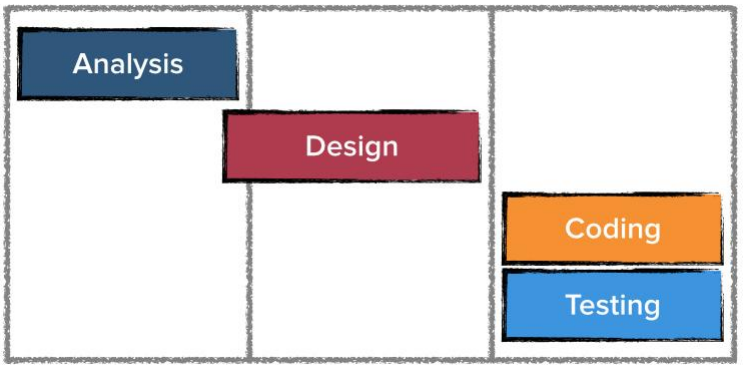
Services, Exceptions and Misc

Slide Series #6

Starting out Iteration 4



Some final pieces!

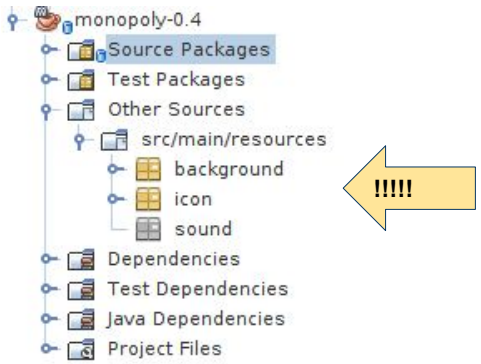
Lombok

```
@EqualsAndHashCode( of = "name")
public class Player {

    @Getter
    private final String name; // Unique name for player
    @Getter
    private int balance;
    private boolean inJail = false;
    @Getter
    private Space position; // The actual position


    ...
}
```

Tiresome and boring to write “boilerplate” code

- We use Lombok for now.
  - Add Maven dependency
- Add “Annotations” to generate setter/getter/ and more ….
- Easy to add to to project thanks to Maven

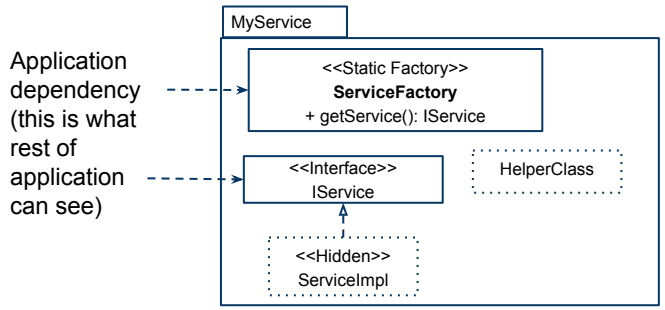
MP : Problems with Resources



MP: Amazingly complicated to get file listing from resources directory

- Try to find something …
- … found Apache Commons IO
- Add Maven dependency and use.

Implementing a Service



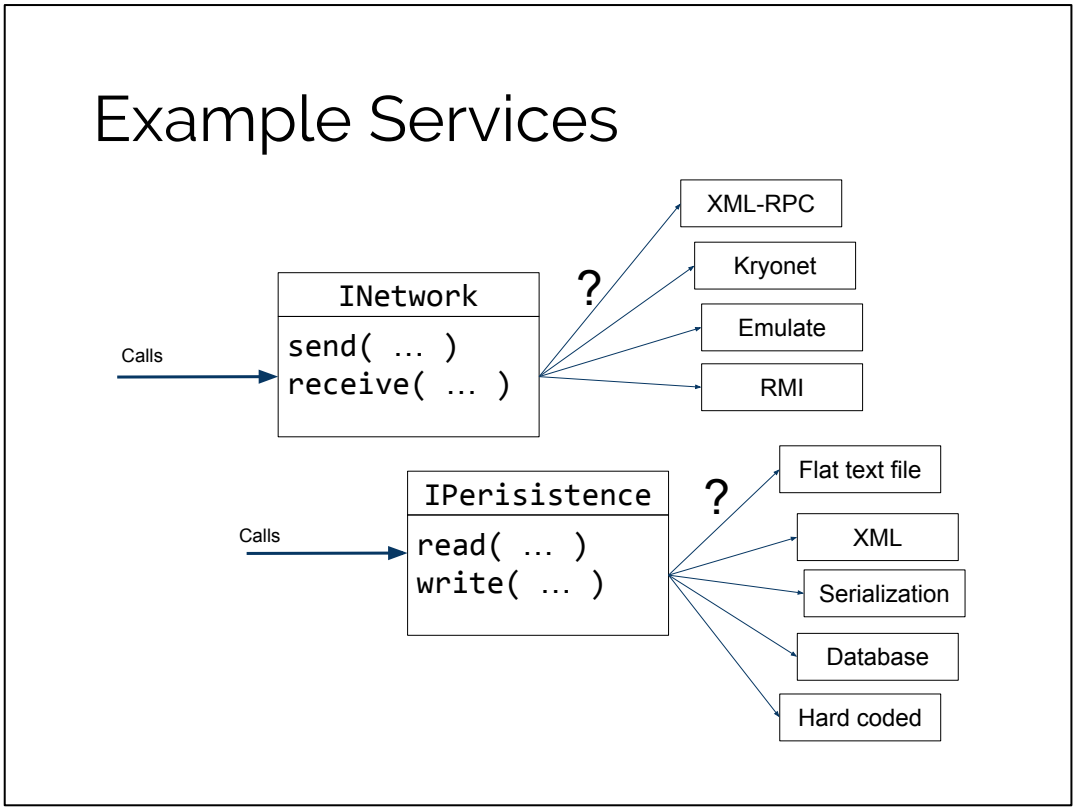

```
IService s = ServiceFactory.getService();
… s.doService( …  );
```

Services implemented using Facade pattern

- I.e. an interface used by control and a Factory to get an implementing object
- All other classes are package private (i.e. no public)
    - Possibly pure data classes implemented as immutable value objects
- For application (control layer) to find a service possibly use the Service locator design pattern
- If problems with dependencies use layering inside service
- Use of generics may remove dependencies

ALSO: Often need to decide on format for data

- Try to shield application from changes in data formats!



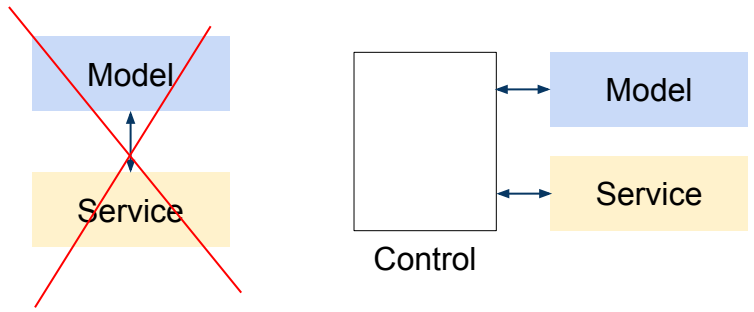
NOTE: This is optional

Any service is accessed via an interface (INetwork or IPersistence)!

- Exact implementation technique never exposed to application
  - Again: Also hide data formats
- Exact implementation technique, is a technical detail, not overly interesting for us
- Interfaces is a crucial part of application design, we are very concerned about this!

NOTE : Relational databases and OO have severe problem.

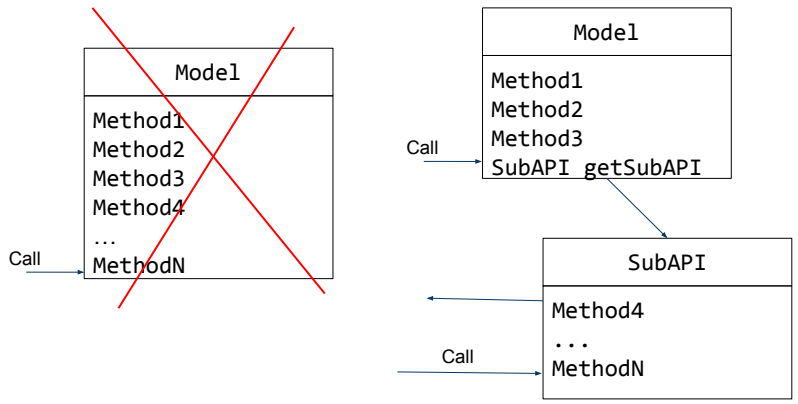
Usage of a Service



We don't want to clutter the model

- No service code in model
- Use a controller
  - Get data from model and shuffle to service or ...
  - Get data from service set in model

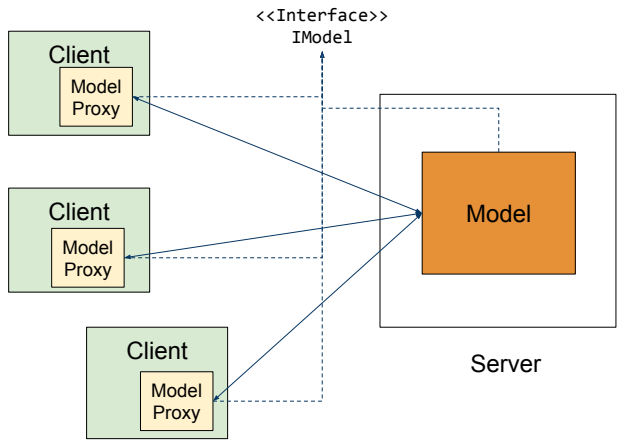
API too Large



I model API becomes too large

- Add methods returning sub APIs

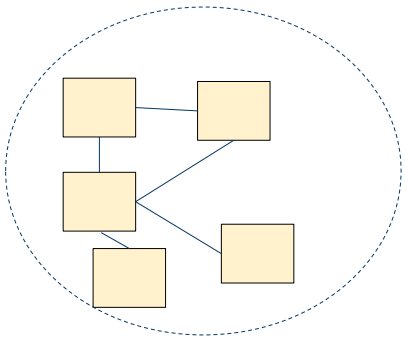
A Note On Client Server



A client server application normally has the model on the server

- Clients act merely as IO channels …
- .. issuing calls on a proxy for the model (Remote proxy pattern)
- The proxy hides the network.

A Note On Databases



OO modell

1	“pelle”	34	
2	“fia”	56	
3			

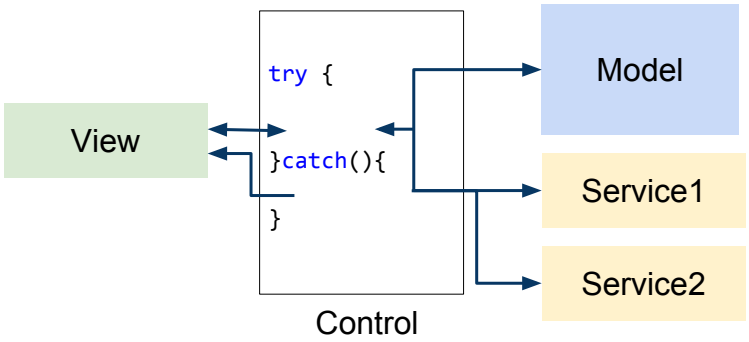
1	“götaplatsen	8000	
2	“avenyn”	7000	
3			

Relational database

OO-models and relational databases hard (unsolved) problem

- OO model is a web of objects
- Database is primitive data in tables
- Object relational impedance mismatch
- Possibly : Use some ORM framework

Exception Handling



Exceptionhandling not well understood subject
- This is an advice

Exceptions may come from Model or Services
- Model or Services called from control …
  - Model never call service directly (except eventbus)
- Handle exceptions in control …
- … propagate message to view to inform user

Possibly create high level exception classes if many layers in application
- Also exception tunneling (Java specific)

MP : Monopoly-0.4

Download from course page, inspect and run!

Summary

- Used some APIs
- Implemented a Service
- Some exception handling

Next: Continue until finished ….