Yıldız Teknik Üniversitesi

Elektrik-Elektronik Fakültesi

Bilgisayar Mühendisliği Bölümü

BLM2012

Nesneye Yönelik Programlama

Mehmet Sıddık Aktaş

Dönem Projesi

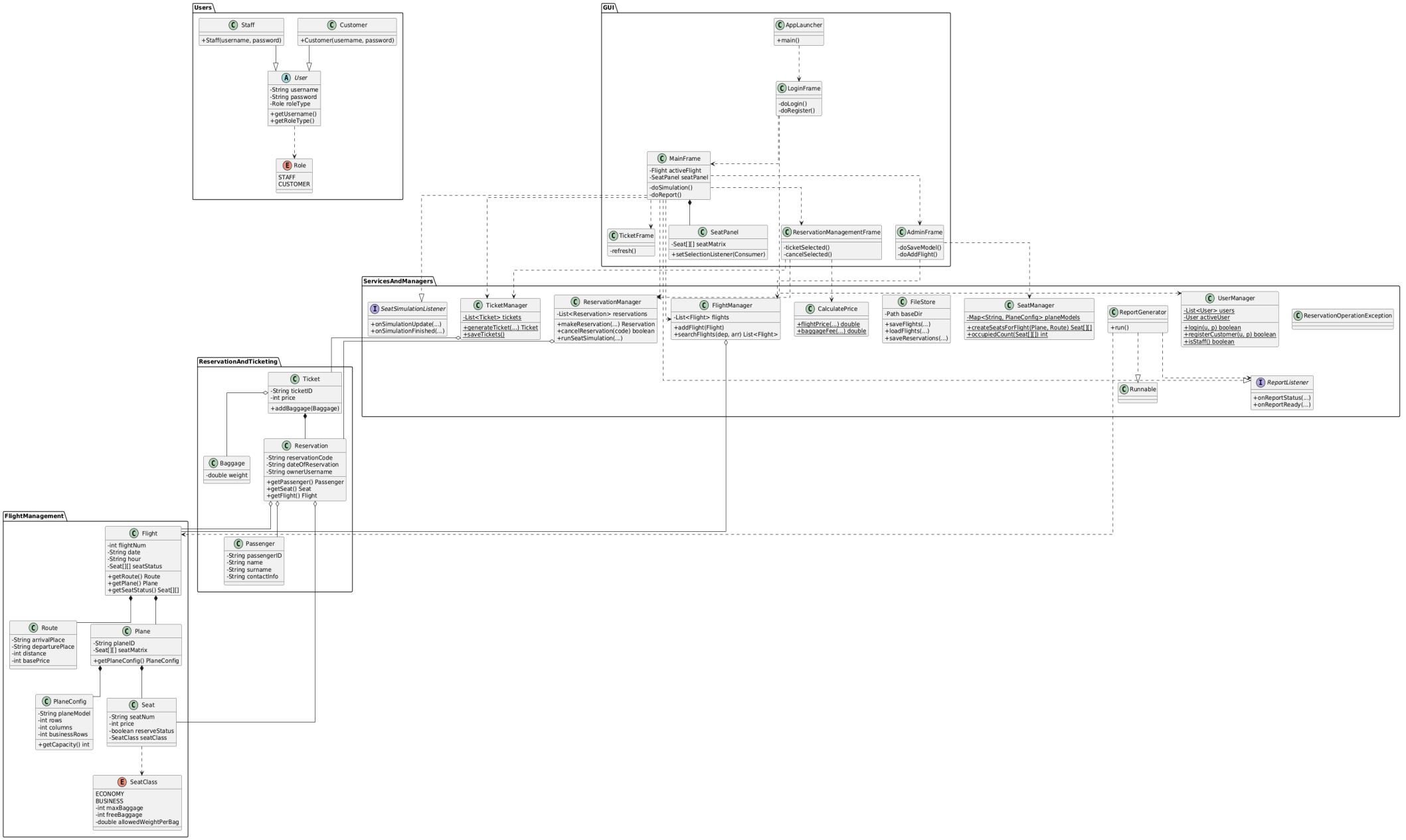
İsim: Veyis Ceylan
Ömer Korkmaz

No: 23011093
23011061

E-posta: **veyis.ceylan@std.yildiz.edu.tr**
omer.korkmaz@std.yildiz.edu.tr

İÇİNDEKİLER 2

Video Linki:


Users
Staff
+Staff(username, password)
Customer
+Customer(username, password)
User
-String username
-String password
-Role roleType
+getUsername()
+getRoleType()
Role
STAFF
CUSTOMER
GUI
AppLauncher
+main()
LoginFrame
-doLogin()
-doRegister()
MainFrame
-Flight activeFlight
-SeatPanel seatPanel
-doSimulation()
-doReport()
TicketFrame
-refresh()
SeatPanel
-Seat[][] seatMatrix
+setSelectionListener(Consumer)
ReservationManagementFrame
-ticketSelected()
-cancelSelected()
AdminFrame
-doSaveModel()
-doAddFlight()
ServicesAndManagers
SeatSimulationListener
+onSimulationUpdate(...)
+onSimulationFinished(...)
TicketManager
-List<Ticket> tickets
+generateTicket(...) Ticket
+saveTickets()
ReservationManager
-List<Reservation> reservations
+makeReservation(...) Reservation
+cancelReservation(code) boolean
+runSeatSimulation(...)
FlightManager
-List<Flight> flights
+addFlight(Flight)
+searchFlights(dep, arr) List<Flight>
CalculatePrice
+flightPrice(...) double
+baggageFee(...) double
FileStore
-Path baseDir
+saveFlights(...)
+loadFlights(...)
+saveReservations(...)
SeatManager
-Map<String, PlaneConfig> planeModels
+createSeatsForFlight(Plane, Route) Seat[][]
+occupiedCount(Seat[][]) int
ReportGenerator
+run()
UserManager
-List<User> users
-User activeUser
+login(u, p) boolean
+registerCustomer(u, p) boolean
+isStaff() boolean
ReservationOperationException
Runnable
ReportListener
+onReportStatus(...)
+onReportReady(...)
ReservationAndTicketing
Ticket
-String ticketID
-int price
+addBaggage(Baggage)
Reservation
-String reservationCode
-String dateOfReservation
-String ownerUsername
+getPassenger() Passenger
+getSeat() Seat
+getFlight() Flight
Baggage
-double weight
Passenger
-String passengerID
-String name
-String surname
-String contactInfo
FlightManagement
Flight
-int flightNum
-String date
-String hour
-Seat[][] seatStatus
+getRoute() Route
+getPlane() Plane
+getSeatStatus() Seat[][]
Route
-String arrivalPlace
-String departurePlace
-int distance
-int basePrice
Plane
-String planeID
-Seat[][] seatMatrix
+getPlaneConfig() PlaneConfig
PlaneConfig
-String planeModel
-int rows
-int columns
-int businessRows
+getCapacity() int
Seat
-String seatNum
-int price
-boolean reserveStatus
-SeatClass seatClass
SeatClass
ECONOMY
BUSINESS
-int maxBaggage
-int freeBaggage
-double allowedWeightPerBag

2. Flight Management

Flight

Flight sınıfı, bir uçuşu temsil etmek amacıyla tasarlanmıştır. Bu sınıf; uçuş numarası, tarih, saat, uçuş süresi, rota ve uçak bilgilerini içerir. Flight sınıfı, Plane ve Route sınıfları ile *composition* ilişkisi içindedir. Ayrıca rezervasyon işlemleri sırasında Reservation sınıfı tarafından kullanılmaktadır.

Attributes

- **route : Route —** Uçuş rotası
- **plane : Plane —** Uçuş uçağı
- **flightNum : int —** Uçuş numarası
- **date : String —** Uçuş tarihi
- **hour : String —** Uçuş saati
- **duration : int —** Uçuş süresi
- **seatStatus : Seat[][] —** Koltuk matrisi

Constructors

- **Flight(Route route, Plane plane, int flightNum, String date, String hour, int duration):** Uçuşu oluşturur ve koltuk matrisini sistem mantığına uygun şekilde hazırlar.

Methods

- **toString():** Uçuşun GUI'de/raporda okunabilir gösterimini üretir.

Plane

Plane sınıfı, bir uçağın fiziksel yapısını temsil eder. Uçağa ait koltuk düzeni ve konfigürasyon bilgileri bu sınıfta tutulur. Plane sınıfı, PlaneConfig sınıfı ile *composition* ilişkisine sahiptir. Her uçuş yalnızca bir uçağa bağlıdır.

Attributes

- **planeID : String —** Uçağın benzersiz kimliği
- **config : PlaneConfig —** Koltuk düzeni konfigürasyonu
- **seatMatrix : Seat[][] —** Koltuk düzenine göre koltukları tutar

Constructors

- **Plane(String planeID, PlaneConfig config):** Uçak nesnesi oluşturur.

PlaneConfig

PlaneConfig sınıfı, uçağın koltuk düzenini tanımlamak için kullanılır. Bu sınıf; toplam satır sayısı, koltuk sayısı ve business sınıfı satırlarını içerir. Plane sınıfı tarafından kullanılarak koltukların otomatik oluşturulmasını sağlar. Bu yapı sayesinde uçak konfigürasyonu dinamik olarak değiştirilebilir.

Attributes

- **planeModel : String —** Uçak modeli
- **rows : int —** Satır sayısı
- **columns : int —** Sütun sayısı (örn 6: A–F)
- **businessRows : int —** Business satır sayısı

Constructors

- **PlaneConfig(String PlaneModel, int rows, int columns, int businessRows):** Uçak konfigürasyonunu tanımlar.

Route

Route sınıfı, bir uçuşun kalkış ve varış bilgilerini temsil eder. Bu sınıf; kalkış noktası, varış noktası, mesafe ve temel fiyat bilgisini içerir. Route sınıfı, bilet fiyatı hesaplamalarında CalculatePrice sınıfı tarafından kullanılmaktadır. Her Flight nesnesi yalnızca bir Route nesnesine sahiptir.

Attributes

- **departurePlace : String —** Kalkış şehri/kodu
- **arrivalPlace : String —** Varış şehri/kodu
- **distance : int —** Mesafe (km vb.)
- **basePrice : int —** Temel fiyat katsayısı

Constructors

- **Route(String departurePlace, String arrivalPlace, int distance, int basePrice):** Rota bilgilerini oluşturur.

Seat

Seat sınıfı, uçak içerisindeki tekil bir koltuğu temsil eder. Koltuk numarası, koltuk sınıfı (ECONOMY veya BUSINESS) ve doluluk durumu bu sınıfta tutulur. Seat sınıfı, Reservation sınıfı ile ilişkilidir ve bir koltuk yalnızca bir rezervasyona atanabilir. Bu sınıf, koltuk seçimi ve doluluk takibinde temel rol oynar.

Attributes

- **seatNum : String**
  Koltuğun benzersiz numarasını tutar.
- **price : int**
  Bu koltuk için belirlenmiş bilet fiyatını tutar. Koltuk sınıfı ve rota bilgisine göre hesaplanan fiyat bu alana atanır.
- **reserveStatus : boolean**
  Koltuğun rezerve edilip edilmediğini belirtir.
- **seatClass : SeatClass**
  Koltuğun ait olduğu sınıfı belirtir (ECONOMY veya BUSINESS).

Enum Attributes

- **freeBaggage : int**
  Ücretsiz taşınabilecek bagaj adedini belirtir.
- **allowedWeightPerBag : double**
  Her bir bagaj için izin verilen maksimum ağırlığı belirtir.
- **maxBaggage : int**
  Yolcunun taşıyabileceği maksimum bagaj sayısını belirtir (sabit değer).

Enum Constructor

- **SeatClass(int freeBaggage, double weight):** Her koltuk sınıfı için ücretsiz bagaj sayısını ve bagaj başına izin verilen ağırlığı tanımlar.

Enum Methods

- **getFreeBaggage():** Koltuk sınıfına ait ücretsiz bagaj hakkını döndürür.
- **getAllowedWeightPerBag():** Koltuk sınıfına göre her bir bagaj için izin verilen ağırlığı döndürür.
- **getMaxBaggage():** Taşınabilecek maksimum bagaj sayısını döndürür.

Constructor

- **Seat(String seatNum, SeatClass seatClass, int price, boolean reserveStatus):** Bir koltuk nesnesi oluşturur.

3. Reservation And Ticketing

Passenger

Passenger sınıfı, uçuşa katılan yolcu bilgilerini temsil eder. Bu sınıf; yolcu kimliği, adı, soyadı ve iletişim bilgilerini içerir. Passenger sınıfı, Reservation sınıfı tarafından kullanılmaktadır. Bir yolcu birden fazla rezervasyon yapabilir.

Attributes

- **id : String**
  Yolcuya ait benzersiz kimlik bilgisini tutar.
- **name : String**
  Yolcunun adını tutar.
- **surname : String**
  Yolcunun soyadını tutar.
- **contact : String**
  Yolcunun iletişim bilgisini (telefon veya e-posta) tutar.

Constructor

- **Passenger(String id, String name, String surname, String contact):** Yolcu nesnesini oluşturur ve yolcuya ait temel kimlik ve iletişim bilgilerini tanımlar.

Reservation

Reservation sınıfı, bir yolcuya ait uçuş rezervasyonunu temsil eder. Bu sınıf; rezervasyon kodu, uçuş bilgisi, yolcu bilgisi, koltuk ve rezervasyon zamanı gibi verileri içerir. Reservation sınıfı, Flight, Passenger ve Seat sınıfları ile ilişkilidir. Sistem içerisindeki temel iş kurallarının merkezinde yer alır.

Attributes

- **reservationCode : String**
  Rezervasyonu benzersiz şekilde tanımlayan koddur.
- **flight : Flight**
  Rezervasyonun ait olduğu uçuşu temsil eder.
- **passenger : Passenger**
  Rezervasyonu yapan yolcuyu temsil eder.
- **seat : Seat**
  Rezervasyon yapılan koltuğu temsil eder.
- **reservationTime : String**

Rezervasyonun oluşturulduğu zamanı tutar.

- **ownerUsername : String**
  Rezervasyonu sistemde hangi kullanıcı hesabının oluşturduğunu belirtir.

Constructor

- **Reservation(String reservationCode, Flight flight, Passenger passenger, Seat seat, String reservationTime, String ownerUsername):** Bir rezervasyon nesnesi oluşturur ve yolcu, uçuş, koltuk ve kullanıcı bilgilerini ilişkilendirir.

Ticket

Ticket sınıfı, onaylanmış bir rezervasyon için oluşturulan bileti temsil eder. Bilet numarası, toplam fiyat ve rezervasyon bilgileri bu sınıfta tutulur. Ticket sınıfı, Reservation sınıfı ile *association* ilişkisine sahiptir. Bilet fiyatı hesaplamalarında CalculatePrice sınıfı kullanılır.

Attributes

- **ticketID : String**
  Bileti benzersiz şekilde tanımlayan koddur.
- **reservation : Reservation**
  Biletin bağlı olduğu rezervasyonu temsil eder.
- **price : int**
  Hesaplanan toplam bilet fiyatını tutar.
- **baggageList : List<Baggage>**
  Yolcuya ait bagaj listesini tutar.

Constructor

- **Ticket(String ticketID, Reservation reservation, int price, List<Baggage> baggageList):** Rezervasyon ve bagaj bilgilerine göre bilet nesnesini oluşturur.

Baggage

Baggage sınıfı, yolcuya ait bagaj bilgilerini temsil eder. Bu sınıf yalnızca bagaj ağırlığını içerir ve fiyat hesaplamalarında kullanılır. Baggage sınıfı, Ticket ve CalculatePrice sınıfları ile ilişkilidir. Bagaj ücretlendirme kuralları bu sınıf üzerinden uygulanır.

Attributes

- **weight : double**
  Bagajın ağırlığını temsil eder.

Constructor

- **Baggage(double weight):** Verilen ağırlık bilgisiyle bagaj nesnesi oluşturur.

4. Services And Managers

FlightManager

FlightManager sınıfı, sistemdeki uçuşların yönetiminden sorumludur. Bu sınıf; uçuş ekleme, silme, listeleme ve kalkış–varış noktalarına göre arama işlemlerini gerçekleştirir. FlightManager, Flight sınıfı ile *association* ilişkisine sahiptir. Uygulamanın uçuş verilerinin merkezi kontrol noktasıdır.

Attributes

- **flights : List<Flight>**
  Sistemde kayıtlı tüm uçuşların tutulduğu koleksiyondur

Constructors

- **FlightManager() *(default)* :** Uçuş listesini başlatır.

Methods

- **addFlight(flight):** Yeni uçuş ekler.
- **deleteFlight(flightNum):** Uçuş numarasına göre uçuş siler.
- **searchFlights(dep, arr):** Kalkış-varış filtreli uçuş araması yapar.
- **searchUpcomingFlights(dep, arr, nowDate, nowHour):** Kalkış zamanı geçmiş uçuşları eleyerek sadece “upcoming” uçuşları döndürür (JUnit gereksinimi).

ReservationManager

ReservationManager sınıfı, rezervasyon işlemlerini yönetmek amacıyla tasarlanmıştır. Bu sınıf, çoklu yolcu senaryolarını simüle etmek için multithreading kullanır. Senkronize ve senkronize olmayan modlarda koltuk yerleştirme işlemleri yapılabilmektedir. ReservationManager, Reservation, Seat ve Flight sınıfları ile ilişkilidir.

Attributes

- **reservations : List<Reservation>**
  Sistem içerisindeki tüm rezervasyonların tutulduğu liste (eşzamanlı erişime uygun yapı).
- **lock : Object**
  Senkronize rezervasyon oluşturma sürecinde kritik bölgeyi korumak için kullanılan kilit nesnesi.

Constructors

- **ReservationManager()** ***(default)*****:** Rezervasyon listesini ve senkronizasyon yapısını hazırlar.

Methods

- **makeReservation(Flight flight, Passenger passenger, Seat seat, String dateOfReservation, String ownerUsername, boolean synchronizedMode):** Senkronize veya senkronize olmayan moda göre rezervasyon oluşturmayı başlatan üst metottur.
- **makeReservationSafe(Flight flight, Passenger passenger, Seat seat, String dateOfReservation, String ownerUsername):** Senkronize (thread-safe) rezervasyon oluşturur; veri tutarlılığını garanti eder.
- **makeReservationUnsafe(Flight flight, Passenger passenger, Seat seat, String dateOfReservation, String ownerUsername):** Senkronize olmayan rezervasyon oluşturur; yarış durumlarını (race condition) göstermek için kullanılır.
- **cancelReservation(String reservationCode):** Rezervasyon koduna göre rezervasyonu iptal eder.
- **runSeatSimulation(Flight flight, int passengerCount, boolean synchronizedMode, SeatSimulationListener listener):** Çoklu thread ile koltuk rezervasyon simülasyonu çalıştırır ve GUI'yi listener ile günceller.
- **loadReservations(List<Reservation> list):** Dışarıdan yüklenen rezervasyon listesini sistemdeki listeye aktarır (dosyadan yükleme sonrası).

TicketManager

TicketManager sınıfı, rezervasyonlardan bilet oluşturma işlemlerinden sorumludur. Bu sınıf, rezervasyon bilgisi, rota ve bagaj bilgilerini kullanarak bilet üretir. TicketManager, Ticket, Reservation, Baggage ve Route sınıfları ile etkileşim içerisindedir. Bilet üretimi sırasında fiyat hesaplaması CalculatePrice sınıfı ile yapılır.

Attributes

- **tickets : List<Ticket>** ***(static)***
  Sistemde oluşturulan biletlerin tutulduğu koleksiyon.
- **FILE_PATH : Path** ***(static)***
  Biletlerin saklandığı metin dosyası yolu (data/tickets.txt).

Constructors

- **TicketManager()** ***(default)*****:** Bilet yönetimi için gerekli dosya/başlangıç hazırlıklarını kullanıma hazır hale getirir.

Methods

- **generateTicket(Reservation reservation, List<Baggage> baggageList, Route route):** Rezervasyon ve bagaj bilgilerine göre fiyatı hesaplayıp bilet oluşturur.
- **saveTickets():** Bilet listesini metin dosyasına kaydeder.
- **loadTickets(List<Reservation> allReservations):** Metin dosyasından biletleri okuyup ilgili rezervasyonlarla eşleştirerek yükler.
- **deleteTicketsByReservationCode(String resCode):** Belirli bir rezervasyon koduna bağlı bilet(leri) siler.

UserManager

UserManager sınıfı, kullanıcı kayıt ve giriş işlemlerini yönetir. Bu sınıf; kullanıcı doğrulama, aktif kullanıcı takibi ve yetkilendirme işlemlerini gerçekleştirir. Kullanıcı verileri, kalıcı depolama için metin dosyaları (.txt) kullanılarak saklanmaktadır. UserManager, User, Customer ve Staff sınıfları ile ilişkilidir.

Attributes

- **activeUser : User** ***(static)***
  O an giriş yapmış kullanıcıyı tutar (oturum mantığı).

Constructors

- **UserManager()** ***(default)***
  Kullanıcı dosya sistemini ve kullanıcı listesini hazırlar.

Methods

- **ensureFile():** Kullanıcı verisinin saklanacağı dosya/dizin yapısının varlığını garanti eder.

- **login(String username, String password):** Kullanıcı giriş doğrulaması yapar; başarılıysa activeUser belirlenir.
- **logout():** Aktif kullanıcıyı sıfırlayarak oturumu kapatır.
- **registerCustomer(String username, String password):** Yeni müşteri kaydı oluşturur ve dosyaya yazar.
- **saveUsers():** Kullanıcı listesini metin dosyasına kaydeder.
- **loadUsers():** Metin dosyasından kullanıcı listesini yükler.

SeatManager

SeatManager sınıfı, uçak içerisindeki koltukların oluşturulması ve durumlarının yönetilmesinden sorumludur. Bu sınıf, uçak konfigürasyonuna göre koltukları otomatik olarak üretir. SeatManager, Plane ve Seat sınıfları ile ilişkilidir. Koltuk doluluk haritasının oluşturulmasında temel rol oynar.

Attributes

- **planeModels : Map<String, PlaneConfig>**
  Uçak modeli–konfigürasyon eşleşmelerini tutar (model bazlı koltuk düzeni).
- **CONFIG_PATH : Path**
  Uçak modellerinin saklandığı metin dosyasının yolunu tutar (data/plane_models.txt).

Constructors

- **SeatManager() *(default)*:** Model konfigürasyon verisini yönetmek için gerekli altyapıyı hazırlar.

Methods

- **defineModel(PlaneConfig config):** Yeni bir uçak model konfigürasyonu tanımlar ve map içine ekler.
- **saveModels():** Tanımlı uçak modellerini metin dosyasına kaydeder.
- **loadModels():** Metin dosyasından uçak modellerini okuyup sisteme yükler.
- **createPanelLayout(PlaneConfig config):** GUI'de çizim için koltukların panel yerleşimini oluşturur.
- **createSeatsForFlight(Plane plane, Route route):** Bir uçuş için uçağın konfigürasyonuna göre koltuk matrisini oluşturur.
- **buildOccupiedMap(Seat[][] seatStatus):** GUI'de göstermek için boolean doluluk haritası üretir.

- **occupiedCount(Seat[][] seatStatus):** Dolu koltuk sayısını hesaplar.
- **emptyCount(Seat[][] seatStatus):** Boş koltuk sayısını hesaplar (JUnit senaryosu için kritik).
- **findSeatByNumber(Seat[][] seatStatus, String seatNum):** Verilen koltuk numarasına göre koltuğu bulur (bulamazsa null).
- **seatNumber(int r, int c):** Satır/sütun indeksinden "1A" gibi koltuk numarası üretir.
- **requireSeatByNumber(Seat[][] seatStatus, String seatNum):** Koltuk bulunamazsa exception fırlatır (JUnit "non-existent seat" senaryosu için kritik).

CalculatePrice

CalculatePrice sınıfı, bilet fiyatlarının hesaplanmasından sorumludur. Bu sınıf; uçuş mesafesi, koltuk sınıfı ve bagaj bilgilerine göre toplam fiyatı hesaplar. CalculatePrice sınıfı statik metotlar içerir ve nesne oluşturulmadan kullanılabilir. Bu yapı, business logic'in merkezi bir sınıfta toplanmasını sağlar.

Methods

- **flightPrice(seatClass, route):** Koltuk sınıfı ve rota bilgisine göre uçuş ücretini hesaplar.
- **baggageFee(seatClass, baggages):** Ücretsiz bagaj hakkı ve limit aşımı kurallarına göre bagaj ücretini hesaplar.
- **totalTicketPrice(seatClass, baggages, route):** Uçuş fiyatı + bagaj ücretini toplayarak toplam bilet fiyatını üretir.

FileStore

FileStore sınıfı, sistemdeki kalıcı veri saklama işlemlerini yönetir. Bu sınıf; uçuş ve rezervasyon verilerini metin dosyalarına yazma ve okuma işlemlerini gerçekleştirir. Veri saklama için herhangi bir veritabanı veya XML yapısı kullanılmamıştır. Bu yaklaşım, proje gereksinimlerine uygun olarak dosya tabanlı kalıcı depolama sağlar.

Attributes

- **baseDir : Path**

  Dosya işlemlerinin yapılacağı ana klasör yolunu tutar (örn: data/).

Constructors

- **FileStore(String folderName):** Verilen klasör adına göre baseDir yapısını oluşturur ve dosya işlemlerini bu dizin altında yapar.

Methods

- **saveFlights(List<Flight> flights, String fileName):** Uçuş listesini metin dosyasına kaydeder.
- **loadFlights(String fileName):** Metin dosyasından uçuşları okuyup List<Flight> olarak döndürür.
- **saveReservations(List<Reservation> reservations, String fileName):** Rezervasyon listesini metin dosyasına kaydeder.
- **loadReservations(String fileName, List<Flight> flights):** Metin dosyasından rezervasyonları okuyup ilgili uçuş nesneleriyle eşleştirerek yükler.

ReportGenerator (Runnable)

ReportGenerator sınıfı, sistemdeki uçuş ve rezervasyon verilerini kullanarak rapor üretmek amacıyla tasarlanmıştır. Bu sınıf Runnable arayüzünü implemente eder ve ayrı bir thread içerisinde çalışır. Rapor oluşturma işlemi ana uygulamadan bağımsız olarak gerçekleştirilir. Bu yapı, sistemin eşzamanlı çalışma yeteneklerini göstermek için kullanılmıştır.

Attributes

- **flights : List<Flight>**
  Rapor hazırlanırken kullanılacak uçuş listesini tutar.
- **listener : ReportListener**
  Raporun durum ve sonuç mesajlarını GUI'ye iletmek için kullanılır.

Constructors

- **ReportGenerator(List<Flight> flights, ReportListener listener):** Rapor üretiminde kullanılacak uçuş listesini ve callback mekanizmasını alır.

Methods

- **run():** Ayrı bir thread içerisinde raporu üretir ve sonuçları ReportListener üzerinden iletir.

SeatSimulationListener (Interface)

SeatSimulationListener arayüzü, koltuk simülasyonu sırasında GUI'nin bilgilendirilmesi amacıyla kullanılır. Bu arayüz, simülasyonun her adımında koltuk doluluk durumunun güncellenmesini sağlar. ReservationManager tarafından tetiklenir ve GUI sınıfları tarafından implemente edilir. Bu yapı, observer pattern benzeri bir yaklaşım sunar.

Methods

- **onSimulationUpdate(boolean[][] occupiedMap, int occupiedCount, int emptyCount):** Simülasyon sürerken ara güncellemeleri GUI'ye iletir.
- **onSimulationFinished(boolean[][] occupiedMap, int occupiedCount, int emptyCount):** Simülasyon bittiğinde final sonucu GUI'ye iletir.

ReportListener (Interface)

ReportListener arayüzü, rapor oluşturma sürecinin durumunu takip etmek amacıyla tasarlanmıştır. ReportGenerator sınıfı tarafından rapor ilerleme ve tamamlanma bilgileri bu arayüz aracılığıyla iletilir. GUI sınıfları bu arayüzü implemente ederek kullanıcıya anlık geri bildirim sunar. Bu yapı, arka plan işlemleri ile kullanıcı arayüzü arasındaki iletişimi sağlar.

Methods

- **onReportStatus(String message):** Rapor üretimi sırasında ara durum mesajlarını bildirir.
- **onReportReady(String reportText):** Rapor tamamlandığında rapor metnini bildirir.

5. Users

User (Abstract)

User sınıfı, sistemdeki tüm kullanıcılar için soyut bir üst sınıftır. Bu sınıf; kullanıcı adı, şifre ve rol bilgilerini içerir. User sınıfı doğrudan örneklenemez ve Customer ile Staff sınıfları tarafından genişletilir. Bu yapı sayesinde inheritance ve polymorphism prensipleri uygulanmıştır.

Attributes

- **username : String**
  Kullanıcının giriş için kullandığı kullanıcı adını tutar.
- **password : String**
  Kullanıcının giriş için kullandığı şifre bilgisini tutar.
- **roleType : Role**
  Kullanıcının sistemdeki yetki tipini belirtir.

Inner Enum: Role

- **STAFF —** Yönetici/personel kullanıcı
- **CUSTOMER —** Müşteri/yolcu kullanıcı

Constructor

- **User(String username, String password, Role role):** Ortak kullanıcı alanlarını (username, password) ve rol tipini (roleType) atar. Bu sınıf abstract olduğu için doğrudan örneklenmez; alt sınıflar tarafından kullanılır.

Customer

Customer sınıfı, sisteme yolcu olarak giriş yapan kullanıcıları temsil eder. Customer sınıfı, User sınıfından türetilmiştir. Rezervasyon yapma ve bilet oluşturma işlemleri Customer kullanıcıları tarafından gerçekleştirilir. Bu sınıf, User sınıfının davranışlarını özelleştirir.

Constructor

- **Customer(String username, String password):** Üst sınıfın constructor'ını çağırır ve rolü otomatik olarak Role.CUSTOMER olarak belirler.

Staff

Staff sınıfı, sistem yöneticisi veya personel kullanıcılarını temsil eder. Staff sınıfı, User sınıfından türetilmiştir. Uçuş ekleme, silme ve yönetimsel işlemler yalnızca Staff kullanıcıları tarafından yapılabilir. Bu sınıf, sistem güvenliği ve yetkilendirme açısından kritik rol oynar.

Constructor

- **Staff(String username, String password):** Üst sınıfın constructor'ını çağırır ve rolü otomatik olarak Role.STAFF olarak belirler.

6. Tests

CalculatePriceTest

Bilet fiyatlarının doğru şekilde hesaplandığını doğrulamak amacıyla CalculatePrice sınıfı test edilmiştir.

Bu testler kapsamında:

- Economy sınıfı için bilet fiyatının, uçuş mesafesi ve temel fiyat bilgisine göre doğru hesaplandığı doğrulanmıştır.
- Business sınıfı için hesaplanan bilet fiyatının, Economy sınıfı fiyatının iki katı olduğu test edilmiştir.

Bu testler sayesinde, koltuk sınıfına bağlı fiyatlandırma kurallarının doğru şekilde uygulandığı garanti altına alınmıştır.

FlightSearchEngine Test

Uçuş arama ve filtreleme işlemlerinin doğruluğunu test etmek amacıyla FlightManager sınıfı üzerinde birim testler gerçekleştirilmiştir.

Bu testler kapsamında:

- Kullanıcının girdiği kalkış ve varış şehirlerine göre doğru uçuşların filtrelenip listelendiği doğrulanmıştır.
- Kalkış zamanı geçmiş olan uçuşların, arama sonuçlarından doğru şekilde elendiği test edilmiştir.

Bu testler, sistemin yalnızca geçerli ve ileri tarihli uçuşları kullanıcıya sunmasını sağladığını göstermektedir.

SeatManager Test

Koltuk yönetimi ve rezervasyon işlemlerinin doğruluğunu kontrol etmek amacıyla SeatManager sınıfı test edilmiştir.

Bu testler kapsamında:

- Bir koltuk rezerve edildikten sonra, boş koltuk sayısının (emptySeatsCount) doğru şekilde azaldığı doğrulanmıştır.
- Mevcut olmayan bir koltuk numarası ile rezervasyon yapılmaya çalışıldığında, sistemin uygun bir exception fırlattığı test edilmiştir.

Bu testler sayesinde, koltuk yönetimi sırasında veri tutarlılığının korunduğu ve hatalı işlemlerin güvenli şekilde engellendiği gösterilmiştir.

7.GUI

LoginFrame

LoginFrame sınıfı, kullanıcıların sisteme giriş yaptığı grafik arayüzü temsil eder. Bu sınıf, kullanıcı adı ve şifre bilgilerini alarak UserManager üzerinden doğrulama yapar. Giriş işlemi başarılı olduğunda ana uygulama ekranına geçiş yapılır. LoginFrame, kullanıcı–sistem etkileşiminin başlangıç noktasıdır.

Attributes

- **txtUsername : JTextField —** Kullanıcı adı girişi
- **txtPassword : JPasswordField —** Şifre girişi
- **btnLogin : JButton —** Giriş butonu
- **btnRegister : JButton —** Kayıt ol butonu

Constructors

- **LoginFrame():** Login penceresini oluşturur, bileşenleri yerleştirir ve buton aksiyonlarını bağlar.

Methods

- **doLogin():** Kullanıcı adı/şifreyi alır, UserManager.login(...) ile doğrular; başarılıysa MainFrame açılır.
- **doRegister():** Girilen bilgilerle UserManager.registerCustomer(...) çağırarak yeni müşteri kaydı oluşturur**.**

MainFrame

MainFrame sınıfı, uygulamanın ana grafik arayüzünü temsil eder. Bu sınıf; uçuş arama, koltuk seçimi, rezervasyon oluşturma ve rapor oluşturma işlemlerini içerir. MainFrame, FlightManager, ReservationManager ve UserManager sınıfları ile etkileşim içerisindedir. Uygulamanın tüm ana fonksiyonları bu arayüz üzerinden gerçekleştirilir.

Attributes

- **fileStore : FileStore —** dosya tabanlı kalıcı kayıt/yükleme için
- **flightManager : FlightManager —** uçuş yönetimi ve arama işlemleri
- **reservationManager : ReservationManager —** rezervasyon/simülasyon işlemleri
- **activeFlight : Flight —** kullanıcının seçtiği aktif uçuş

Ana GUI bileşenleri

- **Arama alanları:** txtDeparture, txtArrival, btnSearch
- **Uçuş seçimi:** cmbFlights, btnSelectFlight
- **Simülasyon:** chkSynchronized, btnSimulate, lblCounts
- **Koltuk çizimi:** seatPanel : SeatPanel
- **Aksiyonlar:** btnReport, btnSave, btnLoad, btnAdmin, btnReservations, btnTickets
- **Bilgi çıktısı:** txtOutput : JTextArea
- **Seçili koltuk bilgisi:** lblSelectedPrice : JLabel

Constructors

- **MainFrame():** Ana pencereyi kurar, demo veriyi hazırlar, bileşenleri yerleştirir ve tüm buton aksiyonlarını bağlar.

Methods

- **doSearch():** Kalkış/varış girdilerine göre flightManager.searchFlights(...) çalıştırır ve bulunan uçuşları combobox'a yükler.
- **selectFlight():** Combobox'tan seçilen uçuşu activeFlight yapar ve koltuk ekranını bu uçuşa göre yeniler.
- **refreshSeatView():** Aktif uçuşun seatStatus verisini SeatPanel üzerinde tekrar çizer ve dolu/boş sayısını günceller.
- **doSimulation():** Senaryo-1 koltuk simülasyonunu reservationManager.runSeatSimulation(...) ile başlatır (sync / not sync).
- **doReport():** ReportGenerator thread'ini başlatır ve raporu arka planda üretir.
- **bookSelectedSeat():** Kullanıcının SeatPanel'den seçtiği koltuğu alır, yolcu bilgisiyle rezervasyon oluşturma akışını başlatır.
- **doSave():** Uçuş/rezervasyon verilerini FileStore ile dosyaya kaydeder.
- **doLoad():** Dosyadan uçuş/rezervasyon verilerini okuyup sisteme yükler ve ekranı günceller.

Listener metotları (GUI güncelleme)

- **onSimulationUpdate(boolean[][] occupiedMap, int occupiedCount, int emptyCount):** Simülasyon sürerken koltuk haritasını ve sayaçları günceller.
- **onSimulationFinished(boolean[][] occupiedMap, int occupiedCount, int emptyCount):** Simülasyon sonunda final haritayı ve sayaçları günceller.
- **onReportStatus(String message):** Rapor üretilirken gelen ara durum mesajlarını txtOutput alanına yazar.
- **onReportReady(String reportText):** Rapor tamamlandığında rapor metnini ekrana basar.

SeatPanel

SeatPanel sınıfı, uçak koltuklarının görsel olarak gösterildiği paneldir. Bu sınıf, koltukların dolu veya boş durumlarını renkler ile ifade eder. Kullanıcıların koltuk seçimini fare tıklamaları ile yapabilmesini sağlar. SeatPanel, rezervasyon sürecinde kullanıcı deneyimini artırmak için tasarlanmıştır.

Attributes

- **seatMatrix : Seat[][] —** aktif uçuşun koltuk matrisi
- **selectedRow : int —** seçili satır indeksi
- **selectedCol : int —** seçili sütun indeksi
- **selectionListener : Consumer<Seat> —** koltuk seçildiğinde dışarı bilgi vermek için callback
- **Çizim parametreleri:** panel boyutu, hücre ölçüleri vb**.**

Constructors

- **SeatPanel():** Paneli hazırlar ve mouse event dinleyicilerini bağlar.

Methods

- **setSeatMatrix(Seat[][] seatMatrix):** Çizilecek koltuk matrisini günceller ve paneli yeniden çizdirir.
- **setSelectionListener(Consumer<Seat> listener):** Koltuk seçildiğinde çağrılacak callback'i tanımlar.
- **updatePanelSize():** Koltuk düzenine göre panelin kaydırılabilir alan boyutunu hesaplar/günceller.
- **getSelectedSeatNum():** Seçili koltuğun numarasını "1A" formatında üretir (GUI'de kullanılır).
- **clearSelection():** Seçimi temizler ve paneli günceller.
- **paintComponent(Graphics g):** Koltukları (dolu/boş + seçili durum) çizerek görselleştirir.
- **Mouse event metotları (seçim ve hover davranışı):** mouseClicked(...), mouseMoved(...), vb. — kullanıcı etkileşimlerini yönetir.

ReservationManagementFrame

ReservationManagementFrame sınıfı, mevcut rezervasyonların görüntülenmesini sağlayan arayüzdür. Bu sınıf, kullanıcıya ait rezervasyonları listelemek için ReservationManager kullanır. Rezervasyon bilgileri tablo veya liste formatında sunulmaktadır. Bu ekran, rezervasyon takibi ve kontrolü için kullanılır.

Attributes

- **reservationManager : ReservationManager**
- **flight : Flight**
- **Liste yapıları: DefaultListModel<Reservation> ve JList<Reservation>** (rezervasyonları göstermek için)
- **Butonlar: btnRefresh, btnCancel, btnTicket**

Constructors

- **ReservationManagementFrame(ReservationManager reservationManager, Flight flight):** Rezervasyon ekranını kurar, listeyi hazırlar ve buton aksiyonlarını bağlar.

Methods

- **refresh():** Seçili uçuşa ait rezervasyon listesini yeniler.
- **cancelSelected():** Listeden seçilen rezervasyonu iptal eder ve listeyi günceller.
- **ticketSelected():** Seçili rezervasyon için bagaj bilgileri ile bilet üretme sürecini başlatır (TicketManager/CalculatePrice akışı).

AdminFrame

AdminFrame sınıfı, sadece Staff kullanıcılarının erişebildiği yönetim ekranını temsil eder. Bu sınıf üzerinden uçuş ekleme, silme ve sistem yönetimi işlemleri yapılabilir. AdminFrame, UserManager aracılığıyla kullanıcı yetkisini kontrol eder. Bu yapı, sistemin güvenli ve kontrollü kullanılmasını sağlar.

Attributes

- **flightManager : FlightManager —** uçuş işlemleri için
- **onChange : Runnable —** yapılan değişiklikleri ana ekrana yansıtmak için callback

Model tanımlama bileşenleri

- **txtModelName, txtModelRows, txtModelCols, txtModelBiz**

- **btnSaveModel**

Uçuş ekleme bileşenleri

- **cmbPlaneModels**
- **txtFlightNum, txtDep, txtArr, txtDistance, txtBasePrice, txtPlaneID, txtDate, txtHour, txtDuration**
- **btnAddFlight, btnDeleteFlight**

Constructors

- **AdminFrame(FlightManager flightManager, Runnable onChange):** Admin panelini kurar, combobox modellerini doldurur ve buton aksiyonlarını bağlar.

Methods

- **refreshModelCombo():** Mevcut uçak modellerini (SeatManager üzerinden) combobox'a yükler/günceller.
- **doSaveModel():** Girilen bilgilerle PlaneConfig oluşturur ve model listesini kaydeder.
- **doAddFlight():** Girilen bilgilerle Route, Plane, Flight nesnelerini üretip flightManager.addFlight(...) ile sisteme ekler.
- **doDeleteFlight():** Uçuş numarası ile uçuşu flightManager.deleteFlight(...) üzerinden siler.

TicketFrame

Kullanıcıya ait biletleri listeleyen ekrandır. Bilet verilerini yenileme işlemi içerir.

Attributes

- **model : DefaultListModel<String> —** bilet metinlerini tutar
- **list : JList<String> —** biletleri gösterir
- **btnRefresh : JButton —** yenileme butonu

Constructors

- **TicketFrame():** Bilet penceresini oluşturur ve yenileme aksiyonunu bağlar.

Methods

- **refresh():** Aktif kullanıcıya ait biletleri yükler ve listede gösterir.