Agenda

Inheritance

Strategien

Single Table

Joined

Table per Concrete Class

Best practices

Mappingstrategien

- **Entity erbt von Entity**
  - Drei Strategien
    - Single Table (Table per Class Hierarchy)
    - Joined (Table per Subclass)
    - Table per Concrete Class
- **Entity erbt von Non-Entity**
  - MappedSuperClass

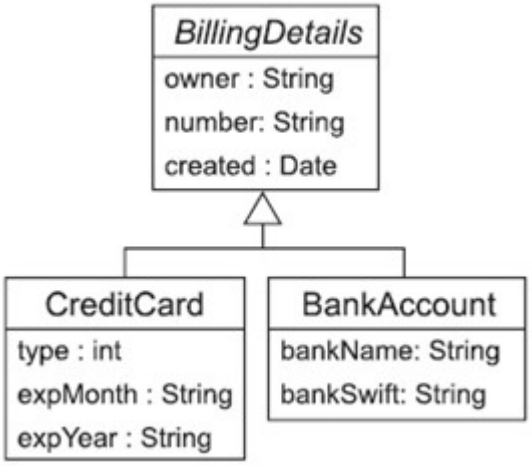
MappedSuperclass

- Oberklasse hat keine eigene Tabelle, aber Subklassen erben Attribute

```
@MappedSuperclass
public abstract class AbstractEntity {
   @Id protected Long id;
   @Version protected Integer version;

   @Temporal(TemporalType.TIME)  protected Date
   changeDate;

   …
}
@Entity
public class Person extends AbstractEntity {

   …
```

Beispiel Inheritance Strategie


```
public abstract class BillingDetail {
   private Long id;
   private String owner;
   private String number;
   …
}
public class CreditCard extends BillingDetail
{
   private int type;
   …
}
…
```



Agenda

Inheritance

Strategien

Single Table

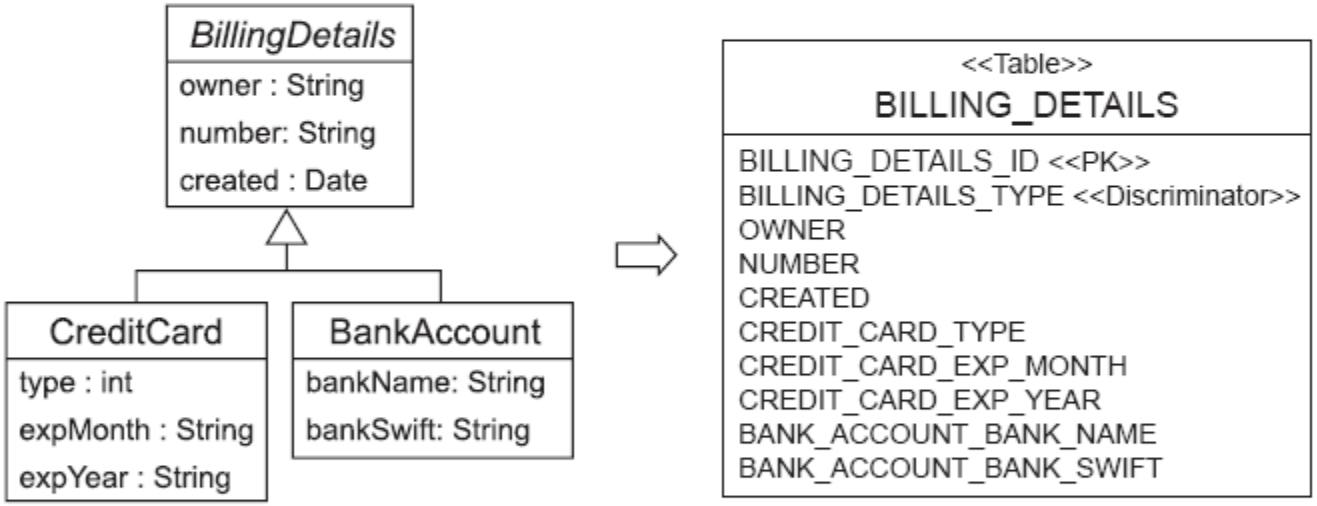
Joined

Table per Concrete Class

Best practices

Single Table

- Eine Tabelle für alle Klassen
- Keine joins
- Mehrere NULL Spalten
- Achtung: Spalten von Unterklassen können keine NOT NULL constraints haben



Single Table


```
@Entity
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn ( name="BILLING_DETAILS_TYPE" ,
discriminatorType=DiscriminatorType.STRING)
public abstract class BillingDetail {
   private Long id;
   …
}

@Entity
@DiscriminatorValue("CC")
public class CreditCard extends BillingDetail {
   private int type;
…
}
```

Table per class hierarchy (Hibernate)


```
<class name="BillingDetail“
   table="BILLING_DETAILS">
   <id name=Mid" column=". . . "><generator
     .../></id>
   <discriminator column="BILLING_DETAILS_TYPE"/>
   <property name="name" column="OWNER"/>
   …
   <subclass name="CreditCard" discriminator-value="CC">
       <property name="type"
   column="CREDIT_CARD_TYPE"/>
       …
   </subclass>
</class>
```

Aufgabe

Aufgabe 8:
Vererbung-Single-Table
Vererbung auf DB abbilden

Agenda

Inheritance

Strategien

Single Table

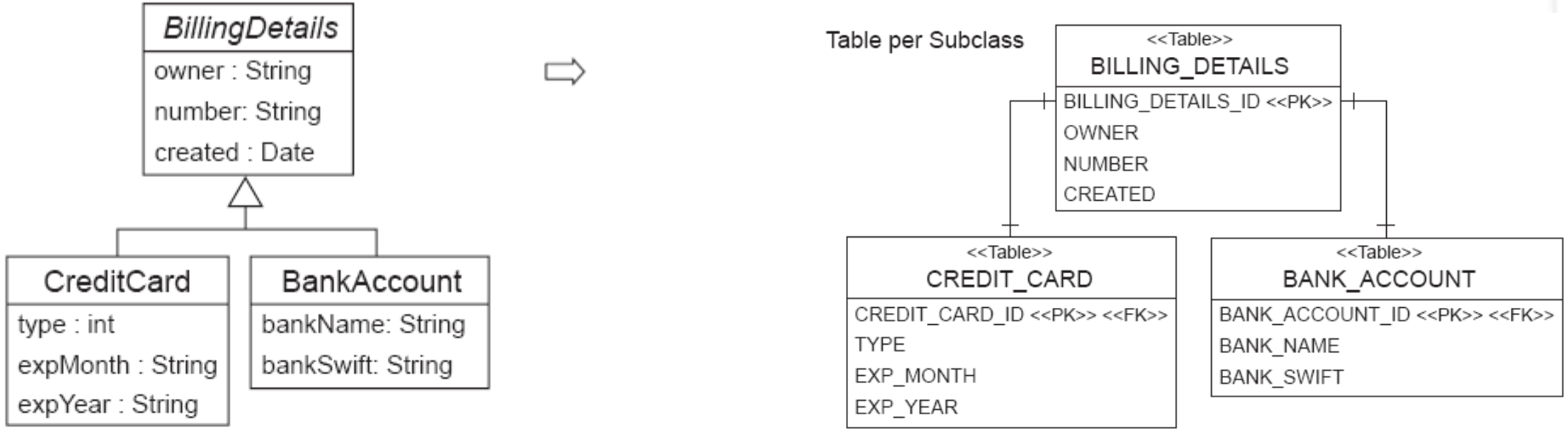
Joined

Table per Concrete Class

Best practices

Joined Table (Table per Subclass)

- Pro Unterklasse eigene Tabelle
  - Tabelle enthält nur Unterklassenattribute
- PK der Unterklasse ist gleich PK der Oberklasse
  - PK ist in Mutterklasse definiert
- Wenige NULL Werten
- polymorphe Abfragen sind möglich (nicht performant)



Joined Table


```
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
public abstract class BillingDetail {
private Long id;
…
}
@Entity
@PrimaryKeyJoinColumn (name = "CC_ID" )
public class CreditCard extends BillingDetail {
…
}
@Entity
@PrimaryKeyJoinColumn(name = "BA_ID")
public class BankAccount extends BillingDetail {
…
}
```

Table per Subclass (Hibernate


```
<class name="BillingDetail" table="BILLING_DETAILS">
   <id name="id" column="...">
       <generator  ... />
   </id>
   <property name="owner" column="OWNER" type="string"/>
   …
   <joined-subclass name="CreditCard" table="CREDIT_CARD">
       <key column="CREDIT_CARD_ID"/>
       <property name="type" column="TYPE"/>
       …
   </joined-subclass>
   …
</class>
```

Table per Subclass mit Diskrimi… (Hibernate)


```
<class name="BillingDetail" table="BILLING_DETAILS">
  <id name="id" column=" . . . "><generator .../>
  </id>
  <discriminator column="BILLING_DETAILS_TYPE"/>
  <property name="name" column="OWNER"/>
  …
  <subclass name="CreditCard" >
        <join table="CREDIT_CARD" fetch="select">
              <key column="CREDIT_CARD_ID"/>
              <property name="type" column="TYPE"/>
              …
        </join>
  </subclass>
</class>
```

Aufgabe

- Aufgabe 9 (optional):
- Vererbung-Joined

Agenda

Inheritance

Strategien

Single Table

Joined

Table per Concrete Class

Best practices

Table per concrete class

- Pro Klasse eigene Tabelle
  - Tabelle besitzt eigene und geerbte Attribute
  - keine polymorphen Abfragen möglich
  - SQL UNION als Behelf

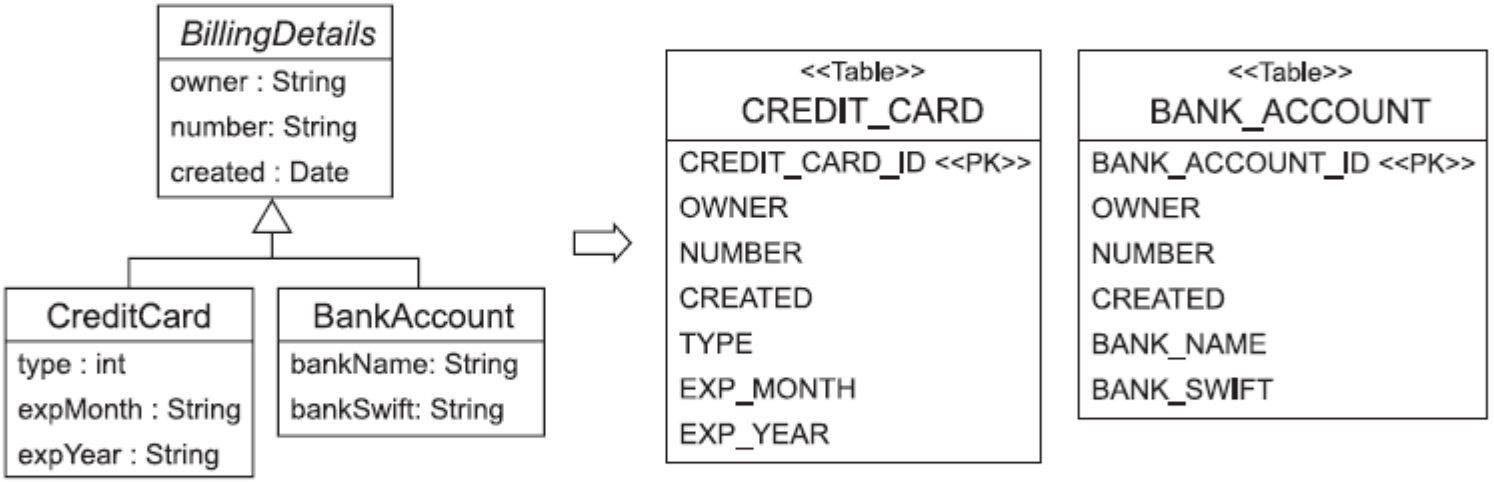


Table per Concrete Class


```
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
   public abstract class BillingDetail {
   …
}
@Entity
public class CreditCard extends BillingDetail {
   …
)
@Entity
public class BankAccount extends BillingDetail {
   …
}
```

Konfiguration Union Subclass (Hibernate)


```
<class name="BillingDetail" >
   <id name="id" column="...">
       <generator ... />
   </id>
   <property name="owner" column="OWNER" type="string"/>
       …
   <union-subclass name="CreditCard" table="CREDIT_CARD">
       <property name="type" column="TYPE"/>
       …
   </union-subclass>
</class>
```

Aufgabe

- Aufgabe 10 (optional):
- Vererbung-Table-Concrete-Class

Agenda

Inheritance

Strategien

Single Table

Joined

Table per Concrete Class

Best practices

Welche Vererbungsstrategie?

- Vorteile SINGLE_TABLE:
  - Volle Unterstützung objektorientierter Polymorphie
  - Sehr performante Abfragen möglich, da keine Joins Einfaches konzeptionelles Modell
  - Einfache Syntax. Im Minimalfall kein zusätzliches JPA-Mapping benötigt
- Nachteile SINGLE.TABLE:
  - NOT-NULL-Constraints nicht möglich
  - Minimaler Speicherbedarf für die NULL-Werte
  - Datenbank-Schema nicht in Normalform, da funktionale Abhängigkeiten

Welche Vererbungsstrategie?

- Vorteile JOINED:
  - Volle Unterstützung objektorientierter Polymorphie
  - 1:1-Entsprechung zwischen Klasse und Tabelle
  - Tabellen in Normalform
- Nachteile JOINED:
  - Abfragen verwenden Joins und sind daher aufwendig. Der Join findet jedoch über einen singulären Fremdschlüssel statt, so dass der Aufwand begrenzt ist.

Welche Vererbungsstrategie?

- Vorteile TABLE.PER.CLASS:
  - Sehr performante Abfragen möglich, da keine Joins
  - Einfache Verwendung in anderen Programmiersprachen
  - Keine Änderung an bestehenden Tabellen beim Einfügen von Blättern in der Vererbungshierarchie
- Nachteile TABLE.PER.CLASS:
  - Aufwendige Abfrage durch mehrere Select- und/oder Union-Anweisungen
  - Schwache objektorientierte Polymorphie
  - Schemaredundanz in Unterklassen, daher nicht in Normalform
  - Keine expliziten Fremdschlüssel auf Datenbankebene
  - Verwendung von Beziehungen eventuell eingeschränkt