SOM
Questions: Analysis & UML

Hans Philippi

September 25, 2019

Today's lecture

- Open questions and discussion
- A Kahoot quiz

Question

- *Wat is precies het verschil tussen compositie en aggregatie?*
- *In de slides van UML 1 – slide 43: welke is aggregatie en welke is compositie en waarom?*

We own a fleet of planes. On of the plane types is the Boeing 737. Each 737 has a left engine and a right engine. Each plane and each engine have known identities (and a maintenance record).

Our catalog has a collection of racing bikes (for instance X-LITE CRS-3000). Each racing bike has a rear derailleur (for instance Ultegra 6800).

Answer

- *Wat is precies het verschil tussen compositie en aggregatie?*
- *In de slides van UML 1 – slide 43: welke is aggregatie en welke is compositie en waarom?*

Vliegtuig: *compositie*. Voor elke concrete motor is er een apart object, met bijvoorbeeld de onderhoudshistorie. Als het vliegtuig neerstort, gaat de motor ook verloren. Een motor kan vervangen worden door een ander exemplaar, maar dan wordt er een ander object toegekend aan Plane.

Fiets: *aggregatie*. Er is één object voor de Ultegra 6800 derailleur die door alle fietsen met dit onderdeel gedeeld wordt.

Answer

- *Wat is precies het verschil tussen compositie en aggregatie?*
- *In de slides van UML 1 – slide 43: welke is associatie en welke is compositie en waarom?*

Ander voorbeeld: een cursus is een aggregatie van studenten, maar studenten zijn composities van lichaamsdelen.

Question

- *Slide 27 maakt onderscheid tussen B aggregates A objects en B contains A objects. Wat is precies het verschil tussen deze twee voorwaarden?*

Answer

- *Slide 27 maakt onderscheid tussen B aggregates A objects en B contains A objects. Wat is precies het verschil tussen deze twee voorwaarden?*

- Een voorbeeld: B bevat (aggregeert) een object C, dat een container is van objecten van class A
- Denk eens terug aan het programma *Shapes*

Question

- *Slide 45 beschrijft het verschil tussen aggregatie en dependency maar er staat dat Objects of class B create objects of type A. Boven deze zin staat dat dit niet aggregatie is maar dependency. Maar slide 38 van dezelfde set beschrijft met een stukje code dat class A objecten maakt van type B? Dus dan is het toch wel aggregatie?*

Allereerst: aggregatie is ook een dependency, maar niet iedere dependency is een aggregatie.

Answer

- *Object aggregation vs object creation*

Punt twee: kijk eens naar deze code. A aggregeert B, maar creëert niet zelf een B-object!

```
public class A
{
  public B myB;

  ...
      myB = myBfactory(42);
  ...
}
```

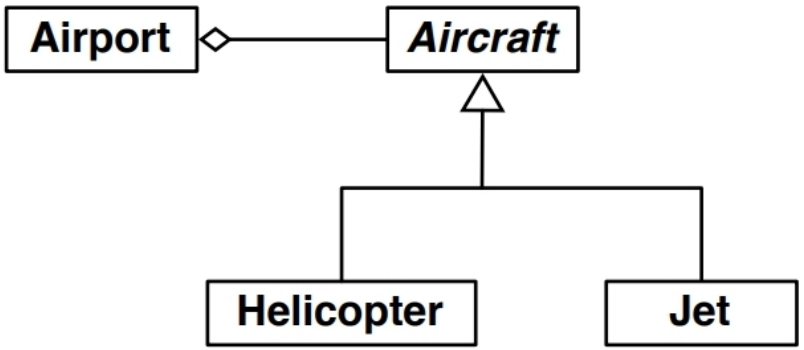
Answer

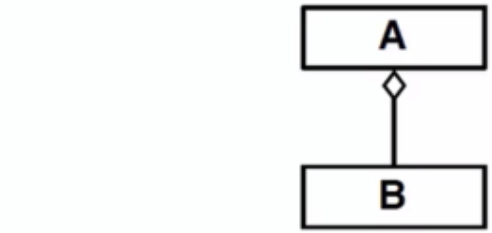
- *Object aggregation vs object creation*

Punt twee bis: kijk eens naar deze code. F creëert B, maar aggregeert niet zelf een B!

```
public class F
{
  public B myBfactory (int v) {
      if (v == 42)
          return new B ("ultimate");
      else
          return new B ("other");
  }
  ...
}
```

Question



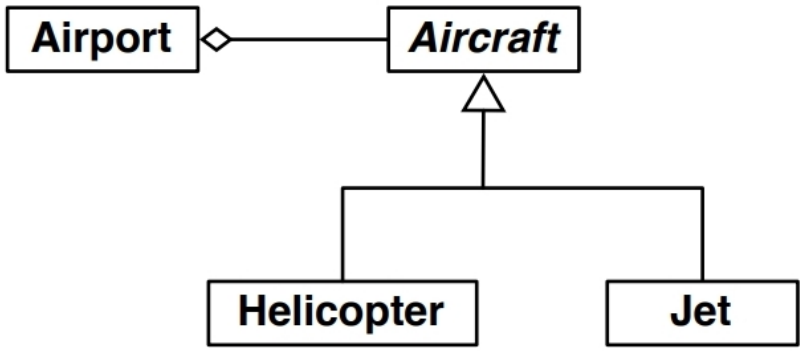


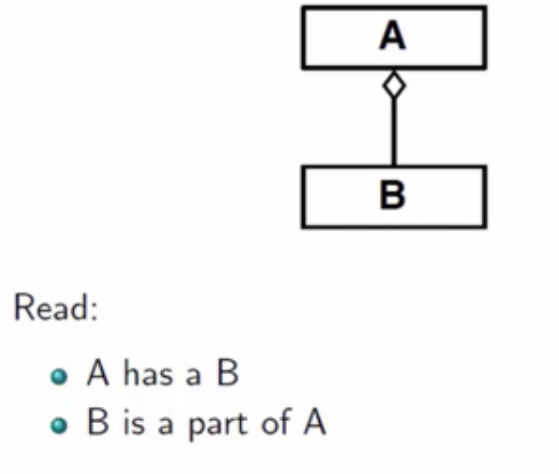
Read:

- A has a B
- B is a part of A

- Wat betekent dit?
- Elk vliegtuig of helikopter heeft een thuisbasis, ... of
- Elk vliegveld heeft een aantal standaard vliegtuigen en helikopters?

Answer





- Wat betekent dit?
- "Elk vliegtuig of helikopter heeft een thuisbasis" wordt gemodelleerd via deze aggregatie, ...
- ... wat je er niet van hoeft de weerhouden om ook de andere kant op te aggregeren!

Question

In de clip van OO design 1 staat op slide 15 dat functional design tightly coupled is. Omdat bij functional design het probleem in kleine delen wordt opgesplitst had ik verwacht dat het loose zou zijn. Waarom is dit dan wel tightly coupled?

Answer

In de clip van OO design 1 staat op slide 15 dat functional design tightly coupled is. Omdat bij functional design het probleem in kleine delen wordt opgesplitst had ik verwacht dat het loose zou zijn. Waarom is dit dan wel tightly coupled?

- Het lijkt natuurlijk logisch: je knipt je probleem op in logische deelproblemen ...
- ... maar dat hoeft niet te betekenen dat je afdoende anticipeert op *change*

Answer

In de clip van OO design 1 staat op slide 15 dat functional design tightly coupled is. Omdat bij functional design het probleem in kleine delen wordt opgesplitst had ik verwacht dat het loose zou zijn. Waarom is dit dan wel tightly coupled?

- Het lijkt natuurlijk voor de hand liggend: je knipt je probleem op in logische deelproblemen ...
- ... je onderkent de deelproblemen die op dit moment spelen
- ... maar wellicht onderken je niet dat je allerlei stilzwijgende aannames meegenomen hebt
- ... die ervan uitgaan dat de huidige situatie bindend is

Answer

In de clip van OO design 1 staat op slide 15 dat functional design tightly coupled is. Omdat bij functional design het probleem in kleine delen wordt opgesplitst had ik verwacht dat het loose zou zijn. Waarom is dit dan wel tightly coupled?

- Het lijkt natuurlijk voor de hand liggend: je knipt je probleem op in logische deelproblemen ...
- ... maar dat hoeft niet te betekenen dat je afdoende anticipeert op *change*
- Voorbeeld: de *Bridge pattern*
- Heb je voldoende geanticipeerd op nieuwe characters in je game?
- Heb je voldoende geanticipeerd op meerdere platforms?