsahuje všechny kroky – prázdné virtuální metody pro přepsání potomky
* Mění části algoritmu beze změny původní části algoritmu
* Způsob odstranění duplicit v kódu
* Pro úlohy s parametry, které budou známy až za běhu
* Použití:
  + Když chceme, aby se naše aplikace dala rozšířit, ale ne modifikovat
  + Když se nám v kódu objevují podobné algoritmy (lišící se jen v pár krocích, kostra je stejná) Obsah obrázku text, snímek obrazovky, plaketa

    Popis byl vytvořen automaticky

# Optimalizace rozhraní

## Fasáda (Facade)

* Zjednodušuje komunikaci mezi uživatelem a systémem
* Vytvoření jednotného rozhraní pro celou logickou skupinu tříd, které se tak sdruží do subsystému, který je většinou příliš složitý (toolbox, imagine brát si nářadí s jedné hromady)
* Zabalí komplikovaný subsystém do jednoduššího uceleného rozhraníObsah obrázku text

  Popis byl vytvořen automaticky

## Adaptér (Adapter)

* Převede zastaralé / nehodící se / chybné rozhraní třídy na rozhraní, které klient očekává
* Zabezpečuje spolupráci tříd a usnadňuje implementaci nových
* Může celou třídu zabalit do nové (object adaper), nebo z ní dědit (class adapter)

## Strom (Composite)

* Popisuje skupinu objektů, se kterými je zacházeno stejně, jako s jednou instancí tohoto objektu
* => ignorování rozdílu mezi vnořenými objekty a samostatným objektem
* Složka která obsahuje další prvky, což mohou být další složky