**TEHNICI DE PROGRAMARE FUNDAMENTALE**



**ASSIGNMENT IV:**

**RESTAURANT MANAGEMENT SYSTEM**

**Documentatie**

STUDENT: MARCU MIHAI-ALEXANDRU

GRUPA: 30227

**CUPRINS:**

1. Obiectivul temei;
2. Analiza problemei, modelare, scenarii si cazuri de utilizare;
3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator);
4. Implementare:

Pachete:

* 1. BusinessLayer:

1. AdministratorOperations;
2. WaiterOperations;
3. MenuItem;
4. BaseProduct;
5. CompositeProduct;
6. Restaurant;
7. Order.
   1. DataLayer:
      * 1. FileWriter;
        2. RestaurantSerializator.
   2. PresentationLayer:
8. MainGraphicalUserInterface;
9. MainGUIController;
10. AdministratorGraphicalUserInterface;
11. AdministratorGUIController;
12. WaiterGraphicalUserInterface;
13. WaiterGUIController;
14. ChefGraphicalUserInterface.
    1. Main:
15. MainClass.
16. Concluzii;
17. Bibliografie.
18. Obiectivul temei:

Obiectivul principal al temei este realizarea unei aplicatii care simuleaza un sistem de organizare a unui restaurant.

Obiectivele secundare ale acestei teme sunt:

1. Realizarea unei Java Archive;
2. Utilizarea serializarii pentru a salva datele restaurantului intr-un fisier ca obiecte;
3. Realizarea facturilor comenziilor in format .txt;
4. Realizarea unei interfete grafice atat pentru administrator cat si pentru ospatar.
5. Analiza problemei, modelare, scenarii si cazuri de utilizare:

Utilizatorul va comunica cu aplicatia prin intermediul interfetelor grafice. Pentru inceput va aparea o interfata grafica care va pune utilizatorul sa selecteze pozitia pe care o are in cadrul restaurantului: administrator, ospatar sau chef. Prin apasarea butonului corespunzator functiei sale, se va deschide o alta interfata grafica specifica functiei selectate. Administratorul are posibilitatea de a adauga un produs nou in meniu, de a modifica pretul(pentru produsele de baza), numele sau compozitia(daca este un produs compus) produselor si de a sterge un produs din meniu. Pentru operatiile de adaugare si modificare a produselor administratorul va trebui sa aleaga unul dintre cele doua butoane de tip al produsului: „Base product” sau „Composite product”. Administratorul mai poate si vedea meniul prin apasarea butonului „Show menu”. Dupa operatiile executate administratorul trebuie sa apese butonul „Update menu” pentru a salva modificarile facute meniului. Ospatarul poate realiza operatiile de: creeare a unei comenzi noi: va specifica produsele comandate si masa unde au fost comandate produsele, iar aplicatia ii va atribui comenzii un id specific, calculare a pretului final al unei comenzi: aceasta se va realiza prin introducerea id-ului comenzii, si generare a unei facturi pentru o comanda cu un id precizat. Ospatarul poate vedea si tabelul cu comenzi curente prin apasarea butonului „Show orders”. Pentru a salva comenziile actuale ospatarul trebuie sa apese pe butonul „Update orders table”.

Pentru interfata chef-ului, utilizatorul va trebui sa deschida aceasta fereastra apasand pe butonul „Chef” din interfata principala, apoi va trebui sa deschida interfata ospatarului si sa realizeze creearea unei noi comenzi prin introducerea datelor comenzii si apasarea butonului „Create new order”, iar dupa ce a creeat comanda atunci cand utilizatorul va da click din nou pe interfata chef-ului, un mesaj cu datele comenzii va aparea in aceasta.

Butoanele „Close” din interfete inchid fereastra respectiva, iar butonul „Exit” din interfata principala inchide programul aplicatiei.

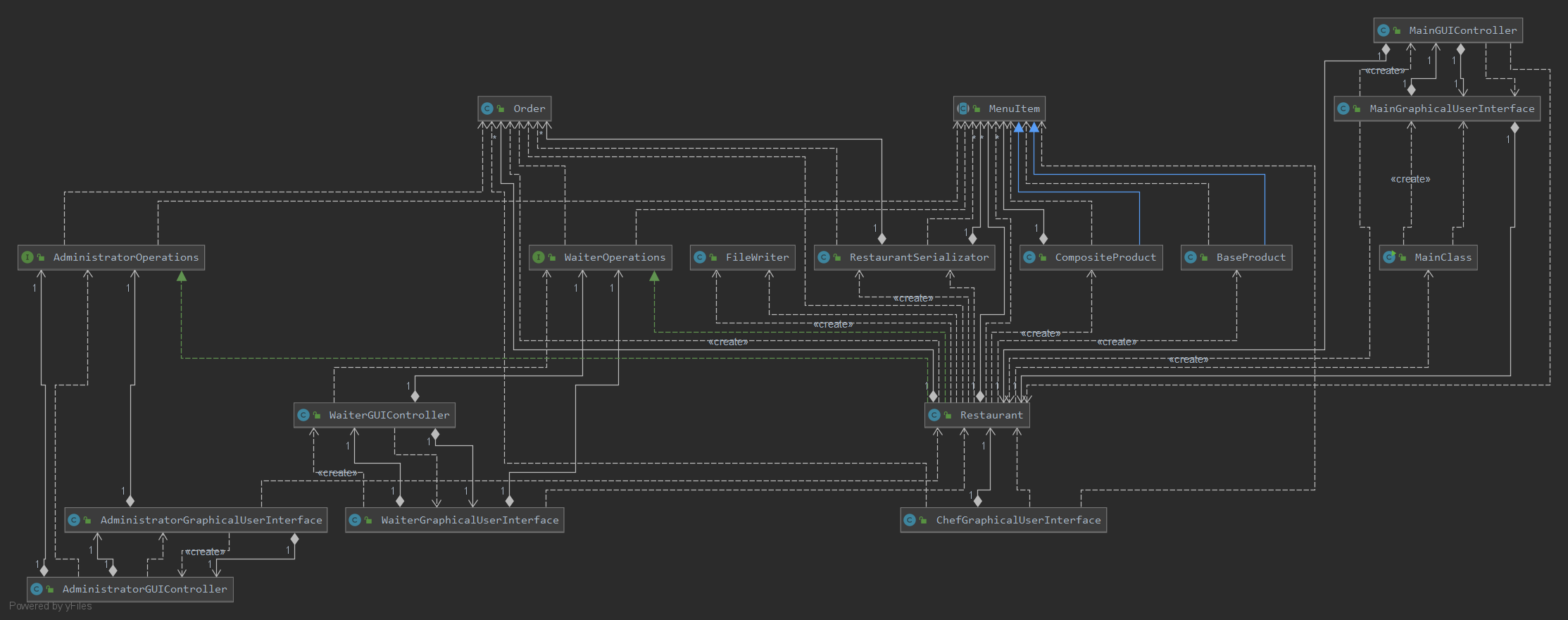
Pentru rularea programului prin intermediul jar-ului trebuie adaugate la comanda java -jar PT2020\_30227\_Marcu\_Mihai-Alexandru\_Assignment\_4.jar restaurant.ser si optiunile –enable-preview.

1. Proiectare( decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, pachete, algoritmi, interfata utilizator):

Proiectul este structurat in 4 pachete:

* 1. BusinessLayer: acest pachet contine clasele si interfetele in care sunt declarate si implementate operatiile care manipuleaza datele restaurantului si clasele care reprezinta tipurile produselor din meniu;
  2. DataLayer: contine clasa in care este implementata metoda de scriere in fisier de tipul .txt a facturilor si clasa in care se regasesc metodele de serializare si deserializare a datelor in si din fisierul dat ca parametru in functia main;
  3. PresentationLayer: contine clasele care reprezinta interfetele grafice pentru fiecare angajat al reestaurantului: administrator, ospatar si chef;
  4. Main: acest pachet contine clasa main care deschide interfata principala.

Diagrama UML a claselor este urmatoarea:



1. Implementare:
2. Pachetul BusinessLayer:
3. Interfata AdministratorOperations: In aceasta interfata sunt declarate metodele care sunt executate de catre administrator;
4. Interfata AdministratorOperations: In aceasta interfata sunt declarate metodele care sunt executate de catre un ospatar;
5. Clasa MenuItem: reprezinta caracteristicile generale ale unui produs: numele si pretul. Pretul va fi calculat in functie de tipul produsului: de baza sau compus. Relatia dintre clasele MenuItem (superclasa), BaseProduct si CompositeProduct (subclase) este una de mostenire.

Metodele acestei clase sunt:

* Un constructor care initializaeaza numele produsului;
* Un getter pentru numele produsului;
* computePrice: este implementata in subclasele acestei clase, in functie de tipul produsului.

1. Clasa BaseProduct: reprezinta un produs de baza, pretul acestuia fiind atribuit prin intermediul constructorului. Metoda computePrice este un getter pentru variabila price;
2. Clasa CompositeProduct: reprezinta un produs compus, care este alcatuit dintr-o lista de produse de baza. Metoda computePrice parcurge lista de produse din care este alcatuit produsul compus, care este reprezentata de un arraylist si aduna preturile acestora, dupa care returneaza pretul total;
3. Clasa Restaurant: reprezinta restaurantul si are ca variabile: orderid care reprezinta numarul pe care il va avea un order drept id cand este creeat de catre un ospatar,items care reprezinta un arraylist care contine produsele din meniul restaurantului si hmap care reprezinta o structura de tipul HashMap care contine toate comenzile impreuna cu listele(arraylist-urile) de produse care le alcatuiesc pe acestea. In aceasta clasa sunt implementate metodele declarate in interfetele specifice unui administrator si unui ospatar.

Metodele acestei clase sunt:

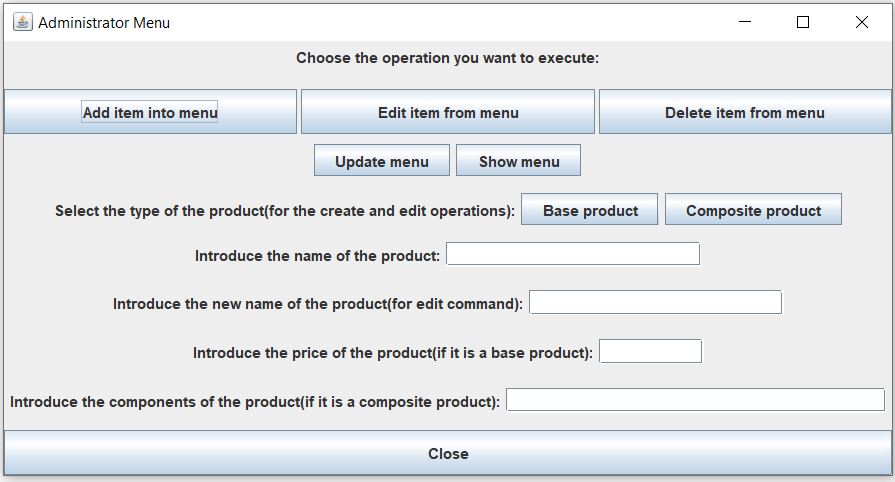
* Constructorul clasei, in acesta se initializeaza un obiect de tipul RestaurantSerializator care preia obiectele din fisierul in care se salveaza datele din restaurant si le atribuie pe acestea variabilelor din clasa;
* createMenuItem: adauga meniului(arraylist-ului) un produs nou care poate fi ori base product ori composite product. Cand se adauga produsul este parcurs si arraylist-ul pentru a face verificarea daca acesta se regaseste deja in meniu sau nu;
* editMenuItem: are scopul de a modifica datele despre un produs din meniu. Daca se modifica pretul unui produs de baza, atunci se va actualiza si pretul produselor compuse care il au pe acesta in compozitie;
* deleteMenuItem: are scopul de a sterge un produs din meniu. Daca produsul sters este unul de baza atunci se vor sterge si produsele compuse care il au pe acesta in compozitie;
* createOrder: aceasta metoda are scopul de a creea o comanda noua si un arraylist care contine produsele ce constituie aceasta comanda. Comanda si arraylist-ul sunt salvate intr-o structura de tipul HashMap;
* computePriceForOrder: aceasta metoda are scopul de a calcula pretul final al unei comenzi;
* generateBill: Are scopul de a genera o factura pentru fiecare comanda curenta care sa contina datele despre aceasta;
* Getter-e pentru items si hmap.

1. Pachetul DataLayer:
2. Clasa FileWriter: in aceasta clasa se realizeaza scrierea in fisier de tipul .txt a facturilor comenzilor. Aceasta operatie se realizeaza cu ajutorul unui PrintWriter;
3. Clasa RestaurantSerializator: in aceasta clasa se regasesc metodele de serializare si deserializare a datelor in si din fisierul dat ca parametru in functia main.

Metodele clasei sunt:

* deserialization: initializeaza un arraylist de obiecte de tipul MenuItem care si un HashMap<Order,Collection<MenuItem>> in care salveaza obiectele cu aceleasi specificatii din fisier. In aceasta metoda se realizaeaza si verificarea daca fisierul exista sau nu. Aceasta metoda foloseste FileInputStream si ObjectInputStream;
* serialization: primeste ca parametri un arraylist de obiecte de tipul MenuItem care si un HashMap<Order,Collection<MenuItem>> pe care le scrie in fisier in aceasta ordine: intai arraylist-ul si apoi map-ul. Aceasta metoda foloseste FileOutputStream si ObjectOutputStream;
* Getter-e pentru arraylist de obiecte de tipul MenuItem si pentru HashMap<Order,Collection<MenuItem>>.

1. Pachetul PresentationLayer:
2. AdministratorGraphicalUserInterface: reprezinta view-ul interfetei administratorului. Aceasta interfata contine: butoane pentru cele trei operatii care pot fi executate de catre administrator, un buton care salveaza datele modificate in fisier: „Update menu”, un buton care afiseaza o fereastra noua care contine un JTable in care sunt afisate produsele meniului si datele lor: ”Show menu”, text field-uri pentru numele, pretul(daca produsul este de baza), lista de componente date prin numele produselor separete prin virgula(pentru produsele compuse) si pentru noul nume al produsului(pentru operatia de modificare a produsului), camp care daca este lasat necompletat atunci numele produsului care este modificat va ramane acelasi.

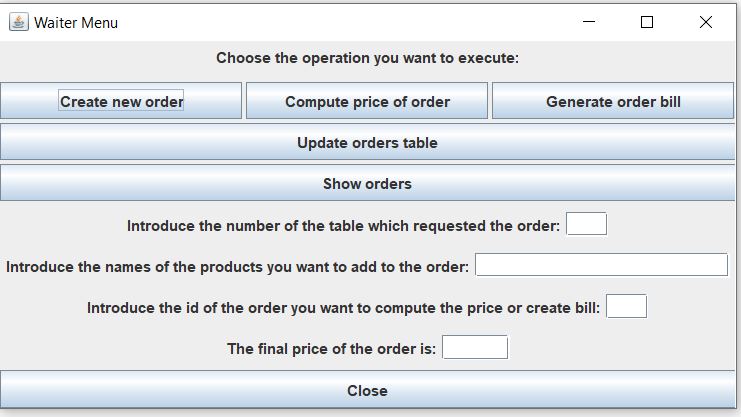


1. AdministratorGUIController: reprezinta controller-ul interfetei administratorului. In aceasta clasa sunt implementate listener-urile pentru butoanele interfetei administrator.

Metodele clasei sunt:

* Un constructor in care se initializeaza modelul de tipul AdministratorOperations si view-ul de tipul AdministratorGraphicalUserInterface si initializeaza in view listener-urile prezentate mai jos:
* Clasa CreateItemListener: prelucreaza datele de intrare introduse de catre utilizator in campurile lor specifice: numele, pretul si lista de componente si verifica starea string-ului type. La final este apelata functia createMenuItem din clasa Restaurant;
* Clasa EditItemListener: prelucreaza datele de intrare introduse de catre utilizator in campurile lor specifice: numele, noul nume, pretul si lista de componente si verifica ce buton a fost apasat pentru tipul produslui si verifica starea string-ului type. Apoi este apelata functia editMenuItem din clasa Restaurant, iar la final se verifica daca produsul era componenta a unui alt produs de tipul compus, in caz ca acesta este, atunci se va actualiza lista de componente a produslui compus cu produsul modificat actualizat si se va apela functia editMenuItem si pentru produsul compus;
* Clasa DeleteItemListener: prelucreaza numele produsului care este introdus de catre utilizator in campul sau specific si apeleaza functia deleteMenuItem din clasa Restaurant. Tot in aceasta metoda se verifica si daca produsul era componenta a unui alt produs de tipul compus, iar in caz ca acesta este, atunci se va apela metoda deleteMenuItem si pentru produsul compus;
* Clasa BaseProductListener: Daca a fost apasat butonul „Base product” atunci string-ul type va deveni ”base” si va semnala faptul ca produsul introdus este unul de baza;
* Clasa CompositeProductListener: Daca a fost apasat butonul „Composite product” atunci string-ul type va deveni ”composite” si va semnala faptul ca produsul introdus este unul compus;
* Clasa ShowMenuListener: in aceasta clasa este prelucrat tabelul care afiseaza produsele din meniu. Daca in meniu nu se afla niciun produs atunci se va afisa mesajul: "The menu is empty!". Aceasta clasa foloseste un obiect de tipul Jtable;
* Clasa UpdateMenuListener: in aceasta clasa se instantiaza un obiect de tipul RestaurantSerializator si prin intermediul acestuia se apeleaza functia serialization pentru a salva modificarile aduse meniului in fisier;
* Clasa ExitBtnListener: are scopul de a inchide interfata atunci cand utilizatorul apasa pe butonul „Close”.

1. WaiterGraphicalUserInterface: reprezinta view-ul interfetei ospatarului. Aceasta interfata contine: butoane pentru cele trei operatii care pot fi executate de catre ospatar, un buton care salveaza comenzile creeate in fisier: „Update orders table”, un buton care afiseaza o fereastra noua care contine un JTable in care sunt afisate produsele meniului si datele lor: ”Show orders”, text field-uri pentru numarul mesei care a solicitat comanda, lista de produse comandate date prin numele produselor separete prin virgula, id-ul comenzii(pentru operatiile de calculare a pretului final si de a genera factura pentru comanda respectiva) si un camp unde va fi afisat pretul final al comenzii.



1. WaiterGUIController: reprezinta controller-ul interfetei ospatarului. In aceasta clasa sunt implementate listener-urile pentru butoanele interfetei ospatarului.

Aceasta clasa contine un constructor in care se initializeaza modelul de tipul WaiterOperations si view-ul de tipul WaiterGraphicalUserInterface si initializeaza in view listener-urile prezentate mai jos:

* Clasa CreateOrderListener: prelucreaza datele de intrare introduse de catre utilizator in campurile lor specifice: numarul mesei care a solicitat comanda si lista de produse comandate si creeaza o comanda nou cu aceste date prin intermediul functiei createOrder din clasa Restaurant;
* Clasa ComputePriceListener: prelucreaza id-ul introdus de catre utilizator si calculeaza pretul final prin metoda computePriceForOrder din clasa Restaurant si il afiseeaza pe acesta in campul specific;
* Clasa GenerateBillListener: prelucreaza id-ul introdus de catre utilizator si genereaza factura cu datele specifice comenzii prin intermediul metodei generateBill din clasa Restaurant;
* Clasa ShowOrdersListener: in aceasta clasa este prelucrat tabelul care afiseaza comenzile curente din restaurant. Daca in restaurant nu exista nicio comanda atunci se va afisa mesajul: "There are no currnet orders!". Aceasta clasa foloseste un obiect de tipul Jtable;
* Clasa UpdateOrdersTableListener: in aceasta clasa se instantiaza un obiect de tipul RestaurantSerializator si prin intermediul acestuia se apeleaza functia serialization pentru a salva modificarile aduse restaurantului in fisier;
* Clasa ExitBtnListener: are scopul de a inchide interfata atunci cand utilizatorul apasa pe butonul „Close”.

1. MainGraphicalUserInterface: Reprezinta view-ul interfetei principale. Aceasta contine patru butoane: primele trei sunt pentru a selecta pozitia pe care o ocupa utilizatorul in restaurant, iar cel de-al patrulea este un buton de exit .



1. MainGUIController: reprezinta controller-ul interfetei principale. In aceasta clasa sunt implementate urmatoarele listener-uri:

* Clasa AdministratorListener: initializeaza un view nou de tipul AdministratorGraphicalUserInterface;
* Clasa WaiterListener: initializeaza un view nou de tipul WaiterGraphicalUserInterface;
* Clasa ChefListener: initializeaza un view nou de tipul ChefGraphicalUserInterface;
* Clasa ExitBtnListener: are scopul de a inchide proiectul atunci cand este apasat butonul „Exit”.

1. ChefGraphicalUserInterface: Reprezinta view-ul interfetei destinate bucatarului. Prin intermediul acestui view chef-ul este notificat atunci cand s-a realizat o comanda noua. Aceasta notificare se face utilizandu-se Observer Design Pattern. Detaliile noilor comenzi sunt afisate prin intermediul unei JtextArea.
2. Pachetul Main: contine clasa main care deschide interfata principala si are o metoda getter pentru numele fisierului in care sunt salvate obiectele restaurantului dat ca si parametru in metoda main.
3. Concluzii:

Dupa realizare acestui proiect, personal mi-am imbunatatit abilitatile Programarii Orientate pe Obiecte. M-am familiarizat cu utilizarea serializarii si mi-am perfectionat abilitatile in creearea interfetelor utilizator grafice.

1. Bibliografie:

* <http://www.tutorialspoint.com/java/java_serialization.htm> ;
* <https://www.baeldung.com/java-serialization> ;
* [https://www.geeksforgeeks.org](https://www.geeksforgeeks.org/serialization-in-java/) ;
* <https://www.journaldev.com/17899/java-simpledateformat-java-date-format>;
* <https://docs.oracle.com/javase/8/docs/api/java/io/FileOutputStream.html>;
* <https://stackoverflow.com/>;
* <https://www.javatpoint.com/java-jscrollpane>;
* <https://www.baeldung.com/java-hashmap>;
* <http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_4/Assignment_4.pdf>;
* <http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_4/Assignment_4_Indications.pdf>.