Predictable Java

Thomas Bøgholm, René R. Hansen, Anders P. Ravn,
Bent Thomsen, and Hans Søndergaard

CISS, Aalborg University
VIA University College, Horsens

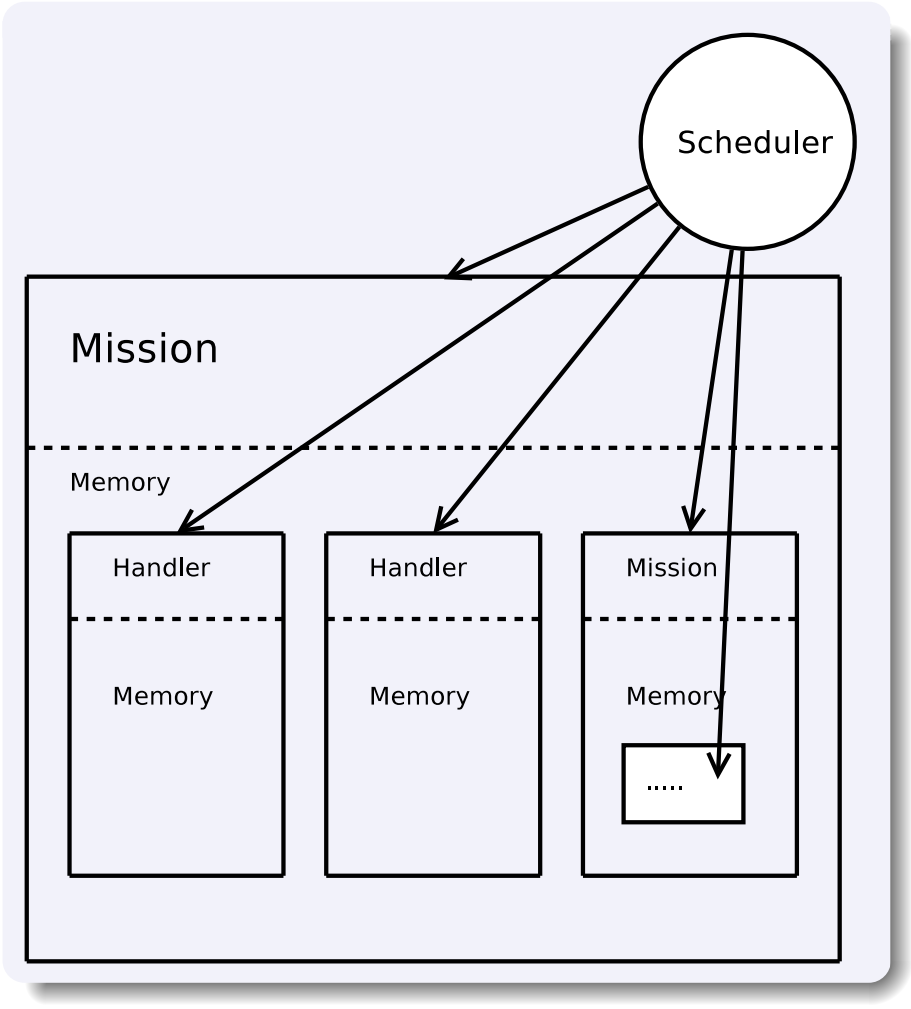
JTRES 2009, september

A constructive comment on SCJ

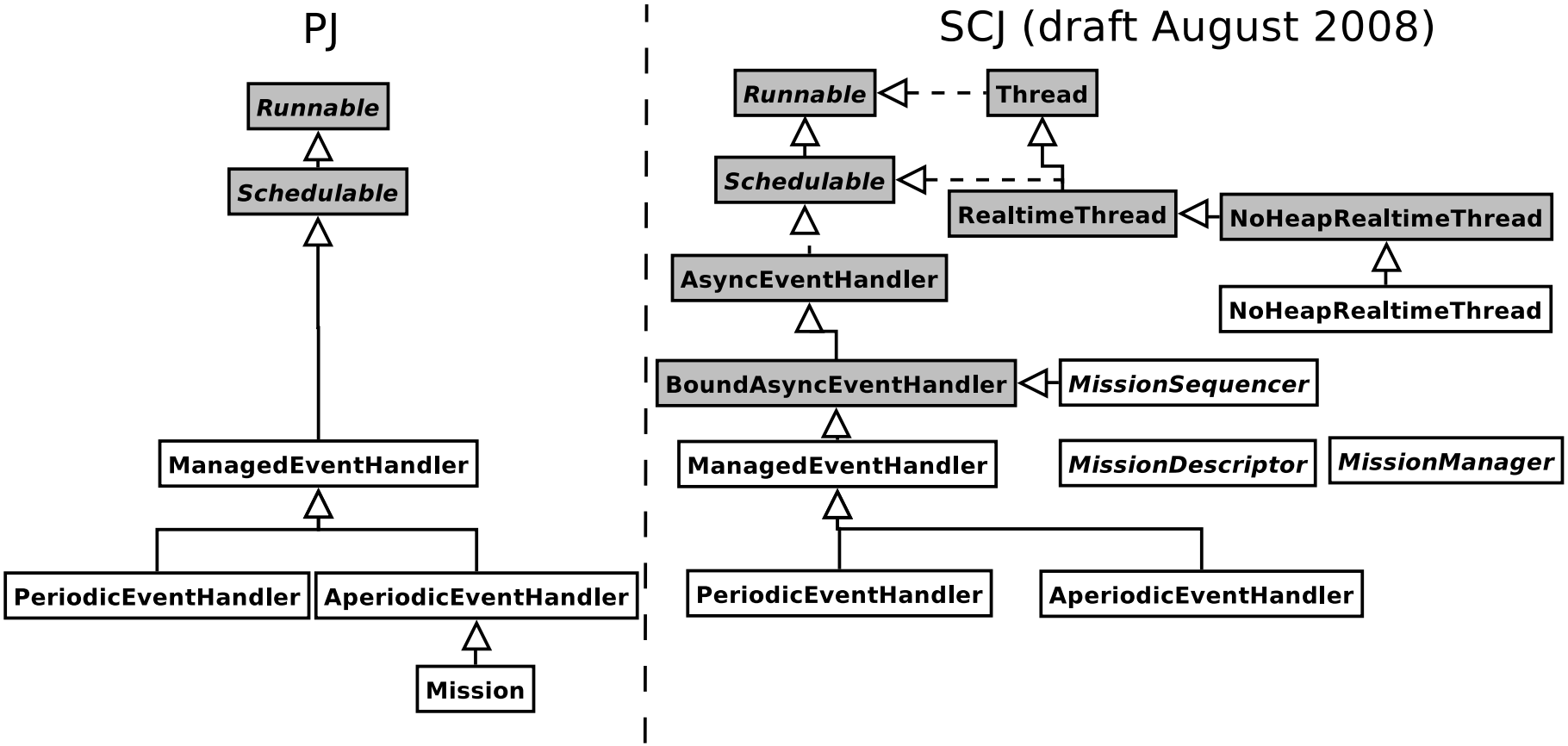
- Rational reconstruction of the class hierarchy
  - Generalization vs. specialization
- Missions
  - Mission concept as a handler
- RTSJ compliant
  - Delegation
- Result: Predictable Java

Predictable Java

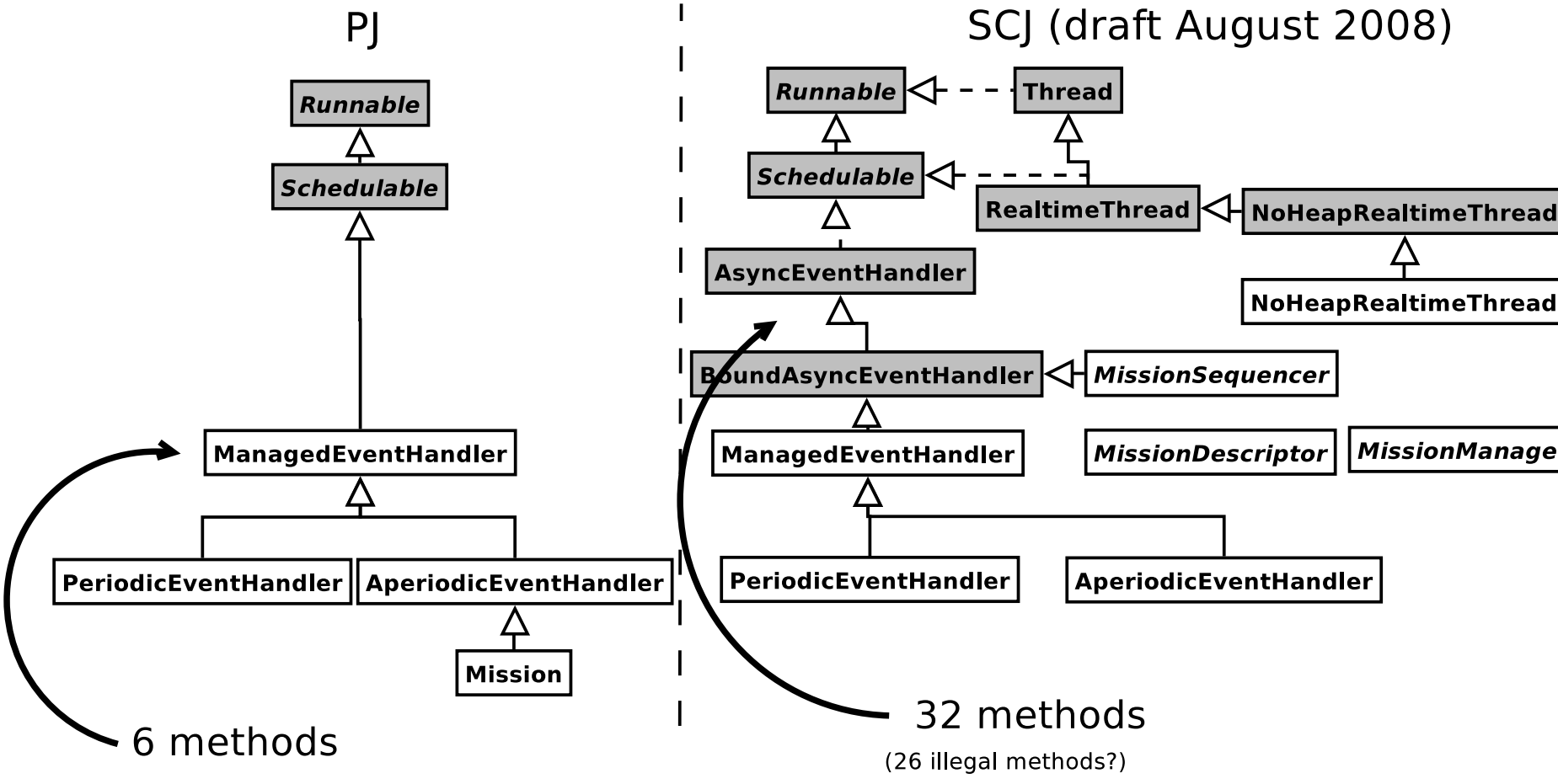
- Handlers
  - as in SCJ
- Mission
  - is a handler
- Schedulers
  - as in SCJ
- Memory areas
  - Simplified



Handlers

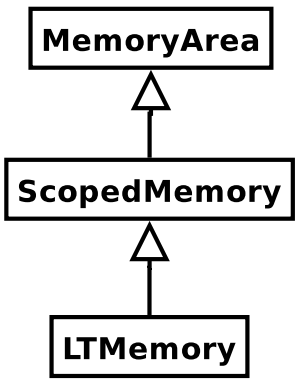

PJ
Runnable
Schedulable
ManagedEventHandler
PeriodicEventHandler
AperiodicEventHandler
Mission
SCJ (draft August 2008)
Runnable
Thread
Schedulable
RealtimeThread
NoHeapRealtimeThread
AsyncEventHandler
NoHeapRealtimeThread
BoundAsyncEventHandler
MissionSequencer
ManagedEventHandler
MissionDescriptor
MissionManager
PeriodicEventHandler
AperiodicEventHandler

Handlers

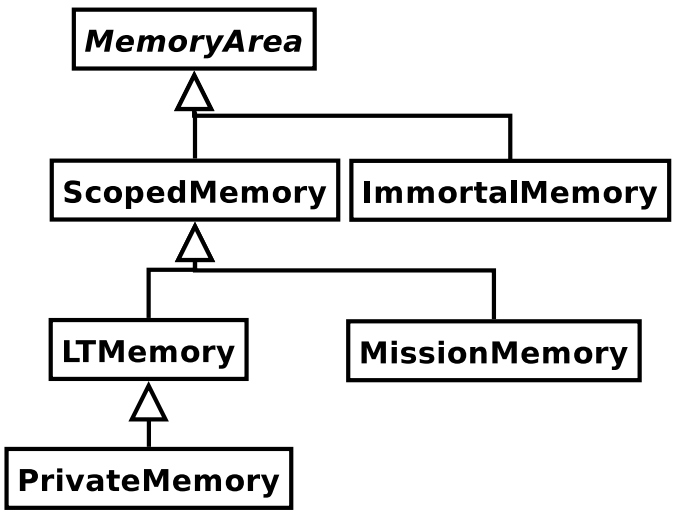

PJ
Runnable
Schedulable
ManagedEventHandler
PeriodicEventHandler
AperiodicEventHandler
Mission
6 methods
SCJ (draft August 2008)
Runnable
Thread
Schedulable
RealtimeThread
NoHeapRealtimeThread
AsyncEventHandler
NoHeapRealtimeThread
BoundAsyncEventHandler
MissionSequencer
ManagedEventHandler
MissionDescriptor
MissionManage
PeriodicEventHandler
AperiodicEventHandler
32 methods
(26 illegal methods?)

Memory

PJ



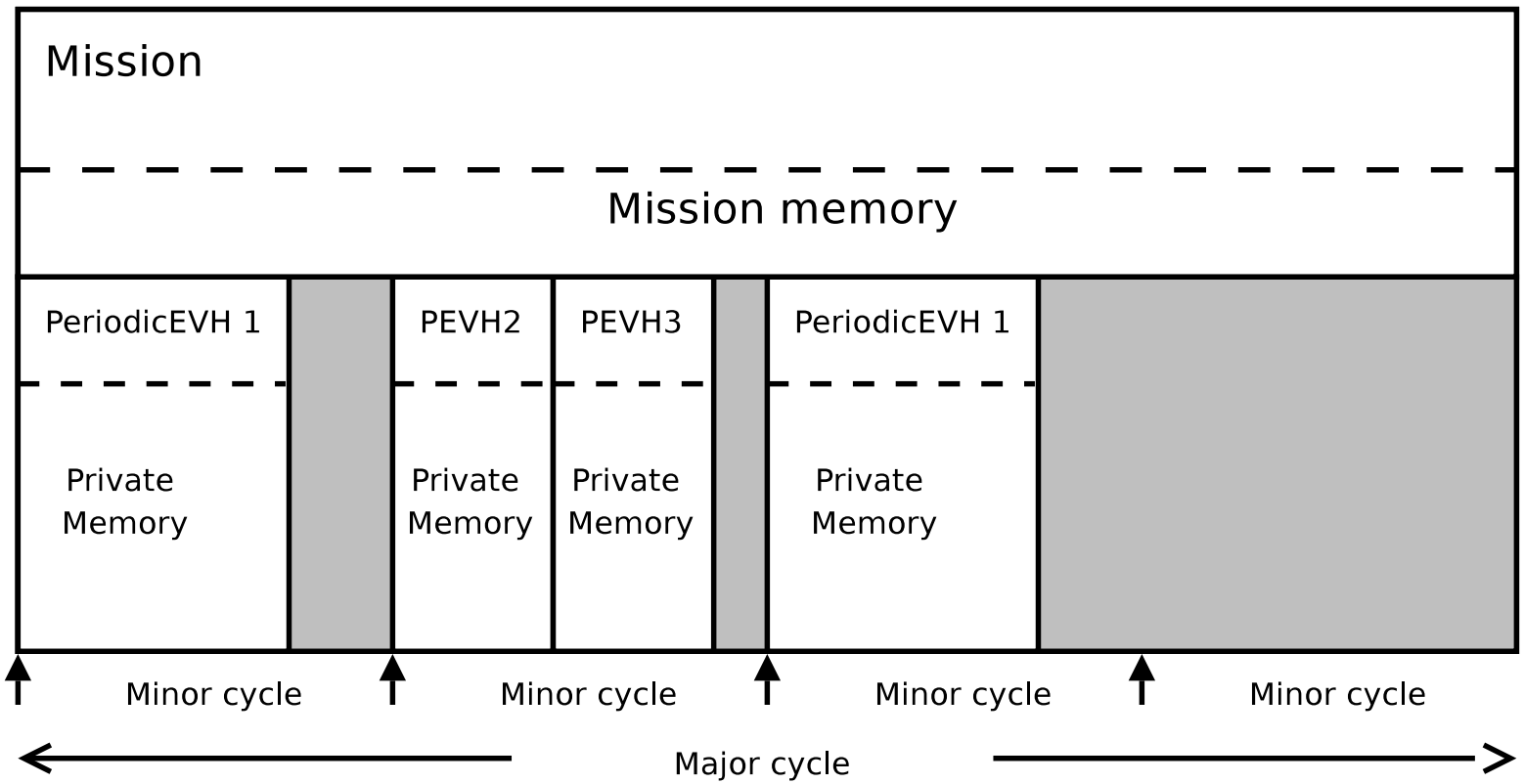
SCJ (draft August 2008)



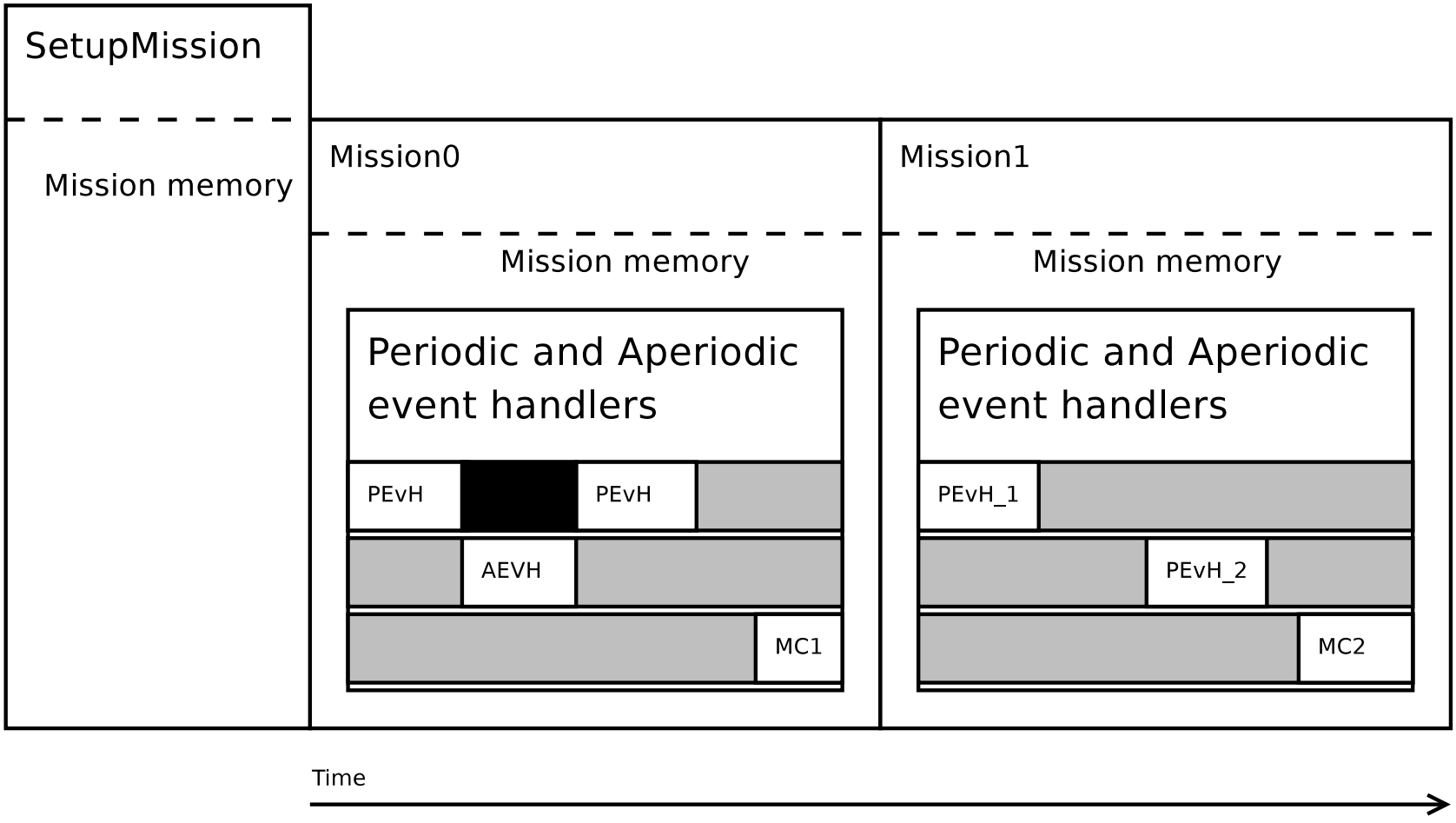
Scheduler

- Cyclic scheduler
- Priority scheduler

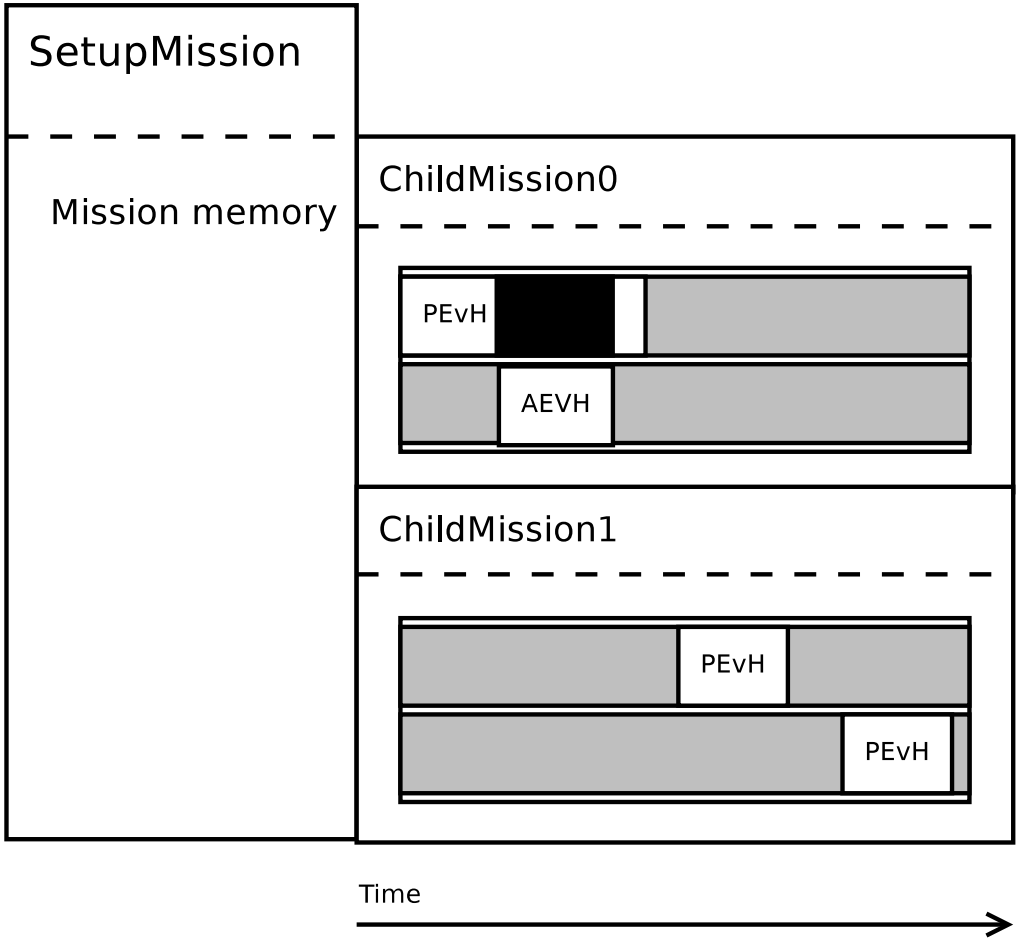
Level 0, Cyclic schedule



Level 1, Priority scheduler



Level 2, Priority scheduler



Simple mission

```
public class Basic extends Mission
{
  protected Basic(.., AperiodicParameters ap,
                  Scheduler scheduler,
                  MemoryArea memoryArea)
  {
    ... // initialization
  }

  public static void main (String[] args) {
    new Basic( null, null,
               new CyclicScheduler(),
               new LTMemory(10*1024));
  }
}
```

Simple mission initialization

```
addToMission(
  new Periodic( null, pp, getScheduler(),
                new LTMemory(1024)));
addToMission(
  new Periodic( null, pp, getScheduler(),
                new LTMemory(1024)));
...
add(); // add mission to its scheduler
```

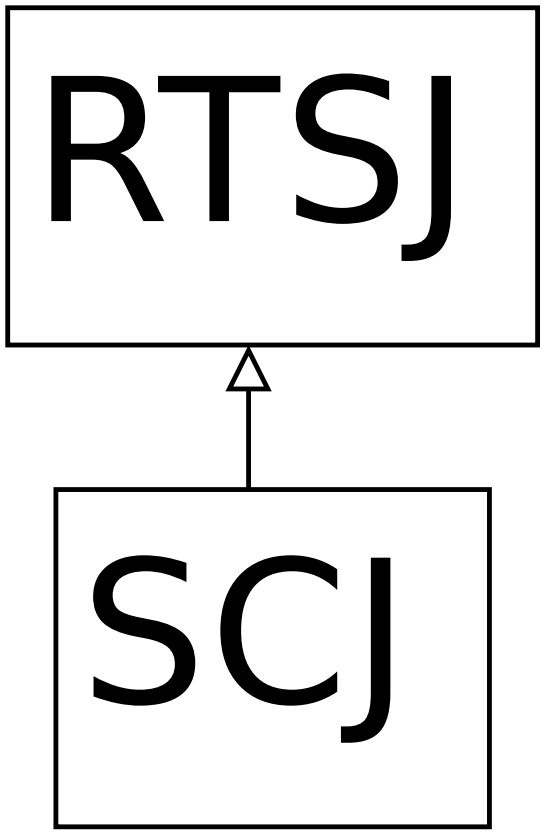
Sequential missions setup

```
private Mission[] mission;
private int active = 0;
static AperiodicEvent event;

public SequentialMissions(...) {
  mission = new Mission[2];

  mission[0..1] = new Mission(...);

  event = new AperiodicEvent(this);

  mission[active].add();
}
```

Mission change

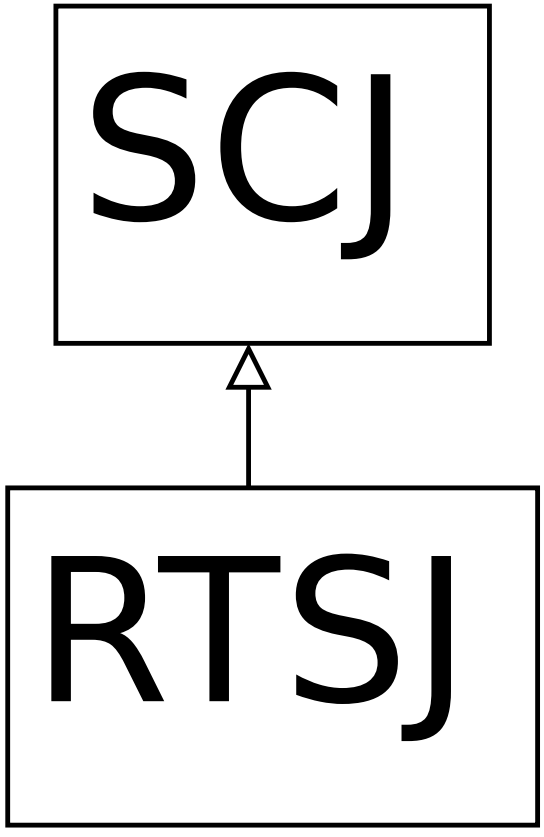
```
private Mission[] mission;
private int active = 0;
static AperiodicEvent event;

public SequentialMissions(...) {
   ...
}

public void handleEvent() {
   mission[active].remove();
   active = (active + 1) % mission.length;
   mission[active].add();
}
```

The SCJ approach



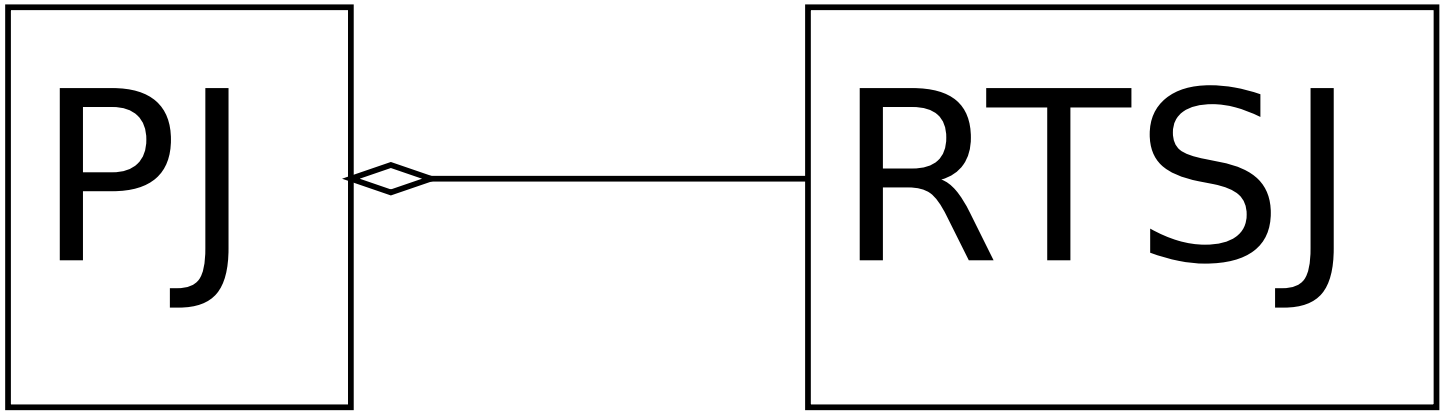
- Inheritance for limitation

Ideal world



- Inheritance for specialization

Implementation approach



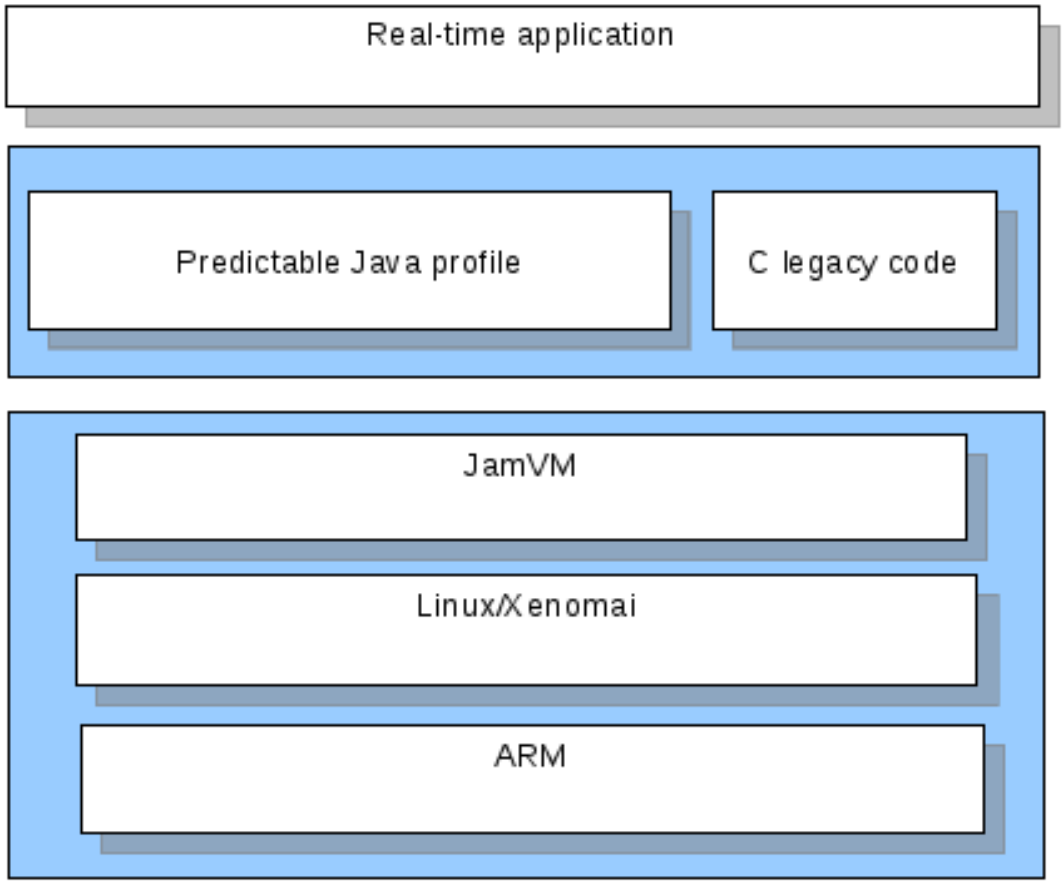
- Aggregation

RTSJ implementation

- Event Handlers
  - NoHeapRealtime Threads
- Schedulers
  - RTSJ Scheduler and NoHeapRealtime Threads
- Memory
  - Immortal and Scoped memory

Native implementation

- JamVM
- Real-time linux
- ARM
- Open source



Planned next steps

- Improved implementations
- Profile compliance checker
- Memory analysis
- WCET analysis
- Schedulability analysis

As Eclipse plugins – a workbench

Predictable Java conclusion

- A constructive comment on SCJ
- Simpler framework
- Experimental implementations
- Workbench