The difference between a Bag and a List

Hibernate's naming of the different collection types is a little bit confusing because *List*s and *Bag*s are both mapped by a *java.util.List*. The difference between them is that a *List* is ordered and a *Bag* is unordered.

You should prefer the unordered *Bag* for most mappings because retrieving the association in a specific order slows down your database queries. You should better use a JPQL query with an ORDER BY clause to define the ordering if you need it.

Should you use a Bag or a Set?

When you just look at the Java types, the answer seems to be easy. In general, a *java.util.List* provides the better performance while a *java.util.Set* doesn't contain any duplicates. As long as you implement the create use case correctly, a *java.util.List* seems like the obvious best choice for your association mapping.

But it's not that easy. A *List* might be more efficient than a *Set*, but the type also influences how Hibernate manages the association in the database. So, there are a few other things you need to take into account when you make your decision.

A critical bug in older Hibernate versions

If you're using a Hibernate version older than 5.0.8, you should be aware of bug HHH-5855. When you used a *java.util.List* and merged the parent entity, Hibernated generated 2 INSERT statements for each new child entity.

Inefficient handling of many-to-many associations

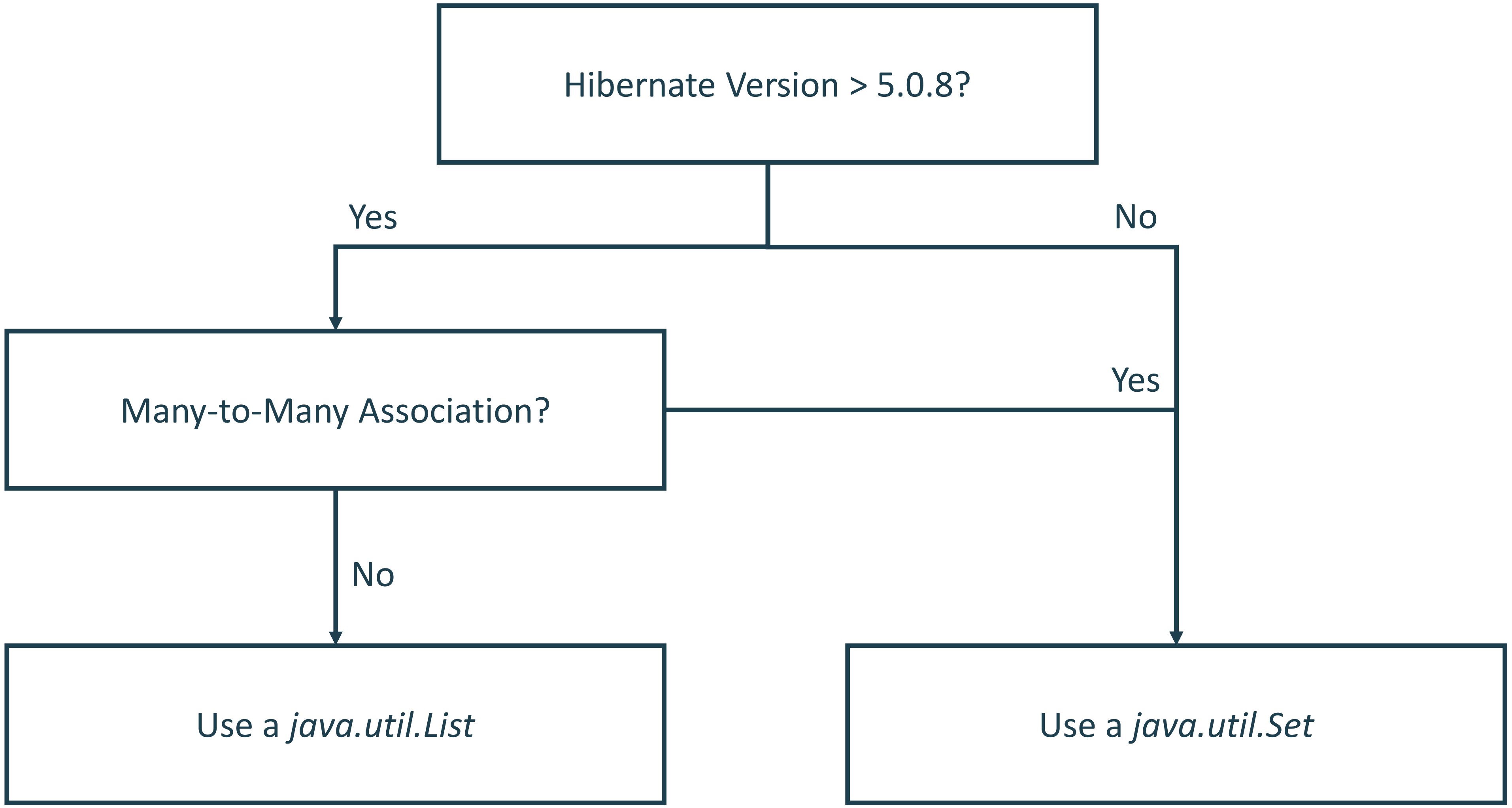
When you're mapping a many-to-many association, you should always use a java.util.Set.

```
@Entity
public class Book {


    @ManyToMany
    @JoinTable(name = "book_author",
            joinColumns = { @JoinColumn(name = "fk_book") },
            inverseJoinColumns = { @JoinColumn(name = "fk_author") })
    private Set authors = new HashSet();


    ...
}
```

If you model the association as a java.util.List, Hibernate handles the removal of associated entities very inefficiently. Instead of removing the record that maps the removed association, Hibernate removes all records from the association table before it inserts a new record for the remaining associations.

Summary

As you've seen, mapping an association as a *java.util.List* can create problems which by far outweigh the small performance gain you get compared to a *java.util.Set*. So, better make sure to update your Hibernate version and to use a *Set* to model many-to-many associations.


Hibernate Version > 5.0.8?
Yes
No
Many-to-Many Association?
Yes
No
Use a *java.util.List*
Use a *java.util.Set*