Advanced Object-oriented Programming

Lecture 7

CardHand
A Hand of Cards

Recall that each player in Battle of the Bands has some number of cards: a hand.

Initially, the pack is dealt out to the two players; at the end of each round, the loser of the round loses their top card; the winner gains the loser’s card, and places both cards at the bottom of their hand.
Recall that each player in Battle of the Bands has some number of cards: a hand.

Initially, the pack is dealt out to the two players; at the end of each round, the loser of the round loses their top card; the winner gains the loser’s card, and places both cards at the bottom of their hand.
A Hand of Cards

Recall that each player in Battle of the Bands has some number of cards: a *hand*.

Initially, the pack is dealt out to the two players; at the end of each round, the loser of the round loses their top card; the winner gains the loser’s card, and places both cards at the bottom of their hand.
Recall that each player in Battle of the Bands has some number of cards: a *hand*.

Initially, the pack is dealt out to the two players; at the end of each round, the loser of the round loses their top card; the winner gains the loser’s card, and places both cards at the bottom of their hand.
A Card Hand is a sequence of Band Cards.

There are operations to:

- add a card to the bottom of the hand;
- look at the card at the top of the hand;
- remove the card from the top of the hand; and
- test whether the hand is empty (losing condition).
ADT: CardHand

A *Card Hand* is a sequence of *Band Cards*.

There are operations to:

- add a card to the bottom of the hand;
- look at the card at the top of the hand;
- remove the card from the top of the hand; and
- test whether the hand is empty (losing condition).
A *Card Hand* is a sequence of *Band Cards*. There are operations to:

- add a card to the bottom of the hand;
- look at the card at the top of the hand;
- remove the card from the top of the hand; and
- test whether the hand is empty (losing condition).
A *Card Hand* is a sequence of *Band Cards*.

There are operations to:

- add a card to the bottom of the hand;
- look at the card at the top of the hand;
- remove the card from the top of the hand; and
- test whether the hand is empty (losing condition).
ADT: CardHand

A *Card Hand* is a sequence of *Band Cards*.

There are operations to:

- add a card to the bottom of the hand;
- look at the card at the top of the hand;
- remove the card from the top of the hand; and
- test whether the hand is empty (losing condition).
*** load file BandCard.maude

in BandCard *** NB: no full stop!

fmod CARD_HAND is

    protecting BAND_CARD.

sort CardHand.

op empty : → CardHand.


op getTop : CardHand → BandCard.

op removeTop : CardHand → CardHand.

op isEmpty : CardHand → Bool.
fmod CARD_HAND is

    protecting BAND_CARD.

    sort CardHand.

    op empty : → CardHand.
    op getTop : CardHand → BandCard.
    op removeTop : CardHand → CardHand.
    op isEmpty : CardHand → Bool.
*** load file BandCard.maude

in BandCard *** NB: no full stop!

fmod CARD_HAND is

  protecting BAND_CARD .

sort CardHand .

op empty : CardHand .


op getTop : CardHand -> BandCard .

op removeTop : CardHand -> CardHand .

op isEmpty : CardHand -> Bool .

*** load file BandCard.maude

in BandCard

*** NB: no full stop!

fmod CARD_HAND is

protecting BAND_CARD.

sort CardHand.

op empty : -> CardHand.


op getTop : CardHand -> BandCard.

op removeTop : CardHand -> CardHand.

op isEmpty : CardHand -> Bool.
File: CardHand.maude

*** load file BandCard.maude

in BandCard *** NB: no full stop!

fmod CARD_HAND is

  protecting BAND_CARD.

sort CardHand.

op empty : CardHand.
op getTop : CardHand -> BandCard.
op removeTop : CardHand -> CardHand.
op isEmpty : CardHand -> Bool.
*** load file BandCard.maude

in BandCard

*** NB: no full stop!

fmod CARD_HAND is

protecting BAND_CARD.

sort CardHand.

op empty : CardHand.


op getTop : CardHand -> BandCard.

op removeTop : CardHand -> CardHand.

op isEmpty : CardHand -> Bool.
*** load file BandCard.maude

in BandCard *** NB: no full stop!

fmod CARD_HAND is

  protecting BAND_CARD.

sort CardHand.

op empty : -> CardHand.


op getTop : CardHand -> BandCard.

op removeTop : CardHand -> CardHand.

op isEmpty : CardHand -> Bool.
File: CardHand.maude

```maude
*** load file BandCard.maude
in BandCard *** NB: no full stop!

fmod CARD_HAND is

  protecting BAND_CARD .

sort CardHand .

op empty : → CardHand .


op getTop : CardHand → BandCard .

op removeTop : CardHand → CardHand .

op isEmpty : CardHand → Bool .
```
File: CardHand.maude

*** load file BandCard.maude

in BandCard *** NB: no full stop!

fmod CARD_HAND is

  protecting BAND_CARD.

  sort CardHand.

  op empty : -> CardHand.
  op getTop : CardHand -> BandCard.
  op removeTop : CardHand -> CardHand.
  op isEmpty : CardHand -> Bool.
**BOOL and Bool**

*Bool* is Maude’s built-in sort of Booleans (true, false). This sort is specified in the built-in module *BOOL*.

Every module, by default, imports *BOOL*; i.e., implicitly contains the declaration

```maude
protecting BOOL .
```

The reason for this is that Booleans are used in *conditional equations* . . . which we’ll see soon!
**BOOL and Bool**

*Bool* is Maude’s built-in sort of Booleans (true, false).

This sort is specified in the built-in module *BOOL*.

Every module, by default, imports *BOOL*; i.e., implicitly contains the declaration

```maude
protecting BOOL .
```

The reason for this is that Booleans are used in **conditional equations** . . . which we’ll see soon!
**BOOL and Bool**

`Bool` is Maude’s built-in sort of Booleans (true, false). This sort is specified in the built-in module `BOOL`.

Every module, by default, imports `BOOL`; i.e., implicitly contains the declaration

```
protecting BOOL .
```

The reason for this is that Booleans are used in **conditional equations** . . . which we’ll see soon!
More Maude fmod CARD_HAND

CARD_HAND contd.

```maude
definition
var C : CardHand .
var B : BandCard .

eq isEmpty(C) = C == empty .
cq getTop(add(C, B)) = B if isEmpty(C) .
```

‘==’ is Maude’s Boolean equality test keyword
cq stands for conditional equation keyword
if introduces the condition
what follows ‘if’ is the condition (a term of sort Bool)
More Maude fmod CARD_HAND

CARD_HAND contd.

\[
\begin{align*}
\text{var} & \quad C : \text{CardHand} . \\
\text{var} & \quad B : \text{BandCard} . \\
\text{eq} & \quad \text{isEmpty}(C) = C == \text{empty} . \\
\text{cq} & \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C) .
\end{align*}
\]

‘==’ is Maude’s Boolean equality test
keyword \textit{cq} stands for \textit{conditional equation}
keyword \textit{if} introduces the condition
what follows ‘if’ is the \textit{condition} (a term of sort \textit{Bool})
More Maude fmod CARD_HAND

CARD_HAND contd.

\[\begin{align*}
\text{var} & \quad C : \text{CardHand} . \\
\text{var} & \quad B : \text{BandCard} . \\
\text{eq} & \quad \text{isEmpty}(C) = C == \text{empty} . \\
\text{cq} & \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C) .
\end{align*}\]

‘==’ is Maude’s Boolean equality test keyword \(\text{cq}\) stands for conditional equation keyword \(\text{if}\) introduces the condition what follows ‘if’ is the condition (a term of sort \text{Bool}\)
More Maude fmod CARD_HAND

CARD_HAND contd.

\[
\begin{align*}
\text{var} & \quad C : \text{CardHand} . \\
\text{var} & \quad B : \text{BandCard} . \\
\text{eq} & \quad \text{isEmpty}(C) = C == \text{empty} . \\
\text{cq} & \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C) .
\end{align*}
\]

‘==’ is Maude’s Boolean equality test

keyword \texttt{cq} stands for \texttt{conditional equation}

keyword \texttt{if} introduces the condition

what follows ‘if’ is the \textit{condition} (a term of sort \textit{Bool})
More Maude fmod CARD_HAND

CARD_HAND contd.

\[
\begin{align*}
\text{var} & \quad C : \text{CardHand} . \\
\text{var} & \quad B : \text{BandCard} . \\
\text{eq} & \quad \text{isEmpty}(C) = C \equiv \text{empty} . \\
\text{cq} & \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C) .
\end{align*}
\]

‘==’ is Maude’s Boolean equality test keyword cq stands for conditional equation keyword if introduces the condition what follows ‘if’ is the condition (a term of sort Bool)
More Maude fmod CARD_HAND

CARD_HAND contd.

\begin{align*}
\textbf{var} & \quad C : \text{CardHand} . \\
\textbf{var} & \quad B : \text{BandCard} . \\
\textbf{eq} & \quad \text{isEmpty}(C) = C == \text{empty} . \\
\textbf{cq} & \quad \text{getTop}(\text{add}(C, B)) = B \quad \textbf{if} \quad \text{isEmpty}(C) .
\end{align*}

‘==’ is Maude’s Boolean equality test keyword \textbf{cq} stays for \textbf{conditional equation} keyword \textbf{if} introduces the condition what follows ‘if’ is the \textit{condition} (a term of sort \textit{Bool})
We’ve seen that equations simplify *terms*:

```plaintext
in the Maude interpreter
reduce pop(push(3, push(2, pop(push(1, push(0, empty))))))) .
result: push(2, push(0, empty))
```

...and that this corresponds to *computation*:

```plaintext
Stack st = new Stack();
st.push(0);
st.push(1);
st.pop();
st.push(2);
st.push(3);
st.pop();
```
We’ve seen that equations simplify terms:

\[
\text{reduce} \quad \text{pop(push(3, push(2, pop(push(1, push(0, empty))']))))}.
\]

result: push(2, push(0, empty))

... and that this corresponds to computation:

```java
Stack st = new Stack();
st.push(0);
st.push(1);
st.pop();
st.push(2);
st.push(3);
st.pop();
```
We’ve seen that equations simplify terms:

in the Maude interpreter

reduce pop(push(3, push(2, pop(push(1, push(0, empty))))))) .
result: push(2, push(0, empty))

... and that this corresponds to computation:

Stack st = new Stack();
st.push(0);
st.push(1);
st.pop();
st.push(2);
st.push(3);
st.pop();
Equations

We’ve seen that equations simplify *terms*:

```plaintext
reduce pop(push(3, push(2, pop(push(1, push(0, empty))))))
result: push(2, push(0, empty))
```

... and that this corresponds to *computation*:

```java
Stack st = new Stack();
st.push(0);
st.push(1);
st.pop();
st.push(2);
st.push(3);
st.pop();
```
Conditional equations are only applied if the condition is true. For example:

\[
\text{cq} \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C).
\]

\[
\text{getTop}( \text{add}(\text{empty}, \text{Bandcard("Goldfrapp", 5, 3, 7, 23, 12)})) = \text{Bandcard("Goldfrapp", 5, 3, 7, 23, 12)}
\]

\[
\text{isEmpty}(\text{empty}) = \text{empty} == \text{empty} = \text{true}
\]
Conditional equations are only applied if the condition is true. For example:

```
cq \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C).
```

```
\text{getTop}(\text{add}(\text{empty}, \text{Bandcard}("Goldfrapp", 5, 3, 7, 23, 12)))
= \text{Bandcard}("Goldfrapp", 5, 3, 7, 23, 12)
```

The Condition

```
\text{isEmpty}(\text{empty})
= \text{empty} == \text{empty}
= \text{true}
```
Conditional Equations

Conditional equations are only applied if the condition is true. For example:

\[
\text{cq} \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C).
\]

\[
\text{getTop}(\text{add}(\text{empty}, \text{Bandcard}("Goldfrapp", 5, 3, 7, 23, 12))) = \text{Bandcard}("Goldfrapp", 5, 3, 7, 23, 12)
\]

The Condition

\[
\text{isEmpty}(\text{empty}) = \text{empty} == \text{empty} = \text{true}
\]
Conditional equations are only applied if the condition is true. For example:

\[
\text{cq } \text{getTop}(\text{add}(C, B)) = B \text{ if isEmpty}(C).
\]

\[
\text{getTop( add( empty, Bandcard("Goldfrapp", 5, 3, 7, 23, 12)) ) } = \text{ Bandcard("Goldfrapp", 5, 3, 7, 23, 12)}
\]

\[
\text{isEmpty( empty ) } = \text{ empty } == \text{ empty } = \text{ true}
\]
Conditional equations are only applied if the condition is true. For example:

```plaintext
cq  getTop(add(C, B)) = B     if     isEmpty(C) .

getTop( add( empty,  Bandcard("Goldfrapp", 5, 3, 7, 23, 12) ) )

=  Bandcard("Goldfrapp", 5, 3, 7, 23, 12)

isEmpty(empty)

=  empty == empty

=  true
```

The Condition
Conditional Equations

Conditional equations are only applied if the condition is true. For example:

cq \( \text{getTop(add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C). \)

getTop( add( empty, Bandcard("Goldfrapp", 5, 3, 7, 23, 12) ) )

= Bandcard("Goldfrapp", 5, 3, 7, 23, 12)

The Condition

isEmpty(empty)

= empty == empty

= true
Conditional Equations

Conditional equations are only applied if the condition is true. For example:

```
cq  getTop(add(C, B)) = B  if  isEmpty(C) .
```

```
getTop( add( empty, Bandcard("Goldfrapp", 5, 3, 7, 23, 12) ) ) = Bandcard("Goldfrapp", 5, 3, 7, 23, 12)
```

```
isEmpty(empty)
 = empty == empty
 = true
```

The Condition
Conditional equations are only applied if the condition is true. For example:

\[
\text{cq } \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C).
\]

\[
\text{getTop( add( empty, Bandcard("Goldfrapp", 5, 3, 7, 23, 12) ) )} = \text{Bandcard("Goldfrapp", 5, 3, 7, 23, 12)}
\]

The Condition

\[
\text{isEmpty}(\text{empty}) = \text{empty == empty} = \text{true}
\]
Conditional equations are only applied if the condition is true. For example:

cq \( \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C) \).

getTop( \text{add}(\text{empty}, \text{Bandcard}("Goldfrapp", 5, 3, 7, 23, 12)) )

\[=\]  
\text{Bandcard}("Goldfrapp", 5, 3, 7, 23, 12)

The Condition

\text{isEmpty}(\text{empty})

\[=\]
\text{empty} == \text{empty}

\[=\]
\text{true}
Conditional equations are **only** applied if the condition is true. For example:

```
cq  getTop(add(C, B)) = B  if  isEmpty(C) .
```

```
getTop(add(add(empty, b1), b2))  can't apply the equation
```

```
The Condition
isEmpty(add(empty, b1))
=  add(empty, b1) == empty
=  false
```
**Conditional Equations**

Conditional equations are only applied if the condition is true. For example:

```plaintext
cq  getTop(add(C, B)) = B    if    isEmpty(C).
```

The equation `getTop(add(add(empty, b1), b2))` can't apply the equation because:

```plaintext
isEmpty(add(empty, b1))
= add(empty, b1) == empty
= false
```
Conditional equations are only applied if the condition is true. For example:

\[
\text{cq } \text{getTop(add(C, B))} = B \quad \text{if} \quad \text{isEmpty(C)}.
\]

getTop(add(add(empty, b1), b2)) can’t apply the equation

The Condition

\[
\text{isEmpty(add(empty, b1))}
\]

\[
= \quad \text{add(empty, b1) == empty}
\]

\[
= \quad \text{false}
\]
Conditional equations are only applied if the condition is true. For example:

\[
\begin{align*}
\text{cq } & \quad \text{getTop(add(C, B)) } = \text{ B } \quad \text{if } \quad \text{isEmpty(C)} . \\
& \text{getTop(add(add(empty, b1), b2)) } \quad \text{can’t apply the equation}
\end{align*}
\]

The Condition

\[
\text{isEmpty(add(empty, b1))}
\]

\[
\begin{align*}
& = \\
& \text{add(empty, b1) }\neq \text{ empty}
\end{align*}
\]

\[
\text{false}
\]
Conditional equations are only applied if the condition is true. For example:

\[
\text{cq} \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C).
\]

getTop(\text{add}(\text{add}(\text{empty}, b1), b2)) \quad \text{can’t apply the equation}

\[
\text{isEmpty}(\text{add}(\text{empty}, b1))
= \text{add}(\text{empty}, b1) == \text{empty}
= \text{false}
\]
Conditional equations are only applied if the condition is true. For example:

```plaintext
cq  getTop(add(C, B)) = B  if  isEmpty(C) .
```

getTop(add(add(empty, b1), b2))  can’t apply the equation

The Condition

isEmpty(add(empty, b1))
=  
add(empty, b1) == empty
=  
false
Conditional Equations

Conditional equations are only applied if the condition is true. For example:

\[
\text{cq } \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C).
\]

\[
\text{getTop}(\text{add}(\text{add}(\text{empty}, b1), b2)) \quad \text{can’t apply the equation}
\]

\[
\text{isEmpty}(\text{add}(\text{empty}, b1))
= \text{add}(\text{empty}, b1) == \text{empty}
= \text{false}
\]
**Conditional Equations**

Conditional equations are only applied if the condition is true. For example:

\[
\text{cq } \text{getTop(add(C, B))} = B \quad \text{if} \quad \text{isEmpty(C)}.
\]

getTop\(\text{add(add(empty, b1), b2)}\) can’t apply the equation

The Condition

\[
\text{isEmpty(add(empty, b1))}
\]

\[
= \text{add(empty, b1) == empty}
\]

\[
= \text{false}
\]
Note the recursion.

These two equations tell us how to evaluate `getTop` when its argument is a non-empty hand of cards (i.e., of the form `add(C,B)` for some `CardHand C` and `BandCard B`).
Note the recursion.

These two equations tell us how to evaluate \texttt{getTop} when its argument is a non-empty hand of cards (i.e., of the form \texttt{add}(C,B) for some \texttt{CardHand} C and \texttt{BandCard} B).
Yet More Maude fmod CARD_HAND

\begin{align*}
\text{var} & \quad C : \text{CardHand} . \\
\text{var} & \quad B : \text{BandCard} . \\
& \ldots \\
\text{cq} & \quad \text{getTop}(\text{add}(C, B)) = B \quad \text{if} \quad \text{isEmpty}(C) . \\
\text{cq} & \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not}(\text{isEmpty}(C)) .
\end{align*}

Note the \textbf{recursion}.

These two equations tell us how to evaluate \textsf{getTop} when its argument is a non-empty hand of cards (i.e., of the form \textsf{add}(C,B) for some \text{CardHand} \ C and \text{BandCard} \ B).
var C : CardHand.
var B : BandCard.

... 

cq getTop(add(C, B)) = B if isEmpty(C).
cq getTop(add(C, B)) = getTop(C) if not(isEmpty(C)).

Note the recursion.

These two equations tell us how to evaluate $\text{getTop}$ when its argument is a non-empty hand of cards (i.e., of the form $\text{add}(C,B)$ for some CardHand $C$ and BandCard $B$).
Conditional Equations

**cq** \( \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \text{ if } \neg \text{isEmpty}(C) \). \[
\text{getTop}(\text{add}(\text{add}(\text{empty}, b1), b2)) = \text{getTop}(\text{add}(\text{empty}, b1)) = b1
\]
The Condition
\[\neg \text{isEmpty}(\text{add}(\text{empty}, b1))) = \neg \text{false} = \text{true} \]
Conditional Equations

cq \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not(\text{isEmpty}(C))}.

\text{getTop}(\text{add}(\text{add}(\text{empty}, b1), b2))
= \quad \text{getTop}(\text{add}(\text{empty}, b1))
= \quad b1

\text{The Condition}

\text{not(\text{isEmpty}(\text{add}(\text{empty}, b1))))}
= \quad \text{not(false)}
= \quad \text{true}
Conditional Equations

\[ \text{cq} \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not(isEmpty}(C)) \ . \]

\[
\begin{align*}
\text{getTop}(\text{add}(\text{add}(\text{empty, b1}), b2)) \\
= \\
\text{getTop}(\text{add}(\text{empty, b1})) \\
= \\
b1
\end{align*}
\]

The Condition

\[
\begin{align*}
\text{not(isEmpty}(\text{add}(\text{empty, b1}))) \\
= \\
\text{not(false)} \\
= \\
true
\end{align*}
\]
Conditional Equations

cq \quad \text{getTop} (\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not} (\text{isEmpty}(C)) .

\text{getTop} (\text{add}(\text{add}(\text{empty}, b1), b2))
= \text{getTop}(\text{add}(\text{empty}, b1))
= b1

\text{The Condition}

\text{not} (\text{isEmpty}(\text{add}(\text{empty}, b1)))
= \text{not} (\text{false})
= \text{true}
Conditional Equations

\[
\text{cq} \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not(isEmpty}(C)) .
\]

\[
\text{getTop}(\text{add}(\text{add}(\text{empty, b1}), b2))
\]
\[
= \text{getTop}(\text{add}(\text{empty, b1}))
\]
\[
= b1
\]

\[
\text{The Condition}
\]
\[
\text{not(isEmpty}(\text{add}(\text{empty, b1})))
\]
\[
= \text{not(false)}
\]
\[
= \text{true}
\]
Conditional Equations

cq  \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \text{ if } \neg(\text{isEmpty}(C)) \\

\text{getTop}(\text{add}(\text{add}(\text{empty}, b_1), b_2)) = \text{getTop}(\text{add}(\text{empty}, b_1)) = b_1

\text{The Condition}

\neg(\text{isEmpty}(\text{add}(\text{empty}, b_1))) = \neg(\text{false}) = \text{true}
Conditional Equations

\[
\text{cq} \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not(isEmpty}(C)) \ .
\]

\[
\text{getTop}(\text{add}(\text{add}(..., b_1), b_2))
= \text{getTop}(\text{add}(..., b_1))
= b_1
\]

\[
\text{The Condition}
\]

\[
\text{not(isEmpty}(\text{add}(...))
= \text{not(false)}
= \text{true}
\]
Conditional Equations

\[ cq \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not(isEmpty}(C)) \].

\[
\text{getTop}(\text{add}(\text{add}(\text{empty, } b1), b2)) \\
= \text{getTop}(\text{add}(\text{empty, } b1)) \\
= b1
\]

The Condition

\[
\text{not(isEmpty}(\text{add}(\text{empty, } b1))) \\
= \text{not(false)} \\
= \text{true}
\]
Conditional Equations

\[
\text{cq} \quad \text{getTop}(\text{add}(C, B)) = \text{getTop}(C) \quad \text{if} \quad \text{not(isEmpty}(C))
\]

\[
\text{getTop}(\text{add}(\text{add}(\text{empty, b1}), \text{b2})) = \text{getTop}(\text{add}(\text{empty, b1})) = \text{b1}
\]

\[
\text{The Condition} \quad \text{not(isEmpty}(\text{add}(\text{empty, b1}))) = \text{not(false)} = \text{true}
\]
Even More Maude fmod CARD_HAND

\begin{verbatim}
 var C : CardHand.
 var B : BandCard.

 cq removeTop(add(C,B)) = empty if isEmpty(C).
\end{verbatim}

Equivalently:

\begin{verbatim}
 var B : BandCard.
 eq removeTop(add(empty, B)) = empty.
\end{verbatim}
Even More Maude fmod CARD_HAND

\[ \text{CARD_HAND contd.} \]

\textbf{var} \ C \ : \ \text{CardHand} . \\
\textbf{var} \ B \ : \ \text{BandCard} . \\
\ldots \\
\textbf{cq} \ \text{removeTop(add}(C,B)) \ = \ \text{empty} \quad \text{if} \ \text{isEmpty}(C) .

\textbf{Equivalently:}

\[ \text{CARD_HAND contd.} \]

\textbf{var} \ B \ : \ \text{BandCard} . \\
\textbf{eq} \ \text{removeTop(add}(\text{empty}, B)) \ = \ \text{empty} .
Even More Maude fmod CARD_HAND

CardHand contd.

\[
\begin{align*}
\text{var} & \quad \text{C : CardHand}.
\text{var} & \quad \text{B : BandCard}.
\quad \ldots \quad \text{cq} \quad \text{removeTop(add}(C,B)) = \text{empty} \quad \text{if isEmpty}(C).
\text{cq} \quad \text{removeTop(add}(C,B)) = \text{add}(\text{removeTop}(C), B) \quad \text{if not(isEmpty}(C)) .
\end{align*}
\]
endfm

Exercise: removeTop(add(add(add(\text{empty}, b1), b2), b3)) !
Even More Maude fmod CARD\_HAND

CARD\_HAND contd.

\textbf{var} \ C: \text{CardHand} .
\textbf{var} \ B: \text{BandCard} .

\ldots

\textbf{cq} \ \text{removeTop(add(C,B))} = \text{empty} \ \textbf{if} \ \text{isEmpty(C)} .
\textbf{cq} \ \text{removeTop(add(C,B))} = \text{add(removeTop(C), B)} \ \textbf{if} \ \text{not(isEmpty(C))} .

\textbf{endfm}

\textbf{Exercise:} \ \text{removeTop(add(add(add(add(empty, b1), b2), b3))} !
Even More Maude fmod CARD_HAND

CARD_HAND contd.

\begin{itemize}
\item \textbf{var} C : CardHand .
\item \textbf{var} B : BandCard .
\end{itemize}

\ldots

\begin{itemize}
\item \textbf{cq} removeTop(add(C,B)) = empty \quad \textbf{if} \text{ isEmpty}(C) .
\item \textbf{cq} removeTop(add(C,B)) = add(removeTop(C), B) \quad \textbf{if} \text{ not}(\text{isEmpty}(C)) .
\end{itemize}

\textbf{endfm}

\textbf{Exercise}: removeTop(add(add(add(add(empty, b1), b2), b3)) !
Even More Maude fmod CARD\_HAND

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{CARD\_HAND contd.} \\
\hline
\texttt{var } C : \texttt{CardHand} . \\
\texttt{var } B : \texttt{BandCard} . \\
... \\
\texttt{cq removeTop(add(C,B)) = empty } \textbf{if} \texttt{isEmpty(C)} . \\
\texttt{cq removeTop(add(C,B)) = add(removeTop(C), B) } \textbf{if} \texttt{not(isEmpty(C))} . \\
\hline
\end{tabular}
\end{center}

\textbf{Exercise:} removeTop(add(add(add(add(\texttt{empty}, b1), b2), b3))) !
public class CardHand {

    public boolean isEmpty() { ...
    }

    public void add(BandCard b) { ...
    }

    public BandCard getTop() { ...
    }

    public void removeTop() { ...
    }

}
public class CardHand {
    public boolean isEmpty() { ... }
    public void add(BandCard b) { ... }
    public BandCard getTop() { ... }
    public void removeTop() { ... }
}
That’s All, Folks!

Summary

- An ADT
- Conditional Equations

Next:

implementation