cphbusiness

COPENHAGEN BUSINESS ACADEMY

Automated test 2

Designing for testability

- A modular design is composed of separate modules, each serving a particular purpose in the design

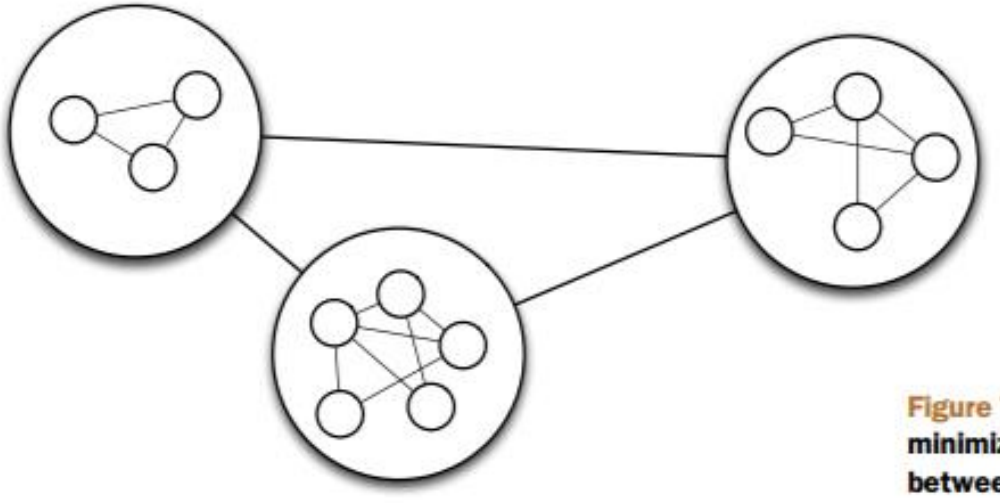


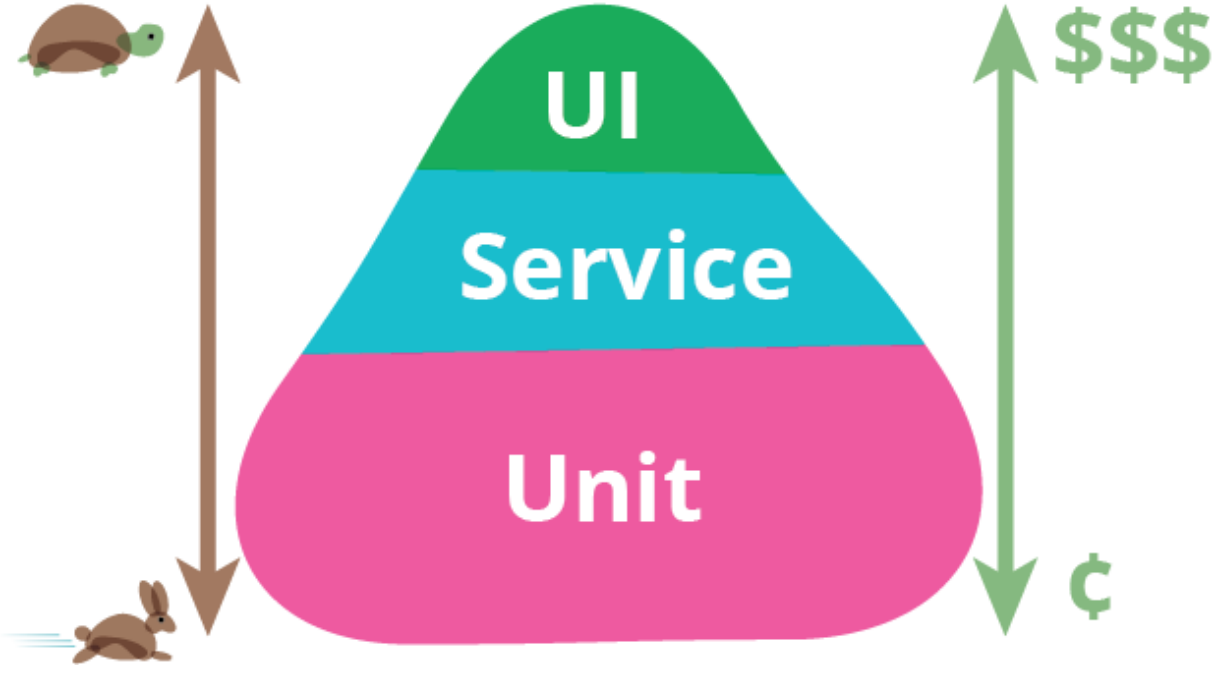
Figure 7.1 **Modular designs minimize the dependencies between modules**

Source: Effective Unit Testing by Lasse Koskela

- SOLID principles and TDD can help you keep your code modular and testable

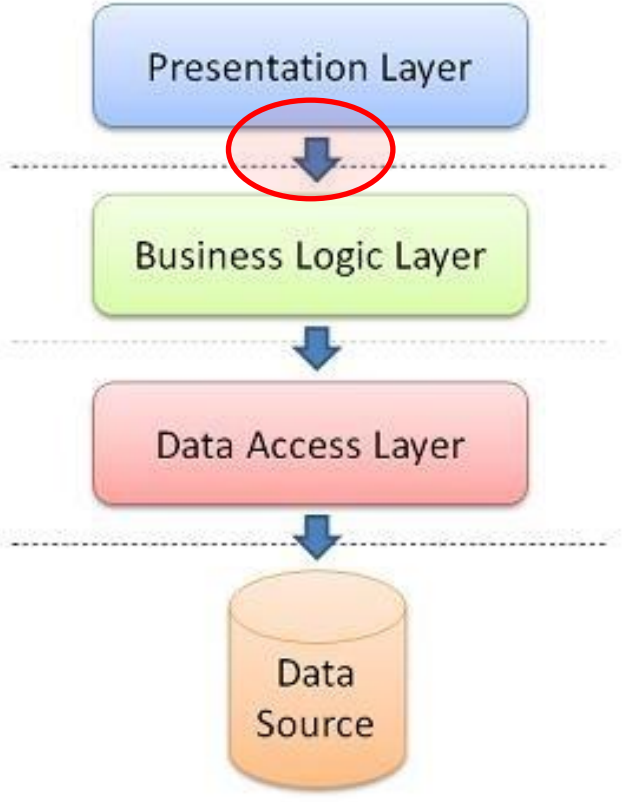
What to test?

- The test pyramid shows that you should have many more low-level unit tests than high level end-to-end tests running through a GUI.



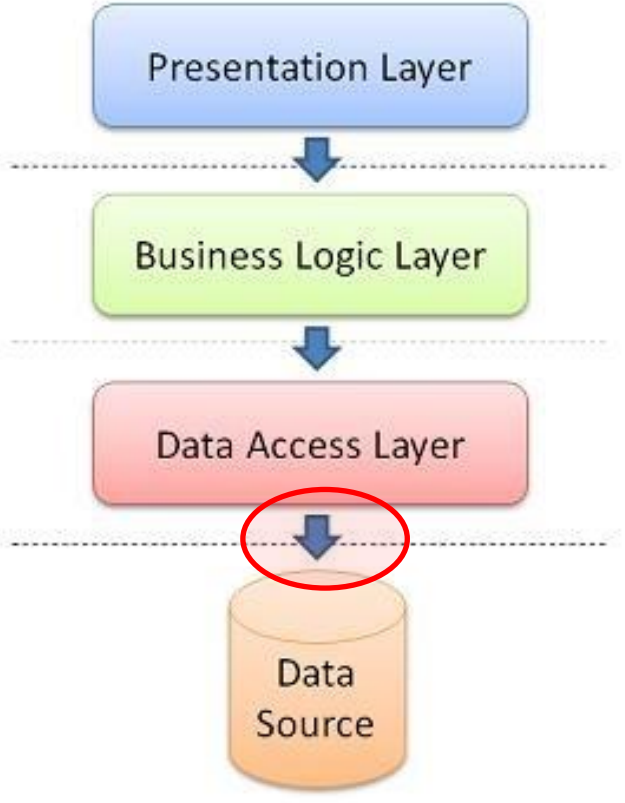
Testing at Service Level

- A layered application often has a frontend to handle the presentation and a backend to execute the business logic.
- Tests can verify that a request passes through the frontend and returns an appropriate response from the backend.



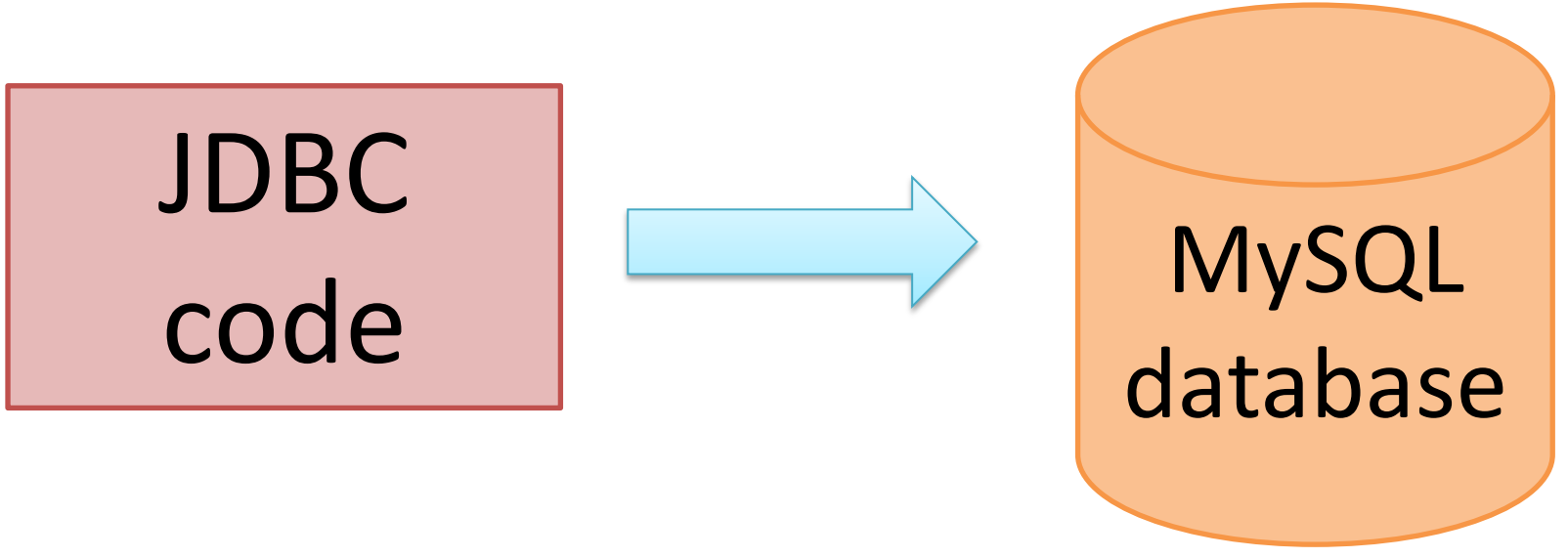
Integration Testing

- We test connection to services such as database, file system or any other external resource or device:



Integration Test Demo

- Test how Data Access Layer communicates with a database – demo!



Considerations when testing the data access layer

- Test database vs. production database
  - demo
- Designing for testability
  - demo
- Clearing and setting up a database

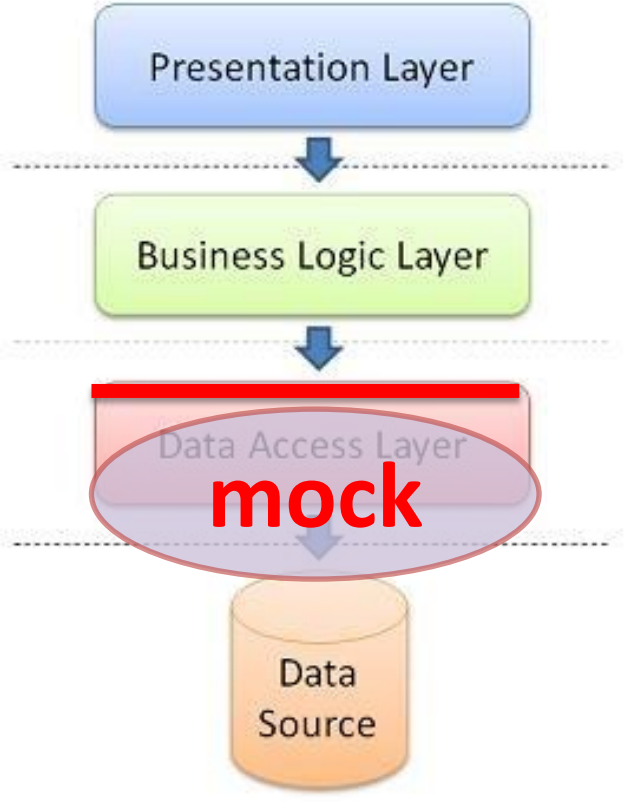
Mocking

- Setting up a fictive environment to test only one part of the system
  - Is much faster than testing the real database
  - Has focus on testing behavior and is useful when doing TDD (you can mock dependencies that you haven't implemented yet)
  - This is hard!!!! -> so if you really do this it leads to a better structure of your program. It becomes more robust and better structured.

- Mockito is a mocking framework

Mocking the Data Access Layer

- Demo: Mocking the data mapper class
- Communication to data access layer happens through a facade



Stubs vs mocks?

Rule of thumb:
Stub queries; mock actions

Stubs

- Pros:
  - Fast and lightweight
  - Easy to write and understand
- Cons:
  - Specialized methods are required to verify state
  - They don’t test behavior of faked objects

Mocks

- Pros:
  - They can track domain logic behavior
- Cons:
  - You need to learn a mocking framework
  - You need to know how the mocked API works

How to manage dependencies

- Use Maven to handle dependencies in Netbeans project

- Pom.xml file:

```
<dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.hamcrest</groupId>
            <artifactId>hamcrest-core</artifactId>
            <version>1.3</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-all</artifactId>
            <version>1.9.5</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.23</version>
        </dependency>
    </dependencies>
```

JDBC hint

- **The try-with-resources Statement**

- https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html