# Exercise: Hibernate Relations Cheat Sheet

This document defines a **cheat sheet for relations** in Hibernate for the ["Databases Advanced – Hibernate" course @ Software University.](https://softuni.bg/trainings/1444/databases-advanced-hibernate-october-2016)

There are 2 types of relationships in Hibernate **unidirectional** and **bidirectional.**

**Unidirectional** means that the relationship is valid in **only one direction**. For example, an operating system is installed on a computer. A computer is not installed on an operating system.

Bidirectional means that the relationship is valid in **both directions**. For example a person owns a phone and the phone is owned by a person.

# Unidirectional

## One-to-Many

The employee knows who his employer is, but the employer does not know who his employees are.

|  |  |
| --- | --- |
| **Employer** = parent table | Employee = child table |
| Employee.java | |
| @Entity **public class** Employee {  **private** Long **id**;  **private** Employer **employer**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **public** Long getId() {  **return id**;  }   **public void** setId(Long id) {  **this**.**id** = id;  }   @ManyToOne  **public** Employer getEmployer() {  **return employer**;  }   **public void** setEmployer(Employer employer) {  **this**.**employer** = employer;  } } | |

## One-to-One

The employee does not know his address but the address knows which employee lives on it.

|  |  |
| --- | --- |
| **Employee** = parent table | **Address** = child table |
| Address.java | |
| @Entity **public class** Address {  **private** Long **employeeId**;  **private** Employee **employee**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **public** Long getEmployeeId() {  **return employeeId**;  }   **public void** setEmployeeId(Long employeeId) {  **this**.**employeeId** = employeeId;  }   @OneToOne(cascade = CascadeType.***ALL***)  @JoinColumn(name = **"employee\_id"**)  **public** Employee getEmployee() {  **return employee**;  }   **public void** setEmployee(Employee employee) {  **this**.**employee** = employee;  } } | |

## Many-to-Many

One book can be written by many authors but the authors do not know which books they published. // …

|  |  |
| --- | --- |
| **Author** = parent table | **Book** = child table |
| Book.java | |
| @Entity **public class** Book {  **private** Long **bookId**;  **private** Set<Author> **authors**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **public** Long getBookId() {  **return bookId**;  }   **public void** setBookId(Long bookId) {  **this**.**bookId** = bookId;  }   @ManyToMany(cascade = CascadeType.***ALL***)  @JoinTable(  name = **"author\_book"**,   joinColumns = @JoinColumn(name = **"book\_id"**),   inverseJoinColumns = @JoinColumn(name = **"author\_id"**))  **public** Set<Author> getAuthors() {  **return authors**;  }   **public void** setAuthors(Set<Author> authors) {  **this**.**authors** = authors;  } } | |

# Bidirectional

## One-to-Many

The employee knows who his employer is and the employer knows who his employees are.

|  |  |
| --- | --- |
| **Employer** = parent table | **Employee** = child table |
| Employee.java | |
| @Entity **public class** Employee {  **private** Long **id**;  **private** Employer **employer**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **public** Long getId() {  **return id**;  }   **public void** setId(Long id) {  **this**.**id** = id;  }   @ManyToOne(cascade = CascadeType.***ALL***)  **public** Employer getEmployer() {  **return employer**;  }   **public void** setEmployer(Employer employer) {  **this**.**employer** = employer;  } } | |
| Employer.java | |
| @Entity **public class** Employer {  **private** Long **id**;  **private** Set<Employee> **employees**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **public** Long getId() {  **return id**;  }   **public void** setId(Long id) {  **this**.**id** = id;  }   @OneToMany(cascade = CascadeType.***ALL***, mappedBy = **"employer"**)  **public** Set<Employee> getEmployees() {  **return employees**;  }   **public void** setEmployees(Set<Employee> employees) {  **this**.**employees** = employees;  } } | |

## One-to-One

An employee lives on exactly one address and the address has reference to the employee that lives there.

|  |  |
| --- | --- |
| **Employee** = parent table | **Address** = child table |
| Address.java | |
| @Entity **public class** Address {  **private** Long **employeeId**;  **private** Employee **employee**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **public** Long getEmployeeId() {  **return employeeId**;  }   **public void** setEmployeeId(Long employeeId) {  **this**.**employeeId** = employeeId;  }   @OneToOne(cascade = CascadeType.***ALL***)  @JoinColumn(name = **"employee\_id"**)  **public** Employee getEmployee() {  **return employee**;  }   **public void** setEmployee(Employee employee) {  **this**.**employee** = employee;  } } | |
| Employee.java | |
| @Entity **public class** Employee {  **private** Long **id**;  **private** Address **address**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  @Column(name = **"employee\_id"**)  **public** Long getId() {  **return id**;  }   **public void** setId(Long id) {  **this**.**id** = id;  }   @OneToOne(cascade = CascadeType.***ALL***, mappedBy = **"employee"**)  **public** Address getAddress() {  **return address**;  }   **public void** setAddress(Address address) {  **this**.**address** = address;  } } | |

## Many-to-Many

A book can be written by many authors and many authors can write single book.

|  |  |
| --- | --- |
| **Author** = parent table | **Book** = child table |
| Book.java | |
| @Entity **public class** Book {  **private** Long **bookId**;  **private** Set<Author> **authors**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  @Column(name = **"book\_id"**)  **public** Long getBookId() {  **return bookId**;  }   **public void** setBookId(Long bookId) {  **this**.**bookId** = bookId;  }   @ManyToMany(cascade = CascadeType.***ALL***)  @JoinTable(  name = **"author\_book"**,   joinColumns = @JoinColumn(name = **"book\_id"**),   inverseJoinColumns = @JoinColumn(name = **"author\_id"**))  **public** Set<Author> getAuthors() {  **return authors**;  }   **public void** setAuthors(Set<Author> authors) {  **this**.**authors** = authors;  } } | |
| Author.java | |
| @Entity **public class** Author {  **private** Long **authorId**;  **private** Set<Book> **books**;   @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  @Column(name = **"author\_id"**)  **public** Long getAuthorId() {  **return authorId**;  }   **public void** setAuthorId(Long authorId) {  **this**.**authorId** = authorId;  }   @ManyToMany(cascade = CascadeType.***ALL***, mappedBy = **"authors"**)  **public** Set<Book> getBooks() {  **return books**;  }   **public void** setBooks(Set<Book> books) {  **this**.**books** = books;  } } | |