### ****ProductType**** Class

java

Copy code

@Entity

public class ProductType {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)

@JoinColumn(name = "productType\_id") // Foreign key in Attribute table

private List<AttributeDefinition> attributes = new ArrayList<>();

}

### New ****AttributeDefinition**** Class

This class defines the structure of attributes with their name, type, and allowed values (if applicable).

java

Copy code

@Entity

public class AttributeDefinition {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name; // Attribute name (e.g., "Size", "Color", "Material")

@Enumerated(EnumType.STRING)

private AttributeType type; // Type of the attribute (e.g., TEXT, NUMBER, ENUM)

@ElementCollection

private List<String> allowedValues; // Applicable only if type is ENUM

}

### Enum ****AttributeType****

Defines the possible types for an attribute.

java

Copy code

public enum AttributeType {

TEXT, // For plain text values

NUMBER, // For numeric values

ENUM, // For enumerated values with constraints

DATE // For date values

}

**Product Class**

java

Copy code

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String description;

private BigDecimal price;

@ManyToOne

@JoinColumn(name = "productType\_id", nullable = false)

private ProductType productType; // Links product to its type

@ManyToMany

@JoinTable(

name = "product\_category",

joinColumns = @JoinColumn(name = "product\_id"),

inverseJoinColumns = @JoinColumn(name = "category\_id")

)

private List<Category> categories = new ArrayList<>();

@OneToMany(mappedBy = "product", cascade = CascadeType.ALL, orphanRemoval = true)

private List<ProductVariant> variants = new ArrayList<>();

// Add a method to create a variant based on ProductType

public ProductVariant createVariant(String sku, Map<String, String> variantAttributes, Integer stock, BigDecimal additionalPrice) {

// Validate variant attributes against product type

validateVariantAttributes(variantAttributes);

ProductVariant variant = new ProductVariant();

variant.setSku(sku);

variant.setVariantAttributes(variantAttributes);

variant.setStock(stock);

variant.setAdditionalPrice(additionalPrice);

variant.setProduct(this);

this.variants.add(variant);

return variant;

}

private void validateVariantAttributes(Map<String, String> variantAttributes) {

for (AttributeDefinition attribute : productType.getAttributes()) {

String attributeName = attribute.getName();

String attributeValue = variantAttributes.get(attributeName);

// Ensure required attributes are provided

if (attributeValue == null) {

throw new IllegalArgumentException("Missing value for required attribute: " + attributeName);

}

// If ENUM, validate allowed values

if (attribute.getType() == AttributeType.ENUM) {

if (!attribute.getAllowedValues().contains(attributeValue)) {

throw new IllegalArgumentException("Invalid value for attribute " + attributeName + ": " + attributeValue);

}

}

}

}

}

**ProductVariant Class**

java

Copy code

@Entity

public class ProductVariant {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String sku;

@ElementCollection

private Map<String, String> variantAttributes; // E.g., {"Size": "M", "Color": "Red"}

private Integer stock;

private BigDecimal additionalPrice;

@ManyToOne

@JoinColumn(name = "product\_id", nullable = false)

private Product product;

}

**Example Usage: Creating a Product and Variants Dynamically**

java

Copy code

// Create ProductType

ProductType shirtType = new ProductType();

shirtType.setName("Shirt");

shirtType.setAttributes(List.of(

new AttributeDefinition("Size", AttributeType.ENUM, List.of("S", "M", "L", "XL")),

new AttributeDefinition("Color", AttributeType.ENUM, List.of("Red", "Blue", "Green"))

));

// Create Product

Product shirt = new Product();

shirt.setName("Slim Fit Shirt");

shirt.setDescription("A stylish slim-fit casual shirt.");

shirt.setPrice(new BigDecimal("25.00"));

shirt.setProductType(shirtType);

// Create Variants based on ProductType

ProductVariant redMedium = shirt.createVariant(

"SHIRT-M-RED",

Map.of("Size", "M", "Color", "Red"),

50,

BigDecimal.ZERO

);

ProductVariant blueLarge = shirt.createVariant(

"SHIRT-L-BLUE",

Map.of("Size", "L", "Color", "Blue"),

30,

new BigDecimal("2.00")

);