**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**FOOD DELIVERY MANAGEMENT SYSTEM**

**ASSIGNMENT4**

**Documentație**

**Săsăujan Cristian-Andrei**

**Grupa 30228**

CUPRINS

1. Obiectivul temei.............................................................................................................................. 1

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

3. Proiectare........................................................................................................................................ 5

4. Implementare..................................................................................................................................7

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

6. Concluzii...........................................................................................................................................9

7. Bibliografie.......................................................................................................................................9

1. **Obiectivul Temei**

Obiectivul temei este de a cum functioneaza serializarea, deserializarea, stream-urile, expresiile lambda, fiind ultima tema aceasta le contine pe toate, inclusiv si 3 Layerd Architecture. Cu toate acestea trebuie sa realizam o aplicatie de catering prin care un utilizator poate plasa comenzi, dar si prin care un administrator poate modifica informatii despre produse.

Sub-obiective : Proiectarea (Capitolul 3) unde putem vedea organizarea proiectului, Implementarea (Capitolul 4) unde vom regăsi informații despre clase, Testarea proiectului (Capitolul 5).

1. **Analiza problemei, modelare, scenarii, cazuri de utilizare**

Pentru a realiza aplicatia propusa am inceput cu documentarea despre cum se poate realiza serializarea, apoi urmatorul pas a fost sa analizez expresiile lambda si stream-urile. Dupa ce am inteles toate cele mentionate mai sus am pus problema cum sa incep, am analizat cerinta mi-am schitat cerintele functionale si problemele ce pot aparea pe parcursul implementarii si am inceput sa implementez.

**Cerințe funcționale:**

* Aplicatia noastra ar trebui sa permita clientului sa se inregistreze in aplicatie.
* Aplicatia noastra ar trebui sa permita clientului sa se conecteze in aplicatie.
* Aplicatia noastra ar trebui sa permita clientului sa vada produsele din meniu.
* permita
* number of calories/proteins/fats/sodium/price.
* Aplicatia noastra ar trebui sa permita clientului sa plaseze o comanda.
* Aplicatia noastra ar trebui sa permita administratorului sa se conecteze in aplicatie.
* Aplicatia noastra ar trebui sa permita administratorului sa se inregistreze in aplicatie.
* Aplicatia noastra ar trebui sa permita administratorului sa importe produsele.
* Aplicatia noastra ar trebui sa permita administratorului sa faca operatii precum adauga produs, sterge produs, modifica produs.
* Aplicatia noastra ar trebui sa permita administratorului sa genereze rapoarte :
  + - intervalul de timp al comenzilor,
    - produsele comandate de mai mult de un anumit număr de ori până acum
    - clienții care au comandat mai mult de un anumit număr de ori până acum și

valoarea comenzii a fost mai mare decât o sumă specificată

* + - produsele comandate într-o zi specificată cu numărul de ori pe care le au
    - fost comandat.

**Use Case**: Register

**Actor principal** : Utilizatorul

**Șcenariu de success** :

1 .Utilizatorul isi introduce credentialele.

2 .Utilizatorul apasa pe butonul de Register.

3. Utilizatorul va fi inregistrat si serializat.

**Șcenarii alternative:**

Daca clientul este deja inregistrat in baza de date a aplicatiei, ni se va afisa un mesaj de eroare care ne va zice: “ That user already exists” . In cazul in care nu se completeaza toate campurile ni se va afisa un mesaj in care ne va zice sa le completam pe toate: “Bad user/passoword.”

**Use Case**: Register

**Actor principal** : Employee

**Șcenariu de success** :

1 Angajatul isi introduce credentialele.

2 . Angajatul apasa pe butonul de Register.

3. Angajatul va fi inregistrat si serializat.

**Șcenarii alternative:**

Daca angajatul este deja inregistrat in baza de date a aplicatiei, ni se va afisa un mesaj de eroare care ne va zice: “ That user already exists” . In cazul in care nu se completeaza toate campurile ni se va afisa un mesaj in care ne va zice sa le completam pe toate: “Bad user/passoword.”

**Use Case**: Login

**Actor principal** : Utilizatorul

**Șcenariu de success** :

1 .Utilizatorul isi introduce credentialele.

2 .Utilizatorul apasa pe butonul de Login.

3. Utilizatorul se va loga.

**Șcenarii alternative:**

In cazul in care nu se completeaza toate campurile ni se va afisa un mesaj in care ne va zice sa le completam pe toate: “Bad user/passoword.”

**Use Case**: Login

**Actor principal** : Employee

**Șcenariu de success** :

1 .Angajatul isi introduce credentialele.

2 . Angajatul apasa pe butonul de Login.

3. Angajatul se va loga.

**Șcenarii alternative:**

In cazul in care nu se completeaza toate campurile ni se va afisa un mesaj in care ne va zice sa le completam pe toate: “Bad user/passoword.”

**Use Case**: Login

**Actor principal** : Administrator

**Șcenariu de success** :

1 . Administratorul isi introduce credentialele.

2 . Administratorul apasa pe butonul de Login.

3. Administratorul se va loga.

**Șcenarii alternative:**

In cazul in care nu se completeaza toate campurile ni se va afisa un mesaj in care ne va zice sa le completam pe toate: “Bad user/passoword.”

**Use Case: Operatii pe produse(add product)**

**Actor principal** : Administrator

**Șcenariu de success** :

1 .Administratorul introduce datele corespunzatoare campuriilor Title, Rating, Calories ,Protein, Fats, Sodium, Price

2 . Administratorul apasa pe butonul de Add product

3. Administratorul va putea vedea continutul meniului in caz ca selecteaza View products.

**Use Case: Operatii pe produse(delete product)**

**Actor principal** : Administrator

**Șcenariu de success** :

1 .Administratorul introduce titlul produsului care doreste sa il stearga.

2 Administratorul apasa pe butonul de delete product.

3. Administratorul va putea vedea continutul meniului in caz ca selecteaza View products.

**Use Case: Operatii pe produse(edit product)**

**Actor principal** : Administrator

**Șcenariu de success** :

1 .Administratorul introduce datele corespunzatoare campuriilor Title, Rating, Calories,Protein,Fats, Sodium, Price

2 . Administratorul apasa pe butonul de Edit product

3. Administratorul va putea vedea continutul meniului in caz ca selecteaza View products.

**Use Case: Operatii pe produse(view product)**

**Actor principal** : Administrator

**Șcenariu de success** :

1 . Administratorul apasa pe butonul de View product

**Use Case: Operatii pe produse(import products)**

**Actor principal** : Administrator

**Șcenariu de success** :

1 . Administratorul apasa pe butonul de Import products.

2. Vor fi importate produsele din fisierul csv.

**Use Case: Operatii pe produse(Generate Reports)**

**Actor principal** : Administrator

**Șcenariu de success** :

1 . Administratorul apasa pe butonul de Generate Reports.

2. Acesta va putea introduce datele pentru fiecare raport in parte.

**Use Case: Clientul cauta produse**

**Actor principal** : Utilizatorul

**Șcenariu de success** :

1 . Utilizatorul introduce datele corespunzatoare campuriilor dorite pentru filtrare (Title, Rating, Calories, Protein, Fats, Sodium, Price)

3. Administratorul va putea vedea produsul in caz ca selecteaza View products.

**Use Case: Clientul plaseaza comenzi**

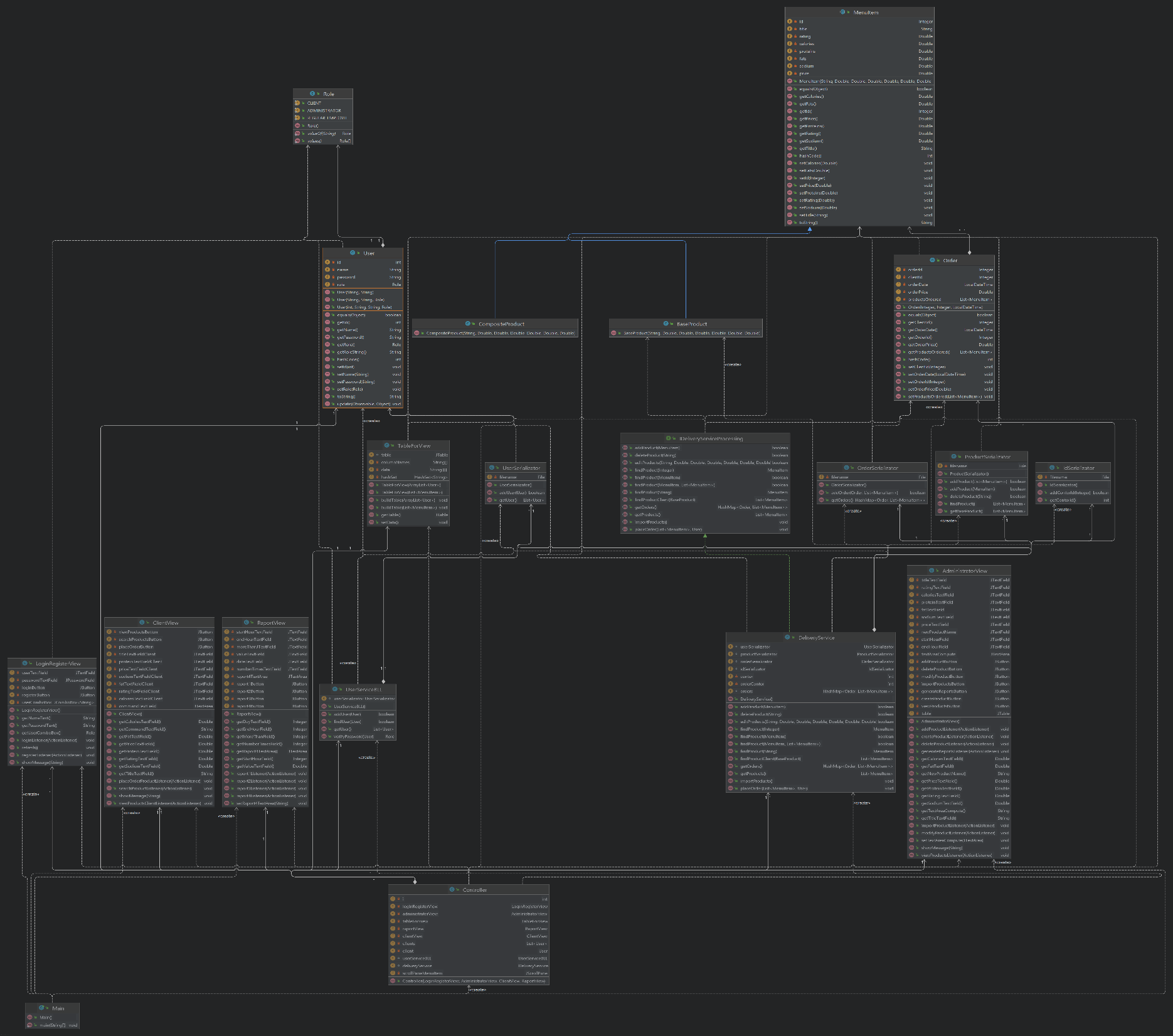
**Actor principal** : Utilizatorul

**Șcenariu de success** :

1 . Utilizatorul introduce numele produsului

2 . Apasa pe place Order.

**3. Proiectare**

Proiectarea aplicatiei a inceput cu package-ul de model, in care se regasesc clasele : BaseProduct, CompositeProduct, MenuItem, Order, Role, User. In clasa user am ca si atribute un id, name, password, role, unde role de tipul Role, care este o enumeratie care contine rolul useriilor, de client, angajat, administrator. Clasa MenuItem este una abstracta, care are ca atribute un id, titlu, rating, calorii, proteine, fats, sodium, price, caracteristicii ale unor produse, aceasta este o clasa abstracta. Clasa BaseProduct extinde clasa MenuItem, asemenea si clasa CompositeProduct. Clasa Order contine ca si atribute, id-ul comenzii, id-ul clientului care a plasat comanda si timpul in care s-a efectuat comanda. Urmatorul pachet a fost cel de prezentare, in care imi este creata interfata grafica care cuprinde clasele AdministratorView, ClientView, Controller-ul, LoginRegisterView, TabelForView. Pachetul urmator este dataAcces care realizeaza serializarea, deci voi avea 3 clase, OrderSerializator, ProductSerializator, UserSerializator, cu ajutor lor voi serializa informatiile din DeliveryService, clasa care face parte din pachetul de businessLogic, pe langa aceasta clasa se mai regasesc alte 2, mai exact o interfata IDeliveryServiceProcessing, si o clasa UserServiceBLL.

**4. Implementare**

Primul pas in implementare a fost sa realizez login-ul, deci in clasa UserServiceBLL, se va gasi interactiune cu dataAcces, mai exact cu serializarea utilizatoriilor. In momentul in care se deschide aplicatia prima fereastra este una de login/register. In cazul in care este un utilizator nou se va inregistra alegandu-si role-ul, in cazul in care acesta deja doreste login-ul fara a fi inregistrat, va fi afisat un pop-ul care ii va comunica ca nu a fost gasit in baza de date si sa se inregistreze. Daca utilizatorul este un client si va da LogIn, i se va deschide o fereastra care numa el ca si client o poate vedea si asupra careia poate efectua operatii de vizionare a produselor disponibile, de cautare dupa filtre a unor produse, de plasarea unei comenzi. In cazul in care credentialele introduse sunt de administrator se va deschide fereastra administratorului care are acces la mai multe comenzi cum ar fi add product, delete product, modify product, create product, vizionare de produse, import de produse, generare de rapoarte.

In clasa deliveryService se realizeaza operatiile cu produse. In cazul in care se doreste de catre administrator adaugarea unui nou produs in deliveryService se regaseste o metoda cu numele addProduct care primeste ca parametru un produs care are atributele create cu ajutorul informatiilor de pe interfata grafica preluate cu gettere. Aici pentru adaugarea unui produs mai intai se obtin toate produsele din serializator intr-o lista de MenuItem’s cu scopul verificarii sa nu se adauge duplicate, acest lucru este posibil tot cu o metoda implementata in clasa deliveryService numite findProduct care primste produsul ca parametru si parcurge toate produsele deja existente si verica cu ajutorul altei metode implementate in clasa MenuItem, metoda equals, care compara obiect cu obiect, daca nu se regaseste un obiect cu aceleasi atribute voi seta ca atribut id-ul produsului si il voi adauga la serializare mai apoi voi returna un boolean true, daca acesta este false voi obtine in controller aceasta informatie si se va afisa pe interfata un mesaj ca acest produs exista deja in serializator. Metoda deleteProduct primeste ca si parametru doar titlul produsului care se doreste sa fie sters, se procedeaza la fel, mai intai se cauta daca exista produsul care doreste sa fie sters, daca nu se gaseste utilizatorul va fi notificat printr-un pop-up ca produsul care doreste sa-l stearga nu exista in serializator, daca exista acesta va fi cautat si va si sters. Metoda editProduct, primeste ca si parametru toate atributele clasei MenuItem, id, titlu, rating, calorii, proteine, fats, sodium, price, se va face cautarea obiectului daca il gaseste ii va modifca atributele altfel utilizatorul va fi notificat ca obiectul care doreste sa fie editat nu exista in serializator. Metoda getProducts se foloseste in momentul in care se doreste de catre utilizator sa vizioneze produsele, atunci metoda ne va returan toate obiectele, si cu ajutorul Clasei TableForView se va crea un Jtable in care se vor putea vedea toate produsele din momentul de fata. Metoda placeOrder primeste ca parametru o lista de menuItems, lista care este preluata de pe interfata, ea reprezinta dorinta clientului in legatura cu comanda lui, aici se va creea order-ul si nota de plata pentru client, totodata aici se va popula si HashMap-ul care va avea ca si cheie un order, iar ca si valoare o lista de menuItems. Importul de produse se face cu ajutorul functiei importProducts care citeste din fisierul csv cu ajutorul buffer-ului BufferedReader si cu row split, care ma va ajuta sa atribui fiecarui baseProduct citit atributele individual.

Pentru fiecare camp care este nevoie sa fie completat pentru anumite functii, daca utilizatorul/administratorul nu il completeaza, acestia vor fi notificati de catre un pop-up care asigura buna functionare a aplicatiei.

Pentru campul de password, pentru a simula cat mai exact o aplicatie de cathering, am folosit un JPasswordField, acesta populand campul de password cu \* \* \* \* \* \* \* \* pe vreme ce utilizatorul isi va introduce parola.

Accesul la date a fost realizat cu ajutorul serializarii. Mai jos se regaseste o explicatie mai clara a serializarii.

Java oferă un mecanism, numit **serializare** obiect, în care un obiect poate fi reprezentat ca o secvență de octeți care include datele obiectului, precum și informații despre tipul obiectului și tipurile de date stocate în obiect.

După ce un obiect serializat a fost scris într-un fișier, acesta poate fi citit din fișier și deserializat, adică informațiile de tip și octeții care reprezintă obiectul și datele acestuia pot fi utilizați pentru a recrea obiectul în memorie.

Cel mai impresionant este faptul că întregul proces este independent de JVM, ceea ce înseamnă că un obiect poate fi serializat pe o platformă și deserializat pe o platformă complet diferită.

**Arhitectura cu trei niveluri** este o arhitectură de aplicații software bine stabilită care organizează aplicațiile în trei niveluri de calcul logice și fizice: nivelul de prezentare sau interfața cu utilizatorul; nivelul aplicației, unde sunt prelucrate datele; și nivelul de date, unde datele asociate cu aplicația sunt stocate și gestionate.

Pentru a genera rapoartele au fost folosite stream-uri si expresii lambda. Introdus în Java 8, Stream API este folosit pentru a procesa colecții de obiecte. Un stream este o secvență de obiecte care acceptă diverse metode care pot fi canalizate pentru a produce rezultatul dorit. Expresiile Lambda ne permit să creăm instanţe ale claselor cu o singură metodă într-un mod mult mai compact. O lambda expresie constă: dintr-o listă de parametri formali, separaţi prin virgulă şi cuprinşi eventual între paranteze rotunde,săgeata direcţională ->,un body ce constă dintr-o expresie sau un bloc de instrucţiuni.

**5. Rezultate**

Am generat un fisier pentru fiecare comanda plasata, un exemplu se poate vedea mai jos

Order with ID = 1 Made by the client with ID = 0  
Food ordered = Menu1,Cream Soup  
Price = 2.0

Rezultatele fiecarei operatii asupra produselor se va putea vedea in JTable prin apasarea butonului de View Products.

**6. Concluzii**

Acest proiect m-a invatat cum sa serializez si sa deserializez informatiile dintr-un fisier, pana acum nu stiam ca se poate face acest lucru stiam doar de notiunea de baze de date pentru pastrarea informatiilor. Totodata un lucru important care l-am invatat a fost utilizarea stream-urilor si a expresiilor lambda, care sunt importante.

**7. Bibliografie**

- - - https://www.ibm.com/cloud/learn/three-tier-architecture

- - - https://www.tutorialspoint.com/java/java\_serialization.htm

- - - https://www.geeksforgeeks.org/serialization-in-java/

- - - https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html#axzz7SzIxXIyJ

- - - https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html

- - - https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#tag

- - - https://javarevisited.blogspot.com/2012/01/what-is-assertion-in-java-java.html#axzz7SzIxXIyJ