Separating UserBillingAddress and UserContactInformation is all about good design principles, following the single responsibility, composition and reusability principles. This way, billing address can be used for companies, or other classes, and contact information can be passed to other systems without all the User data. UserBillingAddress can also be used by CardPaymentInformation if required, as well as HumanResources for sending the payroll.

UserContactInformation

- email: string
- phoneNumber: string

- verifyEmail()
- verifyPhoneNumber()

UserBillingAddress

- address: string
- postalCode: string
- city: string
- country: string

+ validate()
+ getNormalized()

<<abstract>>
User

- UUID: string
- name: string
- surname: string
- passwordHash: string
- governmentID: string
- dateOfBirth: Date
- createdAt: Date
- consentToDataProcessing: boolean

- verifyAge() <<abstract>>
- generateUUID() <<abstract>>
+ getFullName()
+ login()

Employee

- hireDate: Date
- roles: List<Role>

- generateUUID()
- verifyAge()
+ clockIn()
+ clockOut()

The interface Role is a nice way to separate role responsabilities between classes, but with the possibility of an Employee holding more than one Role.

<<interface>>
Role

+ getPermissions():List<String>

CleaningRole

+ getCheckOuts():List<Room>
+ changeRoomStatus(room:Room)

ManagerRole

+ createRoom()
+ modifyRoom(room:Room)
+ removeRoom(room:Room)

HumanResourcesRole

+ addRoleToEmployee(role:Role, employee:Employee)
+ addEmployee()
+ removeEmployee(employee:Employee)
+ getEmployeeInfo(employee:Employee)
+ modifyEmployeeInfo(employee:Employee)
+ getEmployeeInfo()

ReceptionistRole

+ checkInGuest(guest:Guest)
+ checkOutGuest(guest:Guest)
+ payReservation(reservation:Reservation)
+ getBookings()
+ getBookings(room:Room)
+ getBookings(dateMin:Date, dateMax:Date)
+ getBookingInfo(booking:Booking)
+ getReservations()
+ getReservations(room:Room)
+ getReservations(dateMin:Date, dateMax:Date)
+ getReservationInfo(reservation:Reservation)
+ getGuestInfo(guest:Guest)
+ getRoomInfo(room:Room)
+ createGuest()
+ modifyGuest(guest:Guest)
+ removeGuest(guest:Guest)

The interface PaymentInformation is just to be able to select different payment methods without using inheritance (there is no information in common between Card and Cash payment information).

CardPaymentInformation

- cardType: string
- cardNumber: string
- cardHolderName: string
- cardExpirationDate: string
- securityCode: integer

- verifyCard()

CashPaymentInformation

VirtualCardPaymentInformation

<<interface>>
PaymentInformation

Guest

- nationality: Nationality
- reservationList: List<Reservation>
- bookingList: List<Booking>

- generateUUID()
- verifyAge()
+ checkIn()
+ checkOut()
+ getRooms(dateIn:Date, dateOut:Date):List<Room>
+ getDates(room:Room):List<Dates>
+ makeBooking(dateIn:Date, dateOut:Date, room:Room)
+ makeBooking(reservation:Reservation)
+ modifyBooking(booking:Booking)
+ makeReservation(dateIn:Date, dateOut:Date, room:Room)
+ modifyReservation(reservation:Reservation)
+ cancelReservation(reservation:Reservation)
+ addPaymentInformation()

Hotel

- guests: List<Guest>
- rooms: List<Room>

+ getGuests():List<Guest>
+ createGuest()
+ modifyGuest(guest:Guest)
+ deleteGuest(guest:Guest)
+ getRooms():List<Room>
+ createRoom()
+ modifyRoom(room:Room)
+ deleteRoom(room:Room)

Task

- title: string
- description: string
- taskType: TaskType
- assignedEmployee: Employee
- assignedRoom: Room
- scheduledDate: Date
- status: TaskStatus

+ changeStatus(newStatus:TaskStatus)

Payment

- paymentUUID: string
- totalPrice: double
- discountAmount: double
- discountPercentage: integer
- status: PaymentStatus
- paymentMethod: PaymentMethod
- paymentInformation: PaymentInformation
- currency: string

- generateUUID()
+ changeStatus(status:PaymentStatus)
+ getPaymentInformation()
+ executePayment()

Booking

- bookingDate: Date
- dateIn: Date
- dateOut: Date
- price: double

+ calculatePrice()
+ applyDiscount()
+ changeDates(dateIn:Date,dateOut:Date)

BookingDetail

- price: Double
- rooms: List<Room>

+ getPrice()

BookingDetail solves the N:M relation proposed when considering that it should be possible to make a Booking of more than one Room.

Reservation

- status:ReservationStatus

+ calculatePrice()
+ applyDiscount()
+ changeDates(dateIn:Date,dateOut:Date)
+ makeBooking()
+ changeStatus(status:ReservationStatus)

A Reservation is a kind of Booking in which only part of the price is payed beforehand. The rest can be payed both on the reception, or online. This way, in case of cancellation, only the amount payed beforehand is lost.

Room

- name: string
- numberOfBeds: integer
- bathroom: boolean
- price: double
- status: RoomStatus

Built-in JS Date class.

Date

<<Enumeration>>
PaymentStatus

+ Payed
+ Not Payed
+ Partly Payed

<<Enumeration>>
RoomStatus

+ Prepared
+ Unprepared
+ Booked

<<Enumeration>>
ReservationStatus

+ Booked
+ Reserved

<<Enumeration>>
PaymentMethod

+ In Reception (Cash)
+ In Reception (Card)
+ Online (Card)
+ Online (Virtual Card)

<<Enumeration>>
Nationality

+ Spain
+ France
+ Iceland
+ ...

<<Enumeration>>
TaskType

+ Cleaning
+ Mainteinance
+ Office Work
+ Entertainment

<<Enumeration>>
TaskStatus

+ In Progress
+ Done
+ To Do
+ Updated

<<Enumeration>>
Role

+ Manager
+ Receptionist
+ Accountant
+ Cleaning Personnel
+ ...