



Lektion 3

Designmönster, analys och design

Utbildare: Mahmud Al Hakim

NACKADEMIN

Lektionstillfällets mål och metod

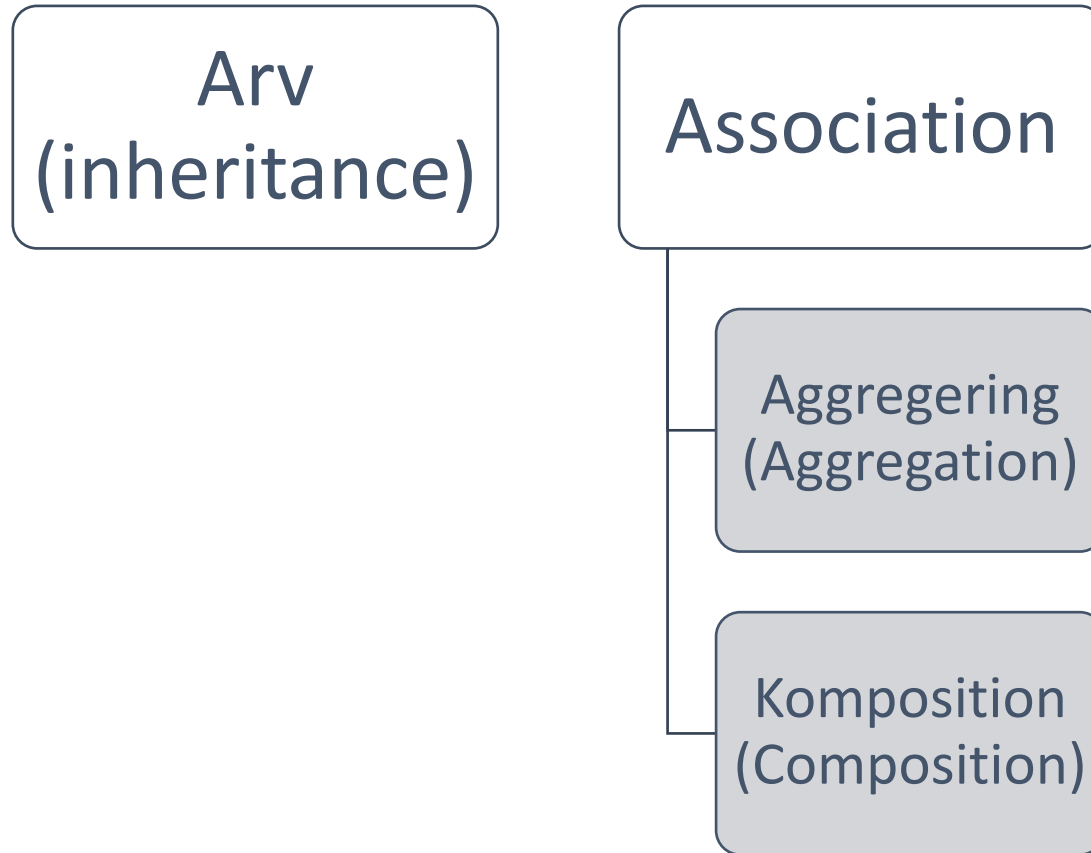
Mål med lektionen

- **Designmönster del 1 av 2**
 - Introduktion till designmönster (Design Patterns)
 - Klassificering av designmönster
 - Singleton
 - Decorator
 - Observer
- **Lektionens arbetsmetoder**
 - **Förmiddag:** Föreläsning, diskussioner och övningar.
 - **Eftermiddag:** Projektarbetet startar
 - **Individuell inlämningsuppgift delas ut**
- **Att läsa:**
 - Design Patterns in Java Tutorial (PDF)

Kort summering av föregående lektion

- **Systemdesign och Modellering**
 - Introduktion till UML
 - Modelleringsverktyg
 - Användningsfall (Use Case)
 - Användningsfallsdiagram (Use Case Diagram)
 - Klassdiagram (Class Diagram)
- Frågor?
- Reflektioner?

Relationer mellan objekt



Repetition

Bra att läsa:


<https://www.kth.se/social/files/57cea9cbf27654540ae93012/Relationer%20mellan%20objekt.pdf>

Vad är designmönster?

“Designmönster (design pattern på engelska) är en problemidentifieringsteknik inom arkitektur och programutvecklingsmetodik som innebär att man katalogiserar olika typiska problem och deras typiska lösningar.”

Källa:

<https://sv.wikipedia.org/wiki/Designm%C3%B6nster>

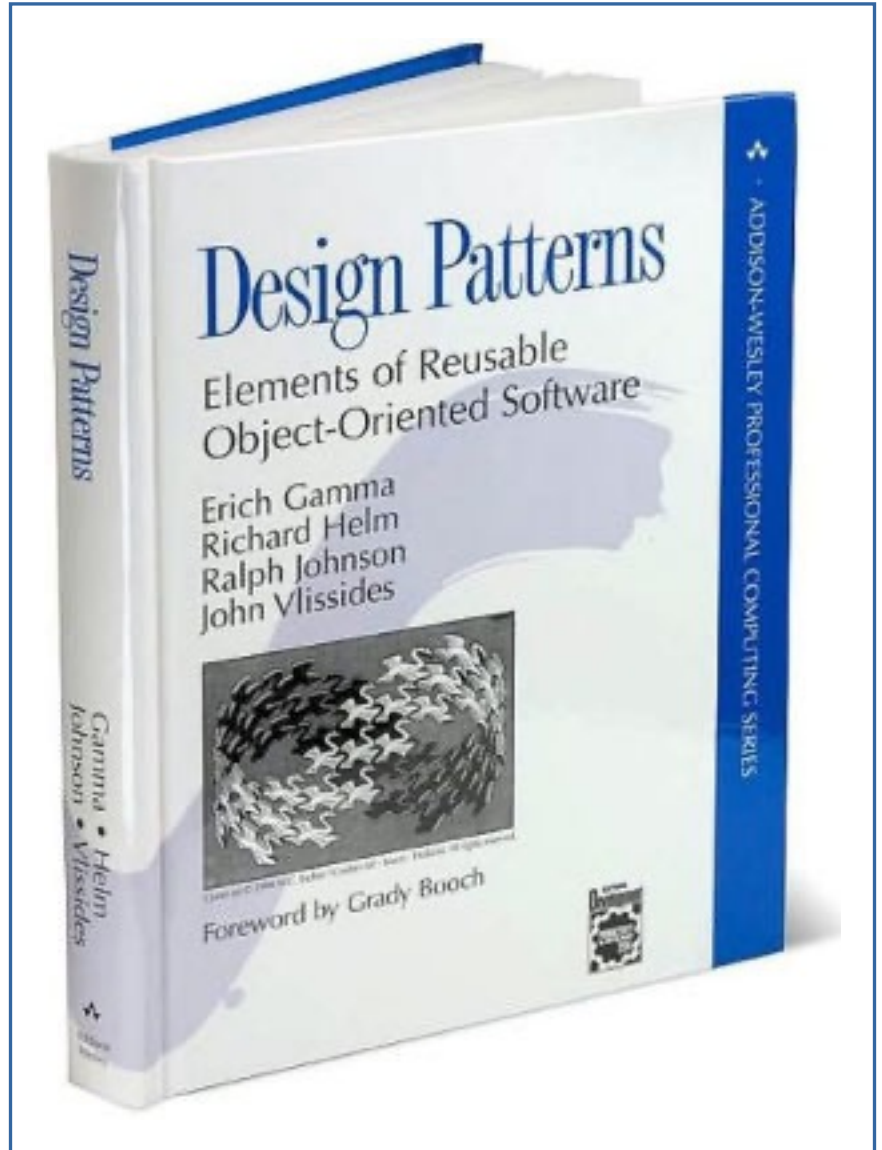


“Design Pattern is a description or template for how to solve a problem that can be used in many different situations.”

- https://en.wikipedia.org/wiki/Software_design_pattern

Gang of Four (GoF)

- Standardverket på området är ***Design Patterns: Elements of Reusable Object-Oriented Software*** (1994)
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides.
- Boken tar upp 23 designmönster som kan appliceras för att lösa vanliga problem inom objektorienterad utveckling.

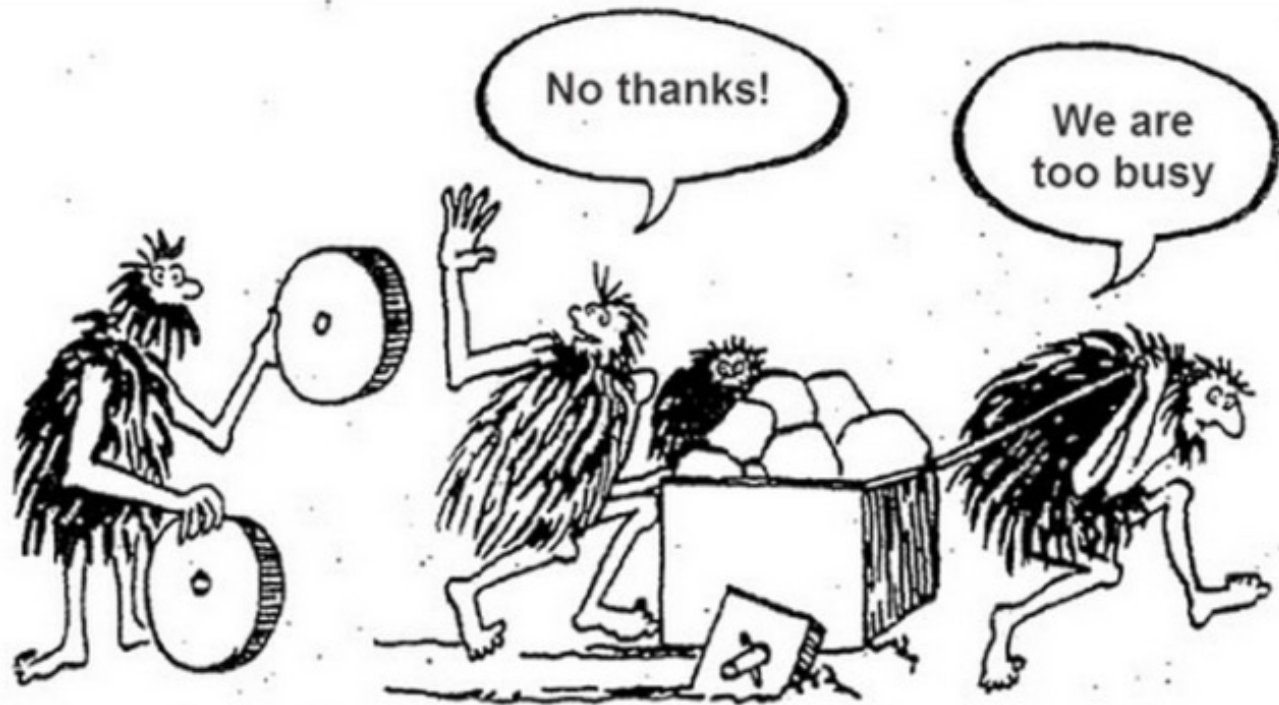


Rekommenderad bok (rolig och inspirerande)

<https://www.bokus.com/bok/9781492078005/head-first-design-patterns>



Varför behövs designmösnter?



- För att inte återuppfinna hjulet, om och om igen.
- För att bygga vidare på goda idéer som andra tänkt ut och erfårit inom programutveckling.

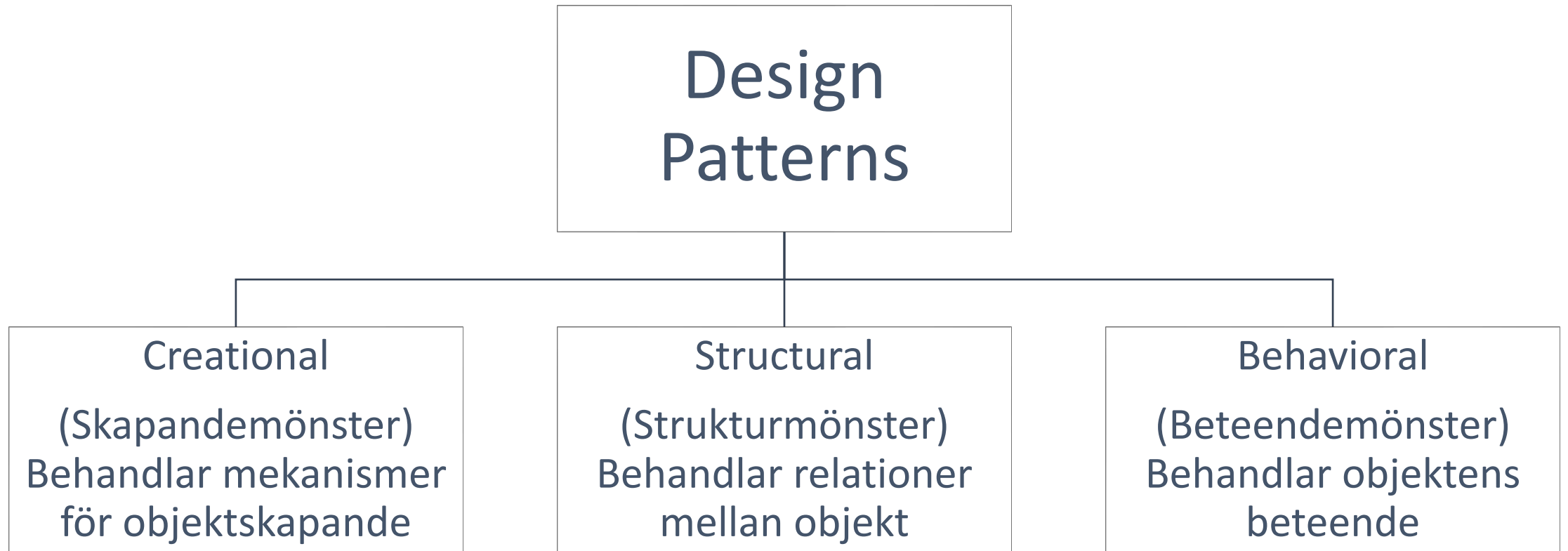
Inga garantier

“Användandet av mönster vid design och programmering ger ingen garanti för återanvändbarhet, felfrihet, robusthet eller högre produktivitet.”

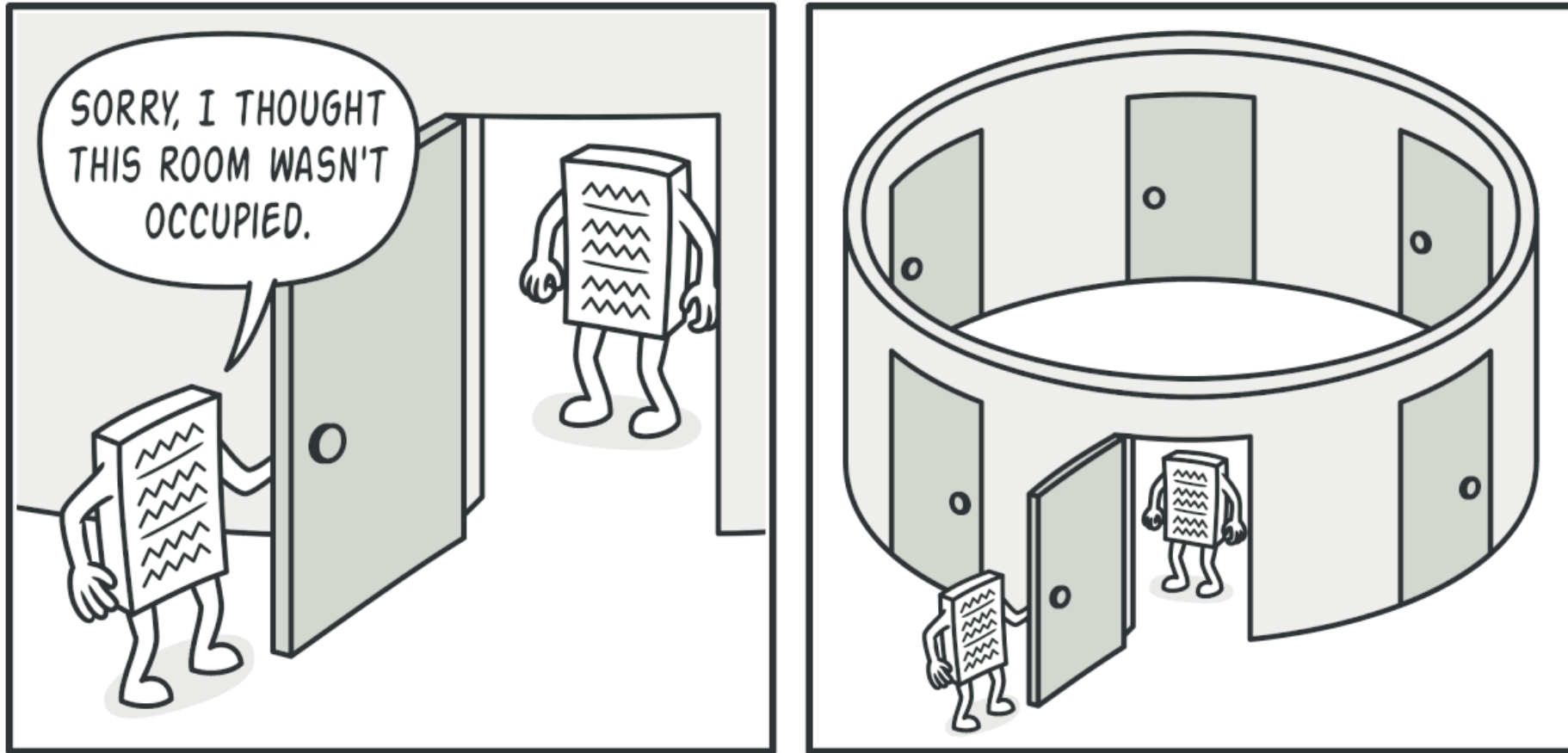
– Ulf Bilting

Designmönster för programmerare (2011)

Klassificering av designmönster



Singleton

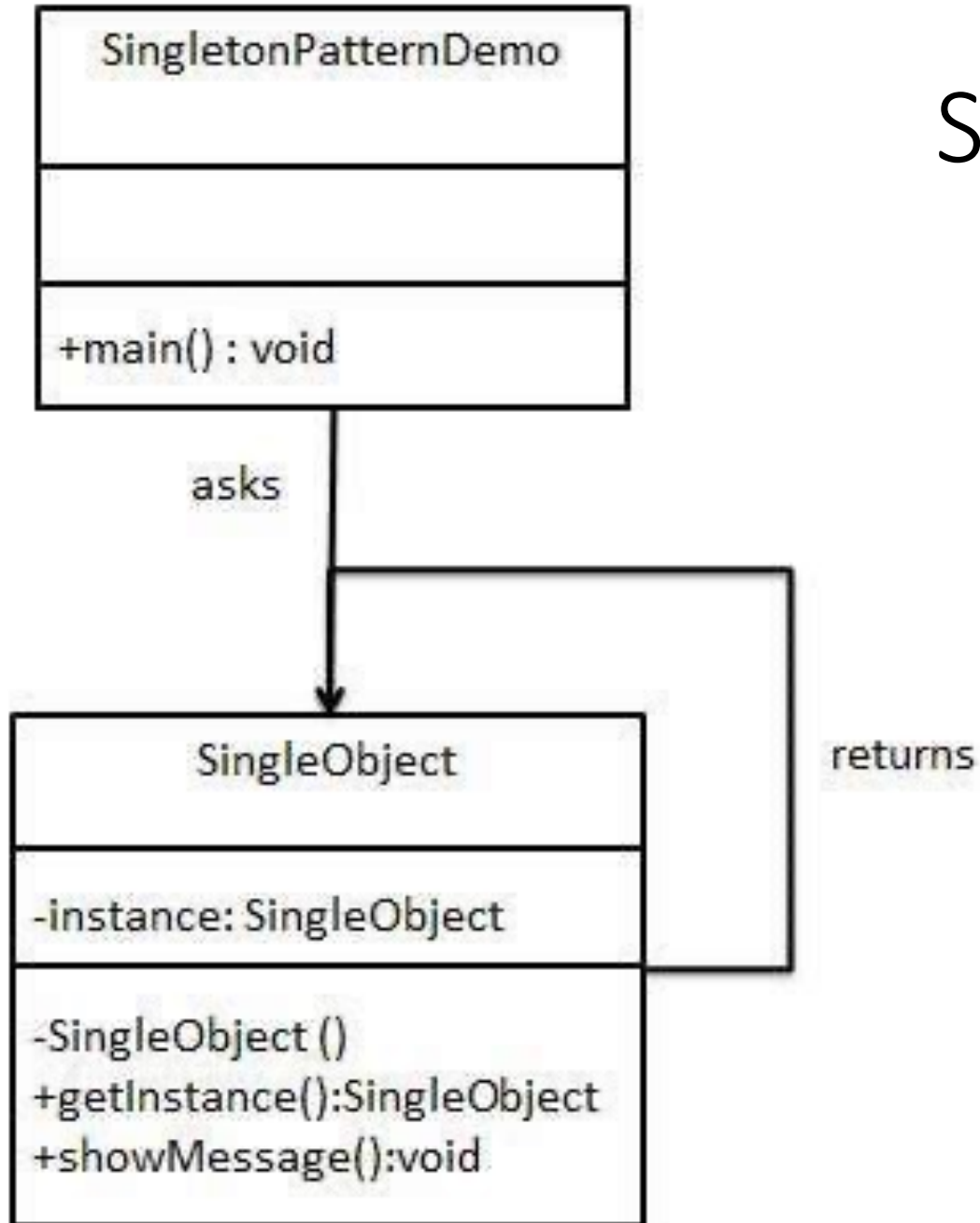


Clients may not even realize that they're working with the same object all the time.

The Singleton Pattern

- Singleton är ett skapandemönster.
- Singleton begränsar antalet instanser av en klass till ett objekt.
- Idén med en singleton är att endast skapa en instans av klassen och att den som använder klassen inte behöver veta när den skapas.
- En singleton skapas första gången när någon ber om en referens till klassen.
- Exempel från Java API
<https://docs.oracle.com/javase/8/docs/api/java/lang/Runtime.html>

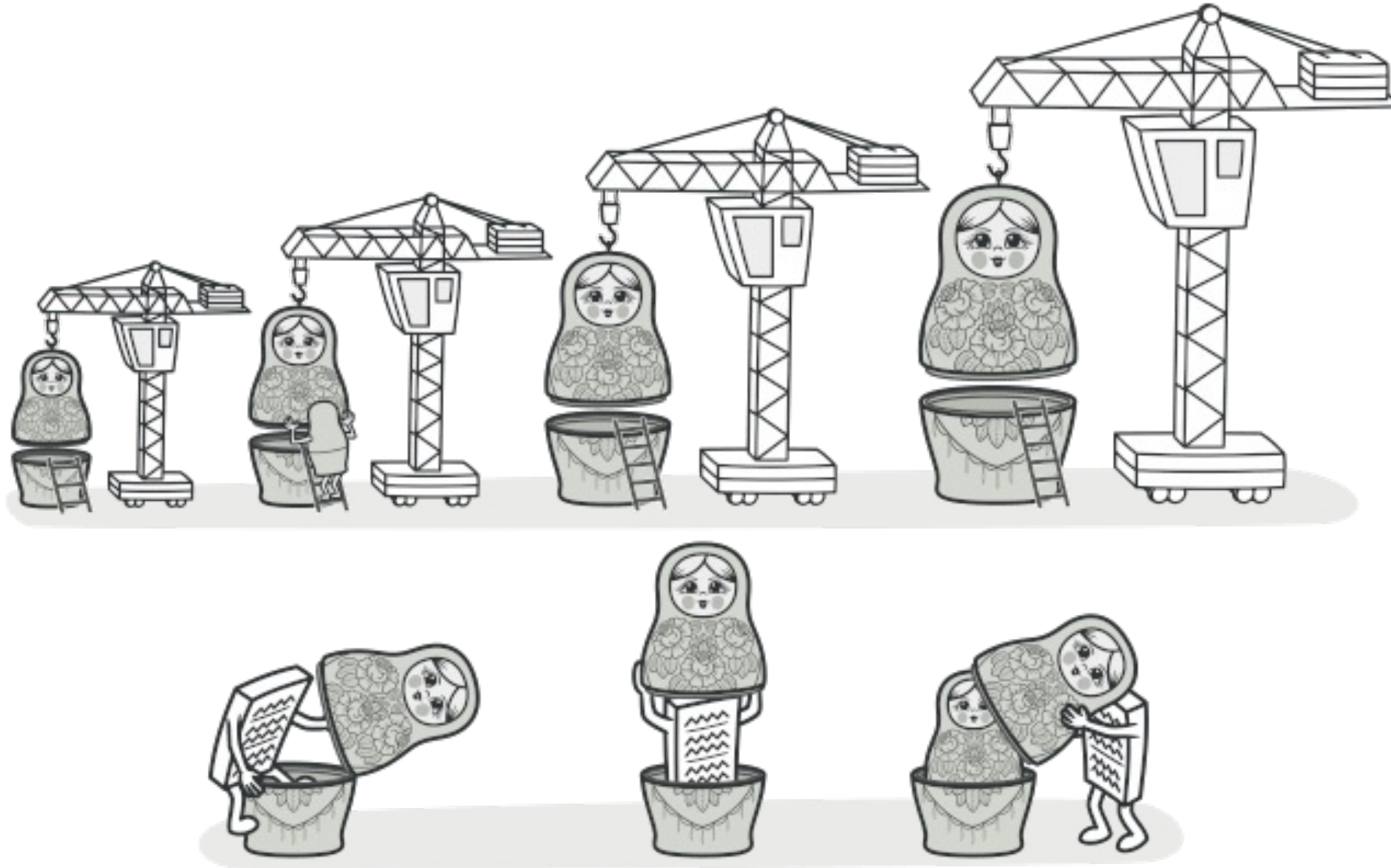
Singleton – Class Diagram



Övning i små grupper

- Implementera designmönstret singleton (skriv Javakod).
- Diskutera: Kan man skapa en singleton på ett annat sätt?

Decorator



The Decorator Pattern

Structural

- Decorator används för att utvidga (dekorera) funktionaliteten för ett valt objekt.

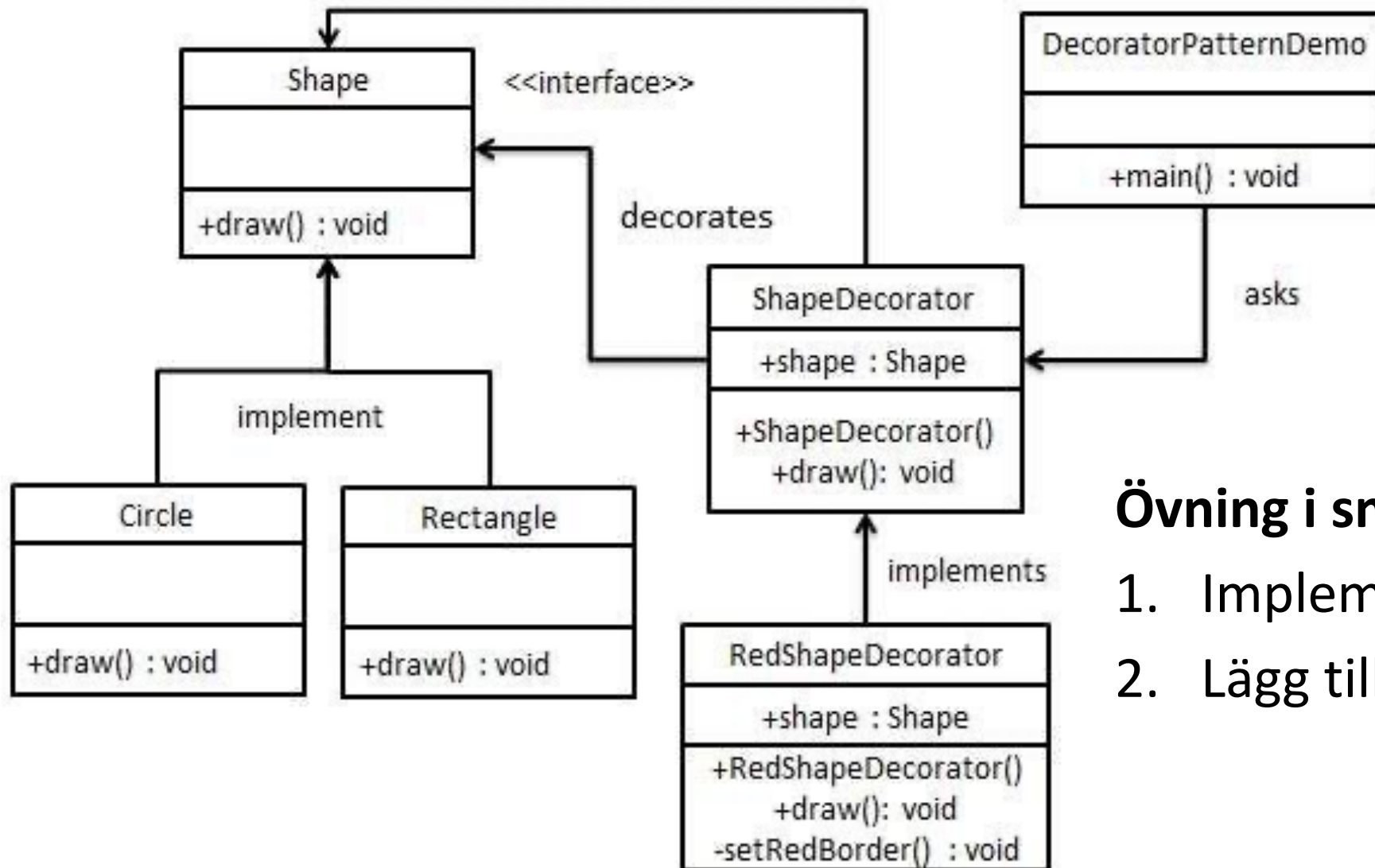
*“**Decorator** is a structural design pattern that lets you attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors”. **

** <https://refactoring.guru/design-patterns/decorator>*

Exempel från Java API

```
1 // Rekommenderat filnamn: Personer.java
2 import java.io.*;
3
4 class Personer {
5     public static void main(String[] arg) throws IOException {
6         var input = new BufferedReader
7             (new InputStreamReader(System.in));
8         PrintWriter utström = new PrintWriter
9             (new BufferedWriter
10              (new FileWriter("personer.txt", true)));
11         while(true) {
12             System.out.print("Skriv ett nytt namn? ");
13             String namn = input.readLine();
14             if (namn == null)
15                 break;
16             utström.println(namn);
17         }
18         utström.close();
19     }
20 }
```

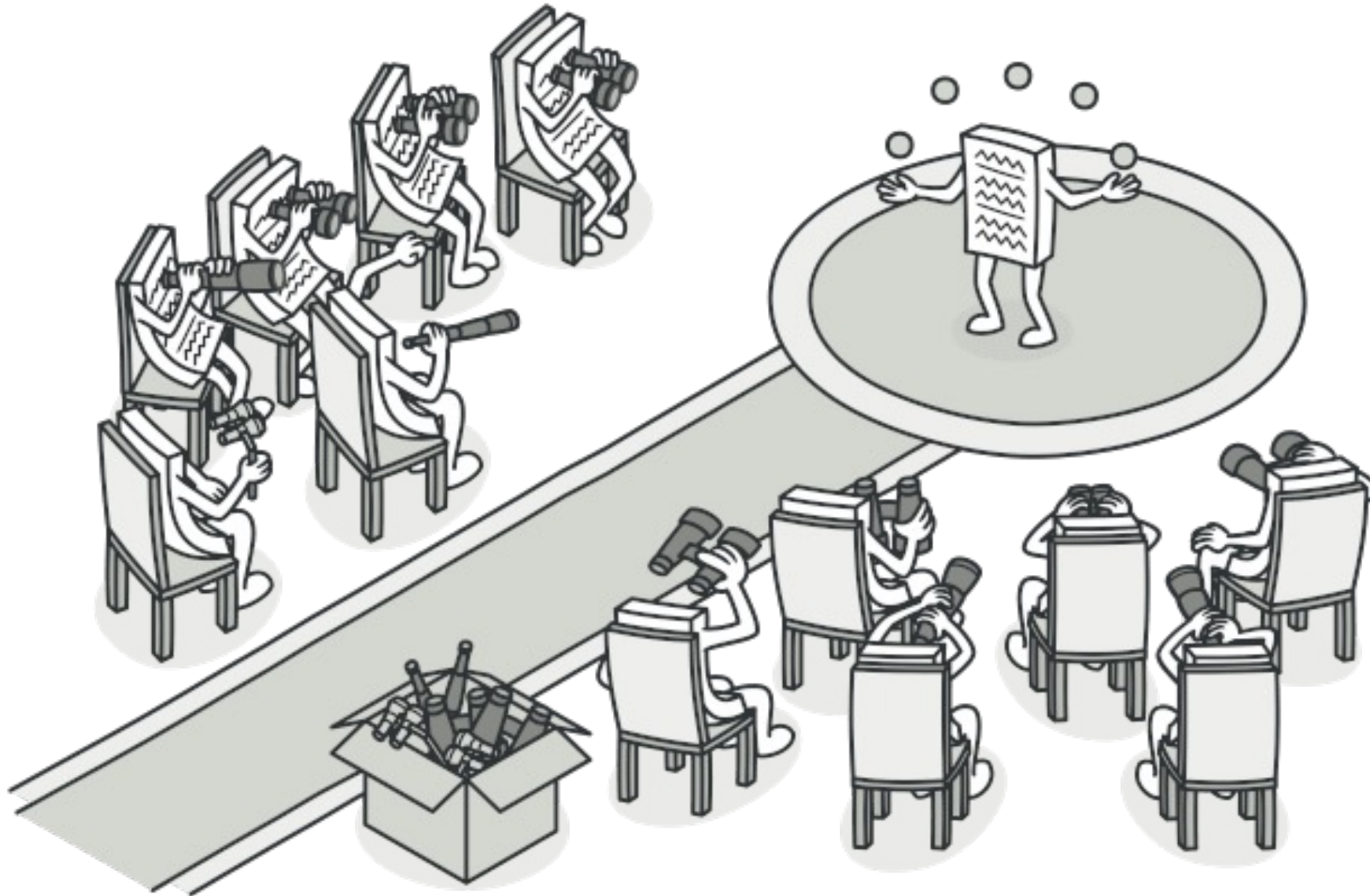
Decorator – Class Diagram



Övning i små grupper

1. Implementera designmönstret.
2. Lägg till flera funktionaliteter.

Observer



The Observer Pattern

- Syftet med en "observatör" är att skapa ett objekt som håller i en lista över beroenden, som kallas "observatörer", och meddelar dem automatiskt när någonting ändras, vanligtvis genom att anropa en av deras metoder.

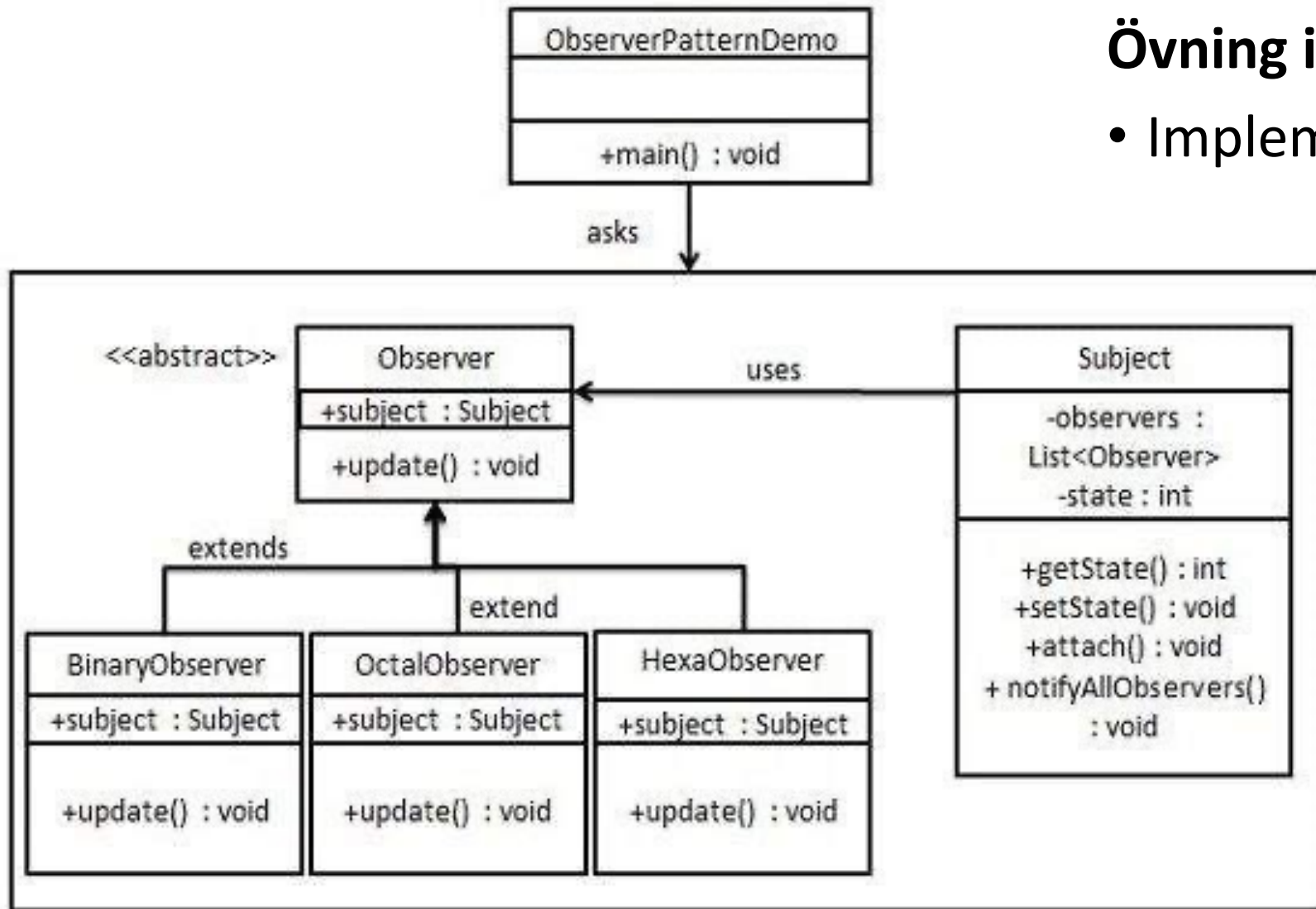
“Observer is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they’re observing.”

<https://refactoring.guru/design-patterns/observer>

The Observer Pattern – Class Diagram

Övning i små grupper

- Implementera designmönstret.



Summering av dagens lektion

- **Designmönster del 1 av 2**

- Introduktion till designmönster (Design Patterns)
- Klassificering av designmönster
- Singleton
- Decorator
- Observer

- **Reflektioner kring dagens lektion**

- Vad tar du med dig från dagens lektion?
- Finns det något som var extra svårt att förstå?
- Finns det något som vi behöver repetera?
- Hur upplevde du dagens arbetsmetoder?

Framåtblick inför nästa lektion

- **Förmiddag**
 - Designmönster del 2 av 2
- **Eftermiddag**
 - Arbeta vidare med projektarbetet.
- **Att läsa**
 - Design Patterns in Java Tutorial