Università degli Studi di Salerno Corso di Ingegneria del Software

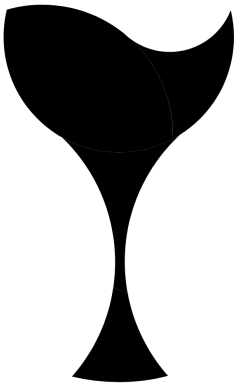
Classe 1 Resto 0

Corso di Laurea in Informatica A.A. 2022/23

diVino

Object Design Document

Versione 1.2



Gruppo di lavoro

Nome	Matricola
Cataldo Gaetano	0512111910
La Torraca Carmine	0512112936

Storico revisioni

Data	Versione	Descrizione	Autori
09/01/2023	1.0	Prima stesura Object Design Document	C. La Torraca, G. Cataldo
12/01/2023	1.1	Completamento documentazione	C. La Torraca, G. Cataldo
13/01/2023	1.2	Controllo ortografico e chiusura documentazione	C. La Torraca, G. Cataldo

Sommario

Introduzione

Lo scopo finale del documento è quello di fornire un modello applicabile per implementare tutte le funzionalità individuate nei documenti di "System Design" (SDD) e "Requirements Analysis" (RAD). In particolare, vogliamo fornire le interfacce e le classi comprese di parametri e firme delle varie funzioni dei sottosistemi individuati nel SDD.

Obiettivi di design trade-offs

Design Pattern

- **FACADE** per permettere un facile accesso alle funzionalità e sottosistemi che riguardano l'ordine (creazione ordine, l'associazione dei prodotti del carrello a quelli presenti nell'ordine e il pagamento) viene utilizzato il Facade pattern attraverso una classe (appunto facade) la quale incapsula il sistema complesso fornendo poche e semplici interfacce.
- **SINGLETON** il pattern viene utilizzato per la gestione della connessione al database con la classe Datasource instanziata all'avvio del sistema. Inoltre, il pattern è utilizzato per gestire l'entità catalogo la quale, anch'essa deve essere accessibile globalmente al sistema ed instanziata una e una sola volta.

Interface documentation guidelines

Variabili:

- I nomi delle variabili devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Quest'ultime devono essere dichiarate ad inizio blocco, solamente una per riga e devono essere tutte allineate e facilitarne la leggibilità.
- È inoltre possibile, in alcuni casi, utilizzare il carattere underscore ("_") per la definizione del nome.

Metodi:

- I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Il nome del metodo tipicamente consiste in un verbo che identifica una azione, seguito dal nome di un oggetto. I nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile().

Classi e pagine:

- I nomi delle classi e delle pagine devono cominciare con una lettera maiuscola, e anche le parole seguenti all'interno del nome devono cominciare con una lettera maiuscola. I nomi di quest'ultime devono fornire informazioni sul loro scopo.

- La dichiarazione di classe deve essere caratterizzata da:
    1. Dichiarazione della classe pubblica
    2. Dichiarazioni di costanti
    3. Dichiarazioni di variabili di classe
    4. Dichiarazione di variabili d'istanza
    5. Costruttore

Commento e dichiarazione dei metodi

Definizioni, acronimi ed abbreviazioni

- **SDD**: System Design Document
- **RAD:** Requirements Analysis Document
- **ODD:** Object Design Document

Riferimenti

Nel corso del documento potrebbero esserci riferimenti alla documentazione precedente RAD e SDD.

Package

Il nostro sistema presenta una suddivisione basata su tre livelli (three-tier):

- Interface layer
- Application Logic layer
- Storage layer

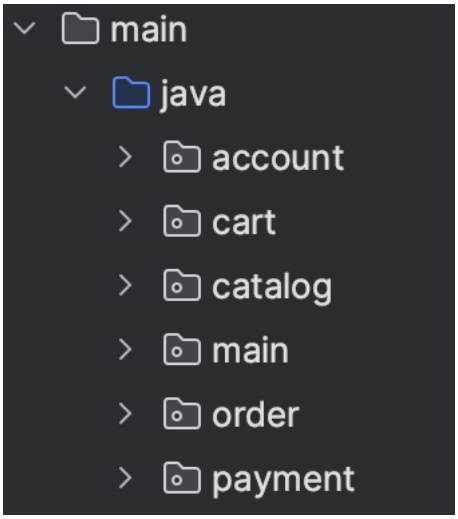
Interface layer	Interfaccia del sistema che offre la possibilità all'utente di interagire con esso.

Application Logic layer	Ha il compito di elaborare i dati e le richieste del client: • Account management • Product management • Order management • Cart management • Catalog management
Storage layer	Si occupa di memorizzare i dati utilizzando un DBMS. Riceve le richieste dall'Application Logic layer, le inoltra al DBMS e poi restituisce la risposta.

Secondo la scomposizione del sistema in sottosistemi, effettuata e descritta durante la fase di system design (vedere System Design Document), sono stati individuati i seguenti package:

- **Order package**: il package fornisce le funzionalità riguardo gli ordini, dalla loro creazione alla gestione. Fanno parte del package le classi Controller, Entity e i Model.
- **Catalog package**: il package fornisce le funzionalità riguardo il catalogo, come la visualizzazione dei prodotti nella homepage. Fanno parte del package le classi Controller, Entity e i Model.
- **Account package**: il package fornisce le funzionalità riguardo gli account, l'autenticazione al sistema e la registrazione. Fanno parte del package le classi Controller, Entity e i Model.
- **Cart package**: il package fornisce le funzionalità che riguardano la gestione del carrello e dei prodotti in esso contenuti, tra cui la modifica della quantità, la rimozione e l'aggiunta, nonché il processo che permette di essere reindirizzati all'acquisto. Fanno parte del package le classi Controller, Entity e i Model.

- **Payment package**: il package fornisce le funzionalità riguardo i pagamenti relativi agli ordini, dal pagamento alla viualizzazione dello storico. Fanno parte del package le classi Controller, Entity e i Model.



Class interfaces

Nome Classe	**CatalogEntity**
Modulo	catalog
Variabili di Istanza	-HashSet<ProductEntity> catalogProducts
Descrizione	
Firme dei Metodi	+ CatalogEntity() + addProdcut(product: ProductEntity) : void + removeProduct (product: ProductEntity) : void + getCatalogProducts() : HashSest<ProductEnity() +setCatalogProducts(catalogProducts : Hashset<ProductEntity>) : void
Pre e Post condizioni	• **Context** : CatalogEntity::CatalogEntity() **Pre**: true **Post**: catalogProducts = null; • **Context**: CatalogEntity::addProduct(product: ProductEntity) **Pre**: true **Post**: self.catalogProducts = product • **Context**: CatalogEntity::removeProduct(product: ProductEntity) **Pre**: true **Post**: self.catalogProducts = product • **Context**: CatalogEntity::getCatalogProducts() **Pre**: true **Post**: result = self.catalogProducts • **Context**: CatalogEntity::setCatalogProducts((catalogProducts : Hashset<ProductEntity>) **Pre**: true **Post**: self.catalogProducts = catalogProducts
Invarianti	

Nome Classe	**ProductEntity**
Modulo	Catalog
Variabili di Istanza	- productid : Integer - productBrand : String - productDescription : String - productFormat : String - productPrice : double - productAvailability : int - isSales : boolean - salesPrice : double - productVat : int - isVisible : boolean - imagePath : String
Descrizione	
Firme dei Metodi	+ getProductId() : String + setProductId(productid : String) : void + getProductBrand() : String + setProductBrand(productBrand : String) : void + getProducDescription() : String + setProductDescription(productDescription : String) : void + getProductFormat() : String + setProductFormat(productFormat : String) : void + getProductPrice() : double + setProductPrice(productPrice : double) : void + getProductAvailability() : int + setProductAvailability(productAvailability : int) : void

	+ isSales() : boolean + setSales(sales : boolean) : void + getSalesPrice() : double + setSalesPrice(salesPrice : double) : void + getProductVat() : int + setProductVat(productVat : int) : void + isVisible() : boolean + setVisible(visible : boolean) : void + getImagePath() : String + setImagePath (imagePath : String) : void
Pre e Post condizioni	• **Context** : ProductEntity::getProductId () **Pre**: true **Post**: result = self.productId • **Context**: ProductEntity::setProductId(productid : String) **Pre**: true **Post**: self.productId = productid • **Context** : ProductEntity::getProductBrand () **Pre**: true **Post**: result = self. productBrand • **Context**: ProductEntity::setProductBrand(productBrand : String) : void **Pre**: true **Post**: self.productBrand = productBrand • **Context** : ProductEntity::getProducDescription() **Pre**: true **Post**: result = self. productDescription • **Context**: ProductEntity::setProductDescription(productDescription : String) : void **Pre**: true **Post**: self. productDescription = productDescription • **Context** : ProductEntity::getProductFormat() **Pre**: true **Post**: result = self. productFormat

- **Context**: ProductEntity::setProductFormat(productFormat : String)
  Pre: true
  Post: self. productFormat = productFormat

- **Context** : ProductEntity::getProductPrice()
  Pre: true
  Post: result = self. productPrice
- **Context**: ProductEntity::setProductPrice(productPrice : double) : void
  Pre: true
  Post: self. productPrice = productPrice

- **Context** : ProductEntity::getProductId ()
  Pre: true
  Post: result = self.productId
- **Context**: ProductEntity::setProductId(productid : String)
  Pre: true
  Post: self.productId = productid

- **Context** : ProductEntity::getProductAvailability()
  Pre: true
  Post: result = self. productAvailability
- **Context**: ProductEntity::setProductAvailability(productAvailability : int)
  Pre: true
  Post: self.productAvailability = productAvailability

- **Context** : ProductEntity**::**isSales()
  Pre: true
  Post: result = self.sales
- **Context**: ProductEntity::setSales(sales : boolean)
  Pre: true
  Post: self.sales = sales

- **Context** : ProductEntity**::**getSalesPrice()
  Pre: true
  Post: result = self.salesPrice
- **Context**: ProductEntity::setSalesPrice(salesPrice : double)
  Pre: true
  Post: self.salesPrice = salesPrice

- **Context** : ProductEntity**::**getProductVat()
  Pre: true
  Post: result = self. productVat

	• **Context**: ProductEntity::setProductVat(productVat : int) : void **Pre**: true **Post**: self. productVat = productVat • **Context** : ProductEntity::isVisible() **Pre**: true **Post**: result = self.visible • **Context**: ProductEntity::setVisible(visible : boolean) **Pre**: true **Post**: self.visible = visible • **Context** : ProductEntity::getImagePath() **Pre**: true **Post**: result = self.imagePath • **Context**: ProductEntity::setImagePath(imagePath : String) **Pre**: true **Post**: self.imagePath = imagePath
Invarianti	

Nome Classe	CatalogDAO
Modulo	catalog
Variabili di Istanza	- connection : Connection - TABLE_NAME : String
Descrizione	
Firme dei Metodi	+ createCatalog() : HashSet<ProductEntity> + removeProduct(product : ProductEntity) : void + addProduct(product : ProductEntity) : void + updateProduct(product : ProductEntity) : void + updateStock(purchasedQuantity : Integer, productid : String) : void - getAvailableQuantity (productid : String) : Integer
Pre e Post condizioni	• **Context** : catalogDAO::createCatalog () **Pre**: **Post**: result = catalog • **Context**: removeProduct(product : ProductEntity): void **Pre**: catalog.size()>0

	Post:
Invarianti	

Nome Classe	**OrderEntity**
Modulo	order
Variabili di Istanza	-orderNumber: int -orderStatus: String -orderTotalAmount: double -orderShippingAddress : String -orderCustomer: int -orderPayment: int -orderProducts: HashSet<OrderItemEntity>
Descrizione	Entità contenente le informazioni dell'ordine.
Firme dei Metodi	+ getOrderNumber(): int + getOrderStatus(): String + getOrderTotalAmount(): double + getOrderCustomer(): int + getOrderPayment(): int + getOrderProducts(): HashSet<OrderItemEntity> + setOrderNumber(orderNumber: int): void + setOrderStatus(orderStatus: String): void + setOrderTotalAmount(orderTotalAmount: double): void + setOrderCustomer(orderCustomer: int): void + setOrderPayment(orderPayment: int): void + setOrderProducts(orderProducts: HashSet<OrderItemEntity>): void
Pre e Post condizioni	• **context** OrderEntity::getOrderNumber() **pre:** **post:** result = self.orderNumber • **context** OrderEntity::setOrderNumber(orderNumber: String) **pre:** true **post:** self.orderNumber = orderNumber

	• **context** OrderEntity::getOrderStatus() **pre:** **post:** result = self.orderStatus • **context** OrderEntity::setOrderStatus(orderStatus: String) **pre:** true **post:** self.orderStatus = orderStatus • **context** OrderEntity::getOrderTotalAmount() **pre: if** orderProducts.size() > 0 **then** result = orderTotalAmount else result = 0 **endif** **post:** result = self.orderTotaleAmount • **context** OrderEntity::setOrderTotalAmount (orderTotalAmount: double) **pre:** true **post:** self.orderTotalAmount = orderTotalAmount • **context** OrderEntity::getOrderCustomer() **pre:** **post:** result = self.orderCustomer • **context** OrderEntity:: setOrderCustomer (orderCustomer: int) **pre:** true **post:** self. orderCustomer = orderCustomer • **context** OrderEntity::getOrderPayment() **pre:** **post:** result = self.orderPayment • **context** OrderEntity:: setOrderPayment (orderPayment: int) **pre:** true **post:** self. orderPayment = orderPayment
Invarianti	

Nome Classe	**OrderItemEntity**
Modulo	order
Variabili di Istanza	-order : int -productVat: int -productPrice: double -productQuantity: int -productDescription: String -productID : int

Descrizione	Entità che contiene le informazioni riguardo i prodotti e le quantità nell'ordine
Firme dei Metodi	+ getOrderID(): int + getProductVat(): int + getProductPrice(): double + getProductQuantity(): int + getProductDescription(): String + setOrderID(orderID : int): void + setProductVat(productVat: int): void + setProductPrice(productPrice: double): void + setProductDescription(productDescription: String): void
Pre e Post condizioni	• **Context** : OrderItemEntity:: getOrderID () **Pre**: true **Post**: result = self. orderID • **Context**: OrderItemEntity:: setOrderID (orderID : int) **Pre**: true **Post**: self. orderID = orderID • **Context** : OrderItemEntity::getProductVat () **Pre**: true **Post**: result = self. productVat • **Context**: OrderItemEntity::setProductVat(productVat: int) **Pre**: true **Post**: self. productVat = productVat • **Context** : OrderItemEntity::getProductPrice () **Pre**: true **Post**: result = self. productPrice • **Context**: OrderItemEntity::setProductPrice(productPrice: double) **Pre**: true **Post**: self. productPrice = productPrice • **Context** : OrderItemEntity::getProductQuantity() () **Pre**: true **Post**: result = self. productQuantity • **Context** : OrderItemEntity::getProductDescription () **Pre**: true

	Post: result = self. productDescription • **Context**: OrderItemEntity::setProductDescription(productDescription: String) **Pre**: true **Post**: self. productDescription = productDescription
Invarianti	

Nome Classe	**OrderDAO**
Modulo	order
Variabili di Istanza	-connection: Connection
Descrizione	
Firme dei Metodi	+ createOrder(customer: CustomerUserEntity): OrderEntity + saveOrder(order OrderEntity): void
Pre e Post condizioni	context OrderDAO::createOrder() pre: true post: result = order→select() context OrderDAO::saveOrder() pre: true post:
Invarianti	

Nome Classe	**AccountEntity**
Modulo	account
Variabili di Istanza	- accountID : int - email : String - password : String - role : Role
Descrizione	
Firme dei Metodi	+ AccountEntity() + AccountEntity (accountID : int, email : String, password : String, role : Role) + getAccountID(): int + setAccountID(accountID : int): void + getEmail(): String + setEmail(email : String): void

	+ getPassword(): String + setPassword(password : String): void + getRole(): Role + setRole(role : Role): void
Pre e Post condizioni	• **Context** : AccountEntity:: AccountyEntity() **Pre**: true **Post**: self. accountID = -1 self. email = null self. Password = null self. role = null • **Context** : AccountEntity:: AccountyEntity(accountID : int, email : String,password : String, role : Role) **Pre**: true **Post**: self. accountID = accountID self. email = email self. Password = password self. role = role • **Context** : AccountEntity:: getAccountID() **Pre**: true **Post**: result = self.accountID • **Context**: AccountEntity::setAccountID(accountID : int) **Pre**: true **Post**: self. accountID = accountID • **Context** : AccountEntity:: getEmail() **Pre**: true **Post**: result = self.email • **Context**: AccountEntity::setEmail(email: String) **Pre**: true **Post**: self. email = email • **Context** : AccountEntity:: getRole() **Pre**: true **Post**: result = self. role • **Context**: AccountEntity::setRole(role: Role) **Pre**: true **Post**: self. role = role
Invarianti	

Nome Classe	**UserEntity**
Modulo	account
Variabili di Istanza	- firstName : String - lastName: String
Descrizione	
Firme dei Metodi	+ UserEntity (account : AccountEntity) + UserEntity (, firstName: String, lastName : String) + getFirstName(): String + setFirstName (firstName: String): void + getLastName (): String + setLastName (lastName : String): void
Pre e Post condizioni	• **Context** : AccountEntity:: UserEntity (account : AccountEntity) **Pre**: true **Post**: Super(account.getAccountID(), account.getEmail(), account.getPassword(), self. firstName = null self. lastName = null • **Context** : AccountEntity:: getFirstName() **Pre**: true **Post**: result = self.firstName • **Context**: AccountEntity:: setFirstName (firstName: String) **Pre**: true **Post**: self. firstName = firstName • **Context** : AccountEntity:: getLastName() **Pre**: true **Post**: result = self.lastName • **Context**: AccountEntity:: setLastName (lastName: String) **Pre**: true **Post**: self. lastName = lastName

Invarianti	

Nome Classe	**CartEntity**
Modulo	cart
Variabili di Istanza	-shoppingCart : HashMap<Integer, CartItemEntity> -totalAmount : double
Descrizione	
Firme dei Metodi	+ CartEntity () + getTotalAmount() : double + addItem(cartItem : CartItemEntity) : void + removeItem(productID : Integer) : void + checkItem (productID : Integer) : boolean + getShoppingCart () : HashMap<Integer, CartItemEntity> + setShoppingCart (shoppingCart : HashMap<Integer, CartItemEntity>) : void
Pre e Post condizioni	• **Context** : cartEntity:: addItem(cartItem : CartItemEntity) **Pre**: self.shoppingCart != null **Post**: self.shoppingCart -> include(CartItemEntity) • **Context** : cartEntity:: removeItem(productID : Integer) **Pre**: self.shoppingCart != null **Post**: self.shoppingCart ->!exsits(shoppingCart(Key)== productID) • **Context** : cartEntity:: checkItem (productID : Integer) **Pre**: self.shoppingCart != null **Post**: result = true || result = false
Invarianti	

Nome Classe	**CartItemEntity**
Modulo	cart
Variabili di Istanza	-product : ProductEntity -productQuantity : Integer
Descrizione	
Firme dei Metodi	+ CartItemEntity(product : ProductEntity, productQuantity : Integer) + getProduct () : ProductEntity + setProduct (product : ProductEntity) : void + getProductQuantity () : Integer + getProductQuantity (quantity : Integer) : void
Pre e Post condizioni	
Invarianti	

Nome Classe	**CartItemEntity**
Modulo	cart
Variabili di Istanza	-product : ProductEntity -productQuantity : Integer
Descrizione	
Firme dei Metodi	+ CartItemEntity(product : ProductEntity, productQuantity : Integer) + getProduct () : ProductEntity + setProduct (product : ProductEntity) : void + getProductQuantity () : Integer + getProductQuantity (quantity : Integer) : void
Pre e Post condizioni	
Invarianti	

Nome Classe	**PaymentEntity**
Modulo	payment
Variabili di Istanza	- paymentNumber : String - paymentStatus: String - paymentMethod : String - paymentDescription: String - paidAmount : double
Descrizione	
Firme dei Metodi	+ getPaymentNumber() : void + setPaymentNumber(paymentNumber : String) + getPaymentStatus () : void + setPaymentStatus(paymentStatus: String) + getPaymentMethod () : void + setPaymentMethod (paymentMethod: String) + getPaymentMethod () : void + getPaymentDescription (paymentDescription: String) + setPaymentDescription () : void + setPaymentAmount (paymentMethod: double)
Pre e Post condizioni	
Invarianti	

Nome Classe	**OrderController**
Modulo	order
Variabili di Istanza	- orderDAO: OrderDAO
Descrizione	La classe controlla la logia relativa all'ordine.
Firme dei Metodi	+protected doPost() +protected doGet()
Pre e Post condizioni	Pre: request!=null and request.getSession().getAttribute("

Invarianti	

Nome Classe	**OrderFacade**
Modulo	order
Variabili di Istanza	
Descrizione	
Firme dei Metodi	
Pre e Post condizioni	
Invarianti	