Exercise 2: Creating Entities

Understanding the Requirements

We need to create two JPA entities: `Employee` and `Department`. There will be a one-to-many relationship between `Department` and `Employee`, meaning one department can have many employees, but an employee belongs to only one department.

Creating the Entities

Java

```java
import javax.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;

    @ManyToOne
    private Department department;

    // Constructors, getters, and setters
}

@Entity
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

    @OneToMany(mappedBy = "department")
    private   Set<Employee> employees   = new HashSet<>();
```

```
    // Constructors, getters, and setters
}
```

Explanation of the Code

- **Annotations:**
    - `@Entity`: Marks a class as an entity, indicating that it should be mapped to a database table.
    - `@Id`: Specifies the primary key column.
    - `@GeneratedValue(strategy = GenerationType.IDENTITY)`: Automatically generates the primary key value.
    - `@ManyToOne`: Specifies a many-to-one relationship, indicating that an Employee belongs to one Department.
    - `@OneToMany`: Specifies a one-to-many relationship, indicating that a Department can have many Employees.
    - `mappedBy`: Specifies the owning side of the relationship in the mapped entity.
- **Relationships:**
    - The `department` field in the `Employee` entity represents the many side of the one-to-many relationship.
    - The `employees` field in the `Department` entity represents the one side of the one-to-many relationship, and the `mappedBy` attribute indicates that the relationship is managed by the `Employee` entity.

Key Points:

- The `Department` class holds a `Set` of `Employee` objects to represent the one-to-many relationship.
- The `Employee` class has a `Department` object to represent the many-to-one relationship.
- Using `HashSet` for the `employees` set prevents duplicate employees in a department.