Tehnici de programare fundamentale – Tema 4

Simulator al managementului unui sistem de catering

1. Obiectivul temei

Principalul obiectiv al acestei teme este de a proiecta și implementa un program care să simuleze gestiunea unui site de catering, unde clienții să își poată crea un cont, să filtreze și să comande produse; angajații să fi notificați la apariția unei noi comenzi plasate pe tura lor, iar administratorii să poată adăuga produse, să le editeze sau să le șteargă pe cele deja existente, să importe produse noi dintr-un fișier .csv, să combine mai multe produse existente pentru a crea un produs compus sau să genereze diferite tipuri de rapoarte. Programul va avea interfață grafică dedicată, concepută într-o manieră intuitivă și atractivă pentru utilizator, atât pentru client, cât și pentru angajați și administratori. De asemenea, pentru fiecare comandă se va genera un fișier text cu rol de confirmare a comenzii și de bon fiscal, pe care să apară datele clientului, numărul comenzii, data și ora la care a fost plasată, produsele comandate (cu cantitatea corespunzătoare), prețul pe bucată și per total al fiecărui produs, dar și totalul de plată al comenzii.

Proiectul își propune să îi permită utilizatorului să:

- introducă numele de utilizator și parola pentru a se loga
- să aibă acces la anumite funcționalități ale programului, în funcție de tipul de utilizator

Obiectivele secundare pe care dezvoltatorul și le propune se regăsesc în tabelul de mai jos, alături de secțiunile din prezenta lucrare în care vor fi adresate:

	Obiectiv secundar	Scurtă descriere	Secțiunea dedicată
1.	Analiza problemei	- stabilirea cerințelor la care programul trebuie să răspundă	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
2.	Modelare	- imaginarea modelului conceptual pentru programul ce urmează a fi dezvoltat	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3.	Scenarii posibile	- analiza exhaustivă a situațiilor în care programul poate fi pus de către utilizator (input valid, șirul nul ca nume de fișier, numere negative ca valori de preț ori cantitate, etc.)	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
4.	Cazuri de utilizare	- determinarea cazurilor pentru care utilizatorul dorește să folosească programul creat	2. Analiza problemei, modelare, scenarii, cazuri de utilizare

5.	Proiectare	- definirea structurilor de date necesare, a claselor și interfețelor ce urmează a fi implementate, a tabelelor bazei de date ce va fi folosită în spatele programului	3. Proiectare
6.	Implementare	- descrierea claselor și a metodelor definite, explicarea funcționării și a utilității acestora; descrierea interfețelor grafice; descriere JavaDoc	4. Implementare
7.	Testare	- generarea fișierelor text cu rol de confirmare a comenzii și de bon fiscal, dar și a rapoartelor create de administrator	5. Rezultate

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

2.1. Analiza problemei

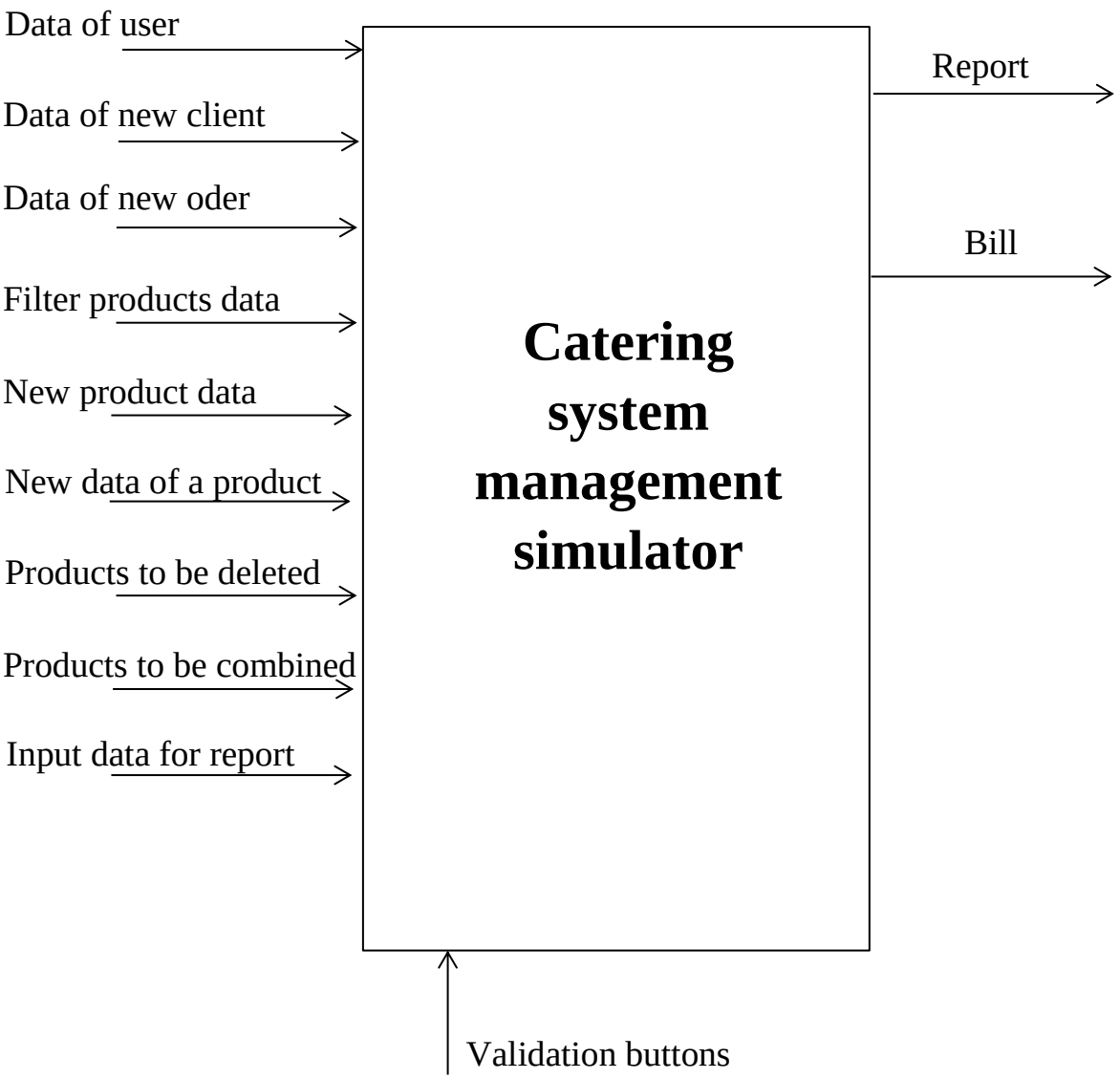
S-au stabilit cerințele funcționale și non-funcționale la care programul, simulatorul managementului unui magazin online, trebuie să răspundă:

a) Cerințe funcționale:
   a) simulatorul trebuie să permită utilizatorului să introducă de logare și să stabilească dacă sunt valide sau nu
   b) simulatorul trebuie să permită unui client nou să își creeze ad-hoc un cont de client
   c) simulatorul trebuie să permită clientului să filtreze produsele din meniu, după oricare dintre caracteristicile afișate și să plaseze o comandă
   d) simulatorul trebuie să permită angajatului să vadă în timp real care sunt comenzile plasate, ce produse conțin și care client a făcut comanda
   e) simulatorul trebuie să permită administratorului să importe produse dintr-un fișier cu format prestabilit (.csv), să adauge, editeze și șteargă produse, să creeze produse compuse din produsele deja existente, să genereze rapoarte referitoare la toată activitatea sistemului
   f) simulatorul trebuie să informeze utilizatorul dacă inputul său nu este valid
   g) simulatorul trebuie să furnizeze un fișier cu rol de confirmare a comenzii și de factură/ bon fiscal pentru fiecare comandă plasată

b) Cerințe non-funcționale:
   - simulatorul trebuie să fie ușor de folosit și intuitiv
   - simulatorul trebuie să aibă o interfață atractivă, atât pentru introducerea datelor unui utilizator la logare sau pentru crearea unui nou cont de client, dar mai ales pentru editarea, ștergerea sau vizualizarea unor intrări de date deja existente, precum și pentru introducerea comenzilor și generarea de rapoarte

2.2. Modelare

S-a definit modelul de lucru al simulatorului de cozi de clienți, sub formă de schemă bloc:



2.3. Scenarii posibile

Scenariile în care se poate găsi simulatorul managementului sistemului de catering pot fi diferențiate în două mari categorii: scenarii uzuale de utilizare și scenarii limită. În cele ce urmează se va prezenta felul în care programul va răspunde la fiecare dintre aceste tipologii de scenarii:

I. Scenariu uzual:
   - utilizatorul introduce corect datele sale de logare
   - dacă este un client, filtrează produsele introducând corect datele după care să se facă filtrarea; realizează o comandă validă
   - dacă este angajat, are doar posibilitatea de a vizualiza plasarea de noi comenzi
   - dacă este administrator, are posibilitatea de a importa produse dintr-un fișier specificat, de a adăuga, edita și șterge produse, de a crea produse combinate din cele deja existente, și se a genera 4 tipuri de rapoarte, în funcție de anumite criterii prestabilite
   - programul permite executarea unui nou ciclu de utilizare/logare fără a părăsi și a reporni aplicația

II. Scenarii limită:
   a. Șir de caractere în loc de numere:
      - utilizatorul client introduce un șir de caractere pe post de valoare întreagă pentru cantitatea unui produs
      - utilizatorul acționează butonul care ar plasa o comandă
      - programul îl notifică despre faptul că datele pot fi doar numere întregi, și îi dă posibilitatea de a corecta eroarea => utilizatorul are posibilitatea să intre într-un scenariu uzual
   b. Rating în afara intervalului 0 - 5:
      - utilizatorul administrator introduce un rating mai mic decât 0 sau mai mare decât 5 pentru un produs deja existent (la editare), sau pentru un produs ce se vrea a fi creat
      - utilizatorul acționează butonul care ar introduce datele in baza de date
      - programul îl notifică despre faptul că rating-ul este o valoare reală între 0 și 5 => utilizatorul are posibilitatea să intre într-un scenariu uzual
   c. Număr negativ:
      - utilizatorul introduce un număr negativ în unul dintre câmpurile pentru cantitatea, rating-ul, caloriile, proteinele, grăsimile, sodiul sau prețul unui nou produs
      - utilizatorul acționează butonul ferestrei de adăugare a unui nou produs
      - programul îl notifică despre faptul că datele nu pot fi numere negative; utilizatorul are posibilitatea să intre într-un scenariu uzual

d. Șirul vid
   - utilizatorul lasă necompletat unul dintre câmpurile pentru un client sau pentru un produs
   - utilizatorul acționează butonul ferestrei curente
   - programul îl notifică despre faptul că field-urile rămase necompletate trebuie să fie scrise, și îi dă posibilitatea să facă această adăugare

Este important de menționat că în cazul editării datelor unor clienți/produse deja existente, notificarea asupra inconsistenței datelor de intrare va fi semnalată imediat după completarea câmpului, și nu doar la apăsarea butonului. Astfel utilizatorul este mai rapid notificat asupra erorii.

2.4. Cazuri de utilizare

Descrierea utilizării:

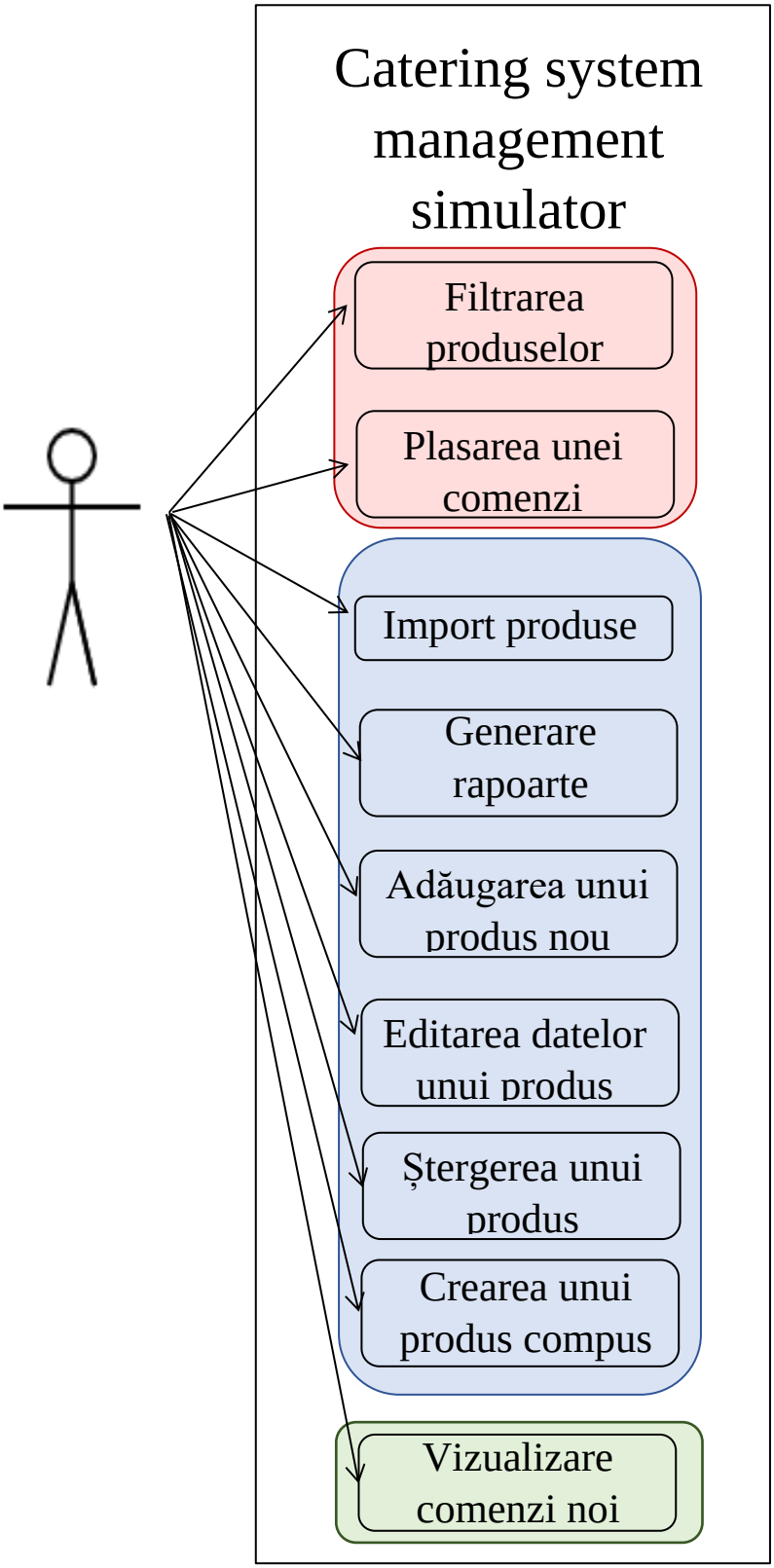
Folosirea simulatorului magazinului online este facilă și intuitivă. Datele de intrare cerute nu conțin niciun fel de termeni de specialitate care să deruteze utilizatorul. Mai mult decât atât, comunicarea cu user-ul în caz de eroare la introducerea datelor este una extrem de complexă și explicită.

De exemplu, pentru adăugarea unui nou produs de către un administrator, utilizatorul introduce datele de intrare în câmpurile specifice folosind tastatura propriului computer. La apăsarea butonului *Add*, dacă există date incorecte se va afișa pe ecran o casetă de dialog ce va semnala eroarea. Dacă toate datele sunt valide, se închide fereastra de adăugare și rezultatul se poate observa prin deschiderea ferestrei de editare/ștergere/combinare a produselor. Obiectul ce tocmai a fost adăugat se va găsi în tabelul afișat (nu se poate specifica poziția la are se va găsi, fiind vorba despre un HashSet pentru reținerea datelor, în scopul accesării cât mai rapide a acestora).

Pentru ștergerea unui produs se va afișa în fereastra corespunzătoare un tabel cu toate produsele. Se vor selecta rândurile care se dorește a fi șterse din baza de date. Se pot selecta oricâte rânduri. La apăsarea butonului *Delete* înregistrările selectate se vor șterge din sistem.

Pentru editarea datelor unui produs se va deschide în fereastra corespunzătoare un tabel în care apar toate produsele. Se modifică datele dorite. Se selectează rândurile ce au fost modificate și se apasă butonul *Edit*.

Pentru plasarea unei comenzi de către un client, pe ecran se va afișa un tabel cu toate produsele din sistem. Coloana de cantitate a tuturor produselor este completată implicit cu 0. Nu se poate plasa nici o comandă goală, nici o comandă în care există cel puțin un produs cu cantitatea 0! Se completează corect cantitatea dorită pentru fiecare produs, se selectează rândurile ce descriu produsele de comandat și se apasă butonul *Order*.



Pentru logare se vor introduce datele de utilizator (nume si parolă) și se va acționa butonul de *Log In.*

Se va deschide o fereastră corespunzătoare tipului de utilizator care tocmai s-a logat. Un utilizator nu are acces la funcționalitățile celorlalte tipuri de utilizatori.

Rațiunile pentru care utilizatorul poate dori să folosească acest simulator al managementului unui sistem de catering, în funcție de tipul de user (roz – client, albastru – admin, verde-angajat):

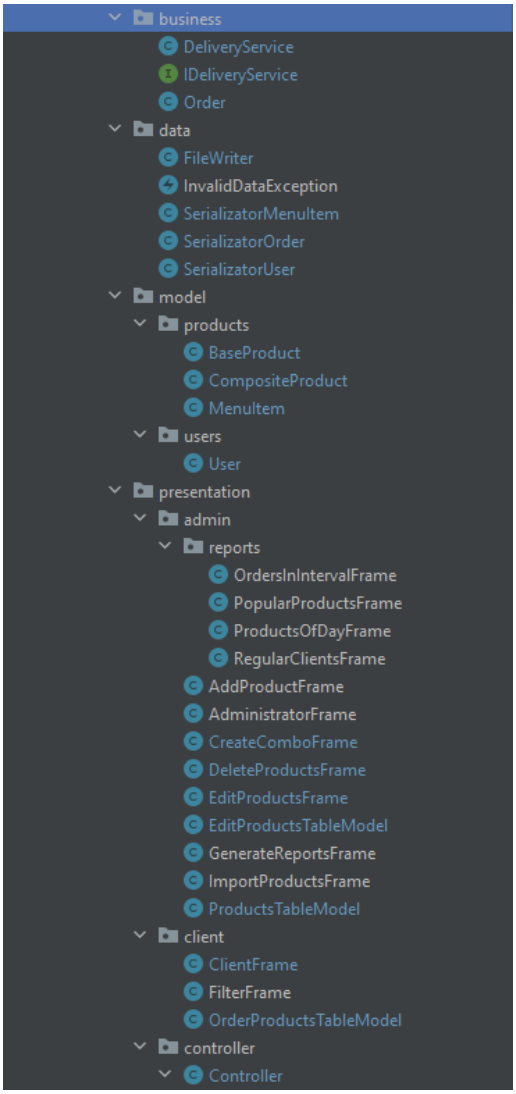
3. Proiectare

Pentru a respecta conceptele programării orientate pe obiect, fiecare clasă implementată este cât mai specifică, având cât mai puține funcționalități (ex: clasă separată pentru fiecare tip de produs: BaseProduct, CompositeProduct, MenuItem).

3.1. Pachete

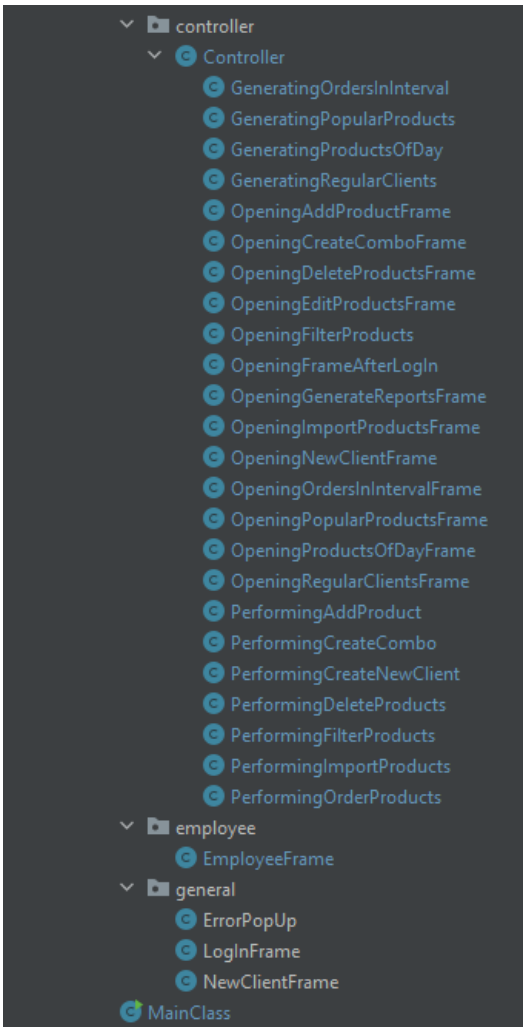
S-a folosit Layered Architecture pentru structurarea proiectului, ceea ce înseamnă că există câte un pachet care să răspundă următoarelor necesități: validarea datelor (Busssiness Layer - bll), accesul prin import, serializare și deserializare (data), modelarea în Java a informațiilor din MySQL (model) și interacțiunea cu utilizatorul (presentation).

Proiectul este structurat pe unități funcționale cât mai specifice, clasele care se ocupă de o anumită ramură a programului fiind grupate în pachete și sub-pachete, după cum se poate vedea alături în lista claselor așa cum apare ea în IDE-ul IntelliJ, dar și mai jos în diagrama de pachete:



Structura proiectului.

Organizarea pe pachete



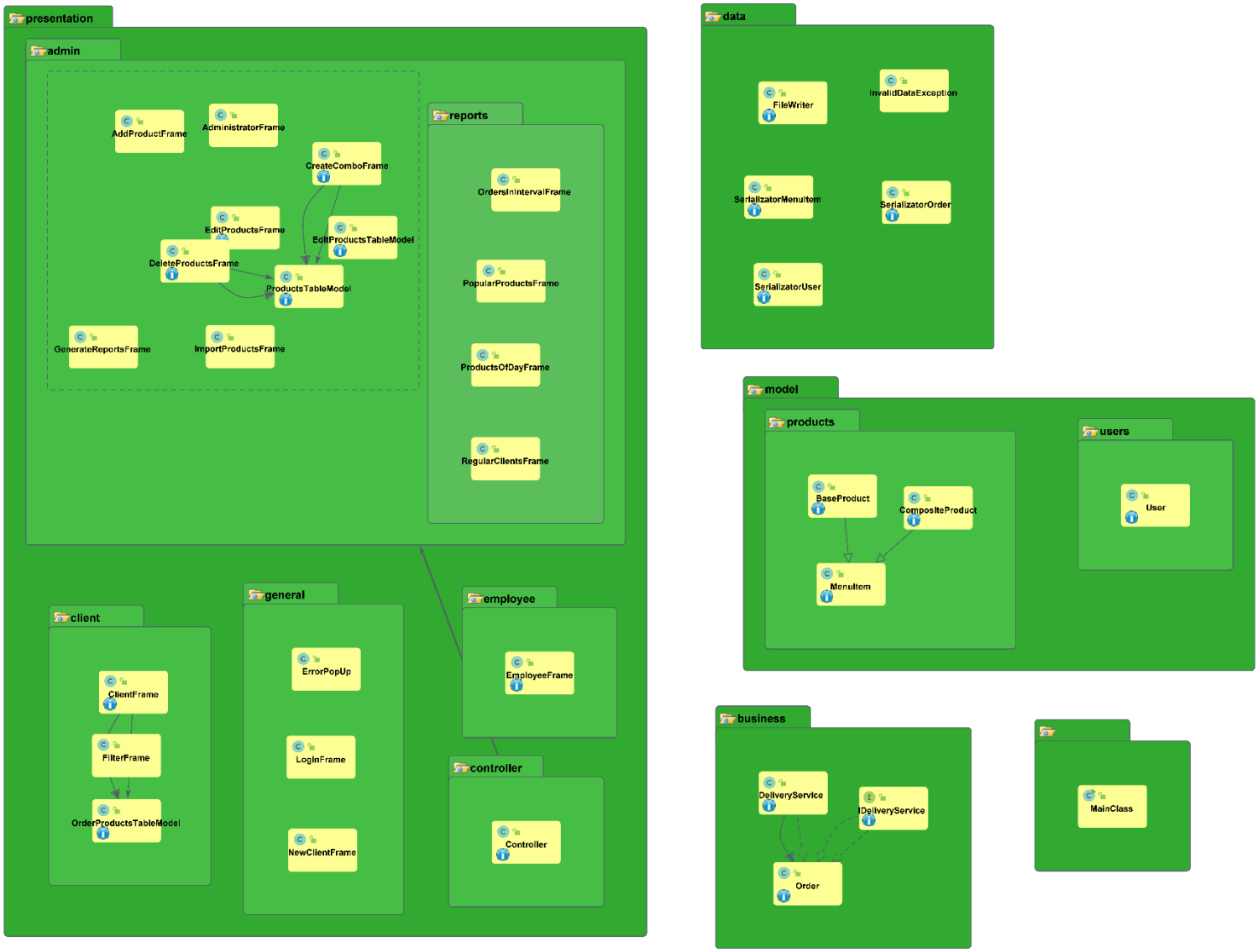


Diagrama de pachete

3.2. Clase

Diagrama de clase a întregului proiect arată relațiile dintre cele 34 de clase și interfețe definite, și este atașată pe următoarea pagină. Clasele vor fi descrise detaliat în secțiunea 4.Implementare.

3.3.Interfețe definite

Singura interfață definită în proiect este *IDeliveryService*, folosită pentru realizarea operațiilor de plasare a unei comenzi, generare a unui bon, generare de rapoarte, filtrare de produse (prin metode statice).

Clasa DeliveryService oferă o implementare particulară a metodelor non-statice din interfață. Pentru această clasă s-a folosit paradigma DesignByContract, prin definirea invariantului validData(), metodă supraîncărcată cu diferite combinații de parametri, în funcție de metoda în care se face assert-ul la validData. Se folosesc precondiții și postcondiții specifice pentru fiecare metodă relevantă.

3.4.Structuri de date folosite

Ca structuri de date s-au folosit *List* și *ArrayList* pentru a defini liste de produse, cantități, comenzi.

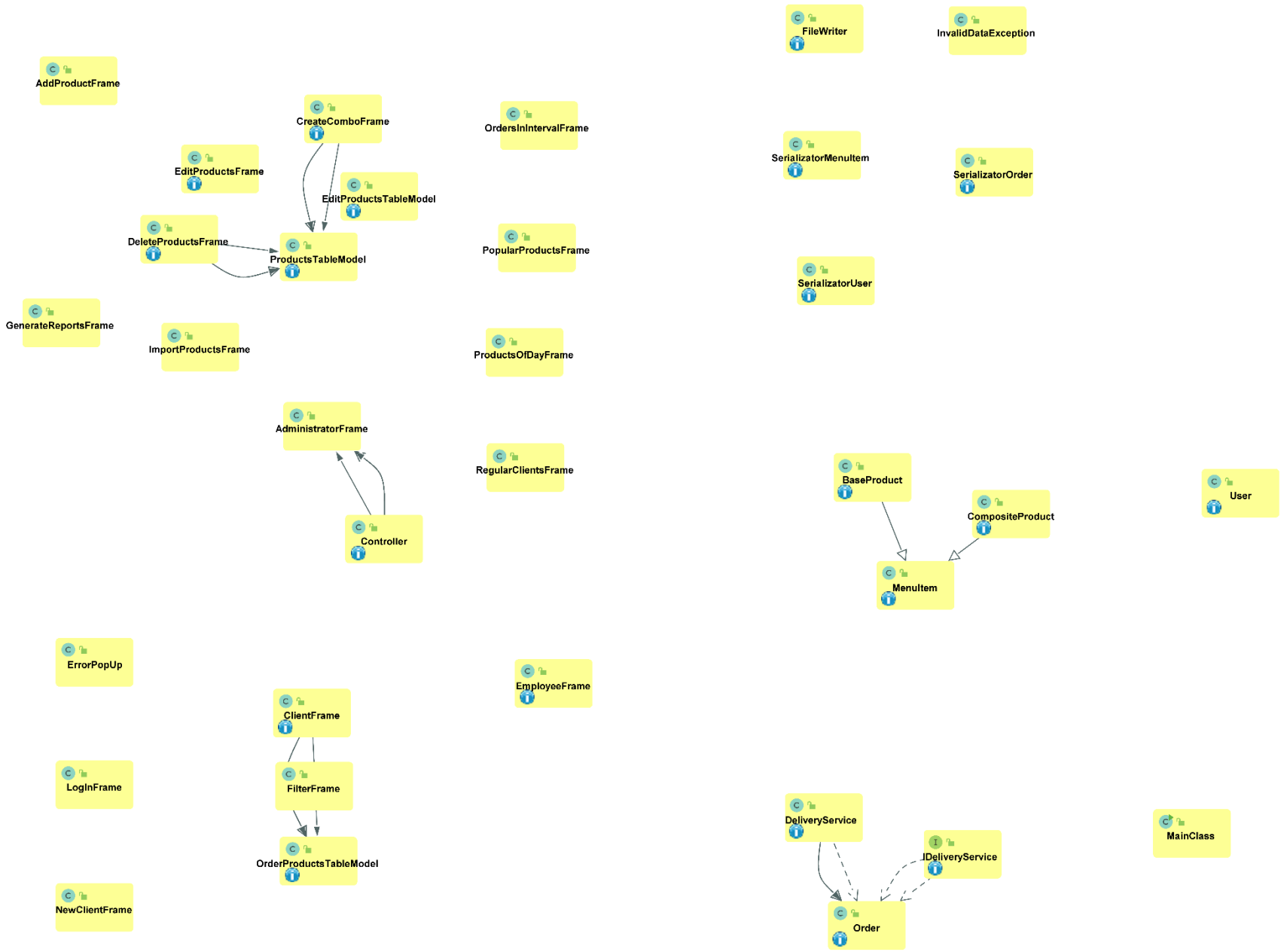
S-au folosit Set și HashSet pentru reținerea de produse și utilizatori.

S-a folosit HashMap pentru reținerea de comenzi (chei) și lista de produse comandate specifică fiecărei comenzi.

S-au folosit Stream-uri pentru extragerea datelor din fișierele .csv, pentru filtrarea produselor de către client și pentru extragerea datelor ce se vor reda în rapoartele generate de administratori.

Clasele *User, MenuItem și Order* fi considerate și ele structuri de date, întrucât acționează ca un tot-unitar în cadrul programului de față.

Diagrama de clase



3.5.Algoritmi folosiți

Cu titulatura de algoritmi se pot considera operațiile pe stream-uri, cele mai folosite dintre acestea fiind *.filter(), .map(), .collect()*.

4. Implementare

S-a folosit conceptul încapsulării din paradigma programării orientate pe obiect, atributele claselor fiind declarate *private* și accesibile claselor exterioare prin metode *getter* și *setter*.

Explicarea detaliată a metodelor din fiecare dintre cele 34 de clase se poate regăsi în documentația JavaDoc (atașată proiectului), care conține și tag-urile pentru invariant, precondiții și postcondiții, ca urmare se vor aminti în continuare doar clasele fiecărui pachet și funcționalitățile acestora:

4.1. Clasele pachetului *business*

Acesta este pachetul în care s-a definit și interfața *IDeliveryService*, explicată anterior, în secțiunea 3.3. Interfețe definite

4.1.1. *DeliveryService:*

Clasa folosita la realizarea anumitor operații caracteristice pentru aplicatia de catering.

4.1.2. *Order*

Clasa pentru modelarea java a unei comenzi.

4.2. Clasele pachetului *data*

4.2.1. *FileWriter*

Clasa pentru importarea de produse si utilizatori.

4.2.2. *SerializatorMenuItem*

Clasa pentru serializarea și deserializarea produselor din meniu.

4.2.3. *SerializatorOrder*

Clasa pentru serializarea și deserializarea comenzilor.

4.2.3. *SerializatorUser*

Clasa pentru serializarea și deserializarea utilizatorilor.

4.3. Clasele pachetului *model.products*

4.3.1. *BaseProduct*

Clasa pentru modelarea java a unui produs de baza.

4.3.2. *CompositeProduct*

Clasa pentru modelarea java a unui produs compus.

4.3.3. *MenuItem*

Clasa pentru modelarea java a unei intrari in meniu.

4.4. Clasele pachetului *model.users*

4.4.1. *User*

Clasa pentru modelarea java a unui utilizator.

4.5. Clasele pachetului *presentation.admin*

4.5.1. *AddProductsFrame*

Clasa pentru crearea interfetei grafice de adaugare a unui nou produs in baza de date.

4.5.2. *AdministratorFrame*

Clasa pentru crearea ferestrei principale a unui admin

4.5.3. *CreateComboFrame*

Clasa pentru definirea modelului general de fereastra de stergere dintr-un tabel din baza de date

4.5.4. *DeleteProductsFrame*

Clasa pentru crearea interfetei grafice de stergere produse

4.5.5. *EditProductsFrame*

Clasa pentru crearea interfetei grafice de editare produse

4.5.6. *EditProductsTableModel*

Clasa pentru crearea modelului abstract de tabel al JTable-ului din interfata grafica de editare produse

4.5.7. *GenerateReportsFrame*

Clasa pentru crearea interfetei grafice de selectare a unui tip de raport care sa fie generat

4.5.8. *ImportProductsFrame*

Clasa pentru crearea ferestrei de importare a unor produse dintr-un fisier .csv

4.5.9. *ProductsTableModel*

Clasa pentru crearea modelului abstract de tabel a unui JTable ineditabil de produse

4.6. Clasele pachetului *presentation.admin.reports*

4.6.1. *OrdersInIntervalFrame*

Clasa pentru crearea ferestrei de generare a unui raport referitor la toate comenzile plasate intr-un anumit interval de timp, indiferent de data; accesibila doar de catre admin

4.6.2. *PopularProductsFrame*

Clasa pentru crearea ferestrei de generare a unui raport al produsleor care au fost comandate de cel putin x ori; accesibila doar de catre admin.

4.6.3. *ProductsOfDayFrame*

Clasa pentru crearea ferestrei de generare a unui raport al tuturor produselor comandate la o anumita data.

4.6.4. *RegularClientsFrame*

Clasa pentru crearea ferestrei de generare a unui raport al tuturor clientilor care au facut cel putin x comenzi, fiecare dintre acestea in valoare de ce putin y ron

4.7. Clasele pachetului *presentation.client*

4.7.1. *ClientFrame*

Clasa pentru crearea ferestrei principale a unui client

4.7.2. *FilterFrame*

Clasa pentru crearea interfetei grafice de filtrare a produselor

4.7.3. *OrderProductsTableModel*

Clasa pentru crearea modelului abstract de tabel al JTable-ului de produse din interfata grafica pentru client

4.8. Clasele pachetului *presentation.controller*

4.8.1. *Controller*

Clasa pentru realizarea conexiunii dintre Model si View

4.9. Clasele pachetului *presentation.employee*

4.9.1. *EmployeeFrame*

Clasa pentru crearea ferestrei unui angajat

4.10. Clasele pachetului *presentation.general*

4.10.1. *ErrorPopUp*

Clasa pentru construirea casetei de dialog ce avertizeaza user-ul ca a introdus un input invalid

4.10.2. *LogInFrame*

Clasa pentru crearea primei ferestre vizualizate de user la deschiderea aplicatiei, fereastra de log-in

4.10.3. *NewClientFrame*

Clasa pentru crearea interfetei grafice de adaugare a unui nou client

4.14. Clasele pachetului *start*

4.14.1. *MainClass*

Clasa principala a programului, singura clasa runnable. Aceasta este metoda care se va rula pentru punerea în funcțiune a programului.

5. Rezultate

În urma definitivării proiectului, s-au realizat o mulțime de teste pentru verificare corectitudini și a consistenței aplicației Java. Mai mult decât atât, în cazul plasării unei comenzi se generează și un fișier .txt confirmare a comenzii, cu rol de factură/bon. Numele fișierului indică a câta comandă plasată în baza de date este cea curentă, considerând, în acest caz, comenzile cu oricâte produse.

Exemplu de fișier de ieșire .txt cu rol de confirmare a comezii/factură:

Order 24

cristianiancu, thank you for your order!

Here is a quick overview of your order from 26/05/2021 13:33:35

Product 1: Cilantro-Tomato Salsa 32.0 RON x 3

Product 2: Ethiopian Spice Tea 49.0 RON x 1

TOTAL: 145.0 RON

Administratorul poate genera 4 tipuri de rapoarte ale activității sistemului de catering:

a) Raport al tuturor comenzilor care au fost plasate într-un anumit interval ordar, indiferent de dată:

Orders placed from 01:00:00 to 03:00:00 regardless of the day:

ORDER ID: 1 ORDER DATE: 23/5/2021 ORDER TIME: 2:20:21
CLIENT ID: 1
PRODUCTS:
1) Cilantro-Tomato Salsa PRICE: 32.0
2) Cilantro-Tomato Salsa PRICE: 32.0
3) Cilantro-Tomato Salsa PRICE: 32.0

TOTAL: 96.0

ORDER ID: 2 ORDER DATE: 23/5/2021 ORDER TIME: 2:20:29
CLIENT ID: 1
PRODUCTS:
1) Fish Stock PRICE: 27.0
2) Fish Stock PRICE: 27.0
3) Fish Stock PRICE: 27.0
4) Fish Stock PRICE: 27.0
5) Quick & Spicy Asian Pickles PRICE: 48.0
6) Quick & Spicy Asian Pickles PRICE: 48.0

TOTAL: 204.0

ORDER ID: 3 ORDER DATE: 23/5/2021 ORDER TIME: 2:20:53
CLIENT ID: 1
PRODUCTS:
1) Fish Stock PRICE: 27.0
2) Fish Stock PRICE: 27.0
3) Smoked Caviar and Hummus on Pita Toasts PRICE: 79.0
4) Smoked Caviar and Hummus on Pita Toasts PRICE: 79.0

TOTAL: 212.0

ORDER ID: 22 ORDER DATE: 26/5/2021 ORDER TIME: 1:12:15
CLIENT ID: 4
PRODUCTS:
1) Quatre-Épices PRICE: 78.0
2) Quatre-Épices PRICE: 78.0
3) Ethiopian Spice Tea PRICE: 49.0

TOTAL: 205.0

ORDER ID: 23 ORDER DATE: 26/5/2021 ORDER TIME: 1:14:57
CLIENT ID: 13
PRODUCTS:
1) Fresh Corn Tortillas PRICE: 79.0
2) Fresh Corn Tortillas PRICE: 79.0

TOTAL: 158.0

REPORT GENERATED: 26/05/2021 01:21:15

b) Raport al tuturor produselor comandate de mai mult de x ori:

Products ordered more than 4 times:

1) Cilantro-Tomato Salsa x 12
2) Fresh Corn Tortillas x 6
3) Quatre-Épices x 5
4) Coriander-Herb Spice Rub x 9
5) Fish Stock x 8

REPORT GENERATED: 26/05/2021 01:21:45

c) Raport al tuturor clienților care au comandat de cel puțin x ori, cu o valoare a fiecărei comenzi de cel puțin y ron

Clients who ordered more than 2 times, with a value of each order higher than 15.0 ron:

1) Client ID: 4 Number of orders:5

REPORT GENERATED: 26/05/2021 01:24:26

d) Raport al tuturor produselor comandate la o anumită dată:

Products ordered on 2021-05-25:

1) Steamed Mussels with Tomato and Chorizo Broth x 2
2) Beef Stock x 3
3) Hot-and-Sour Cabbage Salad x 3
4) Cilantro-Tomato Salsa x 9
5) Fresh Corn Tortillas x 4
6) Toasted Corn Crisps x 2
7) Quatre-Épices x 3
8) Coriander-Herb Spice Rub x 9
9) Ethiopian Spice Tea x 2
10) Fish Stock x 2
11) Quick & Spicy Asian Pickles x 1
12) South American-Style Jícama and Orange Salad x 1
13) Mix random si ciudat x 1
14) Chipotle Tomato Salsa x 1

REPORT GENERATED: 26/05/2021 01:23:12

6. Concluzii

Tema de față a prezentat elemente de noutate în primul rând prin folosirea de stream-uri și definirea de invarianți, precondiții și postcondiții, dar și prin structurile de date folosite (HashSet și HashMap), prin extinderea Observable și implementarea Observer și prin serializare și deserializare

Elementele de grafică au fost, de asemenea, o provocare în realizarea proiectului, în special prin realizarea actualizării instante a tabelelor la intervenția unei modificări, ori la filtrarea produselor.

Alte direcții de dezvoltare ulterioară a proiectului ar putea include:

- îmbunătățirea modului de editare a datelor unui client/produs; să nu mai fie nevoie de selectarea rândurilor ce au fost în prealabil editate, ci la apăsarea butonului Edit să se proceseze toate înregistrările la care s-au efectuat modificări
- îmbunătățire modului de introducere a unei comenzi noi; să nu mai fie nevoie de selectarea rândurilor ce marchează produsele dorite, ci să se ia în calcul automat cele ale căror cantități au fost modificate
- introducerea în interfața grafică, în fereastra de plasare a unei comenzi, în tabelul de produse, a unui element *Spinner* pentru incrementarea sau decrementarea cantității comandate
- adăugarea unor efecte 3D butoanelor ca upgrade al interfeței cu utilizatorul
- adăugarea funcționalității de vizualizare a produselor ce intră în componența unui CompositeProduct

7. Bibliografie

- TP2020-2021_Descriere_Laborator.pdf
- ASSIGNMENT_4_SUPPORT_PRESENTATION.pdf
- Tema_4.pdf
- https://www.infoworld.com/article/2077258/observer-and-observable.html#:~:text=The%20Java%20language%20supports%20the,object%20may%20register%20an%20interest.
- http://www.javapractices.com/topic/TopicAction.do?Id=6
- https://stackoverflow.com/questions/18168257/where-to-add-compiler-options-like-ea-in-intellij-idea
- https://www3.cs.stonybrook.edu/~cse214/Javadoc.htm
- http://users.csc.calpoly.edu/~jdalbey/205/Resources/customtags
- https://www.tutorialspoint.com/java/java_serialization.htm
- https://www.youtube.com/watch?v=t1-YZ6bF-g0&ab_channel=JoeJames
- https://www.w3schools.com/java/java_hashmap.asp
- https://www.w3schools.com/java/java_date.asp