

com.example.demo
com.example.demo::DemoApplication
+main(String[] args): void
+home(): String

com.example.demo.categories	com.example.demo.categories::Category	com.example.demo.categories::CategoryService
	<pre> -id: Long -title: String -description: String  +Category(): ctor +Category(Long id, String title, @Nullable String description): ctor +getId(): Long +setId(Long id): void +getTitle(): String +setTitle(String title): void +getDescription(): String +setDescription(@Nullable String description): void </pre>	<pre> -categoryRepository: CategoryRepository  +CategoryService(CategoryRepository categoryRepository): ctor +checkIfExists(Long id): boolean +getCategoryById(Long id): Category </pre>
	«interface» com.example.demo.categories::CategoryRepository	

```

classDiagram
    class com.example.demo.clients {
        -id: Long
        -name: String
        -last_name: String
        -mobile: String
        -creation_date: LocalDateTime
        +Client(): ctor
        +Client(String name, String last_name, String mobile): ctor
        +setCreation_date(LocalDateTime date): void
        +setId(Long id): void
        +setName(String name): void
        +setLast_name(String last_name): void
        +setMobile(String mobile): void
        +getId(): Long
        +getName(): String
        +getLast_name(): String
        +getMobile(): String
        +toString(): String
    }
    class com.example.demo.clients {
        <<interface>>
        +ClientRepository
    }
    class com.example.demo.clients {
        -clientRepository: ClientRepository
        +ClientService(ClientRepository clientRepository): ctor
        +getClient(int pageNo, int pageSize): List<Client>
        +addClient(Client c): Client
        +updateClient(Client c, Long id): Client
        +checkIfExists(Long id): boolean
    }
    class com.example.demo.clients {
        -clientService: ClientService
        +ClientController(ClientService clientService): ctor
        +getClient(@PathVariable int pageNo, @PathVariable int pageSize): List<Client>
        +addClient(@Valid @RequestBody Client c): Client
        +editClient(@Valid @RequestBody Client c, @PathVariable("id") Long id): Client
    }
    com.example.demo.clients <|-- com.example.demo.clients
    com.example.demo.clients <|-- com.example.demo.clients
    com.example.demo.clients <|-- com.example.demo.clients
    com.example.demo.clients <|-- com.example.demo.clients
  
```

com.example.demo.core	
com.example.demo.core::CustomExceptionHandler	
+handleException(BindException ex): Map<String, String>	

```
classDiagram
    class Product {
        -id: Long
        -name: String
        -description: String
        -category: Category
        -creation_date: LocalDateTime
        +Product(): ctor
        +Product(String name, String description, Category category): ctor
        +setId(Long id): void
        +setName(String name): void
        +setDescription(String description): void
        +setCategory(Category category): void
        +setCreation_date(): void
        +getId(): Long
        +getName(): String
        +getDescription(): String
        +getCategory(): Category
        +getCreation_date(): LocalDateTime
        +toString(): String
    }
    class ProductRepository {
        <<interface>>
    }
    class ProductController {
        -productService: ProductService
        -categoryService: CategoryService
        +ProductController(ProductService productService, CategoryService categoryService): ctor
        +getProducts(@PathVariable int pageNo, @PathVariable int pageSize): List<Product>
        +addProduct(@Valid @RequestBody Product p): Product
        +editProduct(@Valid @RequestBody Product p, @PathVariable("id") Long id): Product
    }
    class ProductService {
        -productRepository: ProductRepository
        +ProductService(ProductRepository productRepository): ctor
        +getProducts(int pageNo, int pageSize): List<Product>
        +addNewProduct(Product p): Product
        +updateProduct(Product p, Category cat, Long id): Product
        +checkIfExists(Long id): boolean
    }
    ProductRepository <|-- ProductController
    ProductRepository <|-- ProductService
```

The diagram illustrates the structure of a demo application. It features four classes and one interface. The **Product** class is a domain entity with attributes `-id: Long`, `-name: String`, `-description: String`, `-category: Category`, and `-creation_date: LocalDateTime`. It includes constructors for creating a new product and a product with specific details, as well as methods for setting and getting these attributes. The **ProductRepository** is an interface defining methods for product management. The **ProductController** and **ProductService** classes implement this interface. **ProductController** uses **ProductService** and **CategoryService** to handle HTTP requests. **ProductService** uses **ProductRepository** to perform database operations. The diagram uses solid lines for class boundaries, dashed lines for associations, and a double line for the interface boundary.

com.example.demo.products::Product	com.example.demo.products::ProductController	com.example.demo.products::ProductService
<code>-id: Long</code> <code>-name: String</code> <code>-description: String</code> <code>-category: Category</code> <code>-creation_date: LocalDateTime</code> <code>+Product(): ctor</code> <code>+Product(String name, String description, Category category): ctor</code> <code>+setId(Long id): void</code> <code>+setName(String name): void</code> <code>+setDescription(String description): void</code> <code>+setCategory(Category category): void</code> <code>+setCreation_date(): void</code> <code>+getId(): Long</code> <code>+getName(): String</code> <code>+getDescription(): String</code> <code>+getCategory(): Category</code> <code>+getCreation_date(): LocalDateTime</code> <code>+toString(): String</code>	<code>-productService: ProductService</code> <code>-categoryService: CategoryService</code> <code>+ProductController(ProductService productService, CategoryService categoryService): ctor</code> <code>+getProducts(@PathVariable int pageNo, @PathVariable int pageSize): List&lt;Product&gt;</code> <code>+addProduct(@Valid @RequestBody Product p): Product</code> <code>+editProduct(@Valid @RequestBody Product p, @PathVariable("id") Long id): Product</code>	<code>-productRepository: ProductRepository</code> <code>+ProductService(ProductRepository productRepository): ctor</code> <code>+getProducts(int pageNo, int pageSize): List&lt;Product&gt;</code> <code>+addNewProduct(Product p): Product</code> <code>+updateProduct(Product p, Category cat, Long id): Product</code> <code>+checkIfExists(Long id): boolean</code>

«interface»
com.example.demo.products::ProductRepository

```
classDiagram
    class com_example_demo_sales {
        <<interface>>
    }
    class com_example_demo_sales_Sale {
        -id: Long
        -total: double
        -client: Client
        -seller: Seller
        -creation_date: LocalDateTime
        ~sale_transactions: List<Transaction>
        +Sale(): ctor
        +Sale(Client client, Seller seller): ctor
        +getId(): Long
        +setId(Long id): void
        +getSale_transactions(): List<Transaction>
        +getTotal(): double
        +setTotal(): void
        +getClient(): Client
        +setClient(Client client): void
        +getSeller(): Seller
        +setSeller(Seller seller): void
        +getCreation_date(): LocalDateTime
        +setCreation_date(): void
        +setSale_transactions(List<Transaction> sale_transactions): void
    }
    class com_example_demo_sales_SaleController {
        -saleService: SaleService
        -productService: ProductService
        -sellerService: SellerService
        -clientService: ClientService
        -transactionService: TransactionService
        +SaleController(SaleService saleService, ProductService productService, SellerService sellerService, ClientService clientService, TransactionService transactionService): ctor
        +getSales(@PathVariable int pageNo, @PathVariable int pageSize): List<Sale>
        +AddSale(@Valid @RequestBody Sale s): Sale
        +editSaleTransactions(@Valid @RequestBody List<Transaction> transactions, @PathVariable("id") Long id): Sale
    }
    class com_example_demo_sales_SaleService {
        -saleRepository: SaleRepository
        +SaleService(SaleRepository saleRepository): ctor
        +checkIfExists(Long id): boolean
        +getAllSales(int pageNo, int pageSize): List<Sale>
        +getSaleById(Long id): Sale
        +addOrUpdateSale(Sale s): Sale
    }
    com_example_demo_sales <|-- com_example_demo_sales_Sale
    com_example_demo_sales <|-- com_example_demo_sales_SaleController
    com_example_demo_sales <|-- com_example_demo_sales_SaleService
```

The diagram illustrates the structure of the `com.example.demo.sales` package. It features four main components: an interface, three concrete classes, and their relationships.

- com.example.demo.sales** (Interface):
  - Attributes: `-id: Long`, `-total: double`, `-client: Client`, `-seller: Seller`, `-creation_date: LocalDateTime`, `~sale_transactions: List<Transaction>`
  - Methods: `+Sale(): ctor`, `+Sale(Client client, Seller seller): ctor`, `+getId(): Long`, `+setId(Long id): void`, `+getSale_transactions(): List<Transaction>`, `+getTotal(): double`, `+setTotal(): void`, `+getClient(): Client`, `+setClient(Client client): void`, `+getSeller(): Seller`, `+setSeller(Seller seller): void`, `+getCreation_date(): LocalDateTime`, `+setCreation_date(): void`, `+setSale_transactions(List<Transaction> sale_transactions): void`
- com.example.demo.sales::Sale** (Class):
  - Attributes: `-saleService: SaleService`, `-productService: ProductService`, `-sellerService: SellerService`, `-clientService: ClientService`, `-transactionService: TransactionService`
  - Methods: `+SaleController(SaleService saleService, ProductService productService, SellerService sellerService, ClientService clientService, TransactionService transactionService): ctor`, `+getSales(@PathVariable int pageNo, @PathVariable int pageSize): List<Sale>`, `+AddSale(@Valid @RequestBody Sale s): Sale`, `+editSaleTransactions(@Valid @RequestBody List<Transaction> transactions, @PathVariable("id") Long id): Sale`
- com.example.demo.sales::SaleService** (Class):
  - Attributes: `-saleRepository: SaleRepository`
  - Methods: `+SaleService(SaleRepository saleRepository): ctor`, `+checkIfExists(Long id): boolean`, `+getAllSales(int pageNo, int pageSize): List<Sale>`, `+getSaleById(Long id): Sale`, `+addOrUpdateSale(Sale s): Sale`

Relationships are indicated by solid lines with hollow triangle heads pointing to the superclass or interface:

- `com.example.demo.sales::Sale` inherits from `com.example.demo.sales`.
- `com.example.demo.sales::SaleController` inherits from `com.example.demo.sales`.
- `com.example.demo.sales::SaleService` inherits from `com.example.demo.sales`.

com.example.demo.sales.transactions:Transaction		com.example.demo.sales.transactions:TransactionKey	
~id: TransactionKey ~sale: Sale ~product: Product ~quantity: int ~unit_price: double +Transaction(): ctor +Transaction(TransactionKey id, Sale sale, Product product, int quantity, double unit_price): ctor +setId(TransactionKey id): void +getSale(): Sale +setSale(Sale sale): void +getProduct(): Product +setProduct(Product product): void +getQuantity(): int +setQuantity(int quantity): void +getUnit_price(): double +setUnit_price(double unit_price): void +equals(Object obj): boolean +toString(): String		~productId: Long ~saleId: Long +TransactionKey(): ctor +TransactionKey(Long productId, Long saleId): ctor +getProductId(): Long +setProductId(Long productId): void +getSaleId(): Long +setSaleId(Long saleId): void +equals(Object obj): boolean +hashCode(): int	
com.example.demo.sales.transactions:TransactionService		«interface» com.example.demo.sales.transactions:TransactionRepository	
~logger = LoggerFactory.getLogger(TransactionService.class): Logger ~transactionRepository: TransactionRepository +TransactionService(TransactionRepository transactionRepository): ctor +addMultipleTransactions(List<Transaction> transactions, Sale sale): List<Transaction> +getTransactionBySale(Long sale_id): List<Transaction> +updateTransactions(List<Transaction> old_trans, List<Transaction> new_trans): void		~findBySaleId(Long sale_id): List<Transaction>	

```

classDiagram
    class Seller {
        -id: Long
        -name: String
        -Agency: String
        -details: String
        +Seller(): ctor
        +Seller(Long id, String name, @Nullable String agency, String details): ctor
        +getId(): Long
        +setId(Long id): void
        +getName(): String
        +setName(String name): void
        +getAgency(): String
        +setAgency(@Nullable String agency): void
        +getDetails(): String
        +setDetails(String details): void
    }
    class SellerRepository {
        <<interface>>
        +SellerRepository()
    }
    class SellerService {
        -sellerRepository: SellerRepository
        +SellerService(SellerRepository sellerRepository): ctor
        +checkIfExists(Long id): boolean
        +getSellerById(Long id): Category
    }
    SellerRepository <|-- Seller
    SellerService --> SellerRepository
  
```

The diagram illustrates the relationships between three classes: `Seller`, `SellerRepository`, and `SellerService`.

- Seller** (Class):
  - Attributes: `-id: Long`, `-name: String`, `-Agency: String`, `-details: String`.
  - Operations: `+Seller(): ctor`, `+Seller(Long id, String name, @Nullable String agency, String details): ctor`, `+getId(): Long`, `+setId(Long id): void`, `+getName(): String`, `+setName(String name): void`, `+getAgency(): String`, `+setAgency(@Nullable String agency): void`, `+getDetails(): String`, `+setDetails(String details): void`.
- SellerRepository** (Interface):
  - Operation: `+SellerRepository()`.
- SellerService** (Class):
  - Attribute: `-sellerRepository: SellerRepository`.
  - Operations: `+SellerService(SellerRepository sellerRepository): ctor`, `+checkIfExists(Long id): boolean`, `+getSellerById(Long id): Category`.

Relationships:

- `SellerRepository` is a generalization of `Seller` (indicated by a solid line with an open arrowhead pointing to `SellerRepository`).
- `SellerService` has a directed association with `SellerRepository` (indicated by a solid line with an open arrowhead pointing to `SellerRepository`).