**FIS - Test 2 - GIT**

Acestia sunt pasii pe care trebuie sa ii urmati. Punctajul maxim pe care il puteti obtine este 5 puncte. Pentru a putea fi punctati, este obligatoriu ca toate modificarile pe care le-ati facut sa ajunga pe un repository remote public(in acest caz pe *Github*). Pentru a fi punctati, este obligatoriu sa urmati acesti pasi cu atentie si sa dat exact numele specificate branch-urilor, si fisierelor. Pentru un punctaj cat mai ridicat este important sa faceti exact modificarile specificate in acest document. ***Cititi cu atentie FIECARE pas!!!***

1. Descarcati [aceasta](https://www.dropbox.com/s/m1q3mjx6w34p767/reservation-manager.zip?dl=0) arhiva pe calculatorul din laborator si dezarhivati-l in folderul /home/student.
2. Pentru rezolvarea urmatoarelor cerinte puteti folosi terminalul, sau pluginul de GIT din IntelliJ. Pentru manipularea (editarea, stergerea, crearea) fisierelor puteti folosi orice editor de text, inclusiv IntelliJ IDEA.
3. In folderul dezarhivat, initializati un repository de git local.
4. Utilizati comenzile git config user.name si git config user.email pentru a va seta numele si emailul vostru:
   * *git config user.name “Nume Prenume”*
   * *git config user.email “adresa.mea@email.com”*
5. Adaugati toate modificarile din folder si creati un commit cu mesajul “***Initial commit of the project. Contains setup and some model classes***”.
6. Creati-va pe contul de github personal un nou repository PUBLIC si gol (fara niciun fisier), si numiti-l “***reservation-manager***”.
7. Legati (link) repository-ul local cu noul repository remote de pe Github.
8. Adaugati (push) branch-ul master din repository-ul local pe repository-ul remote. **Atentie: nerespectarea acestei cerinte va duce automat la incapacitatea de a puncta pasii 1-7!!**
9. Verificati ca exista codul pe Github.
10. In pasii ce urmeaza veti implementa 2 feature-uri: veti scrie documentatie pentru proiect si veti “curata” codul.
11. Incepeti prin a crea 2 branchuri din branch-ul master, numite “***cleanup***” respectiv “***documentation***”.
12. Treceti pe branch-ul ***documentation***.
13. Creati un nou fisier numit ***README.md*** in folderul ***reservation-manager*** si adaugati urmatorul continut:

***# This is a reservations manager project written in java***

The goal of the project is to make it easy for anyone to access a reservation or an appointment anywhere in town.

***# Contributions***

Feel free to contribute in any way by opening Pull Requests with any new features or bug fixes.

***#Happy Coding!***

1. Creati un commit in care sa adaugati acest fisier si puneti mesajul “***Added README***”.
2. Deschideti fisierul Main.java din folderul src si pachetul *org.loose.fis.reservation.manager* si adaugati urmatoarele comentarii:
   * */\*\**

*\* This is the main class of the reservation manager*

*\* Currently this class contains some dummy users and some dummy reservations.*

*\* Finally, after creating them, the main method of this class prints them out.*

*\**

*\* Feel free to run the main method at any time. :)*

*\*/*

* + */\*\**

*\** ***@param args*** *not used*

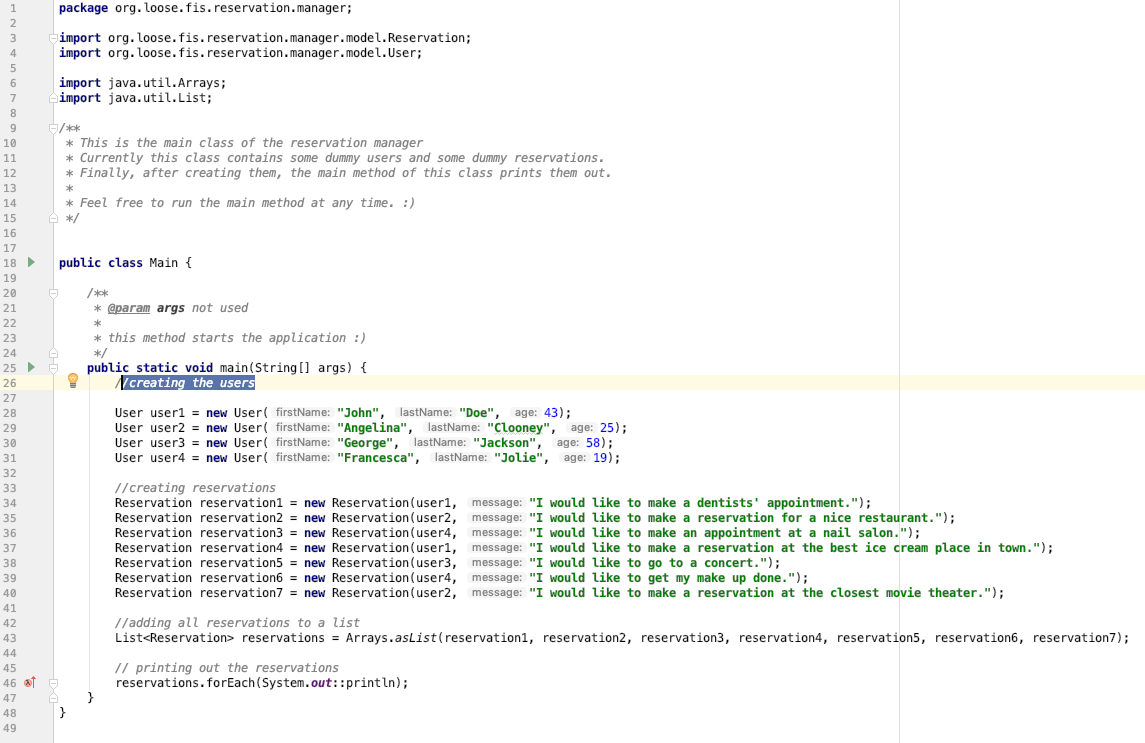
*\**

*\* this method starts the application :)*

*\*/*

* + *//creating the users*
  + *//creating reservations*
  + *//adding all the reservations to a list*
  + *//printing out the reservations*

Dupa adaugare, fisierul Main.java ar trebui sa arate asa:



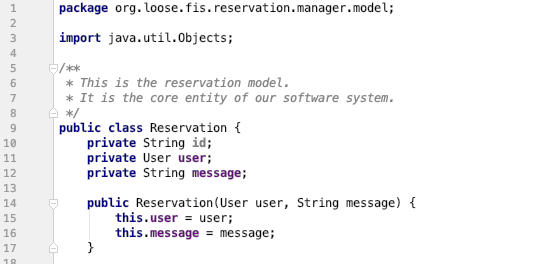
1. Deschideti fisierul ***Reservation.java*** din pachetul *org.loose.fis.reservation.manager.model* si adaugati urmatorul comentariu javadoc deasupra numelui clasei:

*/\*\**

*\* This is the reservation model.*

*\* It is the core entity of our software system.*

*\*/*

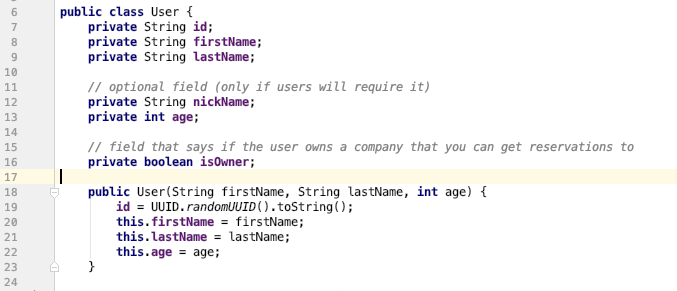


1. Deschideti fisierul User.java din pachetul *org.loose.fis.reservation.manager.model* si adaugati urmatoarele comentariil pentru membrii *nickname* si *isOwner*:
   * *// optional field (only if users will require it)*

***private*** *String* ***nickName****;*

* + *// field that says if the user owns a company that you can get reservations to*

**private boolean isOwner**;



* + Atentie la liniile libere adaugate pe randurile 10 si 14. (Ele ar trebui sa existe!)

1. Creati un commit cu aceste 3 fisiere modificate si puneti ca mesaj “***Added documentation to the most important java classes***”.
2. Creati un nou director numit ***documentation***, la acelasi nivel cu folderul *src*.
3. In directorul ***documentation*** creati fisierul ***LICENSE.txt*** cu urmatorul continut:

Copyright 2019 FIS-Laboratory

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1. Tot in directorul ***documentation*** creati fisierul ***AUTHORS.txt*** cu urmatorul continut:

Authors ordered by first contribution.

John Doe <john.doe@gmail.com>

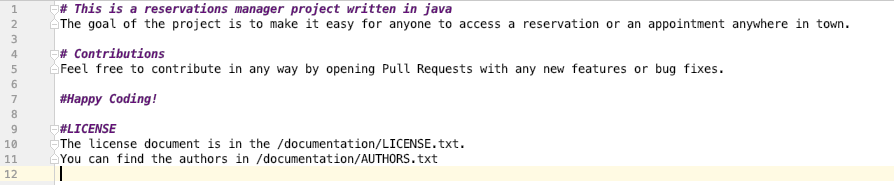
Jane Van der Welt <janewelt@gmail.com>

1. Modificati continutul fisierului README adaugand la final urmatoarele linii:

***#LICENSE***

The license document is in the /documentation/LICENSE.txt.

You can find the authors in /documentation/AUTHORS.txt



* + Atentie la liniile goale pe randurile 8 si 12. (Ele trebuie sa existe!)

1. Creati un commit cu mesajul “***Added license and authors documents***” in care sa adaugati cele 2 fisiere noi, dar si modificarea adusa fisierului README.md
2. Puneti (push) tot codul de pe branch-ul local “documentation” in repositoryul remote de pe github. Dupa aceasta operatie, ar trebui sa aveti 2 branchuri pe github, iar branch-ul documentation ar trebui sa contina toate modificarile aferente pasilor 11-23. **Atentie: nerespectarea acestei cerinte va duce automat la incapacitatea de a puncta pasii 11-23!!**
3. In repository-ul local, treceti pe branchul cleanup.
4. Stergeti urmatoarele fisiere:
   * ***Location.java*** din pachetul *org.loose.fis.reservation.manager.model*.
   * ***DummySevice.java*** din pachetul *org.loose.fis.reservation.manager.service*.
   * Directorul ***services*** din pachetul *org.loose.fis.reservation.manager*.
5. In fisierul User.java stergeti membrul numit nickname, deoarece nu este folosit nicaieri si stergeti si metodele accesor aferente (*getNickName()* si *setNickName()*). Nu lasati mai mult de un rand liber intre metode!!!
6. Creati un commit cu mesajul “***Removed unnecessary files and fields***” in care sa adaugati atat modificarile pentru fisierele sterse, cat si modificarile aduse fisierului User.java.
7. Inlocuiti continutul fisierului Main.java cu urmatorul continut:

**package** org.loose.fis.reservation.manager;

**import** org.loose.fis.reservation.manager.model.Reservation;

**import** org.loose.fis.reservation.manager.model.User;

**import** java.util.Arrays;

**import** java.util.List;

**public class** Main {

**public static void** main(String[] args) {

User user1 = **new** User(**"John"**, **"Doe"**, 43);

User user2 = **new** User(**"Angelina"**, **"Clooney"**, 25);

User user3 = **new** User(**"George"**, **"Jackson"**, 58);

User user4 = **new** User(**"Francesca"**, **"Jolie"**, 19);

List<Reservation> reservations = *createReservations*(user1, user2, user3, user4);

reservations.forEach(System.***out***::println);

}

**private static** List<Reservation> createReservations(User user1, User user2, User user3, User user4) {

Reservation reservation1 = **new** Reservation(user1, **"I would like to make a dentists' appointment."**);

Reservation reservation2 = **new** Reservation(user2, **"I would like to make a reservation for a nice restaurant."**);

Reservation reservation3 = **new** Reservation(user4, **"I would like to make an appointment at a nail salon."**);

Reservation reservation4 = **new** Reservation(user1, **"I would like to make a reservation at the best ice cream place in town."**);

Reservation reservation5 = **new** Reservation(user3, **"I would like to go to a concert."**);

Reservation reservation6 = **new** Reservation(user4, **"I would like to get my make up done."**);

Reservation reservation7 = **new** Reservation(user2, **"I would like to make a reservation at the closest movie theater."**);

**return** Arrays.*asList*(reservation1, reservation2, reservation3, reservation4, reservation5, reservation6, reservation7);

}

}

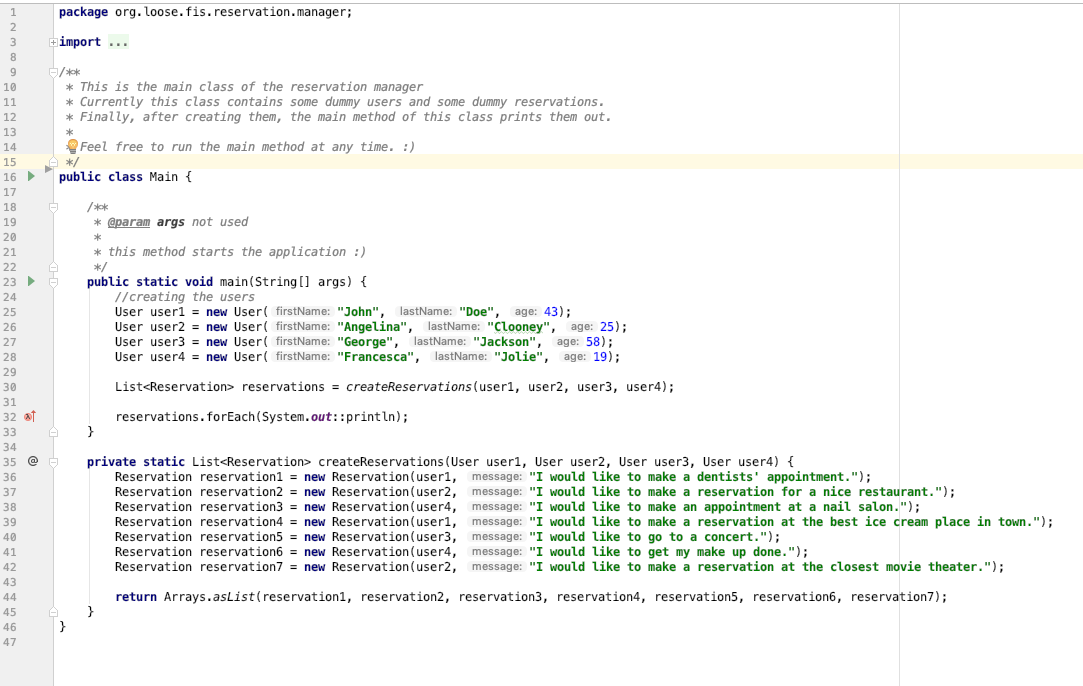
1. Adaugati un nou fisier numit *cleanup.log* in folderul *reservation-manager* si puneti urmatorul continut:

delete Location class

delete DummyService class

delete service package

refactor Main

1. Adaugati atat modificarea adusa fisierului Main.java cat si fisierul cleanup.log la un commit cu mesajul “***Refactored Main class and created cleanup log file***”.
2. Puneti (push) tot codul de pe branch-ul local “cleanup” in repositoryul remote de pe github. Dupa aceasta operatie, ar trebui sa aveti 3 branchuri pe github, iar branch-ul *cleanup* ar trebui sa contina toate modificarile aferente pasilor 24-30. **Atentie: nerespectarea acestei cerinte va duce automat la incapacitatea de a puncta pasii 25-31!**
3. In repository-ul local, treceti pe branchul master.
4. Aduceti toate modificarile (merge) din branch-ul local *documentation* in branch-ul local *master*.
5. Faceti push branch-ului local master pe repository-ul remote de pe Github. **Atentie: nerespectarea acestei cerinte va duce automat la incapacitatea de a puncta pasii 33-34!**
6. In repository-ul local, treceti pe branchul *cleanup*.
7. Aduceti toate modificarile (merge) din master in branch-ul cleanup.
8. Veti avea de rezolvat niste conflicte pentru fisierele ***Main.java*** si ***User.java***.
9. Dupa rezolvarea conflictelor, clasa Main ar trebui sa arate in felul urmator:
10. Dupa rezolvarea conflictelor, clasa ***User.java*** ar trebui sa arate in felul urmator:

1. Daca ati decis sa faceti merge-ul din linia de comanda, va fi nevoie sa creati un nou commit cu toate modificarile, cu urmatorul mesaj “***Merge branch ‘master’ into cleanup***”
2. Faceti push la noile modificari de pe branch-ul cleanup pe repository-ul remote. **Atentie: nerespectarea acestei cerinte va duce automat la incapacitatea de a puncta pasii 36-41!**
3. Pe repository-ul local, treceti pe branch-ul master.
4. Aduceti toate modificarile (merge) din branch-ul cleanup.
5. Faceti push la noile modificari de pe branch-ul master pe repository-ul remote. **Atentie: nerespectarea acestei cerinte va duce automat la incapacitatea de a puncta pasii 43-44!**
6. Completati [urmatorul formular](https://forms.gle/PHiwxM23YsjxVUhXA) cu datele personale si **NU** apasati butonul de submit.
7. Pregatiti actul de identitate si anuntati profesorul supraveghetor ca sunteti pregatiti sa trimiteti rezultatul.
8. Dupa aprobarea profesorului supraveghetor, trimiteti formularul.