GRASP Patterns

(General Responsibility Assignment Software Patterns)

GRASP Patterns

General **R**esponsibility **A**ssignment **S**oftware **P**atterns

Patterns or **D**esign **P**rincipals?

The full set of GRASP patterns are:

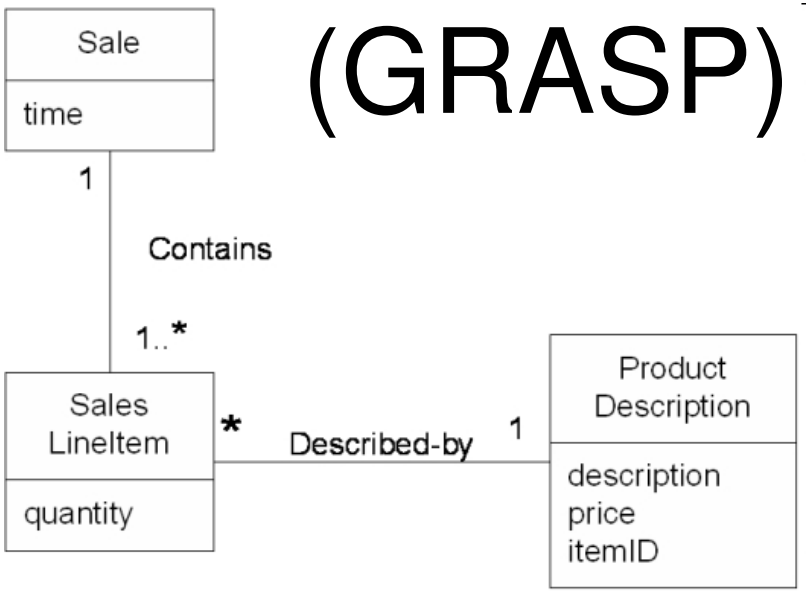
- Information Expert
- Creator
- Controller
- Low Coupling
- High Cohesion
- Polymorphism
- Pure Fabrication
- Indirection
- Protected Variations

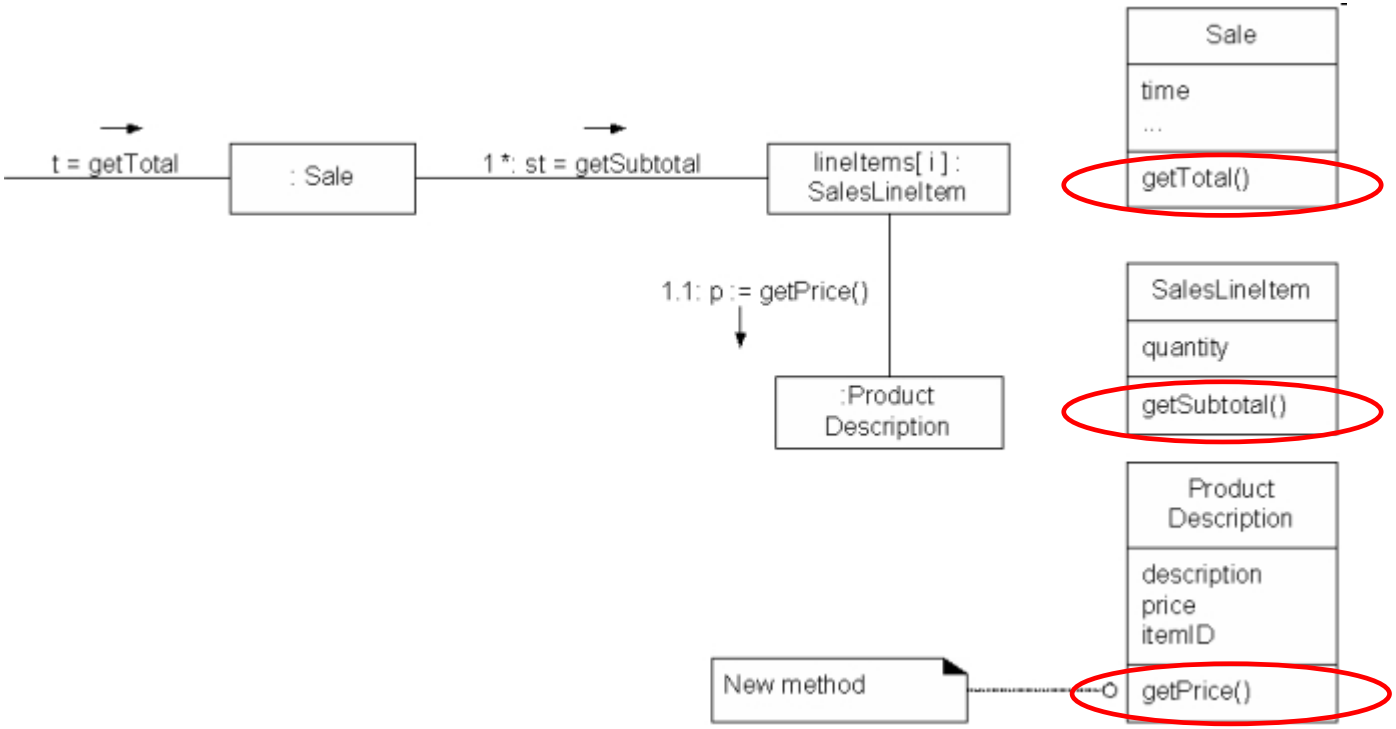
(GRASP) Information Expert

- V průběhu návrhu se rozhodujeme, jakou zodpovědnost za vykonávání kterých úkolů přidělíme jednotlivým třídám.

- Princip *Information Expert* says: Přiřaď zodpovědnost informačnímu expertovi, t.j. třídě, jež má k dispozici informaci nezbytnou pro vykonání příslušného úkolu.

(GRASP) Information Expert

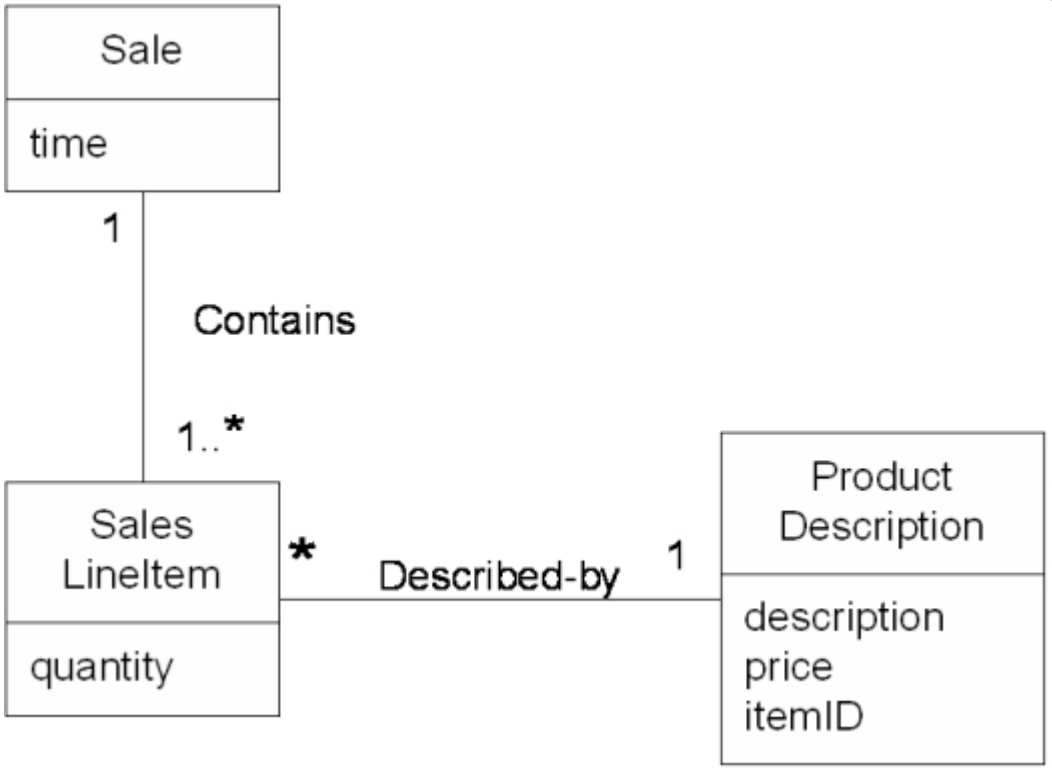




(GRASP) Creator

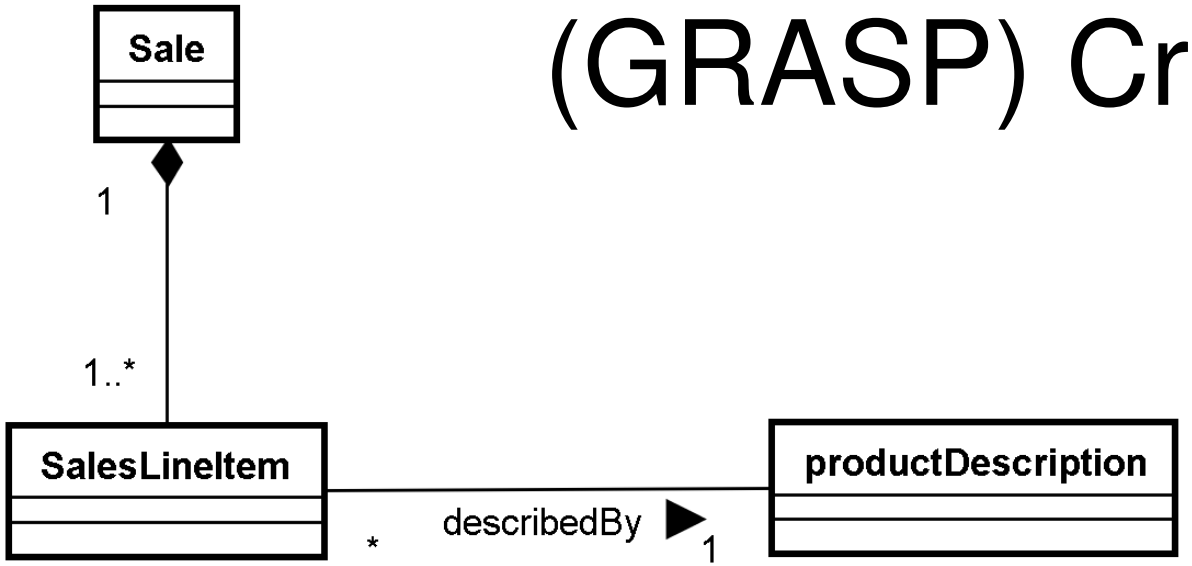
- V průběhu návrhu se rozhodujeme, který objekt má být zodpovědný za vytvoření nových objektů.
- Objekt typu B je tvůrcem (creator) objektů třídy A, když třídy A a B splňují aspoň jednu z těchto podmínek:
  - B *agreguje* A objekty (vztah celek - části).
  - B *obsahuje* A objects (vztah kompozice – objekty B nemohou existovat bez vtahu k A)
  - B *eviduje* instance A objektů.
  - B úzce využívá A objekty.
  - B *vlastní inicializační data, která jsou potřebná pro konstrukci objektů* A (B je expert vzhledem k vytváření objektů A).
- Je-li aplikovatelný více než jeden z těchto bodů, má přednost agregace a kompozice
- Pokud je tvorba objektu A složitá, může se tato tvorba delegovat speciálně navržené třídě (např. Abstract Factory).

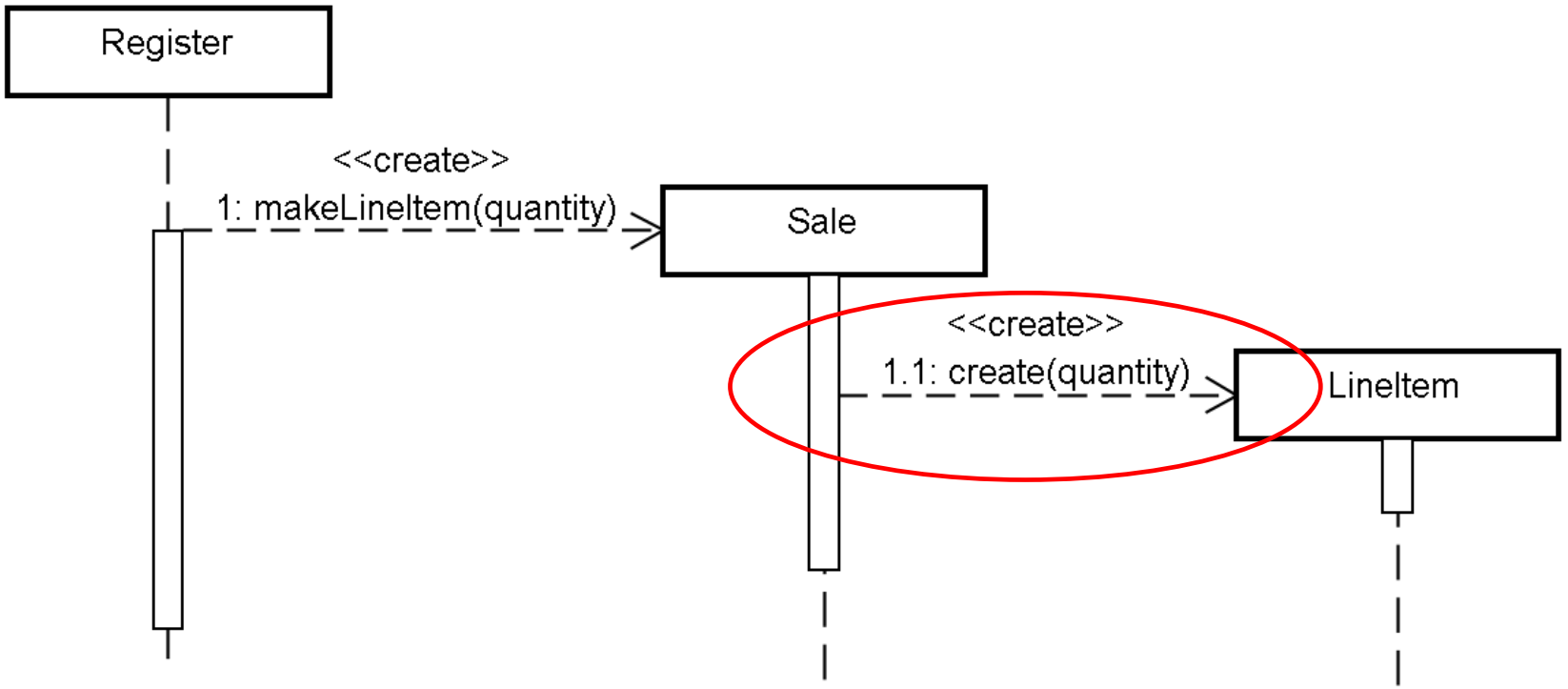
(GRASP) Creator



Jak to bude vypadat zde?

(GRASP) Creator





(GRASP) Controller

Kdo obsluhuje systémové události?

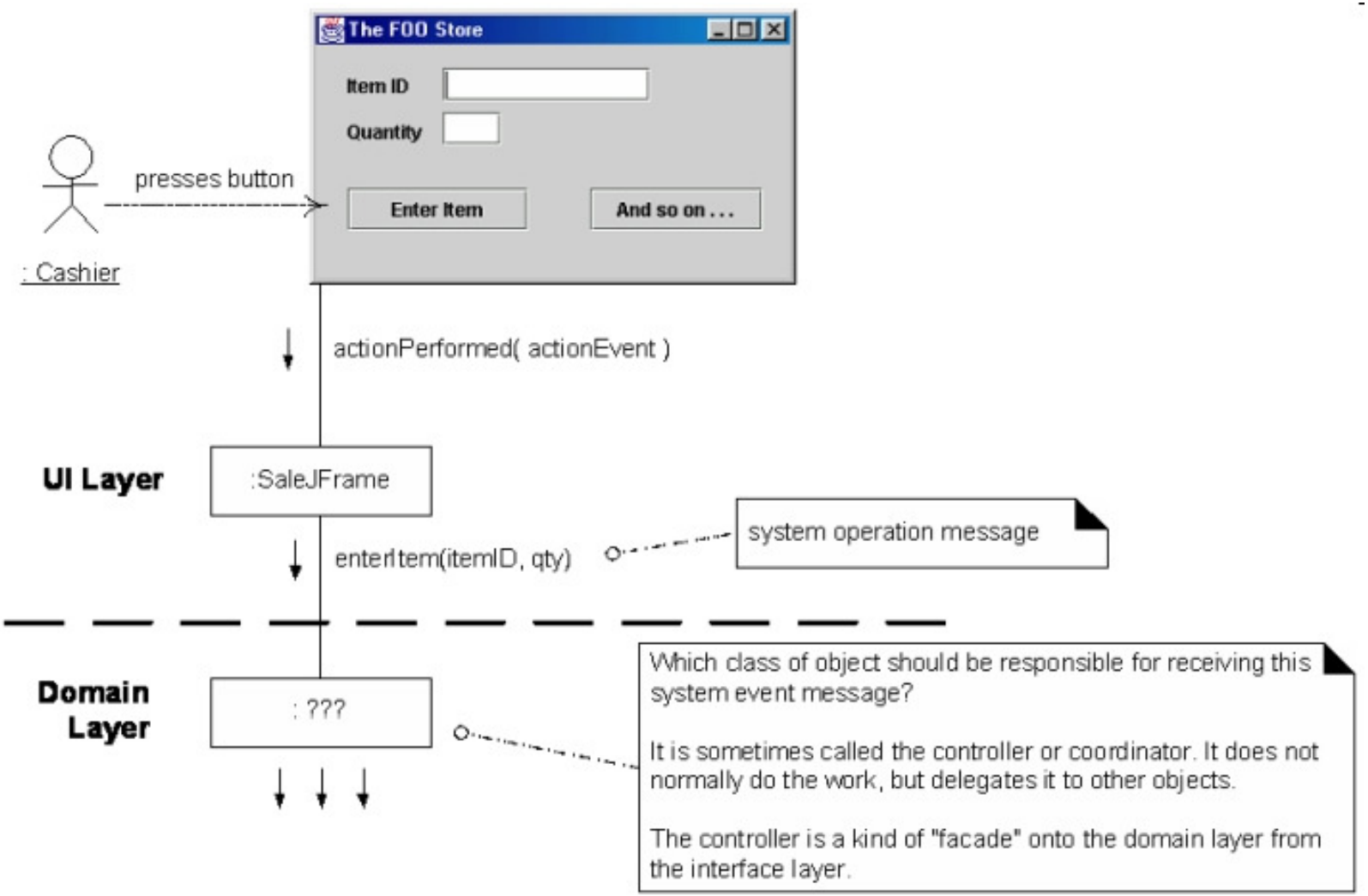
Události jsou často iniciovány z vnějšku navrhovaného systému. Náš systém bude mít třídu, která přijímá takovou událost, ale neznamená to, že ji tatáž třída má také obsloužit.
Princip Controller navrhuje, aby událost obsloužila některá z následujících tříd:

- Třída, jež reprezentuje celý systém, zařízení nebo subsystém (*facade controller*).
- Třída, jež reprezentuje příslušný scénář případu užití, během něhož událost nastala.

  Jméno takové třídy by mělo být <usecasename>Handler, <usecasename>Controller, <usecasename>Manager, atp.

V případě volby druhého případu se v kontextu daného případu užití často používá tentýž controller pro celý systém. Jediný důvod, proč to tak nedělat, by mohla být skutečnost, že by tento controller byl příliš složitý. V takovém případě je lepší práci rozdělit mezi více kontrolerů.

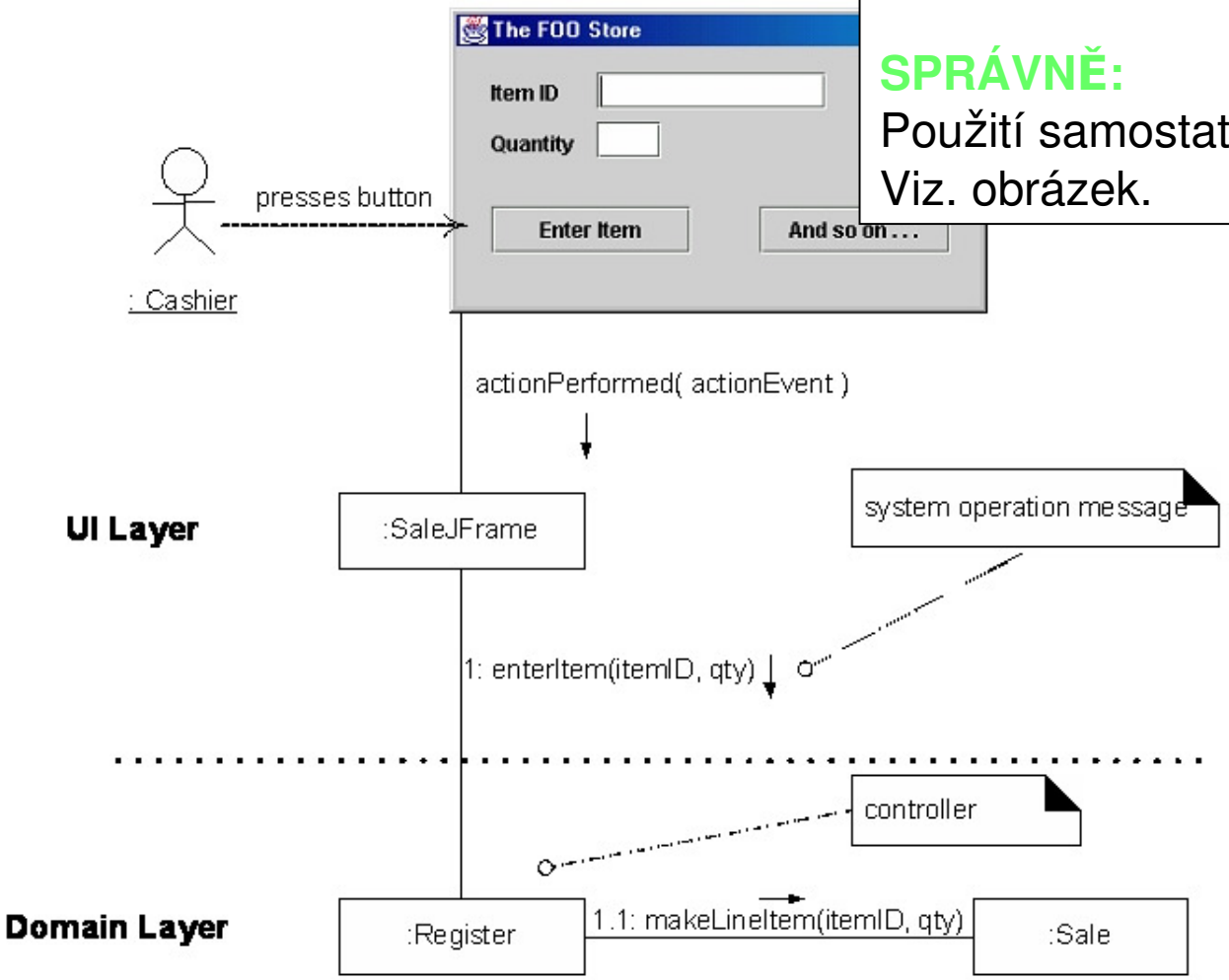
(GRASP) Controller



(GRASP) Controller

ŠPATNĚ:
Někdy vývojáři používají GUI objekty přímo jako controllery.controller objects.

SPRÁVNĚ:
Použití samostatného objektu – controlleru. Viz. obrázek.



(GRASP) Protected Variations

Princip:

1. Identifikovat body návrhu, které mohou podléhat změnám (body nestability)

2. Přiřadit zodpovědnosti tak, aby kolem těchto bodů nestability vzniknul stabilní interface v nejširším významu toho slova (ne nutně java interface!), který oddělí nestabilní část od části stabilní.

Tento pricip je velmi široký a jeho interpretace nemusí být v konkrétním případě jednoduchá.