

Mappings and Queries

with

Hibernate

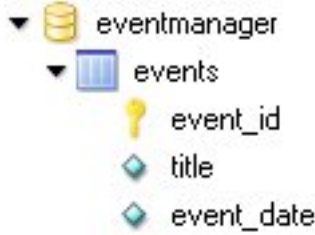
Mappings

- Collection mapping
- Association mapping
- Component mapping

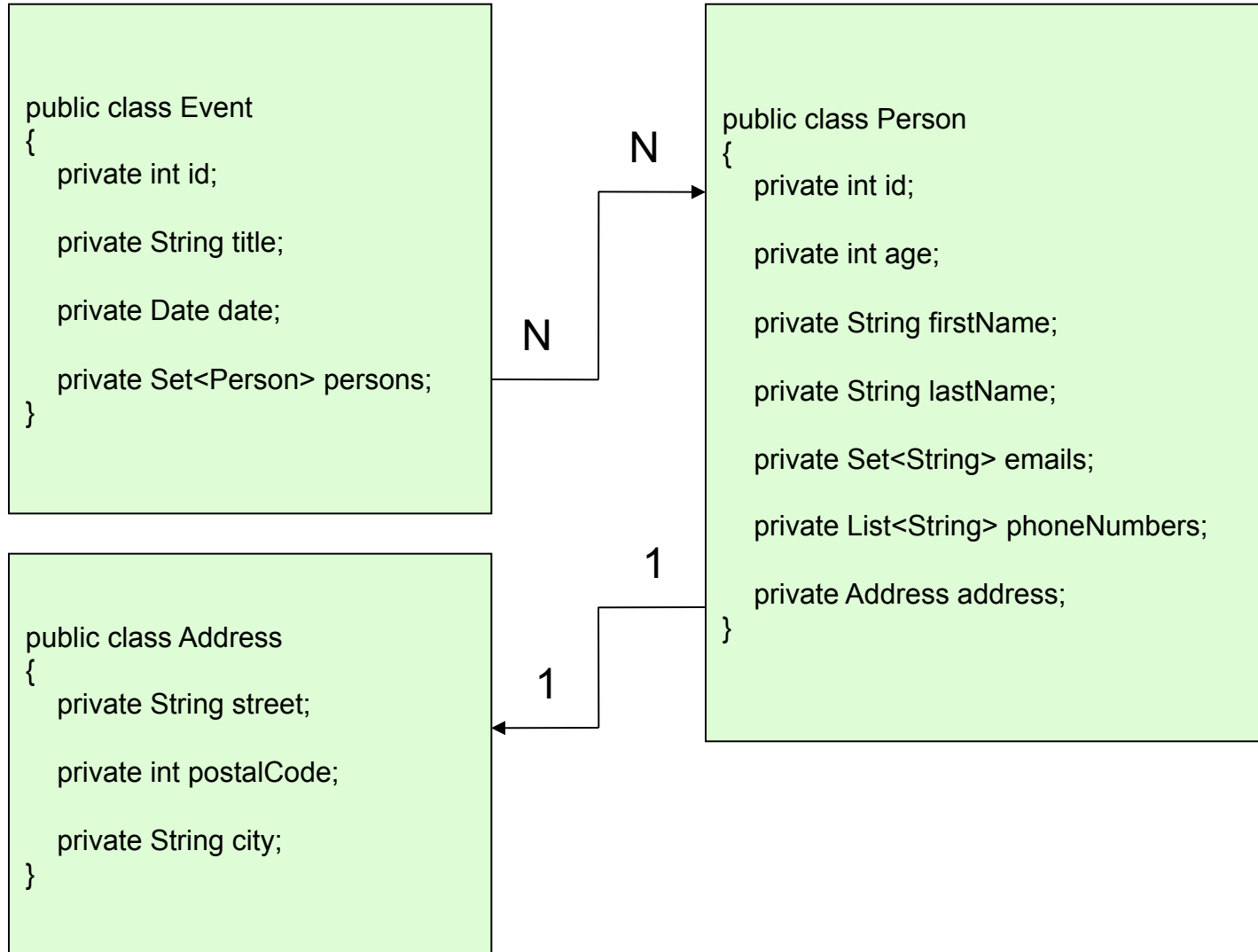
Revision

- Hibernate is an object-relational mapping framework
- Maps persistence operations between object models to relational databases
- Core elements in a Hibernate application are:
 - Your Java objects
 - The Hibernate object mapping files (Event.hbm.xml)
 - The Hibernate configuration file (Hibernate.cfg.xml)
 - Classes working with the Hibernate API (Session, Transaction)

```
public class Event
{
    private int id;
    private String title;
    private Date date;
    private Set<Person> persons;
}
```



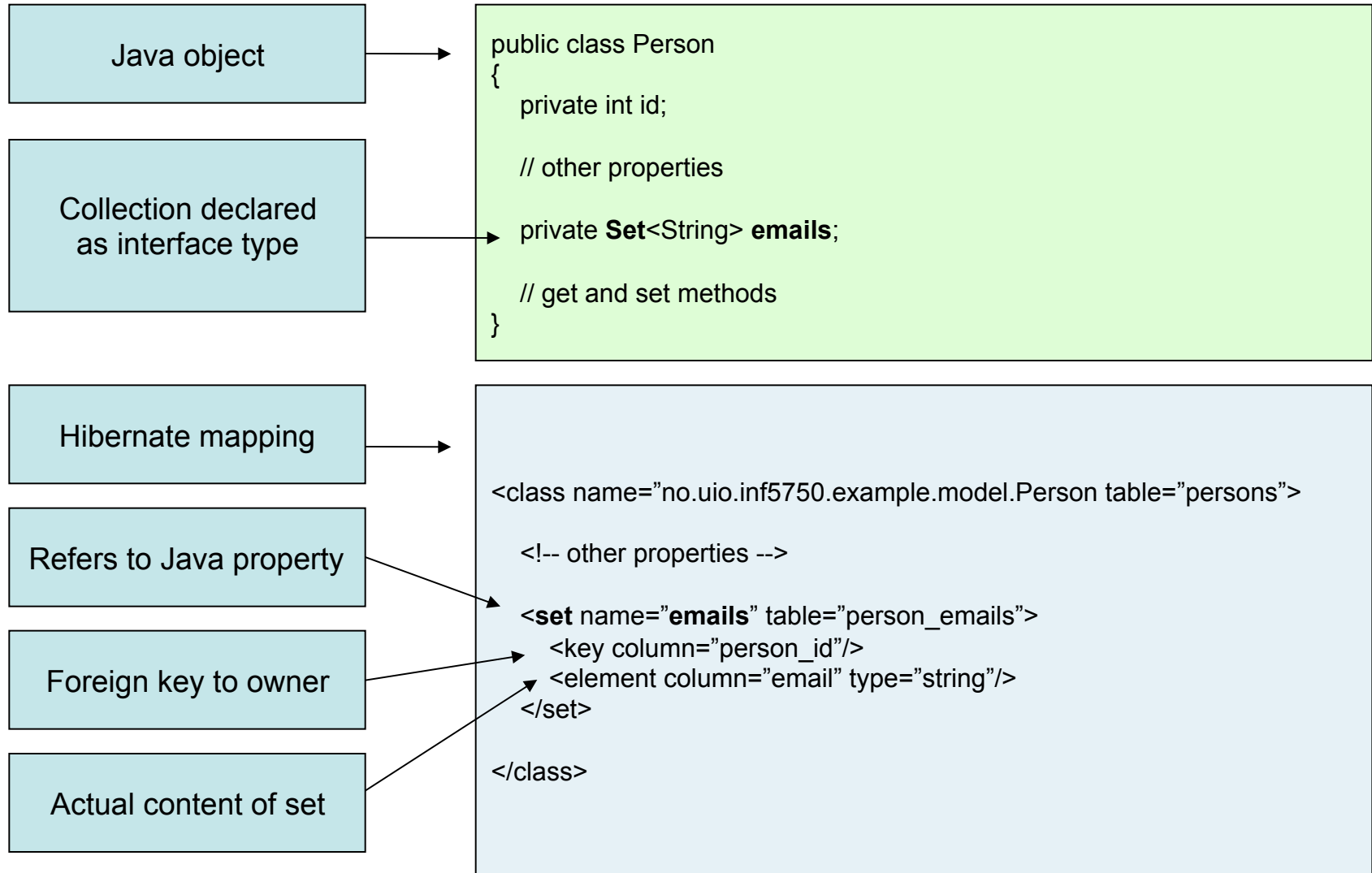
Example: The EventManager



Collection mapping

- Collection properties must be declared as an interface type (Set, not HashSet)
- Hibernate provides built-in mapping for Set, Map, List, and more
- May contain basic types, custom types and references to other Hibernate objects (entities)
- Collections are represented by a *collection table* in the database
 - Collection key: foreign key of owning object
 - Collection element: object in the collection

Collection mapping



Indexed collections

- All *ordered* collection mappings need an *index column* in the collection table to persist the sequence
- Index of List is always of type Integer, index of Map can be of any type

Indexed collection mapping

List is an ordered
type of collection

```
public class Person
{
    private int id;

    // other properties

    private List<String> phoneNumbers;

    // get and set methods
}
```

List mapped to table

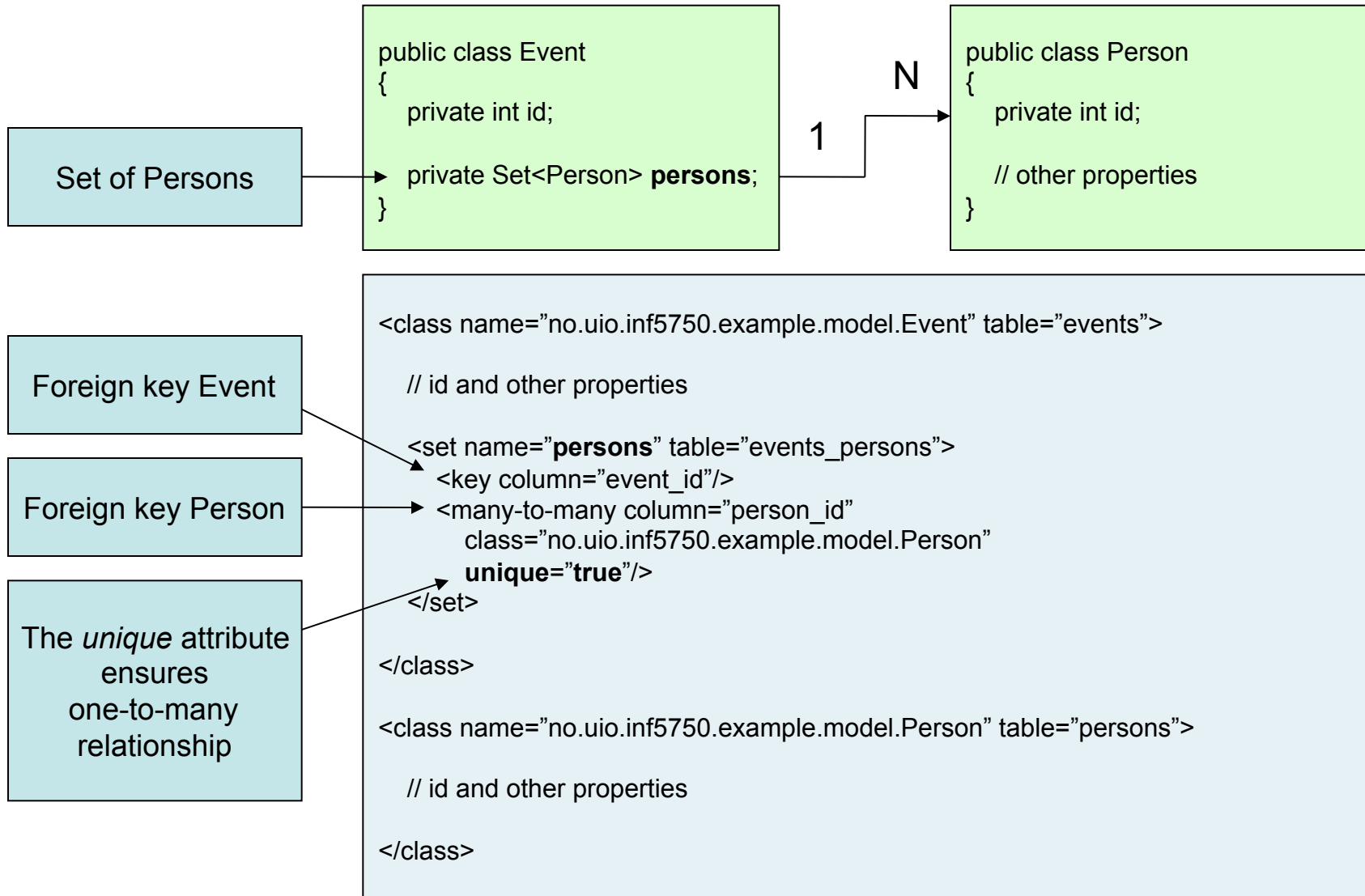
Required mapping of
index column

```
<class name="no.uio.inf5750.example.model.Person table="persons">
    <!-- other properties -->
    <list name="phoneNumbers" table="phone_numbers">
        <key column="person_id"/>
        <list-index column="sort_order" base="0"/>
        <element column="phone_number" type="string"/>
    </list>
</class>
```

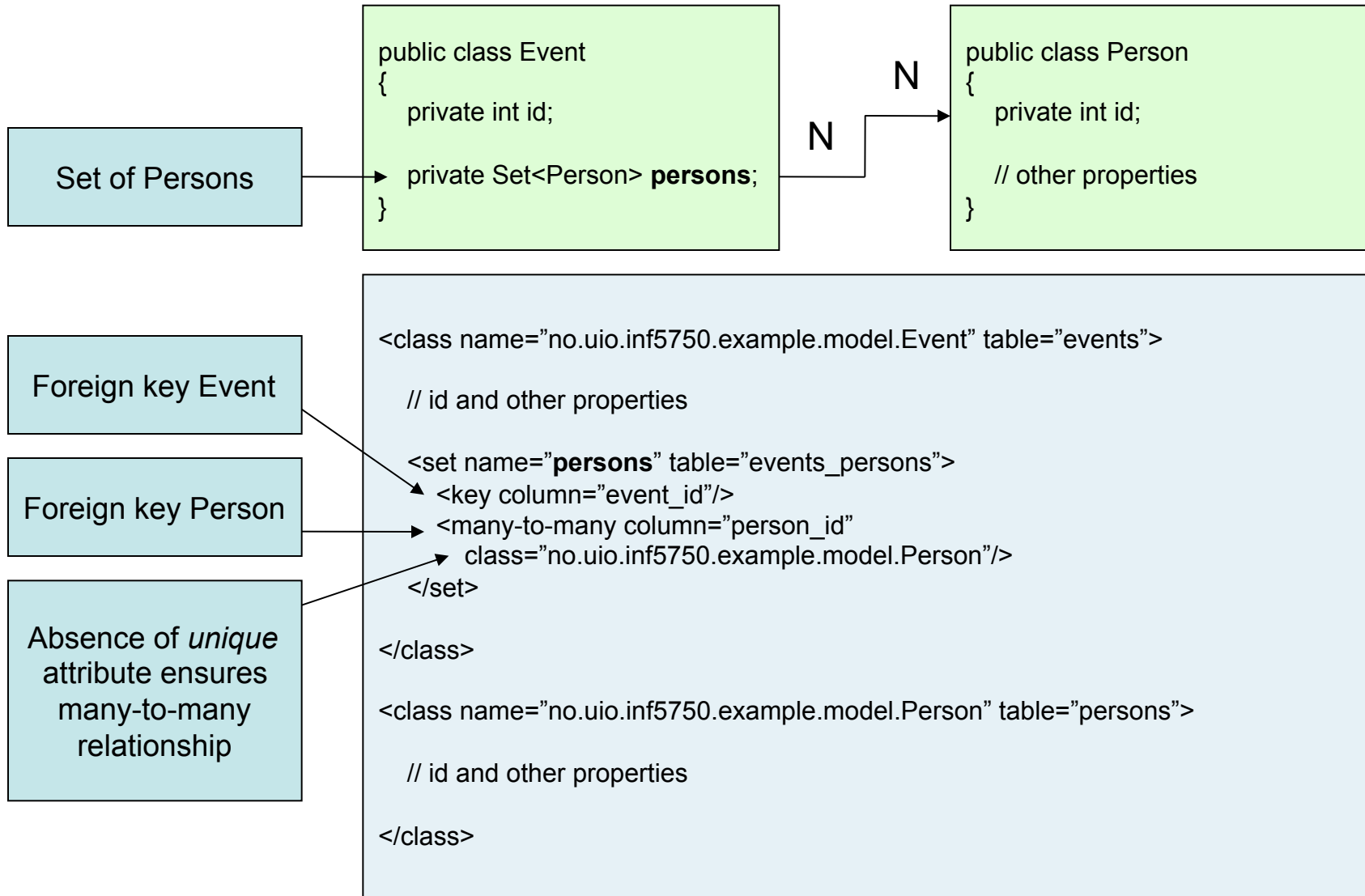

Association mapping

- Hibernate lets you easily specify all kinds of associations between objects
 - Unidirectional one-to-many
 - Unidirectional many-to-many
 - Bidirectional one-to-many
 - Bidirectional many-to-many
- Representing associations with join tables makes the database schema cleaner
- Nullable foreign keys bad practise

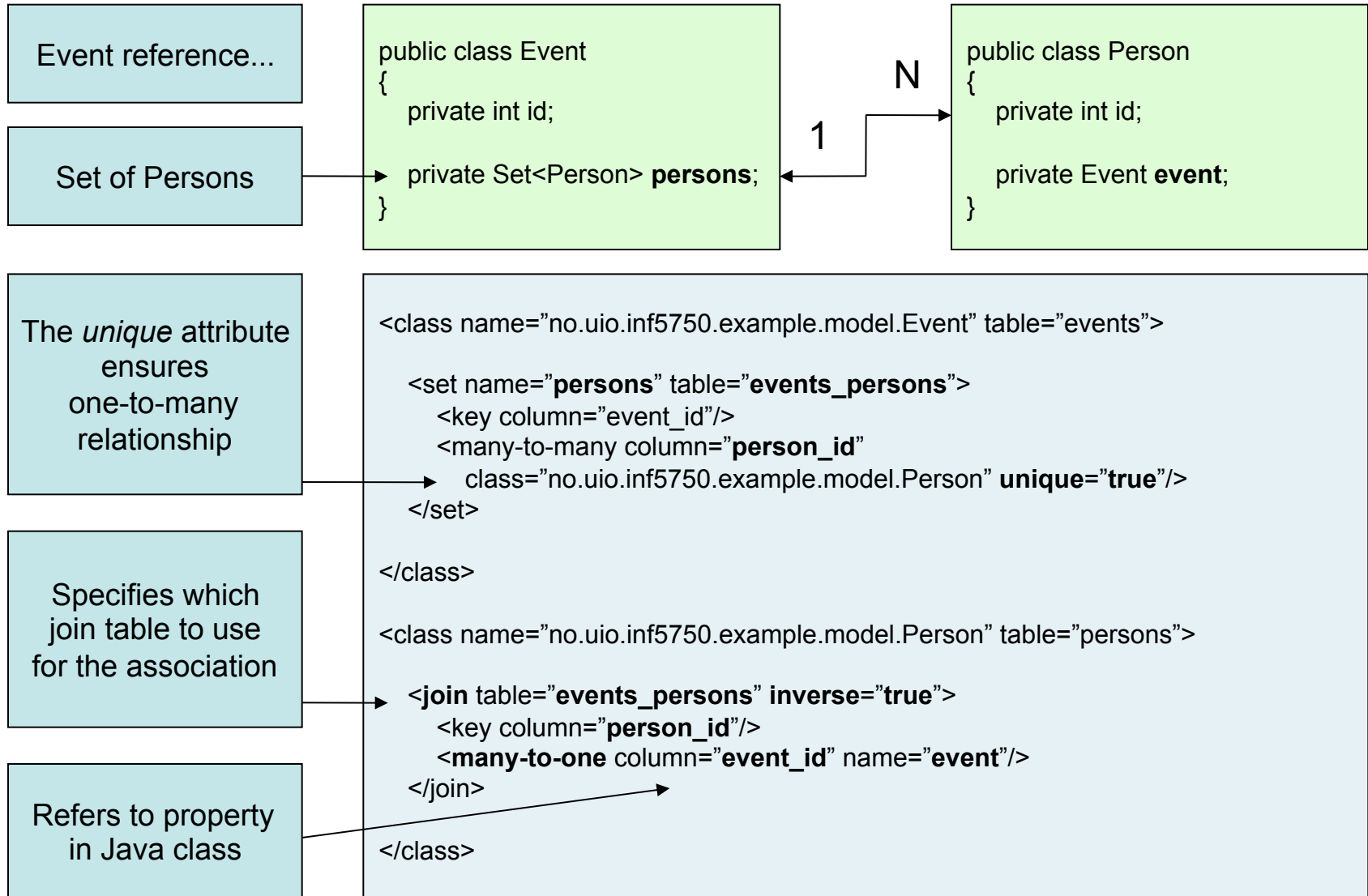
Unidirectional one-to-many



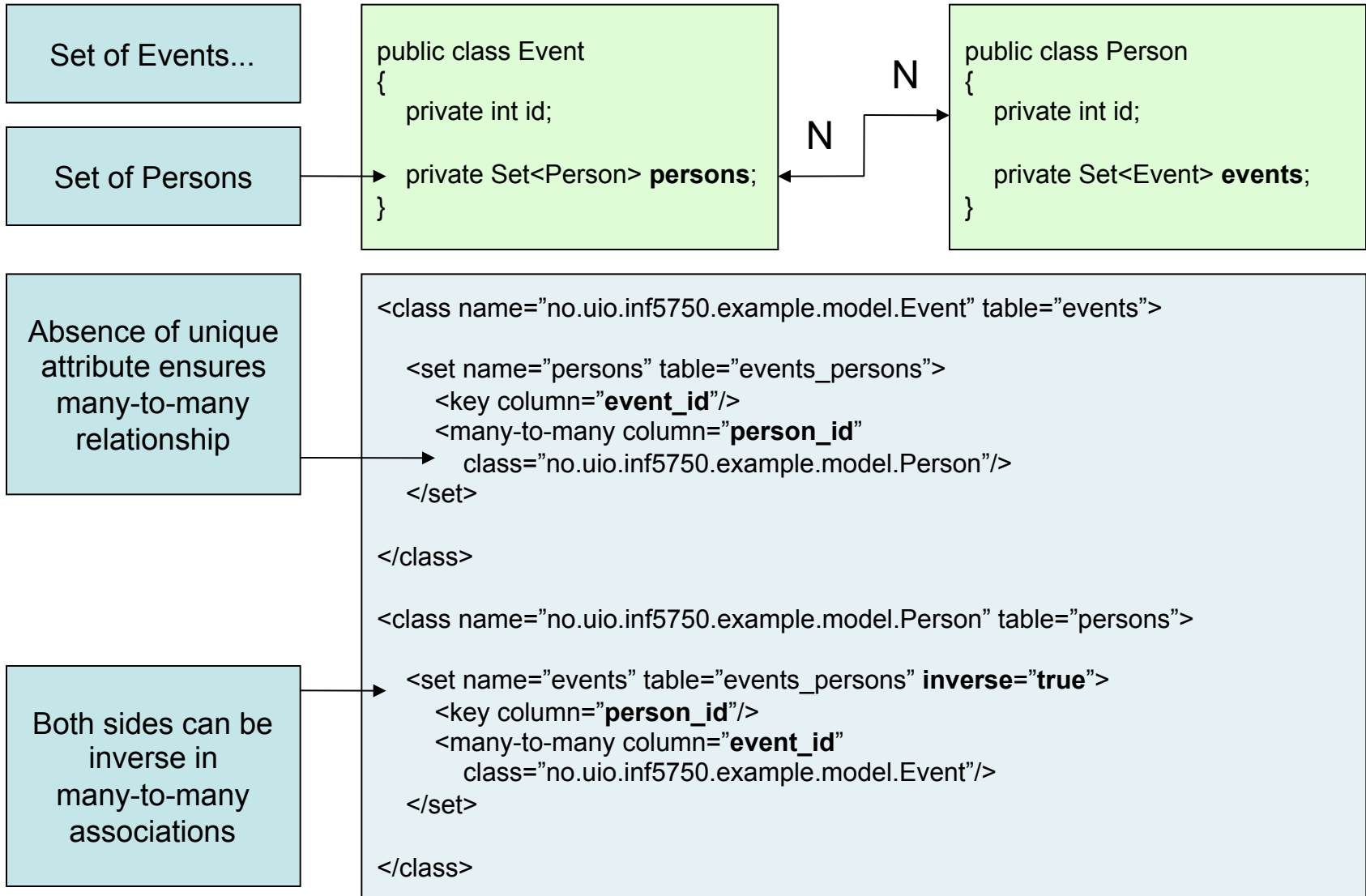
Unidirectional many-to-many



Bidirectional one-to-many

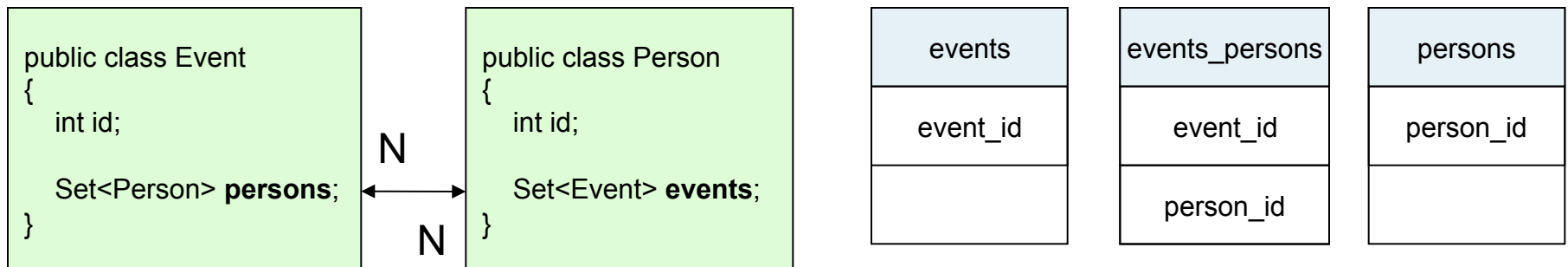


Bidirectional many-to-many



The *inverse* property explained

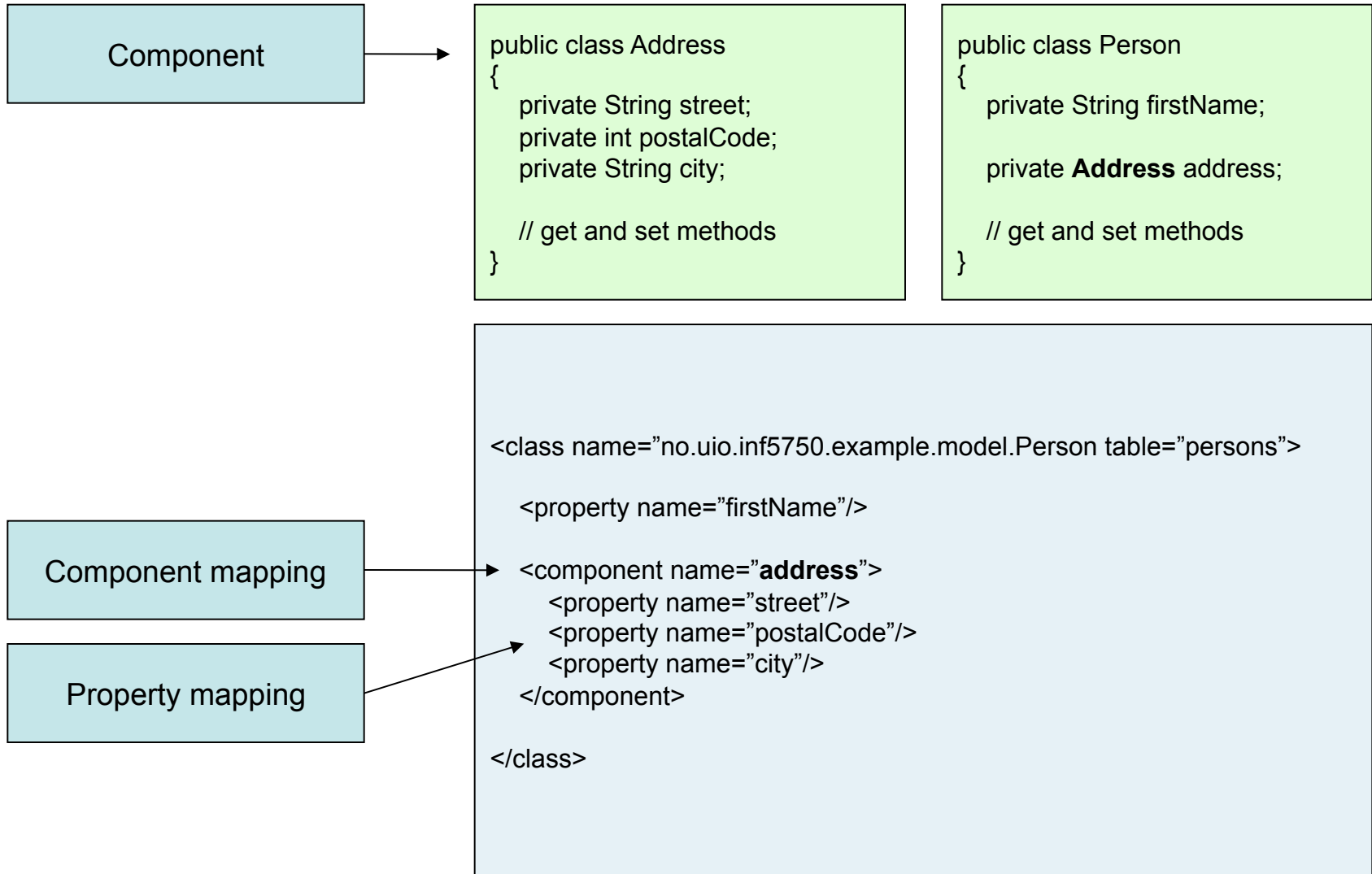
- Bidirectional associations should be updated *on both sides* in the Java code!
- Hibernate maps many-relationships with a *join table*
- Hibernate must ignore one side to avoid constraint violations!
- Must be *many*-side on one-to-many, doesn't matter on many-to-many



Component mapping

- A component is an object saved as a value, not as a entity reference
- Components do not support shared references
- Properties can be of any Hibernate type

Component mapping



Queries

- The Query interface
- The Hibernate Query Language (HQL)

The Query interface

- You need a *query* when you don't know the identifiers of the objects you are looking for
- Used mainly to execute Hibernate Query Language queries
- Obtained from a Hibernate Session instance
- Provides functionality for:
 - Parameter binding to *named query parameters*
 - Retrieving lists of objects or unique objects
 - Limiting the number of retrieved objects

```
Query query = session.createQuery( "some_HQL_query" );
```

The Hibernate Query Language

- HQL is an *object-oriented* query language
 - Syntax has similarities to SQL
 - Not working against tables and columns, but objects!
- Understands object-oriented concepts like inheritance
- Has advanced features like:
 - Associations and joins
 - Polymorphic queries
 - Subqueries
 - Expressions
- Reduces the size of queries

The from clause

Simplest possible query, qualified class name auto-imported, will return all Person instances:

```
from Person
```

Convenient to assign an alias to refer to in other parts of the query:

```
from Person as p
```

Multiple classes may be desired. The alias keyword is optional:

```
from Person p, Event e
```

The where clause

Allows you to narrow the returned list, properties can be referred to by name:

```
from Person where firstName='John'
```

If there is an alias, use a qualified property name:

```
from Person p where p.lastName='Doe'
```

Compound path expressions are powerful:

```
from Person p where p.address.city='Boston'
```

Expressions

In clause:

```
from Person p where p.firstName in ( 'John', 'Tom', 'Greg' )
```

Between and *not* clause:

```
from Person p where p.lastName not between 'D' and 'F'
```

Size clause:

```
from Person p where size ( p.address ) > 2
```

Query examples

HQL query with *named query parameter (age)*

Query obtained from Session

Age parameter binding

Max nr of objects restriction

Returns the result as a List

```
public Collection<Person> getPersonsByAge( int age, int maxResults )
{
    Session session = sessionFactory.getCurrentSession();

    String hql = "from Person where age = :age";

    Query query = session.createQuery( hql );

    query.setInteger( "age", age );

    query.setMaxResults( maxResults );

    return query.list();
}
```

Query examples

HQL query with *named query parameters*

Create query and pass in HQL string as parameter

Parameter binding with the `setString` methods

uniqueResult offers a shortcut if you know a single object will be returned

```
public Person getPerson( String firstName, String lastName )
{
    Session session = sessionFactory.getCurrentSession();

    String hql = "from Person where firstName = :firstName " +
        "and lastName = :lastName";

    Query query = session.createQuery( hql );

    query.setString( "firstName", firstName );

    query.setString( "lastName", lastName );

    return (Person) query.uniqueResult();
}
```


Resources

- Books on Hibernate
 - Christian Bauer and Gavin King: Java Persistence with Hibernate
 - Christian Bauer and Gavin King: *Hibernate in Action*
- The Hibernate reference documentation
 - www.hibernate.org