Translation Rules

High-Level

- $[\![ParmList]\!]_{params}$: translates a list of method parameters
- $[\![ParmList]\!]_{vars}$: translates a list of variables
- $[\![MethBody]\!]_{block}$: translates a Java block, for example a method body
- $[\![Method]\!]_{appMeth}$: translates active application methods into *Circus* actions
- $[\![Method]\!]_{dataMeth}$: translates data methods into an ***Oh Circus*** method

Low-Level

- $[\![Name]\!]_{name}$: translates the *name* to a Z identifier
- $[\![varType]\!]_{type}$: translates types
- $[\![expr]\!]_{expression}$: translates expressions

Auxiliary Functions

- $IdOf(name)$: yields the identifier of a component called *name*
- $MethName(method)$: yields the method name of *method*

Safelet

```
1  public class PName implements Safelet
2  {
3    Vars
4
5    public PName(PParams)
6    {
7      VarInits
8    }
9
10   public void initializeApplication()
11   {
12     SInitBody
13   }
14
15   public MissionSequencer getSequencer()
16   {
17     return TLMS;
18   }
19
20   AppMeth1
21
22   AppMeth2
23   ...
24 }
```

$$\textbf{process}\ [\![PName]\!]_{name}\ App \mathrel{\widehat=} [\![PParams]\!]_{params}\ \textbf{begin}$$

State

$this : \text{ref}\ [\![PName]\!]_{name}$

state *State*

Init

$State'$

$this := \mathbf{new}\ [\![PName]\!]_{name}\ Class()$

$$InitializeApplication \mathrel{\widehat=} \begin{pmatrix} initializeApplicationCall \longrightarrow \\ [\![SInitBody]\!]_{methBody} \\ initializeApplicationRet \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$GetSequencer \mathrel{\widehat=} \begin{pmatrix} getSequencerCall \longrightarrow \\ getSequencerRet\,!\,IdOf(PName) \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$[\![AppMeth1]\!]_{appMeth}$
$[\![AppMeth2]\!]_{appMeth}$
$\ldots$

$$Methods \mathrel{\widehat=} \begin{pmatrix} GetSequencer \\ \Box \\ InitializeApplication \\ \Box \\ MethName(AppMeth1) \\ \Box \\ MethName(AppMeth2) \\ \ldots \end{pmatrix} ;\ Methods$$

$\bullet\ (Init\ ;\ Methods)\ \ \triangle(end\_safelet\_app \longrightarrow \mathbf{Skip})$

end

Mission Sequencer

```
1  public class PName extends MissionSequencer
2  {
3     Vars
4
5     public PName(PParams)
6     {
7        VarInits
8     }
9
10    protected Mission getNextMission()
11    {
12
13       return MId;
14    }
15
16    AppMeth1
17
18    AppMeth2
19    ...
20
21 }
```

$$\textbf{process}\ [\![PName]\!]_{name}\ App \mathrel{\widehat=} [\![PParams]\!]_{params}\ \textbf{begin}$$

State

$$this : \text{ref}\ [\![PName]\!]_{name}$$

$$\textbf{state}\ State$$

Init

$$State'$$
$$this := \textbf{new}\ [\![PName]\!]_{name}\ Class()$$

$$GetNextMission \mathrel{\widehat=} \textbf{var}\ ret : MissionID \bullet \left(\begin{array}{l} getNextMissionCall\,.\,IdOf(PName) \longrightarrow \\ ret := this\,.\,getNextMission(); \\ getNextMissionRet\,.\,IdOf(PName)\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$[\![AppMeth1]\!]_{appMeth}$$
$$[\![AppMeth2]\!]_{appMeth}$$
$$\ldots$$

$$Methods \mathrel{\widehat=} \left(\begin{array}{l} GetNextMission \\ \Box \\ MethName(AppMeth1) \\ \Box \\ MethName(AppMeth2) \\ \ldots \end{array}\right) ;\ Methods$$

$$\bullet\ (Init\ ;\ Methods)\ \ \triangle(end\_sequencer\_app\,.\,IdOf(PName) \longrightarrow \textbf{Skip})$$

$$\textbf{end}$$

Mission

```
1  public class PName extends Mission
2  {
3    Vars
4
5    public PName(PParams)
6    {
7      VarInits
8    }
9
10   protected void initialize()
11   {
12     RegisteredSchedulables
13   }
14
15   public boolean cleanUp()
16   {
17     Console.print("FlatBufferMission Cleanup");
18     return false;
19   }
20
21   AppMeth1
22
23   AppMeth2
24   ...
25 }
```

$$\textbf{process}\ [\![PName]\!]\ App \mathrel{\widehat=} [\![PParams]\!]_{params}\ \textbf{begin}$$

State

$$this : \text{ref}\ [\![PName]\!]_{name}$$

$$\textbf{state}\ State$$

Init

$$State'$$
$$this := \textbf{new}\ [\![PName]\!]_{name}\ Class()$$

$$InitializePhase \mathrel{\widehat=} \left(\begin{array}{l} initializeCall\,.\,IdOf(PName) \longrightarrow \\ [\![RegisteredSchedulables]\!]\ initializeRet\,.\,IdOf(PName) \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$CleanupPhase \mathrel{\widehat=} \left(\begin{array}{l} cleanupMissionCall\,.\,IdOf(PName) \longrightarrow \\ cleanupMissionRet\,.\,IdOf(PName)\,!\,\textbf{True} \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$[\![AppMeth1]\!]_{appMeth}$$
$$[\![AppMeth2]\!]_{appMeth}$$
$$\ldots$$

$$Methods \mathrel{\widehat{=}} \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ MethName(AppMeth1) \\ \Box \\ MethName(AppMeth2) \\ \ldots \end{array} \right) \; ; \; Methods$$

• $(Init \; ; \; Methods) \;\; \triangle (end\_mission\_app \,.\, IdOf(PName) \longrightarrow \textbf{Skip}$

end

Handlers

```
1  class PName extends HandlerType
2  {
3     Vars
4
5     public PName(PParams)
6     {
7        VarInits
8     }
9
10    public void handleAsyncEvent()
11    {
12       HandleAsyncBody
13    }
14
15    AppMeth1
16
17    AppMeth2
18    ...
19 }
```

$$\textbf{process}\ [\![PName]\!]\ App \mathrel{\widehat=} [\![PParams]\!]_{params}\ \textbf{begin}$$

State

$$this : \text{ref}\ [\![PName]\!]_{name}$$

$$\textbf{state}\ State$$

Init

$$State'$$
$$this := \textbf{new}\ [\![PName]\!]_{name}\ Class()$$

$$handleAsyncEvent \mathrel{\widehat=} \begin{pmatrix} handleAsyncEventCall\,.\,IdOf(PName) \longrightarrow \\ [\![HandleAsyncBody]\!]\ ; \\ handleAsyncEventRet\,.\,IdOf(PName) \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$[\![AppMeth1]\!]_{appMeth}$$
$$[\![AppMeth2]\!]_{appMeth}$$
$$\ldots$$

$$Methods \mathrel{\widehat=} \begin{pmatrix} handleAsyncEvent \\ MethName(AppMeth1) \\ \Box \\ MethName(AppMeth2) \\ \ldots \end{pmatrix} ;\ Methods$$

$$\bullet\ (Init\ ;\ Methods)\ \ \triangle(end\_ [\![HandlerTypeIdOf(PName)]\!] \longrightarrow \textbf{Skip})$$

$$\textbf{end}$$

Managed Thread

```
1  public class PName extends ManagedThread
2  {
3     Vars
4
5     public PName(PParams)
6     {
7        VarInits
8     }
9
10    public void run()
11    {
12       RunBody
13    }
14
15    AppMeth1
16
17    AppMeth2
18    ...
19 }
```

$$\textbf{process}\ [\![PName]\!]\ App \mathrel{\widehat{=}} [\![PParams]\!]_{params}\ \textbf{begin}$$

State

$$this : \text{ref}\ [\![PName]\!]_{name}$$

$$\textbf{state}\ State$$

Init

$$State'$$
$$this := \textbf{new}\ [\![PName]\!]_{name}\ Class()$$

$$Run \mathrel{\widehat{=}} \begin{pmatrix} runCall\,.\,IdOf(PName) \longrightarrow \\ [\![RunBody]\!]\ ; \\ runRet\,.\,IfOf(PName) \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$[\![AppMeth1]\!]_{appMeth}$$
$$[\![AppMeth2]\!]_{appMeth}$$
$$\ldots$$

$$Methods \mathrel{\widehat{=}} \begin{pmatrix} Run \\ \Box \\ MethName(AppMeth1) \\ \Box \\ MethName(AppMeth2) \\ \ldots \end{pmatrix} ;\ Methods$$

$$\bullet\ (Init\ ;\ Methods)\ \ \triangle(end\_managedThread\_app\,.\,IdOf(PName) \longrightarrow \textbf{Skip})$$

$$\textbf{end}$$

Data Class

class $[\![PName]\!]_{name}$ *Class* $\widehat{=}$ **begin**

state *State*

$[\![VarName]\!]_{name} : [\![VarType]\!]_{type}$

state *State*

initial *Init*

$State'$

$[\![VarName]\!]'_{name} = [\![VarInit]\!]_{expression}$

$[\![DataMeth1]\!]_{dataMeth}$
$[\![DataMeth2]\!]_{dataMeth}$
...

- **Skip**

end