Persistence - Part 1

Philippe Collet, contains 78,3% of slides from Sébastien Mosser

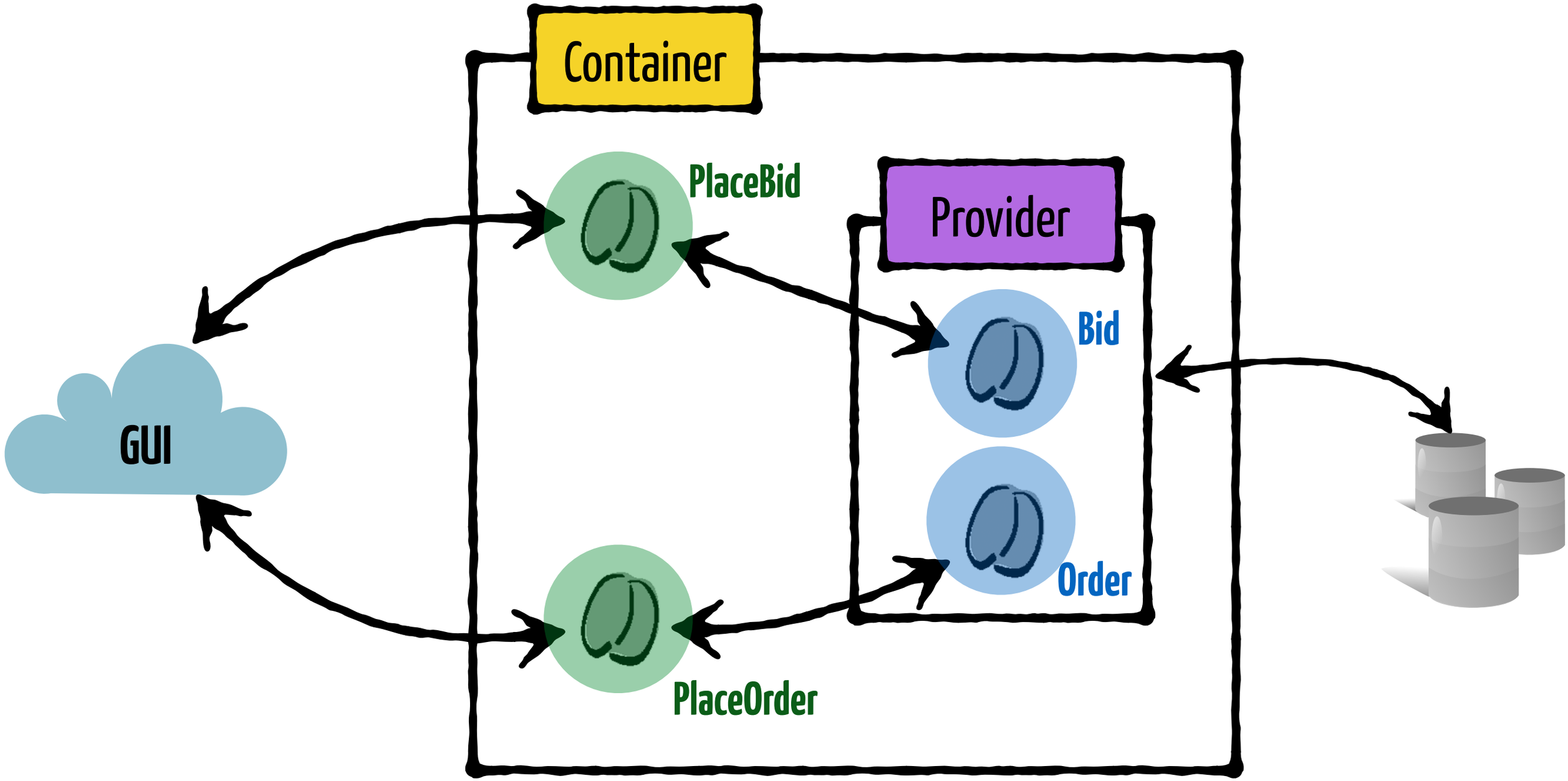
Lecture #5 March 2020

pictures: sxc.hu, pixabay

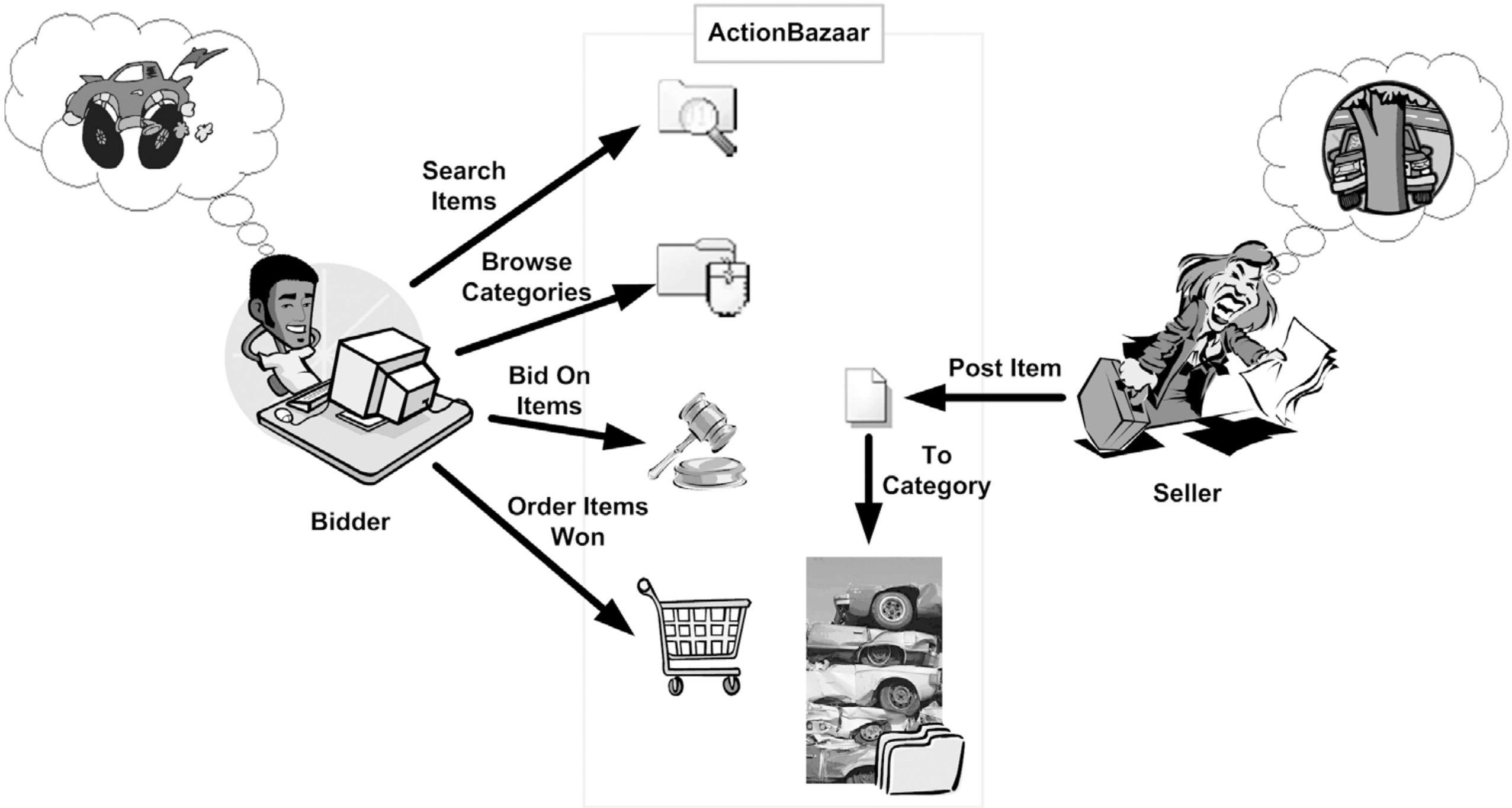
Object-Relational Mapping

Principles & Patterns

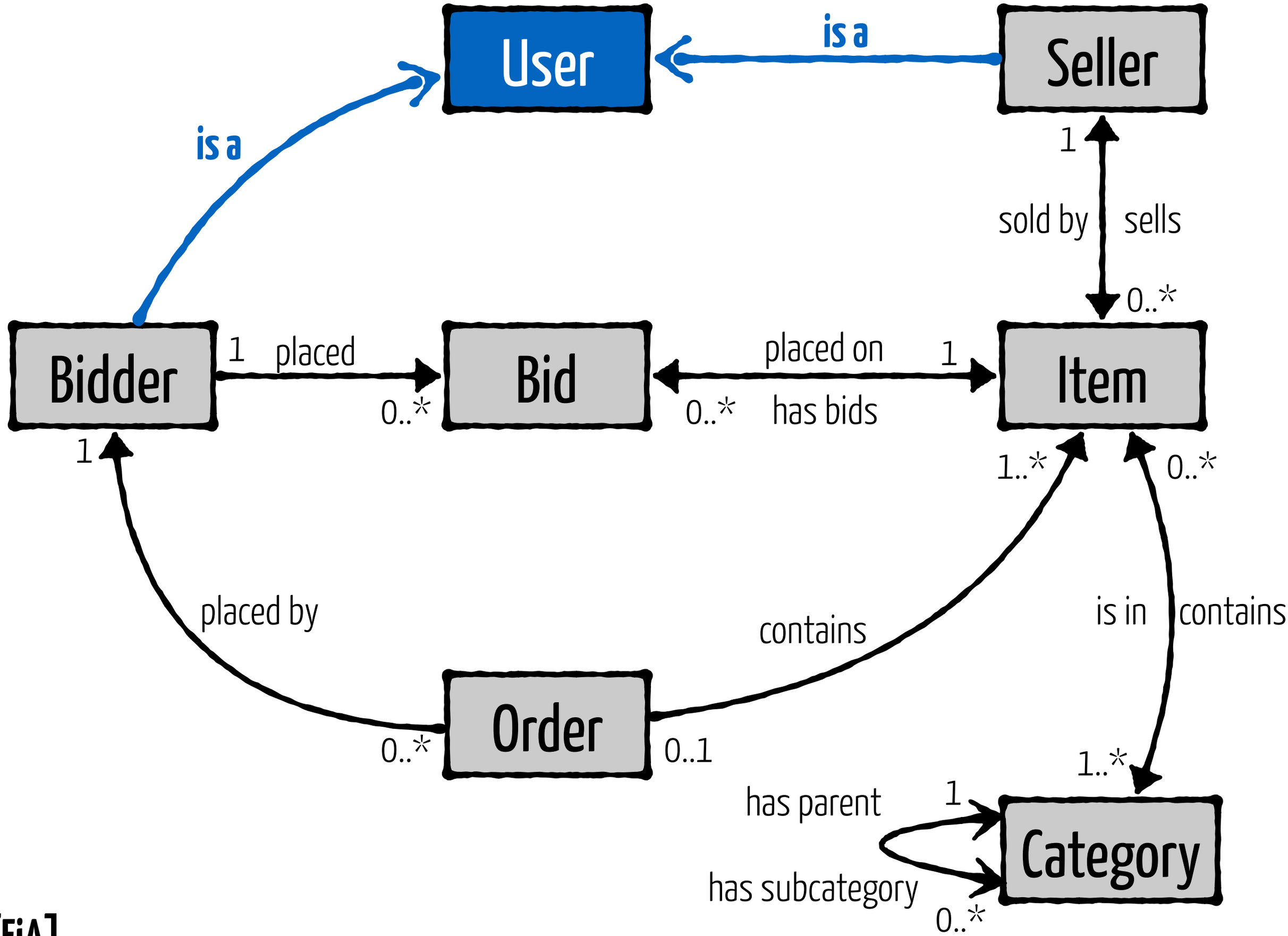
Session & Entity



[EiA]


ActionBazaar
Search Items
Browse Categories
Bid On Items
Order Items Won
Bidder
Post Item
To Category
Seller

[EiA]


User
is a
Seller
is a
1
sold by
sells
0..*
Bidder
1
placed
0..*
Bid
placed on
1
0..*
has bids
Item
1
1..*
0..*
placed by
contains
is in
contains
Order
0..*
0..1
1..*
has parent
1
Category
has subcategory
0..*

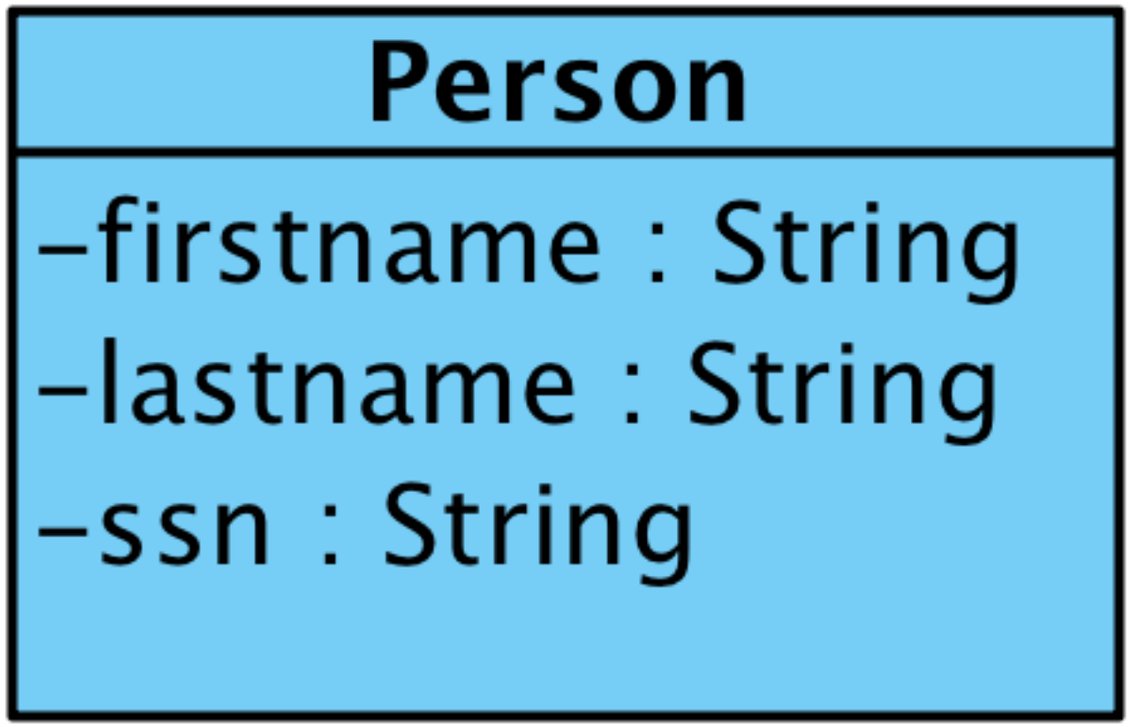
[EiA]

Impedance Mismatch

Object-Oriented	Relational
Classes	Relation (table)
Object	Tuple (row)
Attribute	Attribute (column)
Identity	Primary Key
Reference	Foreign Key
Inheritance	N/A
Methods	~ Stored Procedure

[EJB3 in Action]

Example of **Domain Model**



first_name	last_name	ssn
Sébastien	MOSSER	16118325358
...		

[PoEAA]

EJB Entities need more than simple annotations:

- An empty constructor
- A proper equals method that relies on business elements
- A proper hashCode method to support objects identification in caches

Category

```
@Entity
public class Category {

  public Category() { … }

  protected String name;

  public String getName() {
    return this.name;
  }

  public void setName(String n) {
    this.name = n.toUpperCase();
  }
}
```

property-based access

(JPA)

[EiA]

Categor

```
@Entity
public class Category {

  public Category() { … }

  public String name;
}
```

field-based access

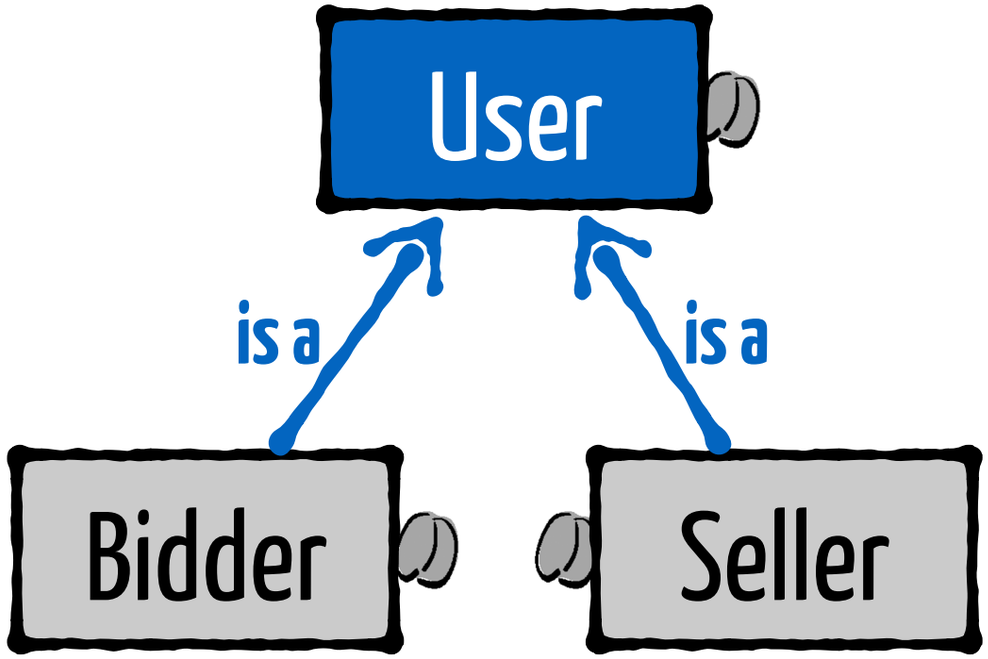
Fields are simple but forbid encapsulation

Do not use fields

We're doing this here just to have examples that fit in a single slide

The container will behave badly with public attributes. Annotate getters.

```
@Entity
public abstract class User {
   // …
}
```




```
@Entity
public class Bidder extends User {
   // …
}

@Entity
public class Seller extends User {
   // …
}
```

[EiA]

Simple Primary Key: @Id

```
@Entity
public class Category {
  // …

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  public Long id;

}
```

Identifiers must define an "equals" method

[EiA]

Composite Key: @IdClass

```
public class CategoryPK extends Serializable {
  String name;
  Date createDate;
}

@Entity
@IdClass(CategoryPK.class)
public class Category {

  @Id
  protected String name;

  @Id
  protected Date createDate;
}
```

[EiA]

Identifiers must define an "equals" method

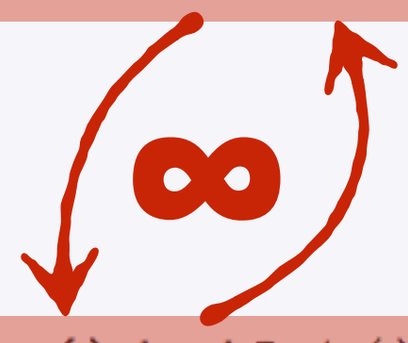
```
public class CategoryPK extends Serializable {

  public boolean equals(Object other) {

    if (other instanceOf CategoryPK) {
      final CategoryPK that = (CategoryPK) other;
      return that.name.equals(name) &&
             that.createDate.equals(createDate);
    return false;

  }

  public int hashCode() {
    return super.hashCode();
  }
}
```

[EiA]

Serializable requirements

Auto-generated equals / hashcode

```
// Customer
public int hashCode() {
        int result = getName() != null ? getName().hashCode() : 0;
        result = 31 * result + (getCreditCard() != null ? getCreditCard().hashCode() : 0);
        result = 31 * result + (getOrders() != null ? getOrders().hashCode() : 0);
        return result;
}

// Order
public int hashCode() {
        int result = getCustomer() != null ? getCustomer().hashCode() : 0;
        result = 31 * result + (getItems() != null ? getItems().hashCode() : 0);
        result = 31 * result + (getStatus() != null ? getStatus().hashCode() : 0);
        return result;
}
```



Never ever use a database primary key as part of your business object equality definition

Equals is used when:

- putting objects in Sets
- when reattaching entities to a new persistence context

Embeddable Objects

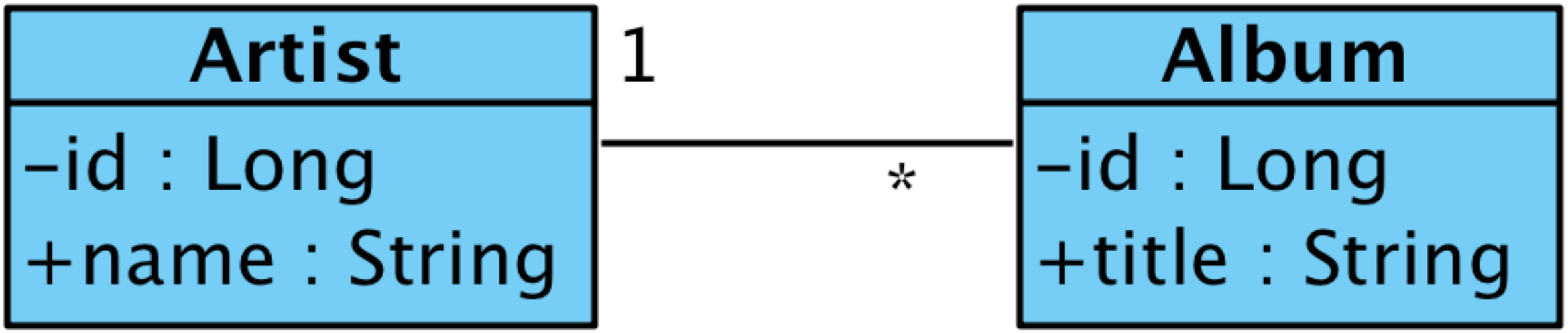
```
@Embeddable
public class Address {
  protected String street;
  protected String city;
  protected String zipcode;
}

@Entity
public class User {

  @Id
  protected Long id;

  @Embedded
  protected Address address;
}
```

does not need an UID

Shared Identifier

[EiA]

Problem: Representing associations



artists

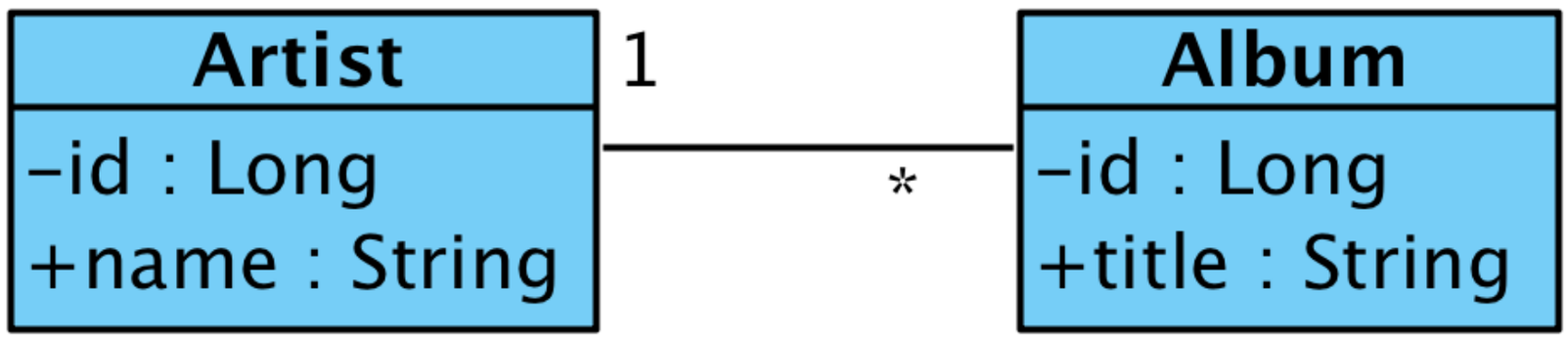
id	name
1	Linkin Park
	...

albums

id	title
1	A Thousand Suns
2	Minutes to Midnight
	...

[PoEAA]

Solution #1: **Association Table** [M-N]



artists

id	name
1	Linkin Park
...	

albums

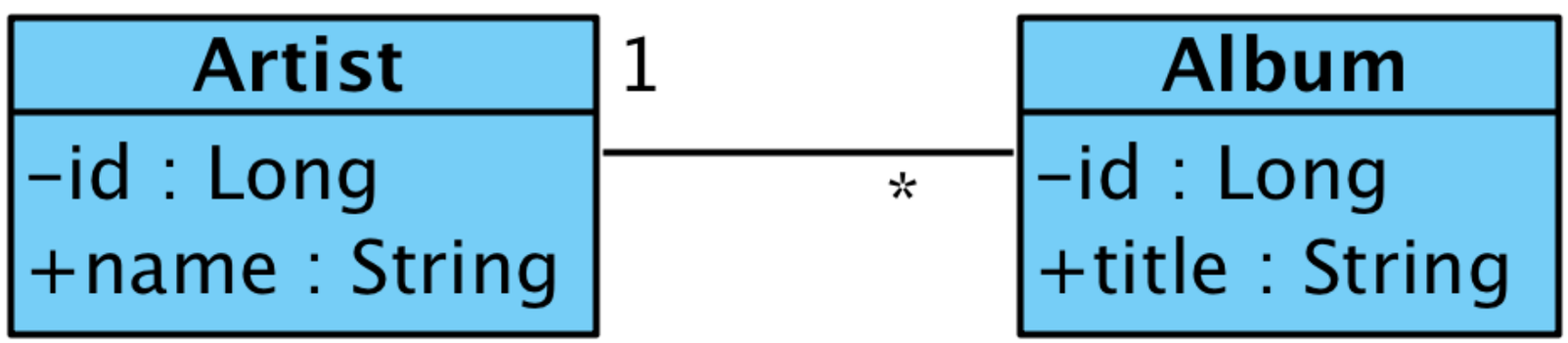
id	title
1	A Thousand Suns
2	Minutes to Midnight
...	

artists_to_albums

artist_id	album_id
1	1
1	2
...	

[PoEAA]

Solution #2: **Foreign Key** [1-N]



artists

id	name
1	Linkin Park
...	

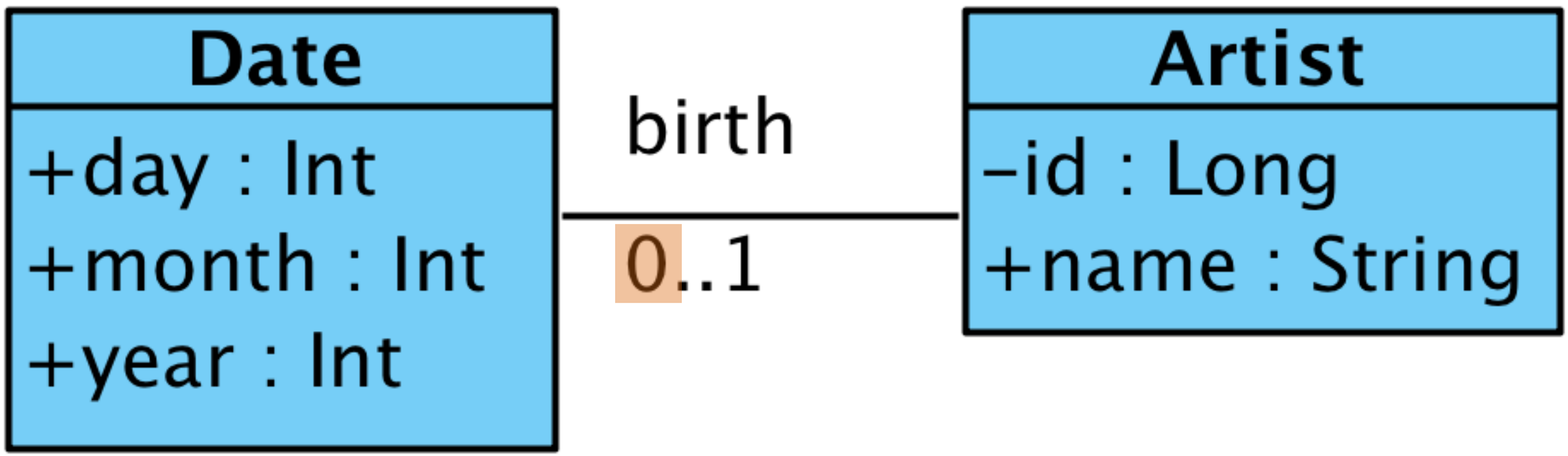
albums

id	title	artist_id
1	A Thousand Suns	1
2	Minutes to Midnight	1
...		

or **[1-N]** ≡ **[M-N]** when N =1

[PoEAA]

Solution #3: **Relation Merge** [1-1]



artists

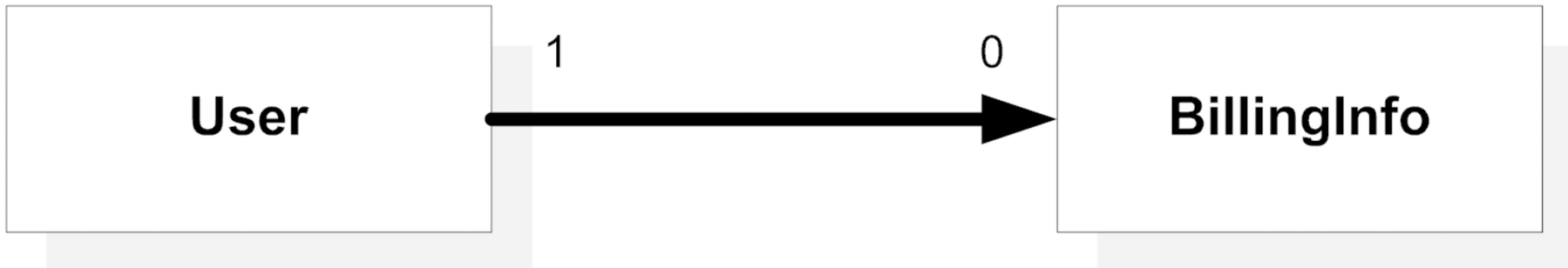
id	name	birth_day	birth_month	birth_year
1	Linkin Park	-1	-1	-1
...				

or **[1-1]** ≡ **[1-N]** when N =1

[PoEAA] or **[1-1]** ≡ **[M-N]** when M = 1 and N =1

Type of Relationship	Annotation
1-1	`@OneToOne`
1-n	`@OneToMany`
n-1	`@ManyToOne`
n-m	`@ManyToMany`

[EiA]



Unidirectional 1-1 mapping

```
@Entity
public class User {
   @Id
   protected String userId;
   protected String email;

   @OneToOne
   protected BillingInfo billingInfo;
}
```

[EiA]

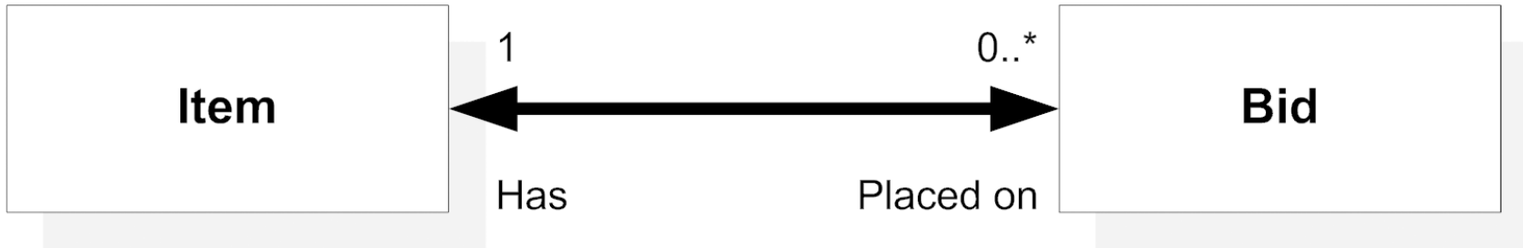
For property-based beans, annotate the getter.

Bidirectional

1-1 mapping

```
@Entity
public class User {
   @Id
   protected String userId;
   protected String email;

   @OneToOne
   protected BillingInfo billingInfo;
}

@Entity
public class BillingInfo {
   @Id
   protected Long billingId;

   @OneToOne(mappedBy="billingInfo",optional=false)
   protected User user;
}
```

[EiA]

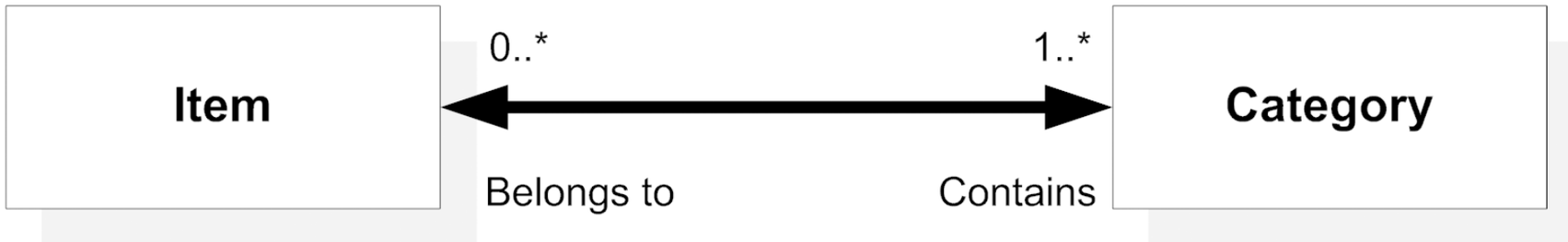



```
@Entity
public class Bid {
   @Id
   protected String bidId;
   @ManytoOne
   protected Item item;
}

@Entity
public class Item {
   @Id
   protected String itemId;
   @OneToMany(mappedBy="item")
   protected Set<Bid> bids;
}
```

Owner

1-n mapping

[EiA]




```
@Entity
public class Category {
   @Id
   protected String categoryId;
   @ManytoMany
   protected Set<Item> items;
}
```

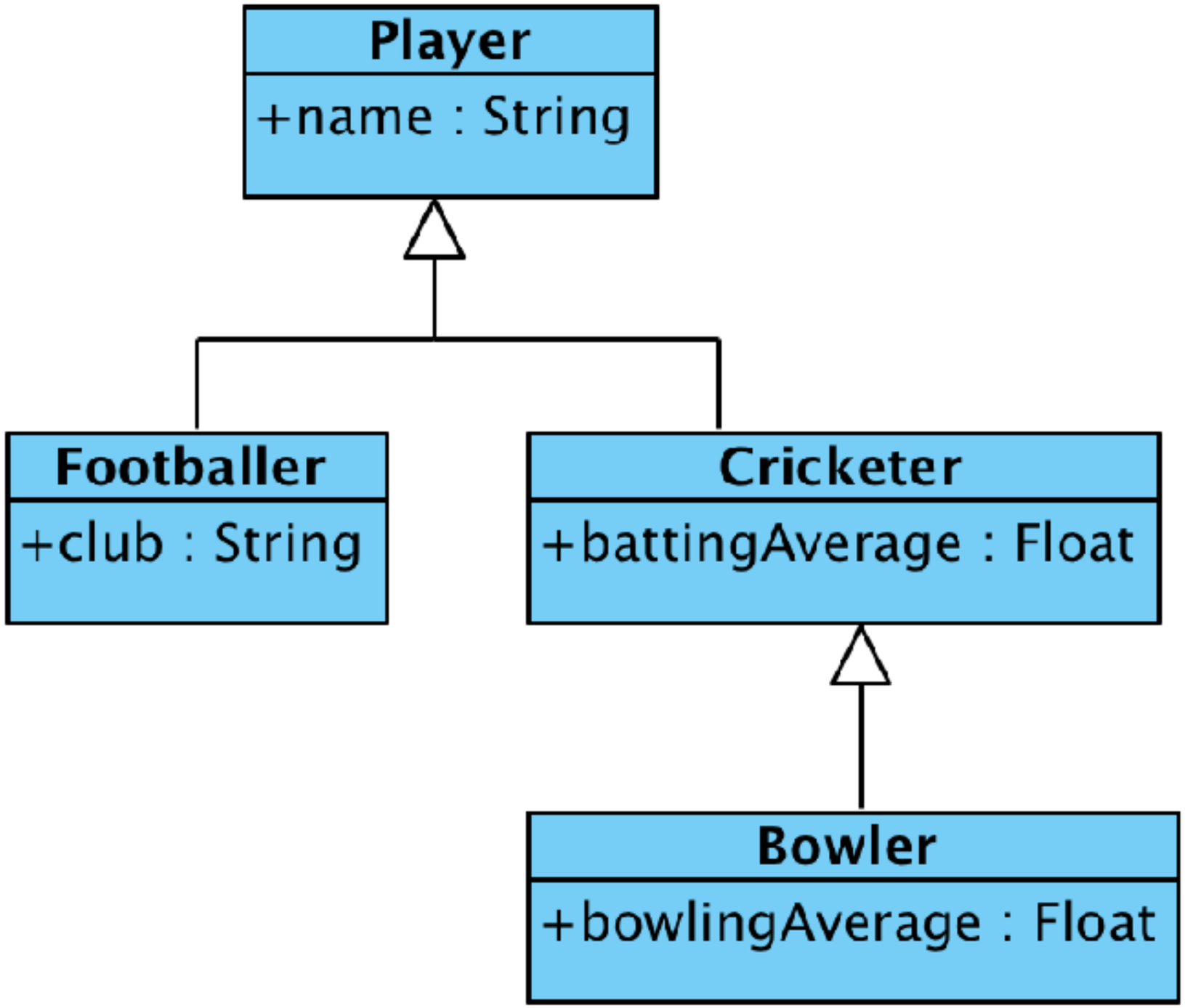
Owner

```
@Entity
public class Item {
   @Id
   protected String itemId;
   @ManytoMany(mappedBy="items")
   protected Set<Category> categories;
}
```

n-m mapping

[EiA]

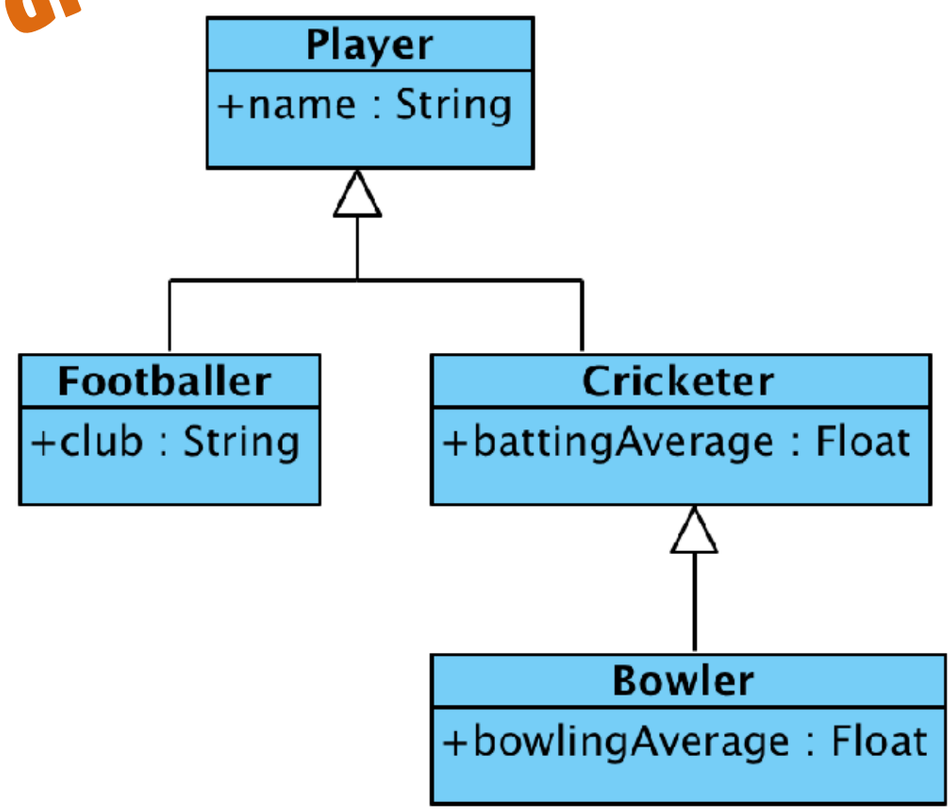
Problem: Implementing Inheritance



[PoEAA]

Solution #1: **Single-Table** Inheritance

Par défaut

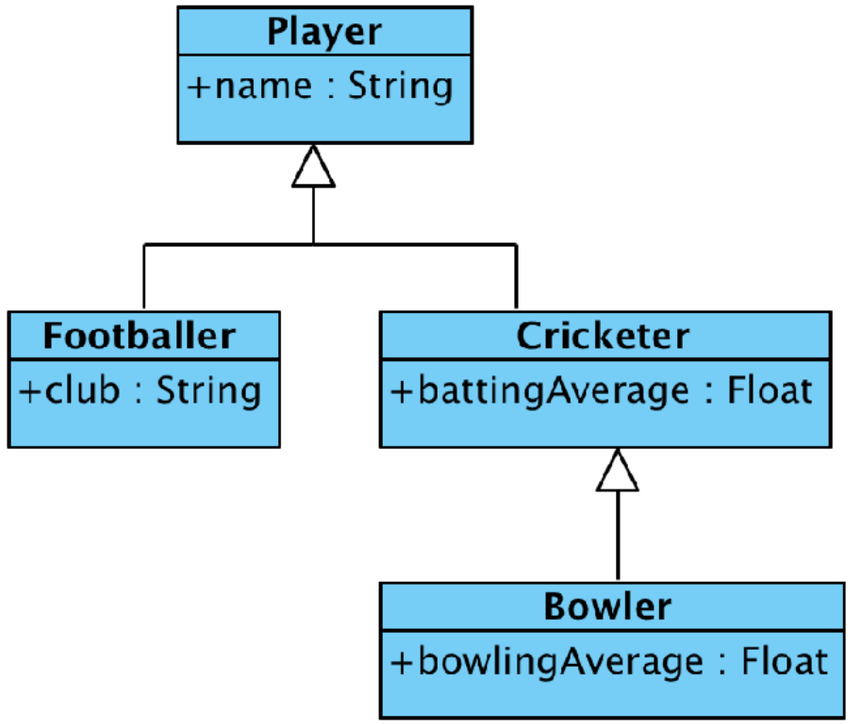


players

name	club	batting_avg	bowling_avg	type

[PoEAA]

Solution #2: **Class-Table** Inheritance



players

id	name
42	...
74	...
96	...

footballers

id	club
42	...

cricketers

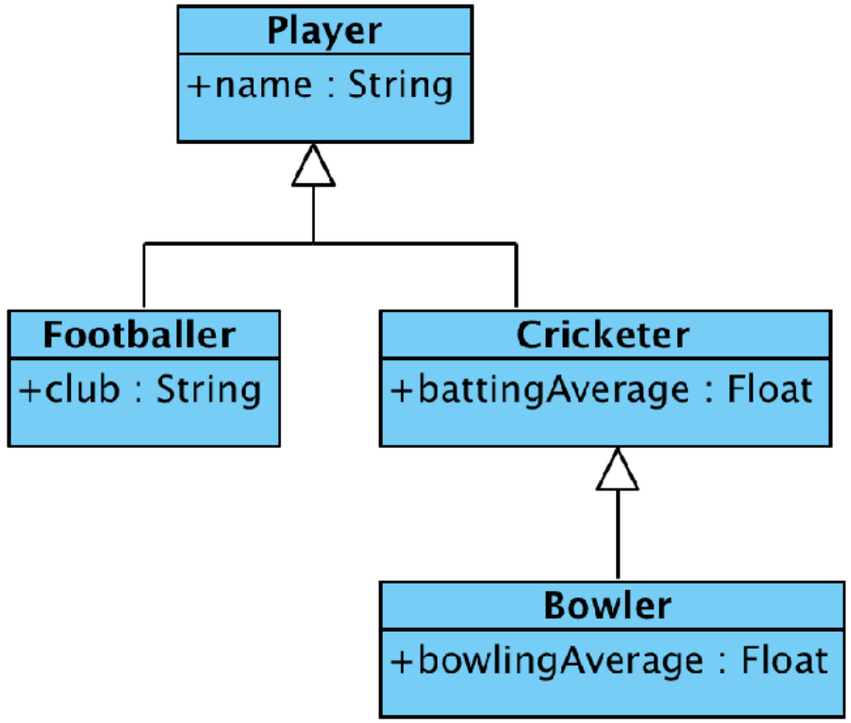
id	batting_avg
74	...
96	...

bowlers

id	bowling_avg
96	...

[PoEAA]

Solution #3: **Concrete-Table** Inheritance



footballers

id	name	club
42	...	...

cricketers

id	name	batting_avg
74	...	...

bowlers

id	name	batting_avg	batting_avg
96	...	...	

[PoEAA]

Controlling Inheritance

```
@Entity
@Table(name="USERS")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="USER_TYPE", …)
public class User {
   // …
}

@Entity
@DiscriminatorValue(value="S")
public class Seller extends User { … }

// …
```

See [EiA], chapter 9

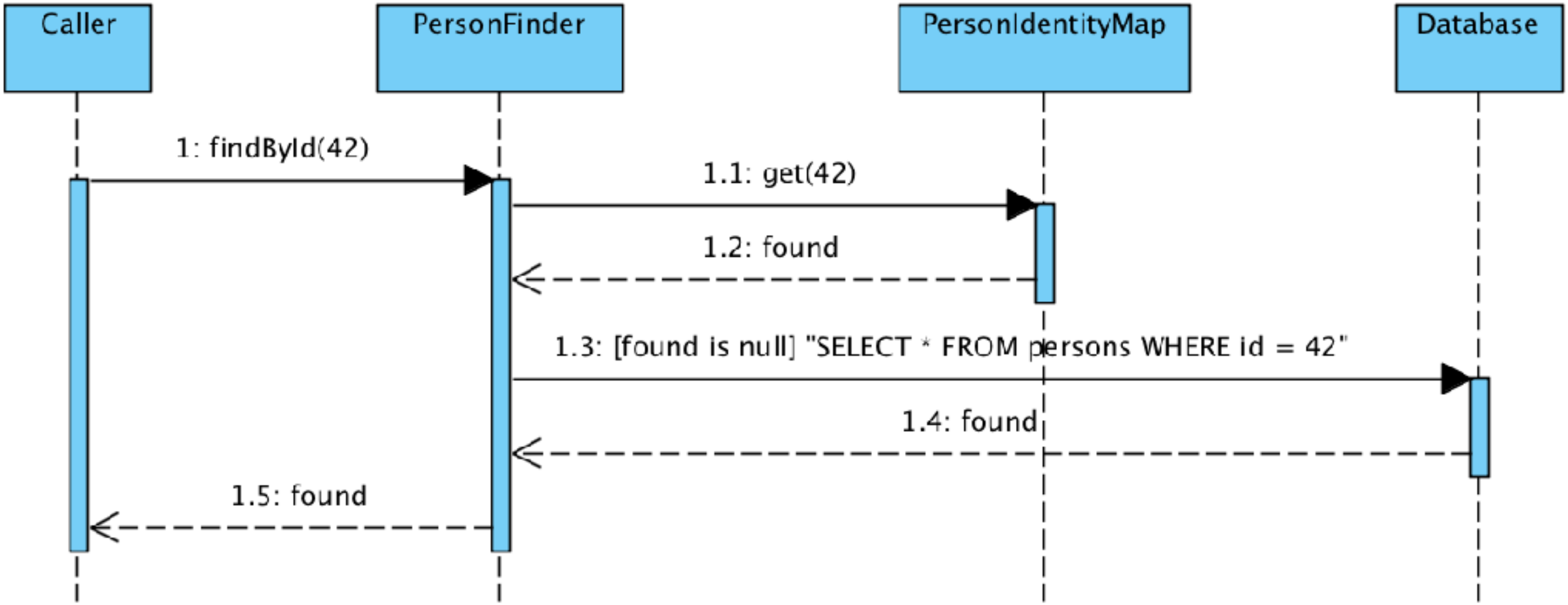
[EiA]

Make your beans **persistent**

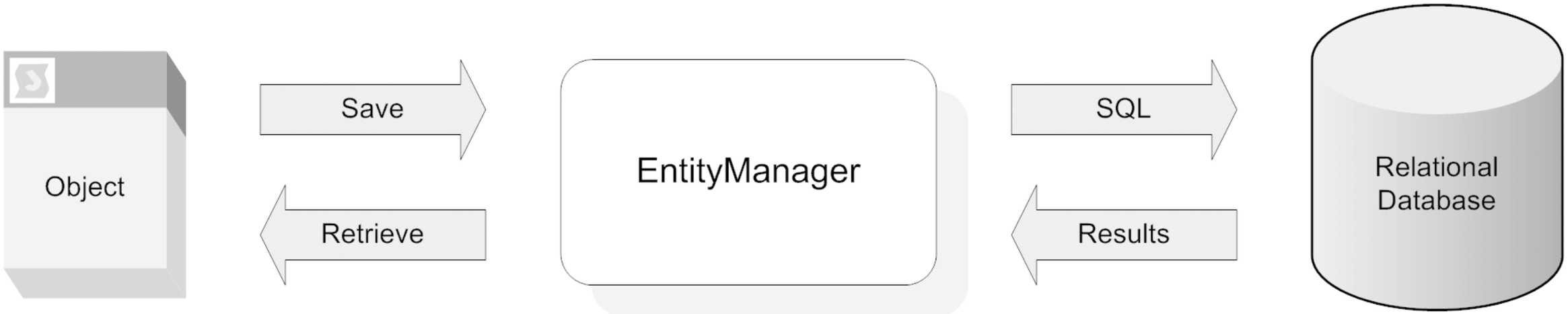
Again...

How to accelerate the access to the persistent layer?

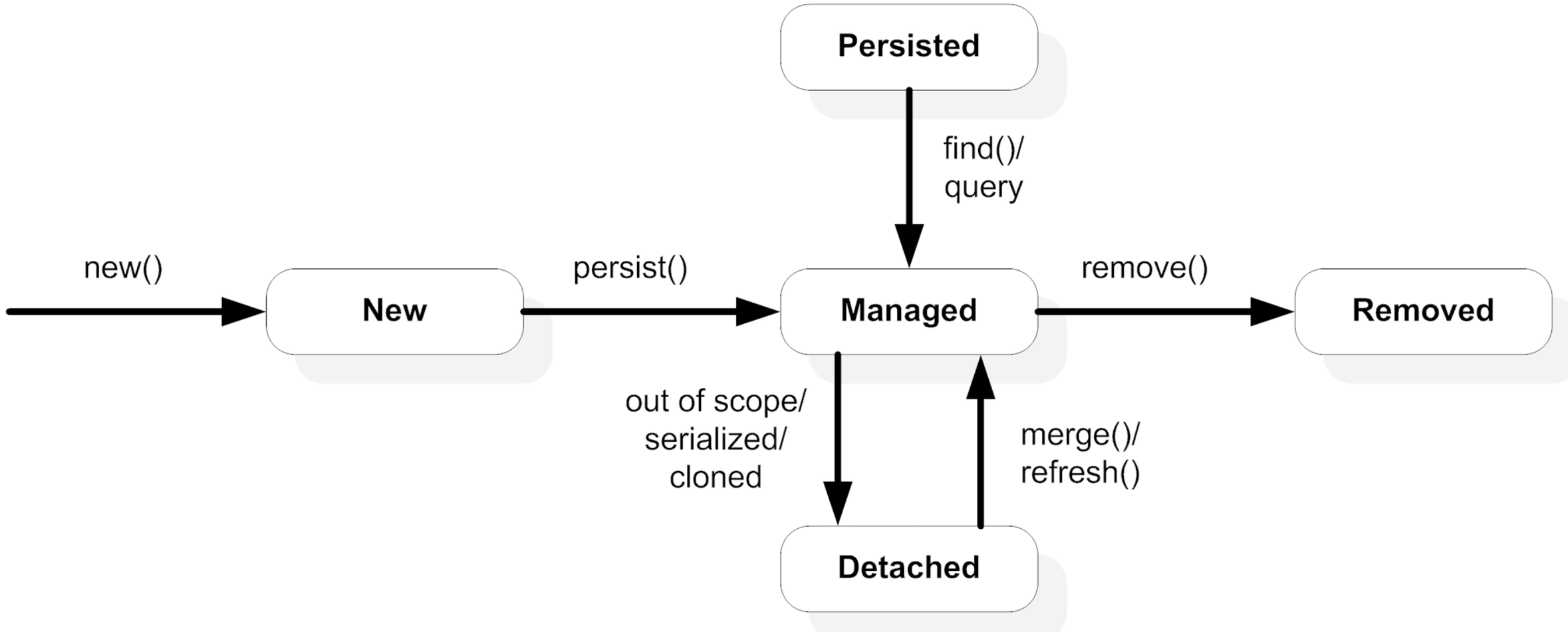
The Identity Map Pattern



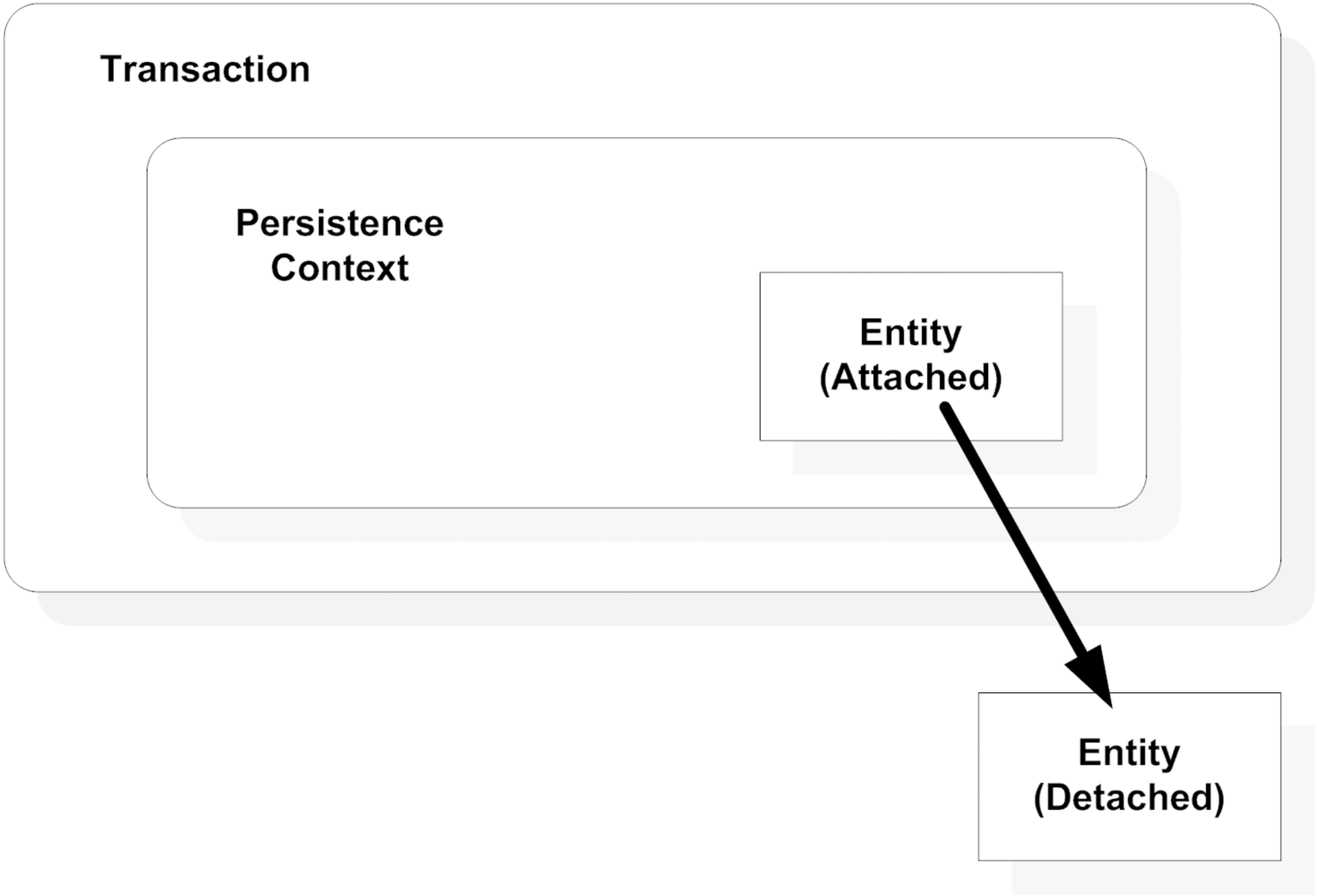
[PoEAA]


Object
Save
Retrieve
EntityManager
SQL
Results
Relational
Database

[EiA]


Persisted
find()/
query
new()
New
persist()
Managed
remove()
Removed
out of scope/
serialized/
cloned
merge()/
refresh()
Detached

[EiA]


Transaction
Persistence Context
Entity (Attached)
Entity (Detached)

[EiA]

Persistence context is **Injected**

```
@PersistenceContext(unitName="admin")
EntityManager manager

@Resource
private UserTransaction transaction;

public void createAndStore() {

  AnEntityBean b = new AnEntityBean("Parameters");
  transaction.begin();
  try {
    manager.persist(b);
  } finally {
    transaction.commit();
  }

}
```

See [EiA], chapter 10

Advanced concepts & tricks...

Stop!

https://github.com/collet/4A_ISA_TheCookieFactory/blob/develop/chapters/Persistence.md

First