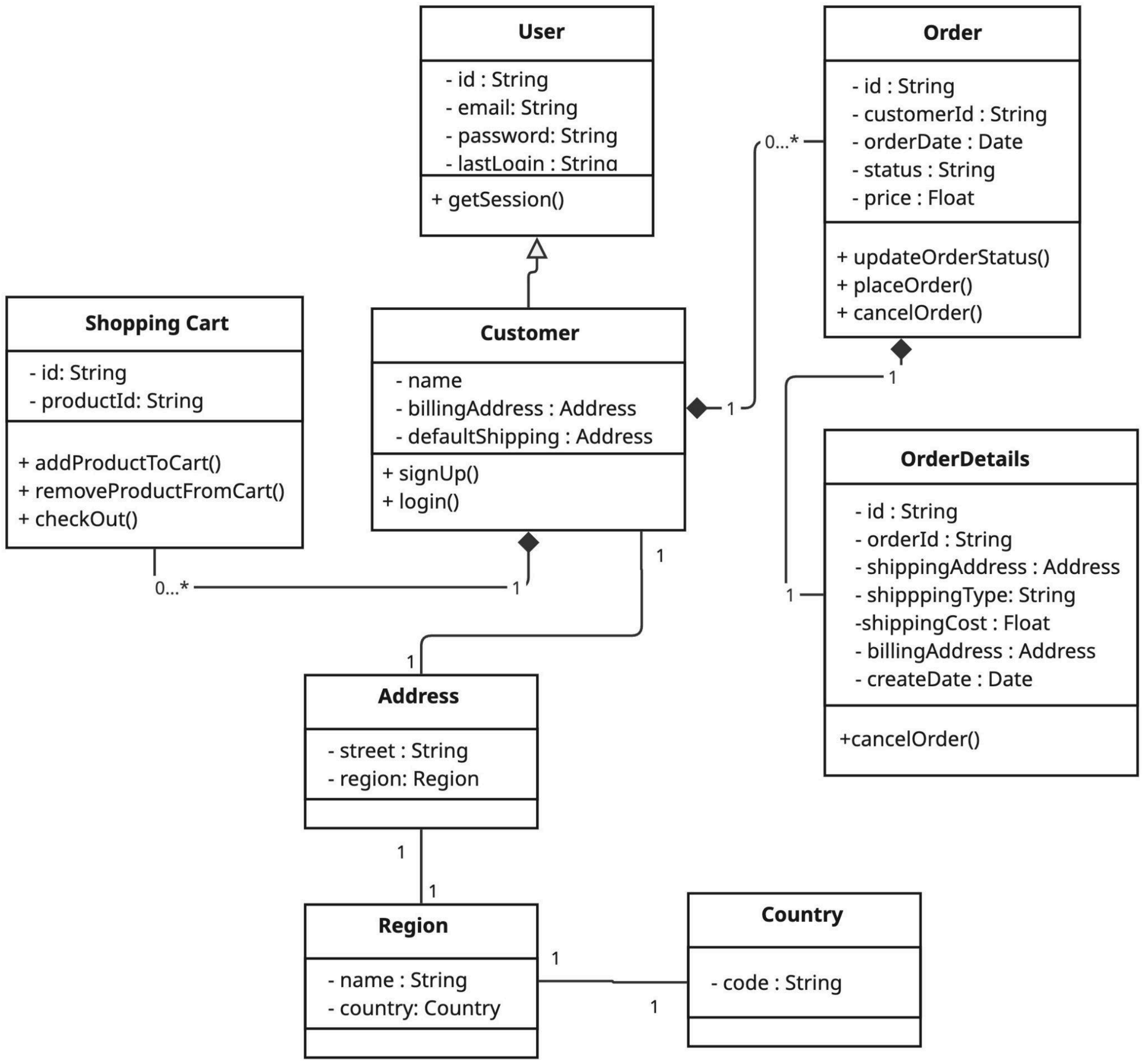
Problem - Quiz B



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.

1) If an order exists, and its price is above 1000, and it's pending, then the customer must have logged in within the last 24 hours.

```
context Order
inv:
  self.price > 1000 implies
  self.status = 'Pending' implies
  self.customer.lastLogin >= Date.now() - 1
```

2) Each customer's orders must all have valid details and those details must have a non-empty shipping address.

```
context Customer
inv:
  self.orders->forAll(o | o.details->forAll(d |
     d.shippingAddress <> ''  ) )
```

3) If a cart has products, the user must not be a guest.

```
context ShoppingCart
inv:
  if self.productId->size() > 0 then
    self.customer.user.email <> 'guest@example.com'
  else    true   endif
```

4) All orders must have a status and a price defined, and no order should be in 'Canceled' state.

```
context Order
inv:
  self.status <> '' and self.price > 0 and self.status <> 'Canceled'
```

5) If a user is a customer, they must have a billing address.

```
context User
inv:
  self.oclIsTypeOf(Customer) implies  self.oclAsType(Customer).billingAddress <>""
```

6) If the shipping type is 'Express', then shipping cost should be over 20.

```
context OrderDetails
inv:
  if self.shippingType = 'Express' then  self.shippingCost > 20
  else   true endif
```

7) Every Order's customer's default shipping address must be in Pakistan.

```
context Order
inv:
   self.customer.defaultShipping.region.country.code = 'PK'
```

Solution

1. Implies Chain

```
context Order
inv:
  self.price > 1000 and self.status = 'Pending' implies
  self.customer.lastLogin >= Date.now() - 1
```

Smell: Implies chain (A implies B implies C)

Refactoring:

Decompose Conditional: Breaks chained implications into logical conjunction before implication for clarity.

---

2. ForAll Chain

```
context Customer
inv:
  self.orders.details->forAll(d |
    d.shippingAddress <> ''
  )
```

Smell: Nested forAll

Refactoring:

Flatten forAll Chain / **Change Navigation Root**: Replaces deep nested forAll with simpler navigation using combined paths.

---

3. Verbose Expression

```
context ShoppingCart
inv:
  (self.productId->size() > 0) implies
  self.customer.user.email <> 'guest@example.com'
```

Smell: Redundant if-then-else returning a boolean

Refactoring:

Simplify Conditional: Converts if-then-else to direct logical expression using implies.

---

4. Duplication

```
context Order
inv:
  let validStatus = self.status <> ""  and
                    self.status <> 'Canceled' in
                              validStatus and self.price > 0
```

Smell: Repeated field checks (status repeated twice)
Refactoring:
Extract Variable (via let) to avoid repeating logic.

---

5. Downcasting

```
context Customer
inv:
  self.billingAddress <> ''
```

Smell: Use of oclAsType()
Refactoring:
Move Constraint to Subclass: Shifts logic directly to Customer class to avoid type checks.

---

6. Type-Based Conditionals

```
context OrderDetails
inv:
  self.shippingType = 'Express' implies self.shippingCost > 20
```

Smell: if-then-else to check type and apply condition
Refactoring:
Replace Conditional with Implication: Makes logic declarative, concise, and easier to verify.

---

7. Long Journey

Smell: Long Journey – chaining through 4 associations in one go.
Refactoring: Extract a helper on `Address` to collapse the chain:

Refactoring 1: Extract Local Variable

```
context Order
inv deepShippingCountry:
  let c = self.customer.defaultShipping.region.country
  in c.code = 'PK'
```

Refactoring 2: Introduce Query Helper

```
context Customer
def: countryCode : String =
  self.defaultShipping.region.country.code

context Order
inv deepShippingCountry:
  self.customer.countryCode = 'PK'
```