# How to find and fix N+1 Select Issues with Hibernate

Part 2:

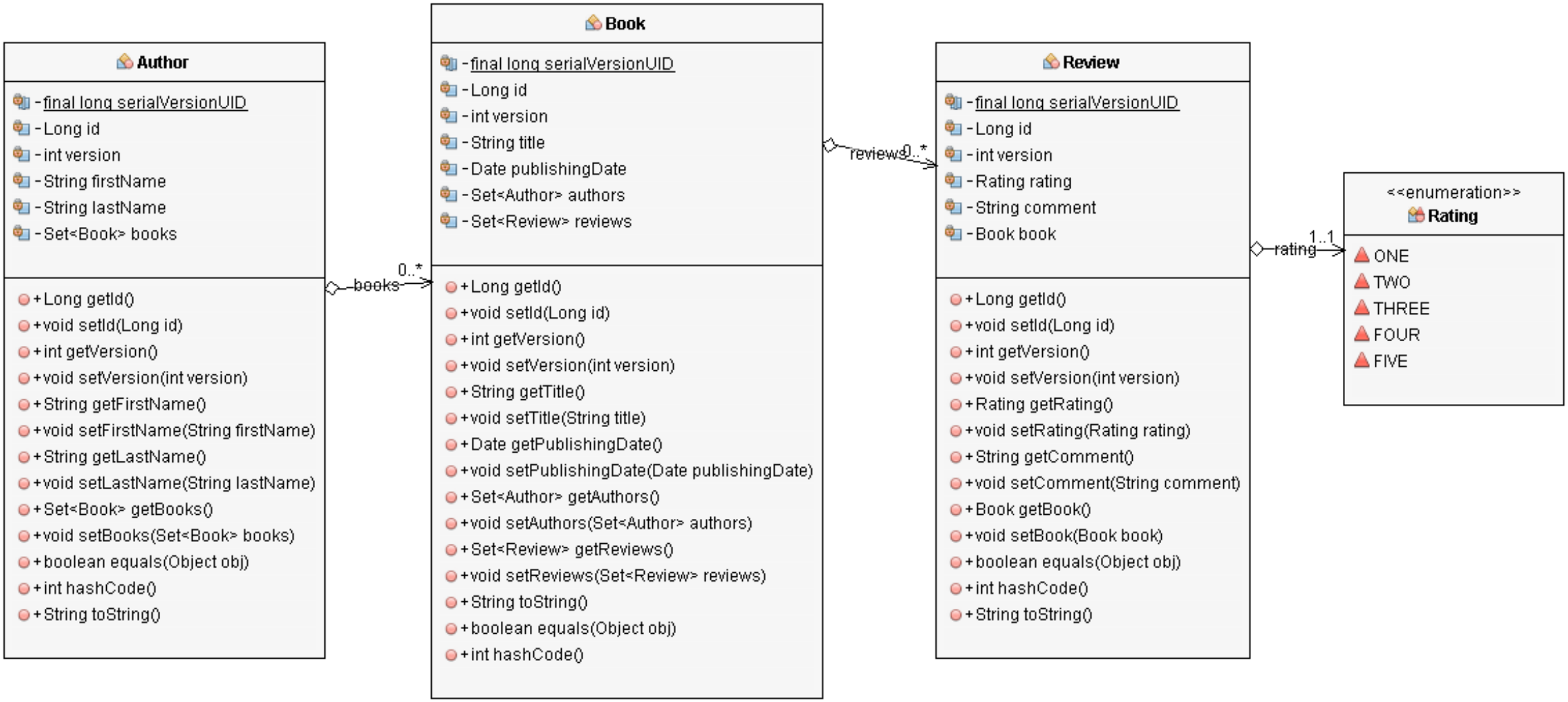
Solving n+1 select issues

with @NamedEntityGraphs

# Course Structure

- Part 1:
  - What is the n+1 select issue?
  - How to find it in your project?
- Part 2:
  - Solving n+1 select issues with @NamedEntityGraphs
- Part 3:
  - Solving n+1 select issues with dynamic EntityGraphs

# Application



# @NamedEntityGraph

# @NamedEntityGraph

- Introduced in JPA 2.1
- Declaratively defines a graph of entities which will be loaded
- Graph is query independent

- Define a simple @NamedEntityGraph

```
@NamedEntityGraph(
        name = "graph.AuthorBooksReviews",

        attributeNodes =
                @NamedAttributeNode(value = "books")
)
```

- Define a multi-level @NamedEntityGraph

```
@NamedEntityGraph(
        name = "graph.AuthorBooksReviews",

        attributeNodes =
                @NamedAttributeNode(value = "books", subgraph = "books"),

        subgraphs =
                @NamedSubgraph(
                        name = "books",
                        attributeNodes = @NamedAttributeNode("reviews")
                )
)
```

- Provide entity graph as a hint

```
EntityGraph graph = this.em.getEntityGraph("graph.AuthorBooks");

this.em.createQuery("SELECT DISTINCT a FROM Author a")
                .setHint("javax.persistence.loadgraph", graph);
```

- Fetch graph
  - Eager loading for all elements of the graph
  - Lazy loading for all other attributes

- Load graph
  - Eager loading for all elements of the graph
  - Loads all other attributes with their defined FetchType

- **Hibernate always uses a load graph**
  - HHH-8776

- Advantages
  - Query specific EAGER loading
  - Definition of the graph is independent of the query
- Disadvantages
  - Creates cartesian product

# Want to learn how to identify and fix other Hibernate performance issues?

Join my

Hibernate Performance Tuning Online Training:

www.thoughts-on-java.org/course-hibernate-performance-tuning