Unified Modeling Language

Szoftvertechnológia

Dr. Goldschmidt Balázs

BME, IIT

Tartalom

- UML diagramok:
  - Szekvenciadiagram
  - Kommunikációs diagram
  - Interakciós áttekintő diagram

Hol tartunk?

Use Case diagram ← A funkcionális követelmények magas szintű leírása: A use case-ek alapvető interakciós sorozatok a rendszer és a felhasználói között

Aktivitásdiagram ← Use case interakciós sorozatok munkafolyamatként ábrázolva

Komponens-diagram ← Áttekintő kép a rendszer architektúrájáról: A komponensek és kapcsolataik

Hova kell telepíteni a komponenseket → Telepítési diagram

Hol tartunk?

Osztály-diagram ← Egy komponens/rendszer struktúrája objektumorientált modellként

Csomag-diagram ← Csomagok és a közöttük lévő függőségek

Objektum-diagram ← A rendszer részletes állapotának ábrázolása egy adott időpillanatban

Most következik:
Hogyan tervezzük meg egy komponens belsejét?
(Viselkedés)

Hol tartunk?

Most következik:
Hogyan tervezzük meg egy komponens belsejét?
(Viselkedés)

Strukturális UML diagramok:

Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

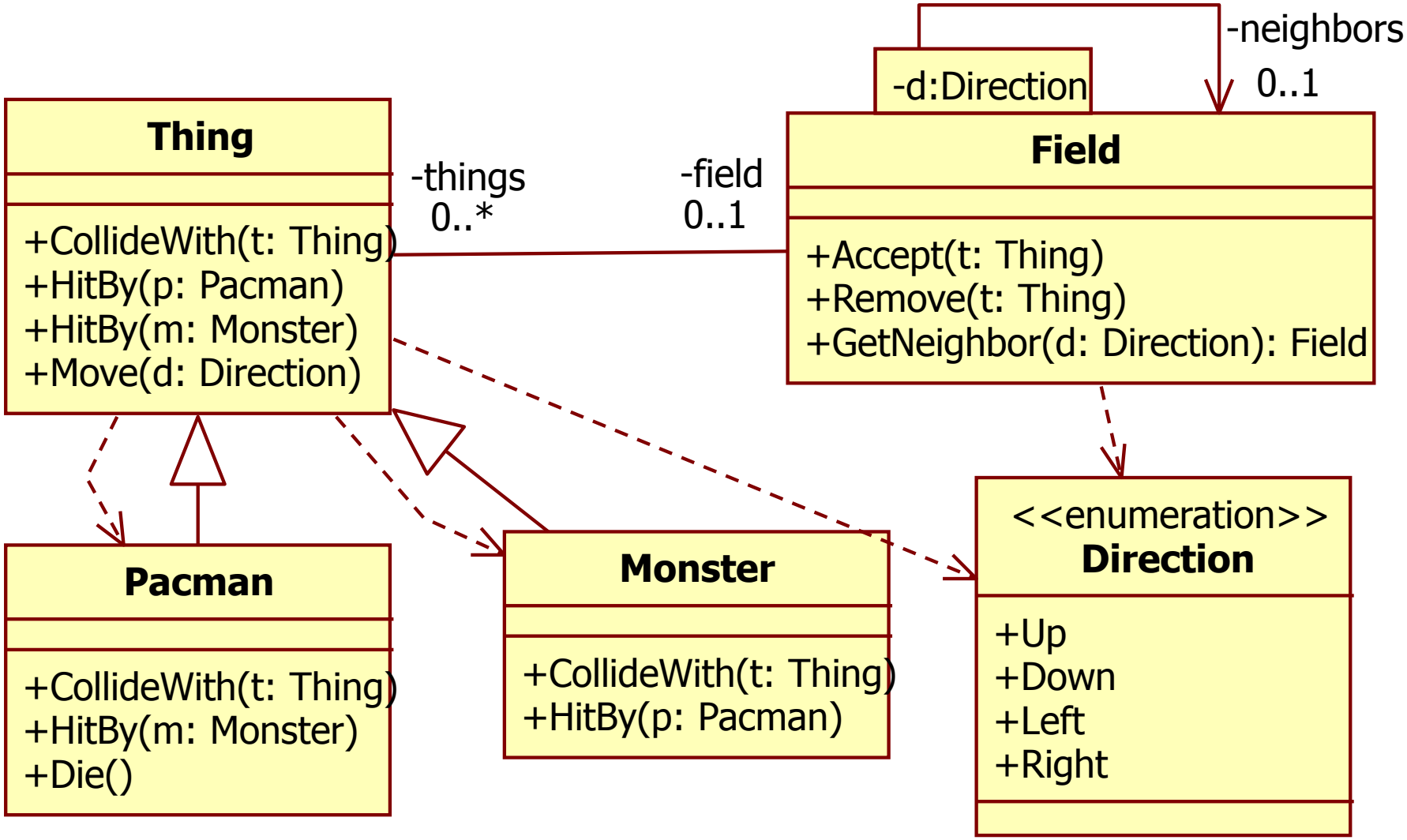
Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakciós áttekintő diagram	

Szekvenciadiagram (Sequence Diagram)

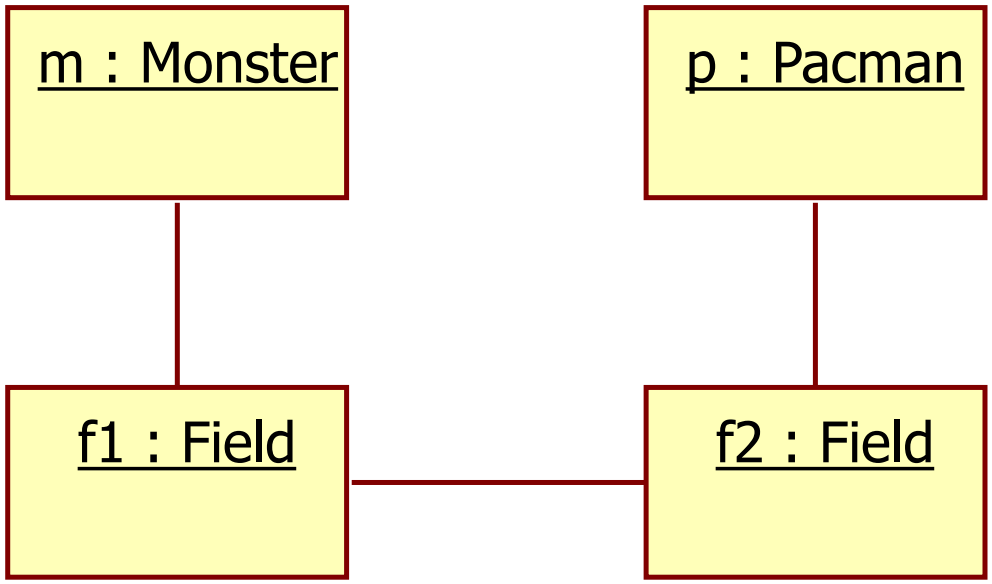
Szekvenciadiagram (Sequence Diagram)

- Interakciók grafikus ábrázolására szolgál
  - a rendszer dinamikus viselkedését mutatja
  - az interakciók a részvevők közötti információcserére fókuszálnak
- A szekvenciadiagram használható use case forgatókönyvek leírására, metódusok belső logikájának definiálására és egy protokollban történő üzenetváltások ábrázolására
- Egy szekvenciadiagram üzenetváltások egy lehetséges időbeli lefutását mutatja
  - egyszerű futássorozatok vannak ábrázolva
  - le lehet írni helyes és helytelen lefutásokat is
  - a nem ábrázolt lefutásokról nem mindig dönthető el egyértelműen, hogy helyesek vagy helytelenek

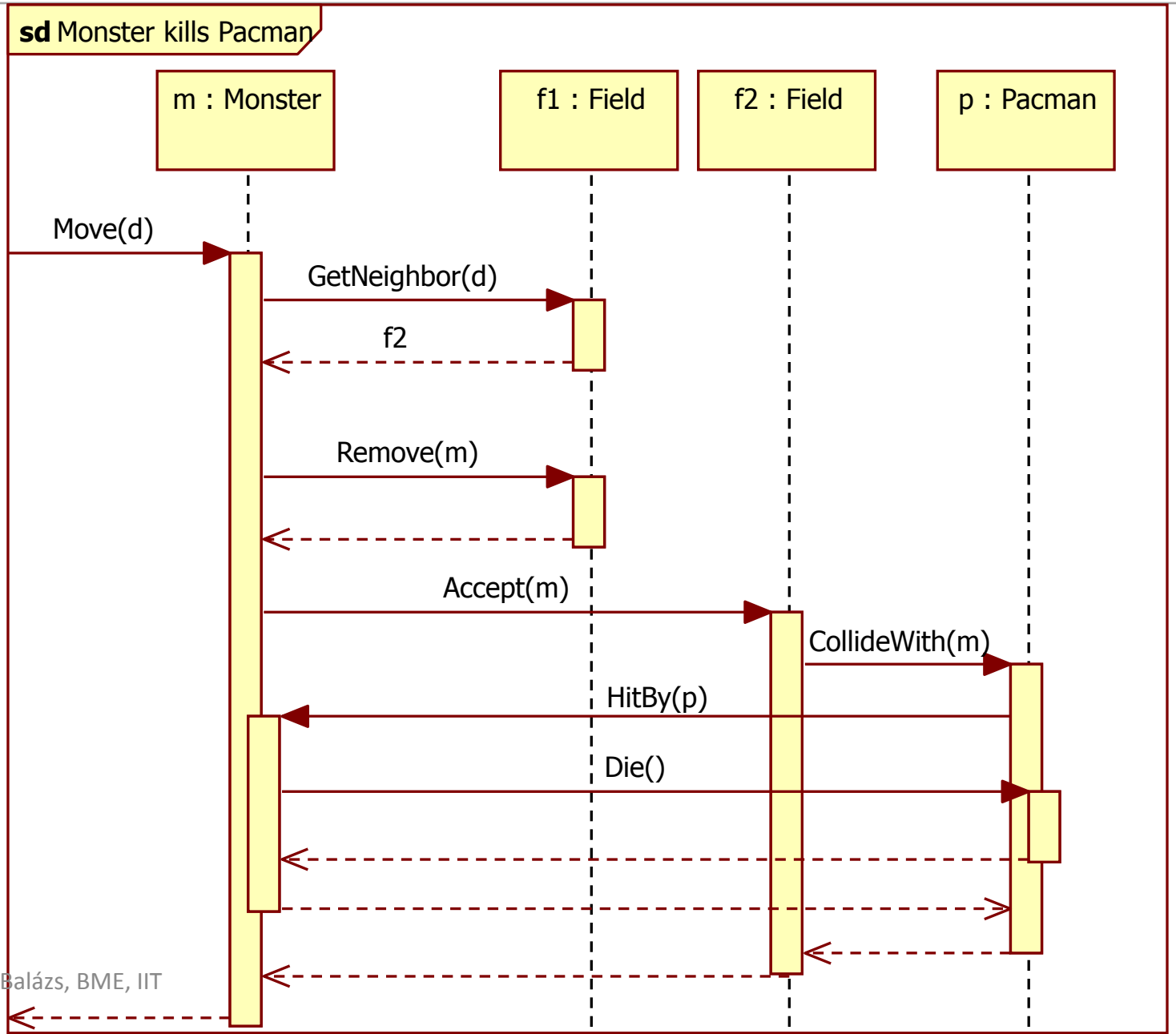
Pacman: osztálydiagram részlet



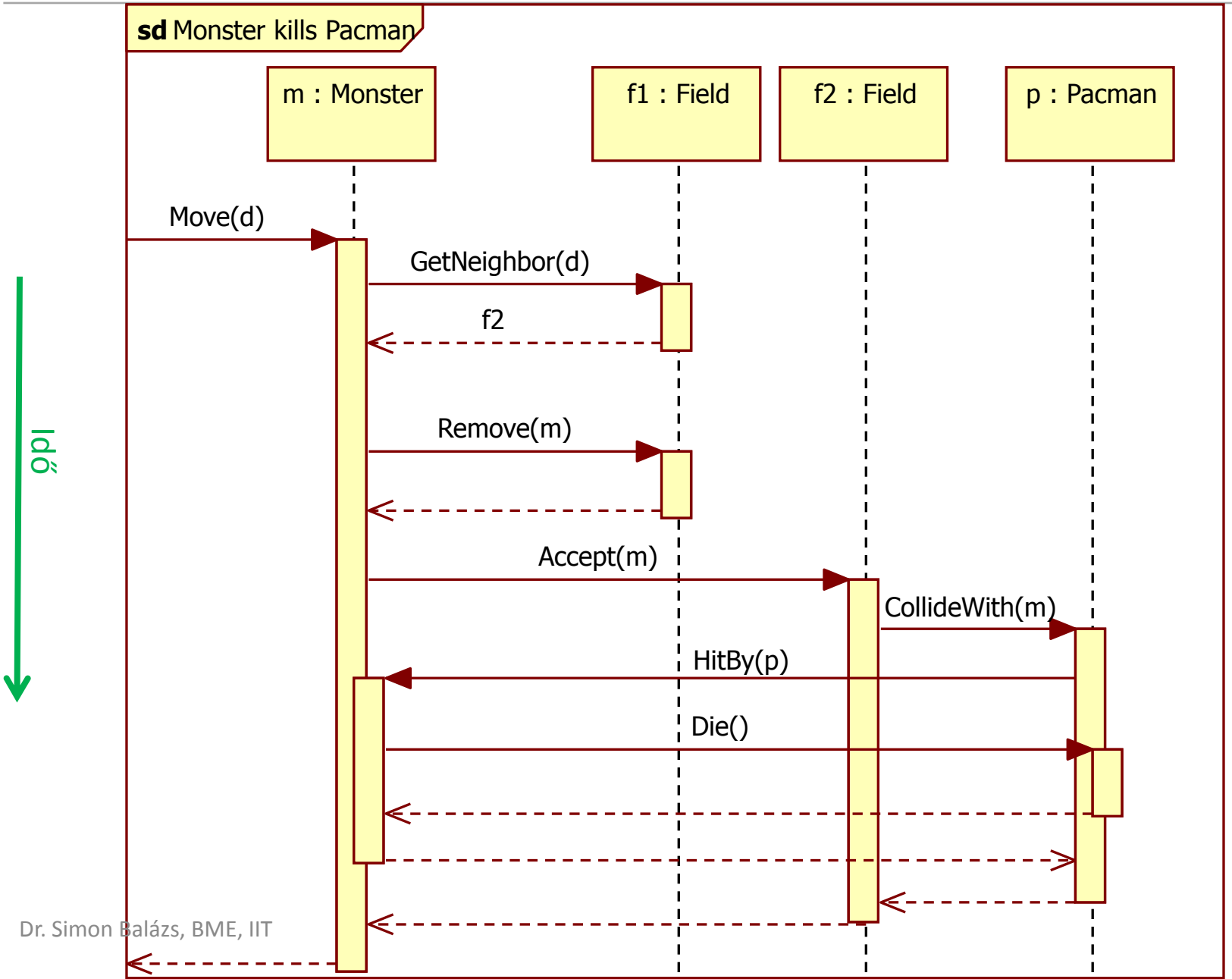
Pacman: objektumdiagram egy lehetséges kezdőállapotra



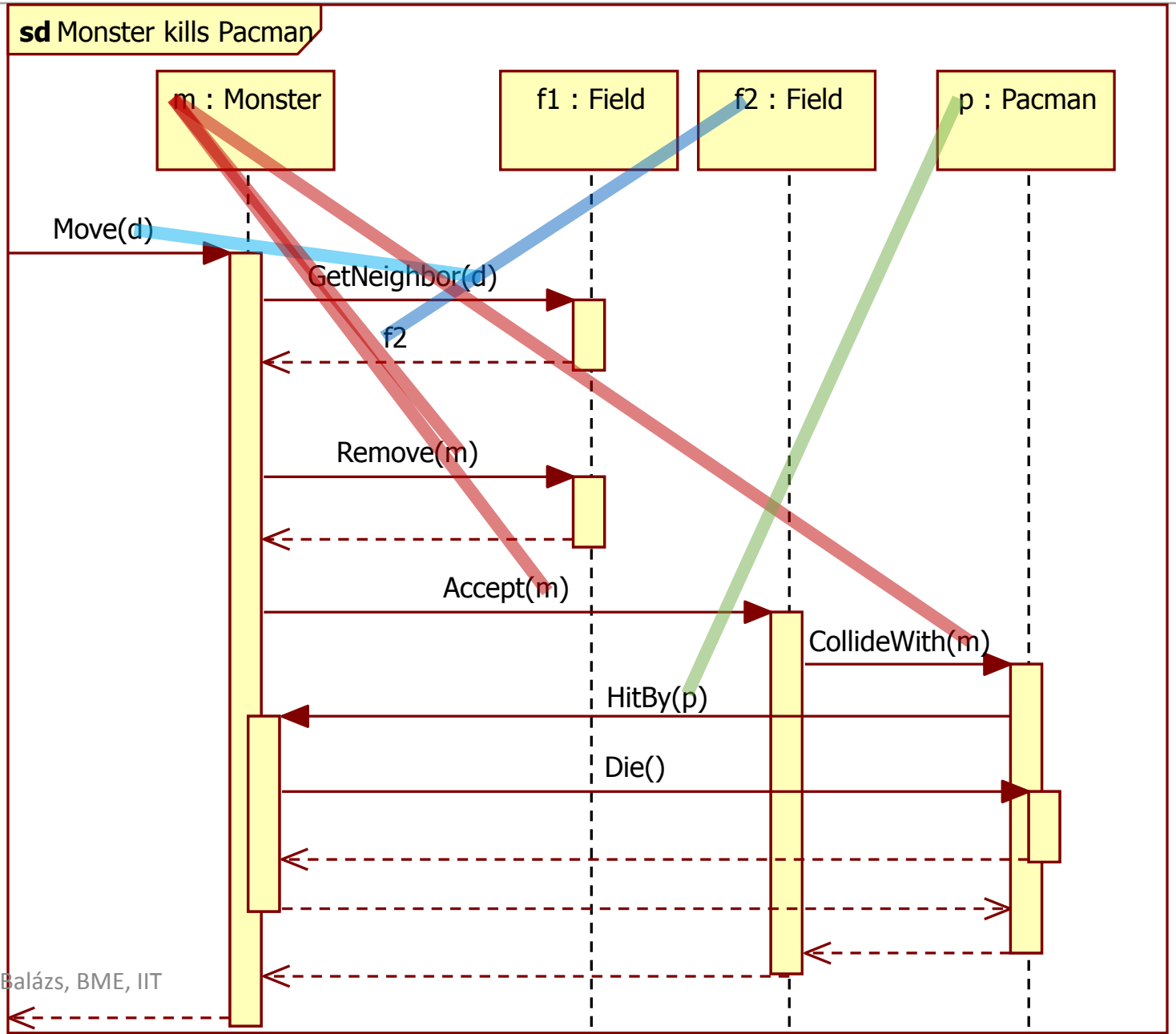
Szekvenciadiagram: a szörny megeszi a Pacmant



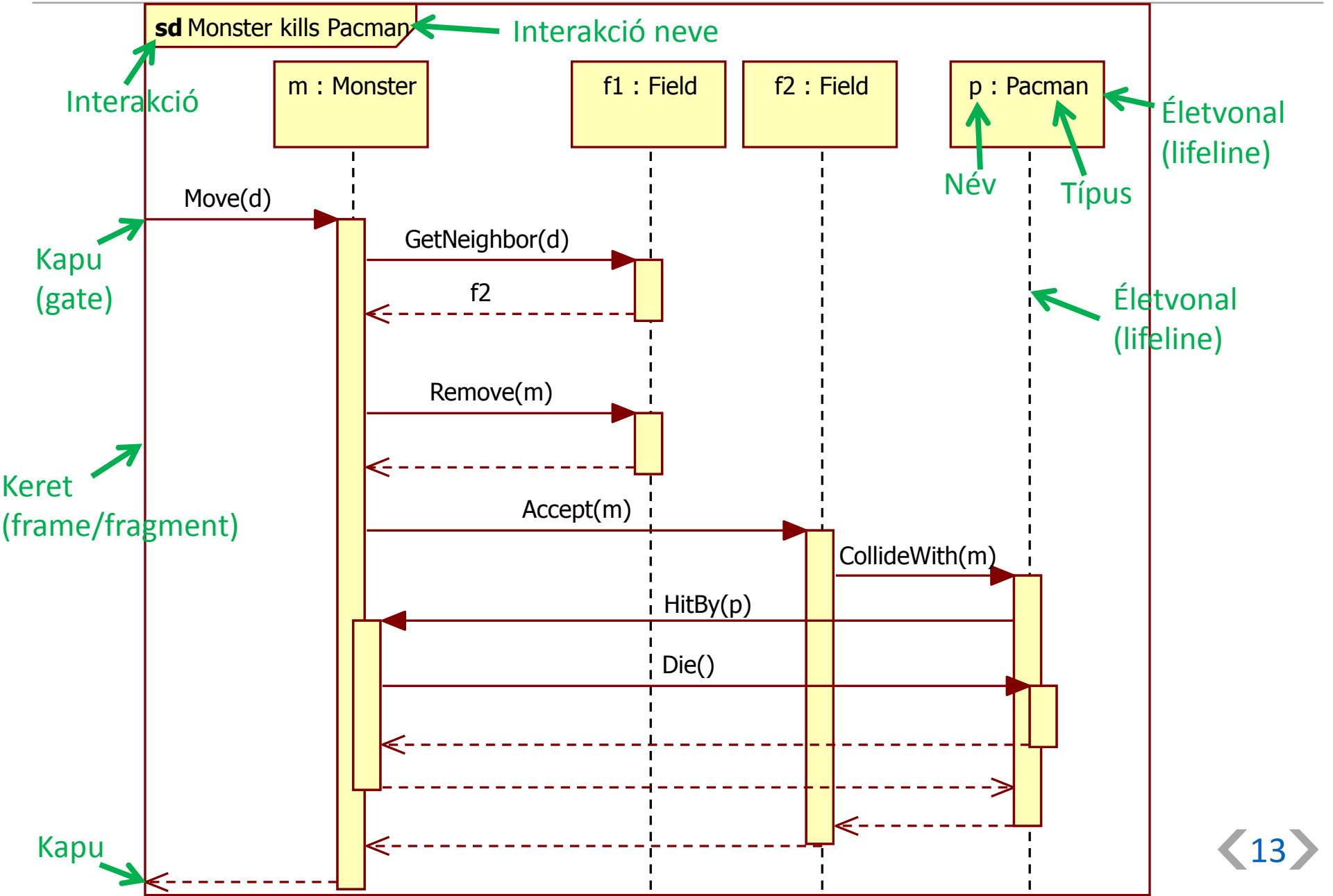
Szekvenciadiagram: a szörny megeszi a Pacmant



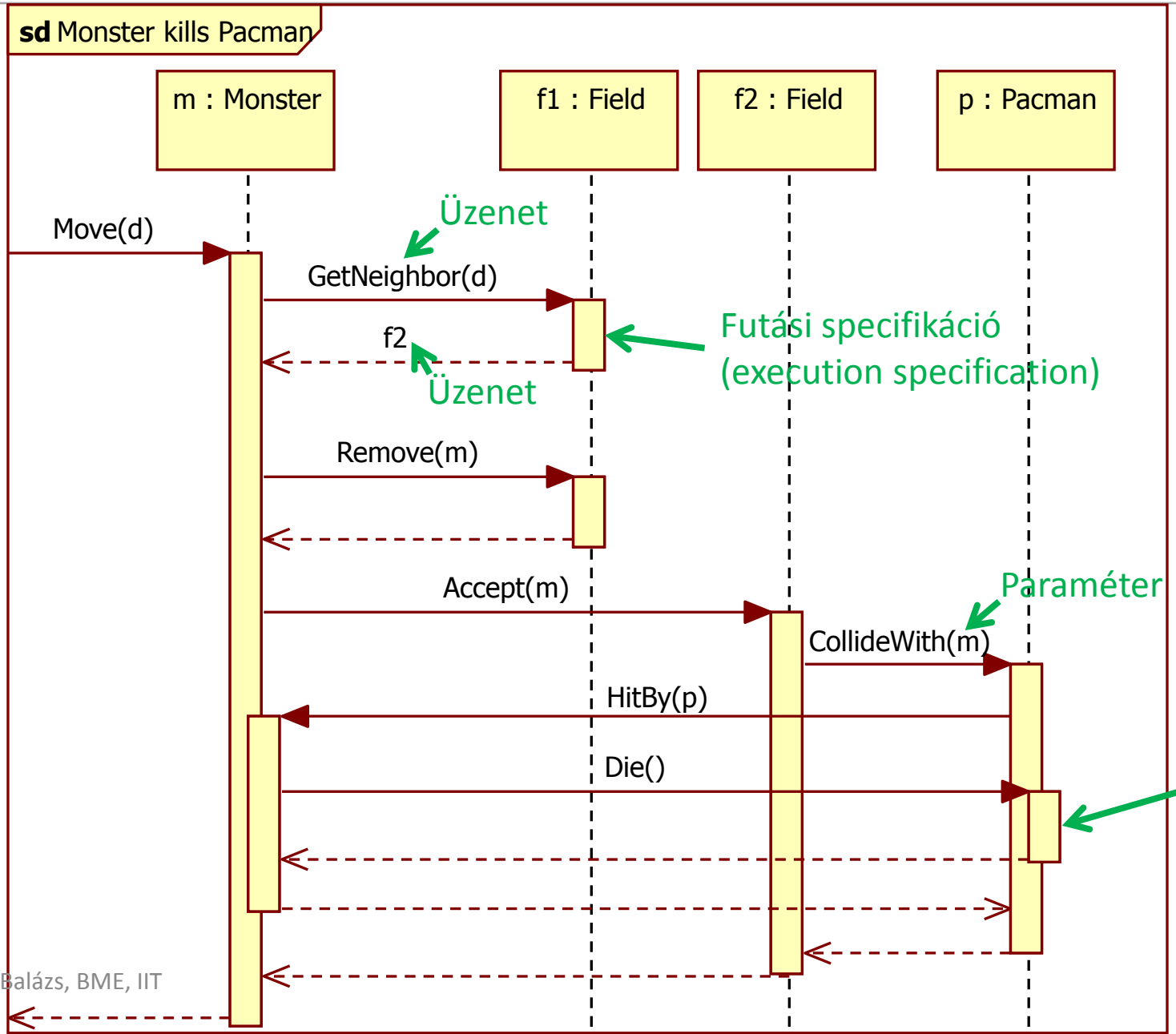
Szekvenciadiagram: a szörny megeszi a Pacmant



Szekvenciadiagram: a szörny megeszi a Pacmant



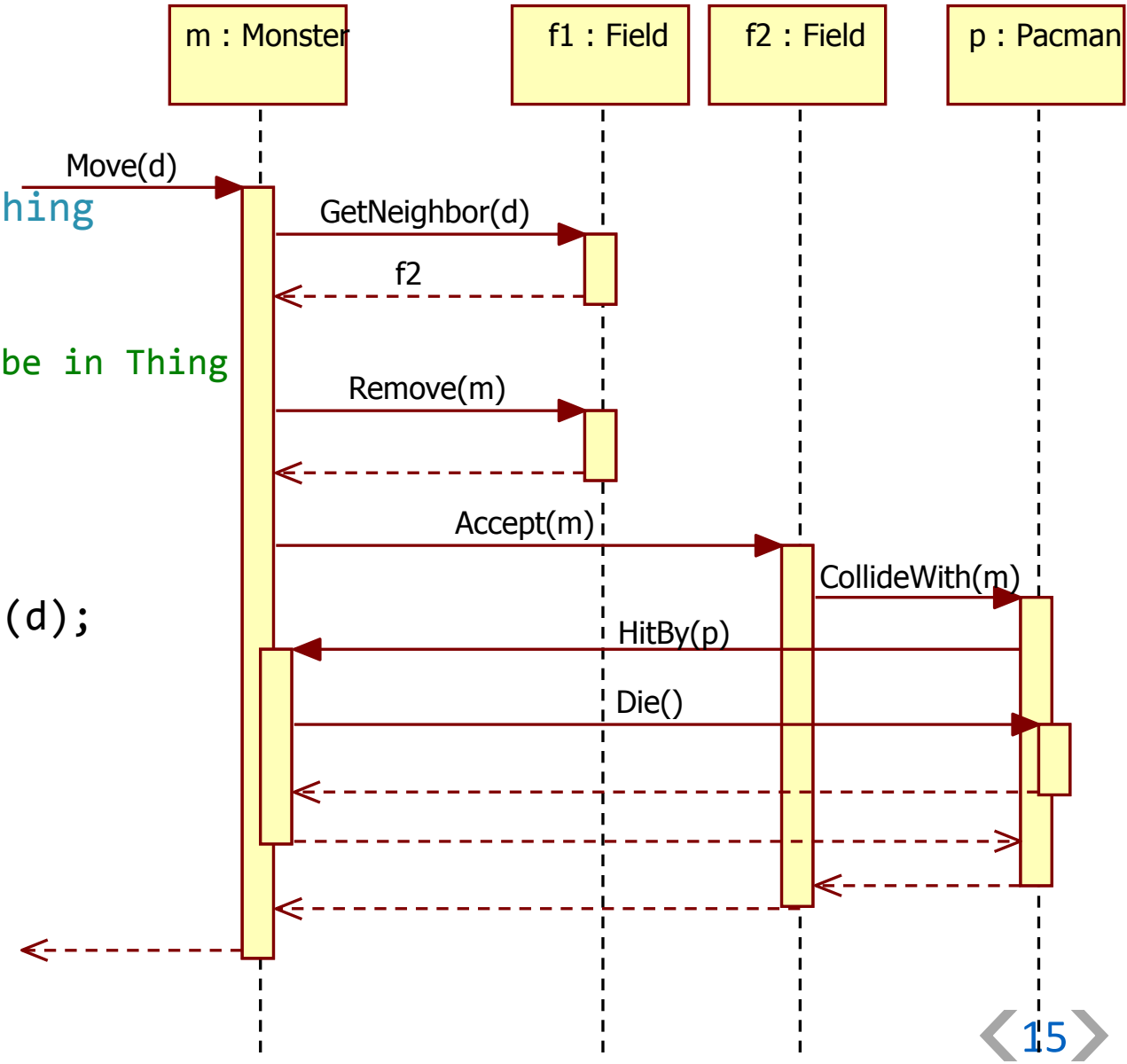
Szekvenciadiagram: a szörny megeszi a Pacmant



Szekvenciadiagram: a szörny megeszi a Pacmant

C++ leképzés:

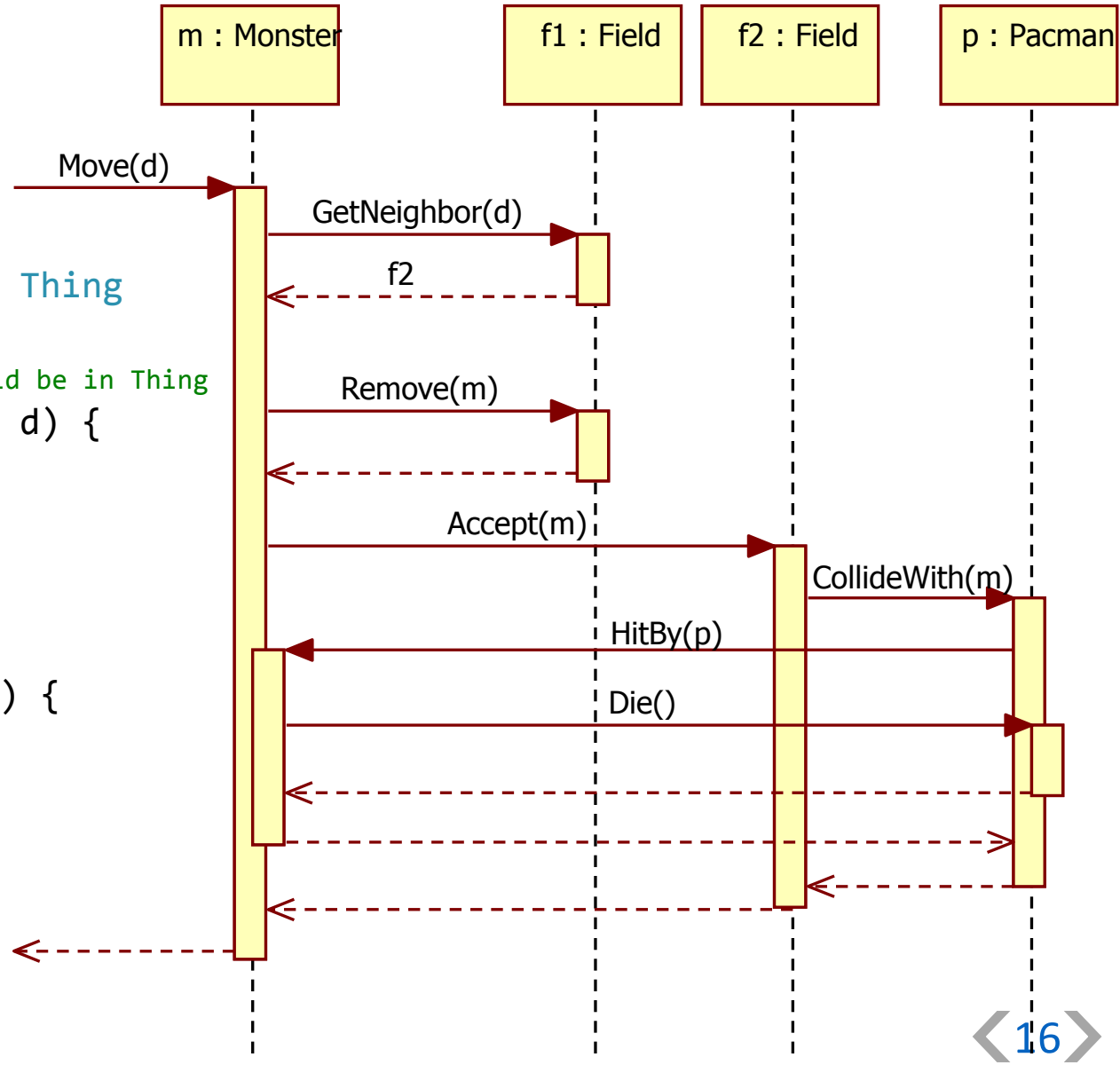
```
class Monster : public Thing
{
private:
  Field* field; //should be in Thing
public:
  void Move(Direction d)
  {
    Field* next =
      field->GetNeighbor(d);
    field->Remove(this);
    next->Accept(this);
  }
  void HitBy(Pacman* p)
  {
    p->Die();
  }
  // ...
}
```



Szekvenciadiagram: a szörny megeszi a Pacmant

Java leképzés:

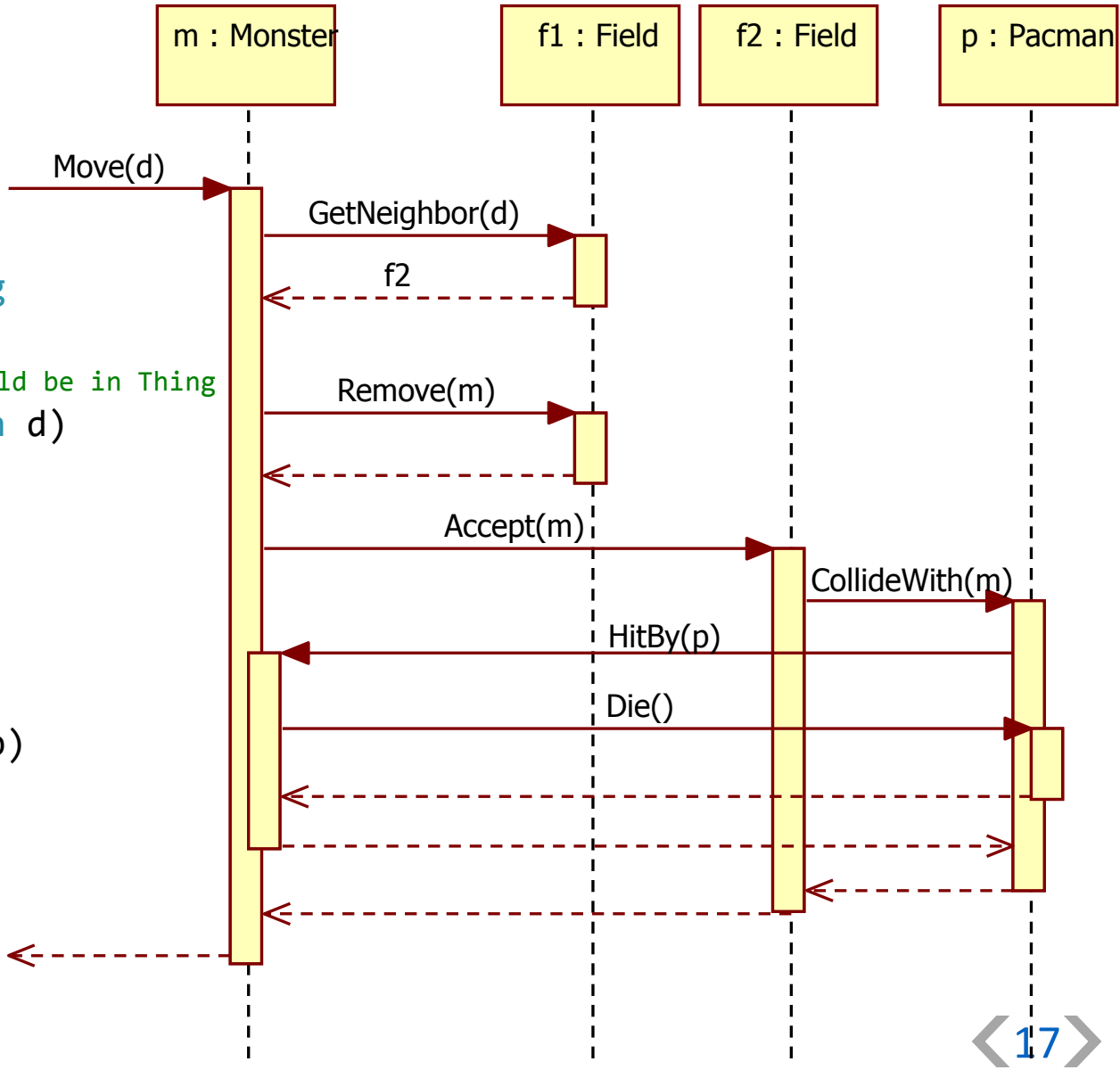
```
public class Monster extends Thing
{
  private Field field; //should be in Thing
  public void Move(Direction d) {
    Field next =
      field.GetNeighbor(d);
    field.Remove(this);
    next.Accept(this);
  }
  public void HitBy(Pacman p) {
    p.Die();
  }
  // ...
}
```



Szekvenciadiagram: a szörny megeszi a Pacmant

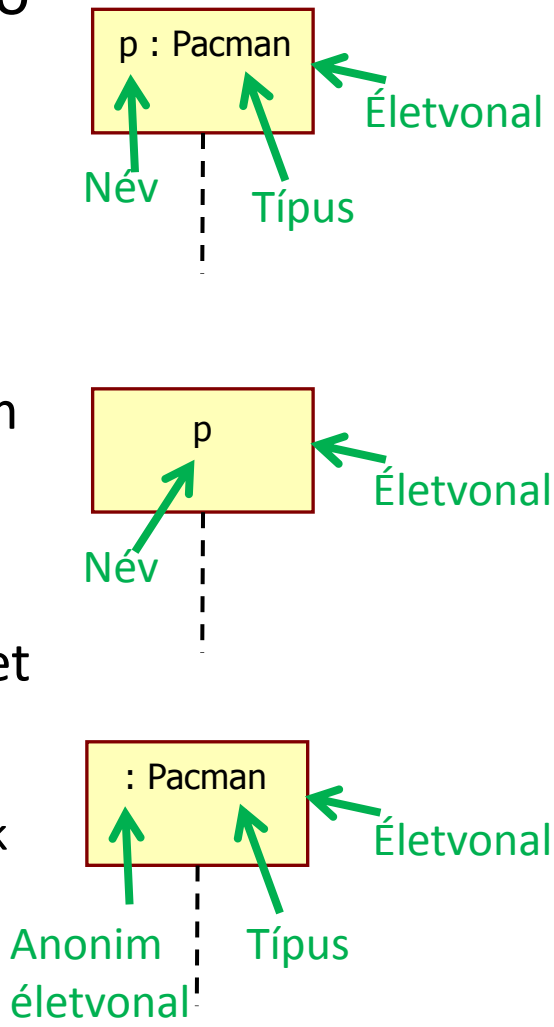
C# leképzés:

```
public class Monster : Thing
{
  private Field field; //should be in Thing
  public void Move(Direction d)
  {
    Field next =
      field.GetNeighbor(d);
    field.Remove(this);
    next.Accept(this);
  }
  public void HitBy(Pacman p)
  {
    p.Die();
  }
  // ...
}
```



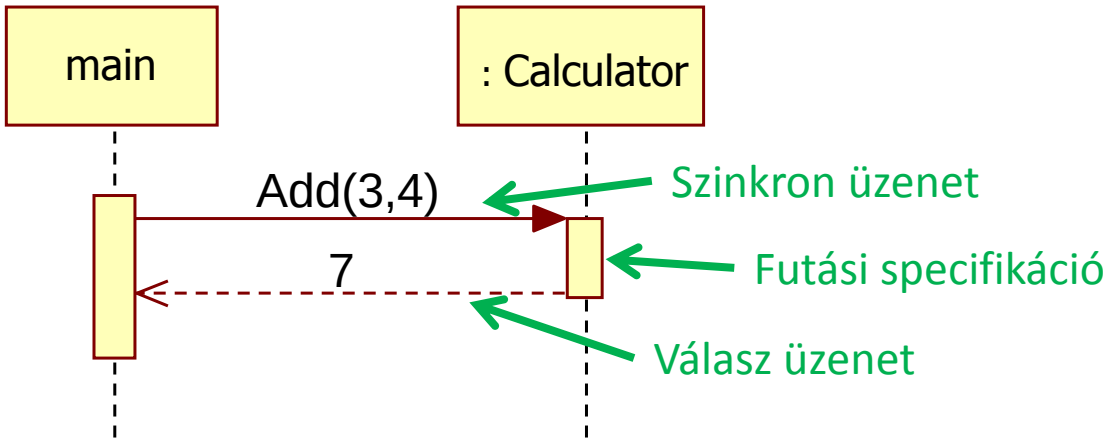
Életvonal (lifeline)

- Egy folyamat idővonalát ábrázolja, ahol az idő fentről lefelé telik
- Az életvonal feje nem objektum!
  - egy objektum (példány specifikáció) az csak egy pillanatkép a dinamikus példányról, amit ő modellez
  - a szekvenciadiagram nem egy pillanatkép, hanem egy időbeli folyamat
  - vagyis: a lifeline fejében *nem kell aláhúzni* a szöveget
  - akár interfész vagy absztrakt osztály is szerepelhet a lifeline fejében
    - de természetesen absztrakt függvények vagy interfészek függvényei csak hívhatók, de mivel nekik nincs implementációjuk, ők nem hívhatnak más függvényeket
- A név és a típus is opcionális, de legalább egyiknek szerepelnie kell



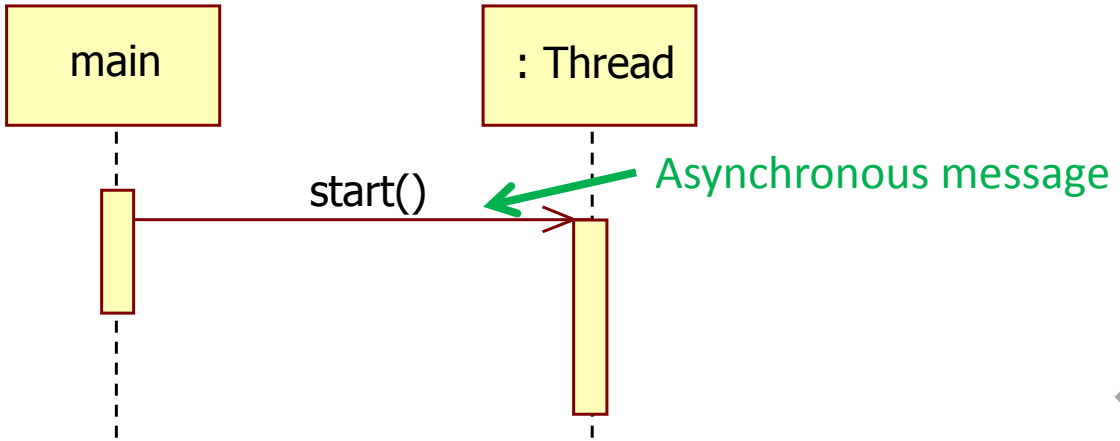
Szekvenciadiagram: szinkron üzenet

- Szinkron üzenet (synchronous message):
  - szinkron függvényhívás
  - az operáció neve és a paraméterek a nyíl felett szerepelnek
  - hatására egy futási specifikáció (execution specification) indul
  - a hívó megvárja, amíg a meghívott függvény véget ér
- Válasz üzenet:
  - a futási specifikáció végétől van rajzolva
  - a visszatérési érték a szaggatott vonal felett szerepel
- Futási specifikáció (execution specification):
  - egy függvényhívás törzsének futását jelöli, ekkor aktív a függvény
  - a programozási nyelvekben ez a stack frame



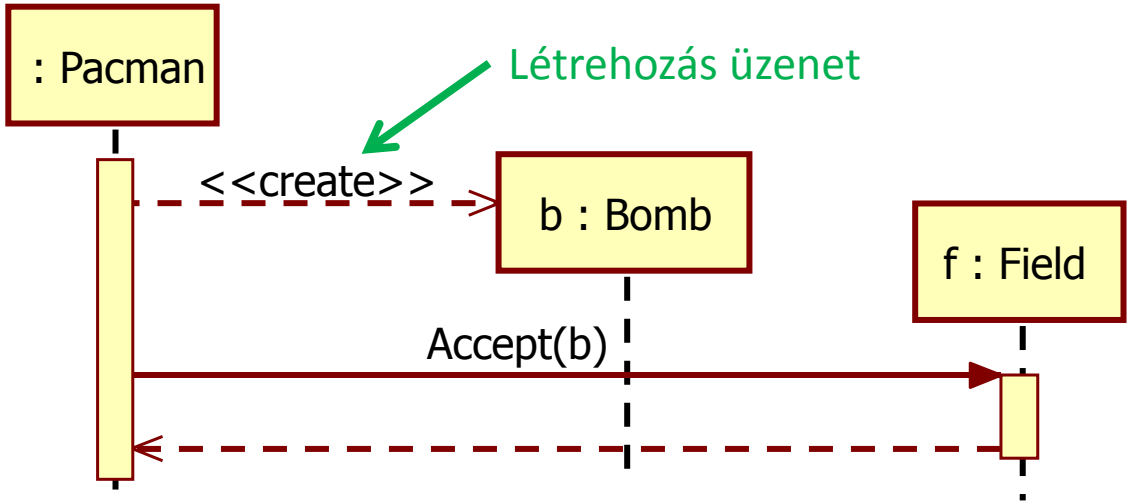
Szekvenciadiagram: aszinkron üzenet

- Aszinkron üzenet (asynchronous message), jelzés (signal):
  - aszinkron függvényhívás
  - a hívó nem várja meg a függvény lefutásának végét
    - egy másik szálat vagy folyamatot indít el
  - az operáció neve és a paraméterek a nyíl felett szerepelnek
  - hatására egy futási specifikáció (execution specification) indul
    - ami nem feltétlenül fér bele a hívó futási specifikációjának időtartamába
  - aszinkron üzenetnek nincs válaszüzenete
    - egy másik aszinkron hívással lehet visszahívást (callback) végezni

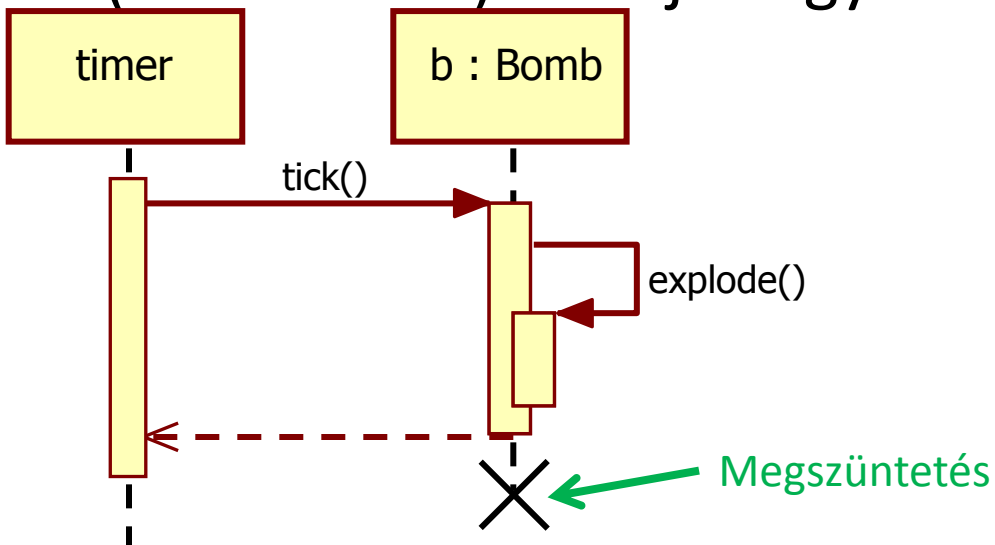


Szekvenciadiagram: létrehozás és megszüntetés

- Létrehozás üzenet (create message): új életvonalat készít



- Megszüntetés (destruction): befejez egy életvonalat



Szekvenciadiagram: alternatívák (alternatives)

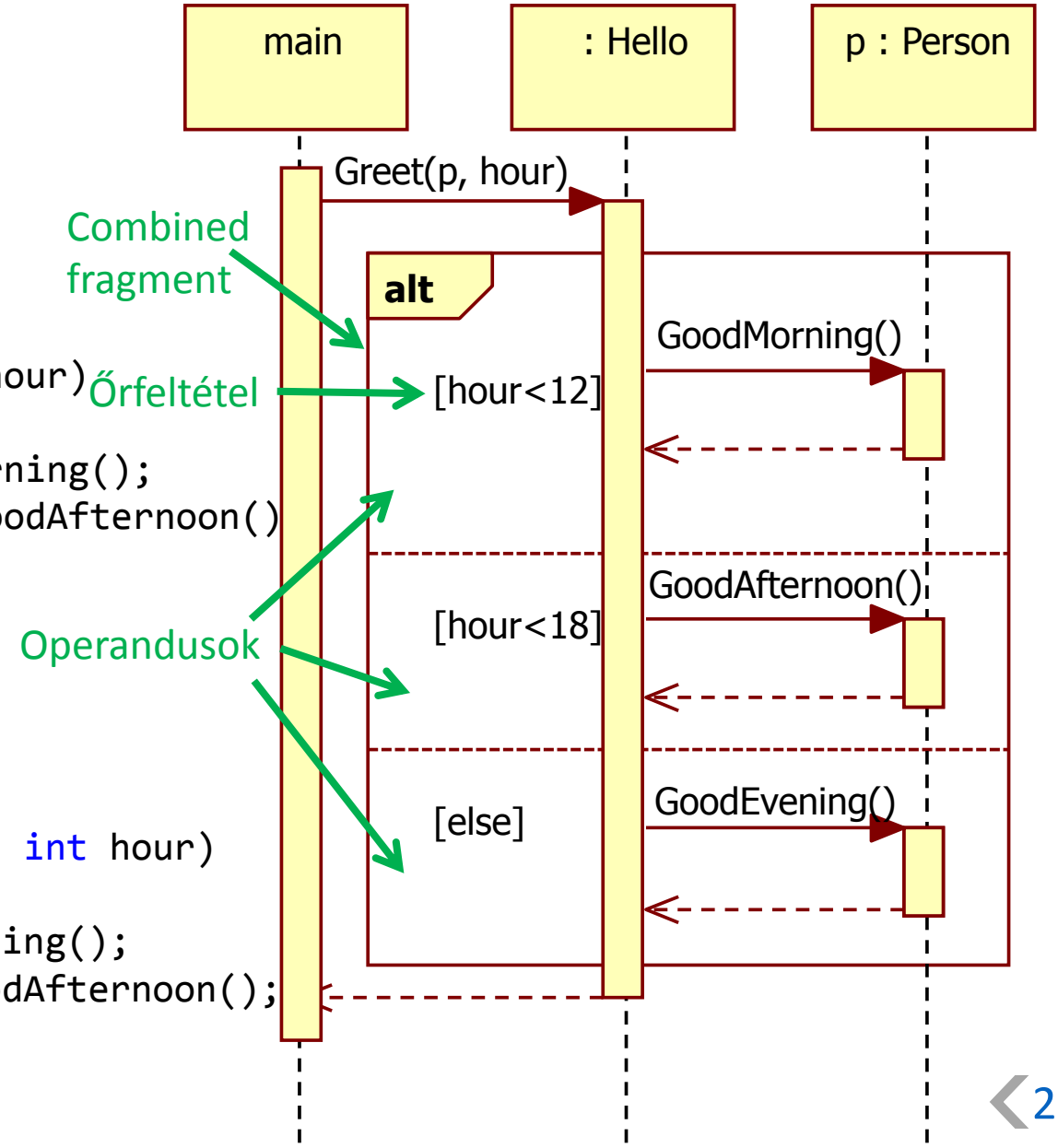
- Alternatívák: **alt**
  - if-else ágak

C++ leképzés:

```
class Hello {
public:
  void Greet(Person* p, int hour)
  {
    if (hour < 12) p->GoodMorning();
    else if (hour < 18) p->GoodAfternoon();
    else p->GoodEvening();
  }
}
```

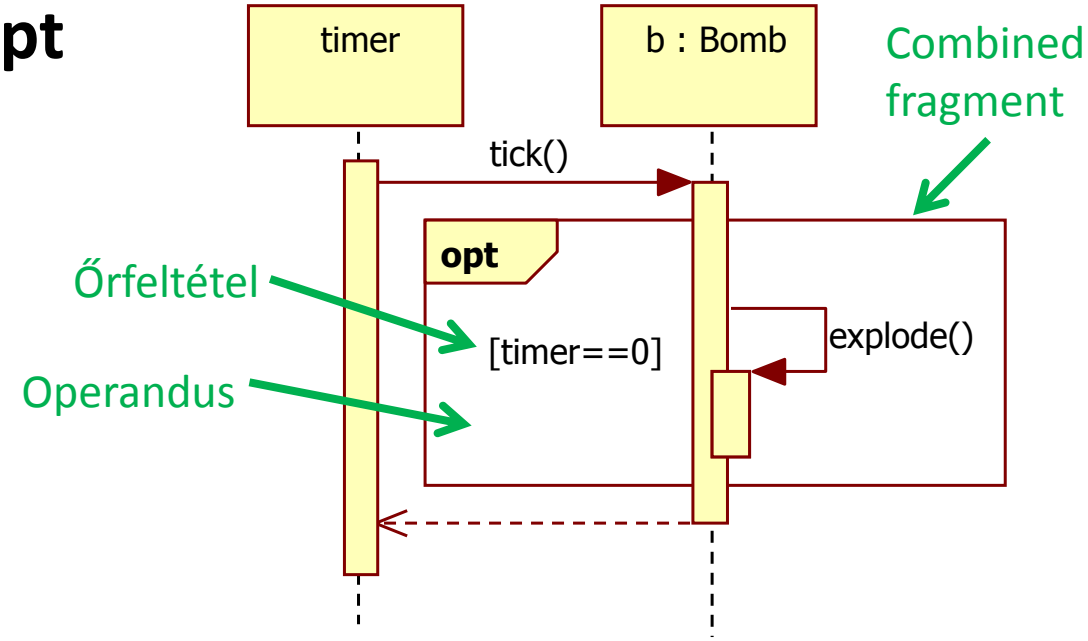
Java/C# leképzés:

```
public class Hello {
  public void Greet(Person p, int hour)
  {
    if (hour < 12) p.GoodMorning();
    else if (hour < 18) p.GoodAfternoon();
    else p.GoodEvening();
  }
}
```



Szekvenciadiagram: opció (option)

- Feltételes viselkedés: **opt**
  - else nélküli if



C++ leképzés:

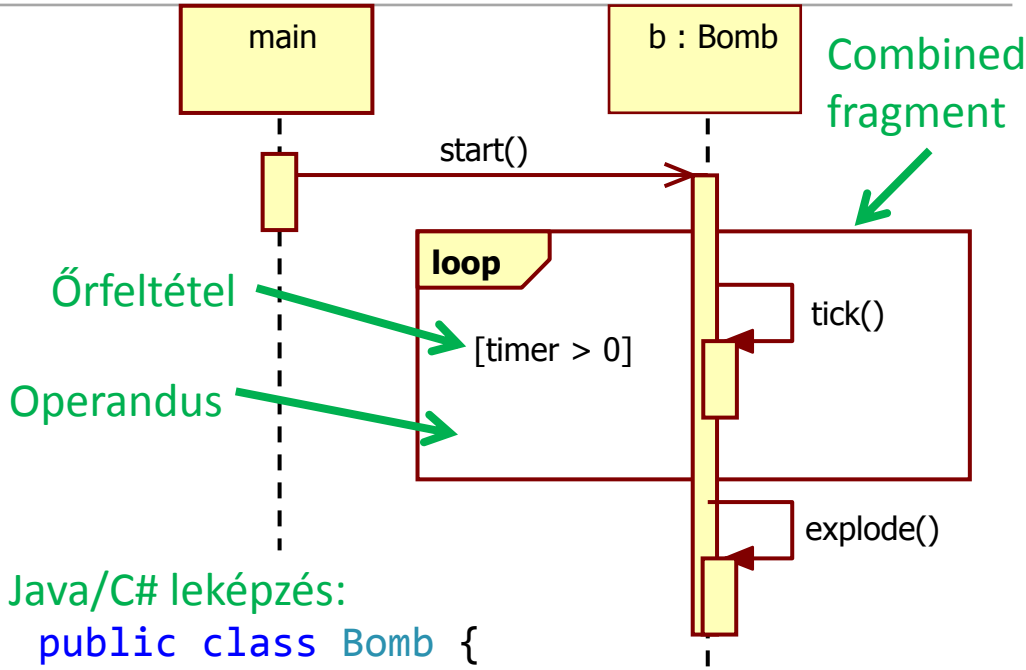
```
class Bomb {
private:
  int timer;
public:
  void Tick() {
    --timer;
    if (timer == 0) {
      this.Explode();
    }
  }
  void Explode() { /*...*/ }
}
```

Java/C# leképzés:

```
public class Bomb {
  private int timer;
  public void Tick() {
    --timer;
    if (timer == 0) {
      this.Explode();
    }
  }
  public void Explode() { /*...*/ }
}
```

Szekvenciadiagram: ciklus (loop)

- Ismételt viselkedés: **loop**



C++ leképzés:

```
class Bomb
{
private:
  int timer;
public:
  void Start()
  {
    while(timer > 0)
    {
      this.Tick();
    }
    this.Explode();
  }
  void Tick()
  {
    --timer;
  }
  void Explode() { /*...*/ }
}
```

Java/C# leképzés:

```
public class Bomb {
  private int timer;
  public void Start() {
    while(timer > 0) {
      this.Tick();
    }
    this.Explode();
  }
  public void Tick() {
    --timer;
  }
  public void Explode() { /*...*/ }
}
```

Szekvenciadiagram : combined fragments

Rövidí-tés	Fajta	Jelentés
alt	Anternatívák	Viselkedés kiválasztása feltétel alapján: legfeljebb egy operandus lesz lefuttatva.
opt	Opció	Opcionális viselkedés feltétel alapján: vagy lefut az egyetlen operandus, vagy nem történik semmi.
break	Megszakítás	A tartalmazó fragment futása megszakad, és a maradék rész nem fut le.
par	Párhuzamos	Párhuzamosan futnak le az operandusok.
seq	Gyenge sorrend	Gyenge sorrendet határoz meg az operandusok között: csak az azonos lifeline-on belül kell tartani a sorrendet.
strict	Erős sorrend	Erős sorrendet határoz meg az operandusok között: a függőleges koordináta szigorúan meghatározza a sorrendet.

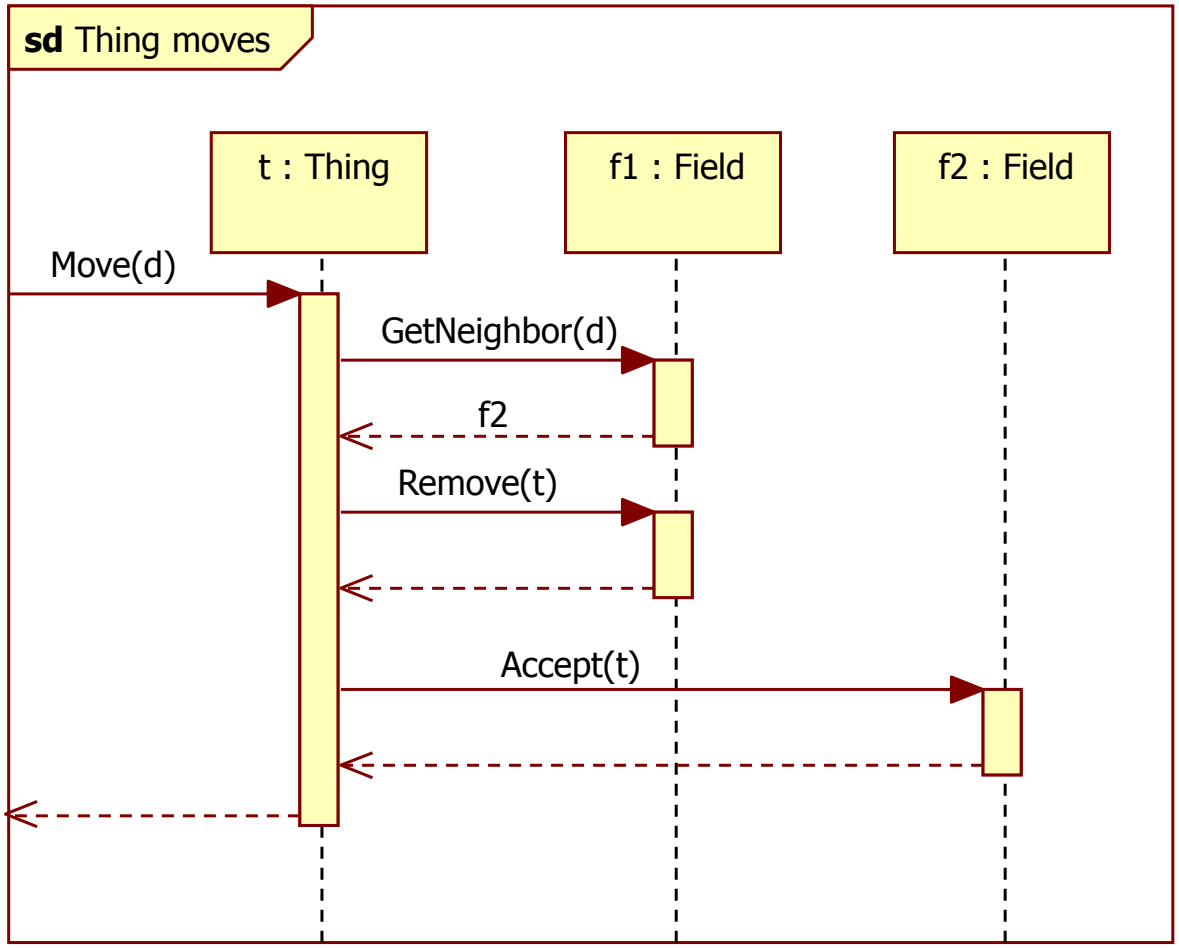
Szekvenciadiagram : combined fragments

Rövidí-tés	Fajta	Jelentés
neg	Negatív	Helytelen lefutásokat mutat. Minden ezektől eltérő lefutás helyesnek és lehetségesnek számít.
critical	Kritikus szakasz	Atomi műveletnek tekinthető a tartalmazó fragment szempontjából.
ignore	Rejtett üzenetek	Azt jelzi, hogy néhány üzenet nincs megmutatva az adott fragmentben.
consider	Fontos üzenetek	Azt jelzi, hogy néhány üzenet fontos az adott fragmentben, a többi üzenet rejtettnek tekinthető.
assert	Állítás	Egy állítást reprezentál. Csak az operandus szekvenciái a helyes folytatások, minden más folytatás helytelen.
loop	Ciklus	Ciklust reprezentál: az operandus ismételten le lesz futtatva.

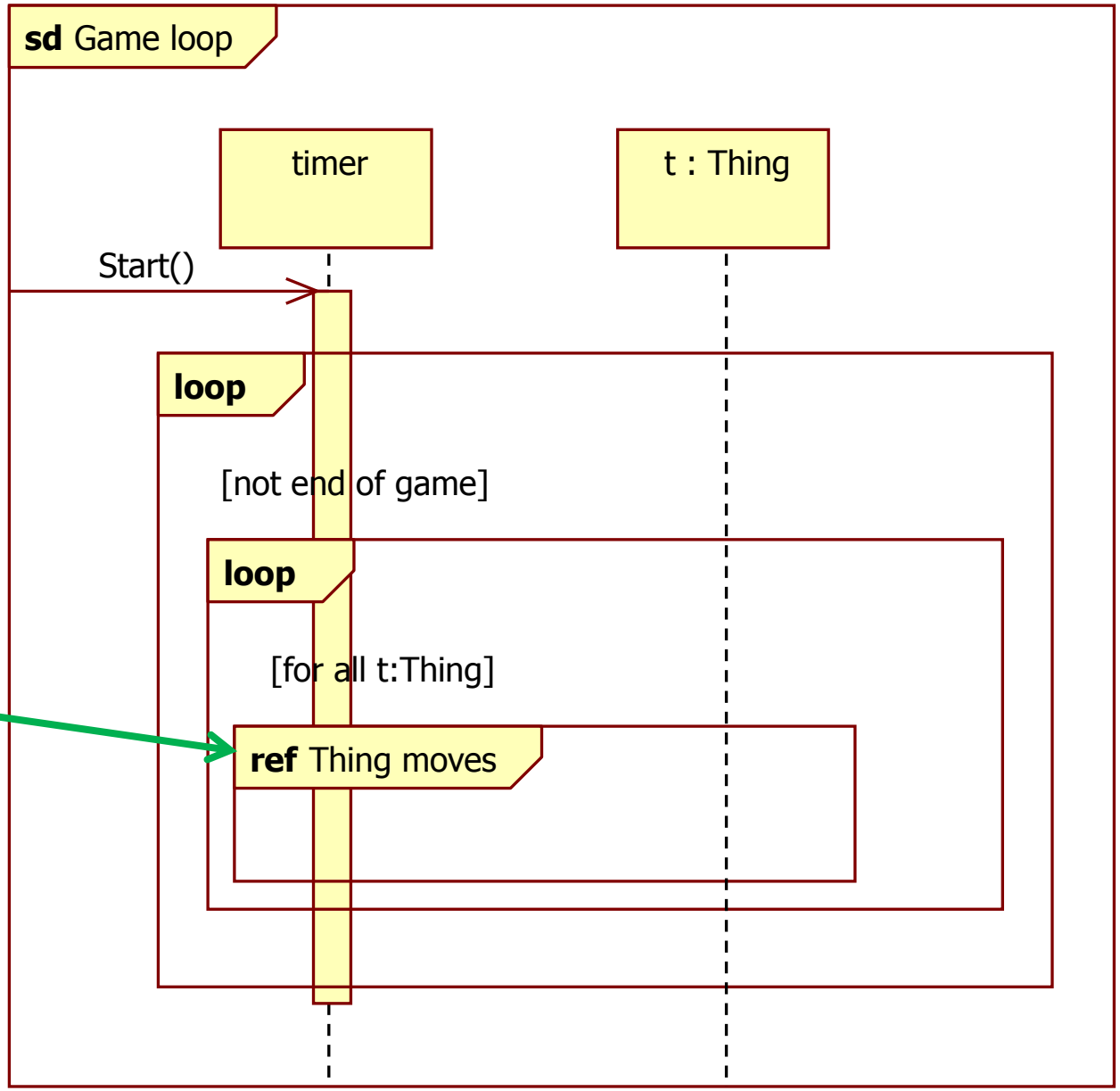
Szekvenciadiagram : interakció használata (interaction use)

- A szekvenciadiagramok meghivatkozhatnak más szekvenciadiagramokat az interakció használata segítségével
- Előnyök:
  - modularitás: nagy diagramok feldarabolhatók több kisebb diagramra
  - újrahasznosítás: több diagramon előforduló azonos viselkedés kiemelhető és meghivatkozható egy közös diagram alapján
  - olvashatóság: magasabb szintű diagramok finomíthatók alacsonyabb szintű diagramok segítségével

Szekvenciadiagram: Thing moves



Szekvenciadiagram: Interakció használata



Interakció használata

(meghivatkozza az előző dián szereplő szekvenciát)

Konzisztens és kezelhető szekvenciadiagramok

- Egy diagram pontosan egy viselkedést ábrázoljon
- Azonos típusú, de különböző objektumok külön lifeline-t kapjanak
- A hívónak ismernie kell a cél objektumot
  - egy objektumdiagramon ábrázolhatjuk a kezdeti ismeretségeket
- A hívott függvénynek elérhetőnek kell lennie a használt objektum ismert statikus típusa alapján
- A polimorfikus viselkedés leírására külön diagramokat rajzoljunk
- Ugyanaz a függvény ugyanúgy viselkedjen különböző diagramokon
- Tüntessük fel a paramétereket és a visszatérési értékeket is
- Az operációknak léteznie kell az osztálydiagramon

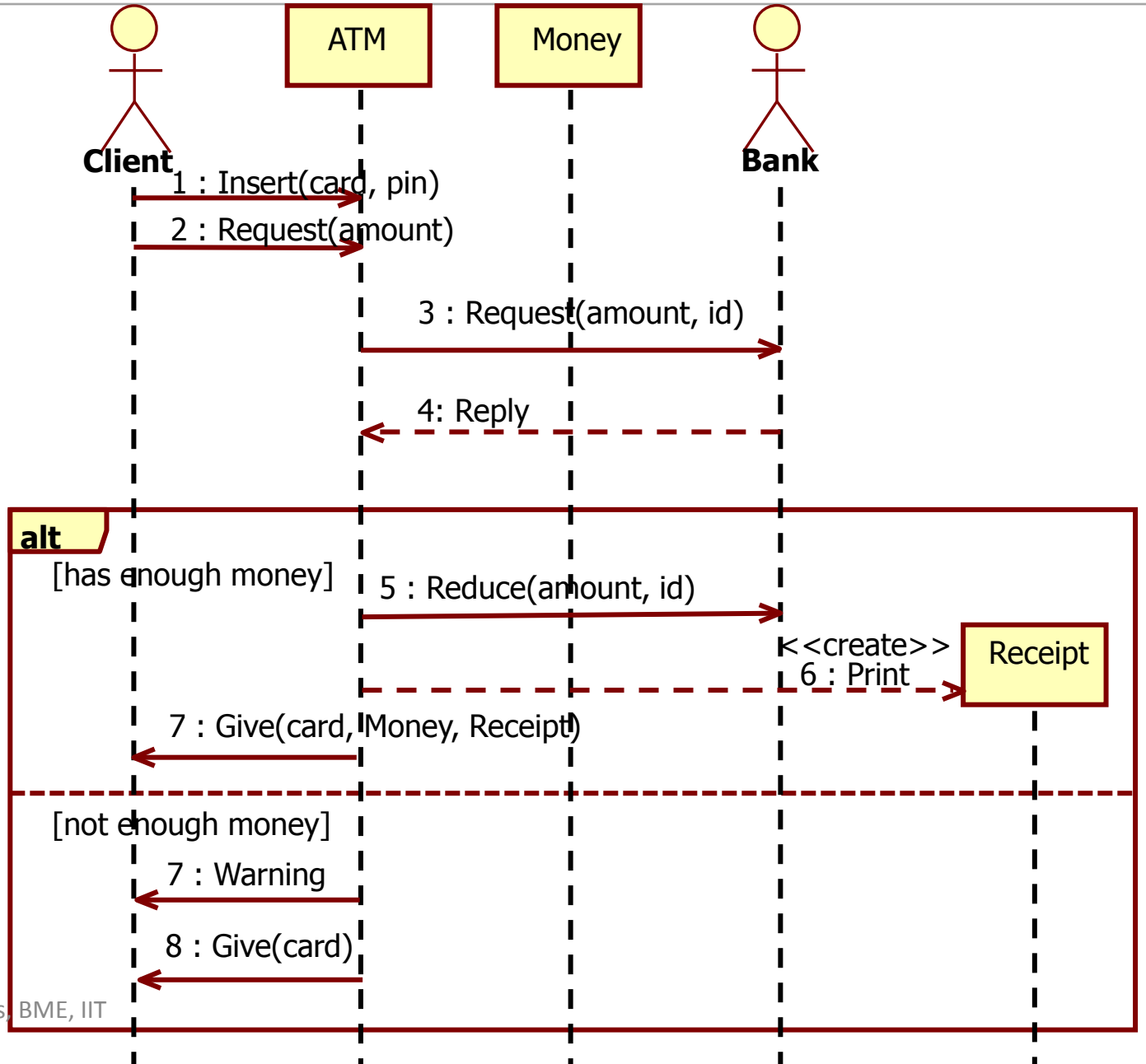
Informális szekvenciadiagram

- Informális szekvenciadiagramok:
  - enyhítenek az UML szigorú formalizmusán
  - nem pontosan követik az UML szabványt
  - de kiválóan alkalmasak ötletek felvázolására
- Tipikusan nem rajzoljuk meg a rendszer minden egyes apró részletét formális diagramokon
  - általában ez szükségtelen
  - sokszor időpazarló: a viselkedés leprogramozása egyszerűbb és gyorsabb, mint béna eszközökkel szekvenciadiagramokat rajzolgatni
  - szinkronban tartani őket a forráskóddal is időpazarló, és nehéz
- Általában az informális diagramok elegendő információt adnak a rendszer működéséről
  - ezeket könnyebb használni vázlatként
  - elegendő részletet adnak az áttekintő kép megértéséhez
  - csak akkor kell őket frissíteni a dokumentációban, ha az általuk mutatott viselkedést befolyásoló tervezői döntések megváltoznak
- Megjegyzés: a házi feladatban és a vizsgán, valamint a „Szoftver projekt labor” c. tárgyban részletes ***formális*** diagramokat várunk el
  - a célunk, hogy lássuk, sikerült-e elsajátítani az UML szabványt

Példa: Pénzfelvétel use case

- **Use case:** Pénzfelvétel
- **Aktorok:** Ügyfél, Bank
- **Főforgatókönyv:**
  - 1. Az Ügyfél átadja a bankkártyát és a pinkódot
  - 2. Az ATM ellenőrzi, hogy a bankkártyához tartozik-e a pinkód
  - 3. Az Ügyfél megadja, hogy mennyi pénzt venne fel
  - 4. Az ATM megkérdezi a Bank-ot, hogy ez így rendben van-e
  - 5. A Bank megerősíti, hogy mehet a tranzakció
  - 6. Az ATM kiadja a bankkártyát
  - 7. Az Ügyfél elveszi a bankkártyát
  - 8. Az ATM kinyomtatja a bizonylatot és kiadja a pénzzel együtt
  - 9. Az Ügyfél elveszi a pénzt és a bizonylatot
- **Alternatív forgatókönyv 5.A:**
  - 5.A.1. A Bank jelzi, hogy az Ügyfél számláján nincs elég pénz
  - 5.A.2. Az ATM visszaadja a bankkártyát és hibaüzenetet ír
  - 5.A.3. Az Ügyfél elveszi a bankkártyát

Informális szekvenciadiagram a pénzfelvételre



Hol tartunk?

Strukturális UML diagramok:

Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

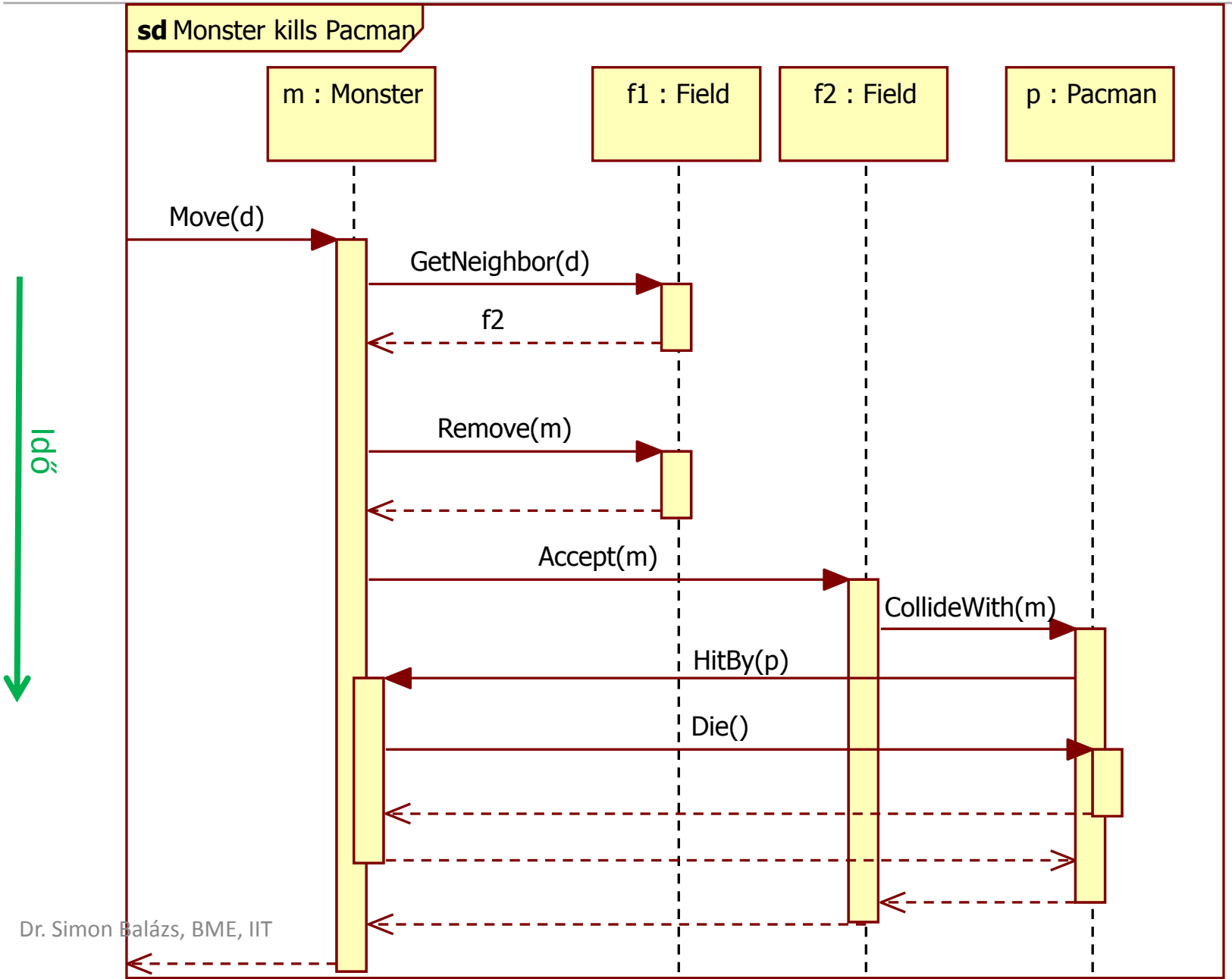
Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakciós áttekintő diagram	

Kommunikációs diagram (Communication diagram)

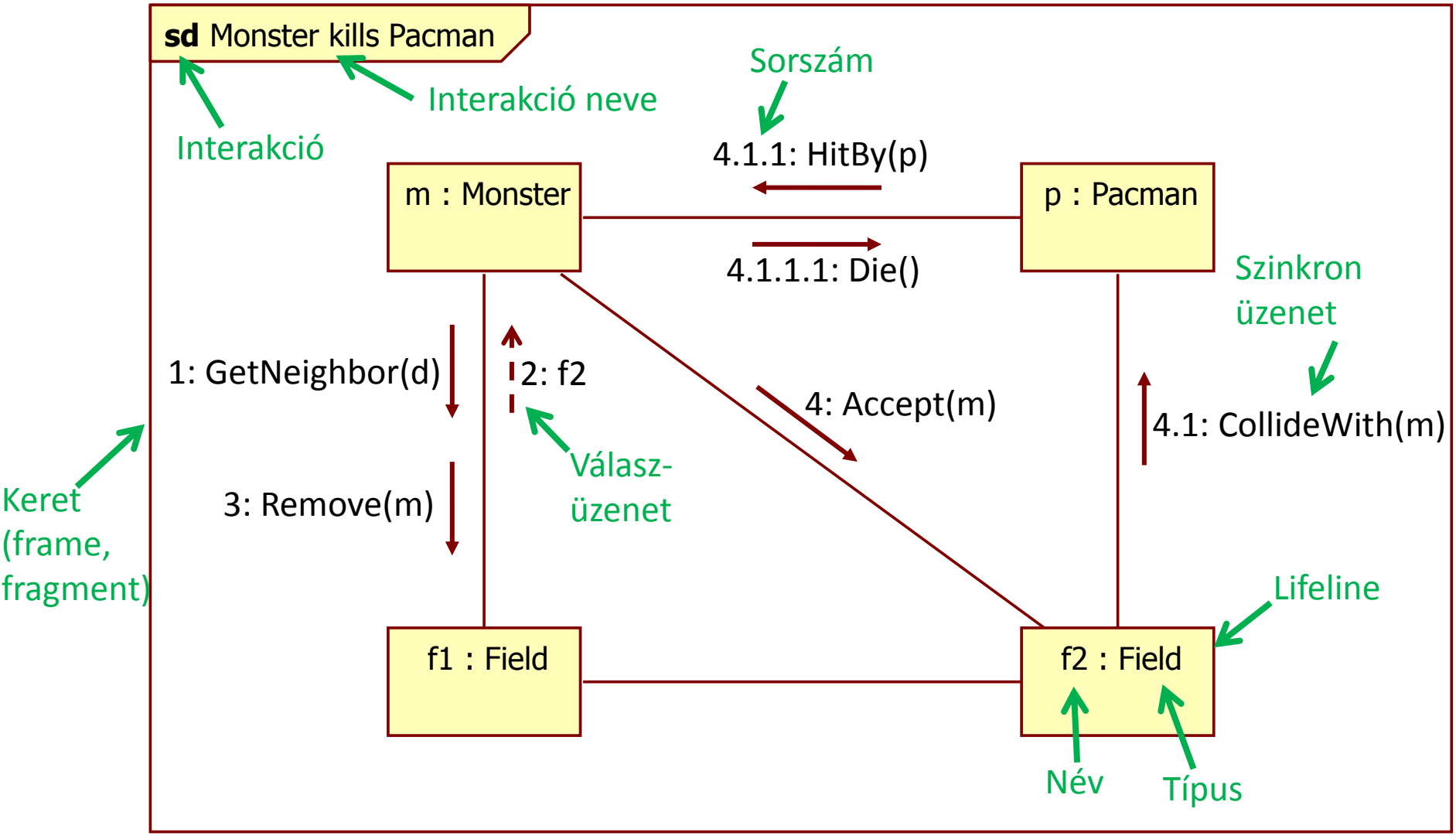
Kommunikációs diagram

- Interakciók grafikus ábrázolására szolgál
  - a rendszer dinamikus viselkedését mutatja
  - az interakciók a részvevők közötti információcserére fókuszálnak
  - az üzenetek sorrendjét hierarchikus számozás határozza meg
- Egy kommunikációs diagram olyan szekvenciadiagramnak felel meg, amelyen nincs strukturális jelölés (interakció használata, combined fragment)
  - a szekvenciadiagramok erőssége a logikai sorrend mutatása, az áttekintő képet azonban nehéz látni
  - a kommunikációs diagramok nagyon jó áttekintő képet adnak (szereplők és kapcsolataik), de nehéz követni a logikai sorrendet
  - a kommunikációs diagram olyan, mintha a szekvenciadiagramot felülről néznénk

Emlékeztető: a szörny megeszi a Pacmant



Kommunikációs diagram: a szörny megeszi a Pacmant



Hol tartunk?

Strukturális UML diagramok:

Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakciós áttekintő diagram	

Interakciós áttekintő diagram (Interaction Overview Diagram)

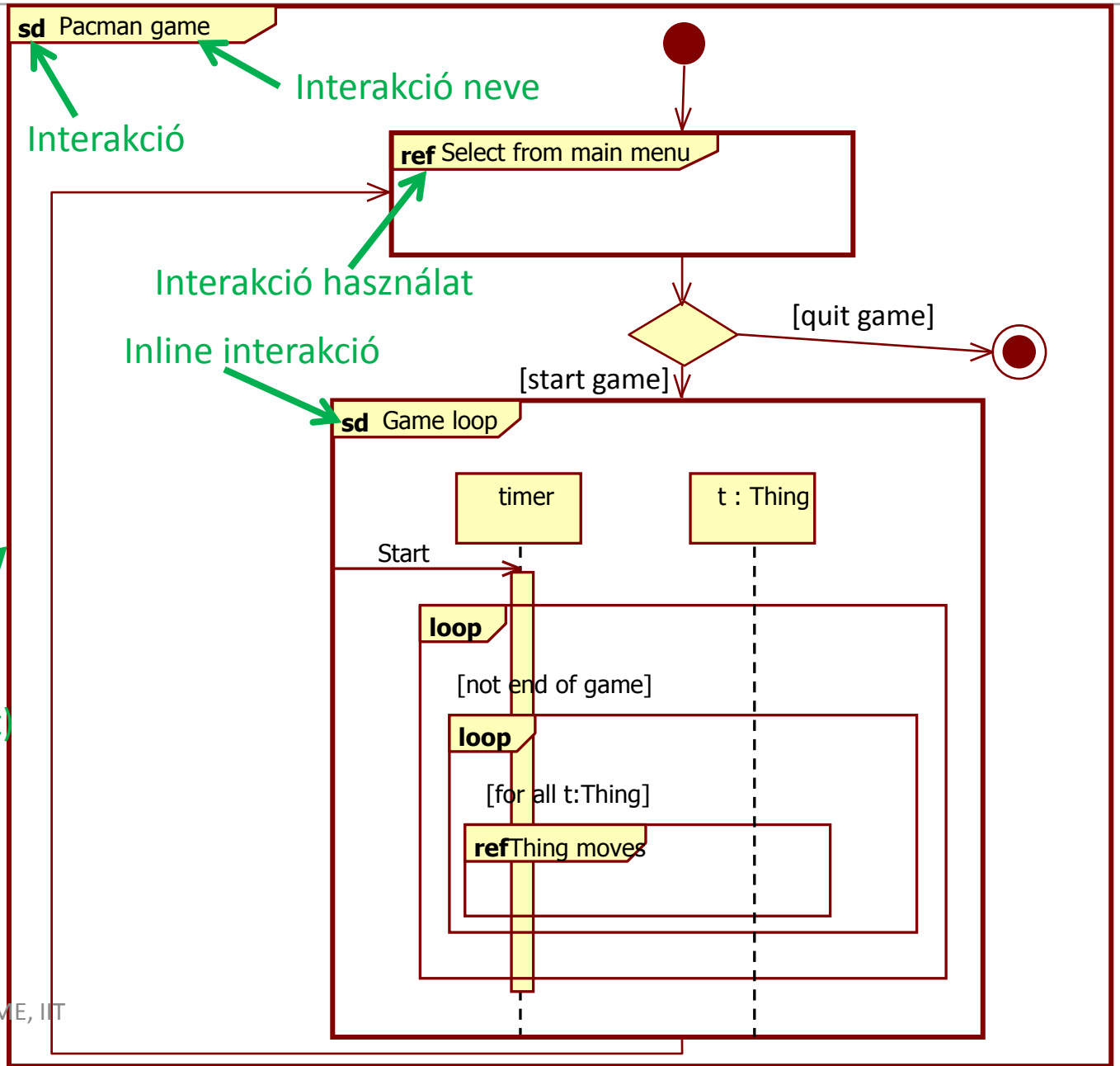
Interakciós áttekintő diagram

- Az interakciós áttekintő diagramok az aktivitásdiagramok egyfajta változatai, amelyek az interakciók egymásutániságáról adnak áttekintő képet
- A különbség az aktivitásdiagramhoz képest: a csomópontok nem akciók, hanem interakciók és interakció használatok
  - Interakciók: inline szekvenciadiagram, kollaborációs diagram, vagy interakciós áttekintő diagram
  - Interakció használatok: referencia egy szekvenciadiagramra, kollaborációs diagramra vagy interakciós áttekintő diagramra

Korlátozások

- További különbségek az aktivitásdiagramokhoz képest:
  - csomópontok: interakciók vagy interakció használatok
  - feltételes combined fragment-ek megfelelője a döntési-merge csomópontpárok
  - a párhuzamos combined fragment-ek megfelelője a fork-join csomópontpárok
  - a ciklus combined fragment-ek megfelelője a gráfban létrehozott körök
  - a feltételes és párhuzamos részeknek hierarchikusan egymásba ágyazottnak kell lennie
  - az interakciós áttekintő diagramok is ugyanolyan interakciós keretben vannak, mint a szekvenciadiagramok és a kollaborációs diagramok

Interakciós áttekintő diagram példa: Pacman játék



Hol tartunk?

Strukturális UML diagramok:

Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakciós áttekintő diagram	

Összefoglalás

- Interakciós grafikus reprezentációja
- A fókusz a részvevők közötti információcserén van
- UML diagramok:
  - **Szekvenciadiagram:** a logikai sorrendet mutatja, az áttekintő képet azonban nehéz látni
  - **Kommunikációs diagram:** nagyon jó áttekintő képet ad (szereplők és kapcsolataik), de nehéz követni a logikai sorrendet
  - **Interakciós áttekintő diagram:** az aktivitásdiagramok egyfajta változata, amely az interakciók egymásutániságáról ad áttekintő képet

Köszönöm a figyelmet!