# Hold *bønnene* dine unna **domenemodellen** min!

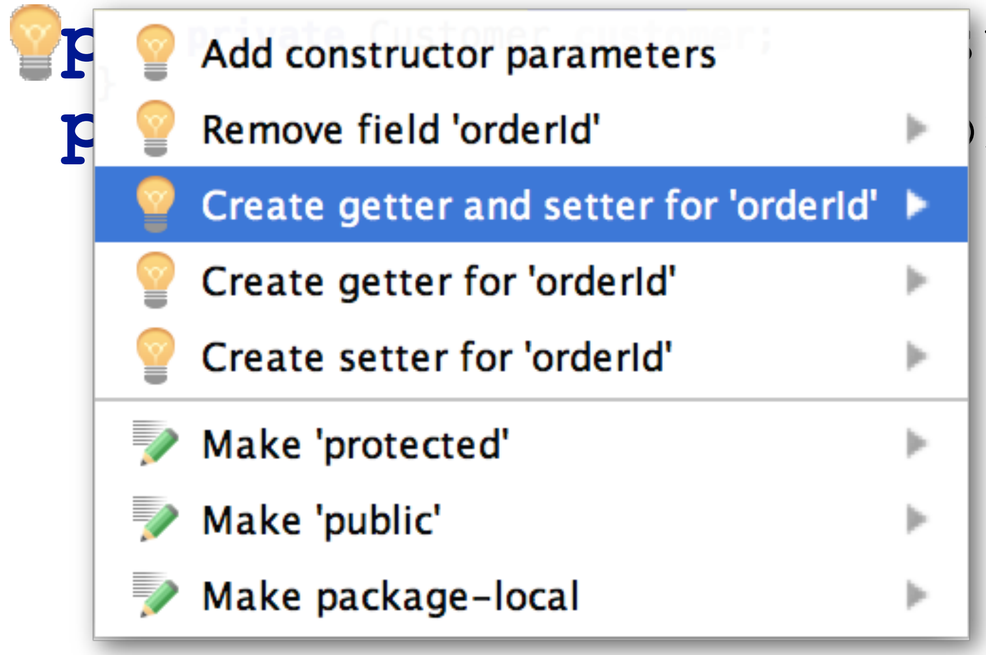
Kenneth Pedersen

Miles

FAGLIG AUTORITET OG VARME

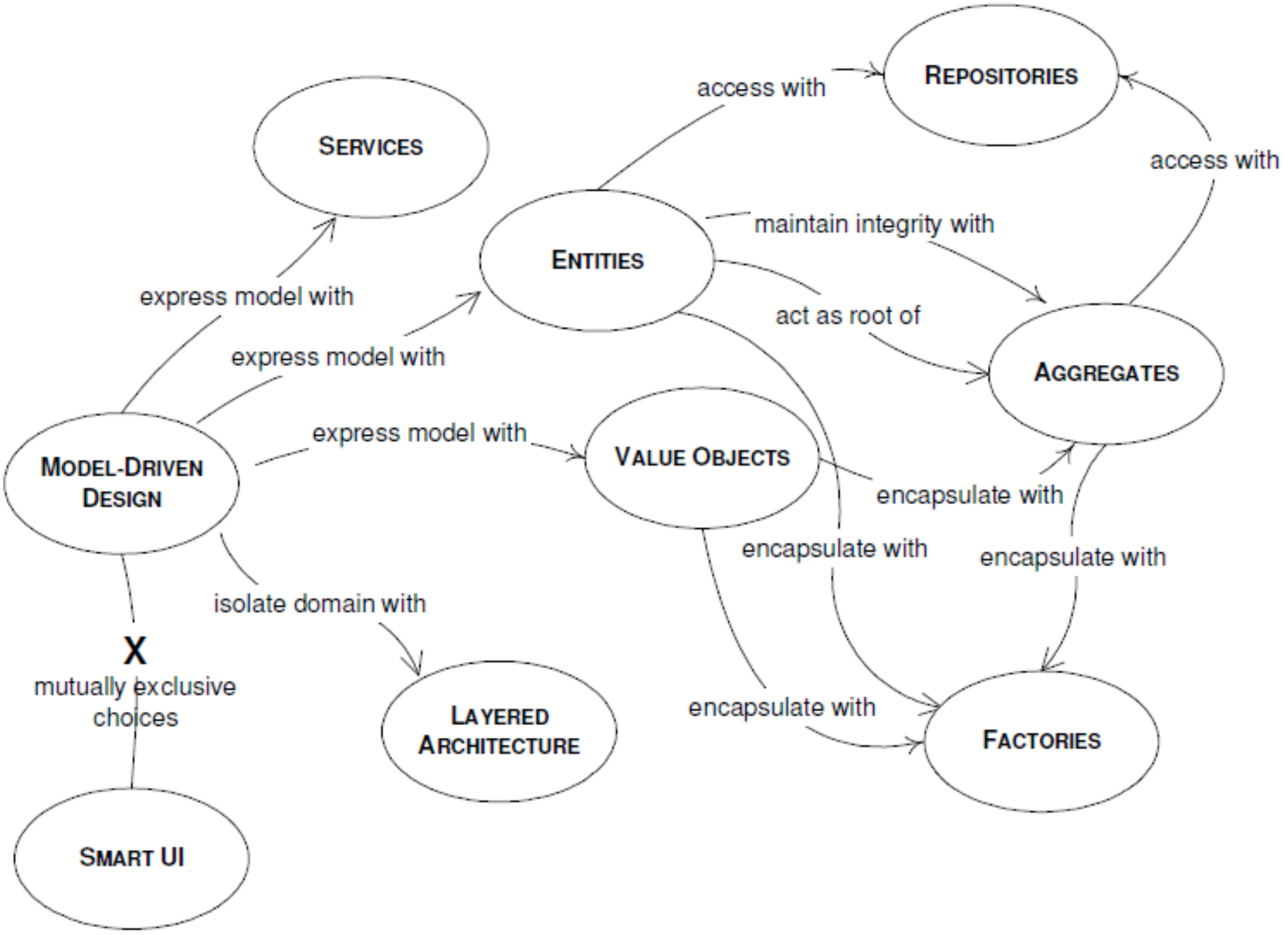
```
public class Order {
    public String orderId;
    public Customer customer;
    public OrderState orderState;
}
```

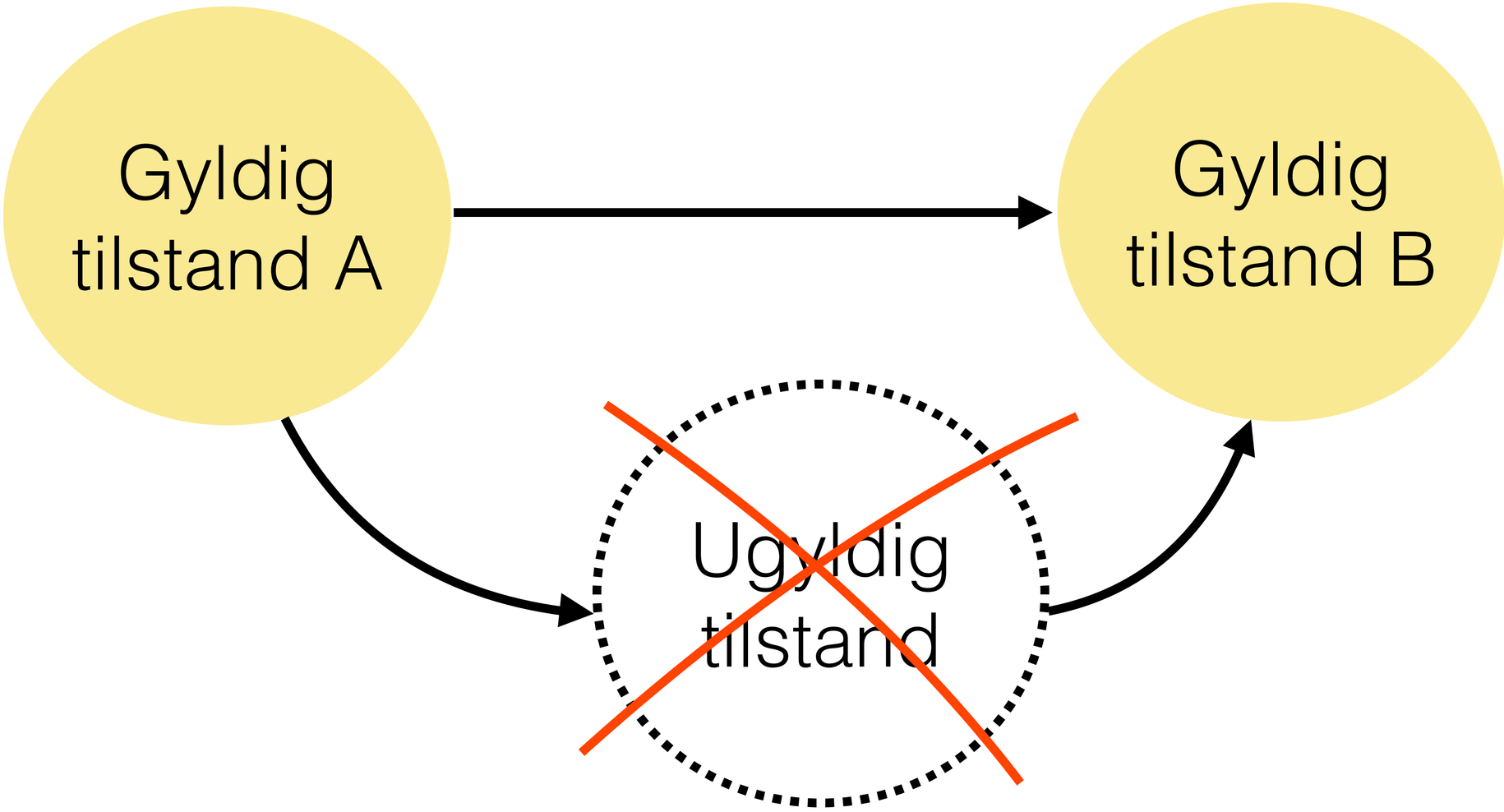
```
public class Order {
    private String orderId;
    p                                    tomer;
    p                                    rderState;
}
```



```java
public class Order {
    private String orderId;
    private Customer customer;
    private OrderState orderState;

    public String getOrderId() {
        return orderId;
    }

    public void setOrderId(String orderId) {
        this.orderId = orderId;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public OrderState getOrderState() {
        return orderState;
    }

    public void setOrderState(OrderState orderState) {
        this.orderState = orderState;
    }
}
```

```java
@Entity(name = "ORDER")
public class Order {
    @Id
    @Column(name = "ORDER_ID", nullable = false)
    private String orderId;

    @ManyToOne(optional = false)
    @JoinColumn(name = "CUSTOMER_ID", referencedColumnName = "CUSTOMER_ID")
    private Customer customer;

    @Column(name = "ORDER_STATE", nullable = false)
    private OrderState orderState;

    public String getOrderId() {
        return orderId;
    }

    public void setOrderId(String orderId) {
        this.orderId = orderId;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public OrderState getOrderState() {
        return orderState;
    }

    public void setOrderState(OrderState orderState) {
        this.orderState = orderState;
    }
}
```

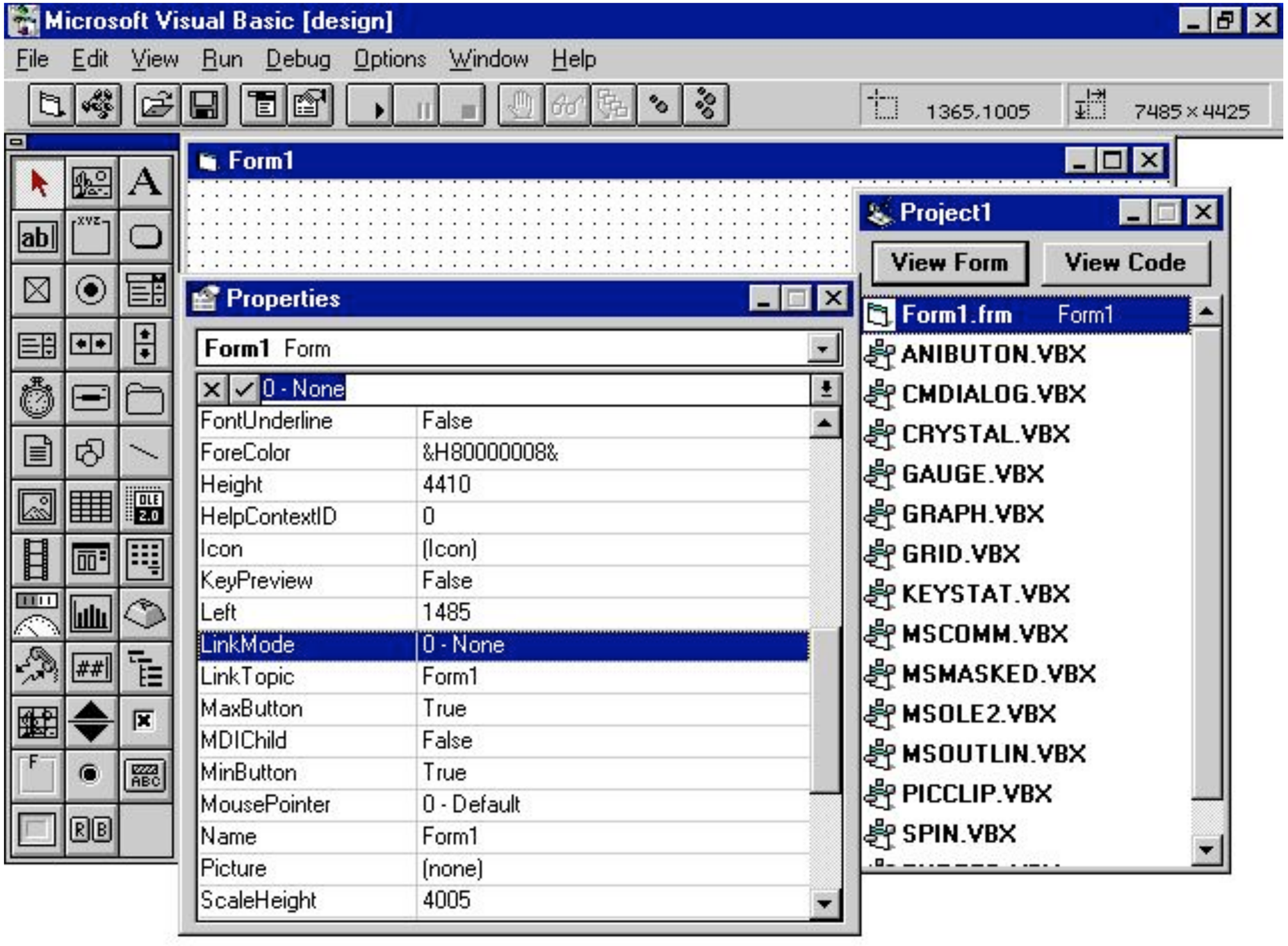

SERVICES
REPOSITORIES
access with
access with
express model with
ENTITIES
maintain integrity with
express model with
act as root of
AGGREGATES
MODEL-DRIVEN DESIGN
express model with
VALUE OBJECTS
encapsulate with
encapsulate with
encapsulate with
isolate domain with
X
mutually exclusive choices
LAYERED ARCHITECTURE
encapsulate with
FACTORIES
SMART UI


Gyldig
tilstand A
Gyldig
tilstand B
Ugyldig
tilstand

```
public Order() { }
```

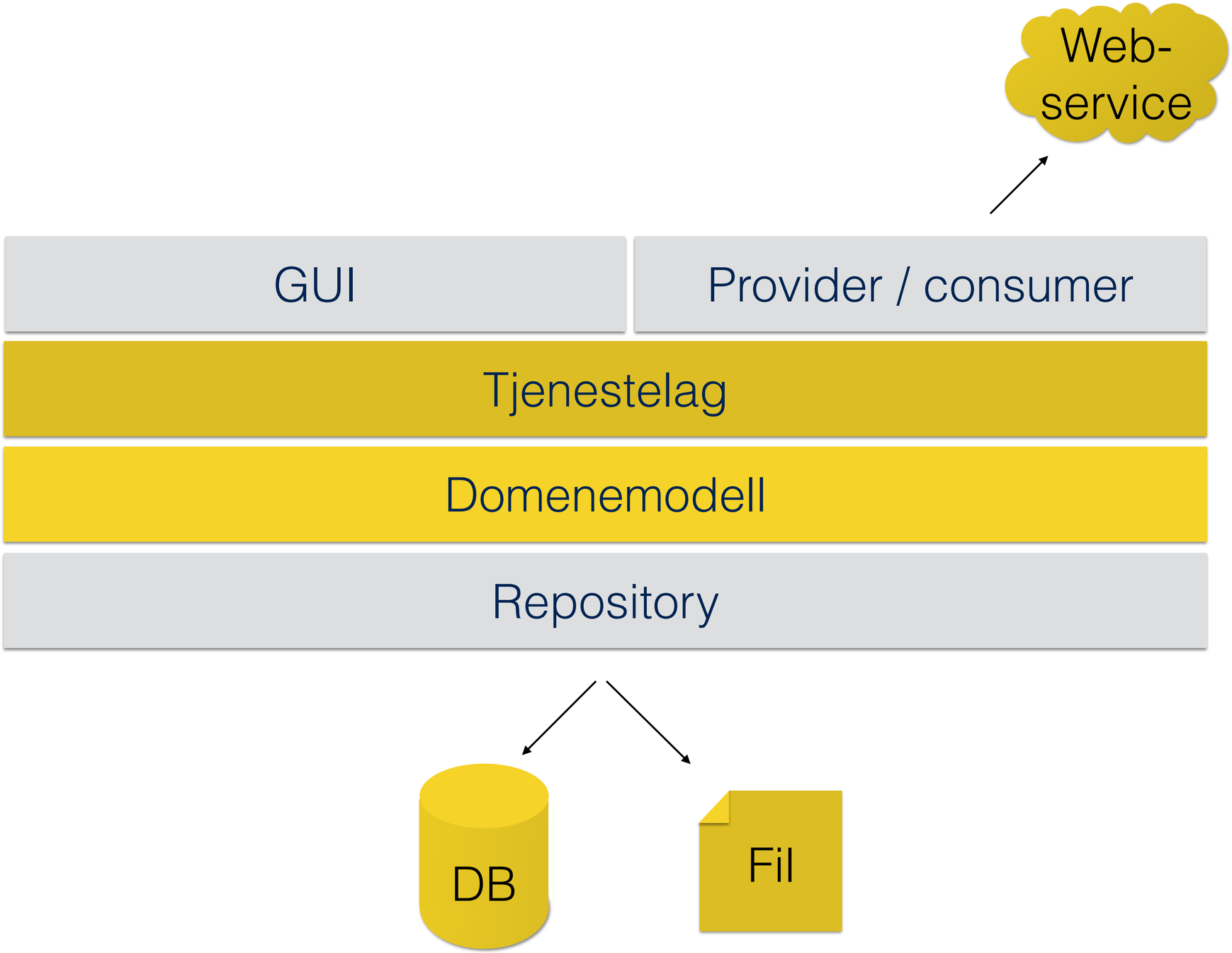
```
public OrderState getOrderState();
public void setOrderState(
                    OrderState orderState);
```

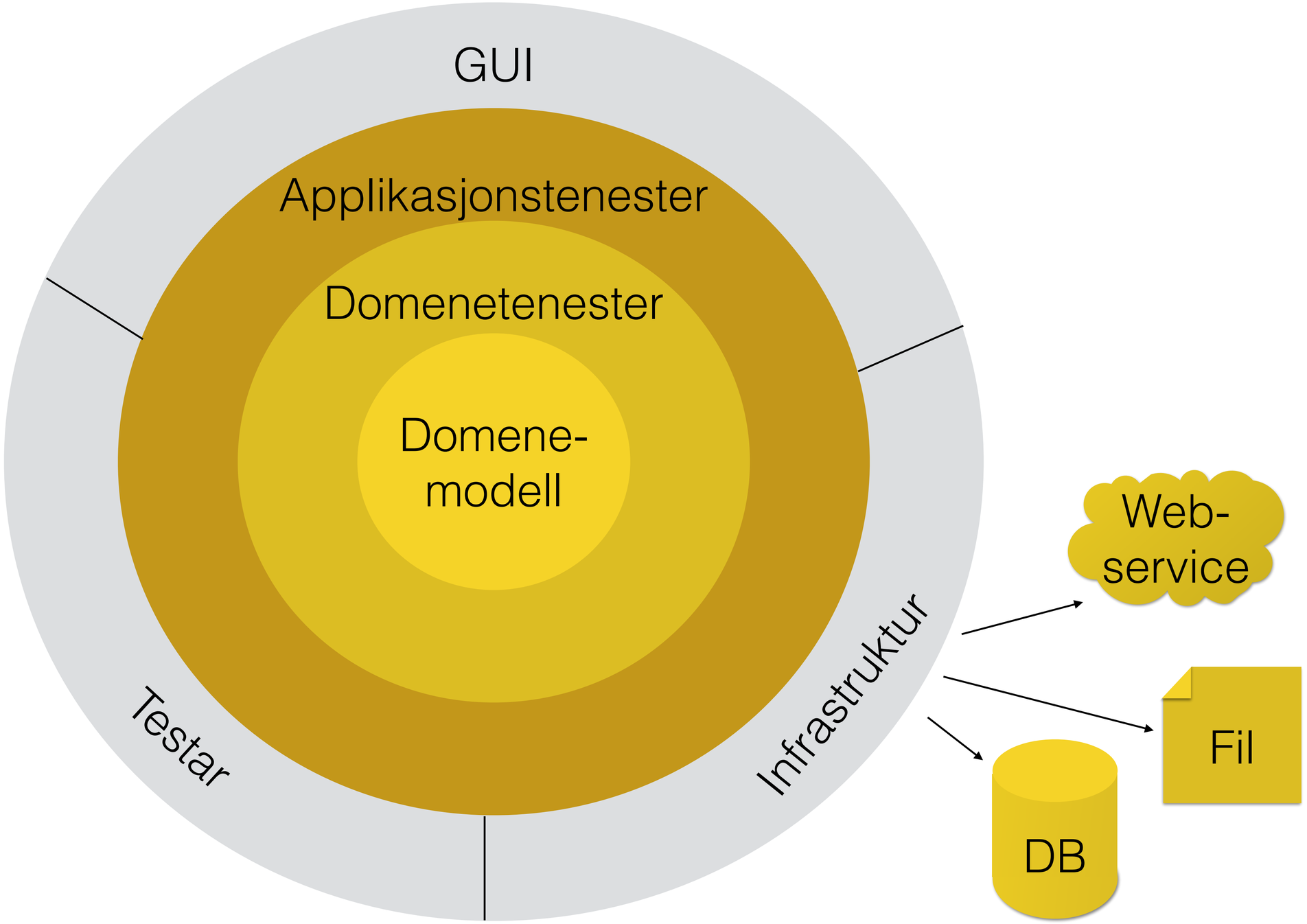
*“A Java Bean is a reusable software component that can be manipulated visually in a builder tool.”*


Microsoft Visual Basic [design]
File Edit View Run Debug Options Window Help
1365,1005
7485 x 4425
Form1
Project1
View Form
View Code
Form1.frm Form1
ANIBUTON.VBX
CMDIALOG.VBX
CRYSTAL.VBX
GAUGE.VBX
GRAPH.VBX
GRID.VBX
KEYSTAT.VBX
MSCOMM.VBX
MSMASKED.VBX
MSOLE2.VBX
MSOUTLIN.VBX
PICCLIP.VBX
SPIN.VBX
Properties
Form1 Form
0 - None
FontUnderline False
ForeColor &H80000008&
Height 4410
HelpContextID 0
Icon (Icon)
KeyPreview False
Left 1485
LinkMode 0 - None
LinkTopic Form1
MaxButton True
MDIChild False
MinButton True
MousePointer 0 - Default
Name Form1
Picture (none)
ScaleHeight 4005

```
public class Order {
    public final String orderId;
    public final Customer customer;
    private OrderState orderState;

    public Order(String orderId,
                 Customer customer) {
        this.orderId = orderId;
        this.customer = customer;
    }

    public OrderState orderState() {
        return orderState;
    }

    public void complete() {
        // Perform validation logic and
        // update status of dependent
        // object, then set order state
        orderState = OrderState.COMPLETE;
    }
}
```

# Kva no?


Web-
service
GUI
Provider / consumer
Tjenestelag
Domenemodell
Repository
DB
Fil


GUI
Applikasjonstenester
Domenetenester
Domene-
modell
Testar
Infrastruktur
Web-
service
Fil
DB

# Oppsummering

- Bruk av JavaBeans-mønstereret i domenemodellen gjer at domeneklassene mister evnen til å sjølv kunne kontrollere at dei er konsistente til ei kvar tid

- Hibernate, Dozer og mange andre fine ting som vi ønsker å bruke forutsetter bønner

- For å kunne fortsette å bruke desse biblioteka har ein to strategiar:

    1. Implementere dei tinga som biblioteka krever i domenemodellen (0-arguments-konstruktør, setters/getters), som ikkje-public medlemmer i domeneklassene

    2. Bruke arkitekturmønstre som gjer at ein kan holde slike avhengigheiter ute av domenemodellen

*«Not all useful software modules should necessarily turn into beans. Beans are appropriate for software components that can be visually manipulated and customized to achieve some effect. Class libraries are an appropriate way of providing functionality that is useful to programmers, but which doesn't benefit from visual manipulation.»*