HOG

{*acro:HoG*}

{*acro:HoC*} {*acro:OZ*}

Abstract. The House of Games (HoG) is a collection of classes and functions working together to provide a basic facility for the writing of programs involving games. We write the specification of **HoC!** (**HoC!**) using Object-Z (OZ)

Figure 1. House of Games. Directed by David Mamet. Produced by Michael Hausman. Written by David Mamet. Distributed by Orion Pictures.

1. Definitions

1.1. **The board.** A (board) game [1] [2], as far as HoG is concerned, is a collection of *places* that are themselves collections of *counters*[3].

{*basictype:counters*} The set of counters is given by the basic type:

Basic Type 1.1. [**C**]

{*basictype:places*} , while the set of places is given by the basic type:

Basic Type 1.2. [**P**]

.

Axiom 1.1. The basic types 1.1 and 1.2 satisfy the partition axiom:

$$\mathbf{C} \cap \mathbf{P} = \varnothing$$

{*schema:board*} **Schema 1.1.** A *board* is a collection of counters and places such that each counter has (is placed on) one and only one place. The schema *Board* declares and defines this in a rigorous way.

The available counters C on the board are taken from the pool of counters $\mathbf{C}$.

$Counter \mathrel{\widehat{=}} [\, C : \mathbb{P}\,\mathbf{C} \,]$

The available places P on the board are taken from the pool of places $\mathbf{P}$.

$Place \mathrel{\widehat{=}} [\, P : \mathbb{P}\,\mathbf{P} \,]$

The administration of the distribution of counters over the places is kept by the partial set-valued function $C\_$.

$Dist \mathrel{\widehat{=}} [\, C\_ : \mathbf{P} \nrightarrow \mathbb{P}\,\mathbf{C} \,]$

Board

Counter	
Place	
Dist	[1]
$\operatorname{dom} C\_ = P$	[2]
$\{C_p\}_{p \in P}$ partitions C	[3]

[1] "Board games are traditionally a subset of tabletop games that involve counters or pieces moved or placed on a pre-marked surface or "board", according to a set of rules."

https://en.wikipedia.org/wiki/Board_game,[2].

[2] "**board game** ● *n.* a game that involves the movement of counters or other objects around a board.",[3]

[3] "or bit, checker, chip, disc, draughtsman, man, meeple, mover, pawn, game piece, player piece, playing piece, singleton, stone, token, unit." and HoG wants to add "card or playing card".

1.2. **Building a board.** There are four operations to build a board, i.e. adding and removing places and counters, resp. from and to their respective pools.

{operation:aplace}

Operation 1.1. Adding a place $p?$ from the pool $\mathbf{P}$ to the board adds an *empty* place, i.e. a place $p?$ such that $C_{p?} = \varnothing$. If the place $p?$ is already on board then the operations skips.

$$\begin{array}{|l}
\hline
\textit{APlace1} \\
\Xi \textit{Counter} \\
\Delta \textit{Place} \\
\Delta \textit{Dist} \\
p? : \mathbf{P} \\
\hline
p? \notin P \\
P' = P \cup \{p?\} \\
C'_{_} = C_{_} \cup \{p? \mapsto \varnothing\} \\
\hline
\end{array}$$

$\textit{APlace2} \mathrel{\widehat=} [\ \Xi \textit{Board};\ p? : \mathbf{P} \mid p? \in P\ ]$

$\textit{APlace} \mathrel{\widehat=} \textit{APlace1} \vee \textit{APlace2}$

{lem:aplace}

Lemma 1 (Consistency lemma APlace). $\vdash \forall\ \textit{Board} \wedge \textit{APlace} \bullet \textit{Board}'$
See section 1.2 for the proof.

(1) $\operatorname{dom} C'_{\_} = \operatorname{dom} C_{\_} \cup \{p?\} = P \cup \{p?\} = P'$
(2) Let $p, q \in \operatorname{dom} C'_{\_} \wedge p \neq q$. Then $C'_p \cap C'_{p?} = C'_{p?} \cap C'_q = \varnothing$.
$\bigcup_p C'_p = \left(\bigcup_p C_p\right) \cup C'_{p?} = C \cup \varnothing = C = C'$

{operation:rplace}

Operation 1.2. Removing a place $p?$ from the board removes the counters that were placed on $p?$ as well. If the place $p?$ is not on the board then the operations skips.

$$\begin{array}{|l}
\hline
\textit{RPlace1} \\
\Delta \textit{Counter} \\
\Delta \textit{Place} \\
\Delta \textit{Dist} \\
p? : \mathbf{P} \\
\hline
p? \in P \\
C' = C \setminus C_{p?} \\
P' = P \setminus \{p?\} \\
C'_{_} = \{p?\} \lhd C_{_} \\
\hline
\end{array}$$

$\textit{RPlace2} \mathrel{\widehat=} [\ \Xi \textit{Board};\ p? : \mathbf{P} \mid p? \notin P\ ]$

$\textit{RPlace} \mathrel{\widehat=} \textit{RPlace1} \vee \textit{RPlace2}$

{lem:rplace}

Lemma 2 (Consistency lemma RPlace). $\vdash \forall\ \textit{Board} \wedge \textit{RPlace} \bullet \textit{Board}'$

The dashed components of the declaration satisfy the *Board* properties:

(1) $\operatorname{dom} C'_{\_} = \operatorname{dom} C_{\_} \setminus \{p?\} = P \setminus \{p?\} = P'$
(2) Let $p, q \in \operatorname{dom} C'_{\_} \wedge p \neq q$. Then $C'_p \cap C'_q = C_p \cap C_q = \varnothing$.
$\bigcup_p C'_p = \left(\left(\bigcup_p C'_p\right) \cup C_{p?}\right) \setminus C_{p?} = C \setminus C_{p?} = C'$

{operation:acounter}

Operation 1.3. Put a counter $c?$ from the pool at a place $p?$. If the counter $c?$ is already on the board or the place $p?$ is not on the board then the operations skips.

$$
\begin{array}{|l}
\multicolumn{1}{l}{ACounter1} \\ \hline
\Delta Counter \\
\Xi Place \\
\Delta Dist \\
c? : \mathbf{C} \\
p? : \mathbf{P} \\ \hline
c? \notin C \\
p \in P \\
C' = C \cup \{c?\} \\
C'_{\_} = \{p? \mapsto C_{p?} \cup \{c?\}\} \oplus C_{\_} \\ \hline
\end{array}
$$

$ACounter2 \mathrel{\widehat=} [\ \Xi Board;\ c? : \mathbf{C};\ p? : \mathbf{P} \mid c? \in C \vee p? \notin P\ ]$

$ACounter \mathrel{\widehat=} ACounter1 \vee ACounter2$

{lem:acounter}

Lemma 3 (Consistency lemma ACounter). $\vdash \forall\ Board \wedge ACounter \bullet Board'$

The dashed components of the declaration satisfy the *Board* properties:

(1) $\operatorname{dom} C'_{\_} = \operatorname{dom} C_{\_} \cup \{p?\} = P = P'$

(2) Let $p, q \in \operatorname{dom} C'_{\_} \wedge p \neq q \wedge q = (\mu\ p : P \mid c? \in C_p\ )$. Then $C'_p \cap C'_q = C_p \cap (C_q \setminus \{c?\}) = (C_p \cap C_q) \setminus \{c?\} = \varnothing$.
$\bigcup_p C'_p = \left(\bigcup_{p \neq q} C'_p\right) \cup C'_q = \left(\bigcup_{p \neq q} C_p\right) \cup (C_q \setminus \{c?\}) = \left(\left(\bigcup_{p \neq q} C_p\right) \cup C_q\right) \setminus \left(\{c?\} \setminus \left(\bigcup_{p \neq q} C_p\right)\right) = C \setminus \{c?\} = C'$

{operation:rcounter}

Operation 1.4. Removing a counter $c?$ from the board also removes it from its place. If the counter $c?$ is not on the board then the operations skips.

$$
\begin{array}{|l}
\multicolumn{1}{l}{RCounter} \\ \hline
\Delta Counter \\
\Xi Place \\
\Delta Dist \\
c? : \mathbf{C} \\ \hline
c? \in C \\
C' = C \setminus \{c?\} \\
C'_{\_} = \{(\mu\ p : P \mid c? \in C_p\ ) \mapsto C_p \setminus \{c?\}\} \oplus C_{\_} \\ \hline
\end{array}
$$

$RCounter2 \mathrel{\widehat=} [\ \Xi Board;\ c? : \mathbf{C} \mid c? \notin C\ ]$

$RCounter \mathrel{\widehat=} RCounter1 \vee RCounter2$

{lem:rcounter}

Lemma 4 (Consistency lemma RCounter). $\vdash \forall\ Board \wedge RCounter \bullet Board'$

Lemma 4
The dashed components of the declaration satisfy the *Board* properties:

(1) $\operatorname{dom} C'_{\_} = \operatorname{dom} C_{\_} \cup \{(\mu\ p : P \mid c? \in C_p\ )\} = P = P'$

(2) Let $p, q \in \operatorname{dom} C'_{\_} \wedge p \neq q \wedge q = (\mu\ p : P \mid c? \in C_p)$. Then $C'_p \cap C'_q = C_p \cap (C_q \setminus \{c?\}) = (C_p \cap C_q) \setminus \{c?\} = \varnothing$.
$\bigcup_p C'_p = \left(\bigcup_{p \neq q} C'_p\right) \cup C'_q = \left(\bigcup_{p \neq q} C_p\right) \cup (C_q \setminus \{c?\}) = \left(\left(\bigcup_{p \neq q} C_p\right) \cup C_q\right) \setminus \left(\{c?\} \setminus \left(\bigcup_{p \neq q} C_p\right)\right) = C \setminus \{c?\} = C'$

Proof

[Lemma 1]

$$B1\begin{cases}\left\lceil Board \wedge Aplace1\right\rceil^{[1]} \\ \lceil p \in \mathbf{P} \wedge c \subseteq \mathbf{C} \wedge d \subseteq \mathbf{C}\rceil^{[2]} \\ \lceil (p, c) \in C'_\_ \wedge (p, d) \in C'_\_\rceil^{[3]}\end{cases}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B1}{p \in \operatorname{dom} C'_\_}}{p \in \operatorname{dom}\,(C\_ \cup \{p? \mapsto \varnothing\})}}{p \in \operatorname{dom} C\_ \cup \operatorname{dom}\{p? \mapsto \varnothing\}}}{p \in \operatorname{dom} C\_ \vee p \in \operatorname{dom}\{p? \mapsto \varnothing\}}}{p \in \operatorname{dom} C\_ \vee p = p?} \quad \dfrac{\begin{matrix} B1 \\ \lceil p \in \operatorname{dom} C\_\rceil^{[4]}\end{matrix}}{c = d} \quad \dfrac{\dfrac{\begin{matrix} B1 \\ \lceil p = p?\rceil^{[4]}\end{matrix}}{c = \varnothing \wedge d = \varnothing}}{c = d}}{c = d}\;\vee -, 4}{p \in \mathbf{P} \wedge c, d \subseteq \mathbf{C} \wedge (p, c), (p, d) \in C'_\_ \Rightarrow c = d}\;\Rightarrow +, 3}{C'_\_ \in \mathbf{P} \nrightarrow \mathbb{P}\,\mathbf{C}}\;\forall +, 2}{\forall\ Board \wedge APlace1 \bullet\ C'_\_ \in \mathbf{P} \nrightarrow \mathbb{P}\,\mathbf{C}}\;\forall +, 1$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\left\lceil Board \wedge Aplace1\right\rceil^{[1]}}{\operatorname{dom} C'_\_ = \operatorname{dom}\,(C\_ \cup \{p? \mapsto \varnothing\})}}{\operatorname{dom} C'_\_ = \operatorname{dom} C\_ \cup \operatorname{dom}\{p? \mapsto \varnothing\}}}{\operatorname{dom} C'_\_ = P \cup \{p?\}}}{\operatorname{dom} C'_\_ = P'}}{\forall\ Board \wedge APlace1 \bullet \operatorname{dom} C'_\_ = P'}\;\forall +, 1$$

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\left\lceil Board \wedge Aplace1\right\rceil^{[1]} \\ p \in \operatorname{dom} C'_\_ \wedge q \in \operatorname{dom} C'_\_\end{matrix}}{p \in \operatorname{dom} C\_ \cup \operatorname{dom}\{p? \mapsto \varnothing\} \wedge q \in \operatorname{dom} C\_ \cup \operatorname{dom}\{p? \mapsto \varnothing\}}}{\dfrac{(p \in \operatorname{dom} C\_ \vee p = p?) \wedge (q \in \operatorname{dom} C\_ \vee q = p?)}{(p \in \operatorname{dom} C\_ \wedge q \in \operatorname{dom} C\_) \vee (p = p? \vee q = p?)}}}{C'_p \cap C'_q = \varnothing}\;\textit{Case analysis}$$

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\left\lceil Board \wedge Aplace1\right\rceil^{[1]} \\ \bigcup_p C'_p = \bigcup_p (C\_ \cup \{p? \mapsto \varnothing\})_p\end{matrix}}{\bigcup_p C'_p = \bigcup_p C_p \cup \varnothing}}{\bigcup_p C'_p = C}}{\bigcup_p C'_p = C'}$$

Counters have defining features that make it possible to know the roles they fulfill in the game and that catogorize them in a sensible way understood by (players of) the game. The set of features that are defining for the category to which counters can belong is F and is introduced as a basic type:

The categorizing function $c : \mathbb{N} \pinj F$, a partial finite injection, assigns to a finite set of numbers the features that define the category to which a conter belongs. We may ask for c to be normalized in the sense that there is a number $N : \mathbb{N}$ such that $\operatorname{dom} c = 0 \mathrel{..} N-1$. This is axiomatically defined:

{*axiom:category-function*}

Axiom 1.2. The category function c and the number of categories C.

$$
\begin{array}{|l}
c : \mathbb{N} \rightarrowtail\!\!\!\!\!\!\!\!\!\;\;\; \twoheadrightarrow F \\
C : \mathbb{N} \\
\hline
\exists\, N : \mathbb{N} \bullet \operatorname{dom} c = 0 \mathrel{..} N-1 \land C = N
\end{array}
$$

Remark. If $F = \operatorname{ran} c$ then c is also a bijection.

{*exa:playing-card*}

Example 5. In some playing card game G that has to be played with one standard deck, i.e. a deck of cards of which every card (counter) has one the of four suits ♦, ♥, ♠ or ♣, and one of the thirteen ranks Ace, 2, 3, 4, 5, 6, 7, 8 , 9, 10, Jack, Queen or King. If the four suits are represented by the four numbers $0 \mathrel{..} 3$ and the thirteen ranks by the thirteen numbers $0 \mathrel{..} 12$, then the defining feature set is $F = 0 \mathrel{..} 3 \times 0 \mathrel{..} 12$. This means that there are $4 \times 13 = 52$ different kinds of cards with which we can play the card game G. The function $c : 0 \mathrel{..} 51 \rightarrowtail\!\!\!\rightarrow 0 \mathrel{..} 3 \times 0 \mathrel{..} 12$ that is defined by $c\, i = (i \operatorname{div} 13, i \operatorname{mod} 13)\,,\ i \in 0 \mathrel{..} 51$ maps the number i to a defining suit and rank in a one-to-one way. c together with the number $C = 52$ satisfy axiom 1.2. Notice that in this case c is a bijection.

A game like *Canasta* needs two standard decks of cards to which two Jokers have been added, so there are 53 categories, i.e. categories $0 \mathrel{..} 51$ for the standard cards and the category 52 for Jokers. To fit the Joker to the suit-rank feature system, we can say that a Joker's rank is 13 while its suit is undetermined and considered unimportant. Thus the category function becomes $c : 0 \mathrel{..} 52 \pinj 0 \mathrel{..} 3 \times 0 \mathrel{..} 13$ defined by $c\, i = \begin{cases} (i \operatorname{div} 13, i \operatorname{mod} 13) & i \in 0 \mathrel{..} 51 \\ (0, 13) & i = 52 \end{cases}$. Notice that in this case c is not bijective.

- **ACard:** The abstract playing card. An (object of type) **ACard** knows about its *suit* and *rank*, i.e. its *index*, and whether or not it is *faceup* or *facedown*. **ACard**s also have a reference to the **AHand** that holds them.
- **CCard:** The concrete playing card. A **CCard** represents a *real-world* card in the sense that it holds real-world characteristics of a playing card, like position on the table, faceup image and facedown image etc. **CCard**s also have a reference to the **CHand** that holds them.
- **AHand:** The abstract hand. The **AHand** is a sequential container of references to **ACard**s. **AHand**s also have a reference to the **AGame** to which they are assigned.
- **CHand:** The concrete hand. The **CHand** is a sequential container of references to **CCard**s. Like a **CCard**, they represent a real-world hand of cards and therefore hold the real-world characteristics of a hand of cards. **CHand**s also have a reference to the **CGame** to which they are assigned.

1.3. A Mediator.

1.3.1. *The injections.*

InAC	*InAC*
$accc : ACard \rightarrowtail\!\!\!\!\!\!\!+ CCard$	$accc : ACard \rightarrowtail\!\!\!\!\!\!\!+ CCard$
$ahch : AHand \rightarrowtail\!\!\!\!\!\!\!+ CHand$	$ahch : AHand \rightarrowtail\!\!\!\!\!\!\!+ CHand$

{*schema:InAC*}

Schema 1.2. This schema gives the partial injections *accc* and *ahch* that resp. map an *ACard* to a unique *CCard* (and vice versa) and an *AHand* to a unique *CHand* (and vice versa).

InAC

$accc : ACard \rightarrowtail\!\!\!\!\!\!\!+ CCard$
$ahch : AHand \rightarrowtail\!\!\!\!\!\!\!+ CHand$

1.3.2. *The abstract game.*

{*schema:AGame*}

Schema 1.3. This is the schema that specifies the abstract game, i.e. a collection of abstract hands (and cards). The hands *aCentral* and *aDiscard* are present in every *AGame*.

AGame

$aCentral, aDiscard : AHand$
$aHand : \mathrm{iseq}\, AHand$

Initially the injective sequence over objects of *AHand* is an empty sequence.

InitAGame

$AGame$

$aHand = \langle\, \rangle$

1.3.3. *The concrete game.*

{*schema:CGame*}

Schema 1.4. This is the schema that specifies the concrete game, i.e. a collection of concrete hands (and cards). The hands *cCentral* and *cDiscard* are present in every *CGame*.

CGame

$cCentral, cDiscard : CHand$
$cHand : \mathrm{iseq}\, CHand$

Initially the injective sequence over objects of *AHand* is an empty sequence.

InitCGame

$CGame$

$cHand = \langle\, \rangle$

1.3.4. *The mediator.*

{*schema:Mediator*}

Schema 1.5. The schema *Mediator* combines the schemas **??** and schemas 1.3 and 1.4 that specifies the concrete game, i.e. a collection of concrete hands (and cards). The hands *cCentral* and *cDiscard* are present in every *CGame*.

$$
\begin{array}{|l}
\hline
\textit{Mediator} \\
\textit{BijAC} \\
\textit{AGame} \\
\textit{CGame} \\
\hline
\end{array}
$$

Initially the injective sequence over objects of $AHand$ is an empty sequence.

$$
\begin{array}{|l}
\hline
\textit{InitMediator} \\
\textit{Mediator} \\
\hline
\textit{InitAGame} \\
\textit{InitCGame} \\
accc = \varnothing \\
ahch = \{aCentral \mapsto cCentral, aDiscard \mapsto cDiscard\} \\
\hline
\end{array}
$$

1.4. **Abstract cards and hands.**

{class:ACard}

1.4.1. *The abstract card.*

Class 1.1. The class $ACard$ represents an abstract playing card, which could be seen as an element of $\mathbb{B} \times \mathbb{N} \times 0 \mathinner{..} R-1$, where $R \in \mathbb{N}_1$ is the number of ranks that such a card can have . A standard playing card is said to have a .

$$
\begin{array}{|l}
\hline
\textit{ACard} \\
\upharpoonright (index, rank, suit, faceup) \\
\quad S, R : \mathbb{N}_1 \qquad \text{[Number of suits and ranks]} \\
\quad index : \mathbb{N} \\
\quad faceup : \mathbb{B} \\
\quad hand : AHand \\
\quad \Delta \\
\quad rank, suit : \mathbb{N} \\
\quad \text{---} \\
\quad rank = index \bmod R \\
\quad suit = index \operatorname{div} R \\
\quad \textsc{Init} \\
\quad faceup \land index \in 0 \mathinner{..} S \cdot R - 1 \\
\quad hand \in \operatorname{dom} \theta Mediator.ahch \\
flip \mathrel{\widehat=} [\ \Delta(faceup) \mid faceup' \Leftrightarrow \neg faceup\ ] \\
face \mathrel{\widehat=} [\ \Delta(faceup);\ faceup? : \mathbb{B} \mid faceup' \Leftrightarrow faceup?\ ] \\
\hline
\end{array}
$$

{class:AHand}

1.4.2. *The abstract hand.*

Class 1.2. The class $AHand$ represents an abstract hand of abstract cards. An object of this class is essentially an element of $\operatorname{iseq} ACard$, i.e. an injective sequence of abstract cards.

AHand

$\upharpoonright (card, size)$

$Id : String$
$card : \mathrm{iseq}\, ACard$
Δ
$size : \mathbb{N}$
$empty : \mathbb{B}$

$size = \# card$
$empty \Leftrightarrow size = 0$

INIT

$card = \langle\, \rangle$

1.5. **Concrete cards and hands.**

{class:CCard} 1.5.1. *The concrete card.*

Class 1.3. The class *CCard* represents a concrete playing card, which is the view of an abstract playing card in a user interface system, where cards have, e.g, images and positions relative to another position, etc..

CCard

$\upharpoonright (xyz, im, pos, z, index, rank, suit, faceup)$

$[CImage]$

$fuim : 0 \mathinner{..} S \cdot R - 1 \rightarrow CImage$
$fdim : 0 \mathinner{..} S \cdot R - 1 \rightarrow CImage$

$xyz : \mathbb{R}^3$
$im : CImage$
Δ
$pos : \mathbb{R}^2$
$z : \mathbb{R}$
$index, rank, suit : \mathbb{N}$

$im = \mathbf{if}\ faceup\ \mathbf{then}\ fuim\ index\ \mathbf{else}\ fdim\ index$
$pos = (xyz.1, xyz.2)$
$z = xyz.3$
$index = (\theta Mediator.accc^{\sim}\ self).index$
$rank = (\theta Mediator.accc^{\sim}\ self).rank$
$suit = (\theta Mediator.accc^{\sim}\ self).suit$
$faceup = (\theta Mediator.accc^{\sim}\ self).faceup$

INIT

$xyz = (0, 0, hand.size)$

$hand \in \mathrm{dom}\, \theta Mediator.ahch^{\sim}$

{class:CHand} 1.5.2. *The concrete hand.*

Class 1.4. The class *CHand* represents a concrete hand of concrete cards. An object of this class is essentially an element of iseq $CCard_{©}$.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CHand}} \\
\hline
\upharpoonright (card, size) \\
[HImage] \\
\quad fuim : 0 \mathrel{..} S \cdot R - 1 \rightarrow HImage \\
\quad fdim : 0 \mathrel{..} S \cdot R - 1 \rightarrow HImage \\
\\
\quad card : \text{iseq } CCard_{©} \\
\quad xyz : \mathbb{R}^3 \\
\quad im : HImage \\
\quad \Delta \\
\quad size : \mathbb{N} \\
\quad empty : \mathbb{B} \\
\quad pos : \mathbb{R}^2 \\
\quad z : \mathbb{R} \\
\hline
\quad size = \#card \\
\quad empty \Leftrightarrow size = 0 \\
\quad pos = (xyz.1, xyz.2) \\
\quad z = xyz.3 \\
\\
\quad \textsc{Init} \\
\quad card = \langle\,\rangle
\end{array}
$$

2. Constant Operations

The number of hands in the game.

{op:gamehandsize} **Operation const 2.1.** The number of hands currently present in the game, not counting *aCentral* and *aDiscard*.

$GameHandSize \mathrel{\widehat{=}} [\ AGame;\ ghsize! : \mathbb{N} \mid ghsize! = \#aHand\ ]$

Test whether there are no hands present besides *aCentral* and *aDiscard*.

$GameHandEmpty \mathrel{\widehat{=}} [\ AGame;\ ghempty! : \mathbb{B} \mid ghempty! \Leftrightarrow aHand = \langle\,\rangle\ ]$

The number of cards in the game.

{op:gamecardsize} **Operation const 2.2.** The total number of cards presently in the game.

$GameCardSize \mathrel{\widehat{=}} [\ BijAC;\ gcsize! : \mathbb{N} \mid gcsize! = \#accc\ ]$

3. Operations

{op:addnewhand} **Operation 3.1.** This operation adds a new abstract hand to the game *AGame*. The operations involved are threefold:

(1) Test whether the new hand *ahand* is not a member of the already registered hands, i.e. $ahand \notin \mathrm{dom}\, ahch$.
(2) Then, if not, construct a new concrete hand *chand* and add the mapping $ahand \mapsto chand$ to the partial injection *ahch*.
(3) Add *ahand* and *chand* to the end of resp. *aHand* and *cHand*.

AddNewHand

$\Delta Mediator$

$$
\begin{array}{l}
accc' = accc \\
(\textbf{let}\ ah == (\mu\ ahand : AHand_{©})\ ;\ ch == (\mu\ chand : CHand_{©}) \bullet \\
\quad ahch' = ahch \cup \{ah \mapsto ch\} \wedge \\
\quad aHand' = aHand \frown \langle ah \rangle \wedge cHand' = cHand \frown \langle ch \rangle)
\end{array}
$$

4. C++

4.1. **The AGame and CGame.**

4.1.1. *The class frames.*

12a ⟨*agame.h* 12a⟩≡

```
#ifndef AGAME_H
#define AGAME_H

⟨AGame includes 13h⟩

namespace HOC
{

class AGame {
  ⟨AGame private 13i⟩
public:
  AGame();
 ~AGame();
  ⟨AGame public 13j⟩
};

⟨AGame declarations 13g⟩

}

#endif
```

12b ⟨*agame.cpp* 12b⟩≡

```
#include "agame.h"

⟨AGame definitions includes 13a⟩

⟨AGame definitions 12c⟩
```

12c ⟨*AGame definitions* 12c⟩≡ (12b)

```
// noop
```

13a ⟨*AGame definitions includes* 13a⟩≡ (12b)
```
// noop
```

13b ⟨*cgame.h* 13b⟩≡
```
#ifndef CGAME_H
#define CGAME_H

⟨CGame includes 14a⟩

namespace HOC {

class CGame {
  ⟨CGame private 14b⟩
public:
  CGame();
 ~CGame();
  ⟨CGame public 14c⟩
};

⟨CGame declarations 13f⟩

}

#endif
```

13c ⟨*cgame.cpp* 13c⟩≡
```
#include "CGame.h"
⟨CGame definitions includes 13e⟩
⟨CGame definitions 13d⟩
```

13d ⟨*CGame definitions* 13d⟩≡ (13c)
```
// noop
```

13e ⟨*CGame definitions includes* 13e⟩≡ (13c)
```
// noop
```

13f ⟨*CGame declarations* 13f⟩≡ (13b)
```
// noop
```

13g ⟨*AGame declarations* 13g⟩≡ (12a)
```
// noop
```

4.1.2. *The two standard hands.*

13h ⟨*AGame includes* 13h⟩≡ (12a) 14d ▷
```
#include "ahand.h"
```

13i ⟨*AGame private* 13i⟩≡ (12a) 14e ▷
```
AHand aCentral, aDiscard;
```

13j ⟨*AGame public* 13j⟩≡ (12a) 14f ▷
```
AHand*central() const {return &aCentral;}
AHand*discard() const {return &aDiscard;}
```

14a ⟨*CGame includes* 14a⟩≡ (13b) 14g ▷
```
#include "chand.h"
```

14b ⟨*CGame private* 14b⟩≡ (13b) 14h ▷
```
CHand cCentral, cDiscard;
```

14c ⟨*CGame public* 14c⟩≡ (13b) 14i ▷
```
CHand*central() const {return &cCentral;}
CHand*discard() const {return &cDiscard;}
```

4.1.3. *The injective sequences aHand and cHand.* These are refined to `std::vector`.

14d ⟨*AGame includes* 13h⟩+≡ (12a) ◁ 13h
```
#include <vector>
```

14e ⟨*AGame private* 13i⟩+≡ (12a) ◁ 13i
```
typedef std::vector<AHand*> VAHand;
VAHand aHand;
```

14f ⟨*AGame public* 13j⟩+≡ (12a) ◁ 13j
```
typedef VAHand::size_type size_type;
AHand*hand(size_type i) const {return aHand.at(i);}
```

14g ⟨*CGame includes* 14a⟩+≡ (13b) ◁ 14a
```
#include <vector>
```

14h ⟨*CGame private* 14b⟩+≡ (13b) ◁ 14b
```
typedef std::vector<CHand*> VCHand;
VCHand cHand;
```

14i ⟨*CGame public* 14c⟩+≡ (13b) ◁ 14c
```
typedef VCHand::size_type size_type;
CHand*hand(size_type i) const {return cHand.at(i);}
```

4.2. **The Mediator.**

4.2.1. *The class frame.* The schema *Mediator* is implemented by the class `Mediator` of which the frame is given:

14j ⟨*mediator.h* 14j⟩≡
```
#ifndef MEDIATOR_H
#define MEDIATOR_H

⟨Mediator includes 15e⟩

namespace HOC
{

class Mediator {
  ⟨Mediator private 15f⟩
public:
  Mediator();
 ~Mediator();
  ⟨Mediator public 15d⟩
};
```

⟨*Mediator declarations* 15c⟩

```
}

#endif
```

15a ⟨*mediator.cpp* 15a⟩≡

```
#include "mediator.h"
using namespace HOC;
```

⟨*Mediator definitions* 15b⟩

15b ⟨*Mediator definitions* 15b⟩≡ (15a)

```
// noop
```

15c ⟨*Mediator declarations* 15c⟩≡ (14j)

```
// noop
```

15d ⟨*Mediator public* 15d⟩≡ (14j)

```
// noop
```

4.2.2. *The schema InAC*. The two injections *accc* and *ahch* are implemented with the Boost.Bimap[1] library: a bidirectional maps library for C++[4].

FIGURE 2. Boost.Bimap Logo

15e ⟨*Mediator includes* 15e⟩≡ (14j)

```
#include <boost/bimap.hpp>
class ACard;
class CCard;
class AHand;
class CHand;
```

15f ⟨*Mediator private* 15f⟩≡ (14j)

```
typedef boost::bimap<ACard*,CCard*> ACCC;
typedef boost::bimap<AHand*,CHand*> AHCH;
ACCC accc; // ACard <-> CCard
AHCH ahch; // AHand <-> CHand
```

[4] https://www.boost.org/doc/libs/1_71_0/libs/bimap/doc/html/index.html

Appendix A. The BUILD-SCRIPT

16 ⟨*build-script* 16⟩≡

```
#!/bin/sh
#if [ -z "${NOWEB_SOURCE}" ]then
#NOWEB_SOURCE=myfile.nw
#fi
notangle -Ragame.h ${NOWEB_SOURCE} > ~/Documents/HOC/Program/agame.h
notangle -Ragame.cpp ${NOWEB_SOURCE} > ~/Documents/HOC/Program/agame.cpp
notangle -Rcgame.h ${NOWEB_SOURCE} > ~/Documents/HOC/Program/cgame.h
notangle -Rcgame.cpp ${NOWEB_SOURCE} > ~/Documents/HOC/Program/cgame.cpp
notangle -Rmediator.h ${NOWEB_SOURCE} > ~/Documents/HOC/Program/mediator.h
notangle -Rmediator.cpp ${NOWEB_SOURCE} > ~/Documents/HOC/Program/mediator.cpp
```

References

[1] Chapter 1. Boost.Bimap - 1.71.0, December 2020. [Online; accessed 17. Dec. 2020].
[2] Contributors to Wikimedia projects. Board game - Wikipedia, Dec 2020. [Online; accessed 20. Dec. 2020].
[3] Catherine Soanes. *The paperback Oxford English dictionary*. Oxford University Press, Oxford New York, 2002.