1) **Order Pizza from Dominos:**

Dominos	Data: dominosNumber Behavior: requestPizzaBaseChoice, requestPizzaSizeChoice, confirmOrder, generateBill, checkCouponValidity, checkCouponValidity, confirmCoupon, updateBill, assignDeliveryPerson
Customer	Data: customerName, customerEmailID, customerPhoneNumber, pizzaChoice, baseChoice, pizzaSize, creditCardDetails Behavior: callDominos, placePizzaOrder, applyCoupon, requestDelivery, callDeliveryPerson, makePayment, giveTip
Payment	Data: paymentMode, paymentDate Behavior: makeTransaction
Coupon	Data: couponCode, couponExpiryDate Behavior: isValid
Order	Data: orderNumber Behavior: updateOrder
DeliveryBoy	Data: deliveryPersonNumber, deliveryPersonName Behavior: confirmDeliveryLocation, deliverPizza, confirmPayment

Scenario:

```
jatin Customer;
dom Dominos;
cpa Coupon;
manish DeliveryPerson;

jatin.callDominos(dominosNumber);
/*customer calls dominos*/

Order order = jatin.placePizzaOrder(pizzaChoice);
base = dom.requestPizzaBaseChoice(baseChoice, order);
size = dom.requestPizzaSizeChoice(pizzaSize, order);
order.updateOrder(base, size);
dom.confirmOrder(order);
dom.generateBill(order, customerName, customerPhoneNumber);
/*customer places order and dominos confirms order and generates bill*/

jatin.applyCoupon(couponCode);
dom.checkCouponValidity(couponCode, couponExpiryDate)
if(cpa.isValid)
        dom.confirmCoupon();
        dom.updateBill()
/*customer applies coupon and dominos verifies it*/

Jatin.requestDelivery();
DeliveryPerson manish = dom.assignDeliveryPerson();
dom.provideDelieryDetails(deliveryPersonNumber);

jatin.callDeliveryPerson(deliveryPersonNumber, deliveryPersonName);
manish.confirmDeliveryLocation(customerAddress);
/*customer requests delivery option*/

manish.deliverPizza(customerAddress, customerName, order);
Payment jatinPay = jatin.makePayment(creditCardDetails);
jatinPay.makeTransaction(paymentDate);
manish.confirmPayment();
jatin.giveTip(deliveryPersonName);
/*customer makes payment*/

dom.requestFeedBack(customerName, customerEmailAddress);
```

2) Design a platform for buying tickets of local events:

Platform	Data: platformStatus Behavior: verifyEmail placeOrder confirmOrder paymentStatus sendBill eventReminder sendTicket
User	Data: name, emailID, password, numberOfTickets, eventToAttend, creditCardDetails Behavior: registerToApp loginToApp selectEvent confirmPaymentDetails makePayment cancelRegistration requestRefund
Event	Data: eventName, eventDate, eventTime, eventVenue, seatsVacant Behavior: addEvent isRegistrationOpen generateTicket
Payment	Data: amountDue, paymentDate Behavior: verifyUser allowTransfer sendPaymentConfirmation refundPayment

Scenario:

```
mark User;
eventBrite Platform;
carShow Event;
markPayment Payment;

carShow.addEvent(eventName, eventTime, eventDate, evenVenue)
  /*New event added to platform */

mark.registerToApp(name, emailID, password);
eventBrite.verifyEmail();
mark.loginToApp(emailID, password);
/* user mark registers and logins to app */

eventSelected = mark.selectEvent(eventToAttend, numberOfTickets);
if(eventSelected)
        eventBrite.placeOrder();
registrationOpen = carShow. isRegistrationOpen();
if(registrationOpen)
        eventBrite.confirmOrder();
paymentPending = eventBrite.paymentStatus();
if(paymentPending)
        eventBrite.sendBill();
/* the platform places order, confirms if registration is open and then confirms order and sends bill to
user. */

markPayment.verifyUser(mark.creditCardDetails)
mark.makePayment(amountDue);
markPayment.allowTransfer();
markPayment. sendPaymentConfirmation();
/* Payment veries user, allows transfer and send payment confirmation to user. */

paymentPending = eventBrite.paymentStatus();
if(not paymentPending)
        ticket = carShow.generateTicket();
        eventBrite.sendTicket(ticket);
/* After payment, the event generates ticket and sends to user via platform */

eventBrite.eventReminder(mark.emailID);
/* Send reminder to user about registered event. */

mark.cancelRegistration();
request =  mark.requestRefund();
Payment.refundPayment(request);
/* user cancels registration and request refund */
```

3) **Design a Car Rental System:**

RentalApp	Data: status Behavior: verifyUser, displayCarOptions, filterCarSpecifications, confirmBooking, generateContract, createInvoice, confirmPickUpDateTime, confirmReturnDateTime
Contract	Data: contractType, contractDuration, contractEndDate Behavior: acceptAndSign, extend
User	Data: name, phoneNumber, emailID, password, age, driveLicense, rentStartDate, rentEndDate, priceBudget Behavior: openAccount, loginAccount, enterRentDates, hasInsurance, enterCarRequirements, selectCar, purchaseInsurance, makePayement, extendContract
Car	Data: model, brand, color, mileage, numberOfSeats Behavior: NA
Insurance	Data: insurancePlan, insuranceType, insuranceNumber Behavior: renewPlan

Scenario:

```
louis User;
aero RentalApp;

louis.openAccount(name, phoneNumber, emailID, age)
aero .verifyUser (name, phoneNumber, driverLicense);
Louis.loginAccount(emailID, password);
/*user opens an account with the rental*/

louis.enterRentDates(rentStartDate, rentEndDate);
aero.displayCarOptions();
Louis.enterCarRequirements(model, brand, mileage, numberOfSeats);
aero.filterCarSpecifications();
audiA6 Car = louis.selectCar(priceBudget);
/*user enters rent date and filters using car requirements and selects an option*/

If(not louis.hasInsurance())
        Insurance insure = louis.purchaseInsurance(insurancePlan, insuranceType, audiA6);
Aero.confirmBooking();

/*user will purchase insurance and app confirms booking*/

Contract cncAL = aero.generateContract();
aero.createInvoice();
cncAL.acceptAndSign();
louis.makePayment(creditCardDetails);
aero.confirmPickUpDateTime();
/*new contract is created and user makes payment*/

(when close to return date)
aero.confirmReturnDateTime();
If(louis.extendContract())
        cncAL.extend(returnEndDate);
        insure. renewPlan (insuranceNumber)
aero.createInvoice();
louis.makePayment();
```

4) Design a Parking lot:

ParkingLot	Data: kiosk, parkingCapacity, floorList, exitCounter Behavior: generateTicket, lotIsFull, generateBill
ParkingSpot	Data: spotNumber, spotSize Behavior: isVacant, matchVehicleSize, occupied
Floor	Data: floorNumber, spotList Behavior: isAccessible, isFull
Ticket	Data: date, time , duration, ticketStatus, ticketNumber Behavior: calculateCost, calculateDuration
Vehicle	Data: type, licenseNumber, vehicleSize, entryTime Behavior: driveToKiosk, enterLot, enterFloor, exitFloor, park, depark, driveToCounter, driverMakePayment, exitLot
Payment	Data: amountDue Behavior: NA

Scenario:

```
hondaCar Vehicle;
parkNShop ParkingLot;

hondaCar.driveToKiosk (kiosk);
if(not parkNShop.lotIsFull())
        Ticket tln = parkNShop.generateTicket(entryTime);
hondaCar.enterLot();
/*vehicle drive to kiosk and generates ticket*/

Loop(floor : parkNShop.floorList)
        If(floor.isAccesible())
                hondaCar.enterFloor(floor);
                Loop(spot : floor.spotList)
                        If(not spot.isVacant() or not spot.matchVehicleSize(vehicleSize))
                                Continue
                        Else
                                hondaCar.park(spotNumber);
                                spot.occupied();
                                end
                end
end
/*vehicle loops through floors and spots to find empty spot*/


(while exiting lot)
hondaCar.depark(spotNumber);
hondaCar.exitFloor(floorNumber);
hondaCar.driveToCounter(exitCounter);
duration = tln.calculateDuration(entryTime, currentTime);
cost = tln.calculateCost(duration);
parkNShop.generateBill(cost, ticketNumber, licenseNumber);
Payment pay = hondaCar.driverMakePayment(amountDue);
hondaCar.exitLot();
```

5) Design a Traffic Controller System for a Junction:

Controller	Data: Behavior: setToGreen, setToRed, setToYellow, startTimer, endTimer
Signal	Data: longInterval, shortInterval Behavior: turnGreen, turnRed, turnYellow
sensor	Data: sensorNumber Behavior: detectVehicle
Traffic	Data: listOfCars, Behavior: move, stop

Scenario:

```
cnt Controller;
nsTraffic Traffic;
ewTraffic Traffic;
ewLeftTurnTraffic Traffic;
nsSignal Signal;
ewLeftTurnSignal Signal;
ewSignal signal;
sns sensor;

ewTraffic.stop();
cnt.setToRed(ewSignal);
ewSignal.turnRed();
/*The east west signal turns red*/

cnt.startTimer(longInterval);
cnt.setToGreen(nsSignal);
nsSignal.turnGreen();
nsTraffic.move();
cnt.stopTimer();
/*the north south signal turns green for normal Interval*/

cnt.startTimer(shortInterval);
cnt.setToYellow(nsSignal);
cnt.stopTimer();
nsTraffic.stop();
/*the ns signal turns yellown for short interval*/

cnt.setToRed(nsSignal);
nsSignal.turnRed();
/*the ns signal turns red*/

/*the sensor detects vehicle for left turn for east west traffic*/
If(sns.detectVehicle(ewLeftTurnTraffic))
        cnt.startTimer(shortInterval);
        cnt.setToGreen(ewLeftTurnSignal);
        ewLeftTurnSignal.turnGreen();
        ewLeftTurnTraffic.move();
        cnt.stopTimer();

        cnt.startTimer(shortInterval);
        cnt.setToYellow(ewLeftTurnSignal);
        cnt.stopTimer();
        ewLeftTurnTraffic.stop();

        cnt.setToRed(nsLeftTurnSignal);
        ewLeftTurnSignal.turnRed();
```