Documentatie Tema 4

-**RESTAURANT MANAGEMENT SYSTEM** -

Nume: Hilbert Dennis

Seria: B, Grupa: 30226 An 2, CTI

Cuprins

Cuprins. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2

Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

-Descrierea generala a proiectului . . . . . . . . . . . . . . . . . . . 3

-Descrierea claselor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

-Pachetul business. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

-Pachetul data. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .11

-Pachetul presentation. . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

-Pachetul main. . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .12

Rezultate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

Concluzii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .13

1. Obiectivul temei

Obiectivul temei consta in implementarea unei aplicatii care indeplineste functionalitatile unui restaurant. Sistemul are trei tipuri de utilizatori: administrator, chelner si bucatar. In cazul de fata se considera ca sistemul este folosit de cate un singur utilizator din fiecare tip.

Administratorul poate adăuga, șterge și modifica produse existente din meniu. Chelnerul poate crea o nouă comandă pentru un tabel, poate adăuga elemente din meniu și poate calcula factura pentru o comandă. Bucătarul este înștiințat de fiecare dată când trebuie să gătească mâncare care este comandată printr-un ospătar.

Obiectivele secundare sunt reprezentate de crearea si structurarea intr-un mod avantajos a pachetelor, claselor si functiilor care le compun. Acestea ar fi, in ordine, urmatoarele:

-respectarea diagramei de clase prezentata in assignment

-utilizarea conceptului de serializare(aplicatia preia date despre clasa restaurant dintr-un fisier de tip SER iar dupa utilizare, incarca noile date in acest fisier)

-utilizarea Composite Design Pattern

-utilizarea javaDoc

-crearea unui fisier .jar cu care aplicatia va putea fi rulata

-utilizarea Observer pattern

Acestea urmeaza sa fie detaliate in capitolul 3 care descrie in detaliu proiectarea aplicatiei.

2.Proiectare

2.1Descrierea generala a proiectului:

Din punct de vedere structural, proiectul contine patru pachete si anume main, business, data si presentation.

In pachetul business se afla clasele BaseProduct, Composite product, MenuItem, Observable, Order, Restaurant si interfata IRestaurantProcessing. Acestea contin metode care se ocupa in mare parte de executia programului, mai exact de operatiile pe date.

In pachetul presentation se afla clasele AdministratorGUI, ChefGUI, WaiterGUI si interfata Observer. Acestea se ocupa de implementarea interfetelor grafice pentru utilizator si ajuta la urmarirea evolutiei programului.

In pachetul data se afla clasele AdministratorController, FileWriter, RestaurantSerializator si WaiterController.

In pachetul Main se afla clasa Main care contine metoda Main care controleaza actiunile programului.

2.2 Descrierea claselor:

**Pachetul business:**

-Clasa BaseProduct

* Metode:

public BaseProduct(String name, double price) //constructor  
public double calculatePrice()  
  
public boolean find(MenuItem item)

-Clasa CompositeProduct

Metode:

private List<MenuItem> products; //atribut  
  
 public CompositeProduct(String name, ArrayList<MenuItem> products) {  
 super(name, 0);  
 this.products = products;  
 this.setPrice(calculatePrice());  
 }  
  
 public List<MenuItem> getProducts()  
  
 public void addProduct(MenuItem product)   
  
 public double calculatePrice()  
  
 public void removeProduct(MenuItem product)

public boolean find(MenuItem item)   
}

-Interfata IrestaurantProcessing

Metode:

BaseProduct createProduct(String name, double price);

CompositeProduct createProduct(String name, ArrayList<String> nameProducts);  
  
void deleteProduct(String name);  
void modifyProduct(String name, double newPrice);  
  
Order placeOrder(int table, ArrayList<String> orderedItems);  
  
 double calculatePrice(Order order);  
  
String generateBill(Order order);

-Clasa MenuItem implements Serializable

Metode:

private String name;  
private double price; //atribute  
  
public MenuItem(String name, double price) { //constructor  
 this.name = name;  
 this.price = price;  
}  
  
public double getPrice()  
  
public void setPrice(double price)

public String getName  
  
public void setName(String name)  
  
public abstract double calculatePrice();  
  
public abstract boolean find(MenuItem item);

-Clasa Observable

Metode:

private ArrayList<Observer> obs; //atribut  
  
 public Observable() { //constructor  
 obs = new ArrayList<>();  
 }  
  
 public void createNewObserver(Observer observer, String s)

public void announceObserver()

-Clasa Order

Metode:

public int orderID; //atribute  
 private int table;  
 public static int *order\_nr* = 1;  
 private SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");  
 private String date = "";  
  
 public Order(int orderID, int table) //constructor

public int getOrderID()

public String getOrderDate()

public int getTable()

public int getHashCode()

-Clasa Restaurant

Metode:

private static final long *serialVersionUID* = 15952774314598L;  
private LinkedHashMap<String, MenuItem> menu = new LinkedHashMap<>();  
private HashMap<Order, List<MenuItem>> orders = new HashMap<>(); //atribute  
  
public BaseProduct createProduct(String name, double price)   
public CompositeProduct createProduct(String name, ArrayList<String> nameProducts) throws NullPointerException

public boolean searchItemInMenu(String name)

public void deleteProduct(String name)

public void modifyProduct(String name, double newPrice)

public Order placeOrder(int table, ArrayList<String> orderedItems) throws NullPointerException

public double calculatePrice(Order order

public String getProductInfo(String name)

public String getOrderInfo(Order order)   
  
private boolean isInitialized()

public HashMap<String, MenuItem> getMenu()

public HashMap<Order, List<MenuItem>> getOrders()

**Pachetul data:**

-Clasa AdministratorGUI

private Restaurant model;  
private AdministratorGUI view; //atribute  
  
public AdministratorController(AdministratorGUI view, Restaurant model)

public void updateTable()

private void addAllListeners(Restaurant model, AdministratorGUI view)   
  
class BaseProductListener implements ActionListener

class ModifyListener implements ActionListener

public void deleteRows()

class DeleteListener implements ActionListener   
  
 public void actionPerformed(ActionEvent e)

public void deleteRows()   
  
class CompositeProductListener implements ActionListener   
  
 public void actionPerformed(ActionEvent e)

public ArrayList<String> compositeSplit(String string)

class SelectionListener implements MouseListener

public void mouseClicked(MouseEvent e)

public void mousePressed(MouseEvent e)   
  
 public void mouseReleased(MouseEvent e)  
   
 public void mouseEntered(MouseEvent e)   
  
 public void mouseExited(MouseEvent e)   
  
class DetListener implements ActionListener   
  
 public void actionPerformed(ActionEvent e)

-Clasa FileWriter

public void writeInFile(String string, String fileName) throws FileNotFoundException, UnsupportedEncodingException

-Clasa RestaurantSerializator

package data;  
  
import java.io.\*;  
import business.Restaurant;  
  
public class RestaurantSerializator {  
 private File file;  
 private Restaurant rest; //atribute  
  
 public RestaurantSerializator(Restaurant rest, String fileName)

public Restaurant getRestaurant()  
  
 public void dataLoad()

public void dataSave()

-Clasa WaiterController

private WaiterGUI view;  
 private Restaurant model;  
 public static int *bill\_nr* = 0; //atribute  
   
  
 public WaiterController(WaiterGUI view, Restaurant model) {  
 this.view = view;  
 this.model = model;  
 view.addGetInfoListener(new GetInfoListener());  
 view.addCreateListener(new CreateListener());  
 view.addGenerateBillListener(new GenerateBillListener());  
  
 } //constructor  
  
 public void updateTable

class GetInfoListener implements ActionListener

public void actionPerformed(ActionEvent e)

class CreateListener implements ActionListener

public void actionPerformed(ActionEvent e)

public ArrayList<String> splitString(String string)

class GenerateBillListener implements ActionListener

public void actionPerformed(ActionEvent e)

**Pachetul presentation:**

-Clasa AdministratorGUI

public void createAddProductListener(ActionListener l)   
  
 public void createDetailListener(ActionListener l)

public void createModifyListener(ActionListener l

public void createDeleteListener(ActionListener l) }  
  
 public void createAddCompositeProductListener(ActionListener l)   
  
 public void createSelectionListener(MouseListener l)   
  
 public String getUserProductNameInput()

public String getComparedString()

public String getUserPriceInput()  
  
 public String getComparedStringV2()  
  
 public JTextField getNameTextField()  
  
 public void setNameTextField(JTextField nameTextField)   
  
 public JTextField getPriceTextField()  
  
 public void setPriceTextField(JTextField priceTextField)   
  
 public String getUserCompositeNameInput()  
  
 public String getUserComponentsInput()  
  
 public JTable getTable()

public void setTable(JTable table)   
  
 public JTextField getCompositeNameTextField()  
  
 public void setCompositeNameTextField(JTextField compositeNameTextField)  
  
 public JTextField getComponentsTextField()  
  
 public void setComponentsTextField(JTextField componentsTextField)  
  
 public DefaultTableModel getModel  
  
 public void setModel(DefaultTableModel model)   
  
 public void showErrMessage(String msg)   
  
 public void showMessage(String msg)  
  
}

Metode similare in:

-Clasa ChefGUI

-ClasaWaiterGUI

**Pachetul main:**

-Clasa Main

* Metode:

public static void main(String[] args)

3.Implementare

-Pachetul business

Pachetul business coordoneaza principalele functionalitati ale programului. Clasele BaseProduct si CompositeProduct mostenesc clasa MenuItem iar acestea impreuna definesc produsele cu care restaurantul lucreaza. Clasa cea mai importanta a acestui pachet este Restaurant. Aceasta implementeaza interfetele IRestaurantProcessing si Serializable si detine metode pentru adaugare produs, modificare pret, stergere produs, generare nota in format pdf, obtinere informatii despre produse, etc.

Un exemplu de generare de produs de baza este:

public BaseProduct createProduct(String name, double price) {  
 if (name != null && price >= 0) {  
 BaseProduct baseProduct = new BaseProduct(name, price);  
 menu.put(name.toLowerCase(), baseProduct);  
 assert isInitialized() : "Instanta nu este initializata corect";  
 assert !baseProduct.getName().isEmpty() && baseProduct.getPrice() != 0 : "Nu se poate crea produsul";  
 return baseProduct;  
 } else {  
 throw new IllegalArgumentException("Parametru null pentru nume sau pret negativ");  
 }  
}

Iar un exemplu de generare PDF este:

public String generateBill(Order order) {  
 if (order != null) {  
 String str = " Nota de plata\n";  
 str += ("ID Comanda: " + order.getOrderID() + "\n");  
 str+= "Produse:\n";  
 List<MenuItem> orderedItems = orders.get(order);  
 for (MenuItem item : orderedItems) {  
 if (item != null)  
 str += (item.getName() + " : " + item.getPrice() + "\n");  
 }  
 str += ("Pret total: " + calculatePrice(order) + "\n");  
 return str;  
 } else {  
 throw new IllegalArgumentException("Comanda invalida");  
 }  
}

-Pachetul data

Pachetul data se ocupa mai mult de partea de control, de monitorizare, implementare de Listener-uri. Aici se detecteaza ce actiuni se vor a fi simulate si in functie de functiile selectate in GUI se iau decizii in continuare.

Pachetul data contine metode pentru detectarea acestor actiuni(de click pe butoane, textField-uri sau randuri in tabele) dar si pentru prelucrarea datelor furnizare prin intermadiul acestora.

De exemplu, mai jos este prezentata metoda de splitString care separa produsele unui compozit introduse initial ca un singur String:

public ArrayList<String> compositeSplit(String string) {  
 String[] nameArray = string.split(", ");  
 ArrayList<String> resultArray = new ArrayList<>();  
 for (String s : nameArray) {  
 resultArray.add(s);  
 }  
 return resultArray;  
}

Clasa care se ocupa de Listener-ul pentru produsele de baza:

class BaseProductListener implements ActionListener {  
  
 public void actionPerformed(ActionEvent e) {  
 String[] itemRow = new String[3];  
 if (!view.getUserProductNameInput().isEmpty() && !view.getUserPriceInput().isEmpty()) {  
 if (!model.searchItemInMenu(view.getUserProductNameInput())) {  
 double price = Double.*parseDouble*(view.getUserPriceInput());  
 String itemName = view.getUserProductNameInput();  
 model.createProduct(itemName, price);  
 itemRow[0] = itemName;  
 itemRow[1] = view.getUserPriceInput();  
 itemRow[2] = "Produs simplu";  
  
 view.getModel().addRow(itemRow);  
  
 } else {  
 view.showErrMessage("Produsul a fost deja adaugat in meniu");  
 }  
 } else {  
 view.showErrMessage("Datele produsului nu sunt valide");  
 }  
 }  
}

Si de asemenea, metoda care creaza Listener-uri:

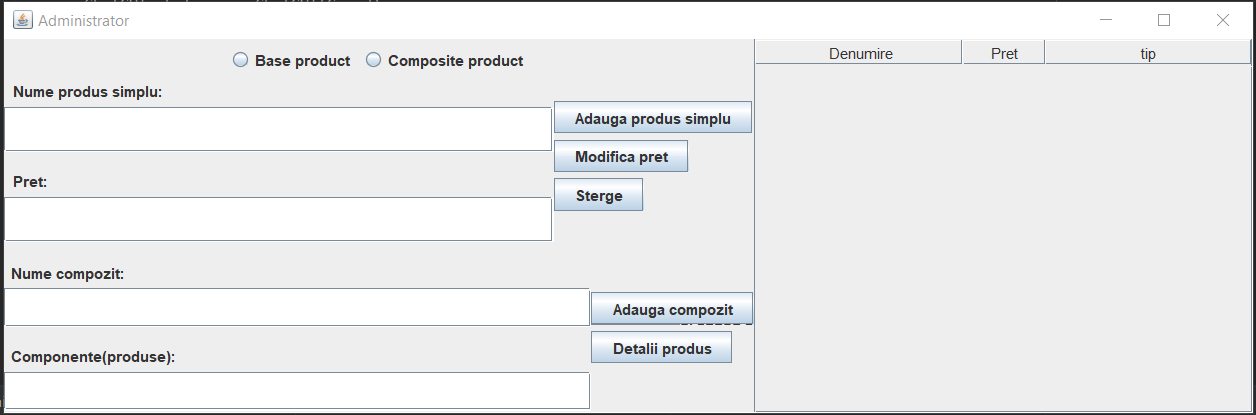
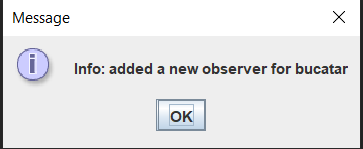
private void addAllListeners(Restaurant model, AdministratorGUI view) {  
 view.createAddProductListener(new BaseProductListener());  
 view.createModifyListener(new ModifyListener());  
 view.createDeleteListener(new DeleteListener());  
 view.createDetailListener(new DetListener());  
 view.createSelectionListener(new SelectionListener());  
 view.createAddCompositeProductListener(new CompositeProductListener());  
}

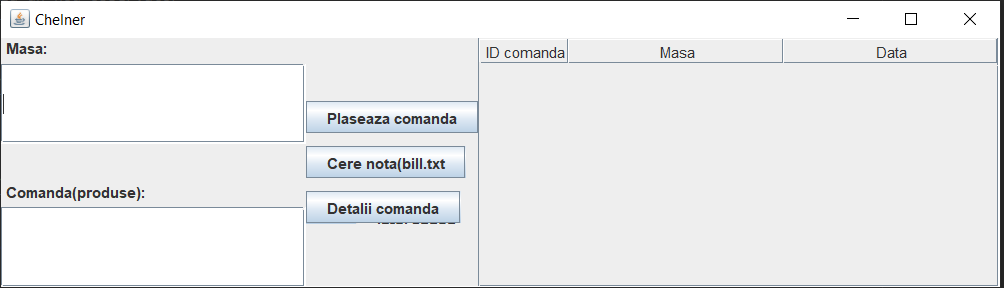
-Pachetul presentation

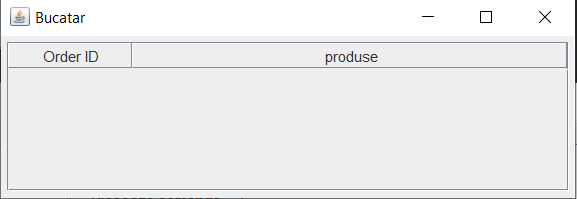
Pachetul presentation contine clasele pentru interfata grafica(AdministratorGUI, ChefGUI, WaiterGUI si interfata observer. Acestea creaza efectiv ferestrele prin intermediul carora utilizatorul interactioneaza cu sistemul.

Un exemplu de construire de panel:

panel5.add(Box.*createVerticalStrut*(5));  
panel5.add(addBaseProductButton);  
panel5.add(Box.*createVerticalStrut*(5));  
panel5.add(modifyButton);  
panel5.add(Box.*createVerticalStrut*(5));  
panel5.add(deleteButton);  
panel5.setLayout(new BoxLayout(panel5, BoxLayout.*Y\_AXIS*));  
panel5.add(Box.*createVerticalStrut*(15));

Si cate un printScreen cu fiecare componenta GUI:



-

Pachetul main

Pachetul main coordoneaza intreaga executie a programului prin functiile acestuia deja implementate(asteapta un argument la executie reprezentand fisierul pentru serializare).

4. Rezultate:

Rezultatele obtinute sunt ilustrate cel mai bine in fisierele de tip PDF generate in urma executiei programului dar si pe parcursul executiei in cele trei ferestre GUI generate de program.

5. Concluzii:

In concluzie, in Java pot fi descrise sisteme din viata reala, mai mult sau mai putin complexe, prin intermediul unor clase care reprezinta obisctele si metode care reprezinta actiunile.

Interfata grafica ajuta foarte mult la interactiunea dintre utilizator si program, simuland un exemplu de desfasurare a actiunilor asa cum acestea ar succede intr-un restaurant real.

Alegerea corespunzatoare a strucurilor de date si a functionalitatilor pe care instantele create le detin usureaza procedura si contribuie la simularea cat mai exacta a sistemului.

6. Bibliografie:

* + http://www.tutorialspoint.com/java/java\_serialization.htm
  + o https://www.baeldung.com/java-serialization
  + o https://www.geeksforgeeks.org/serialization-in-java/
  + o https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html
  + • Java HashMap o http://javarevisited.blogspot.ro/2011/02/how-hashmap-works-in-java.html
  + • Java assert o http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html
  + o http://javarevisited.blogspot.ro/2012/01/what-is-assertion-in-java-java.html
  + o http://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise
  + o https://intellij-support.jetbrains.com/hc/en-us/community/posts/207014815-How-to-enable-assert
  + • Adding custom tags to javadoc o https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .