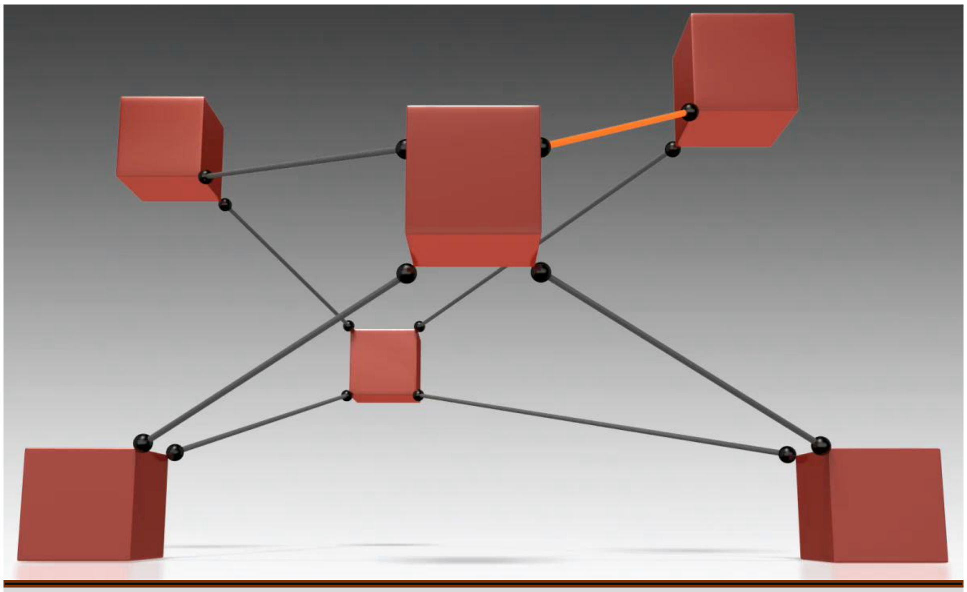
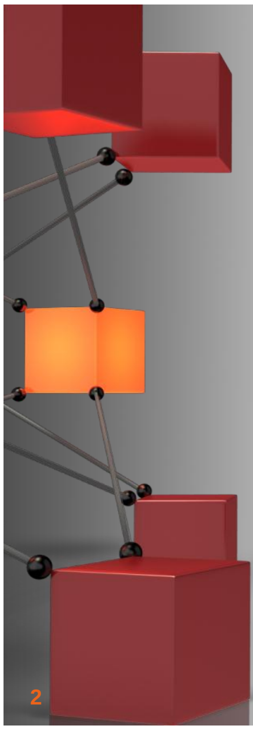


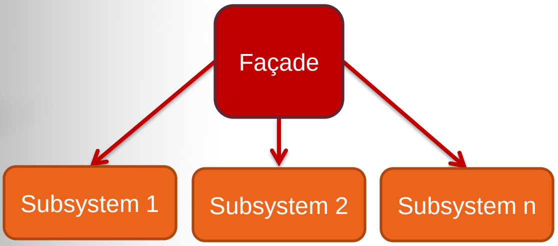
Façade Design Pattern

COS 121 – Christoph Stallmann



Introduction

- A Façade provides an interface to subsystems and classes.
- Classification: Behavioural
- Strategy: Delegation

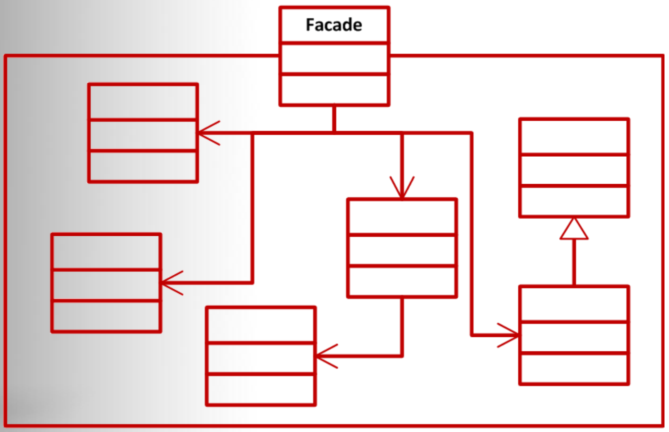


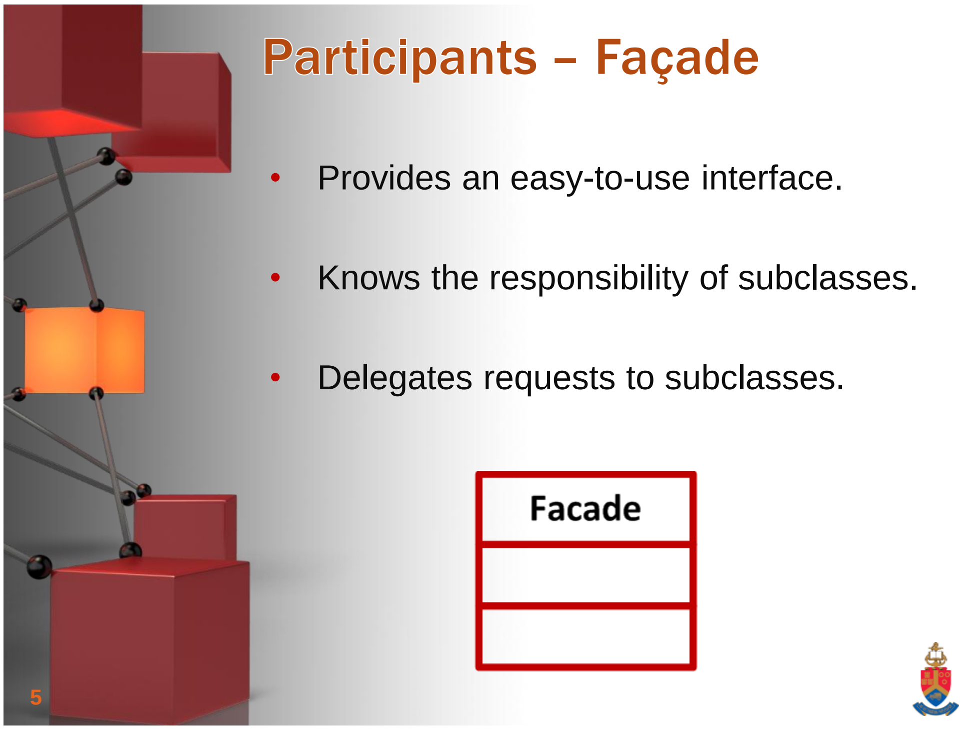
2

Reason

- Reusable and generic systems are often complex.
- Applying design patterns often result in many small classes.
- Third parties want to use code without understanding the complexity behind it.

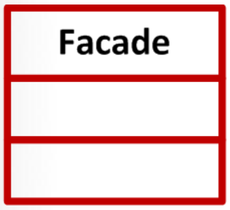
Structure



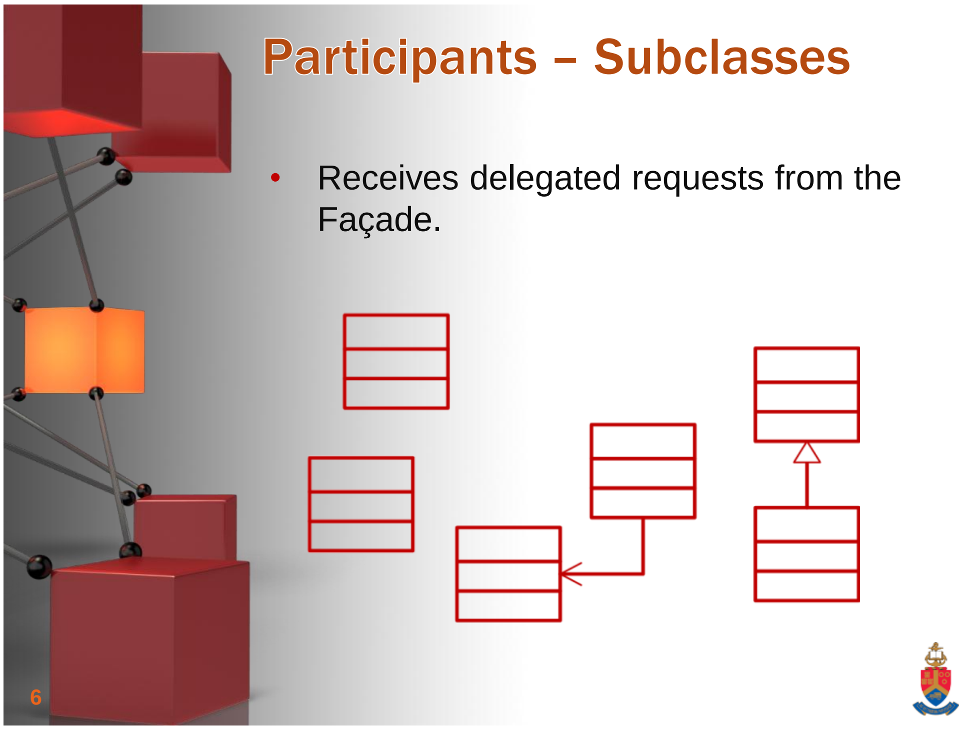


Participants – Façade

- Provides an easy-to-use interface.
- Knows the responsibility of subclasses.
- Delegates requests to subclasses.

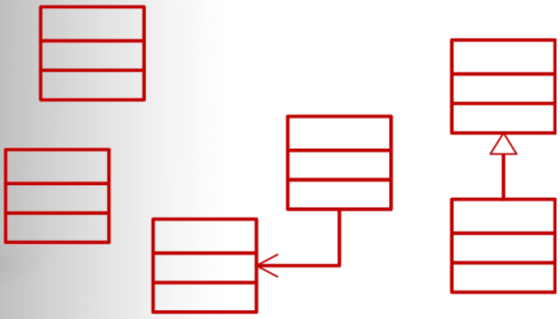


5

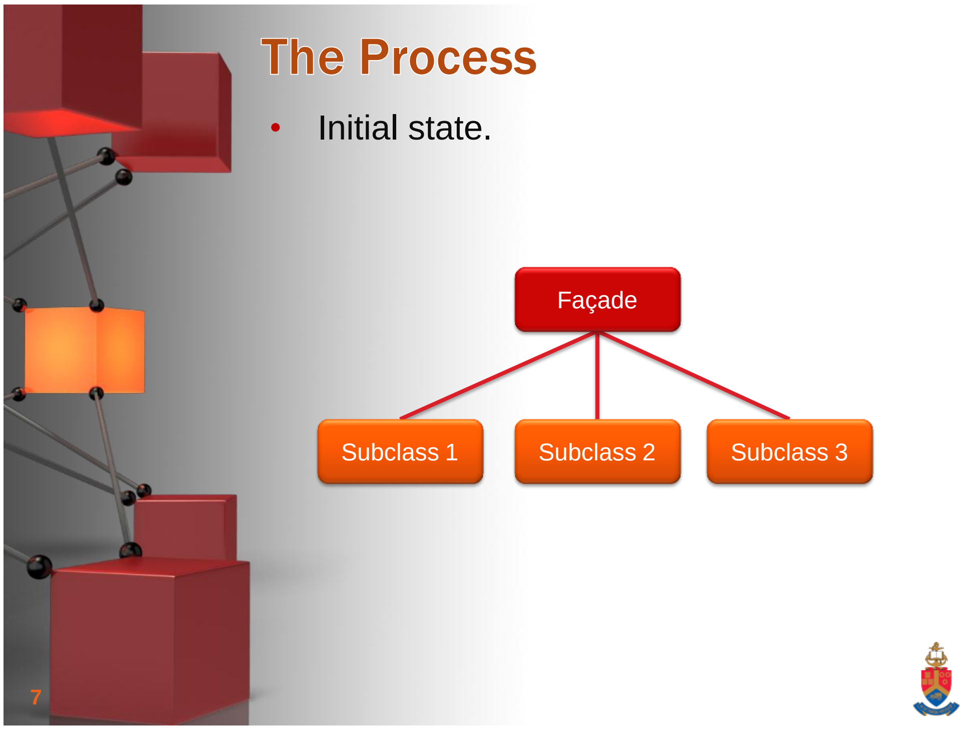


Participants – Subclasses

- Receives delegated requests from the Façade.



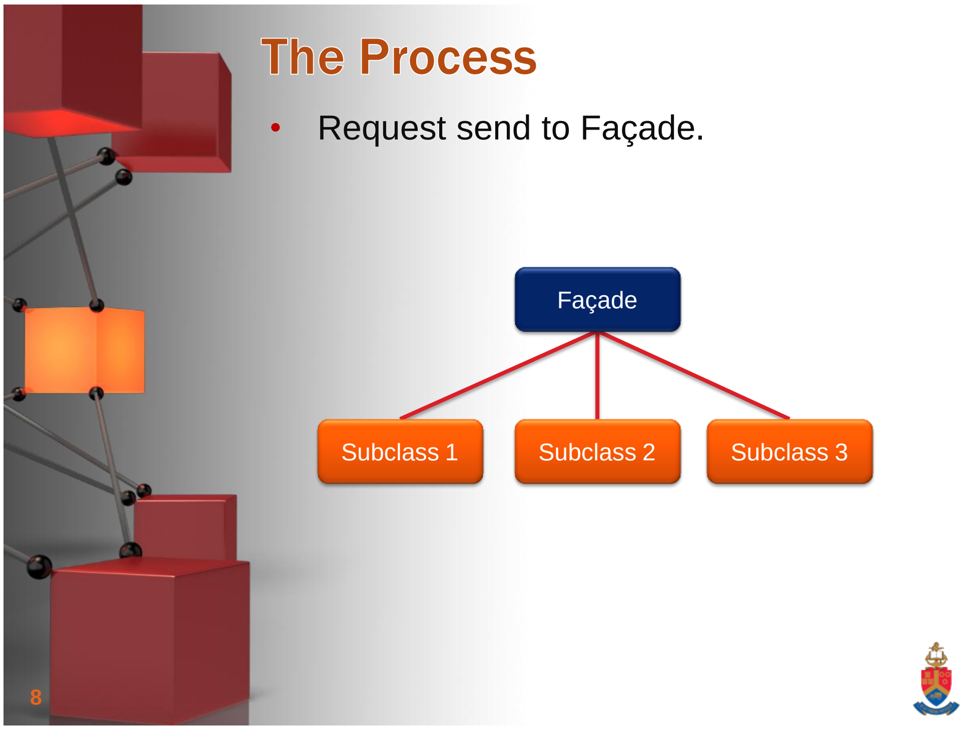
6



The Process

- Initial state.

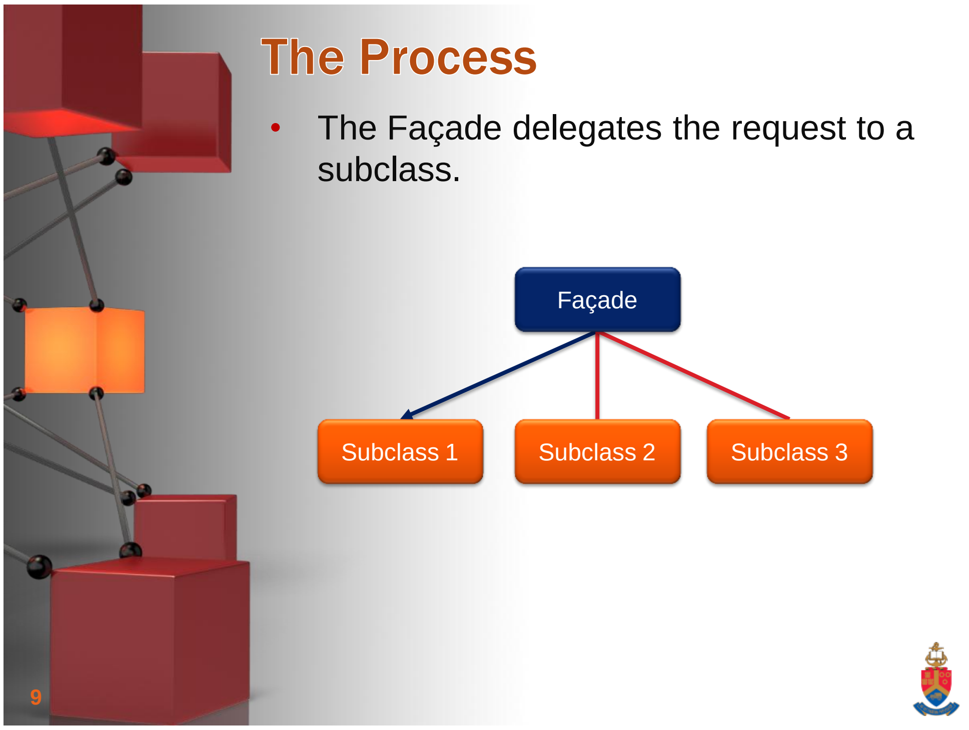
7



The Process

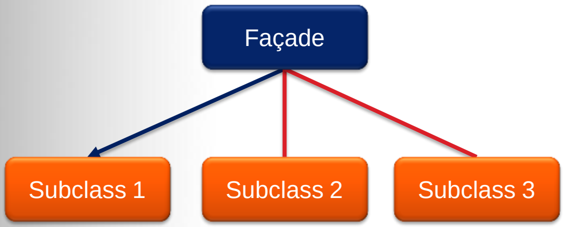
- Request send to Façade.

8

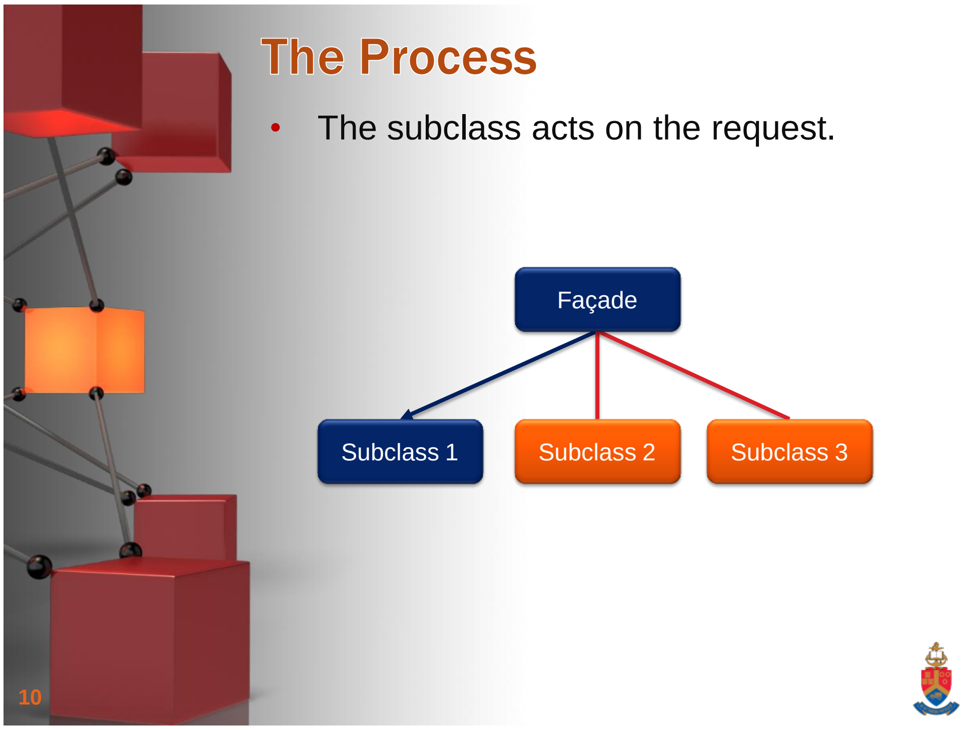


The Process

- The Façade delegates the request to a subclass.

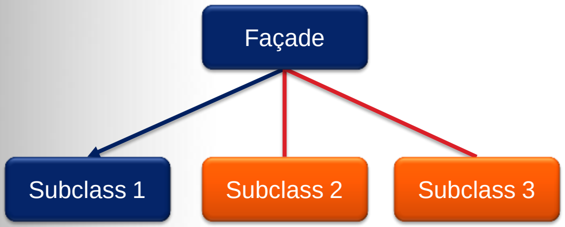


9

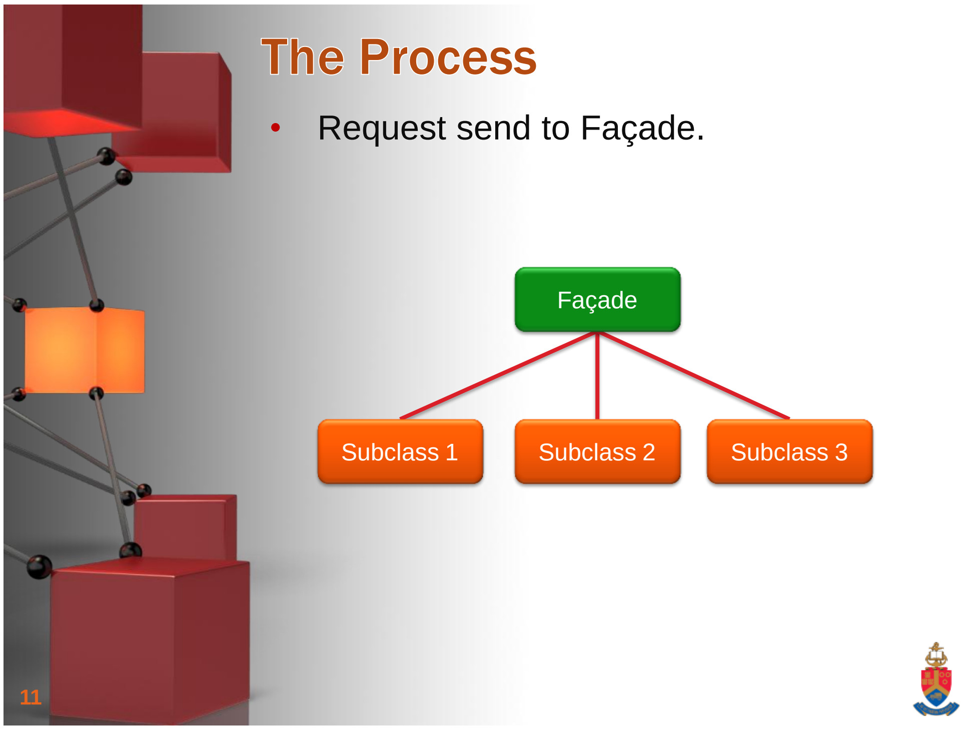


The Process

- The subclass acts on the request.

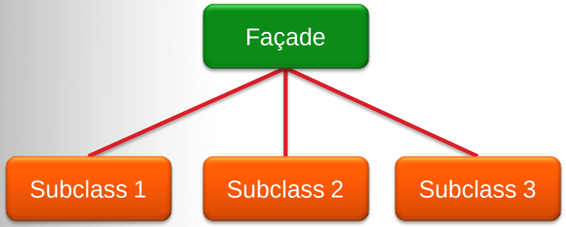


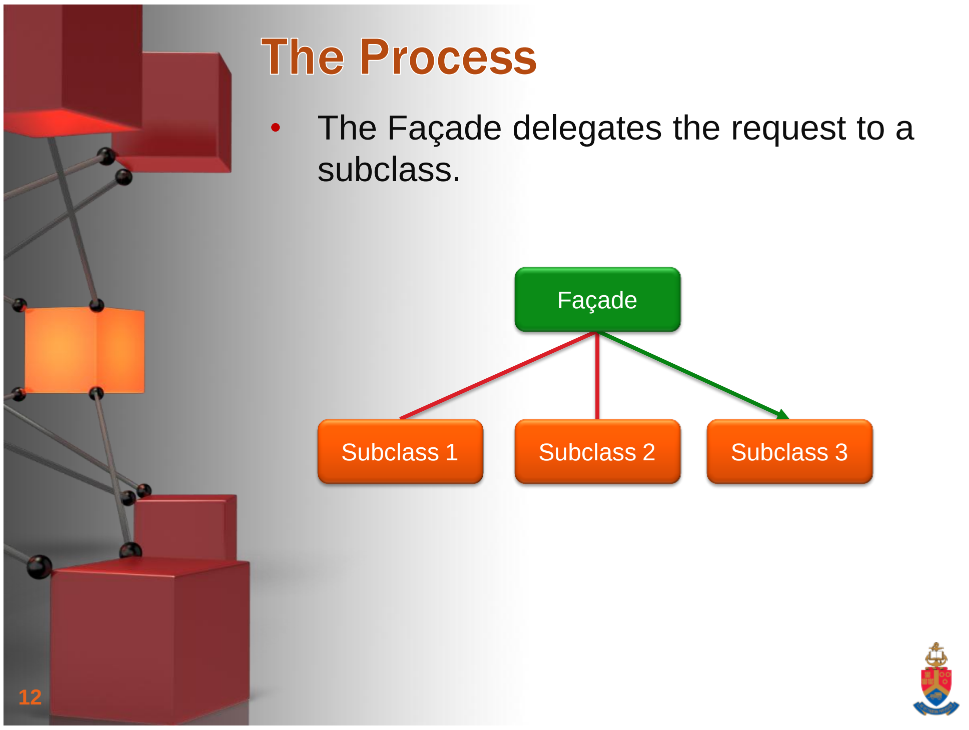
10



The Process

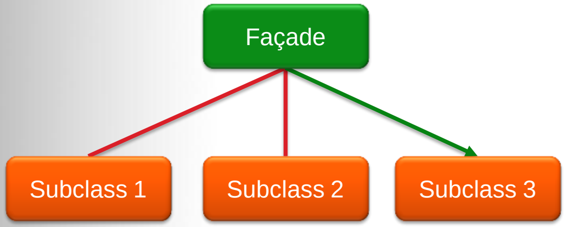
- Request send to Façade.

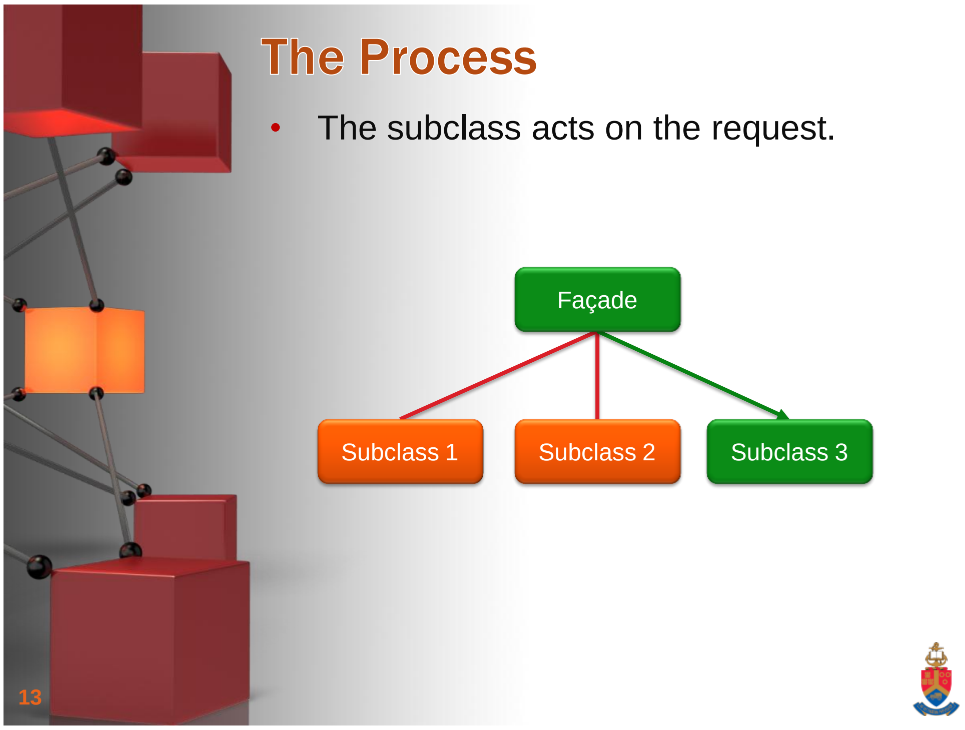




The Process

- The Façade delegates the request to a subclass.



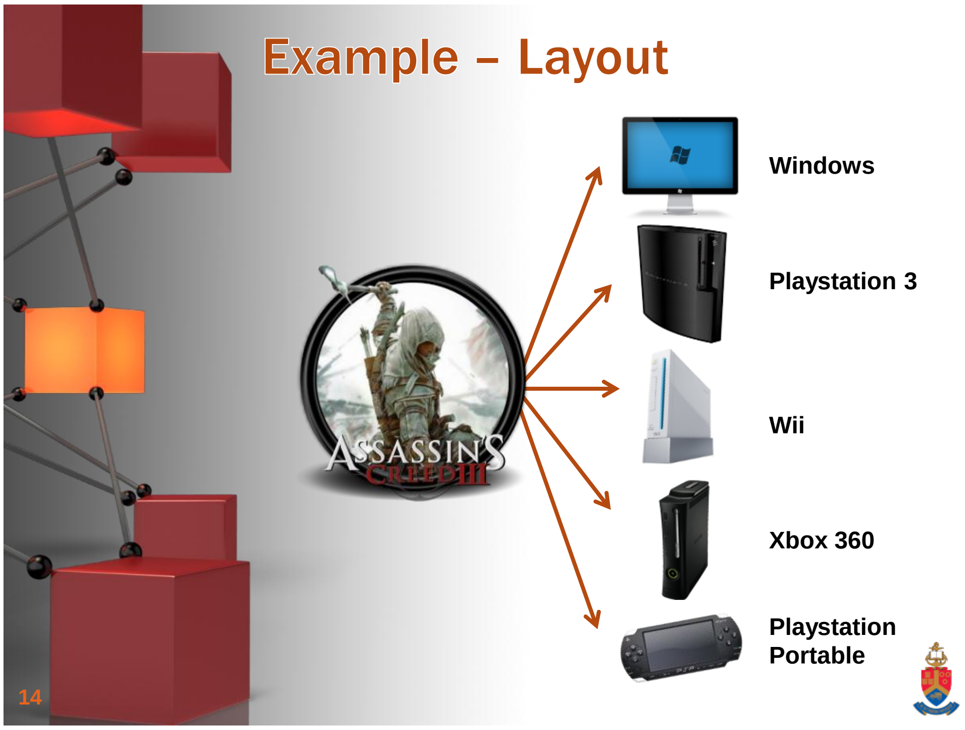


The Process

- The subclass acts on the request.

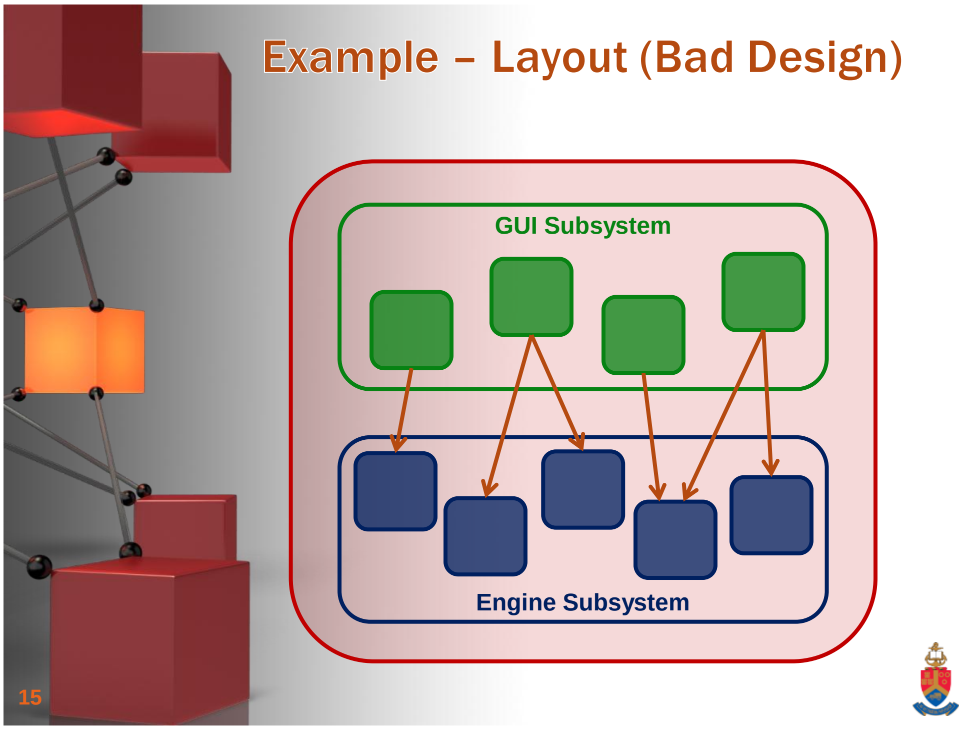
$$\begin{tikzcd}
 & \text{Façade} \arrow[dl, dash] \arrow[d, dash] \arrow[dr] & \\
\text{Subclass 1} & \text{Subclass 2} & \text{Subclass 3}
\end{tikzcd}$$

13

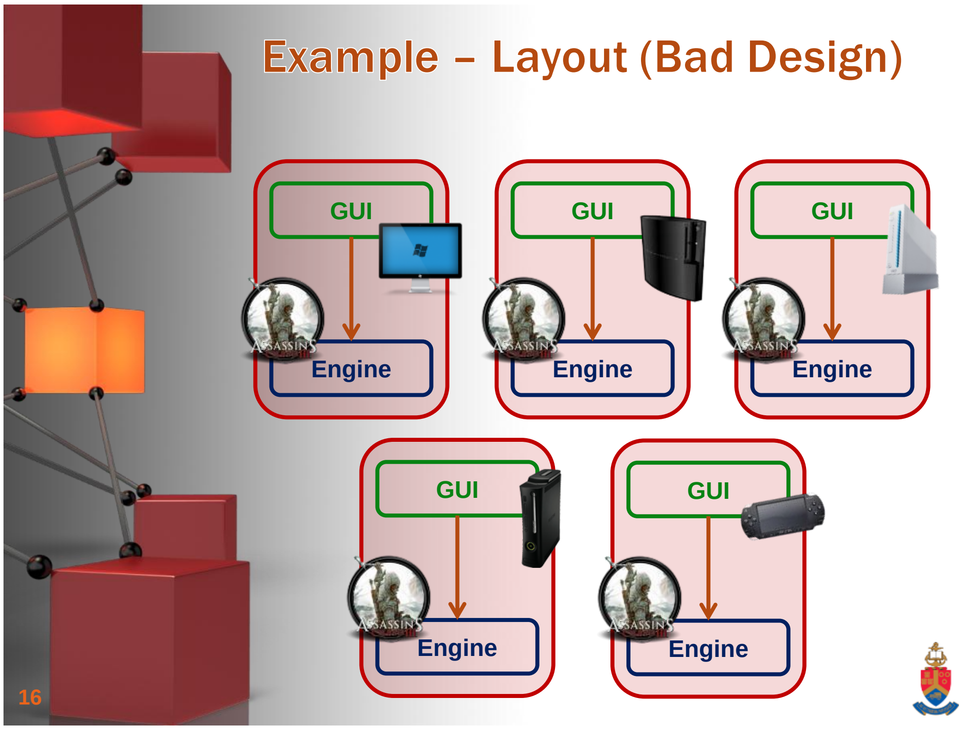


Example – Layout

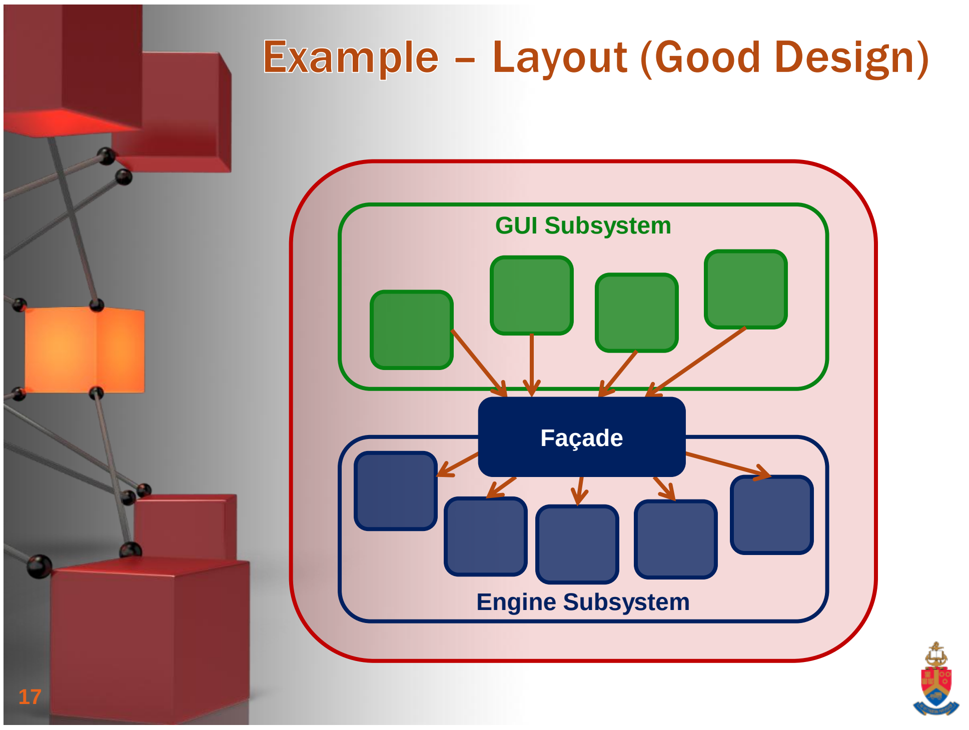
14



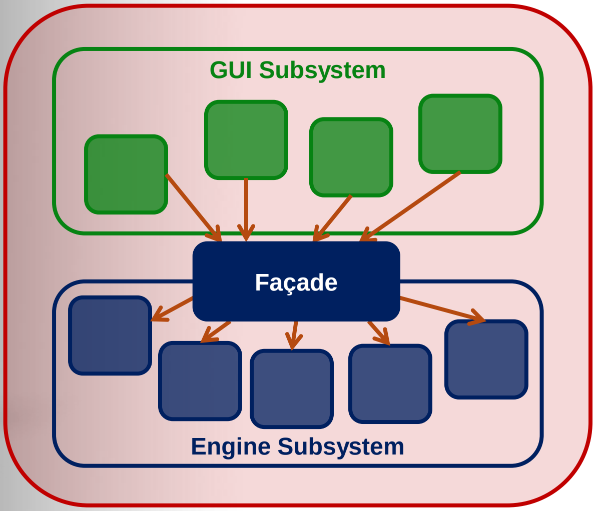
Example – Layout (Bad Design)



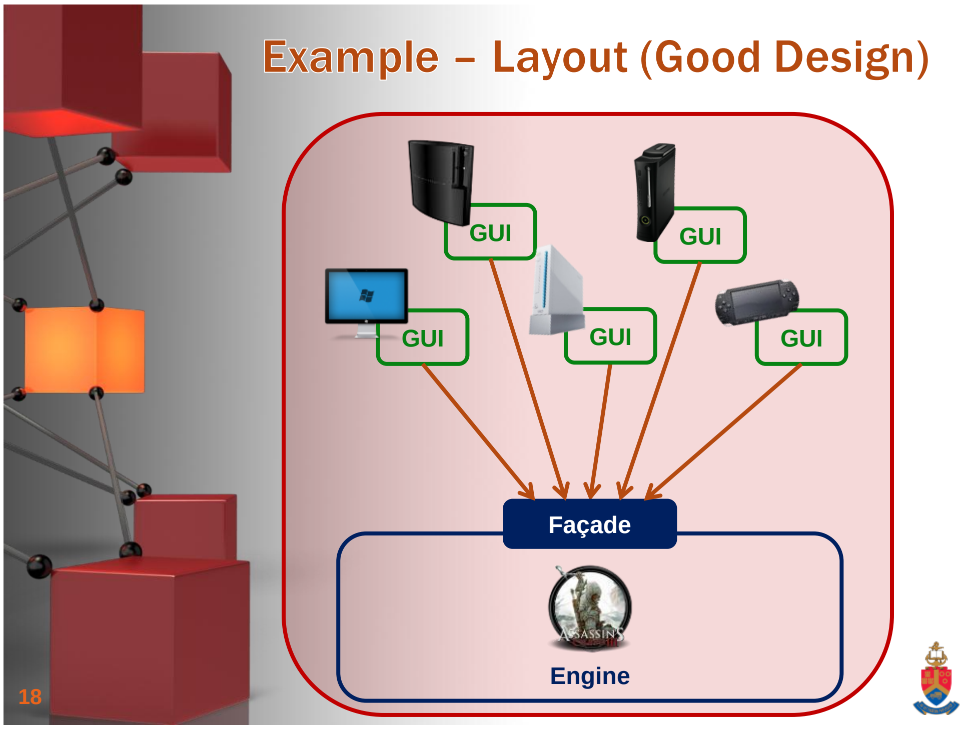
Example – Layout (Bad Design)



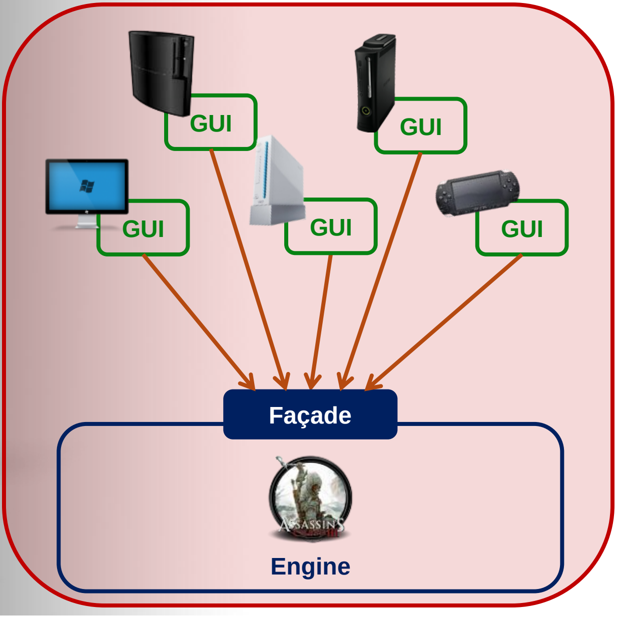
Example – Layout (Good Design)



17



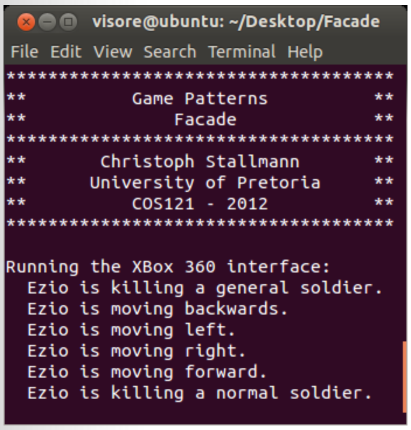
Example – Layout (Good Design)



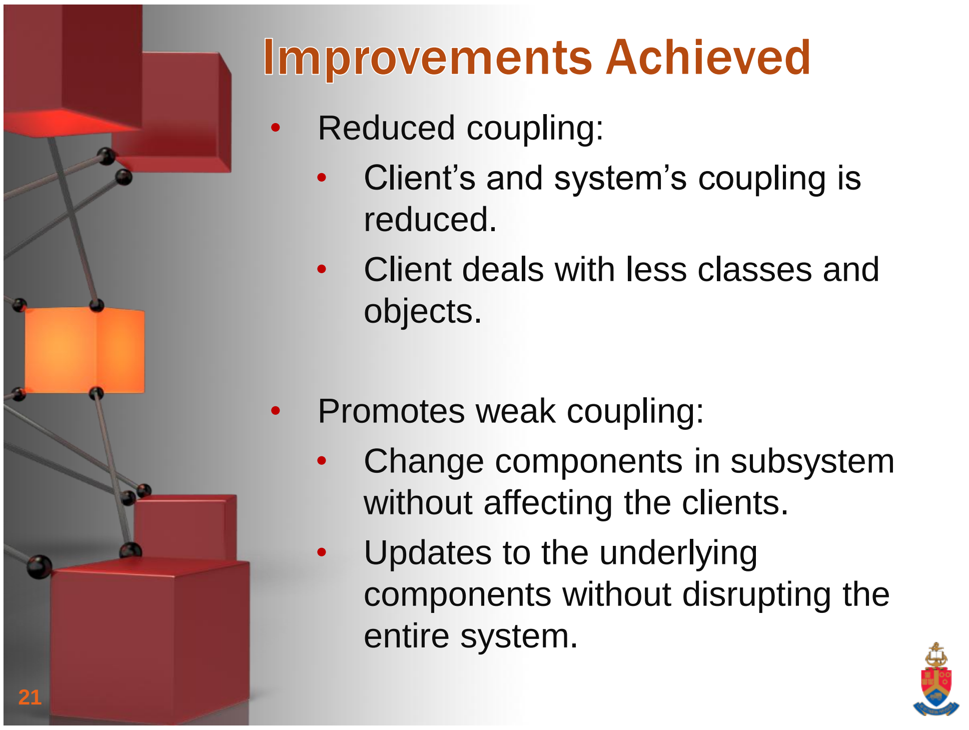
18

Example – Code

Example – Output


visore@ubuntu: ~/Desktop/Facade

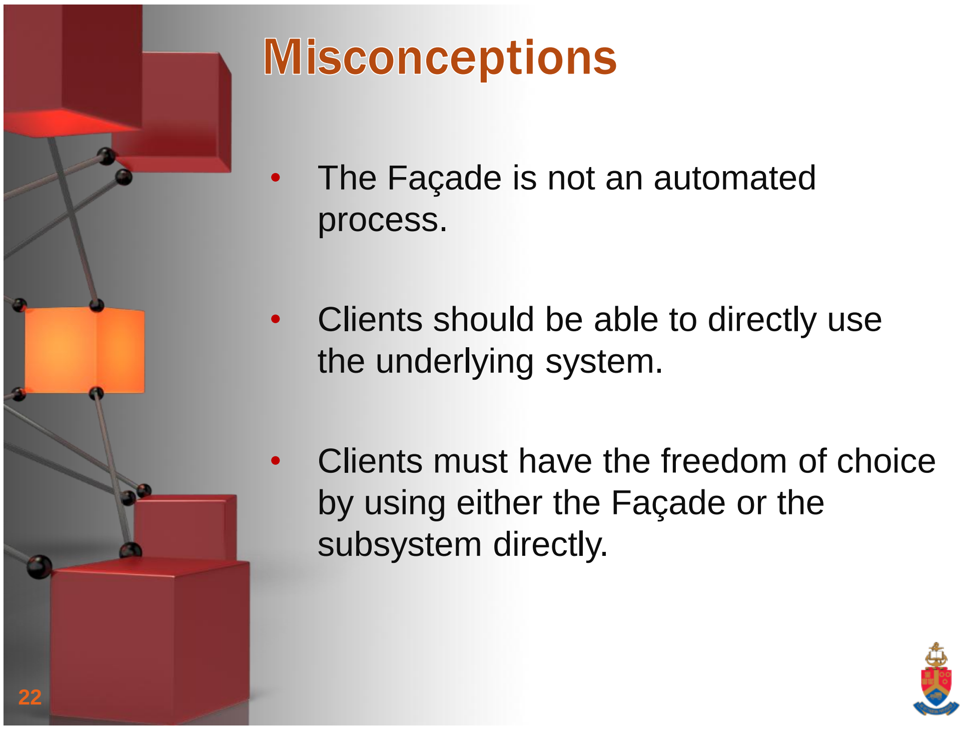
```
*************************************
**          Game Patterns          **
**              Facade             **
*************************************
**       Christoph Stallmann       **
**      University of Pretoria     **
**          COS121 - 2012          **
*************************************

Running the XBox 360 interface:
  Ezio is killing a general soldier.
  Ezio is moving backwards.
  Ezio is moving left.
  Ezio is moving right.
  Ezio is moving forward.
  Ezio is killing a normal soldier.
```



Improvements Achieved

- Reduced coupling:
  - Client's and system's coupling is reduced.
  - Client deals with less classes and objects.
- Promotes weak coupling:
  - Change components in subsystem without affecting the clients.
  - Updates to the underlying components without disrupting the entire system.

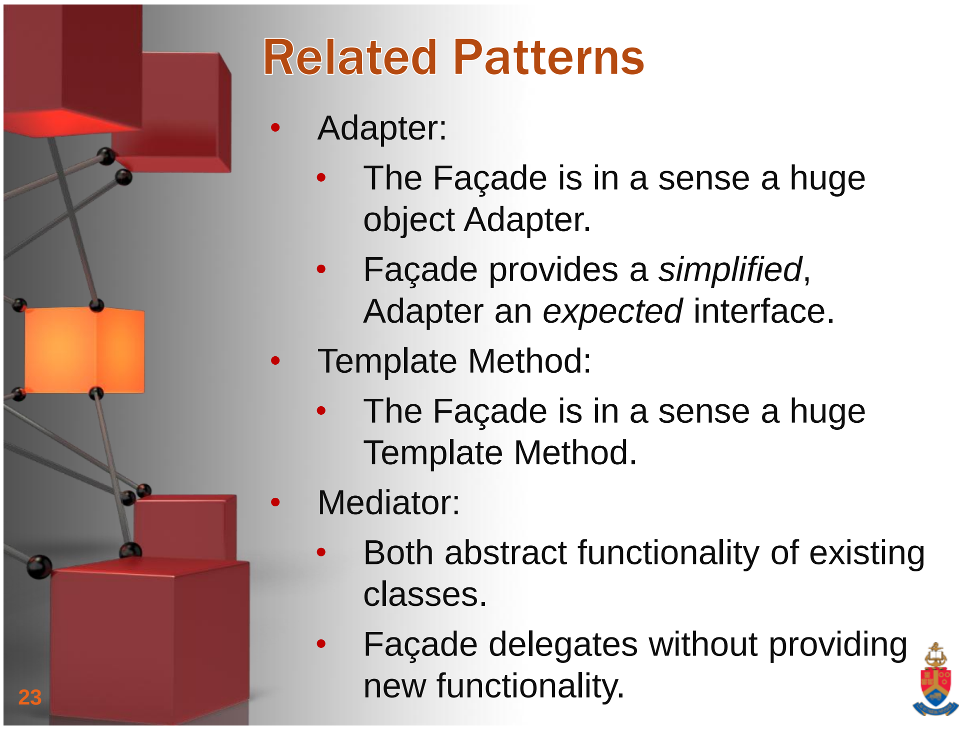
21



Misconceptions

- The Façade is not an automated process.
- Clients should be able to directly use the underlying system.
- Clients must have the freedom of choice by using either the Façade or the subsystem directly.

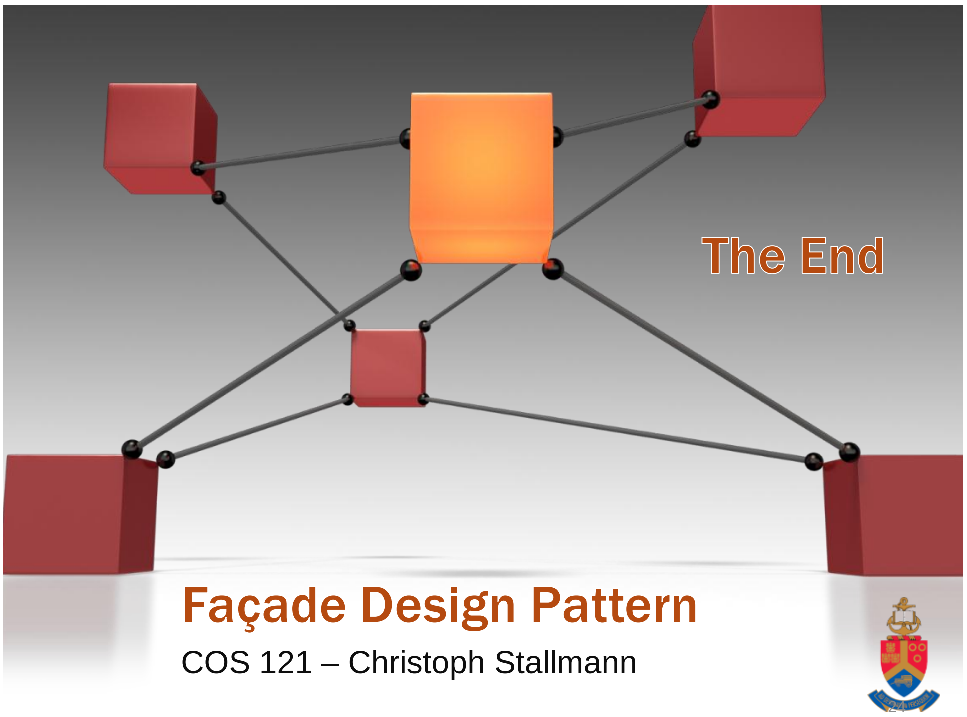
22



Related Patterns

- Adapter:
  - The Façade is in a sense a huge object Adapter.
  - Façade provides a *simplified*, Adapter an *expected* interface.
- Template Method:
  - The Façade is in a sense a huge Template Method.
- Mediator:
  - Both abstract functionality of existing classes.
  - Façade delegates without providing new functionality.

23



The End

Façade Design Pattern

COS 121 – Christoph Stallmann