DOCUMENTATIE

**Tema 3 – ORDERS MANAGEMENT**

**NUME STUDENT: SCUTURICI VLAD LUCIAN**

**GRUPA: 30225**

# CUPRINS

1. Obiectivul temei ………………………………………………………………………3

2. Analiza problemei, modelare, scenarii, cazuri de utilizare …………………………...3

3. Proiectare ……………………………………………………………………………...4

4. Implementare ………………………………………………………………………….5

5. Rezultate ………………………………………………………………………………8

6. Concluzii ………………………………………………………………………………10

7. Bibliografie ……………………………………………………………………………11

Obiectivul Temei

Obiectivul temei este sa dezvoltars unei aplicatii de gestionare a comenzilor pentru procesarea comenzilor clientilor intr-un depozit. Pentru a realiza acest lucru, veti utiliza baze de date relationale pentru a stoca informatii despre produse, clienti si comenzile acestora. Arhitectura aplicatiei va fi Layered Architecture, care separa componentele aplicatiei in mai multe straturi functionale.

Clasele model vor reprezenta modelele de date ale aplicatiei si vor fi responsabile de definirea structurii si comportamentului acestora. Clasele de logica de afaceri vor contine logica aplicatiei, cum ar fi procesarea comenzilor, verificarea stocurilor, calcularea preturilor si gestionarea starii comenzilor. Clasele de prezentare vor fi responsabile de interfata cu utilizatorul, inclusiv afisarea informatiilor, interactiunea cu utilizatorul si gestionarea evenimentelor GUI. Clasele de acces la date vor furniza functionalitatea necesara pentru a accesa si manipula datele din baza de date, cum ar fi interogari, inserari si actualizari.

Prin utilizarea acestor clase si structurarea lor intr-un model de Layered Architecture, se va dezvolta o aplicatie modulara, usor de intretinut si extensibila. Aceasta va facilita separarea responsabilitatilor si va permite echipelor de dezvoltare sa lucreze independent asupra diferitelor componente ale aplicatiei.

Proiectare

Diagrama UML (Unified Modeling Language) este o modalitate standard de a reprezenta grafic elementele de proiectare ale unui sistem software. Această diagramă oferă o viziune de ansamblu asupra structurii și comportamentului unui sistem software, precum și a relațiilor dintre elementele acestuia. Diagrama UML poate fi utilizată pentru a defini modele de proiectare, pentru a planifica implementarea sistemelor software și pentru a documenta procesul de dezvoltare a acestora. Această diagramă este formată dintr-un set de simboluri și relații care descriu elementele componente ale sistemului și modul în care acestea comunică între ele. Diagrama UML este o unealtă importantă pentru proiectarea și dezvoltarea de sisteme software complexe.

A screenshot of a computer program

Description automatically generated with medium confidence

Tehnicile de reflexie in Java permit programatorilor sa acceseze si sa manipuleze dinamic informatii despre clase, obiecte, metode si campuri la timpul de executie. Reflexia este posibila datorita existentei in Java a unei clase numite "Class", care ofera metode si functionalitati pentru a examina si a interactiona cu elementele programului intr-un mod flexibil.

Prin utilizarea tehnicilor de reflexie, programatorii pot obtine informatii despre o clasa, cum ar fi numele, metodele, campurile si anotarile acesteia. De asemenea, pot crea noi instante ale unei clase, apela metode si accesa sau modifica campurile clasei, chiar daca acestea sunt private sau protejate. Tehnicile de reflexie pot fi folosite pentru a inspecta si a manipula dinamic obiecte, oferind o flexibilitate sporita in proiectarea si implementarea aplicatiilor.

Un exemplu de utilizare a reflexiei in Java ar putea fi obtinerea tuturor metodelor publice ale unei clase si apelarea lor dinamic, in functie de anumite criterii sau conditii la timpul de executie. Aceasta permite o adaptabilitate si o personalizare mai mare a comportamentului aplicatiei in functie de cerintele sau configurarile utilizatorului.

Desi reflexia poate fi un instrument puternic, este important de mentionat ca utilizarea excesiva a acesteia poate afecta performanta si poate face codul mai dificil de citit si de intretinut. Prin urmare, trebuie folosita cu atentie si doar atunci cand este absolut necesara intr-un anumit context.

Javadoc este o unealta specifica limbajului Java care permite generarea automata a documentatiei pentru codul sursa al aplicatiilor. Prin adaugarea de comentarii speciale in cod, folosind sintaxa specifica Javadoc, programatorii pot documenta clase, metode, campuri si alte elemente ale aplicatiei, inclusiv descrieri, parametri, valori returnate si exemple de utilizare. Utilizand Javadoc, dezvoltatorii pot crea documentatie cuprinzatoare si bine structurata, care poate fi ulterior generata sub forma de fisiere HTML sau accesata direct in cadrul unui IDE. Aceasta documentatie usureaza intelegerea si utilizarea codului de catre alti dezvoltatori, promoveaza bunatatea codului si favorizeaza colaborarea in echipa.

Implementare

Proiectul este alcatuit din urmatoarele clase si pachete:

I) Pachetul Model

1. Client

Clasa "Client" din pachetul "Model" reprezinta un model de date pentru un client in aplicatia de gestionare a comenzilor. Aceasta clasa contine campuri private pentru ID-ul clientului, nume si adresa de email. De asemenea, clasa include un constructor care primeste valorile ID-ului, numele si adresa de email si initializeaza campurile corespunzatoare. Clasa "Client" ofera metode de acces si modificare pentru campurile private, cum ar fi "getId()", "getName()", "getEmail()", precum si metodele "setId()", "setName()" si "setEmail()" pentru a actualiza valorile acestor campuri. Metoda "toString()" este suprascrisa pentru a returna o reprezentare text a obiectului "Client" in formatul "ID Nume Email". Aceasta clasa este utilizata pentru a stoca si accesa informatiile despre clienti in cadrul aplicatiei de gestionare a comenzilor.

2. Product

Clasa "Product" din pachetul "Model" reprezinta un model de date pentru un produs in aplicatia de gestionare a comenzilor. Aceasta clasa contine campuri private pentru ID-ul produsului, nume si cantitate. Clasa "Product" include un constructor care primeste valorile ID-ului, numele si cantitatea si initializeaza campurile corespunzatoare. Clasa ofera metode de acces si modificare pentru campurile private, cum ar fi "getId()", "getName()", "getQuantity()", precum si metodele "setId()", "setName()" si "setQuantity()" pentru a actualiza valorile acestor campuri. Metoda "toString()" este suprascrisa pentru a returna o reprezentare text a obiectului "Product" in formatul "ID Nume Cantitate". Aceasta clasa este utilizata pentru a stoca si accesa informatiile despre produse in cadrul aplicatiei de gestionare a comenzilor.

3. Order

Clasa "Order" din pachetul "Model" reprezinta un model de date pentru o comanda in aplicatia de gestionare a comenzilor. Aceasta clasa contine campuri private pentru ID-ul comenzii, ID-ul clientului, ID-ul produsului si cantitatea comandata. Clasa "Order" include un constructor care primeste valorile ID-ului comenzii, ID-ul clientului, ID-ul produsului si cantitatea si initializeaza campurile corespunzatoare. Clasa ofera metode de acces si modificare pentru campurile private, cum ar fi "getId()", "getClientId()", "getProductId()", "getQuantity()", precum si metodele "setId()", "setClientId()", "setProductId()" si "setQuantity()" pentru a actualiza valorile acestor campuri. Metoda "toString()" este suprascrisa pentru a returna o reprezentare text a obiectului "Order" in formatul "ID Client\_ID Product\_ID". Aceasta clasa este utilizata pentru a stoca si accesa informatiile despre comenzile clientilor in cadrul aplicatiei de gestionare a comenzilor.

II) Pachetul BusinessLogic

1. Client BLL

Clasa "ClientBLL" din pachetul "BusinessLogic" reprezinta o componenta esentiala in aplicatia de gestionare a comenzilor. Aceasta clasa are rolul de a gestiona operatiile specifice clientilor, cum ar fi gasirea unui client dupa ID, inserarea unui client nou si obtinerea tuturor clientilor existenti. De asemenea, clasa se ocupa de aplicarea regulilor de validare asupra datelor clientului, inclusiv validarea adresei de email. Prin intermediul validarilor si interactiunii cu clasa "ClientDAO" din pachetul "DataAccess", clasa "ClientBLL" asigura integritatea datelor si corectitudinea operatiilor desfasurate in cadrul sistemului. Aceasta reprezinta un intermediar intre componenta de acces la date si componenta de prezentare, oferind servicii si functionalitati necesare pentru manipularea si gestionarea entitatilor client din aplicatie.

2. Product BLL

Clasa "ProductBLL" din pachetul "BusinessLogic" reprezinta o componenta esentiala in aplicatia de gestionare a comenzilor, avand rolul de a gestiona operatiunile specifice produselor. Aceasta clasa furnizeaza metode si functionalitati necesare pentru a gasi produse dupa ID, insera produse noi si obtine lista tuturor produselor existente in sistem. De asemenea, clasa se ocupa de validarea datelor produselor utilizand o lista de validatori asociati. Prin intermediul interactiunii cu clasa "ProductDAO" din pachetul "DataAccess", clasa "ProductBLL" asigura persistenta si integritatea datelor referitoare la produse in baza de date. Aceasta reprezinta un element-cheie in procesul de gestionare a stocului si a informatiilor despre produse, oferind functionalitati esentiale pentru buna functionare a aplicatiei de gestionare a comenzilor.

III) Pachetul DataAcces

Abstract, Client, Order, Product DAO.

Clasele de tip DAO (Data Access Object) sunt componente esentiale intr-o arhitectura software care implica interactiunea cu baza de date sau cu alte surse de date. Aceste clase se ocupa de abstractizarea accesului la date si de oferirea unei interfete standardizate pentru efectuarea operatiunilor de citire, scriere, actualizare si stergere a datelor. Clasele de tip DAO sunt concepute pentru a separa logica de afaceri de detaliile specifice accesului la date, permitand astfel o modularitate mai mare si o intretinere usoara a codului. Prin intermediul metodelor din clasele DAO, se pot executa operatiuni specifice asupra datelor, cum ar fi cautarea, inserarea sau actualizarea, oferind astfel un nivel de abstractizare si reutilizare a codului in cadrul aplicatiei. Aceste clase faciliteaza manipularea datelor si asigura persistenta acestora, asigurand in acelasi timp o separare clara intre logica de afaceri si accesul la date.

IV) Pachetul Presentation cu clasa GUI.

Clasa care este responsabila pentru interfata grafica, in care utilizatorul poate sa vizualizeze tabelele si sa adauge / editeze / stearga randuri. Cu optiunea “Add Order”, utilizatorul poate selecta un client, un produs si poate alege un ID, pentru a crea o comanda.

RezultateA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Concluzii

Realizarea acestui proiect de database management a fost o experiență valoroasă și educativă, care a implicat o serie de competențe precum programare orientată obiect, algoritmi și structuri de date, dezvoltarea de interfețe grafice cu utilizatorul și abilități de gestionare a proiectelor. Proiectul a oferit o oportunitate excelentă de a aplica cunoștințele teoretice într-un proiect practic și de a dezvolta abilități și experiență practică în dezvoltarea de software. În general, acest proiect a fost o modalitate excelentă de a exersa și consolida abilitățile de programare, precum și de a crea un produs final care poate fi utilizat într-un mediu real.

Bibliografie

1. https://dsrl.eu/courses/pt/ - cerinte, prezentari, exemple

2. https://www.javatpoint.com/java-swing - tutorial javaswing