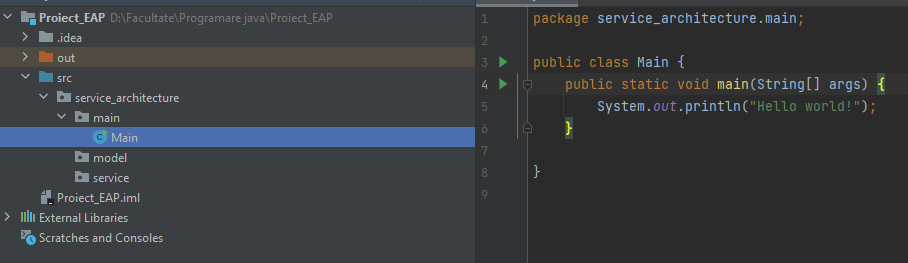
# PAO Project – E-ticketing platform

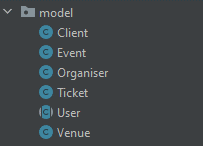
Popescu Mihnea – 264

# Initierea obiectelor

Pentru inceput, am creat package-ul service\_architecture

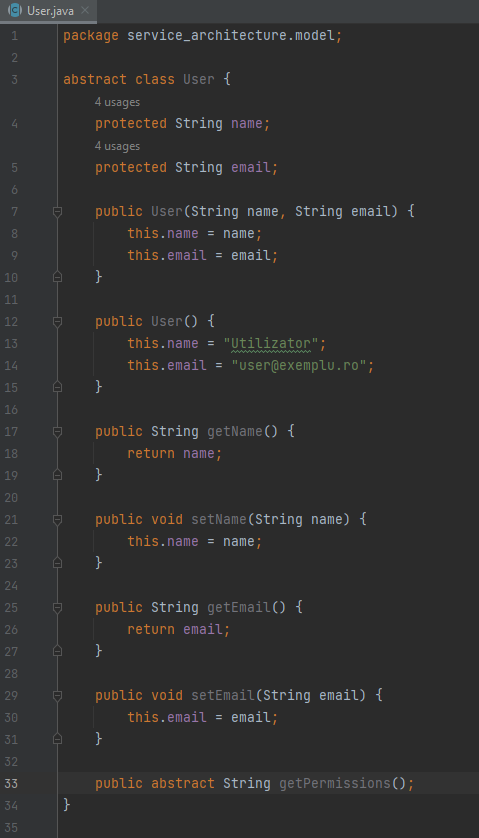


Am adaugat si subpackage-urile main (ce contine clasa principala in care se apeleaza metodele serviciilor), model (in care vom adauga clasele obiectelor) si service (in care avem operatiile de serviciu).

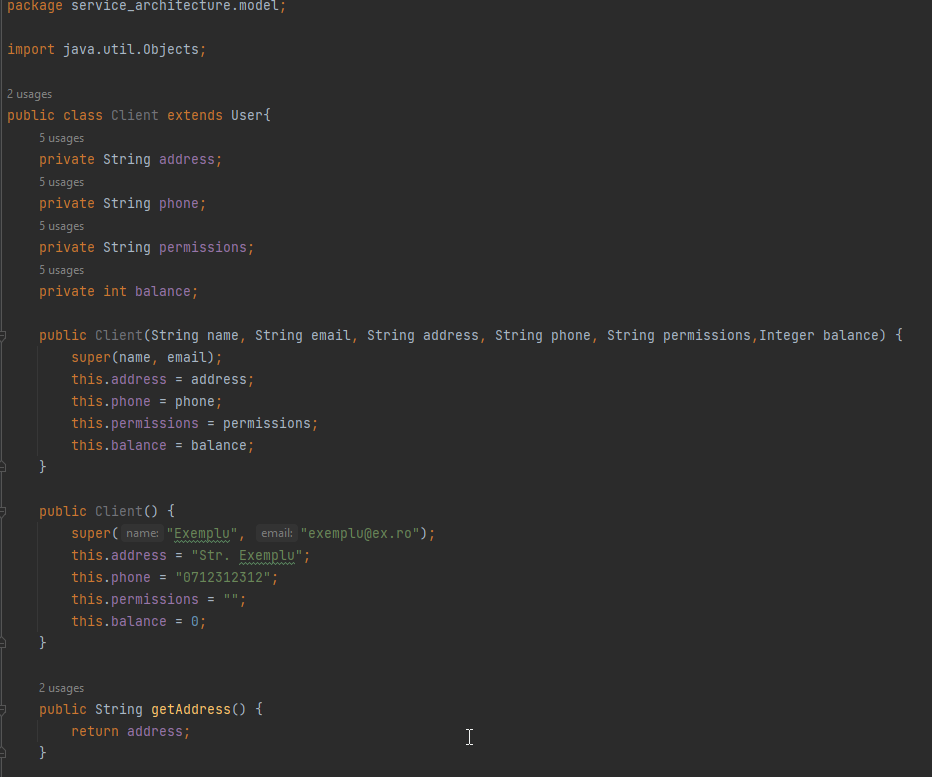


Modelele folosite in implementarea platformei de e-ticketing sunt urmatoarele:

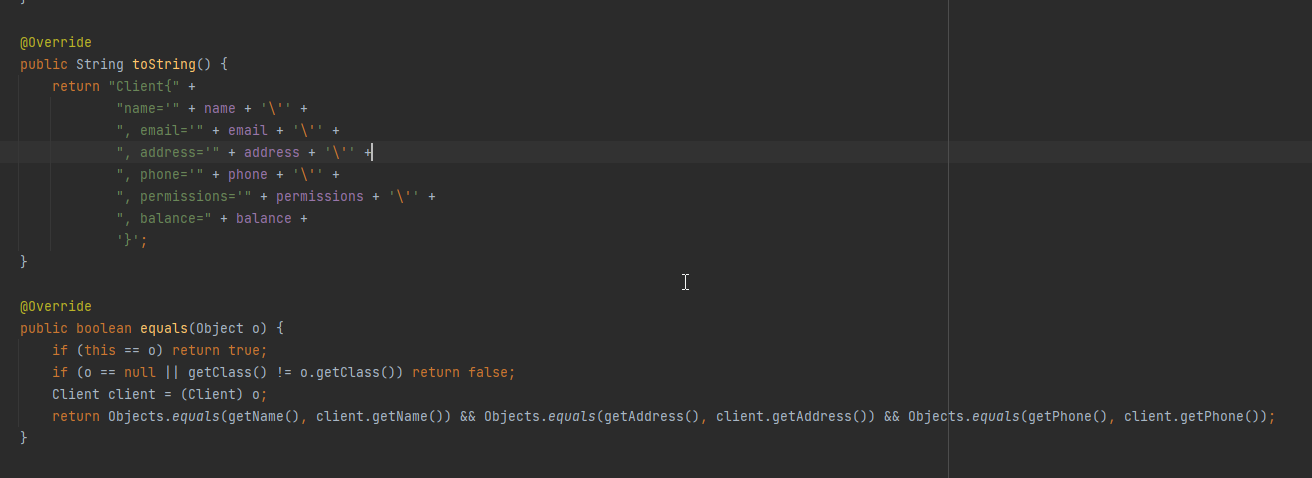
* User (clasa de baza a unui utilizator)
* Client (mosteneste clasa User si reprezinta un utilizator care poate cumpara bilete)
* Organiser (mosteneste clasa User si reprezinta un utilizator care poate organiza evenimente)
* Venue (reprezinta o locatie unde poate avea loc un eveniment si contine detalii precum nr de locuri)
* Event (reprezinta un eveniment, acestuia trebuie sa-i fie atribuit un Organiser si un Venue)
* Ticket (reprezinta un bilet de acces la un eveniment)



Am inceput prin crearea clasei abstracte User, ce defineste un utilizator de baza al platformei (un cont). Aceasta clasa contine si metoda abstracta getPermissions, ce va fi folosite de clasele ce o vor mosteni.



Am adaugat clasei Client atributele address, phone, permissions si balance.



Am facut suprascriere la metodele toString si equals pentru a include atributele speciale. 2 instante ale clasei Client pot fi egale daca au acelasi nume, aceeasi adresa si acelasi numar de telefon.

Asemanator am creat si celelalte clase: Organiser, Ticket, Event si Venue.

Am adaugat in clasa User un ArrayList de tipul <Ticket>

protected ArrayList<Ticket> tickets;

# Citirea si stocarea datelor

In package-ul service\_architecture.service am creat un nou package: fileio.

In el am definit clasa singleton VenueReader unde am efectuat citirea din fisierul storage/venue\_list.csv

public class EventReader {  
 private static EventReader *instance*;  
 private EventReader(){}  
  
 public static EventReader getInstance() {  
 if(*instance* == null) {  
 *instance* = new EventReader();  
 }  
 return *instance*;  
 }  
  
 private static String *line* = "";  
  
 public static ArrayList<Event> readFile(String basePath) {  
 ArrayList<Event> events = new ArrayList<Event>();  
 try(BufferedReader br = new BufferedReader(new FileReader(basePath+"event\_list.csv"))) {  
 while((*line* = br.readLine()) != null) {  
 String[] values = *line*.split(",");  
 if(values.length > 0) {  
 Event event = new Event();  
 event.setOrganiser(Integer.*parseInt*(values[0]));  
 event.setName(values[1]);  
 event.setDate(values[2]);  
 event.setDescription(values[3]);  
 event.setTicket\_price(Integer.*parseInt*(values[4]));  
 event.setVenue(Integer.*parseInt*(values[5]));  
 events.add(event);  
 }  
 }  
 } catch(IOException e) {  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return events;  
 }  
}

Asemanator am facut operatiile de citire si pentru organisers, events and clients si tickets.

Am creat o noua clasa singleton in package-ul service\_architecture.service.fileio numita GetCSVData in care am adaugat metoda getCSVInputData ce returneaza un Map ce contine ArrayList-uri cu toate obiectele salvate in fisierele CSV.

public static Map<String, ArrayList> getCSVInputData(String basePath) {  
 Map map = new HashMap<String, ArrayList>();  
  
 VenueReader vr = VenueReader.*getInstance*();  
 map.put("venues",vr.*readFile*(basePath));  
  
 OrganiserReader or = OrganiserReader.*getInstance*();  
 map.put("organisers", or.*readFile*(basePath));  
  
 EventReader er = EventReader.*getInstance*();  
 map.put("events", er.*readFile*(basePath));  
  
 ClientReader cr = ClientReader.*getInstance*();  
 map.put("clients", cr.*readFile*(basePath));  
  
 TicketReader tr = TicketReader.*getInstance*();  
 map.put("tickets", tr.*readFile*(basePath));  
  
 return map;  
}

Iar in clasa service\_packages.main.Main am adaugat atributele clients, organisers, venues si events, ce reprezinta liste unice ale instantelor acestor obiecte ce vor putea fi prelucrate de aplicatie.

private String basePath = "D:\\facultate\\an2\\EAP\\Proiect\_EAP\\src\\storage\\";  
  
private ArrayList<Client> clients;  
private ArrayList<Organiser> organisers;  
private ArrayList<Venue> venues;  
  
private ArrayList<Event> events;  
  
private void getData() {  
 String basePath = this.basePath;  
  
 //inputs  
 GetCSVData singleton = GetCSVData.*getInstance*();  
  
 Map<String, ArrayList> map = singleton.*getCSVInputData*(basePath);  
  
 if(map.containsKey("venues")) {  
 this.setVenues(map.get("venues"));  
 }  
 System.*out*.println("Found " + this.getVenues().size() +" venues.");  
  
 if(map.containsKey("organisers")) {  
 this.setOrganisers(map.get("organisers"));  
 }  
 System.*out*.println("Found " + this.getOrganisers().size() + " organisers.");  
  
 if(map.containsKey("events")) {  
 this.setEvents(map.get("events"));  
 }  
 System.*out*.println("Found "+ this.getEvents().size() + " events.");  
  
 if(map.containsKey("clients")) {  
 this.setClients(map.get("clients"));  
 }  
 System.*out*.println("Found " + this.getClients().size() + " clients.");  
  
 if(map.containsKey("tickets")) {  
 ArrayList<Ticket> tickets = map.get("tickets");  
  
 for(Client client : this.getClients()) {  
 ArrayList<Ticket> clientTickets = new ArrayList<Ticket>();  
 for(Ticket ticket : tickets) {  
 if(ticket.getUser() == client.hashCode()) {  
 clientTickets.add(ticket);  
 }  
 }  
 client.setTickets(clientTickets);  
 }  
  
 for(Organiser organiser : this.getOrganisers()) {  
 ArrayList<Ticket> organiserTickets = new ArrayList<Ticket>();  
 for(Ticket ticket : tickets) {  
 if(ticket.getUser() == organiser.hashCode()) {  
 organiserTickets.add(ticket);  
 }  
 }  
 organiser.setTickets(organiserTickets);  
 }  
  
 System.*out*.println("Found " + tickets.size() + " tickets.");  
 }  
}

In cazul obiectelor Tickets, le-am prelucrat pe fiecare in parte si le-am adaugat in ArrayList-urile User-ilor respective.