PROGRAMOZÁS II
4. ÓRA: OBJEKTUM ORIENTÁLT ELEMZÉS ÉS TERVEZÉS

Átfogó példa

- Eladás folyamat egy üzletben
  - Árucikkek
  - Értékesítés
  - Szereplők

- Milyen osztályok legyenek?
- Hogyan kapcsolódjanak egymáshoz?

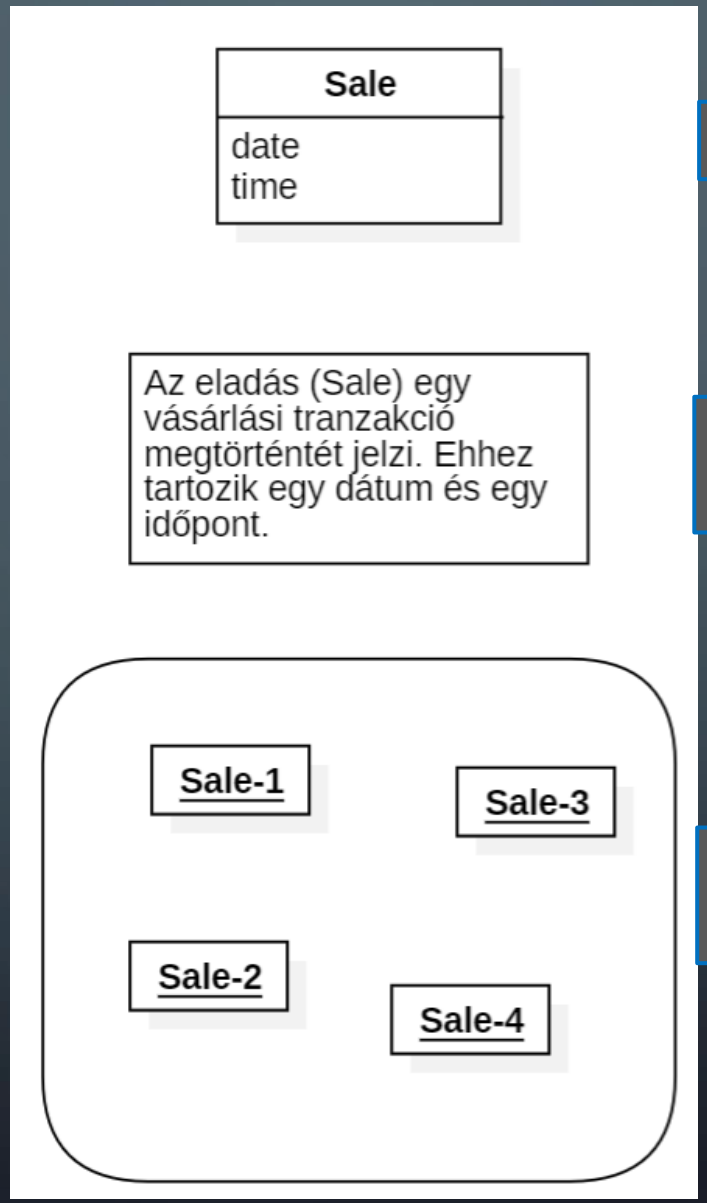
Objektum orientált elemzés és tervezés (OOA&OOD)

- Domain modell
  - I. Meghatározzuk a koncepcionális osztályokat
  - II. Hozzátesszük az összefüggéseket (asszociációkat)
  - III. Hozzátesszük az attribútumokat
- Design modell
  - IV. Felelősségek hozzárendelés
    - Metódusok megadása

Domain modell elkészítése

- Azonosítsuk a koncepcionális osztályokat
- Készítsük el a kezdeti domain modellt
- Tegyünk különbséget korrekt és nem korrekt attribútumok között
- Leíró osztályokat készítsünk ahol annak helye van
- Figyeljünk, hogy ne keverjük össze a koncepcionális és az implementációs nézeteket

Koncepcionális osztály



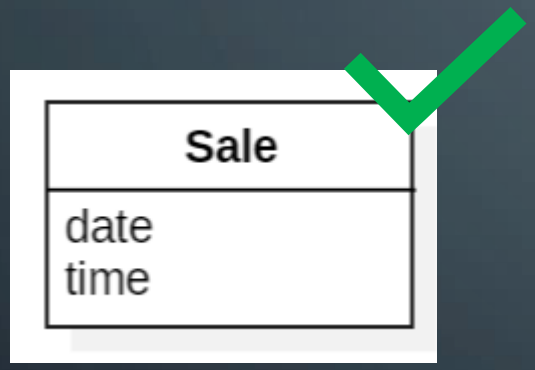
Koncepció

Koncepció kifejtése

Koncepció megvalósítása

Alapvető irányelv

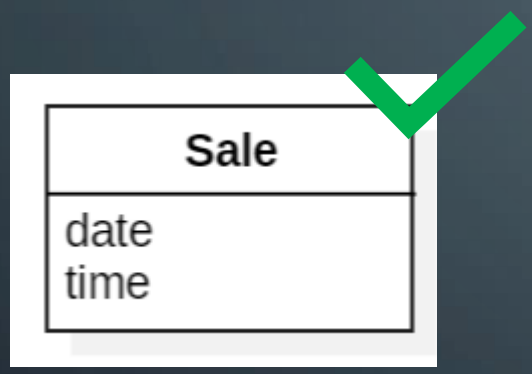
- A domain modell a valóságot írja le koncepcionális osztályokkal.
- Nem egy ábra ami a szoftver osztályokat / objektumokat vagy azok felelősségét írja le.



Nem egy programbéli osztály, hanem egy valós koncepció leírása csak a szükséges részletekkel

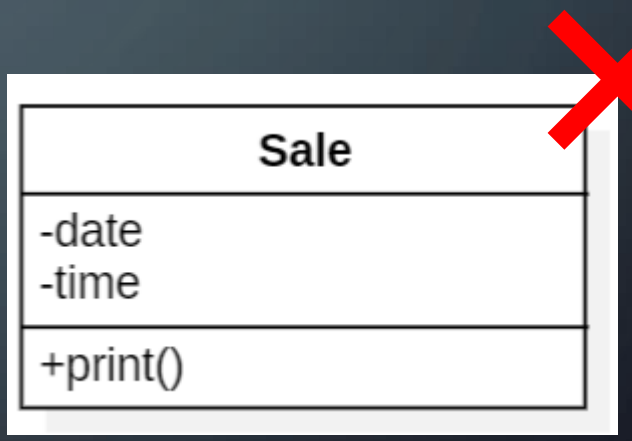
Alapvető irányelv

- A domain modell a valóságot írja le koncepcionális osztályokkal.
- Nem egy ábra ami a szoftver osztályokat / objektumokat vagy azok felelősségét írja le.



Nem egy programbéli osztály, hanem egy valós koncepció leírása csak a szükséges részletekkel

Szoftveres részlet, nem a domain modell része



Szoftveres osztály leírása, nem a domain modell része

Domain modell leírása

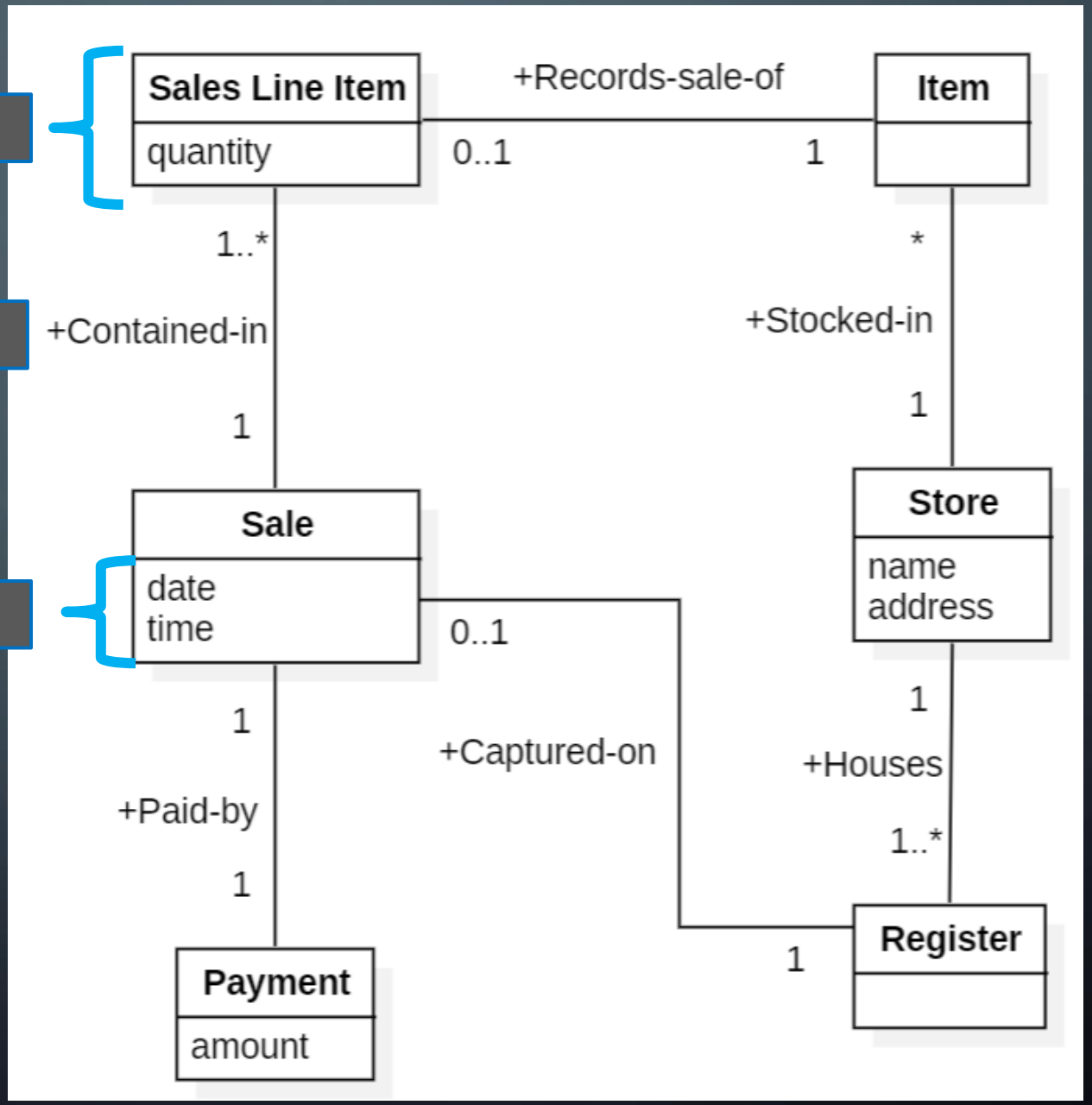
- Valóság objektumait / osztályait
- Koncepcionális osztályok közti összefüggéseket (asszociációk)
- Koncepcionális osztályok attribútumait
- Nincsenek metódusok

Példa

Koncepció

Asszociáció

Attribútum



Irányelvek

- A domain modell inkább túl részletes legyen, mint kevéssé
- Ne gondoljuk, hogy a kevesebb elemet tartalmazó modell jobb
- Gyakori, hogy elfelejtünk valamit első lépésben, később bővítjük
- Ne féljünk megjeleníteni olyan objektumokat amire a követelmények nem kényszerítenek

Hogyan azonosítjuk az osztályokat?

- Koncepcionális osztályok kategória listája alapján
- Követelmény leírásban főneveket keresünk

Koncepcionális osztályok kategória listája (példák)

- Fizikai, kézzel fogható objektum (pl.: pénztárgép, repülőgép)
- Leírása, terve, specifikációja valaminek (pl.: termékleírás)
- Helyek (áruház, repülőtér)
- Tranzakciók (leadás, fizetés)
- Tranzakciók lépései, tételei (eladás egy tétele)
- Más dolgok tárolói (raktár, repülő)
- Külső számítógépes vagy elektronikus rendszerek (hitelkártya leolvasó, légi irányító rendszer)
- Absztrakt fogalmak (éhség, rosszullét)

Koncepcionális osztályok kategória listája (példák)

- Szervezetek (zöldség osztály)
- Események (Találkozó, repülés, eladás, leszállás)
- Folyamatok (Egy termék eladása, helyfoglalás)
- Szabályok és szabályzatok (pénz visszafizetés szabályai, lemondás szabályai)
- Katalógusok (pl.: áru/termék katalógus)
- Feljegyzések (pénzügyi, jogi, vagy munkaügyi)
- Pénzügyi szolgáltatások és eszközök (hitel)
- Dokumentumok, referenciák, leírások, könyvek

Főnevek azonosítása a leírásban

- Fontos: Nem automatikusan használjuk arra, hogy minden főnévhez osztályt rendeljünk, de segít ellenőrizni, hogy nem hagytunk ki semmit.

- Pl.:
  - A vásárló megérkezik a pénztárhoz azokkal az árucikkekkel, amiket meg kíván venni.
  - Pénztáros leolvassa az árucikk azonosítóját.
  - …

Domain modell leírásának lépései

- I. Meghatározzuk a koncepcionális osztályokat
  - Felrajzoljuk a kezdeti domain modellt (csak osztályok)
- II. Hozzátesszük az összefüggéseket (asszociációkat)
- III. Hozzátesszük az attribútumokat

I. OSZTÁLYOK AZONOSÍTÁSA

Példa: osztályok

- Register
- Customer
- Store
- Sale
- Cashier
- Item
- Sales Line Item
- Ledger
- Payment
- Product Catalog
- Product Description

Hogyan nevezzük el az osztályokat?

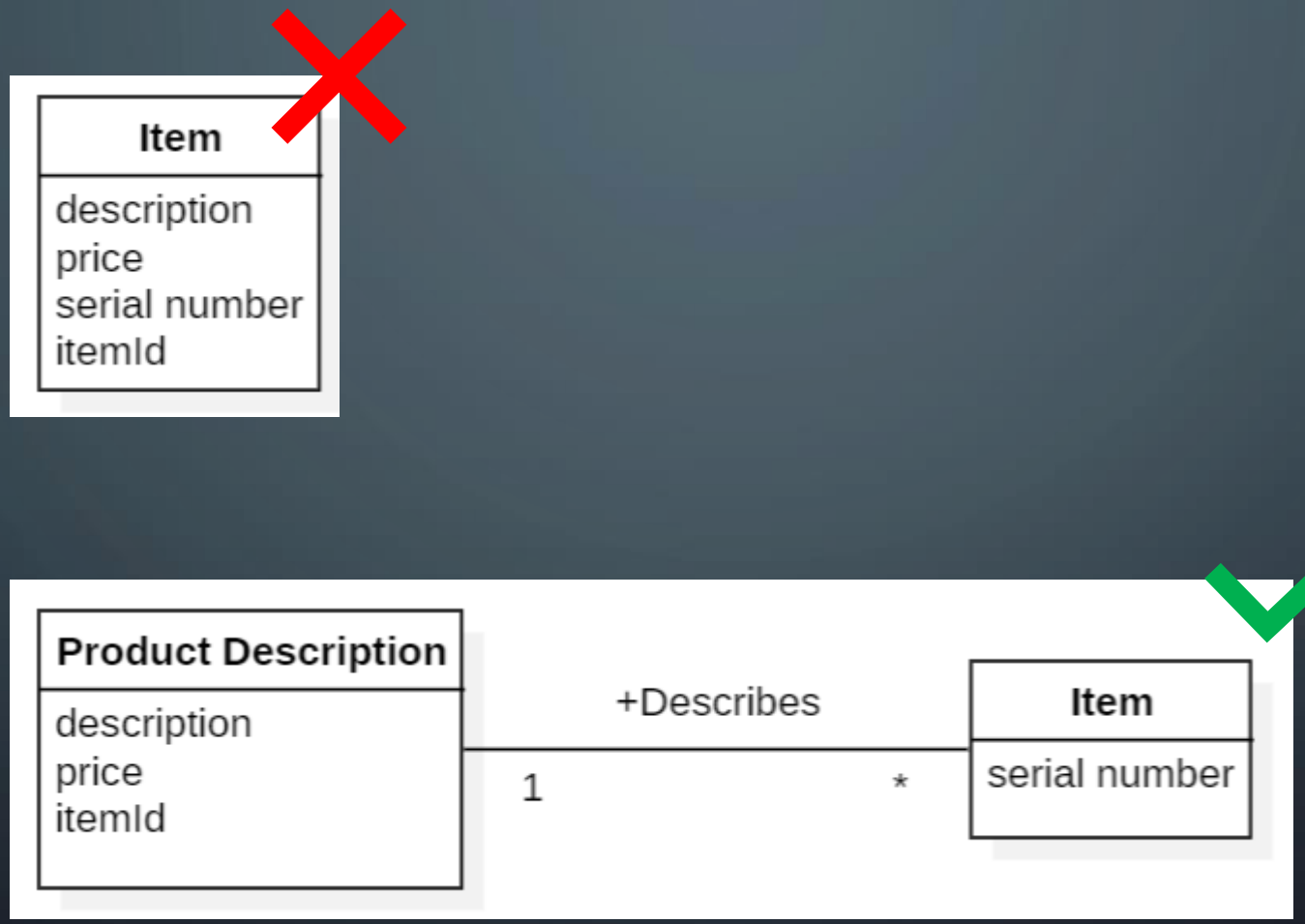
- Irányelv: Tegyünk úgy mint egy térképész:
  - Használjuk a meglevő helyi kifejezéseket
  - Hanyagoljuk el a nem fontos dolgokat
  - Ne vegyünk fel az ábrára olyanokat, mik nincsenek ott a valóságban

- Irányelv: Ha egy X dologról nem az jut eszünkbe, hogy az egy szám, vagy szöveg a valóságban, akkor valószínűleg a modellben X osztály kell legyen, nem attribútum.

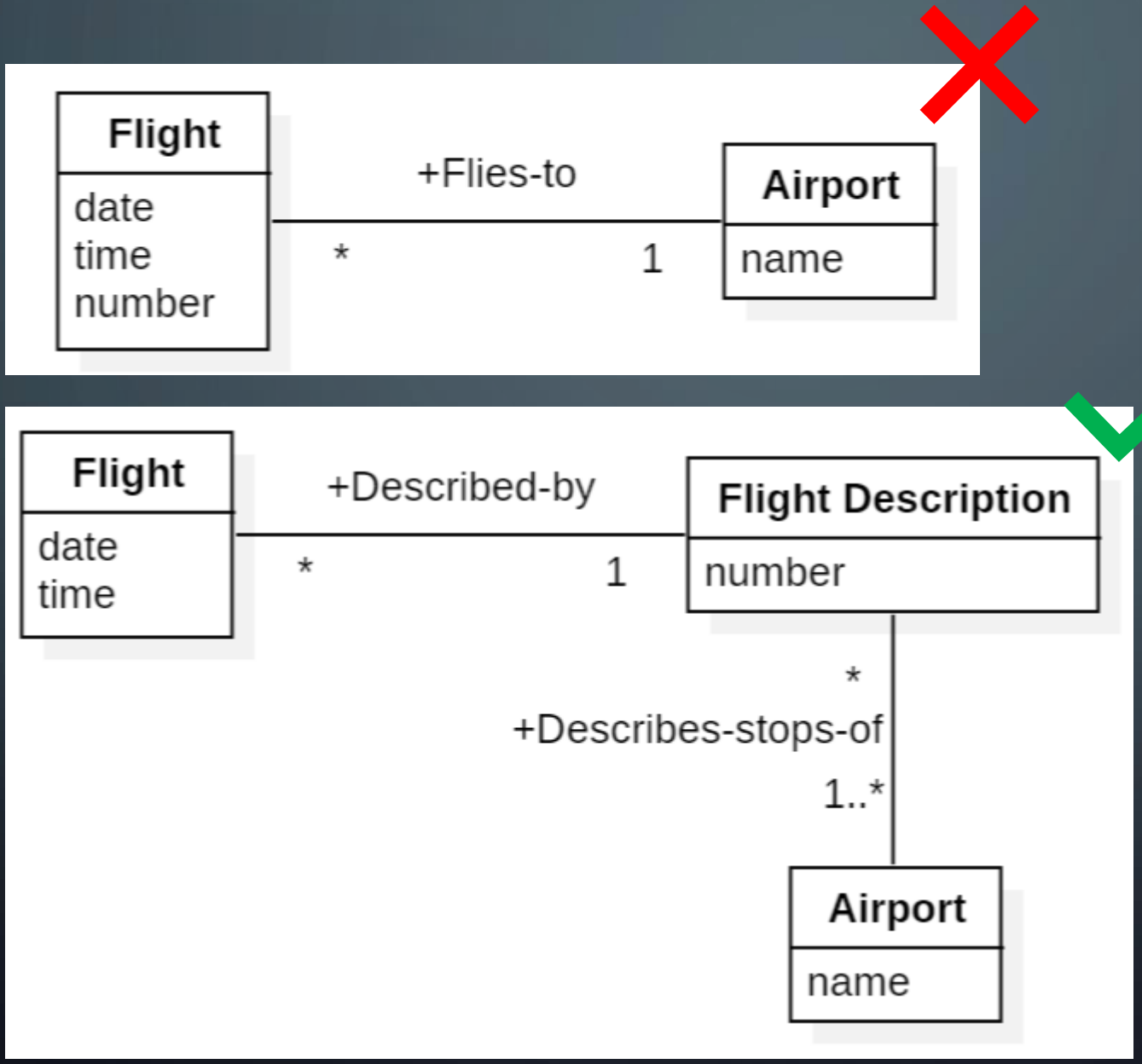
Leíró osztályok azonosítása

- Irányelvek:
  - Olyan leírása van egy árunak vagy szolgáltatásnak, ami független az aktuális példányoktól
  - Az aktuális példányokra vonatkozó információkat rendszeresen nem írjuk felül, hanem az információvesztés elkerüléséhez a leírást megőrizzük
  - Ha segít csökkenteni a redundanciát

Példa: leíró osztály



Példa: leíró osztály



UML és a domain modell

- UML: Unified Modeling Language
- UML alapvető diagram típusokat tartalmaz (osztálydiagram, szekvencia diagram)
- UML nem módszertanfüggő, nincs benne domain modell vagy implementációs modell
- Ez ne korlátozzon minket pl.: osztálydiagram használható koncepcionális, tervezési és implementációs modellben is

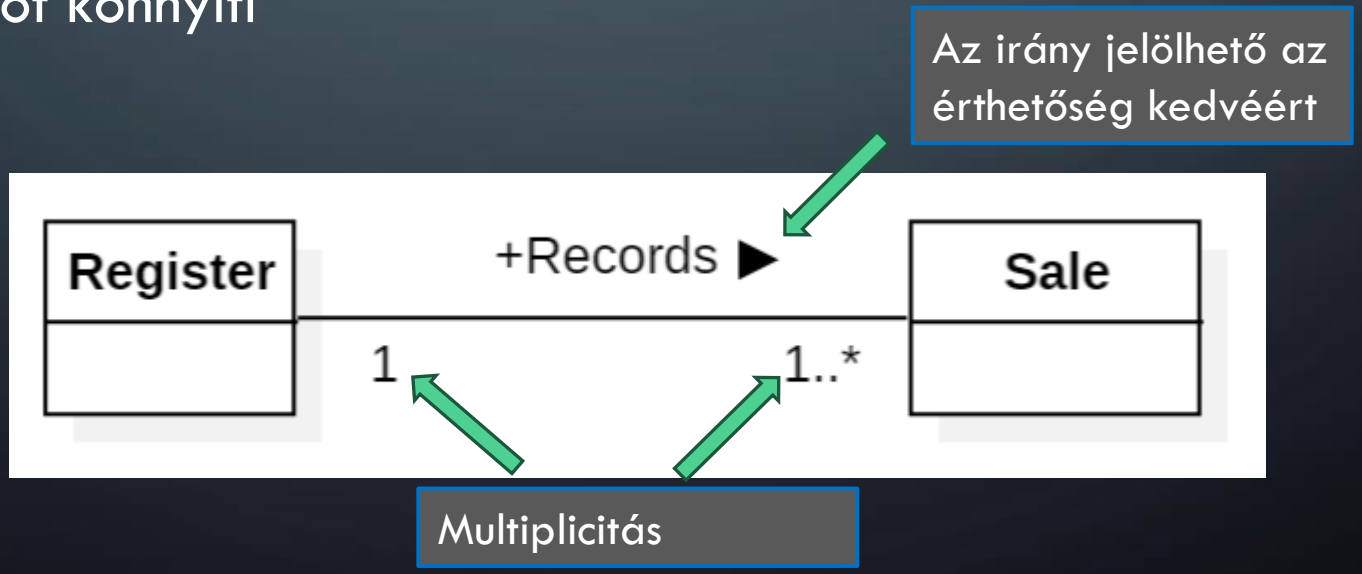
II. ASSZOCIÁCIÓK HOZZÁADÁSA

Asszociációk azonosítása

- Tudást tároló asszociációk
- Gyakori asszociációk listájából

Megjegyzés:

- A modell szempontjából az asszociáció iránya nem lényeges, de az olvashatóságot könnyíti



Gyakori asszociációk (példák)

- A fizikai része B-nek (szárny repülőgép)
- A logikai része B-nek (eladási tételeladás)
- A fizikailag benne vagy rajta szerepel B-n (pénztárgép üzlet)
- A rögzíti B-t
- A leírása B-nek (árucikk cikkleírás)
- A tétele, jelentése, tranzakciója B-nek (eladási tétel – eladás, javítás szervizkönyv)
- A ismert, rögzített, leírt vagy azonosított B-ben (eladás – pénztárgép, helyfoglalás – utazási iroda)
- A tagja B-nek (pénztáros-üzlet, pilóta repülőgép)

Gyakori asszociációk (példák)

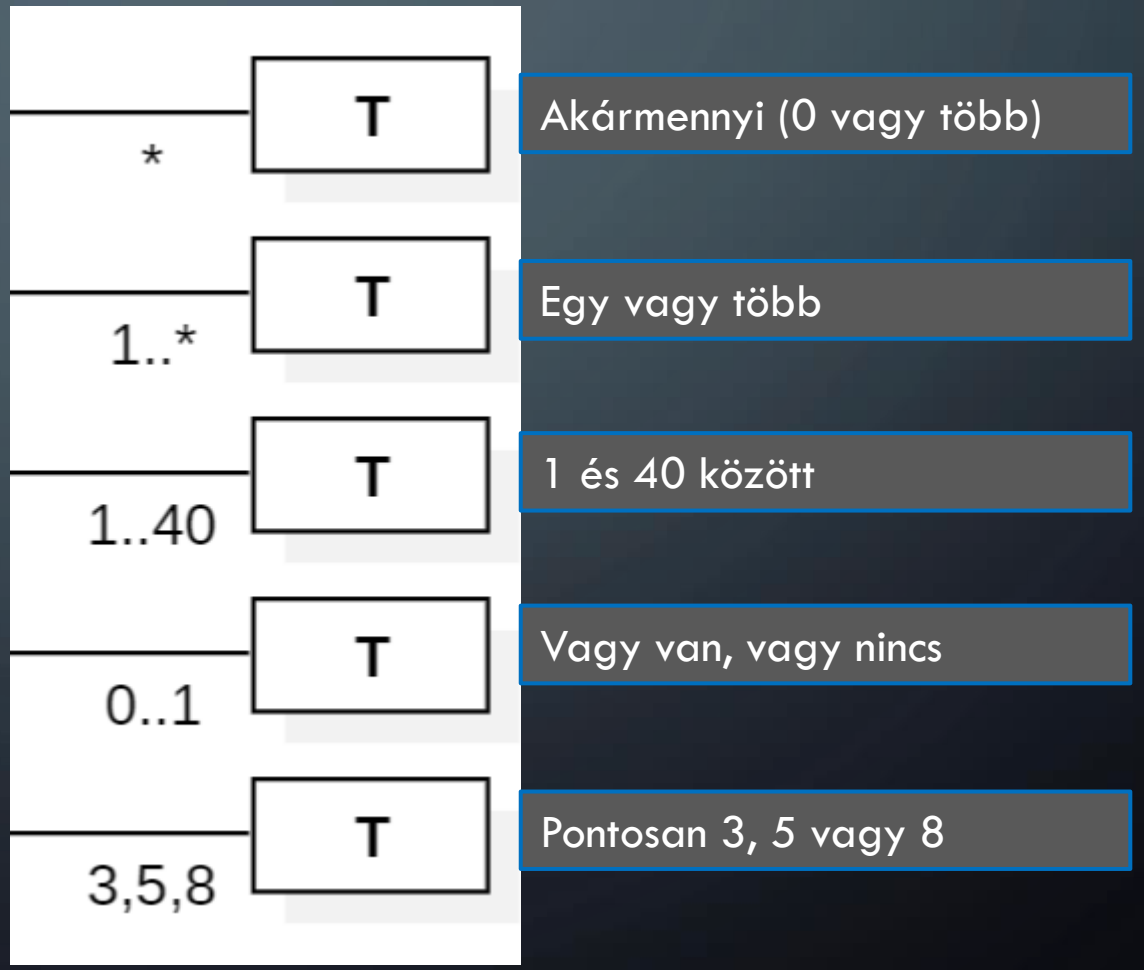
- A szervezeti egysége B-nek (osztály-áruház)
- A használja vagy irányítja B-t (pénztáros-pénztárgép, pilóta repülő)
- A kommunikál B-vel (vásárló – pénztáros)
- A kapcsolatban áll egy B tranzakcióval (vevő – fizetés, utas jegy)
- A tranzakció kapcsolatban áll B tranzakcióval (fizetés – vásárlás, foglaláslemondás)
- A követi B-t (számla tétel-számla tétel, város város)
- A tulajdonában van B-nek (pénztárgép – áruház, repülőgép – légi társaság)
- A egy esemény, ami kapcsolatban van B-vel (eladás – vásárló, eladás – üzlet, leszállás repülőgép)

Irányelvek

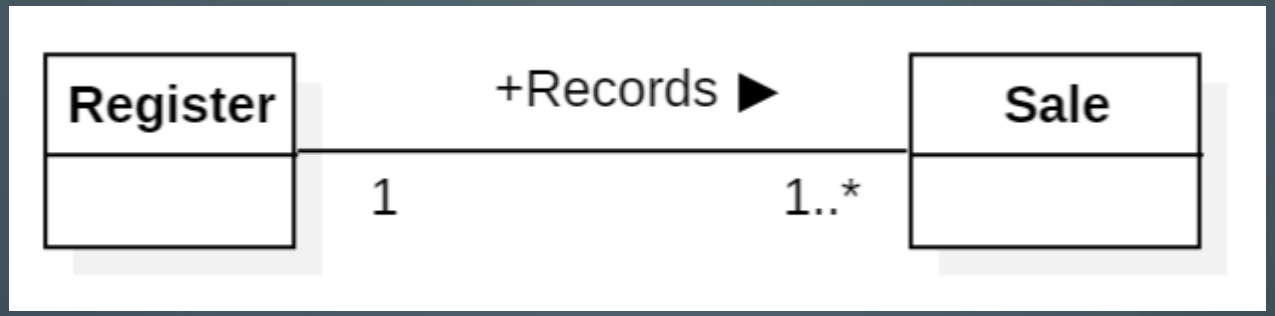
- Fókuszáljunk azokra az asszociációkra, melyek információt kell hordozzanak valamennyi ideig
- A koncepcionális osztályok azonosítása fontosabb mint az asszociációk azonosítása
- A túl sok asszociáció a domain modell érthetőségét / átláthatóságát inkább rontja mint javítja
- Kerüljük a redundáns vagy származtatott asszociációkat

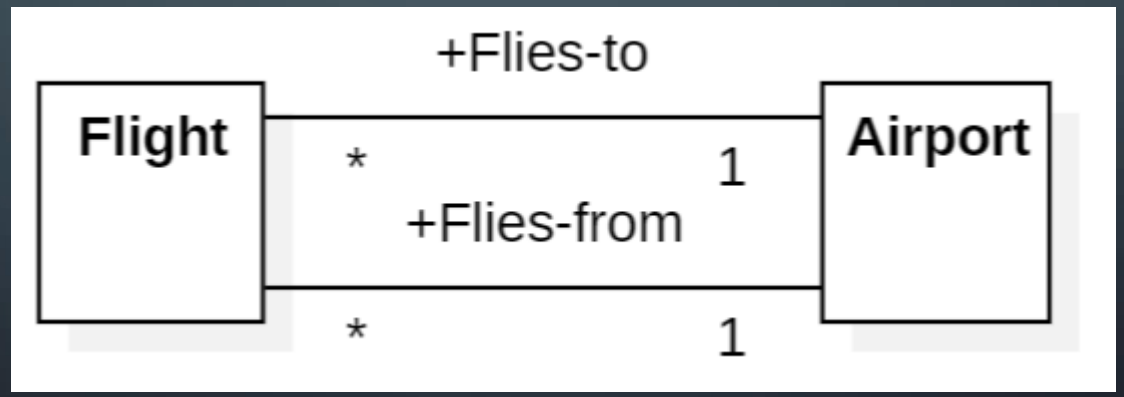
Asszociációk számossága

- *: nulla vagy több
- a..b: legalább a és legfeljebb b számosságú
- a..*: legalább a
- 0..b: legfeljebb b

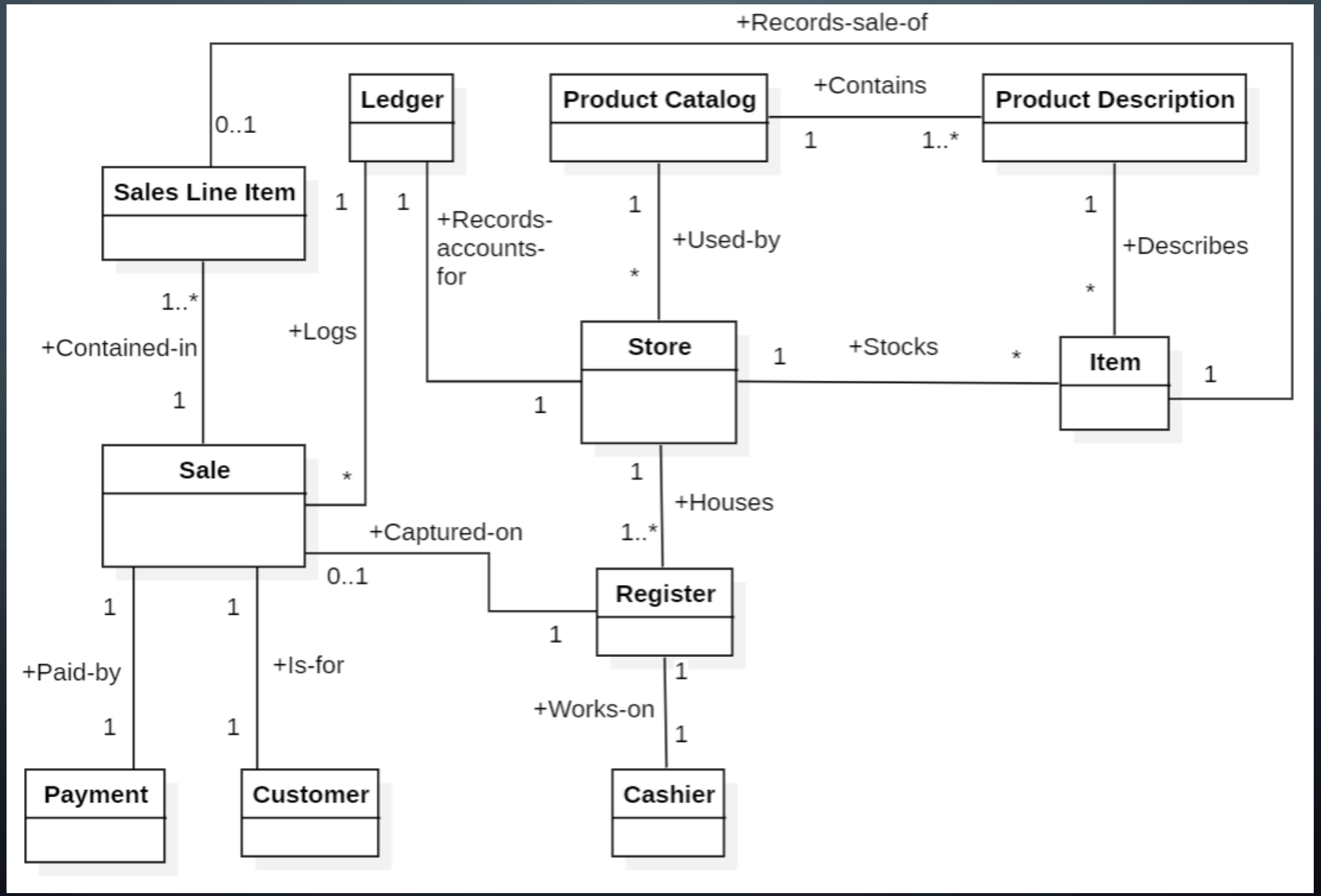


Asszociációk számossága: példák





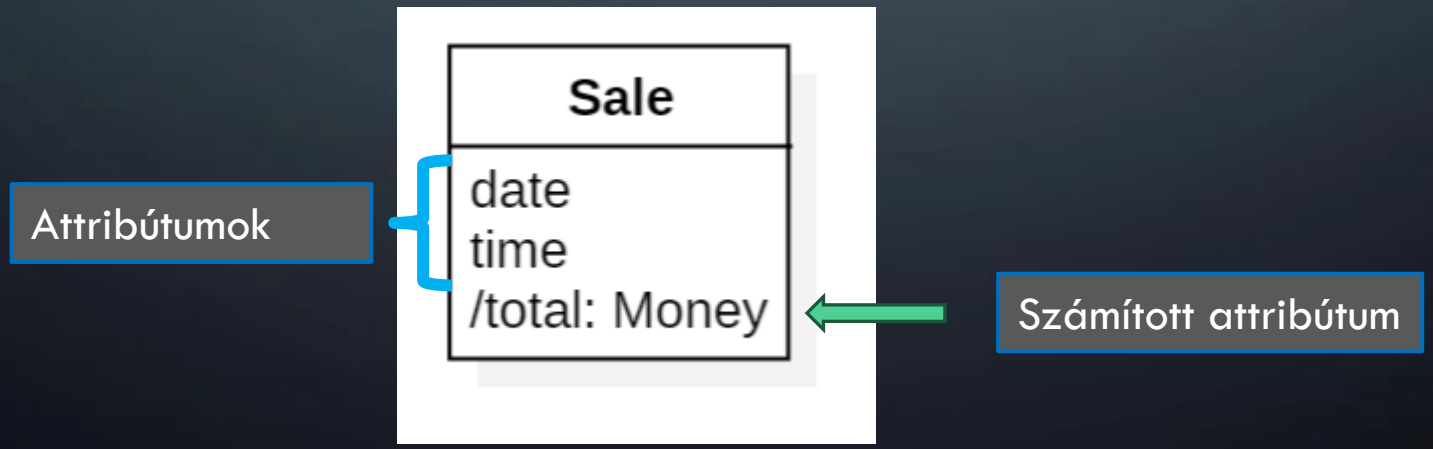
Példa: koncepcionális osztályok asszociációkkal



III. ATTRIBÚTUMOK HOZZÁADÁSA

Alapelvek

- Csak a követelmények szerint tárolandó információkat tartalmazza
- Tipikusan egyszerű adat típusú (szöveg, szám, logikai, dátum, idő)
- Gyakorlati példa: cím, szín, alak, telefonszám, személyi szám, vonalkód, irányítószám, felsorolható típusok
- Ha bizonytalanok vagyunk hogy osztályt vagy attribútumot rendeljünk valamihez, akkor inkább osztályt
- Külső kulcsot soha ne használjunk attribútumként

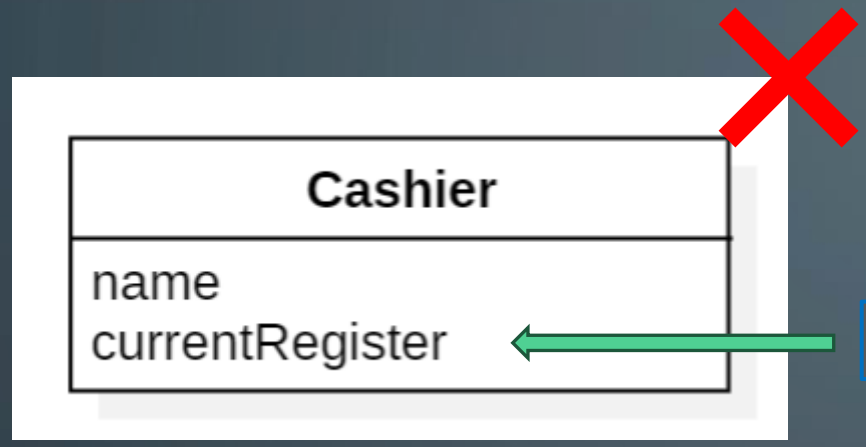


Hogyan vesszük észre, hogy amit attribútumként azonosítottunk osztálynak kellene lennie?

- Ha szétválasztható részekből áll (név (vezeték és kereszt), telefonszám (ország, körzet, helyi))
- Ha műveletek kapcsolódnak hozzá (személyi szám valódiságának ellenőrzése)
- Ha vannak további attribútumai (akciós ár kezdő és vég dátumai)
- Van mértékegysége (összeg, pénznem)
- Összetett kód (vonalkód: ország, gyártó, …)

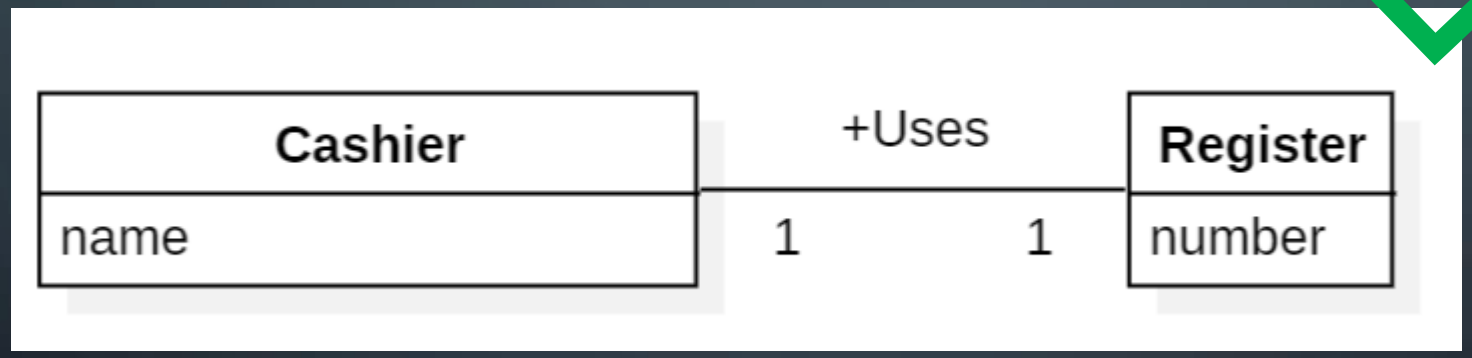
- Függhet a tervezendő szoftver céljától és jellegétől
  - Pl. személy neve: kormányzati nyilvántartó szoftver vagy szimpla regisztráció egy eseményre

Példa: attribútum vagy osztály

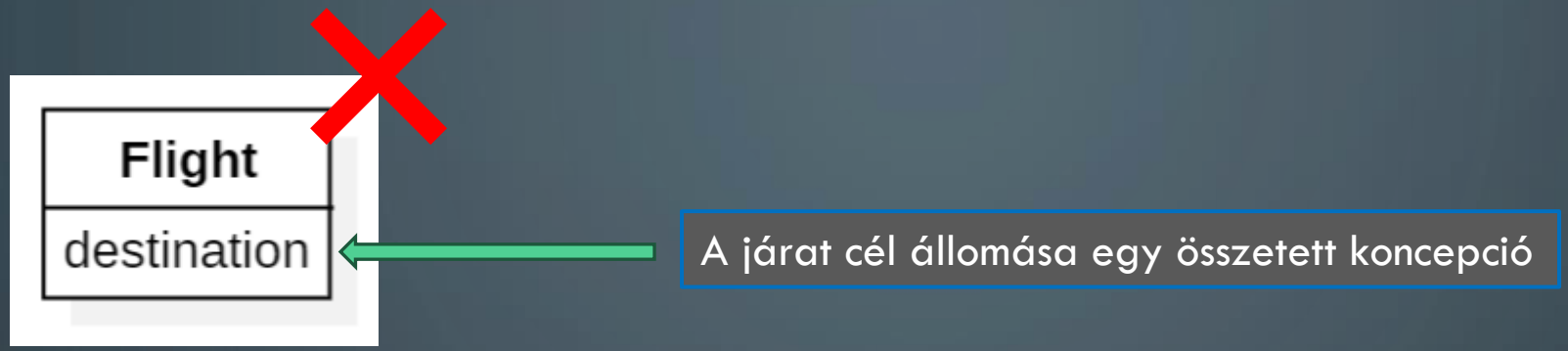


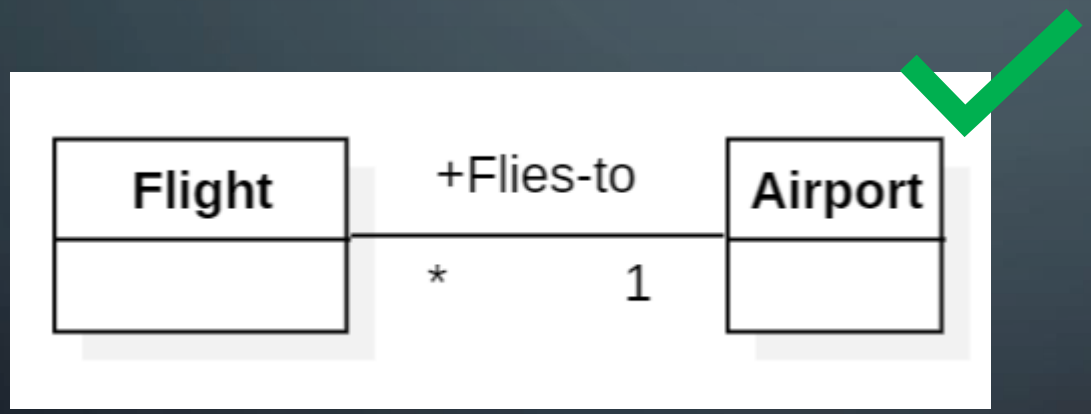
A domain modell nem írja le, hogy az asszociációkat a kódban hogy valósítjuk meg.

Nem egyszerű adat típus



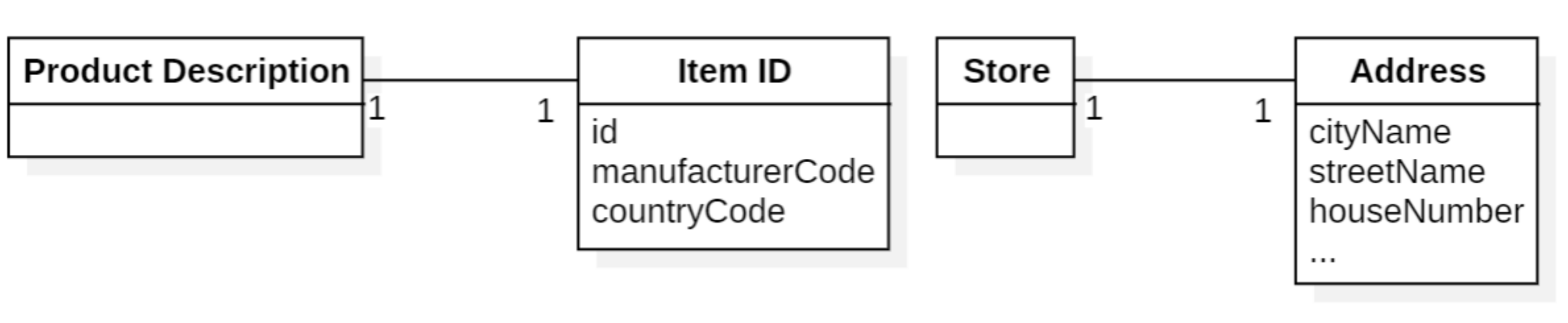
Példa: attribútum vagy osztály



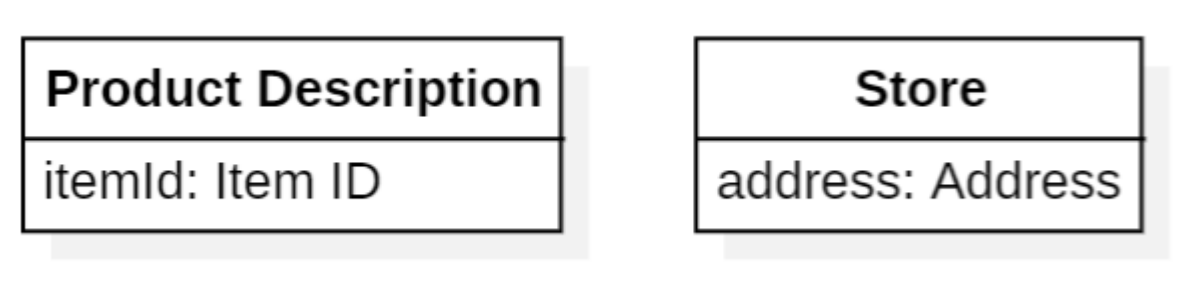


Attribútum vagy osztály

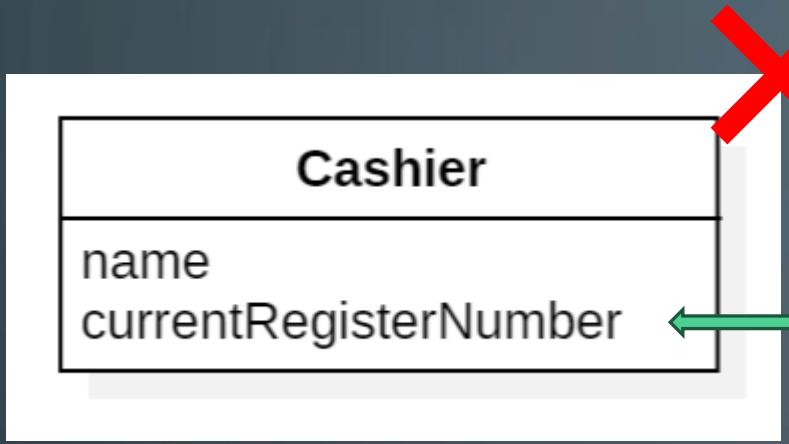
- Ha a kapcsolat egyértelműen tartalmazás jellegű, és a tartalmazott objektum csak egyszerűen adatot tárol, akkor jelölhető tartalmazott adatként, de a típust jelölni kell



Vagy

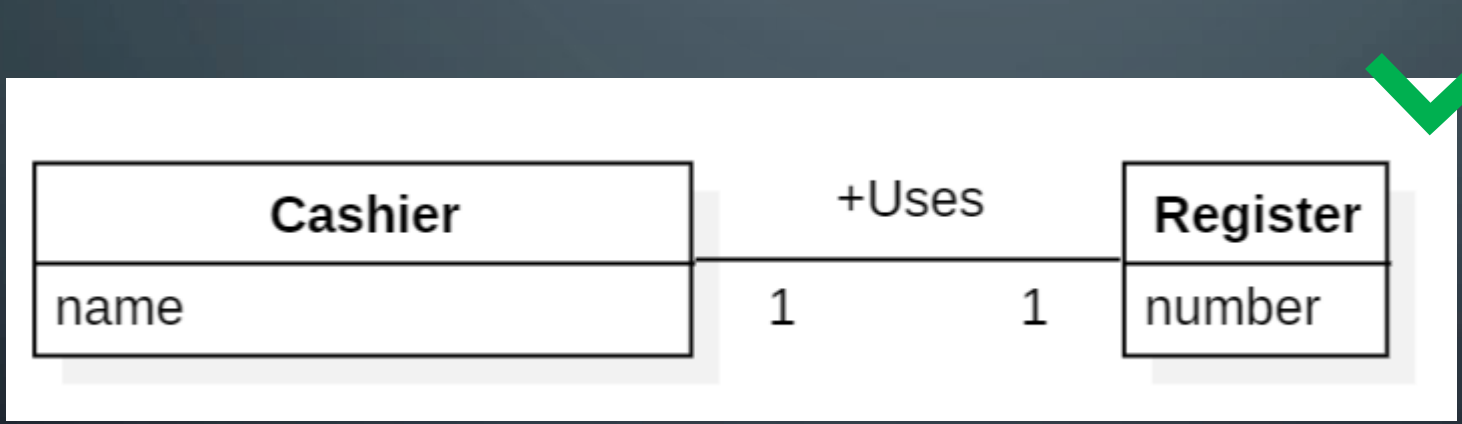


Példa: külső kulcs

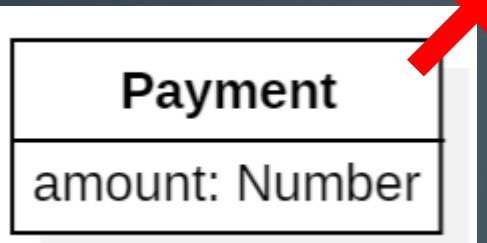


A közvetett hivatkozás szintén olyan részlet, amelyet a domain modell nem tartalmaz

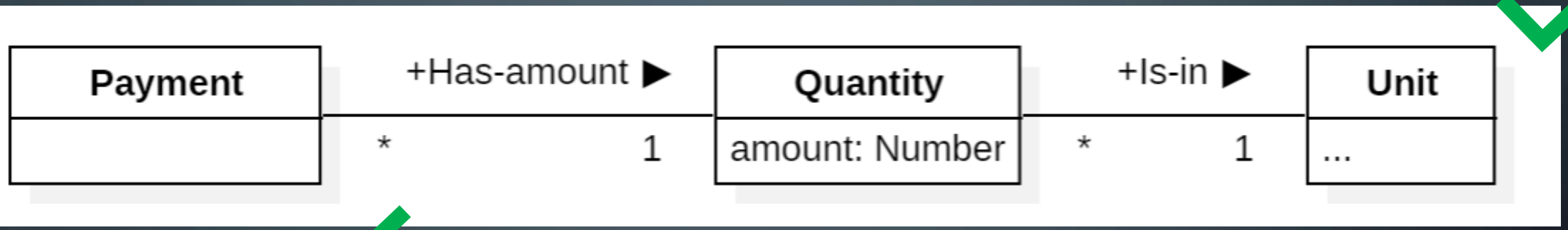
Egyszerű adat (egész szám), de külső kulcsként használjuk egy másik objektum azonosítására.

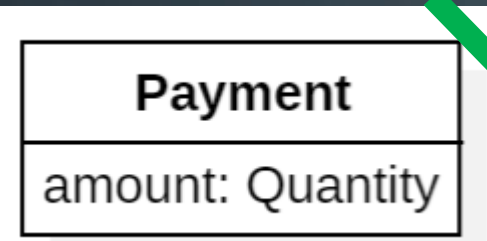


Példa: attribútum típusa

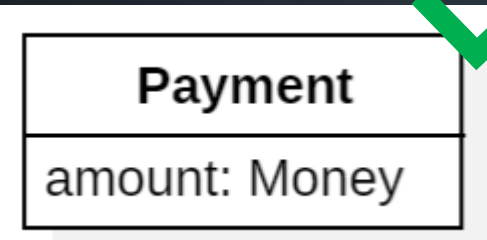


Egy szám ide nem elég, a használt pénznem is fontos infó



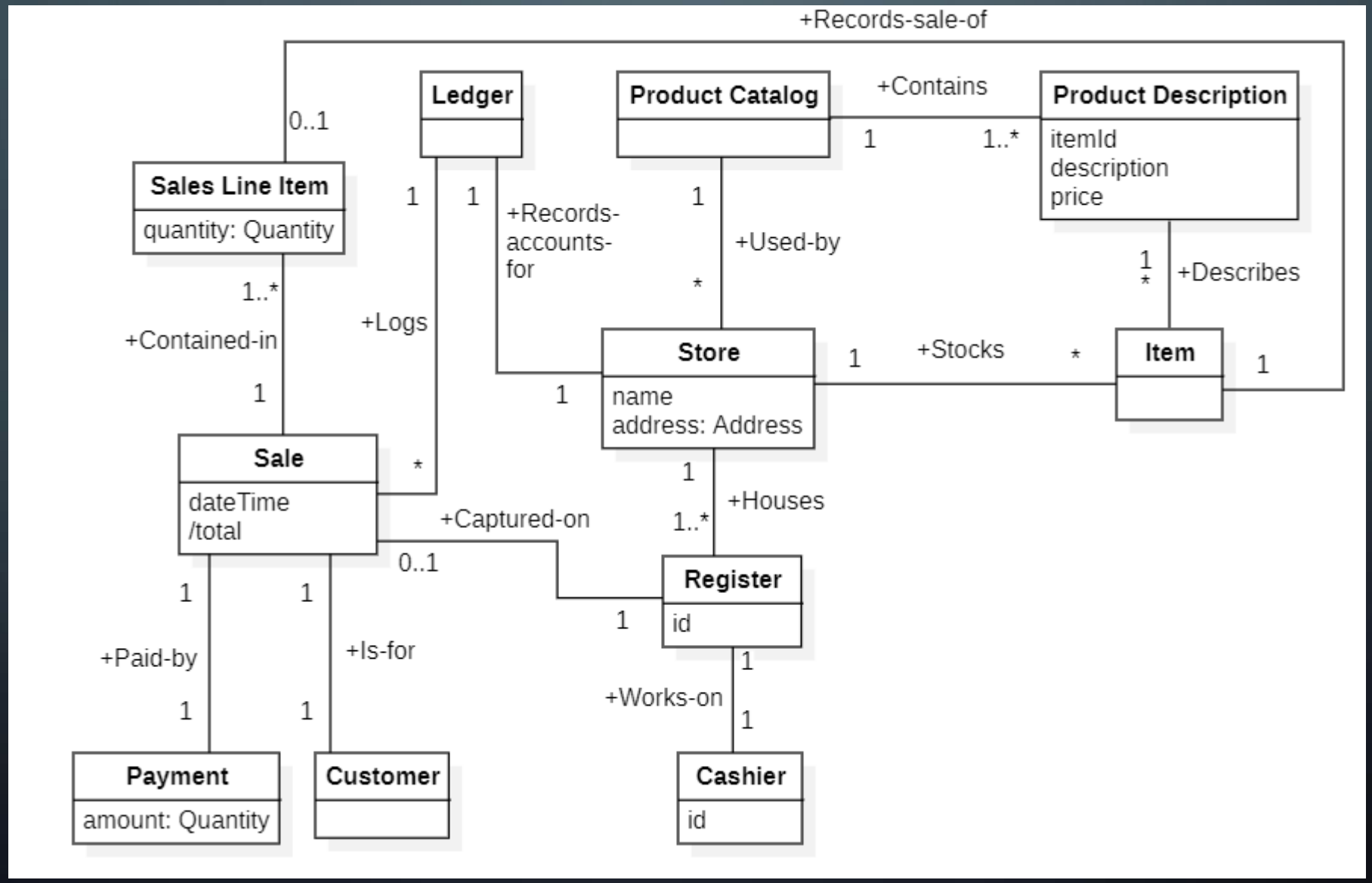


A mennyiség koncepciója magában hordozza a mértékegységet



A pénz lehet a mennyiség egy speciális formája, amiben a mértékegység mindig valamilyen pénznem

Domain modell (osztályok, asszociációk, attribútumok)



IV. FELELŐSSÉGEK HOZZÁRENDELÉSE

GRASP: Objektumok tervezése felelősségekkel

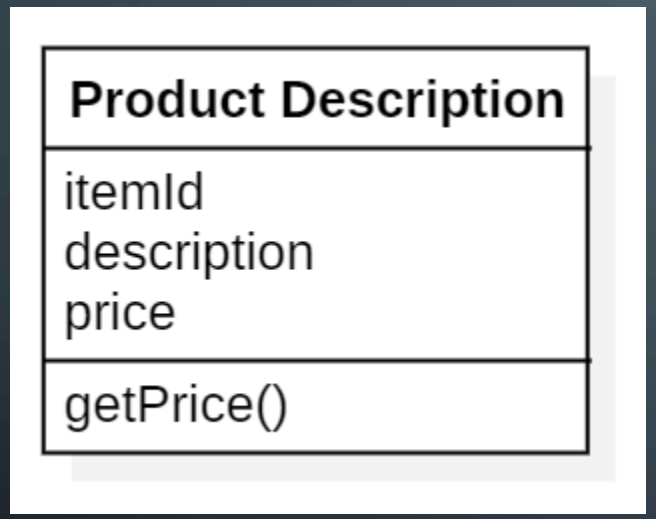
- GRASP Tervezési minták
  - Hogyan rendeljünk osztályokhoz felelősségeket
- Célja:
  - Segítség objektumok megértéséhez
  - Indokok és
  - Alapelvek felelősségek hozzárendeléséhez

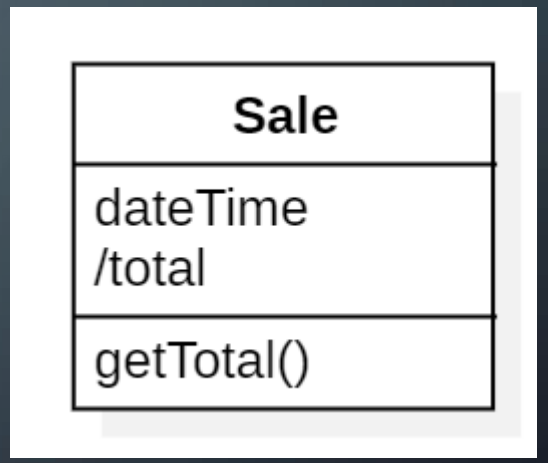
Felelősségek és Metódusok

- Felelősségek típusai:
  - Tudás típusú
    - Tudja a saját privát attribútumait
    - Ismeri a kapcsoló objektumokat
    - Ismeret biztosít olyan információkról, melyek a saját ismeretei szerint kiszámíthatóak vagy levezethetőek
  - Cselekvés típusú
    - A objektum maga végez egy feladatot (létrehoz másik objektumot vagy végrehajt egy számítást)
    - Másik objektumban kezdeményez cselekvést
    - Irányítja vagy koordinálja más objektumok cselekvését

Felelősségeknek a megvalósítása

- Külvilág számára hozzáférhető metódus segítségével





Felelősségek meghatározása során használt jelölések

- Mivel ábrázoljuk
  - Osztálydiagramok
  - Együttműködési diagramok
- Mit ábrázolunk
  - Lehetséges megvalósítások
  - Döntés eredményeként a konkrét megvalósításokat

GRASP: General Responsibility Assignment Software Patterns (Általános alapelveket adó minták felelősségek hozzárendeléséhez)

- Szakértő (Expert)
- Létrehozó (Creator)
- Magas kohézió (High cohesion)
- Gyenge láncolás (Low coupling)
- Vezérlő (Controller)

1. Szakértő (Expert)

- Irányelv
  - Ahhoz az osztályhoz rendeljük a felelősséget, ami elég információval rendelkezik a végrehajtáshoz
- Probléma amire megoldást javasol:
  - Egy alkalmazásban több száz vagy ezer felelősség, amiket osztályokhoz kell rendelnünk.
- Ha jól csináljuk, akkor
  - Könnyű érthetőség
  - Karbantartani,
  - Bővíteni
  - Újrahasznosítani
- Ezért nagy felelősség

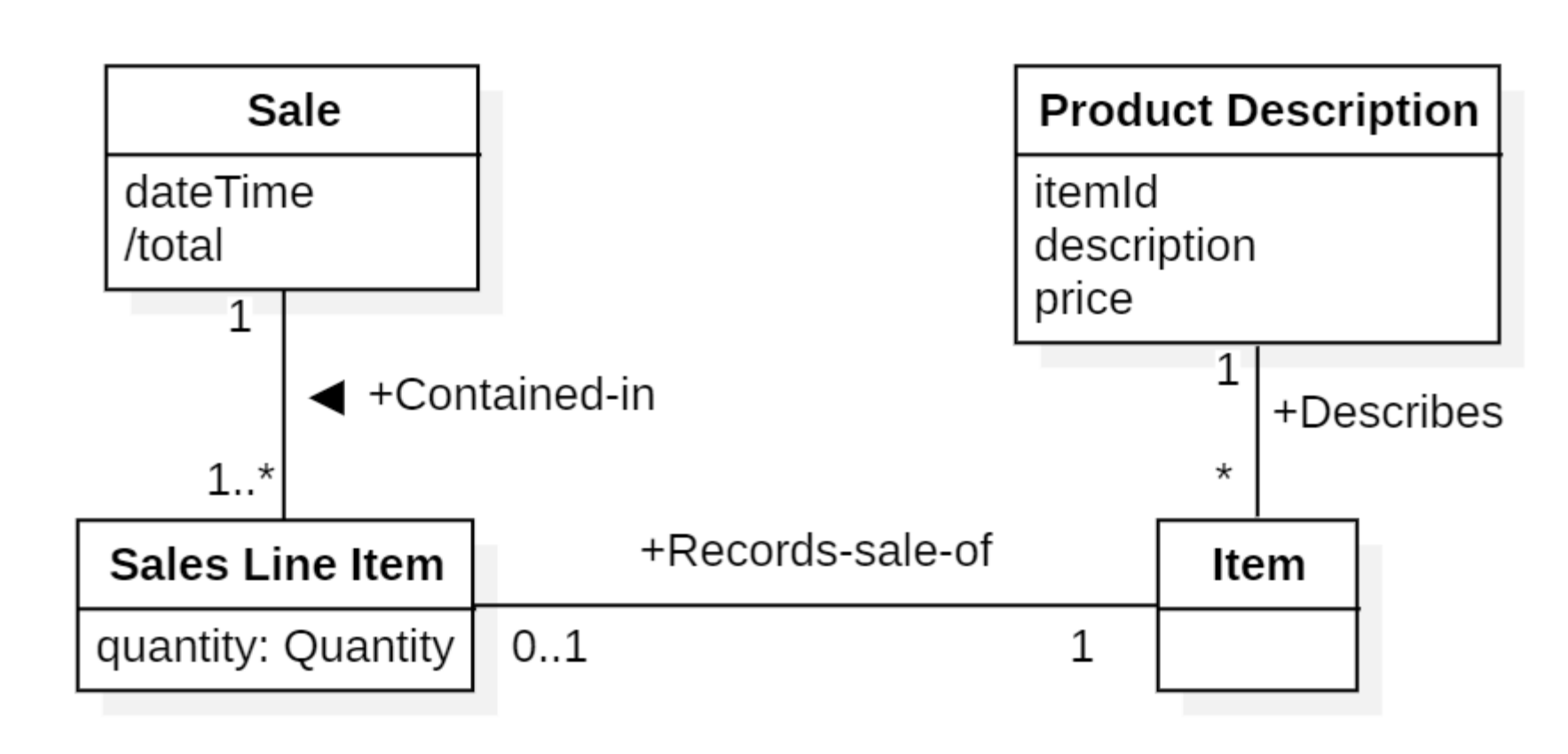
1. Szakértő (Expert)

- Hol keressük ezt az osztályt?
  - Elsősorban a domain modellben
  - Másodsorban a design modellben

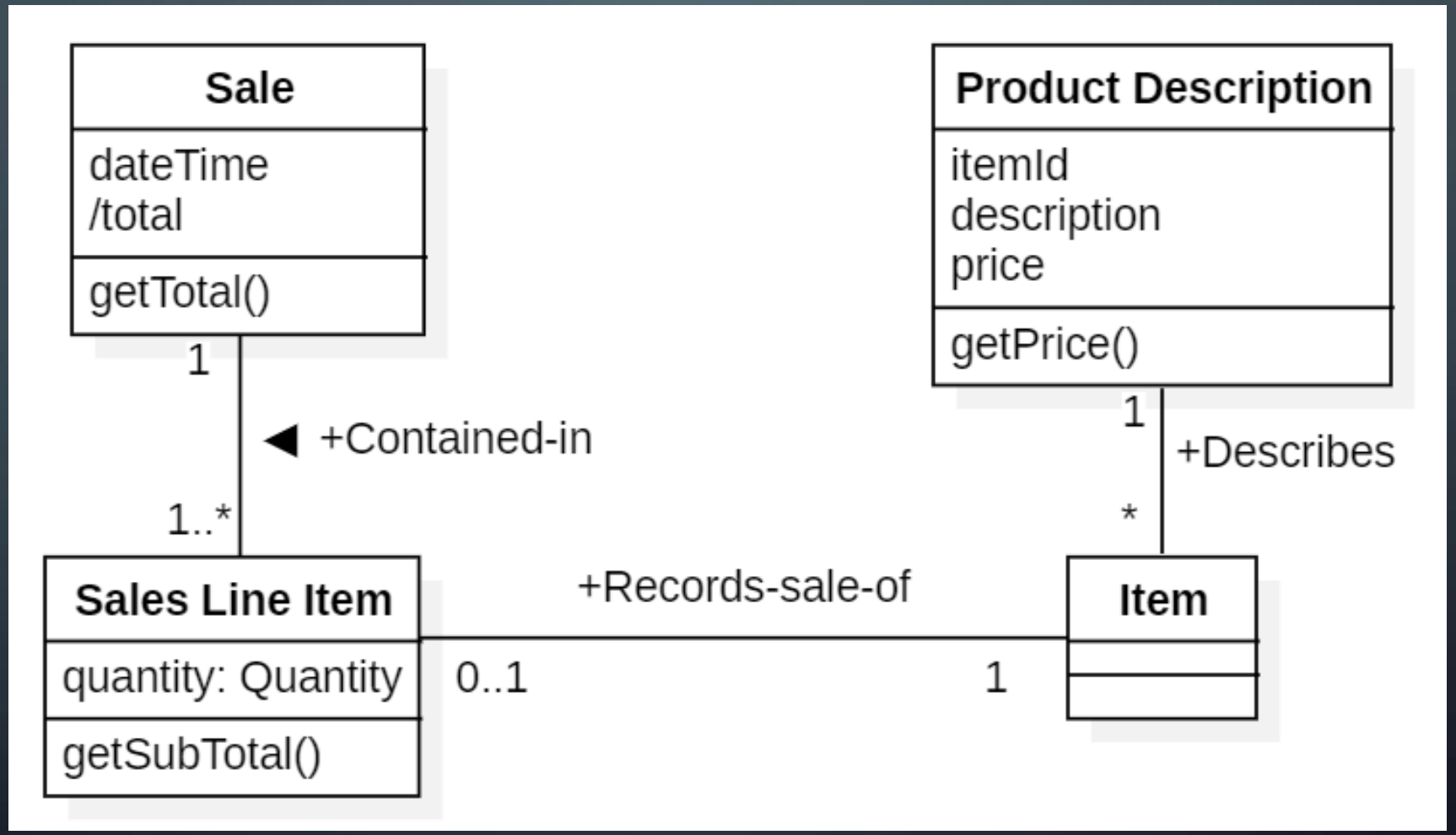
1. Szakértő (Expert): Példa

- Ki a felelőse egy számla végösszegének meghatározásának?
  - Ami a teljes információval kapcsolatban van:
    - Sale
- Ugyanakkor ismernie kell a részösszegeket.
  - Ugyanez az alapelv:
  - Részösszeg kiszámítása:
    - Sales Line Item
- Ugyanekkor ismerni kell a termék árát.
  - Ismét Szakértő minta szerint:
  - Termék árát az mondja meg aki tudja:
    - Product Description

1. Szakértő (Expert): Példa



1. Szakértő (Expert): Példa



1. Szakértő (Expert): Megjegyzés

- A Szakértő minta önmagában sokszor nem elegendő a felelősségek hozzárendeléséhez, számításba kell vennünk a
  - gyenge láncolás és a
  - magas kohéziószempontjait is.

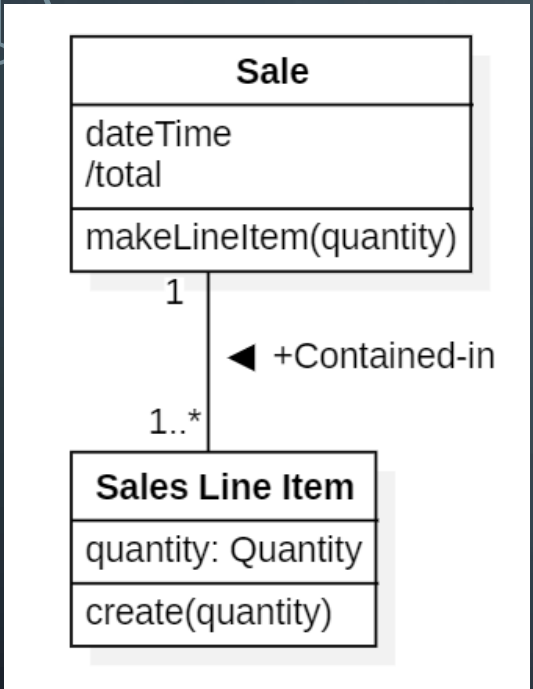
2. Létrehozó (Creator)

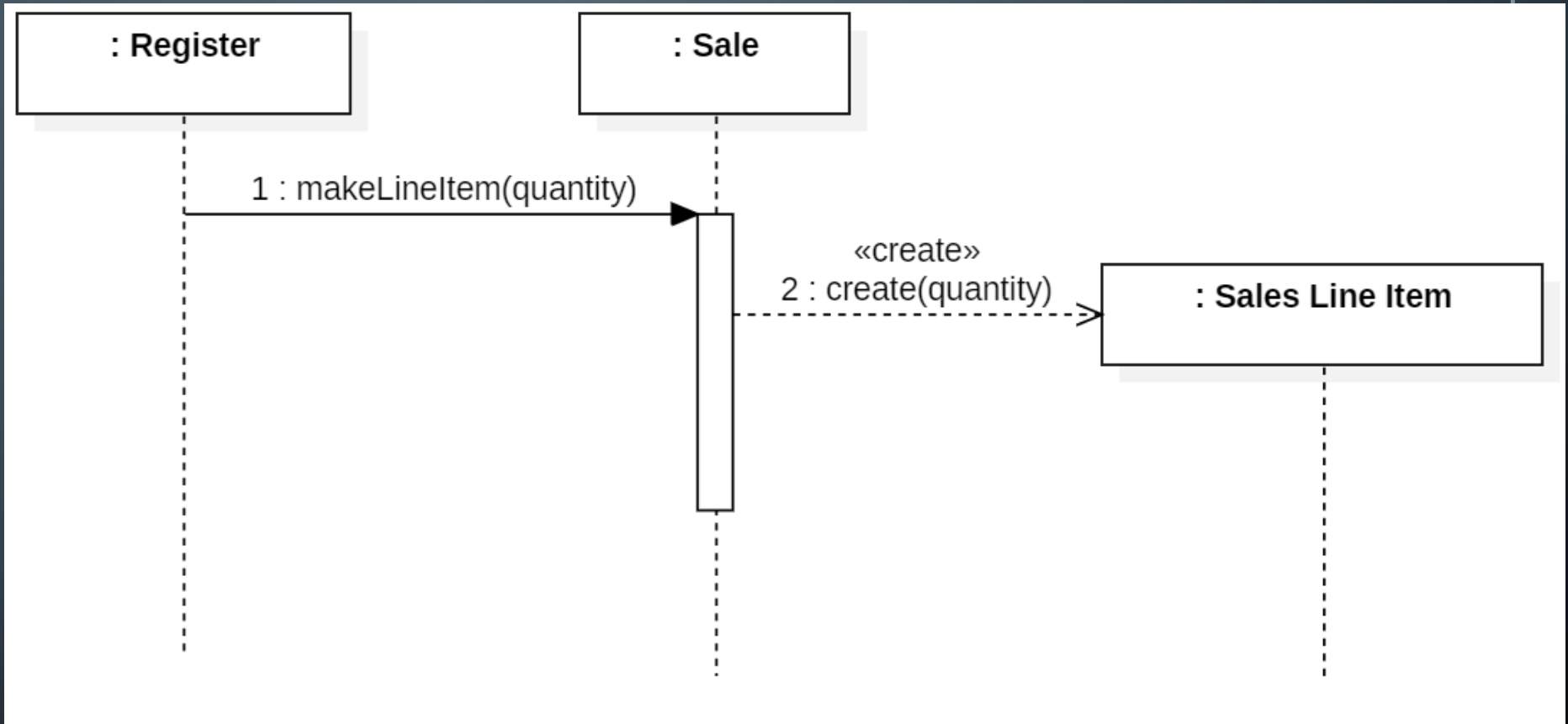
- Irányelv
  - Rendeljük hozzá egy B osztályhoz annak felelősségét, hogy A osztálybeli objektumokat létrehozzon, ha egy vagy több az alábbiak közül teljesül:
    - B tartja nyilván A előfordulásait
    - B tartalmazza A-t
    - B tartalmazza A helyét
    - B közvetlen felhasználója A-nak
    - B rendelkezik azzal az információval, ami A létrehozásához (inicializálásához) szükséges
  - Ha több osztályra is igaz a fentiek valamelyike, akkor a tartalmazásnak és cím tartalmazásának van prioritása

2. Létrehozó (Creator): Példa

- Kinek a felelőssége egy Sales Line Item létrehozása?
  - Mivel Sale tartalmaz számos Sales Line Item objektumot, így ő hozza létre

2. Létrehozó (Creator): Példa





2. Létrehozó (Creator): Ellentmondások feloldása

- Újrahasznosítás gyakorlati megfontolása miatt, létrehozhatunk osztályt a hasonló objektumok kezelésére
  - Objektum gyár (object factory) minta.

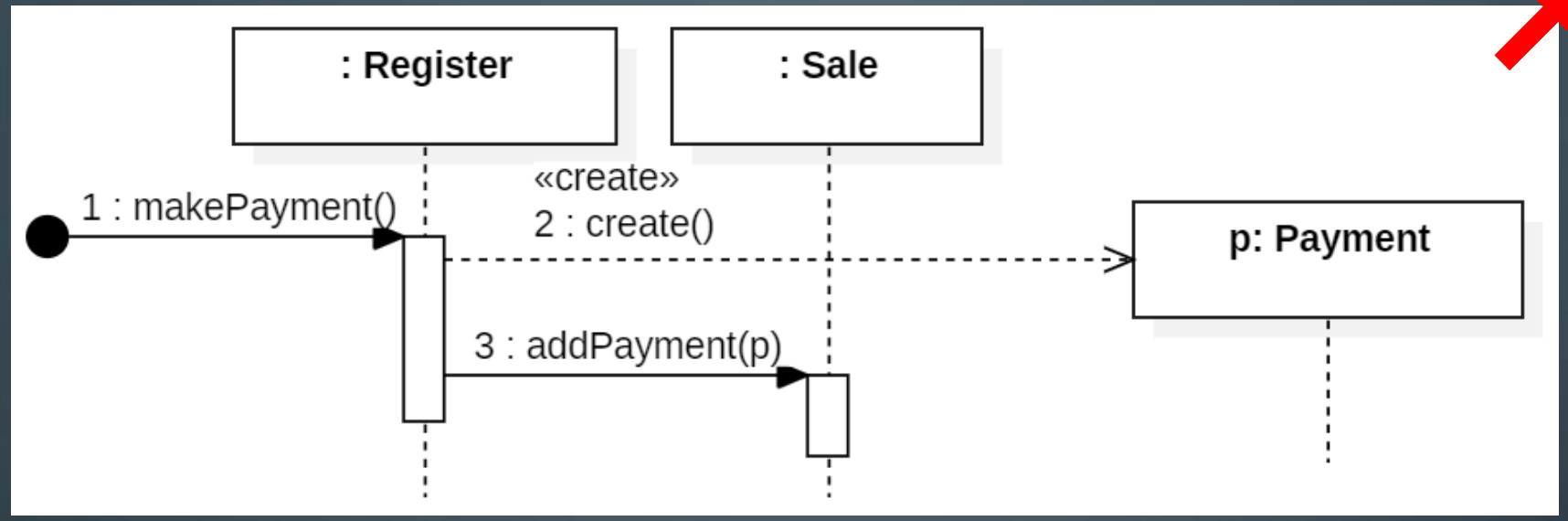
3. Gyenge láncolás (Low coupling)

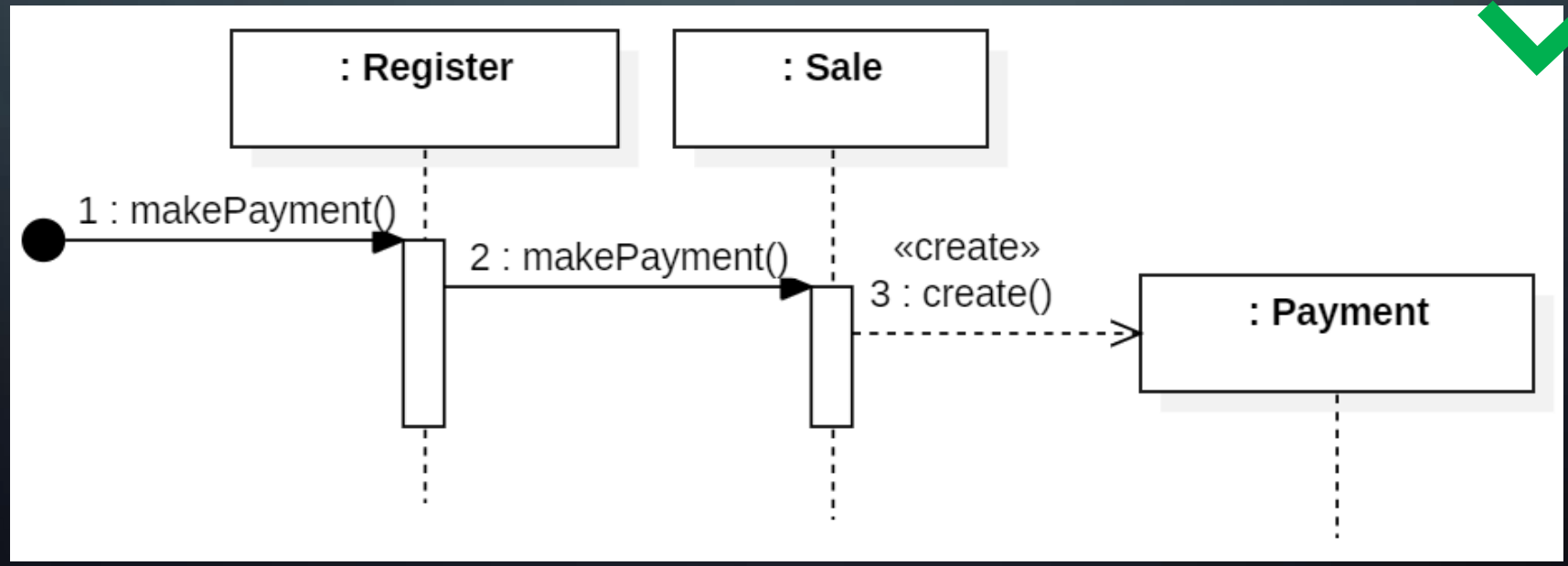
- Irányelv
  - A felelősségeket a láncolások minimalizálásával osszuk ki.
- Probléma amit megold
  - Hogyan biztosítjuk az osztályok közti összefüggések minimalizálását, ezzel növelve az újrahasznosítás lehetőségét és minimalizálva a változások következményeit?

3. Gyenge láncolás (Low coupling): Példa

- Kinek a feladata (felelőssége) a Payment létrehozása
  - Ha Register hozza létre: két másik osztálynak küld üzenetet,
  - Ha Sale osztály hozza létre, minden osztály csak egy másiknak küld üzenetet
  - Utóbbi esetben a láncolások száma kisebb (kedvezőbb megvalósítás)

3. Gyenge láncolás (Low coupling): Példa





Láncolás Programozási Nyelvekben

- X és Y láncolásban áll, ha
  - X-nek van Y típusú vagy Y típusra hivatkozó attribútuma
  - X típusú objektum Y típusú objektum valamely szolgáltatását használja
  - X-ben szerepel olyan metódus, mely hívásához Y típusú vagy Y-ra hivatkozó paraméter szükséges
  - X közvetlen vagy közvetett részosztálya Y-nak
  - Y egy interfész, amit X megvalósít

4. Magas kohézió (High Cohesion)

- Irányelv
  - Úgy határozzuk meg a felelősségeket, hogy a kohézió magas legyen.
- Kérdés, amire választ ad
  - Hogyan tartsuk kezelhető szinten a bonyolultságot?
- Kohézió alacsony, ha
  - Egy osztály olyan feladatokat végez, ami nem feltétlenül hozzá tartozik
  - Túl sok feladatot végez

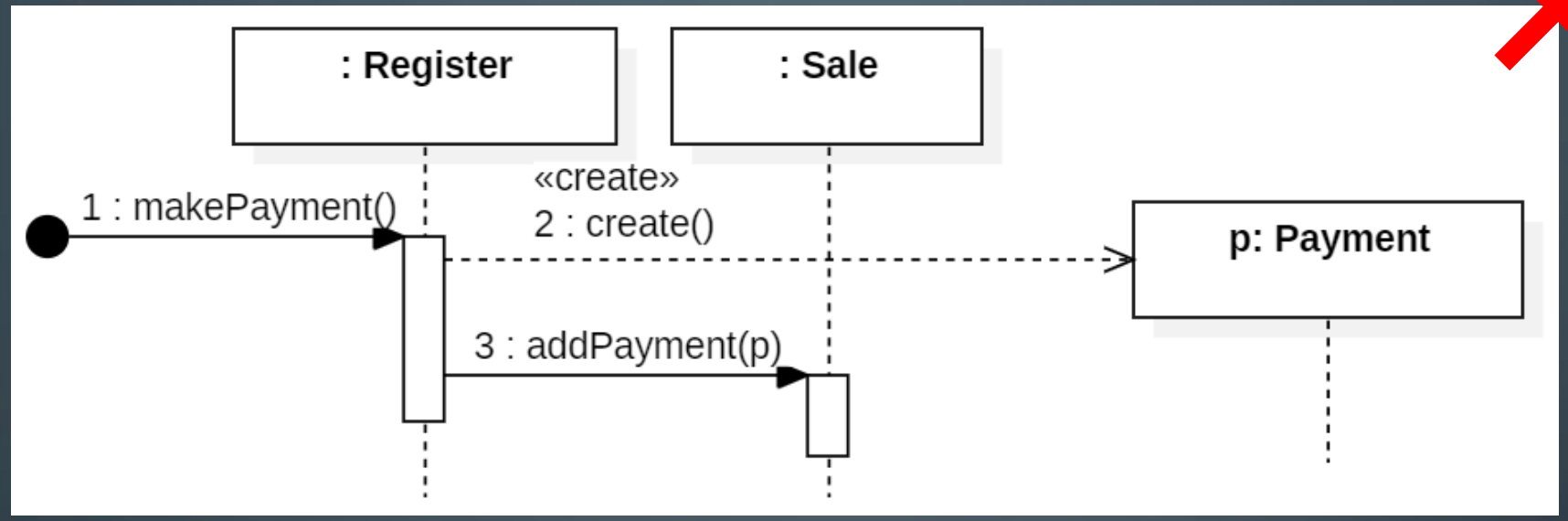
A kohézió szintjei

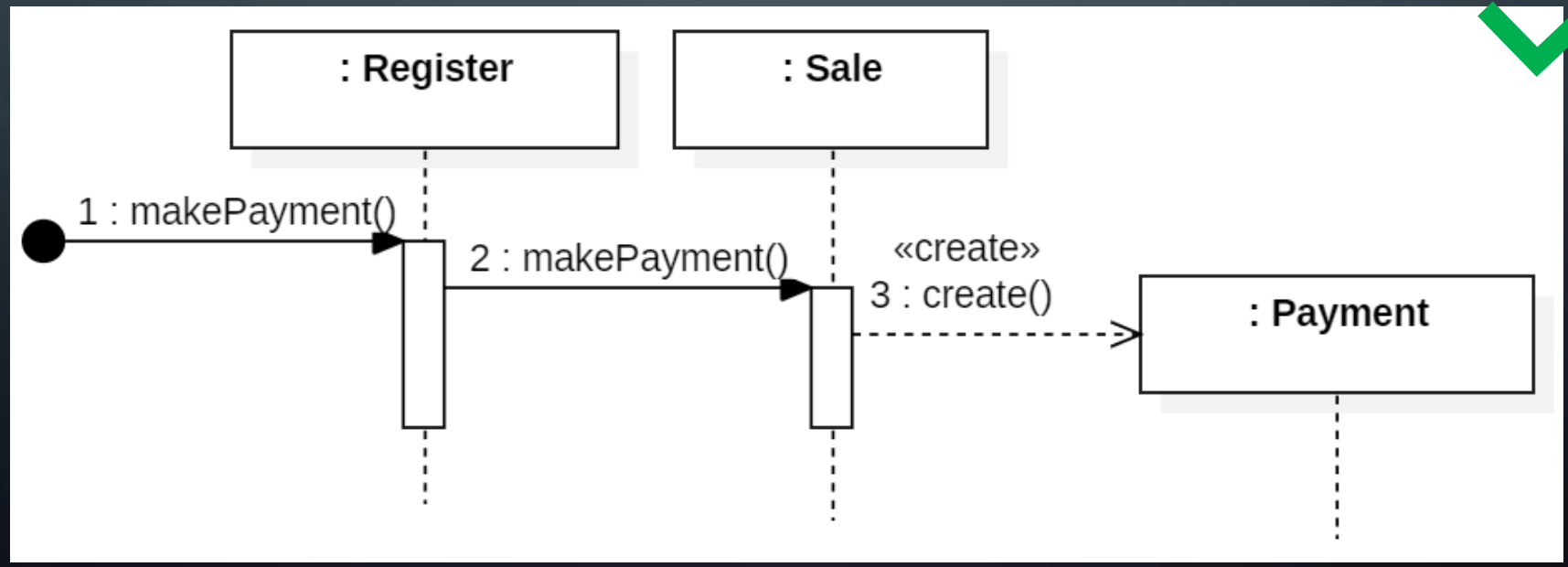
- Kohézió nagyon alacsony
  - Ha egy osztály olyan feladatokat végez, amelyek sok különböző szerepkörhöz tartoznak
- Kohézió alacsony
  - Egy osztály felelőssége egy szerepkörhöz tartozó nagyon sok feladat
- Közepes a kohézió
  - Kevés szerepkörben kevés feladatot végez egy osztály
- Kohézió magas, ha
  - Csak egy szerepkörben és kevés feladatot vállal egy osztály

4. Magas kohézió (High Cohesion): Példa

- Register és Sale feladatai

4. Magas kohézió (High Cohesion): Példa





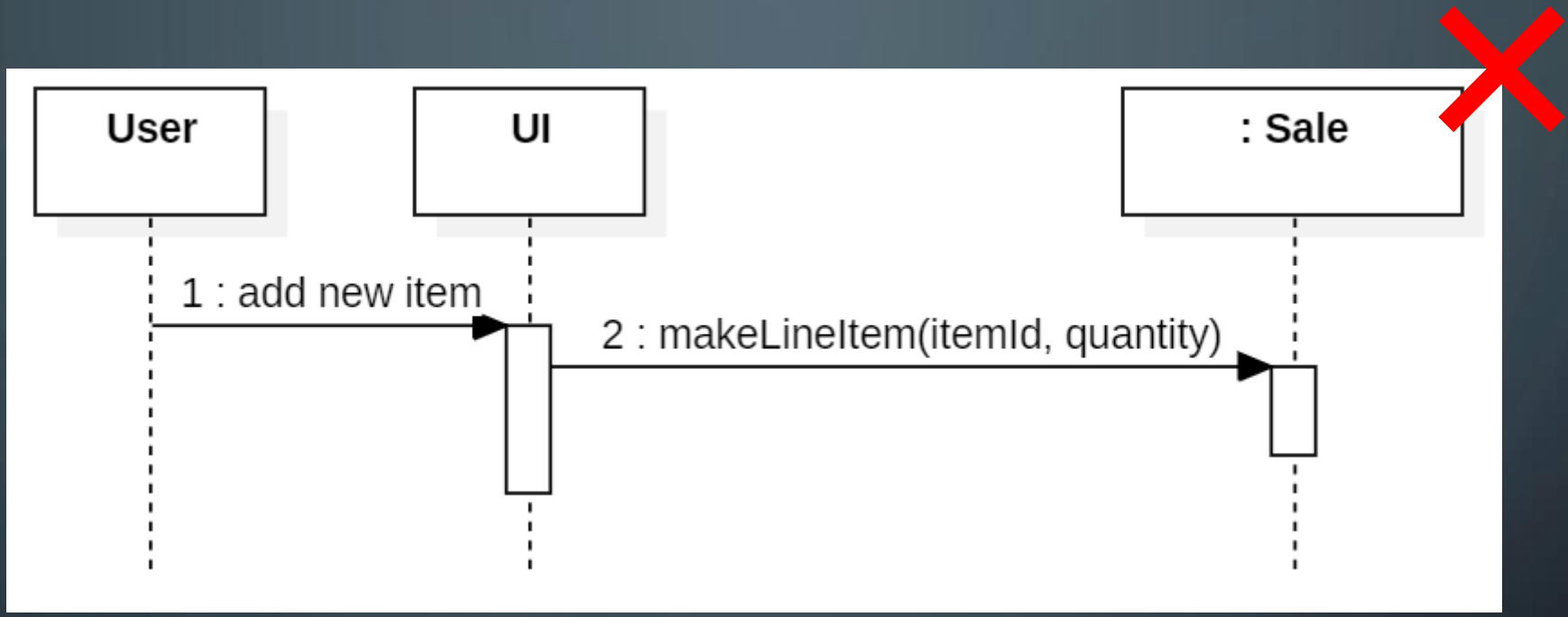
5. Vezérlő (Controller)

- Irányelv
  - A felelősséget a rendszerüzenetek kezelésére rendeljük hozzá a következő osztályok valamelyikéhez
    - Átfogó vezérlő: a rendszerhez, részrendszerhez vagy eszközhöz kapcsolódó teljes üzenetfolyamot ő kezeli
    - Használati esetekben megfogalmazott forgatókönyvekhez rendelt vezérlő
- Kérdés, amit megválaszol
  - Kinek a felelőssége a rendszer felől érkező üzenetek kezelése?
- Rendszer felől érkező üzenetek nem elemi események (egér kattintás), hanem a domain modellben (a modellezett valóságban) értelmezhető üzenetek leképezései.

5. Vezérlő (Controller)

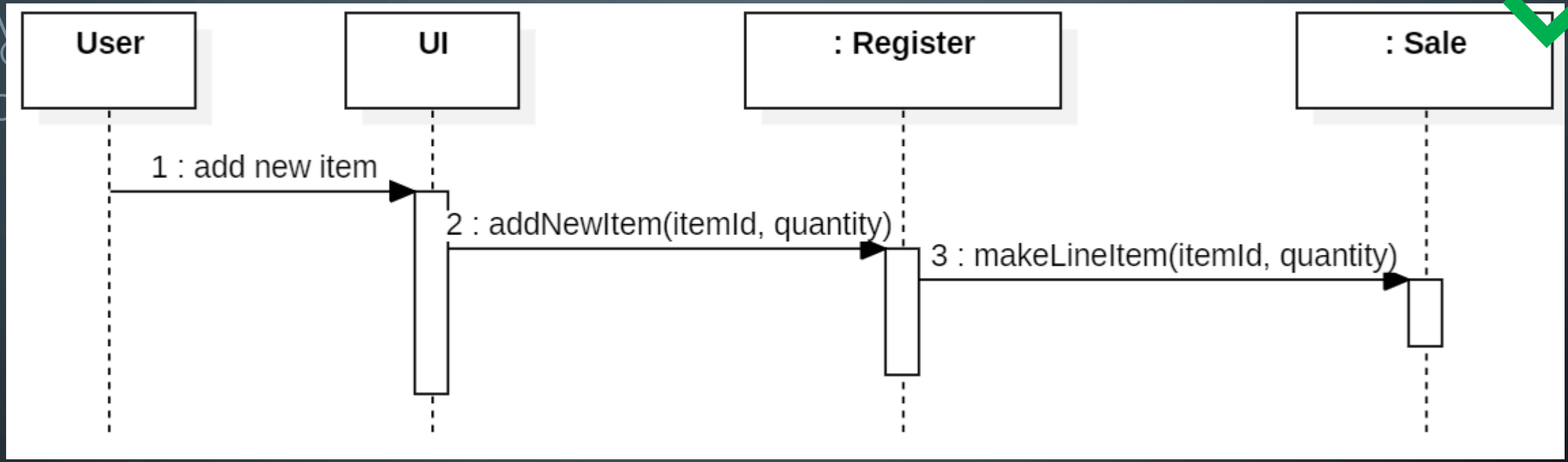
- Fontos megjegyzés
  - A vezérlő nem felhasználói felület (NEM ablak, alkalmazás, nézet, dokumentum) hanem olyan osztály, ami a felületek felől érkező üzeneteket veszi és a feladatokat delegálja a megfelelő osztályoknak.
- Miért válasszuk szét az üzenetkezelést a felhasználói felülettől?
  - Növeljük az újrahasznosíthatóság lehetőségét:
    - Olyan eseménykezelőnk legyen, ami független a felhasználói felület megvalósításától
  - Folyamotok követése egy kézben tartható:
    - pl.: logika miszerint számla tételei csak lezárása előtt rögzíthetőek egy helyen valósul meg

5. Vezérlő (Controller)



Rossz: a felhasználói felület közvetlenül az eladást kezeli (Sale), és annak a metódusait hívja meg

5. Vezérlő (Controller)



Jó: A Register osztály vezérlőként funkcionál, kezeli a felhasználói felületről érkező kéréseket és utasításokat a vásárlás felépítéséhez, és átalakítja azokat többi osztály számára érthető formába

Túlterhelt vezérlők (téves megvalósítás)

- Egyetlen vezérlőnk van, minden rendszerüzenetet az kap, és ilyen üzenetből sokféle van
- A vezérlő az üzenetek kezelésén kívül sok más feladatot önmaga ellát ahelyett, hogy delegálná
- A vezérlő számos attribútumot vagy információt tárol ahelyett, hogy ezt más objektumokra bízná