This document shows about the functionalities of RTSJ. Like Async function, heap and all.

**Heap memory:**

The run-time implementations of most programming languages provide a large amount of memory (called the heap) so that the programmer can make dynamic requests for chunks to be allocated (for example, to contain an array whose bounds are not known at compile time). Runtime system JVM is responsible for allocating the heap memory and taking care of it. (Like how much space is required and when allocated memory can be released or reused). All objects in Java are allocated on the heap, and the language requires garbage collection for an effective implementation.

**Memory areas in RTSJ:**

* **Heap memory**- Heap memory allows objects to be allocated in the standard Java heap.
* **Immortal memory**- Immortal memory is shared among all threads in an application. Objects created in immortal memory are never subject to garbage collection delays and behave as if they are freed by the system only when the program terminates.
* **Scoped memory-** Scoped memory is a memory area where objects with a well-defined lifetime can be allocated. A scoped memory may be entered explicitly or implicitly by

attaching it to a real-time entity (a real-time thread or an asynchronous event handler)

at its creation time. Associated with each scoped memory is a reference count. This

keeps track of how many real-time entities are currently using the area. When the

reference count reaches 0, all objects resident in the scoped memory have their

finalization method executed, and the memory is available for reuse. The ScopedMemory

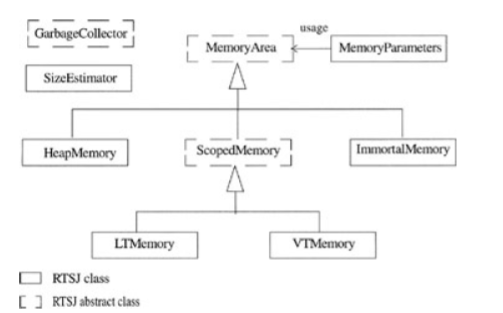
class is an abstract class that has several subclasses.

* **VT Memory-** A subclass of ScopedMemory where allocations may take variable amounts

of time.

* **LT memory-** A subclass of ScopedMemory where allocations occur in linear time (that is, the time taken to allocate the object is directly proportional to the size of the object).

Memory parameters can be given when real time threads or asynchronous event handlers are created.



**Object has 3 things:**

* release requirement (that is, when it should become runnable), [periodic, aperiodic, sporadic]
* memory requirements (for example, the rate at which it will allocate memory on the
* heap),
* scheduling requirements (for example, the priority at which it should be scheduled).

Scheduling parameters are used by a scheduler to determine which object is currently the most eligible for execution.

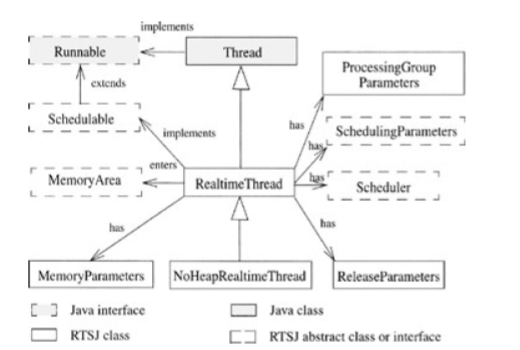
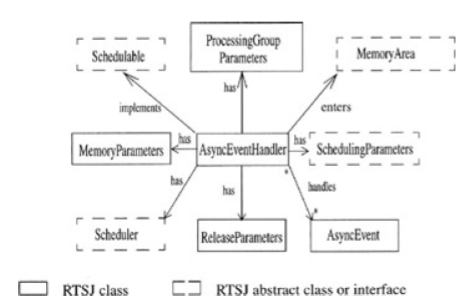


Figure : Real time thread parameters

A periodic real-time thread is a real-time thread that has periodic release parameters. Similarly, an aperiodic (or sporadic) real-time thread is one that has aperiodic (or sporadic) release parameters. A NoHeapRealtimeThread is one that guarantees not to create or reference any objects on the heap. Hence, its execution is totally independent of the garbage collector.

**Asynchronous Event handling:** (Simple words it can be given when we associate with interrupt)

However, it is also often necessary to respond to events that happen asynchronously to a thread's activity. These events may be happenings in the environment of an embedded system or notifications received from internal activities within the system.



Each AsyncEvent can have one or more handlers, and the same handler can be associated with more than one event. When the event occurs, all the handlers associated with the event are released for execution according to their Scheduling-Parameters. Asynchronous event will have a linked timer the timer will cause the event to fire when a specified time (relative to a particular clock) expires.

