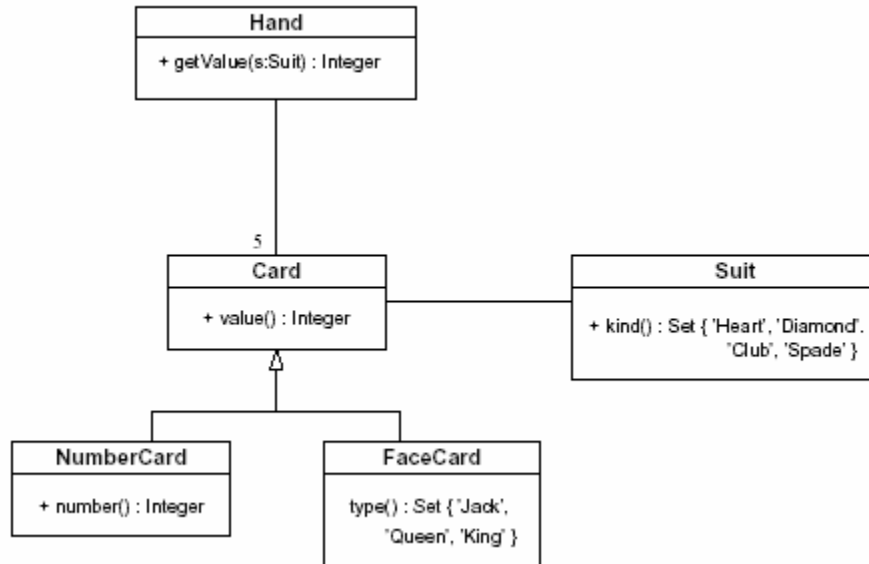


**CS590L: Distributed Component Architectures
Winter 2004
Quiz2 (Suggested Solutions)**

Name:



Here, a Hand consists of five Cards, each of which is either a numbered card (with values from 1 to 10 inclusive), or a “face” card. The “face” cards have a type field, which is one of 'Jack', 'Queen' or 'King'. Each card also has a Suit, which is one of 'Hearts', 'Diamonds', 'Spade' or 'Club'.

Specify the following using the Object Constraint Language (OCL); be sure to explicitly indicate the context in each case:

(a) Specify the following as class invariants:

- In class NumberCard, the result returned by number() is always between 1 and 10 inclusive

`NumberCard::self.number() >=1 and self.number() <= 10`

- In class Card, the value returned by value() is always between 1 and 13 inclusive

`Card::self.value() >=1 and self.value() <=13`

- There must be at least one card in a hand whose suit is 'Heart'

`Hand::self.card -> exist (c: Card | c.suit.kind() = 'Heart')`

(b) Define the method value in each subclass of Card. The method value is overridden in both subclasses of Card. For a NumberCard it returns the number on

the card, whereas for a FaceCard it returns 11 for 'Jack', 12 for 'Queen' and 13 for 'King'.

```
FaceCard::self.kind() = 'Jack' implies self.value() = 11
FaceCard::self.kind() = 'Queen' implies self.value() = 12
FaceCard::self.kind() = 'King' implies self.value() = 13
NumberCard::self.vaue() = self.number()
```

(c) Define the method `getValue` in class `Hand`.

- This method `getValue` in `Hand` takes a `Suit` as input, called the *trump* Suit.

```
Hand::getValue (trump: Suit)
```

- The value of the hand is the sum of the values of all cards in the hand, with cards from the trump suit counted twice.

```
Hand::self.value() =
self->sum(self.getValue(self.card)) + self.getValue(self.card.kind() = 'trump')
```

- The OCL method `sum()` will sum all the elements in a `Collection of Integers`.
`Collect(I)::sum(I->Integer):sum(I)`