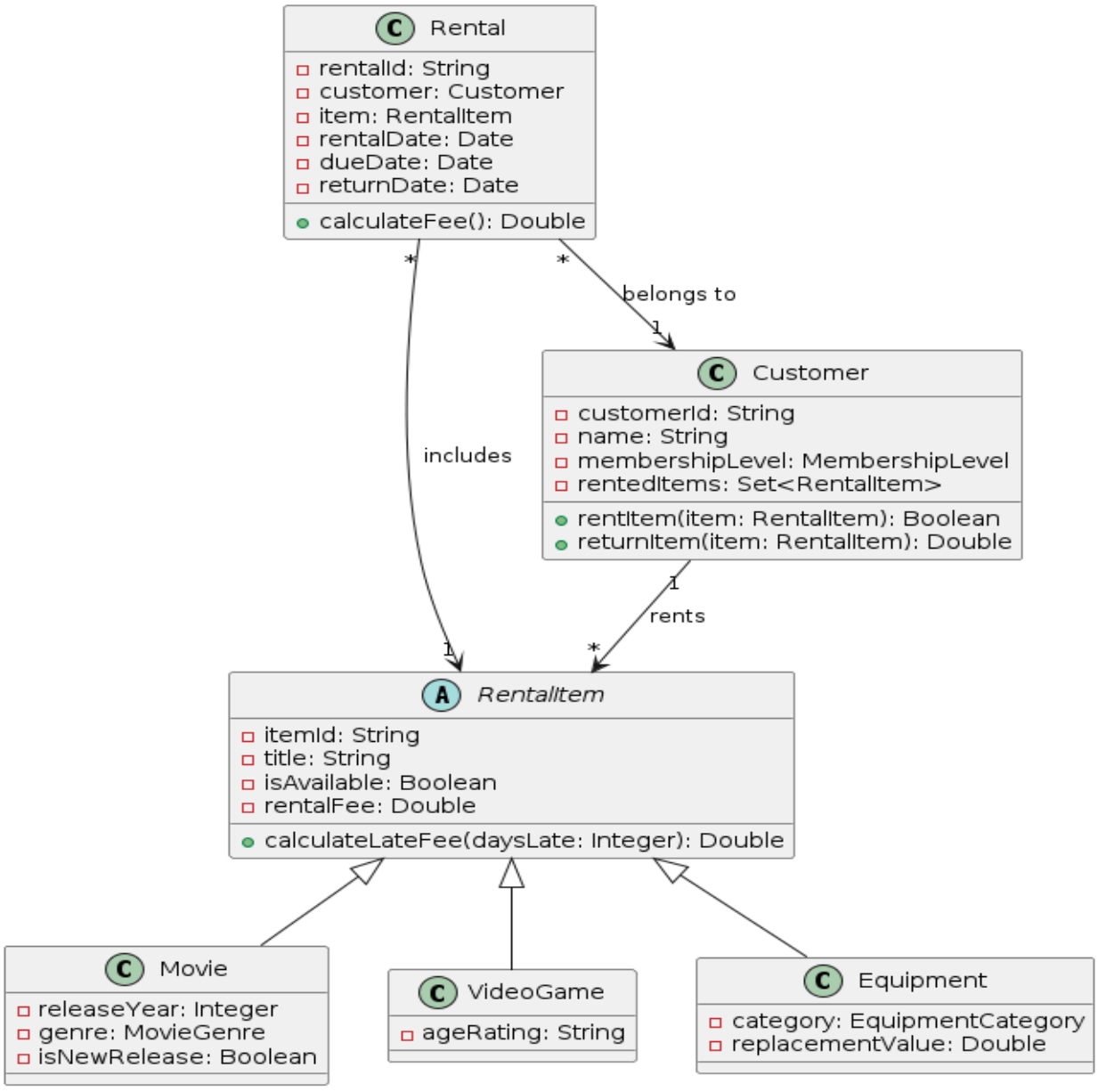
Problem - Quiz A



Quiz - 4

You are given 7 constraints written in OCL. Each contains a specific code smell. Your task is to: Identify the bad smell and give a solution for it.

Part 1: If a movie is a new release and has a rental fee above $5.00, and if it's an Action movie, then its rental fee must be more than $7.00.

```
context Movie
inv newReleaseHigherFee:
self.isNewRelease implies (self.rentalFee > 5.0 implies
(self.genre = MovieGenre::ACTION implies self.rentalFee > 7.0))
```

Part 2: If a customer rents an item that costs more than $10.00, then that item must only be rented by customers with PREMIUM membership

```
context Customer
inv premiumMembersOnlyRentExclusiveItems:
   self.rentedItems->forAll(item |
     item.rentalFee > 10.0 implies
     item.rentedItems->forAll(r |
       r.customer.membershipLevel = MembershipLevel::PREMIUM))
```

Part 3: Return true if the item is available; otherwise, return false.

```
context RentalItem
inv AvailabilityCheck:
   if self.isAvailable = true then
      true
   else
      false
   endif
```

Part 4:For a given rental, all of the rented items belong to that rental and each of those rented items must have the title "AMovieTitle".

context Rental
inv: self.customer.rentedItems->forAll(item |
  item.rental = self and
  self.item.title = "AMovieTitle")

Solutions

Solution 1: Implies Chain

Refactoring: Extract Method/Decompose Conditional

```
context Movie
inv highFeeForNewActionMovies:
    self.isActionNewRelease() implies self.rentalFee > 7.0

context Movie
def: isActionNewRelease(): Boolean =
    self.isNewRelease and self.genre = MovieGenre::ACTION
```

Solution 2: ForAll Chain

Refactoring: Extract Method/Query and Simplify Quantifiers

```
context Customer
inv premiumMembersOnlyRentExclusiveItems:
    self.hasExclusiveItems() implies self.membershipLevel =
MembershipLevel::PREMIUM

context Customer
def: hasExclusiveItems(): Boolean =
    self.rentedItems->exists(item | item.rentalFee > 10.0)
```

Solution 3: Verbose Expression

Refactoring: Simplify Conditional Logic

```
self.isAvailable
```

Solution 4: Long Journey

Refactoring:context Rental

inv: self.item.oclIsTypeOf(Movie)

Solution 5: Duplication

Refactoring: Pull Up Method/Extract Common Logic

```
context RentalItem
inv calculateStandardLateFee:
    self.calculateLateFee(5) = self.rentalFee * 1.5 * 5
```

Solution 6: Downcasting

Refactoring: Replace Conditional with Polymorphism

```
context Customer::returnItem(item: RentalItem): Double
post calculateReturnFee:
    result = item.calculateReturnFee()

context RentalItem
def: calculateReturnFee(): Double = self.rentalFee

context Movie
def: calculateReturnFee(): Double =
    self.rentalFee * (if self.isNewRelease then 1.5 else 1.0 endif)

context VideoGame
def: calculateReturnFee(): Double = self.rentalFee * 1.2
```

Solution 7: Type-Related Conditionals

Refactoring: Replace Type Code with Subclasses/Use Polymorphism

```
context RentalItem
inv minimumRentalFee:
    self.rentalFee >= self.getMinimumFee()

context RentalItem
def: getMinimumFee(): Double = 0.0

context Movie
def: getMinimumFee(): Double = 3.0

context VideoGame
def: getMinimumFee(): Double = 5.0

context Equipment
def: getMinimumFee(): Double = 10.0
```