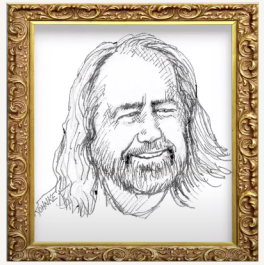
DEPENDENCY INVERSION PRINCIPLE

Clean Code



Gary Booch

- Simple & Direct
- Reads as well-written prose

Bjarne Stroustrup

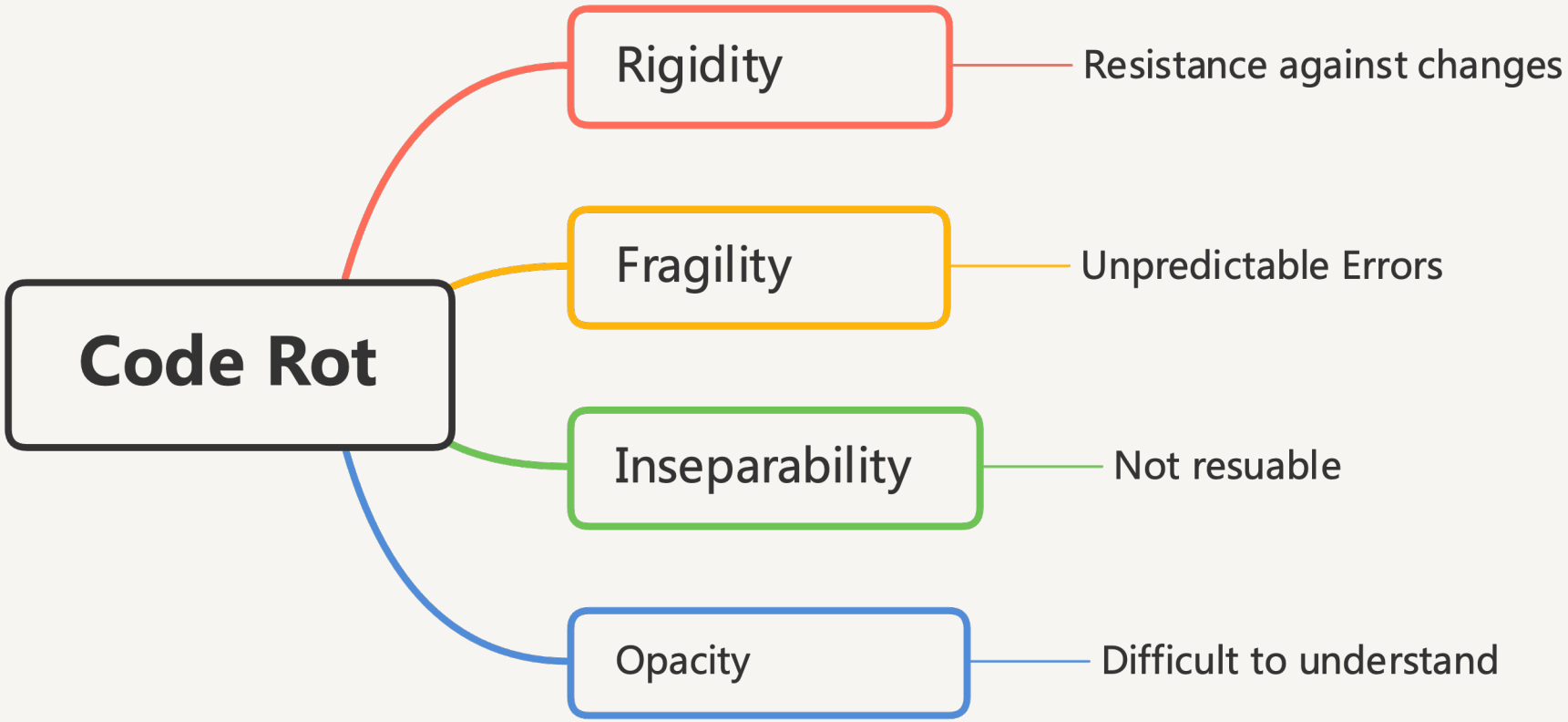
- Elegant & Efficient.
- Should do one thing

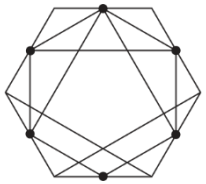
Ward Cunnigham

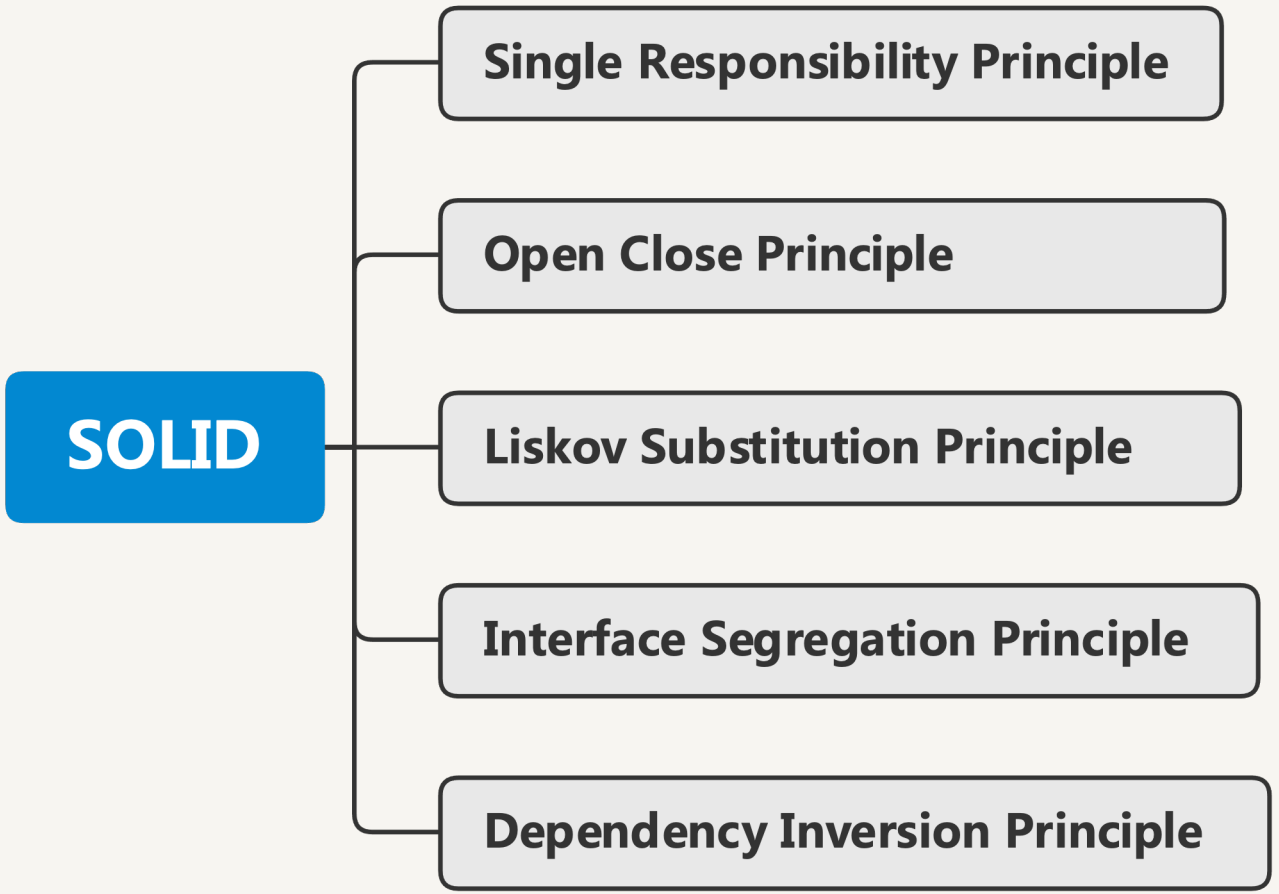
- Every Routine is pretty much what you expected

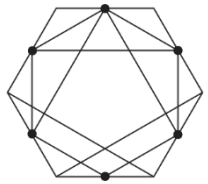
Michael Feathers

- Written by someone who cares

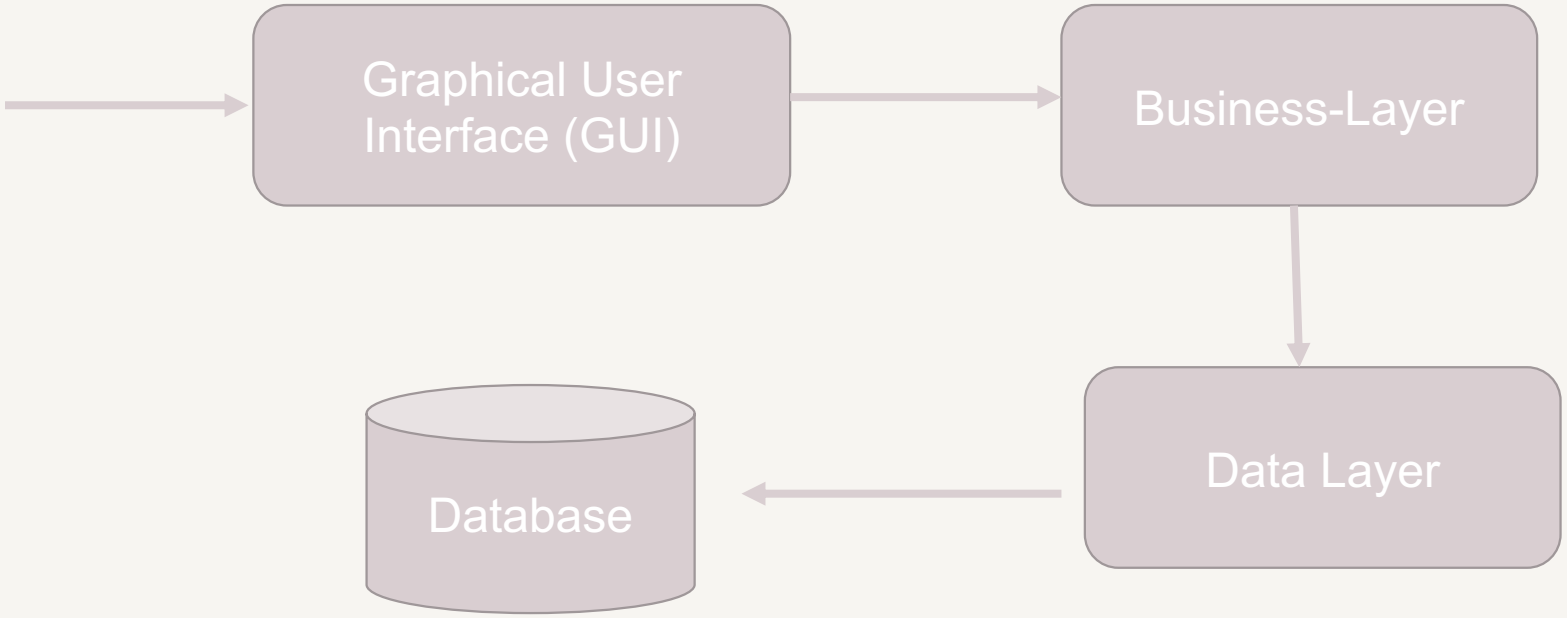








CLASSIC PROGRAM



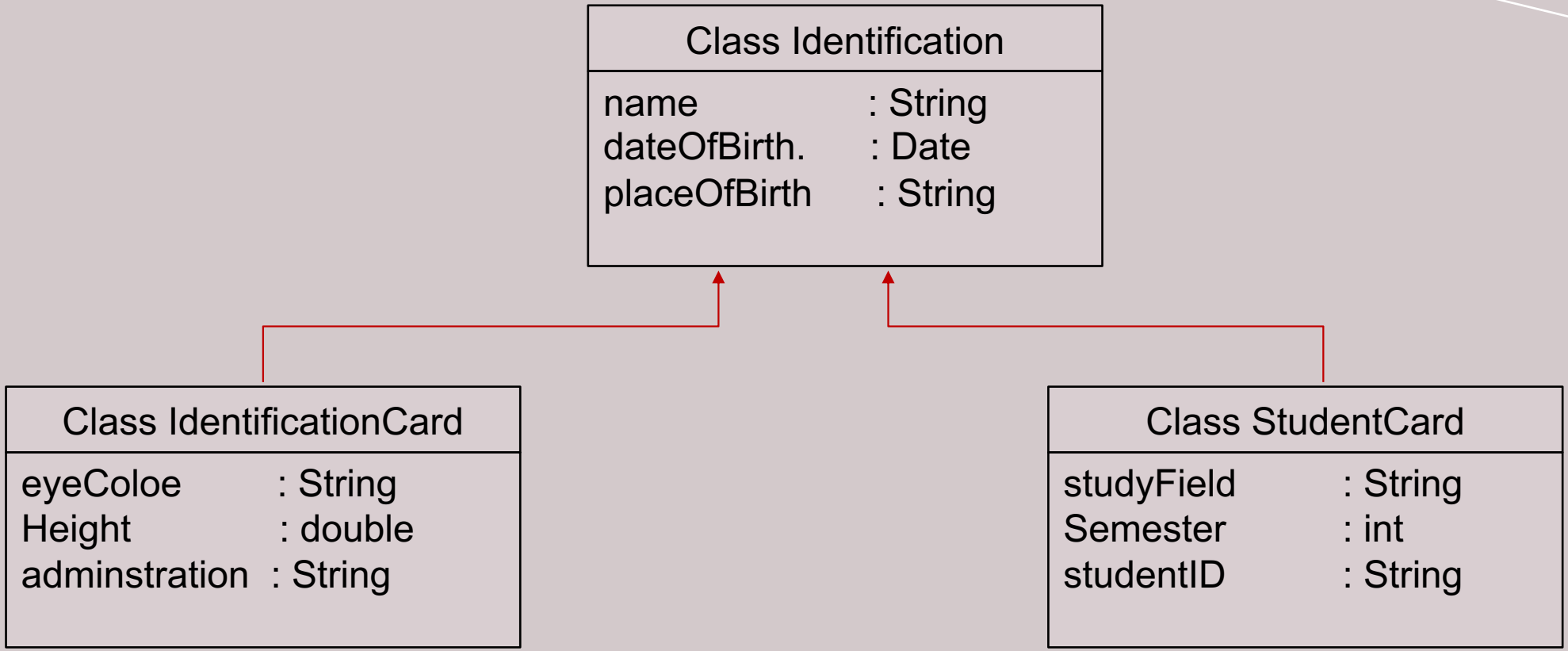
STUDENT

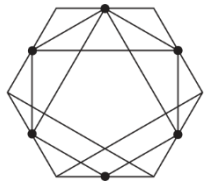
- Name
- Date of Birth
- Place of Birth
- Study
- Semester
- Student ID

CITIZEN

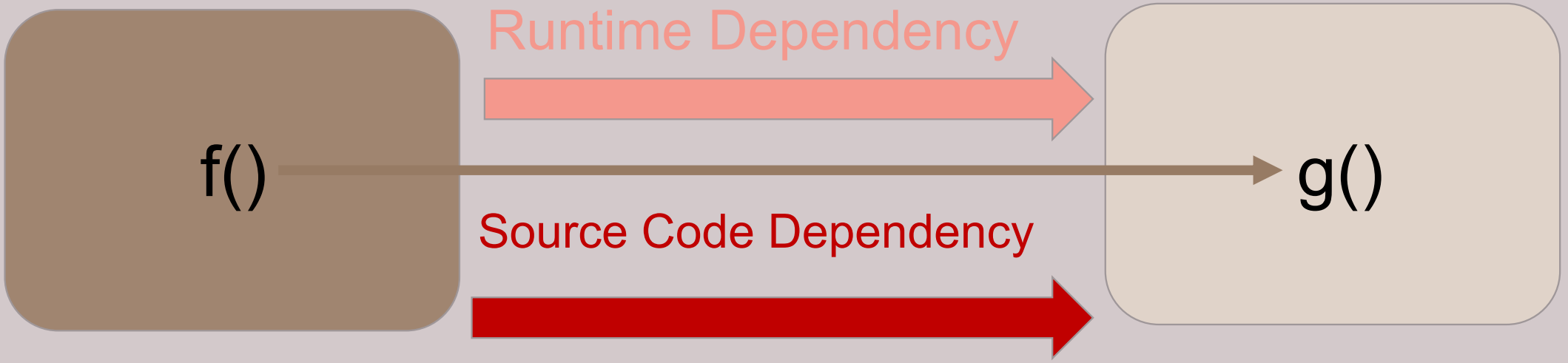
- Name
- Date of Birth
- Place of Birth
- Eye Color
- height
- Adminstration

INHERITANCE





WAS IST OOP WIRKLICH?

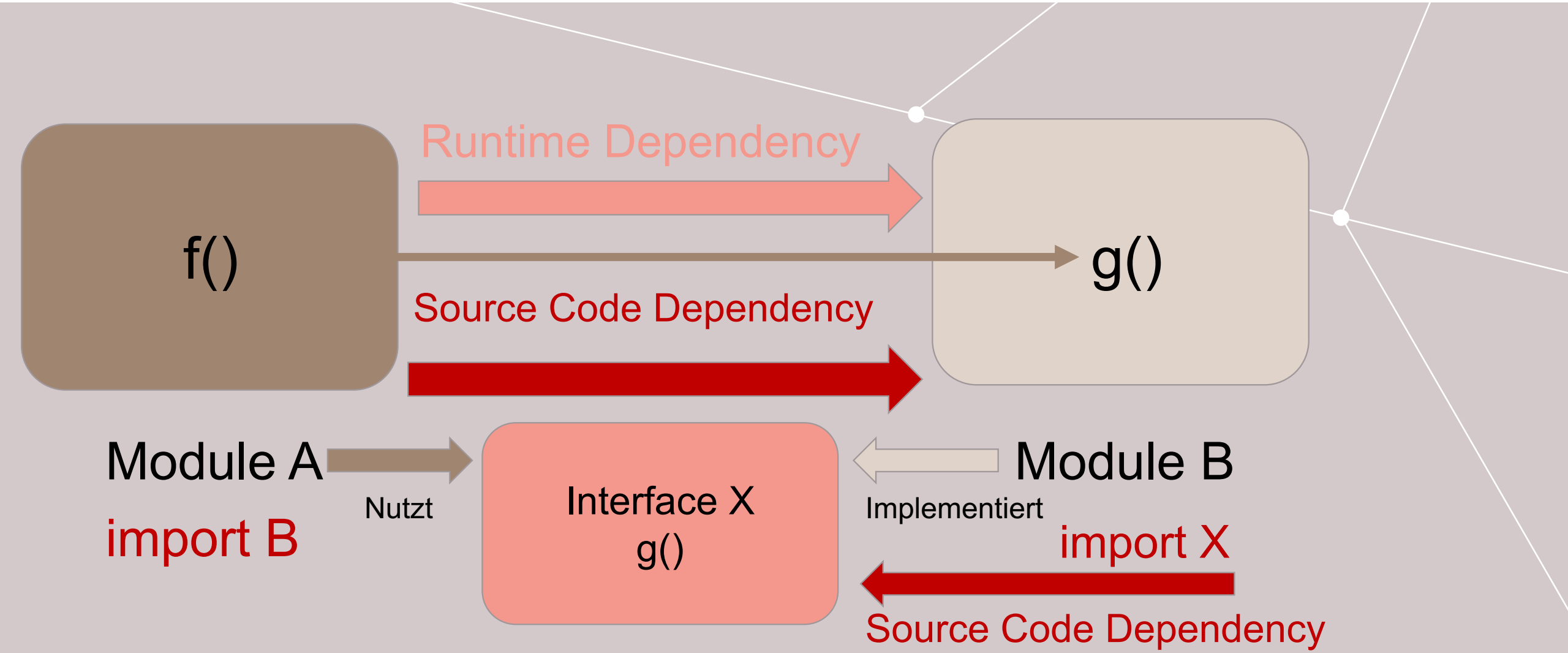


Module A

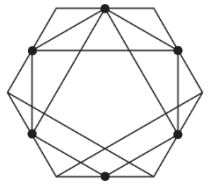
import B

Module B

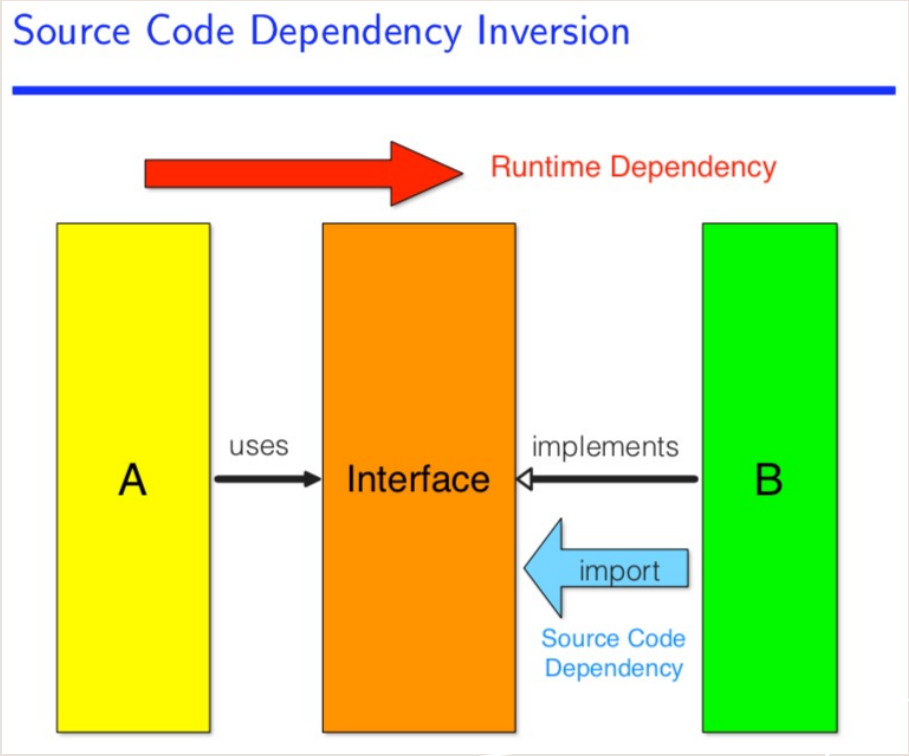
DEPENDENCY?



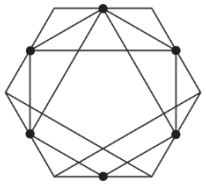
CLEAN ARCHITEKTUR



HEXAGONALE ARCHITECTURE?



DEPENDENCY INVERSION PRINCIPLE



DEPENDENCY INVERSION PRINCIPLE

- The core functionality of a system
- does not depend on its environment
- **Concrete artifacts depend on abstractions** (not *vice versa*)
- **Unstable artifacts depend on stable artifacts** (not *vice versa*)

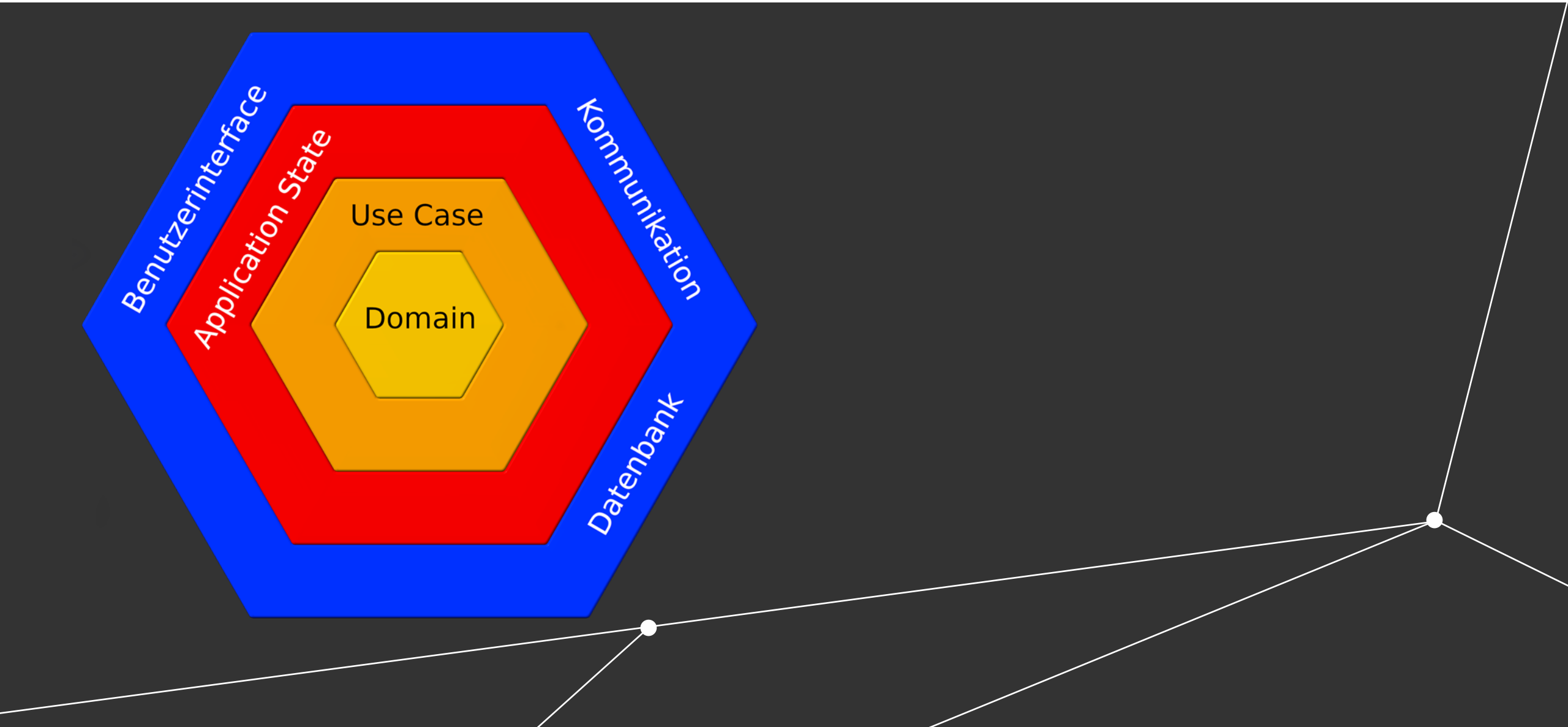
- **Outer layers** of the architecture **depend on inner layers** (not *vice versa*)

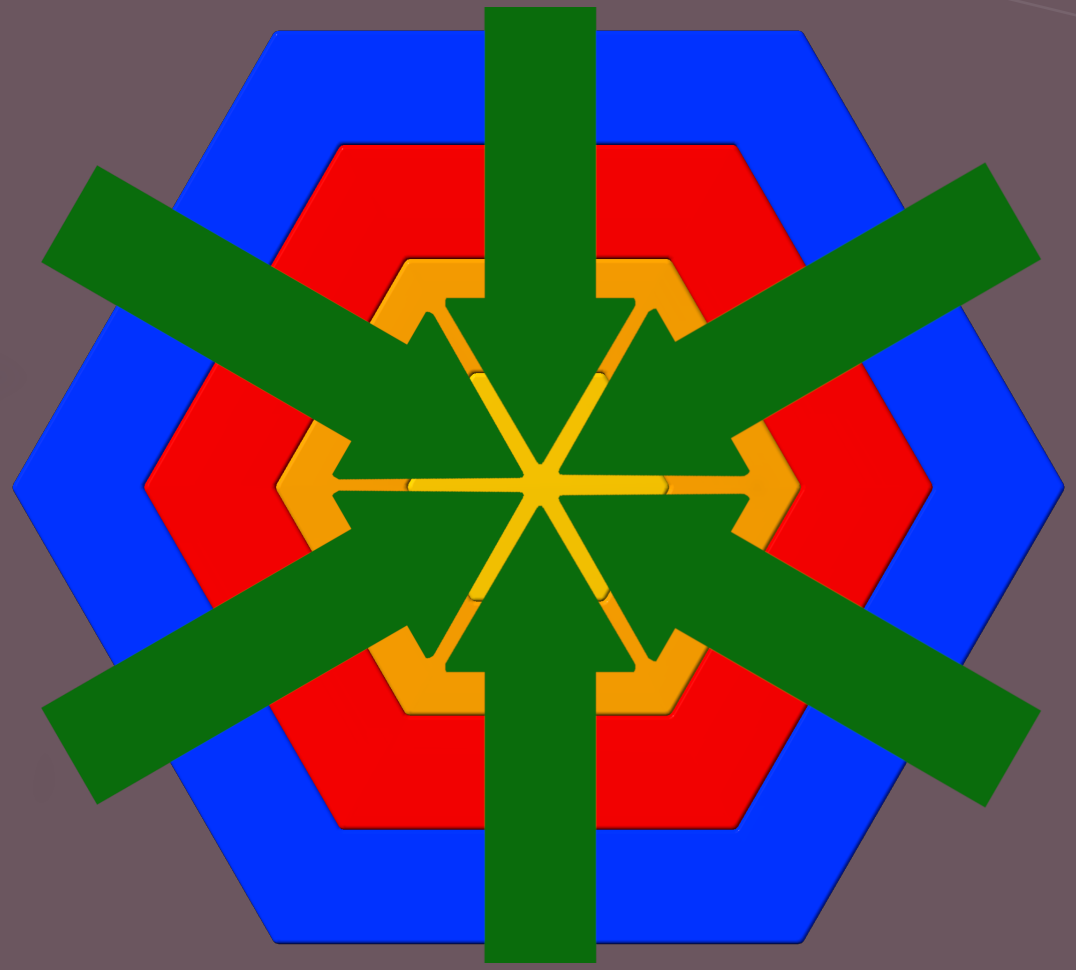
DEPENDENCY INVERSION PRINCIPLE

- Classes/Modules depend on abstractions (e.g., interfaces)
- not on other classes/modules
- Dependency inversion achieves this by introducing interfaces
- that “reverse the dependencies”

- **Outer layers** of the architecture **depend on inner layers** (not *vice versa*)

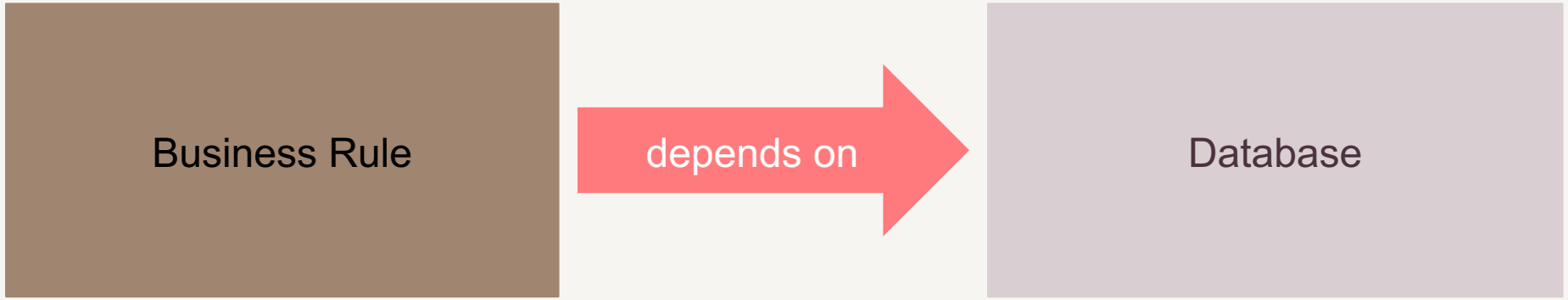
HEXAGONAL ARCHITECTURE?


Benutzerinterface
Application State
Use Case
Domain
Kommunikation
Datenbank

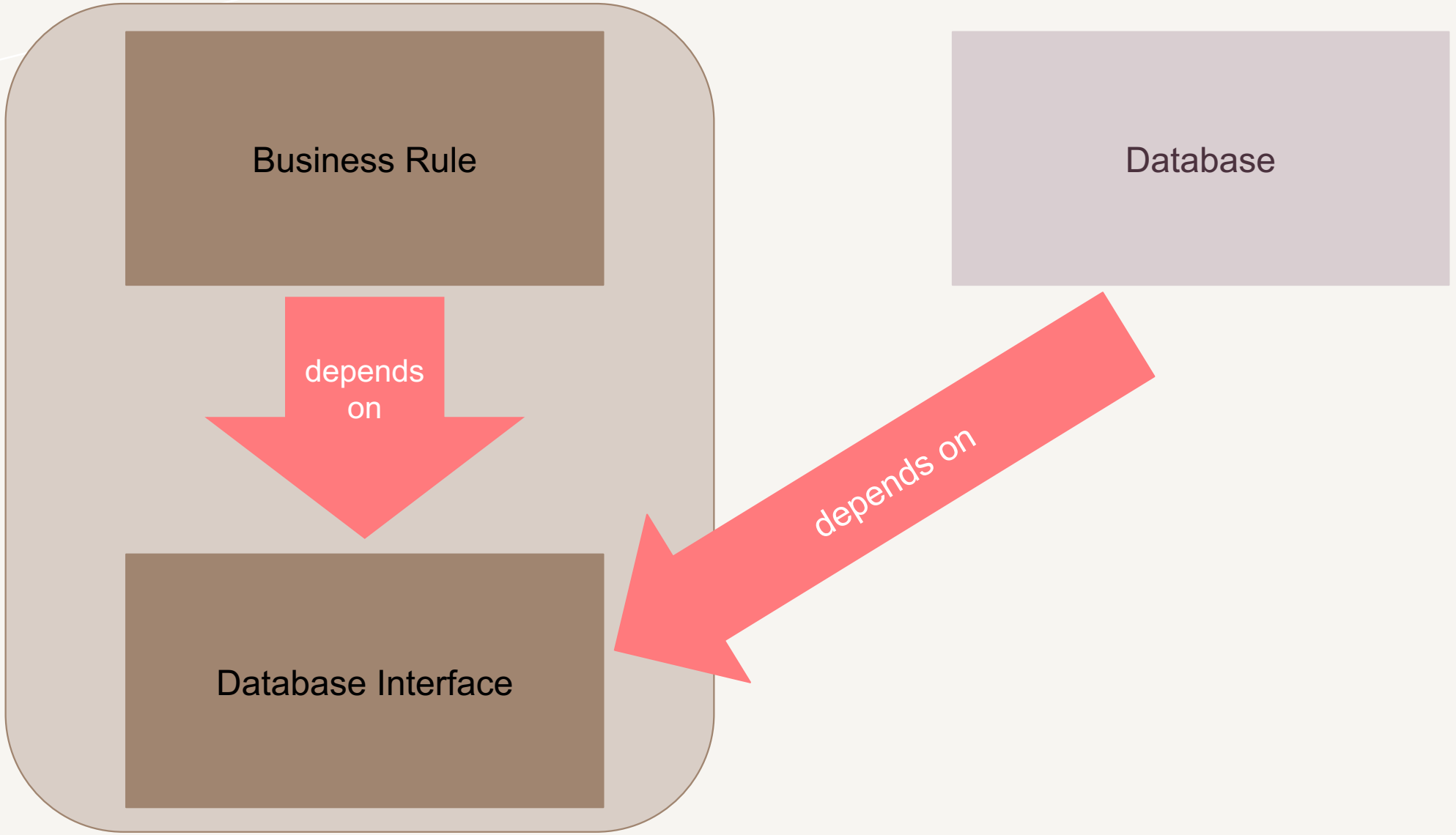


Dependencies

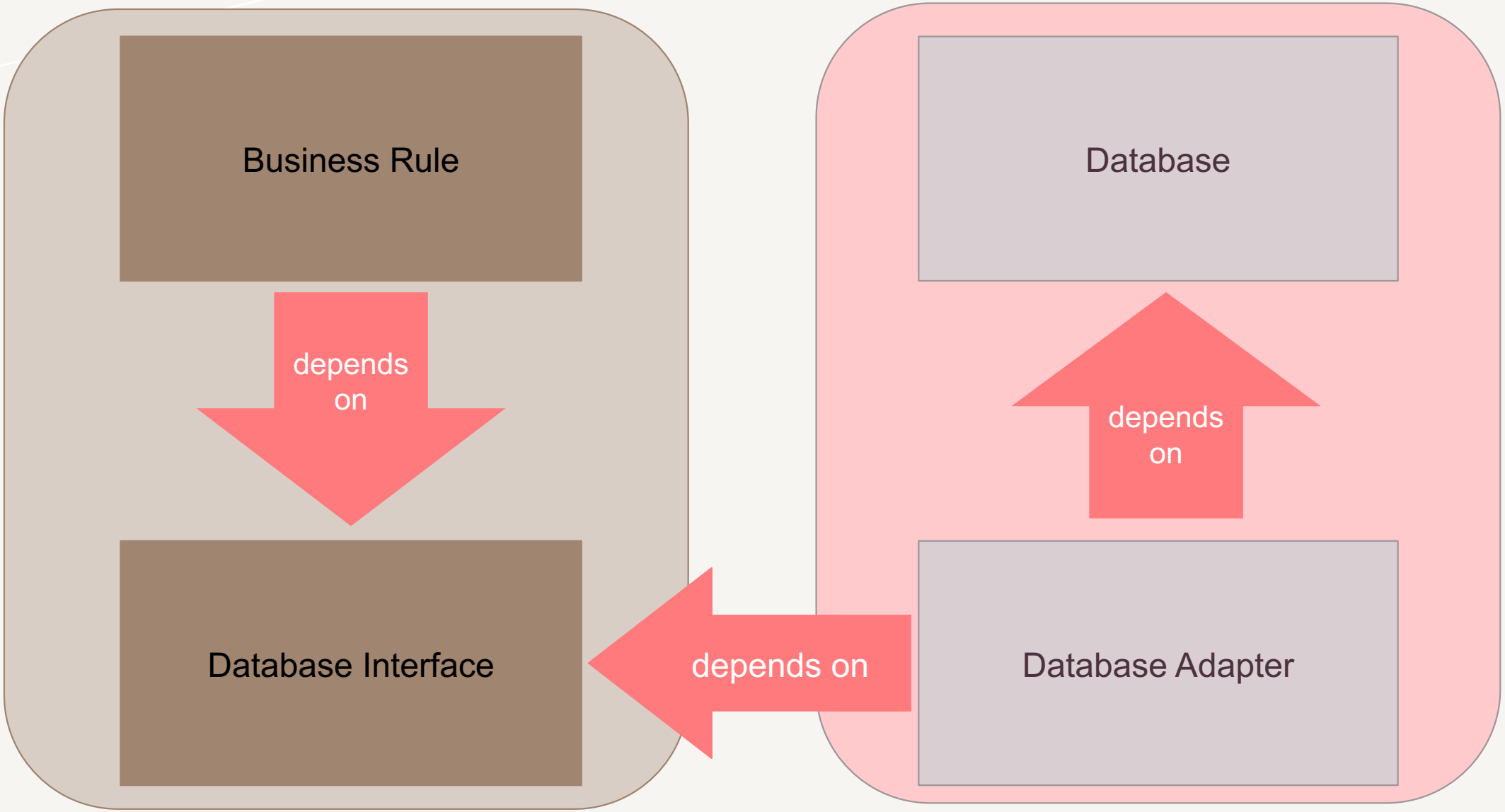
WRONG DEPENDENCY



Dependency Inversion (STEP 1)



Dependency Inversion (STEP 2)



VIELEN DANK FÜR IHRE AUFMERKSAMKEIT.