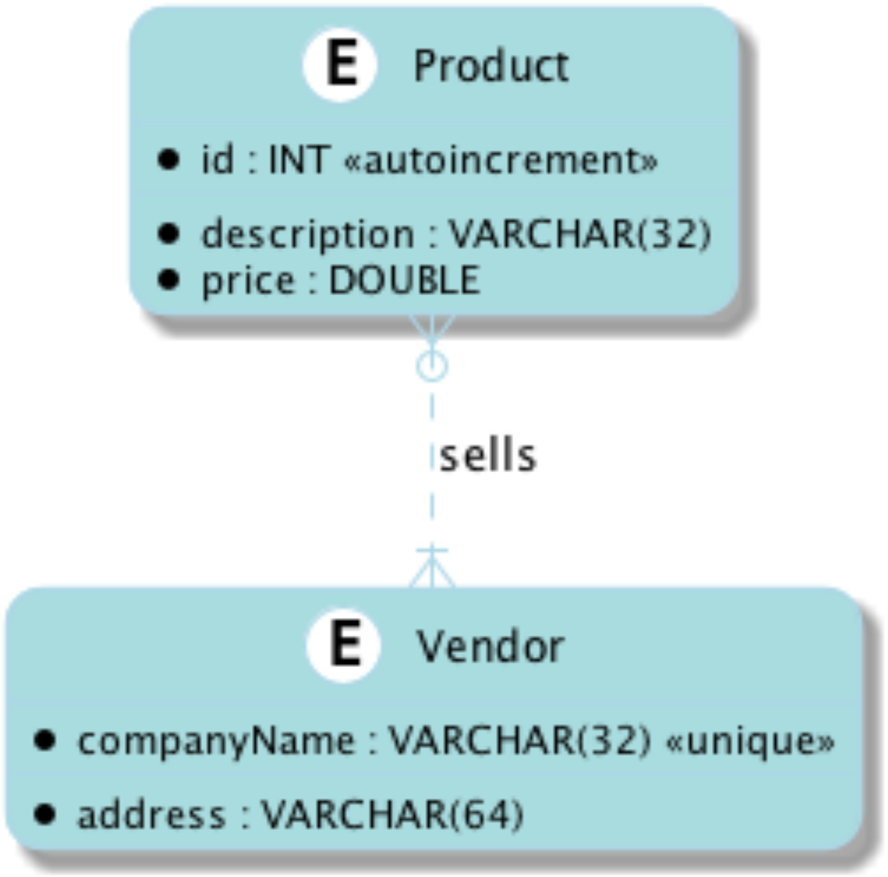
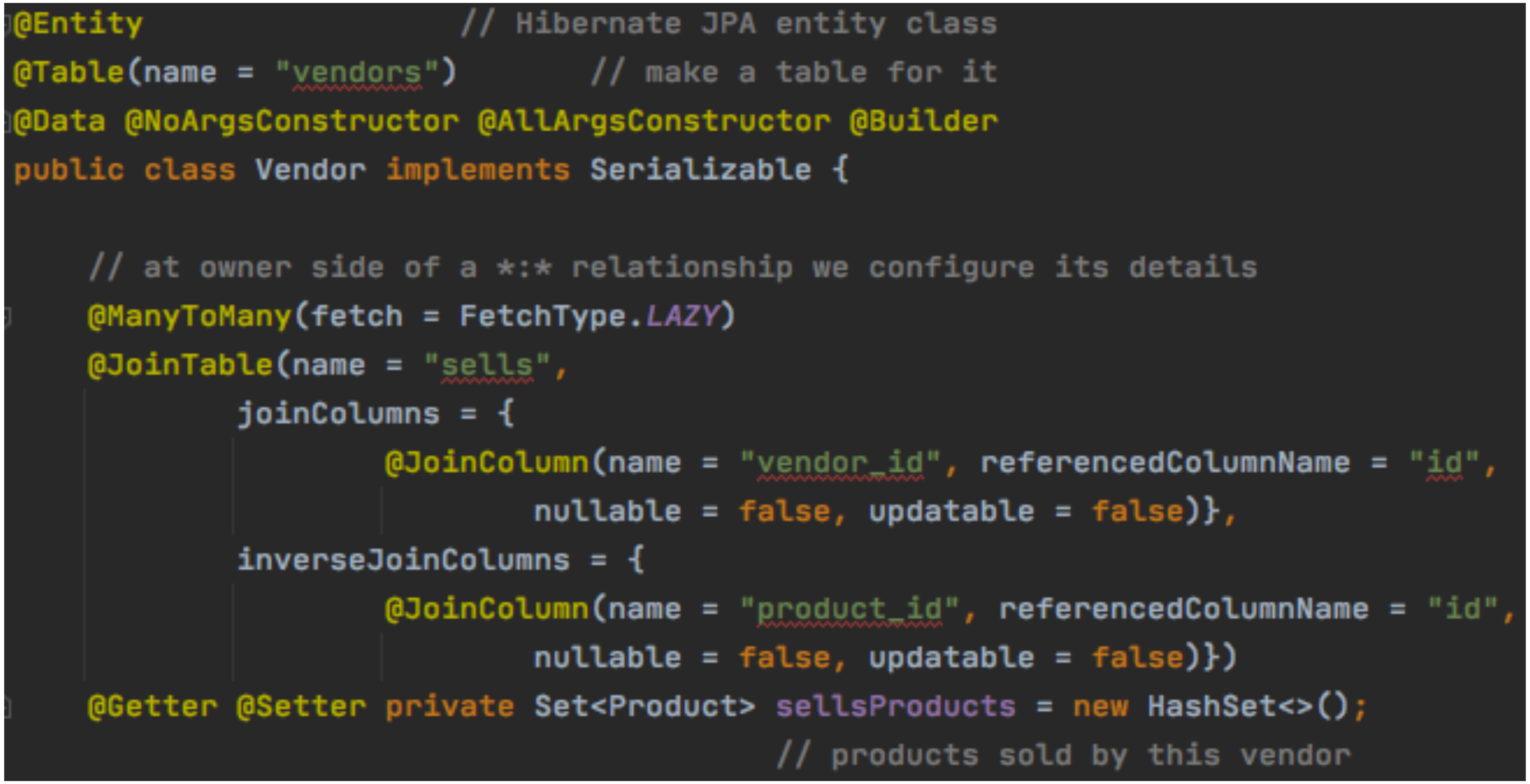
Our goal: many-to-many relationship



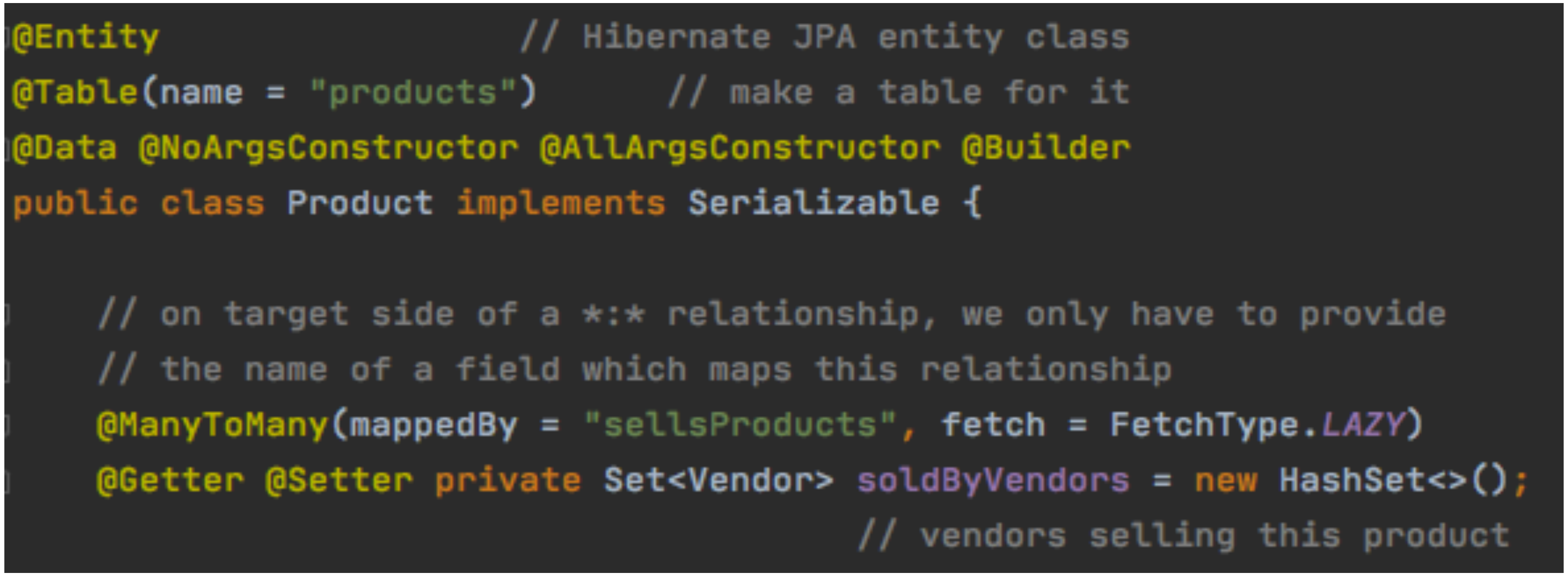
Define ”owner side” details



```
@Entity                    // Hibernate JPA entity class
@Table(name = "vendors")         // make a table for it
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Vendor implements Serializable {

    // at owner side of a *:* relationship we configure its details
    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(name = "sells",
            joinColumns = {
                    @JoinColumn(name = "vendor_id", referencedColumnName = "id",
                            nullable = false, updatable = false)},
            inverseJoinColumns = {
                    @JoinColumn(name = "product_id", referencedColumnName = "id",
                            nullable = false, updatable = false)})
    @Getter @Setter private Set<Product> sellsProducts = new HashSet<>();
                                          // products sold by this vendor
```

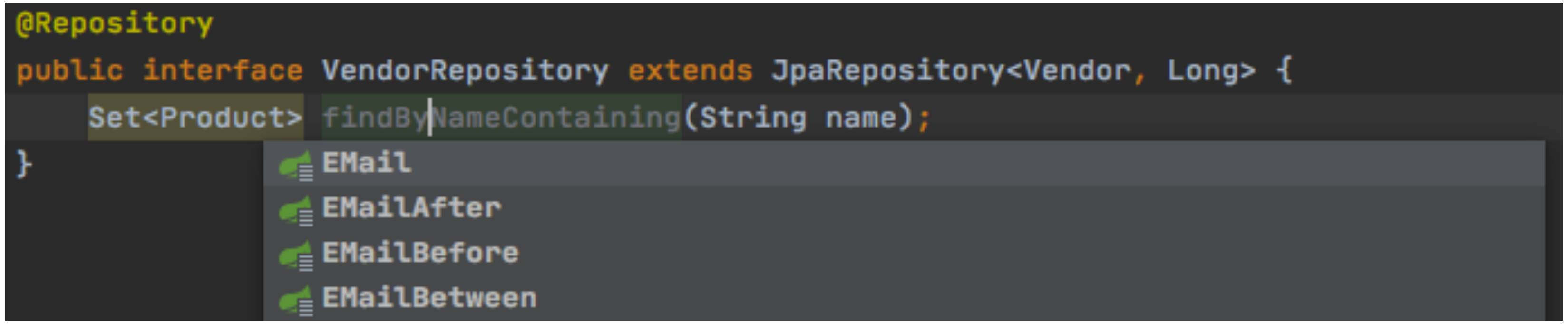
and its target side



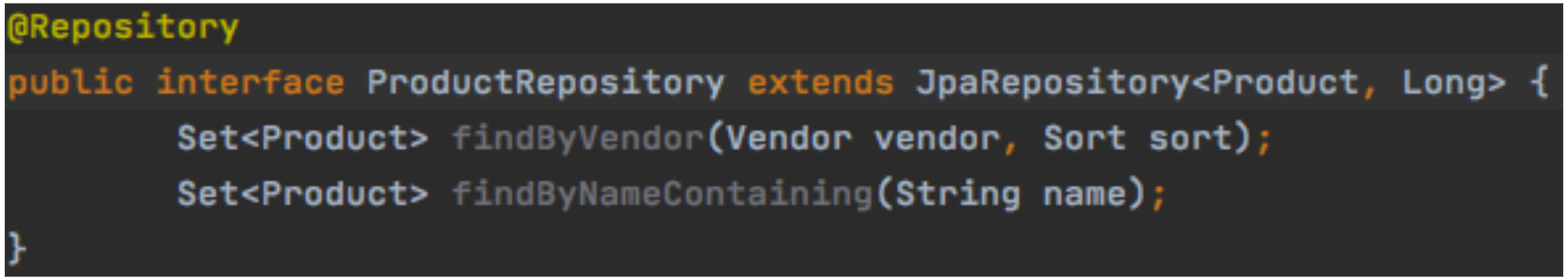
```
@Entity                         // Hibernate JPA entity class
@Table(name = "products")       // make a table for it
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Product implements Serializable {

    // on target side of a *:* relationship, we only have to provide
    // the name of a field which maps this relationship
    @ManyToMany(mappedBy = "sellsProducts", fetch = FetchType.LAZY)
    @Getter @Setter private Set<Vendor> soldByVendors = new HashSet<>();
                                             // vendors selling this product
```

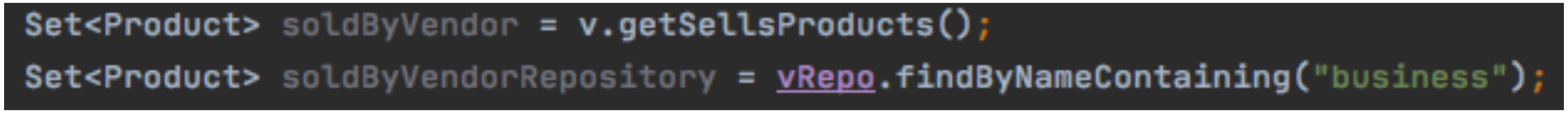
Create/Extend repositories for access



```
@Repository
public interface VendorRepository extends JpaRepository<Vendor, Long> {
    Set<Product> findByNameContaining(String name);
}
```

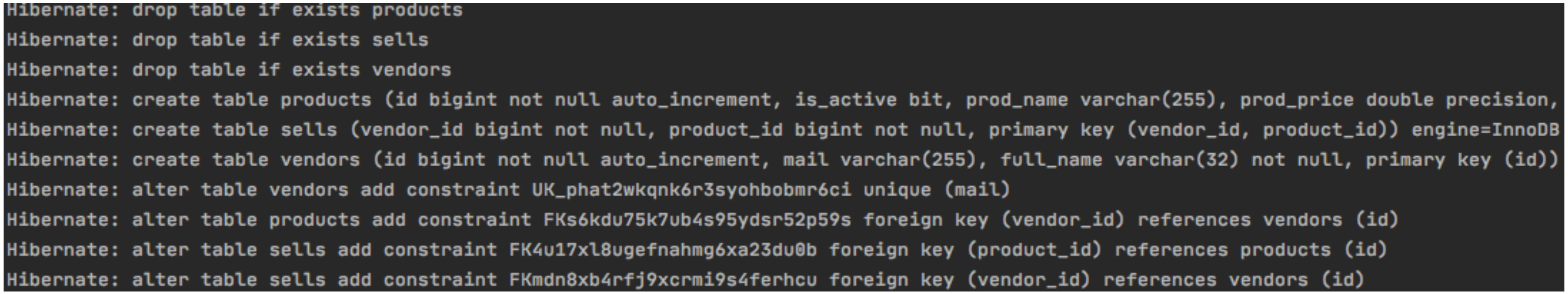


```
@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
        Set<Product> findByVendor(Vendor vendor, Sort sort);
        Set<Product> findByNameContaining(String name);
}
```

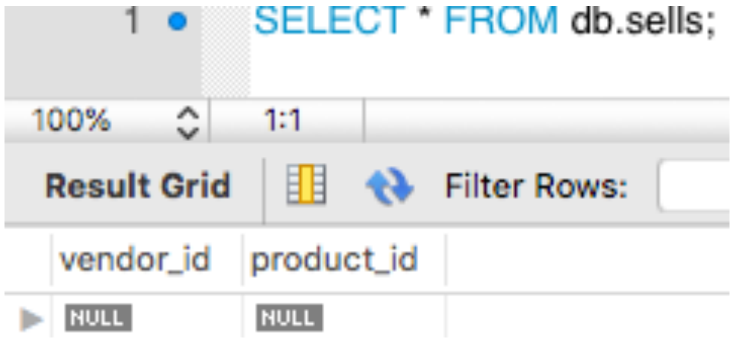


```
Set<Product> soldByVendor = v.getSellsProducts();
Set<Product> soldByVendorRepository = vRepo.findByNameContaining("business");
```

Et voila



```
Hibernate: drop table if exists products
Hibernate: drop table if exists sells
Hibernate: drop table if exists vendors
Hibernate: create table products (id bigint not null auto_increment, is_active bit, prod_name varchar(255), prod_price double precision,
Hibernate: create table sells (vendor_id bigint not null, product_id bigint not null, primary key (vendor_id, product_id)) engine=InnoDB
Hibernate: create table vendors (id bigint not null auto_increment, mail varchar(255), full_name varchar(32) not null, primary key (id))
Hibernate: alter table vendors add constraint UK_phat2wkqnk6r3syohbobmr6ci unique (mail)
Hibernate: alter table products add constraint FKs6kdu75k7ub4s95ydsr52p59s foreign key (vendor_id) references vendors (id)
Hibernate: alter table sells add constraint FK4u17xl8ugefnahmg6xa23du0b foreign key (product_id) references products (id)
Hibernate: alter table sells add constraint FKmdn8xb4rfj9xcrmi9s4ferhcu foreign key (vendor_id) references vendors (id)
```



```
SELECT * FROM db.sells;
```

vendor_id	product_id
NULL	NULL