

Loyalty Program Information System

Requirements

The Royal and Loyal (R&L) company handles loyalty programs for companies that offer their customers various kinds of bonuses (E.g., bonus points, air miles, reduced rates, a larger car for the rate of a smaller one in a Car Hire company, extra or better service on an Airline etc.). Anything a company is willing to offer can be a service rendered in a loyalty program.

The central class in the UML model, below, is `LoyaltyProgram`. A system that administers a single loyalty program will contain only one object of this class. A company that offers its customers a membership in a loyalty program is called a `ProgramPartner`. More than one company can enter into the same program. In such a case, customers who enter the loyalty program can profit from services rendered by any of the participating companies.

Every customer of every program partner can enter the loyalty program by filling in a form and obtaining a membership card. The objects of class `Customer` represent people who have entered the program. The membership card, represented by the class `CustomerCard`, is issued to one person. Card use is not checked so the card could be used as a family or business card. Most loyalty programs allow customers to save bonus points. Each individual program partner decides when and how many bonus points are allotted for a certain purchase. Saved bonus points can be used to “buy” specific services from one of the program partners. To account for the bonus points that are saved by the customer, every membership can be associated with a `LoyaltyAccount`.

Various transactions on this account are possible. For example, the loyalty program “Silver and Gold” has 4 program partners: a supermarket, a chain of petrol stations, a car rental service and an airline:

- At the supermarket a customer earns 5 bonus points for £1 spent above £10: bonus points can be redeemed for items buyable only with bonus points
- The petrol stations offer a discount of 5% on every bill
- The Car Hire company offers 20 bonus points for every £50 spent
- Customers can save bonus points for free flights with the Airline. For every normal flight the airline offers 1 bonus point for each 15 miles of flight.

In this scenario, there are two types of transactions. First, there are transactions in which the customer obtains bonus points. These are `Earning` transactions. Second, there are transactions where the customer consumes bonus points. These are `Burning` transactions. The petrol stations offer discounts to members, but neither offer nor accept bonus points and therefore do not render transactions on the `LoyaltyAccount`.

Customers in the Silver and Gold program who make extensive use of membership are rewarded by a higher level of service: the gold card.

They are also offered additional services:

- Every 2 months the supermarket offers 'Gold Card' customers a completely free item worth about £10 for which not even bonus points are needed
- The petrol stations offer a 10% discount on every purchase
- The Car Rental company offers a larger car for the same price
- The Airline offers a business class trip for the price of an economy class ticket

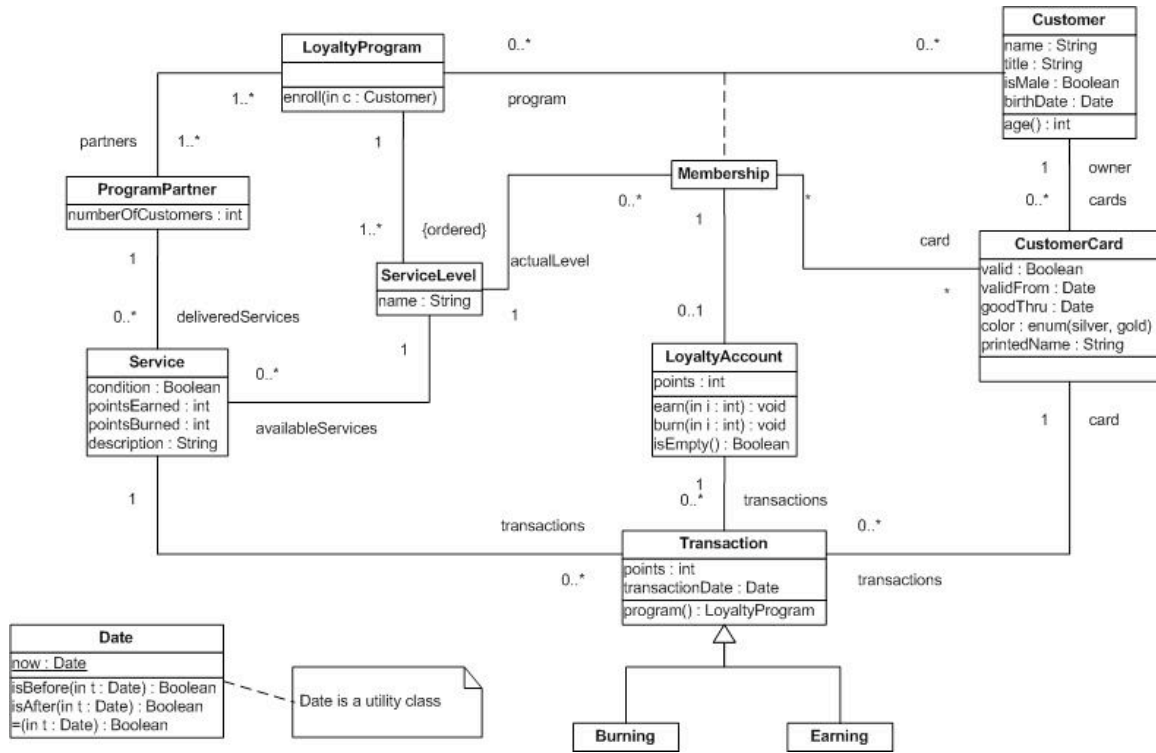
Customers must meet at least one of the following conditions to get a Gold Card:

- Three sequential years of membership with an average annual spend of £2000
- One year of membership with a spend of £6000

To administer different levels of service, the class `ServiceLevel` is introduced in the model. A service level is defined by the loyalty program and used for each membership.

R&L advertises the program and its conditions. It manages all customer data and transactions on the loyalty accounts. For this purpose, program partners must inform R&L of all transactions on loyalty program membership cards. Each year, R&L sends new membership cards to all customers. When appropriate R&L upgrades a membership to a Gold Card. In this case, R&L sends out the new card with a list of additional services and invalidates the old membership card. A customer can withdraw from the program by submitting a withdrawal form to R&L. Any remaining bonus points are cancelled and the card invalidated. R&L can invalidate a membership when the customer has not used the membership card for a certain period. For the Silver and Gold program this period is one year.

Class Diagram



Questions

Write the following constraints in OCL, in each case making the context clear:

1. Customers must have a minimum age of 18 years
2. The `CustomerCard`'s 'valid from' date must be earlier than the 'valid to' date
3. A card issued for membership must be for a customer mentioned in the membership
4. The number of service levels is exactly two
5. The printed name on the customer card must be a title followed by the registered name of the customer
6. The number of valid cards for each customer must be equal to the number of programs in which they participate
7. When a `LoyaltyProgram` does not earn or burn points then its members do not have `LoyaltyAccounts`
8. The number of customers of a `ProgramPartner` is the number who participate in one or more loyalty programs offered by this `ProgramPartner`
9. The first `ServiceLevel` is 'silver'
10. The upper limit of points that can be burned for each `ProgramPartner` is 10,000

Express what the following OCL expressions mean in English.

11.

```
context Customer
  inv: title = (if isMale = true
                then 'Mr.'
                else 'Ms.'
                endif)
```
12.

```
context Transaction
  inv: customerCard.membership.program->
  includes(self.program())
```
13.

```
context Membership
  inv: LoyaltyProgram.serviceLevel ->
  includes(actualLevel)
```
14.

```
context LoyaltyProgram
```

```
inv: serviceLevel->includesAll(membership.actualLevel)
```

15. context LoyaltyAccount
inv: transaction->collect(points) ->
exists (p:Integer | p > 500)
16. context LoyaltyAccount
inv: points > 0 implies transaction->exists (points > 0)
17. context ProgramPartner
inv: self.services.transaction ->
select(oclType=Burning) -> collect(points)->sum()
<= self.services transaction->select(oclType=Earning)
-> collect(points) -> sum()

Write pre- and post- conditions for:

18. The isEmpty() operation of LoyaltyAccount
19. The enrol() operation of LoyaltyProgram. Postconditions include that there is now one more customer than before and that the new customer's loyalty account has no points and no transactions
20. The program() query on Transaction
21. The burn() operation on LoyaltyAccount
22. The age() operation on Customer assuming participants must be adults below retirement age, which is 65

Answers

1. context Customer

inv: age() >= 18

2. context CustomerCard inv

validFrom.isBefore(ValidTo)

3. context Membership

inv: card.customer = customer

4. context LoyaltyProgram

inv: serviceLevel->size() = 2

5. context CustomerCard

inv: printedName = customer.title.concat(customer.name)

6. context Customer

inv: program->size = cards->select(valid = true)->size()

7. context LoyaltyProgram

inv: partners.deliveredServices->forAll(
pointsEarned = 0 and pointsBurned = 0)
implies membership.loyaltyAccount->isEmpty

8. context ProgramPartner

inv: numberOfCustomers = loyaltyProgram.customer->asSet-
>size()

9. context LoyaltyProgram

inv: serviceLevel ->first.name = 'Silver'

10. context ProgramPartners inv:

let tc = deliveredServices->Burning in
collect(tc.points)->sum() <= 10,000

11. A customer's title is 'Mr.' if he is male, otherwise it is 'Ms.'

12. program() returns a for which a CustomerCard has been issued to the member

13. The actual service level of a membership must be a service level offered by the `LoyaltyProgram`
14. The set of service levels for a `LoyaltyProgram` must include the set of all `actualLevels` of memberships (meaning is similar to 13, but here is expressed from the context of `LoyaltyProgram` rather than `Membership`)
15. The set of transactions in a `LoyaltyAccount` that contain points must include at least one transaction of more than 500 points
16. The existence of points in a `LoyaltyAccount` implies at least one transaction that contained points
17. The sum of points burned for a `ProgramPartner`'s offered services must never exceed the number of points earned from its services
18.

```
context LoyaltyAccount :: isEmpty(): Boolean
pre: -- none
post : result = (points=0)
```
19.

```
context LoyaltyProgram::enrol(in c: Customer)
pre: not customer->includes(c)
post: customer = customer@pre->including(c)
post: membership->select(customer = c)->forall(
loyaltyAccount->notEmpty() and
loyaltyAccount.points = 0 and
loyaltyAccount.transactions ->isEmpty())
```
20.

```
context Transaction::program(): LoyaltyProgram
pre: -- none
post: result = self.card.membership.program
```
21.

```
context LoyaltyAccount::burn(in i: Integer)
pre: points >= i
post: points = points@pre - i
```
22.

```
context Customer::age(): Integer
pre: -- none
post: result >= 18 and result < 65
```