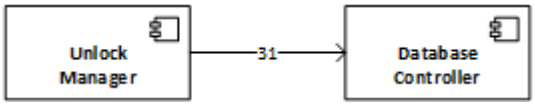
1 Lorem ipsum

1.1 Lorem ipsum

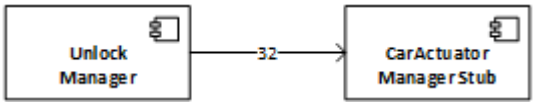
1.1.1 Lorem ipsum

Reservation Controller

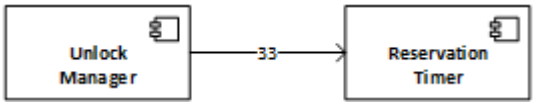
First we integrate the UnlockManger with the DatabaseController



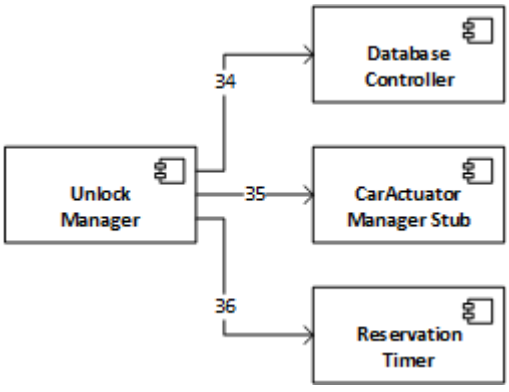
Then we integrate the UnlockManager with the CarActuatorDriver



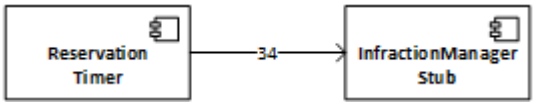
At last we integrate the UnlockManager with the ReservationTimer



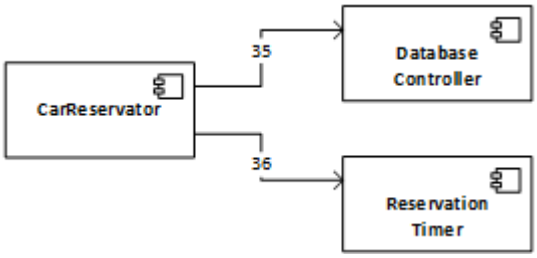
At this point we are able to integrate these components all togheter



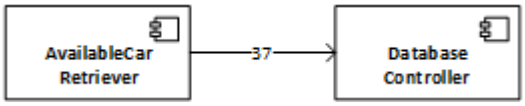
We proceed integrating the ReservationTimer with the InfractionManager-Driver



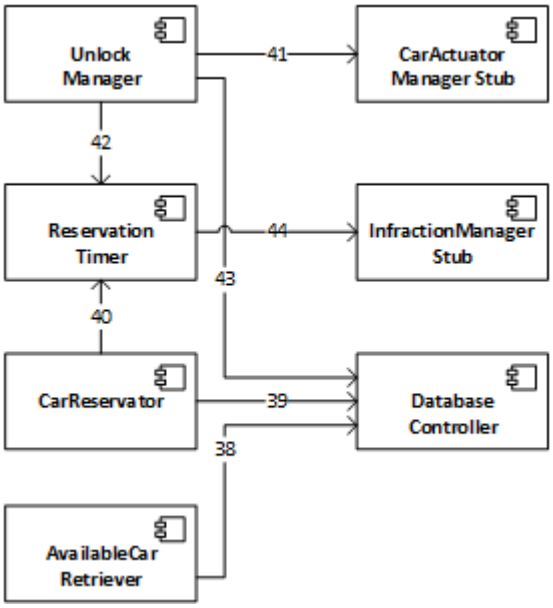
Now we integrate the CarReservator with both DatabaseController and ReservationTimer



To complete the subsystem integration, we integrate AvailableCarRetriever with DatabaseController

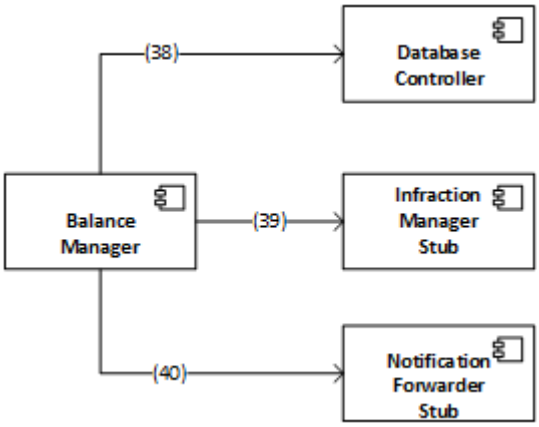


This is the final integrated ReservationController module

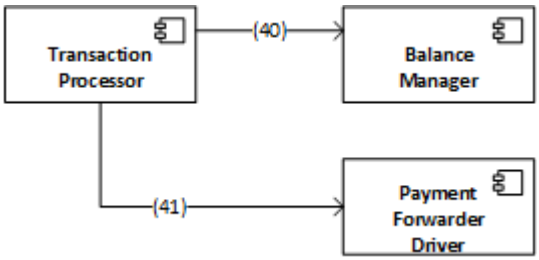


BalanceController

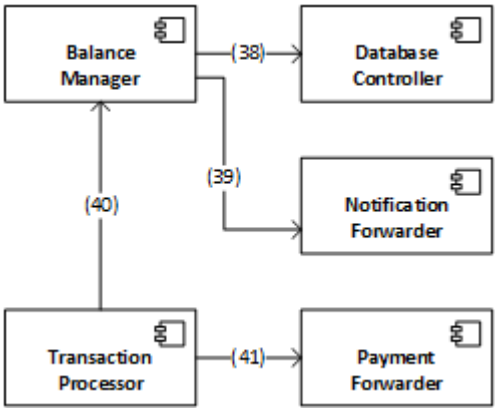
To integrate the components of this module we start from integrating BalanceManager with DatabaseController, InfractionManagerStub and NotificationForwarderStub



Then we integrate TransactionProcessor with both BalanceManager and PaymentForwarder

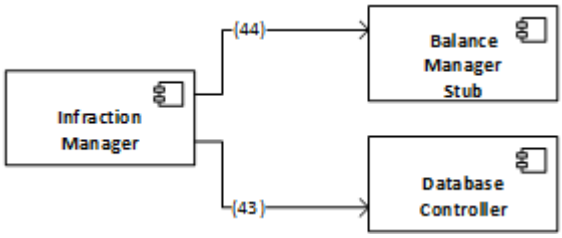


This is the final integrated BalanceController module

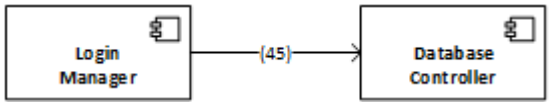


AccountController

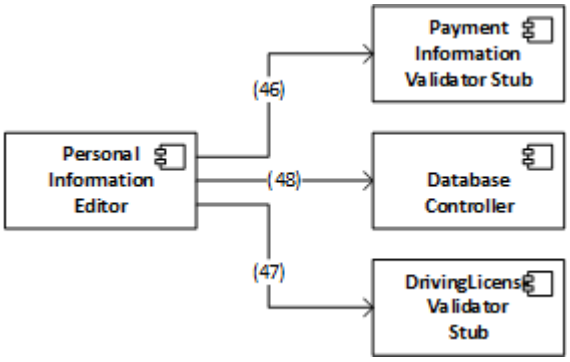
We start integrating InfractionManager with DatabaseController and BalanceManagerStub



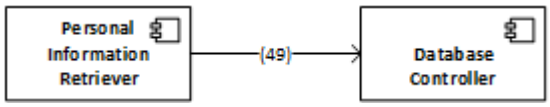
The second step is integrating LoginManager with DatabaseController



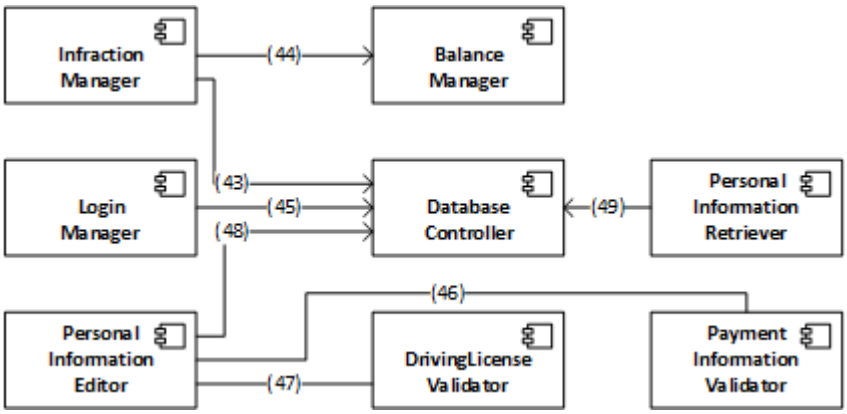
After that we integrate PersonalInformationEditor with PaymentInformationValidator, DrivingLicenseValidator and DatabaseController



The last integration is the one between PersonalInformationRetriever and DatabaseController

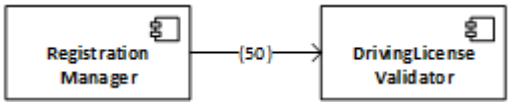


This is the final integrated AccountController module

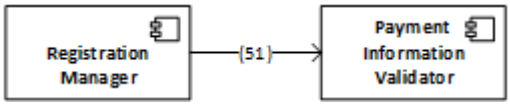


RegistrationController

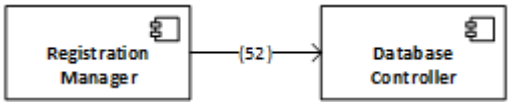
We start the integration procedure from the integration between the RegistrationManager and the DrivingLicenseValidator



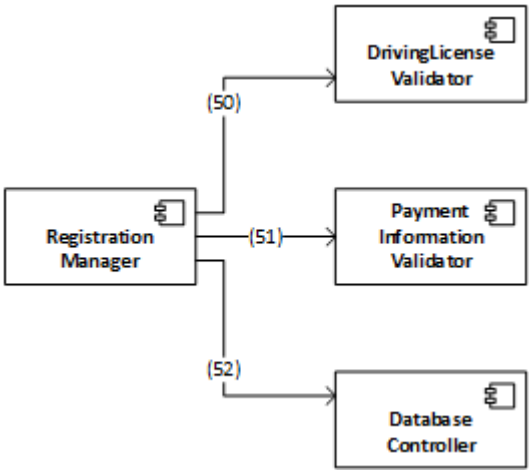
As second step, we integrate the RegistrationManager with the PaymentInformationValidator



In the end we integrate RegistrationManager with DatabaseController

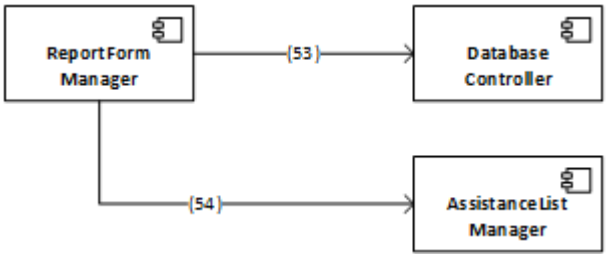


This is the final integrated RegistrationController module



ReportController

The only integration required for this module is the one between the ReportFormManager and both the DatabaseController and the AssistanceListManagerStub



ReservationController

UnlockManager, DatabaseController

isUnlockable(vehicle-id, user-info)	
Input	**Effect**
Non existing vehicle, Any	InvalidPlateNumberException is raised
Existing vehicle, user too far from the car	TooFarException is raised
Existing but unlocked vehicle, valid user-info	AlreadyUnlockedException is raised
Existing and locked vehicle	Returns true

UnlockManager, CarActuatorDriver

unlock(vehicle-id)	
Input	**Effect**
Unlockable vehicle-id	Right function is called

UnlockManager, ReservationTimer

stop-timer(vehicle-id, user)	
Input	**Effect**
Valid vehicle-id, valid user	The timer for reservation is stopped

ReservationTimer, InfractionManagerDriver

timeout(vehicle-id, user-id)	
Input	**Effect**
Any, Any	Right function is called from InfractionManager

CarReservator, DatabaseController, ReservationTimer

create-reservation(user-id, vehicle-id)	
Input	**Effect**
Non existing user-id, Any	InvalidArgumentException is raised
Any, Non existing vehicle-id	InvalidArgumentException is raised
Debtor user, Any	InvalidUserException is raised
Banned user, Any	InvalidUserException is raised
Regular user, Reserved vehicle	UnavailableCarException is raised
Regular user, Needy vehicle	UnavailableCarException is raised
Regular user, Available vehicle	Database entry is created and timer is started

AvailableCarRetriever, DatabaseController

get-near-vehicles(position, range)	
Input	**Effect**
Invalid position, Any	InvalidPositionException is raised
Valid position, Non-positive integer	InvalidArgmuentException is raised
Valid position, Positive integer	Returns the list of available cars in the input range from the input poition

BalanceController

BalanceManager, DatabaseController, NotificationForwarder, InfractionManager

pay(user-id, amount)	
Input	**Effect**
Non existing user, Any	InvalidUserException is raised
Existing user, Non-positive value	InvalidArgumentException is raised
Existing user, More than deposited	Database entry is updated, NotificationForwarder and InfractionManager proper functions are called
Existing user, Positive value	Database entry is updated
deposit(user-id, amount)	
Input	**Effect**
Debtor user, More than debt	Database entry is updated, InfractionManager proper function is called
Regular user, Positive value	Database entry is updated

TransactionProcessor, PaymentForwarder, BalanceManager

deposit(payment-info, amount, user-id)	
Input	**Effect**
Any, Any, Non existing user	InvalidUserException is raised
Invalid info, Any, Existing user	TransactionExeption is raised
Valid info, Non-positive value, Existing user	InvalidArgumentException is raised
Valid info, Positive value more than affordable, Existing user	TransactionException is raised
Valid info, Positive value less than affordable, Existing user	BalanceManager deposit function is called

AccountController

InfractionManager, DatabaseController, BalanceManager

signal-infraction(infraction, user-id)	
Input	**Effect**
Any, Non existing user	InvalidUserException is raised
MissedPickUp, Existing user	Database entry is updated and BalanceManager pay function is called
AbandonedCar, Existing user	Database entry is updated and BalanceManager pay function is called
Debt, Existing user	Database entry is updated

LoginManager, DatabaseController

login(username, password)	
Input	**Effect**
Non existing user, Any	InvalidCredentialException is raised
Existing user, Wrong password	InvalidCredentialException is raised
Existing user, Correct password	Login is granted

PersonalInformationEditor, PaymentInformationValidator, DrivingLicenseValidator, DatabaseController

edit-personal-info(user-info, user-id)	
Input	**Effect**
Any, Non existing user	InvalidUserException is raised
Invalid info, Existing user	Proper PaymentInformationValidator and DrivingLicenseValidator functions are called and database entry is not updated
Valid info, Existing user	Proper PaymentInformationValidator and DrivingLicenseValidator functions are called and database entry is updated

PersonalInformationRetriever, DatabaseController

get-personal-info(user-id)	
Input	**Effect**
Non existing user	InvalidUserException is raised
Existing user	User's information is got from the database

RegistrationController

RegistrationManager, DrivingLicenseValidator

isValid(driving-license)	
Input	**Effect**
Invalid license	InvalidLicenseException is raised
Valid license	Returns true

RegistrationManager, PaymentInformationValidator

isValid(payment-info)	
Input	**Effect**
Invalid payment info	InvalidPaymentInfoException is raised
Valid payment info	Returns true

RegistrationManager, Database

register(user-info)	
Input	**Effect**
Existing username	AlreadyExistingUsernameException is raised
New username	Database entry is created

ReportController

ReportFormManager, DatabaseController, AssistanceListManager

report-damage(vehicle-plate, report)	
Input	**Effect**
Non existing plate	InvalidArgumentException is raised
Existing plate	Proper AssistanceListManager function is called