

Realtime Garbage Collection in the JamaicaVM 3.0

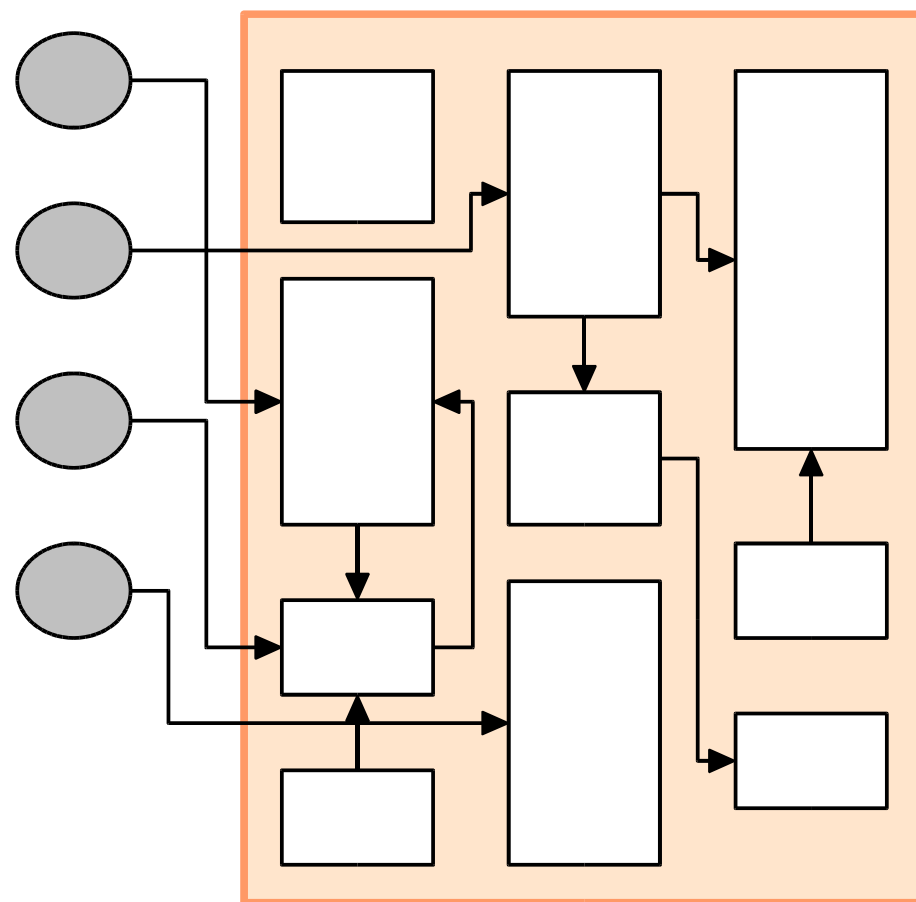
....Realtime Java without IllegalAssignmentErrors



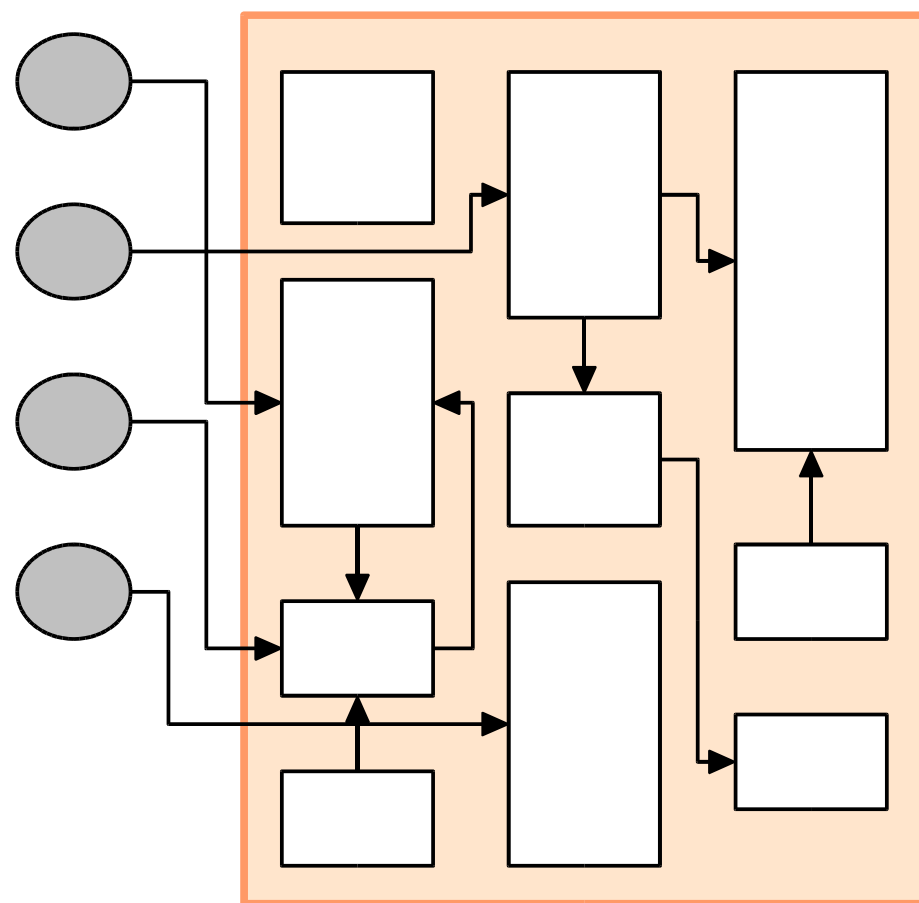
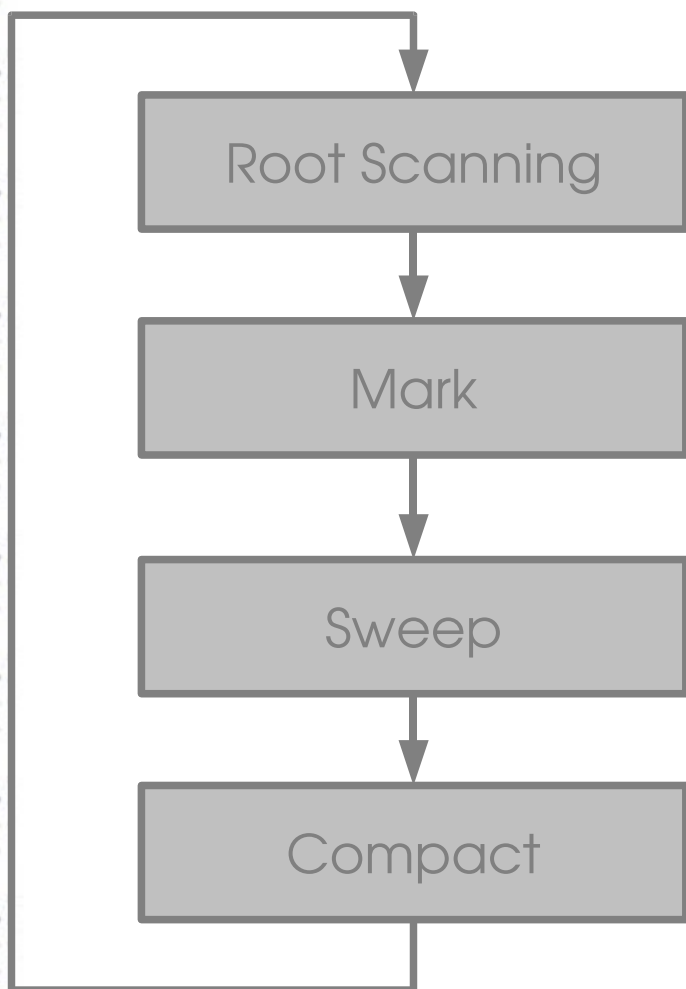
Dr. Fridtjof Siebert
CTO, aicas
Wien, 26. September 2007

How does the GC work?

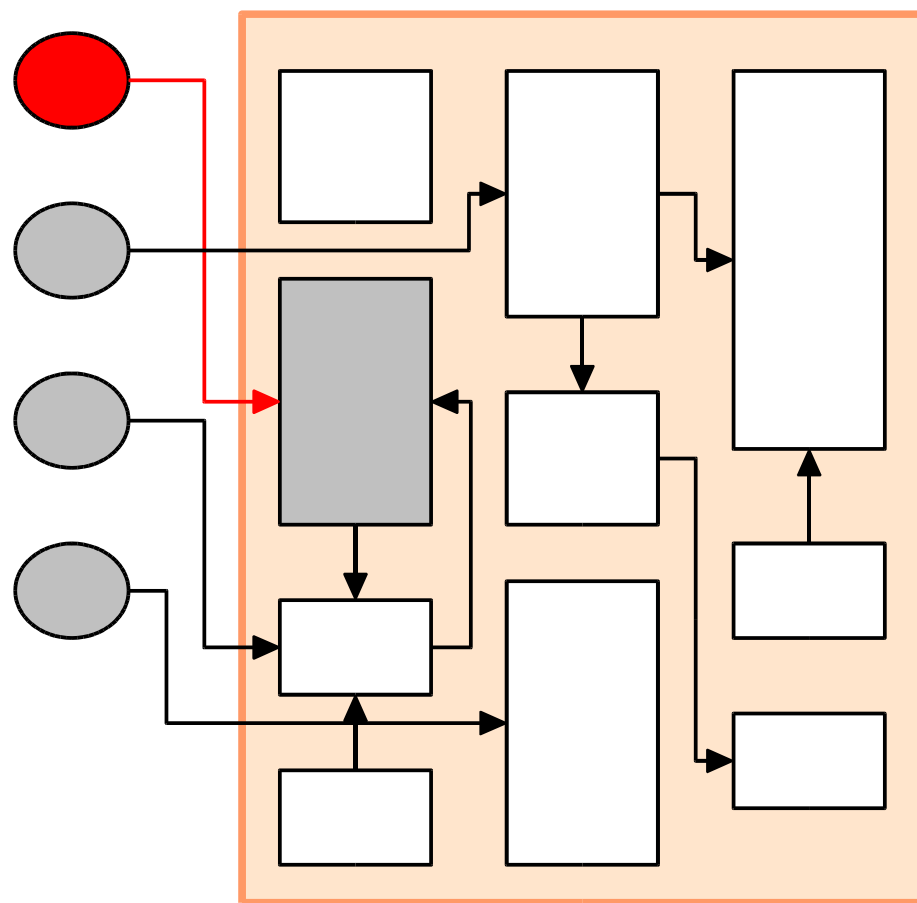
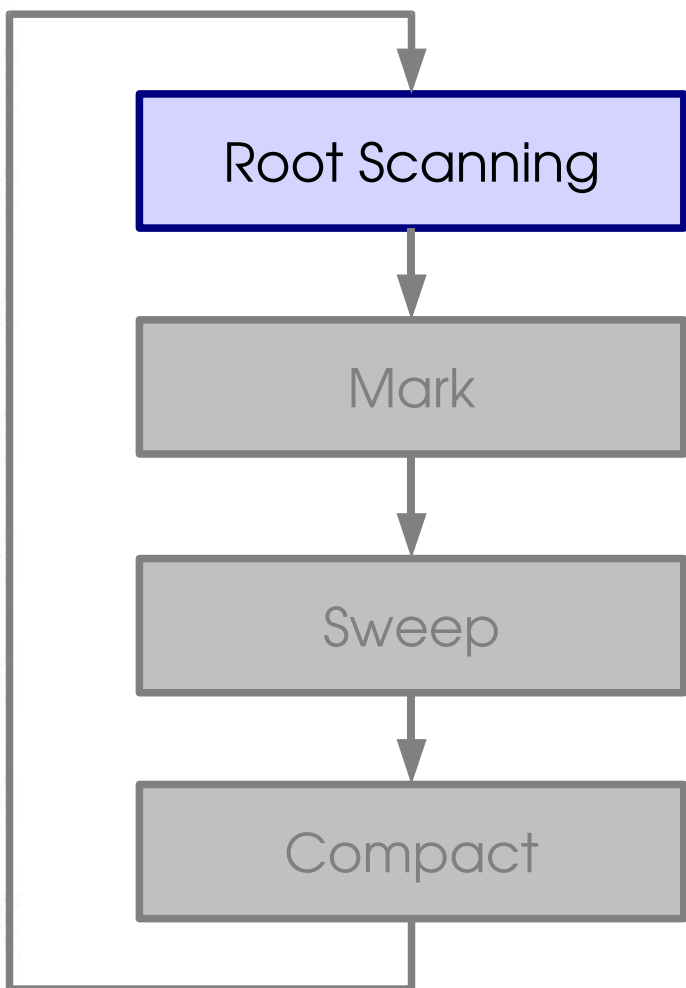
Classic Mark & Sweep GC



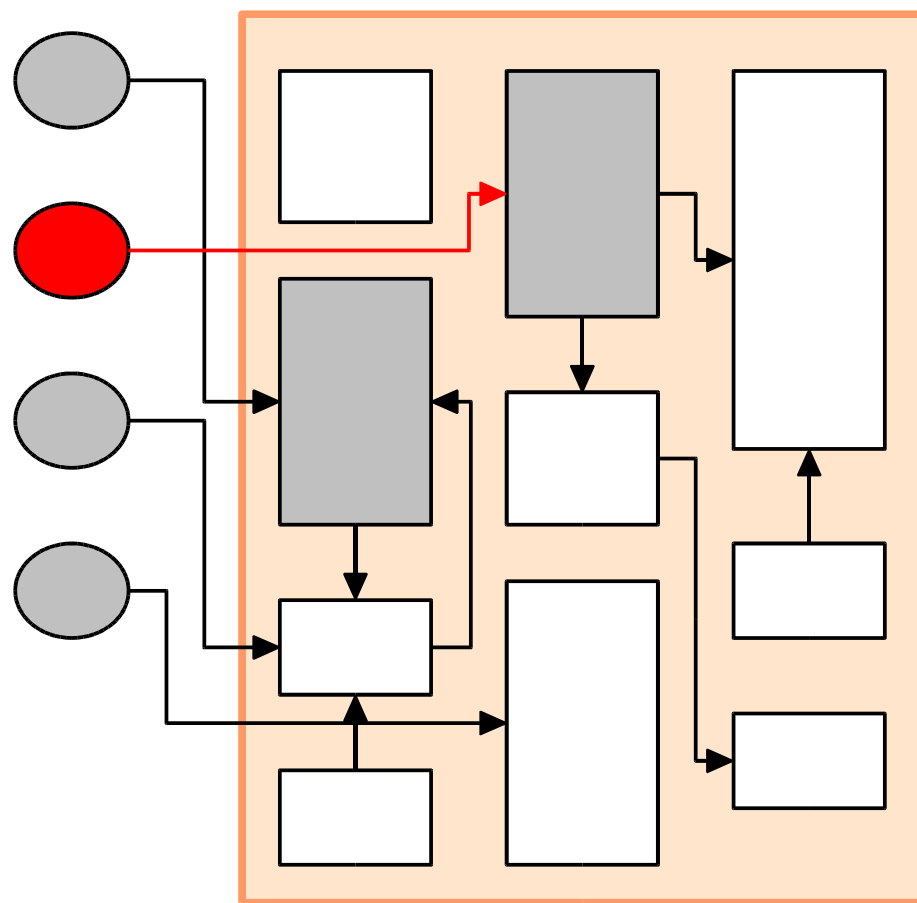
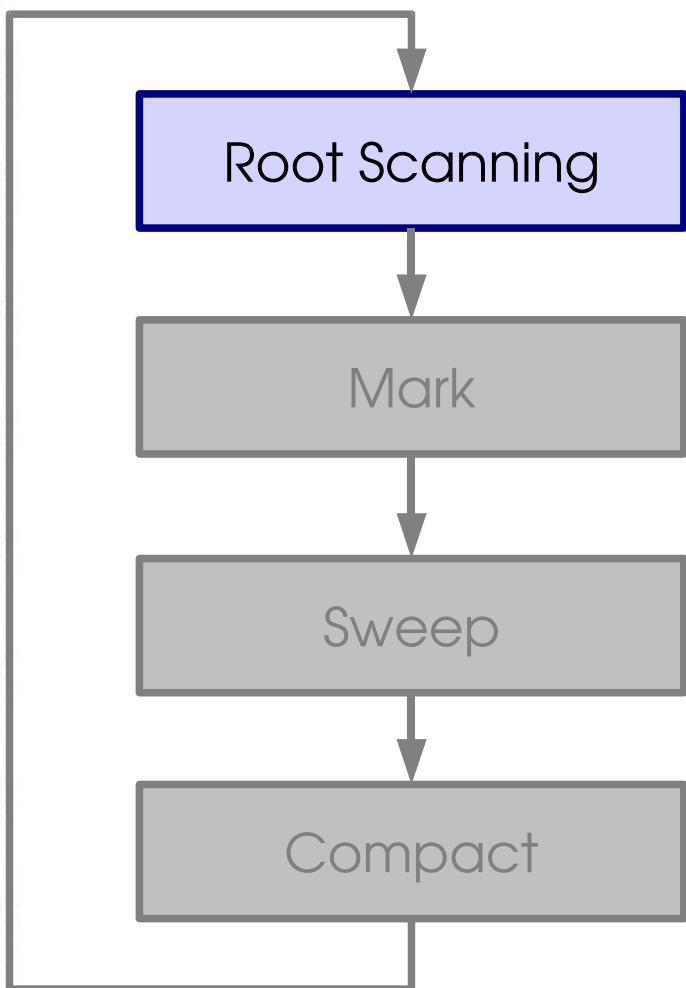
Classic Mark & Sweep GC



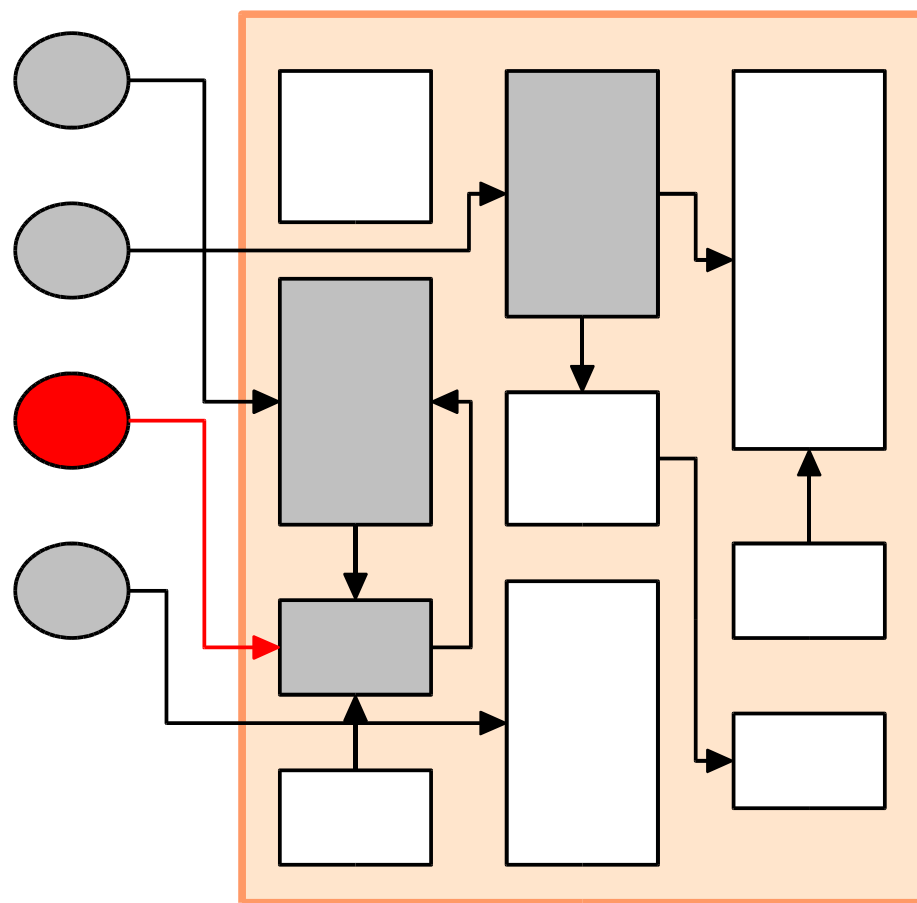
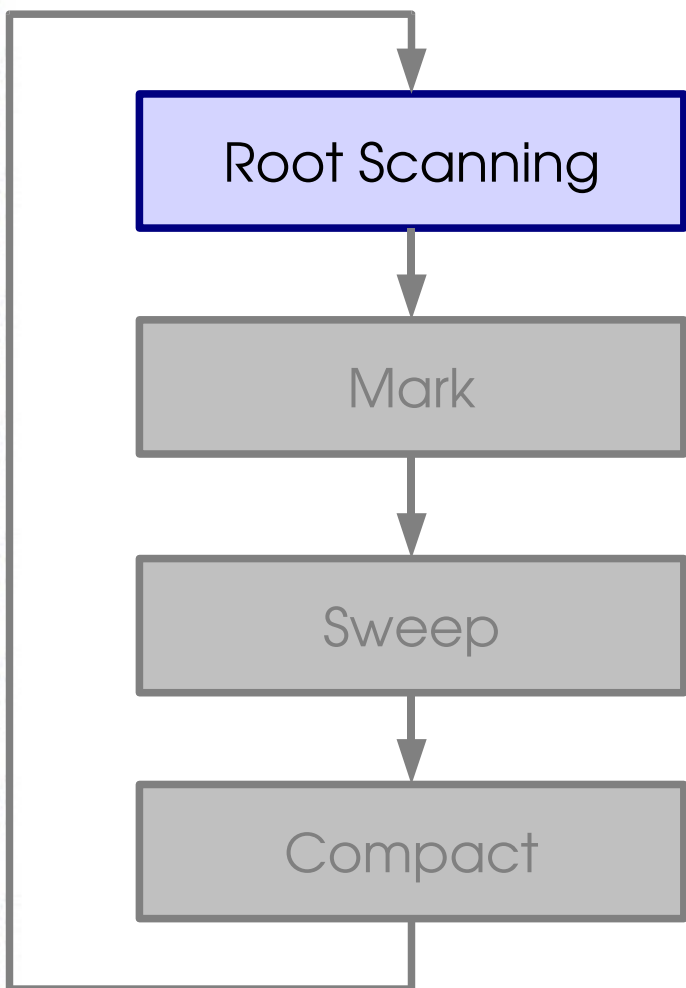
Classic Mark & Sweep GC



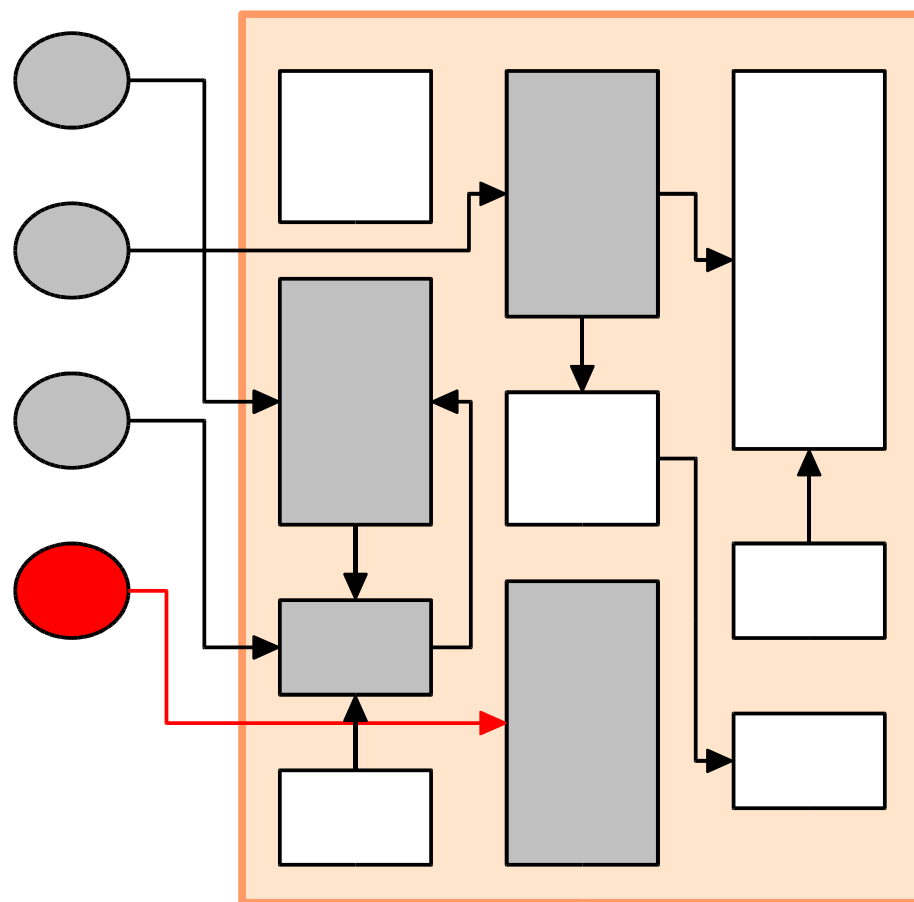
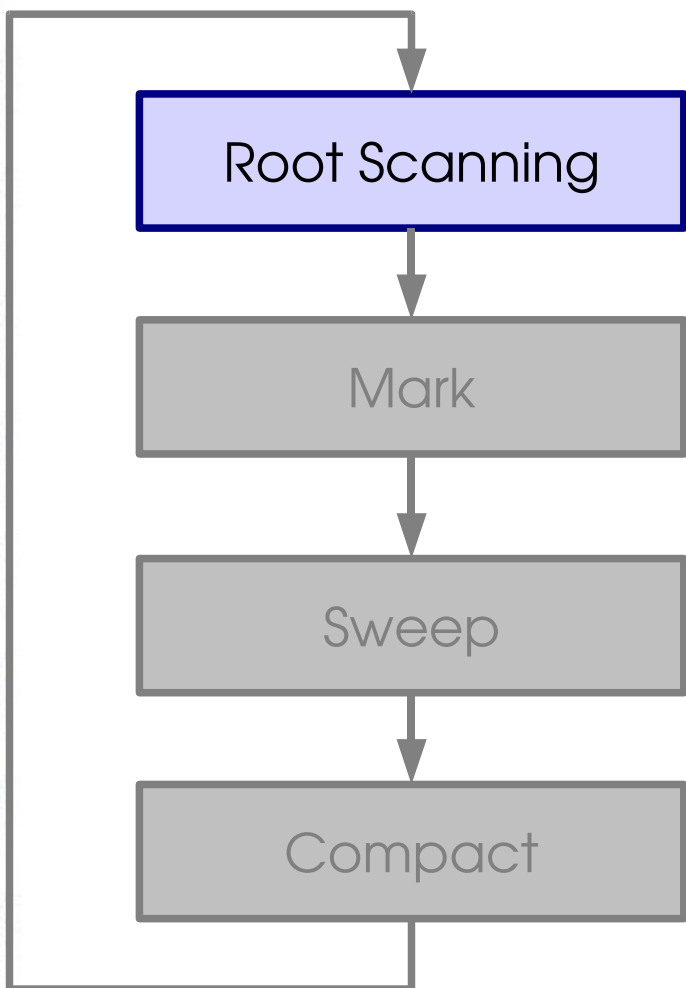
Classic Mark & Sweep GC



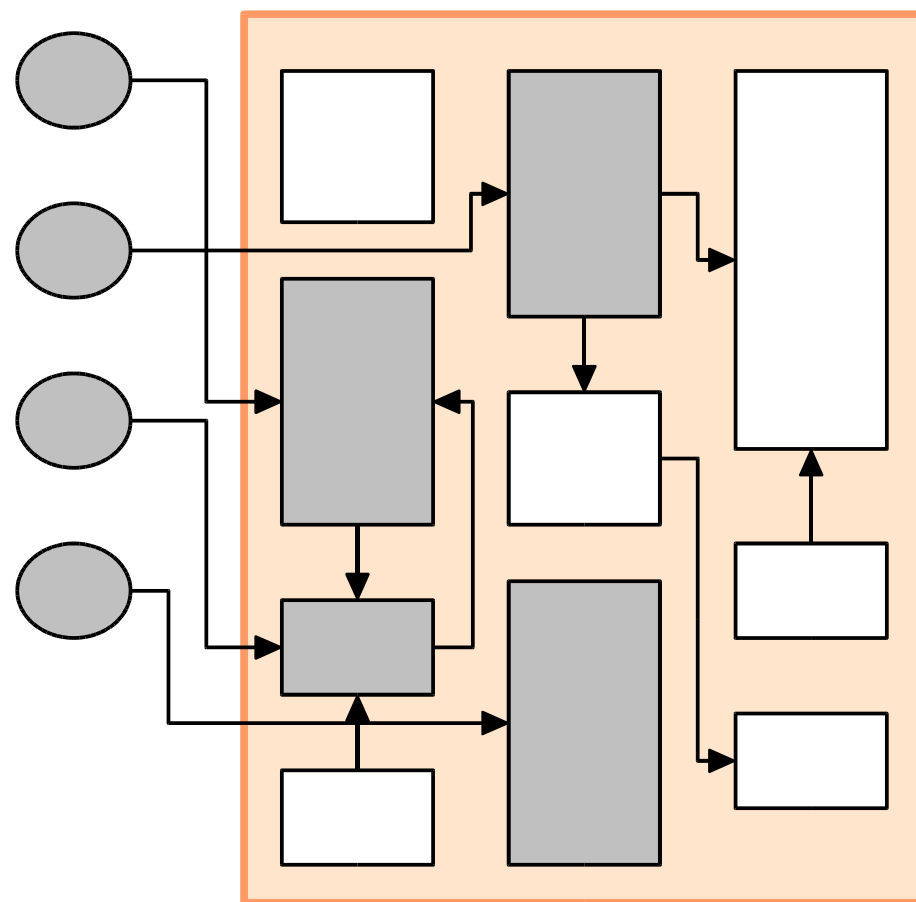
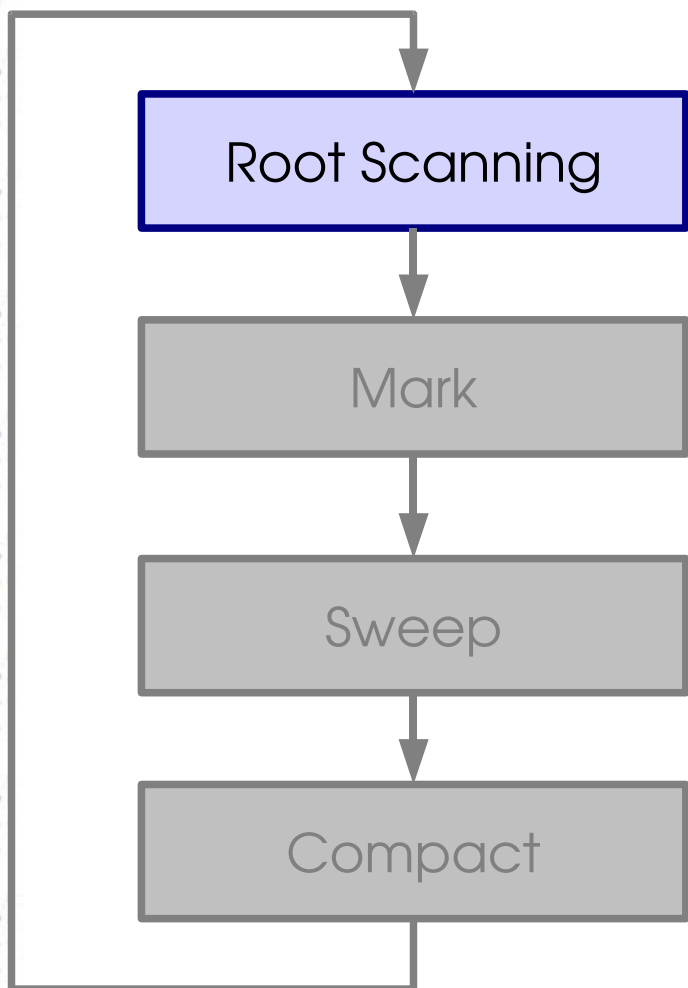
Classic Mark & Sweep GC



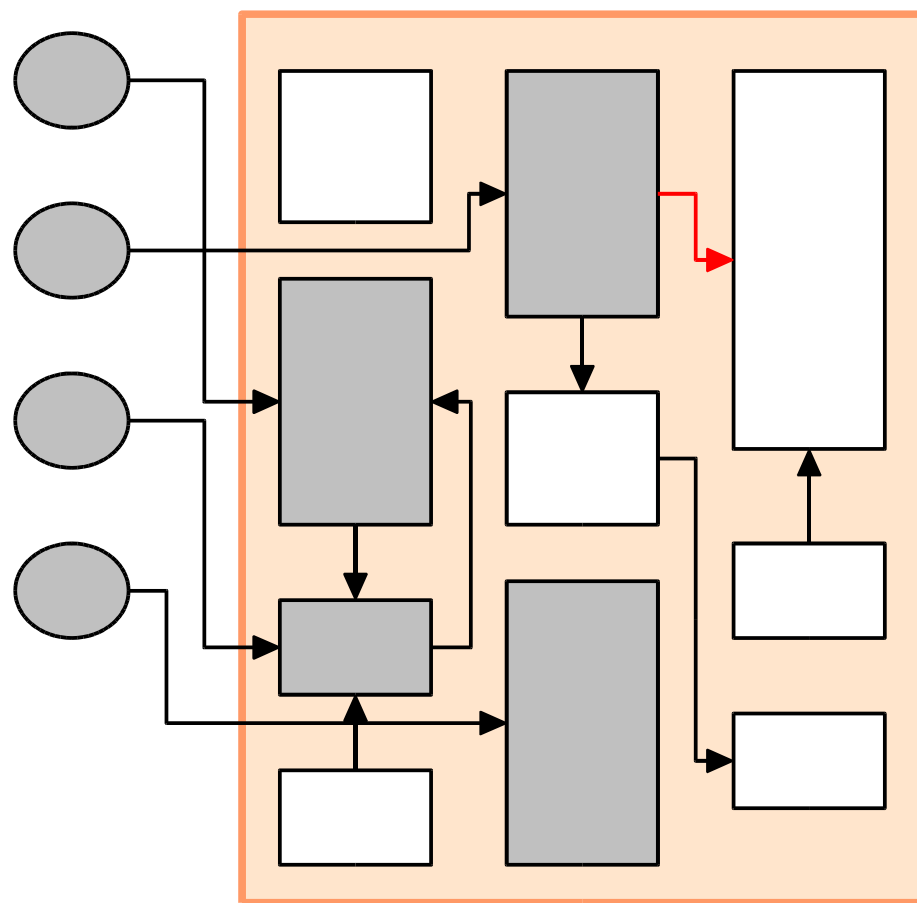
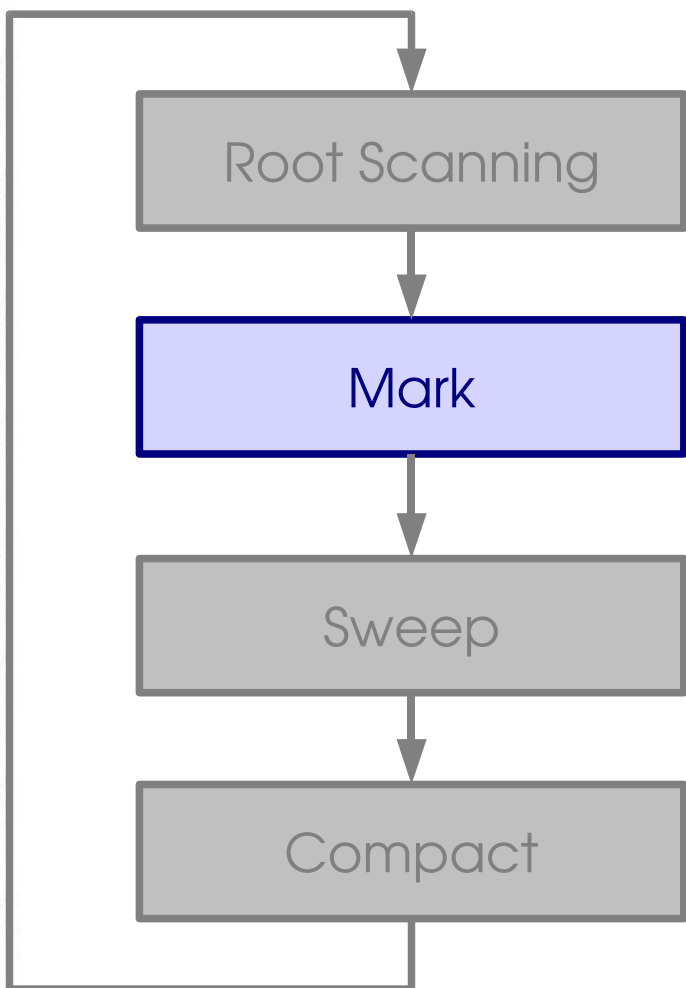
Classic Mark & Sweep GC



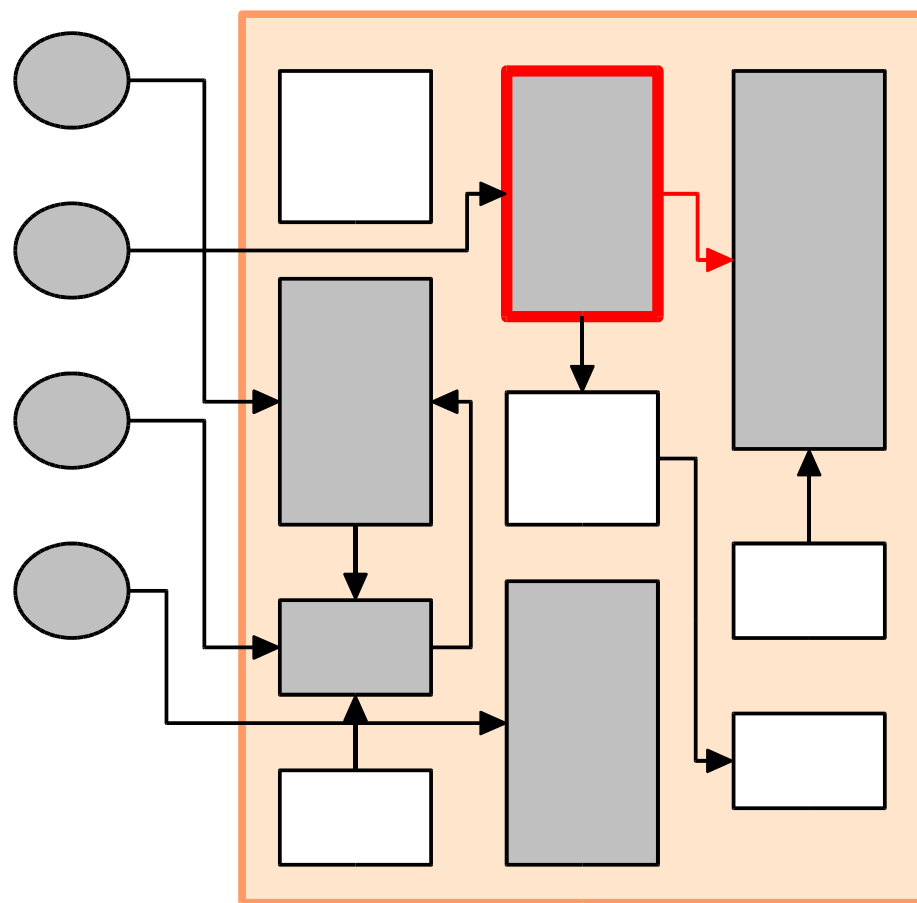
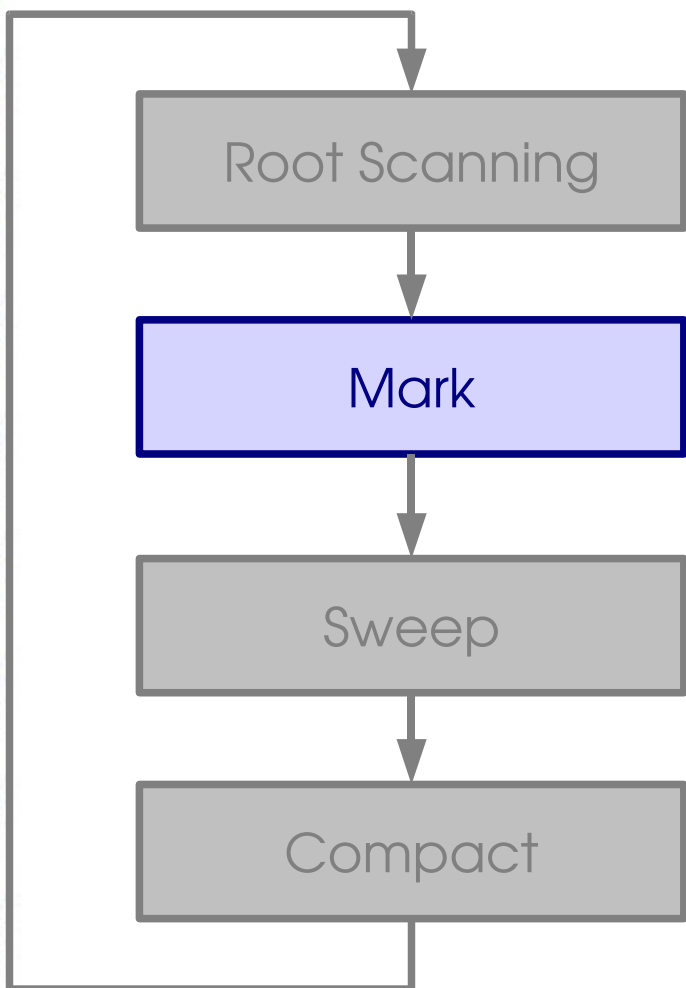
Classic Mark & Sweep GC



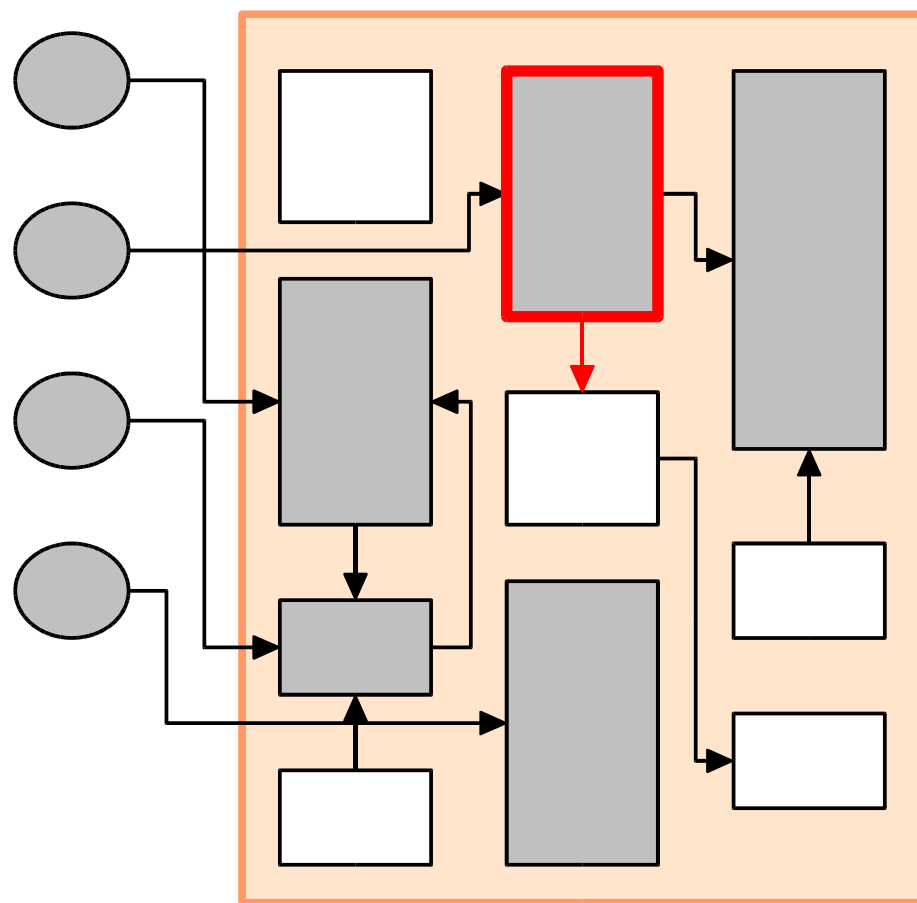
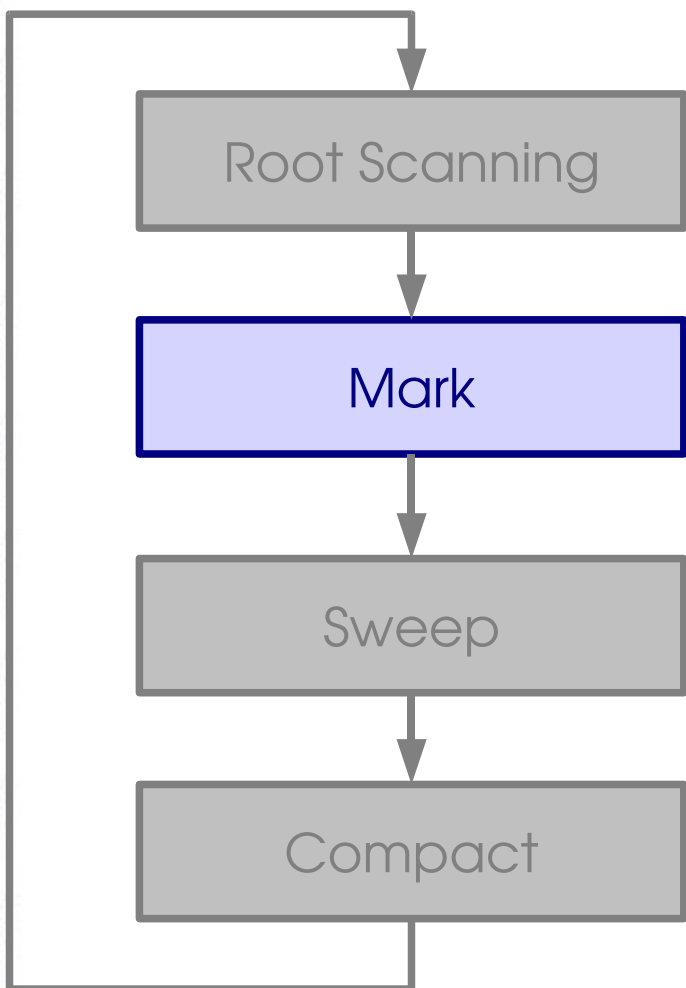
Classic Mark & Sweep GC



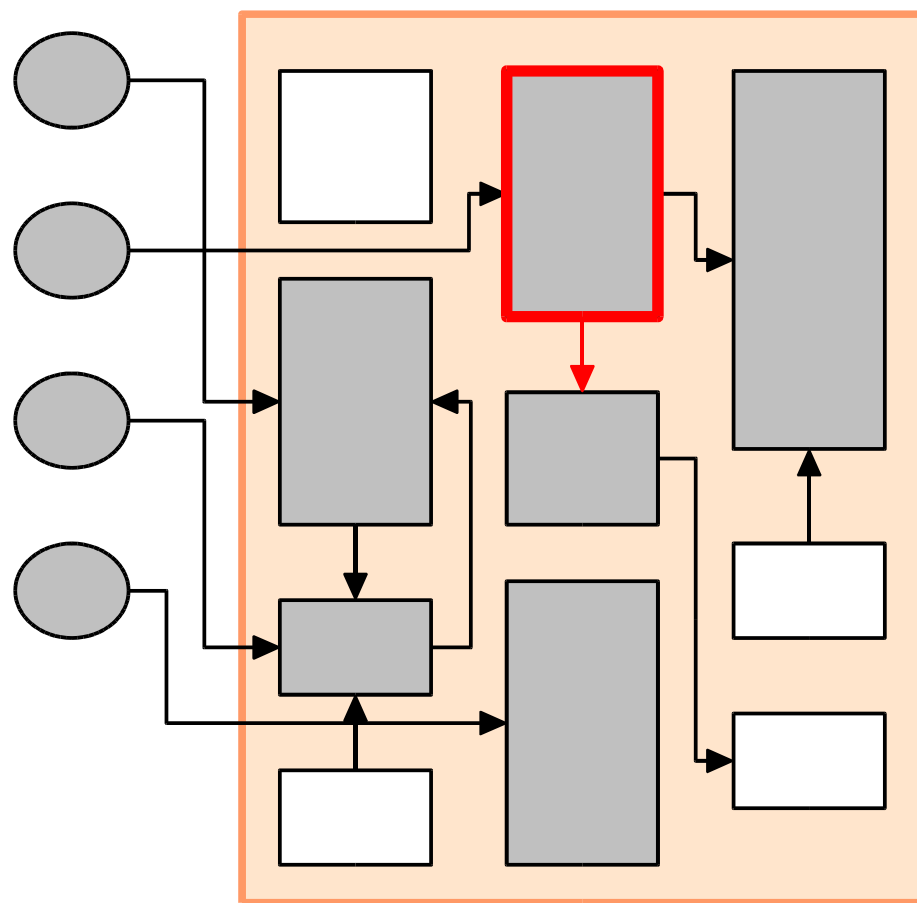
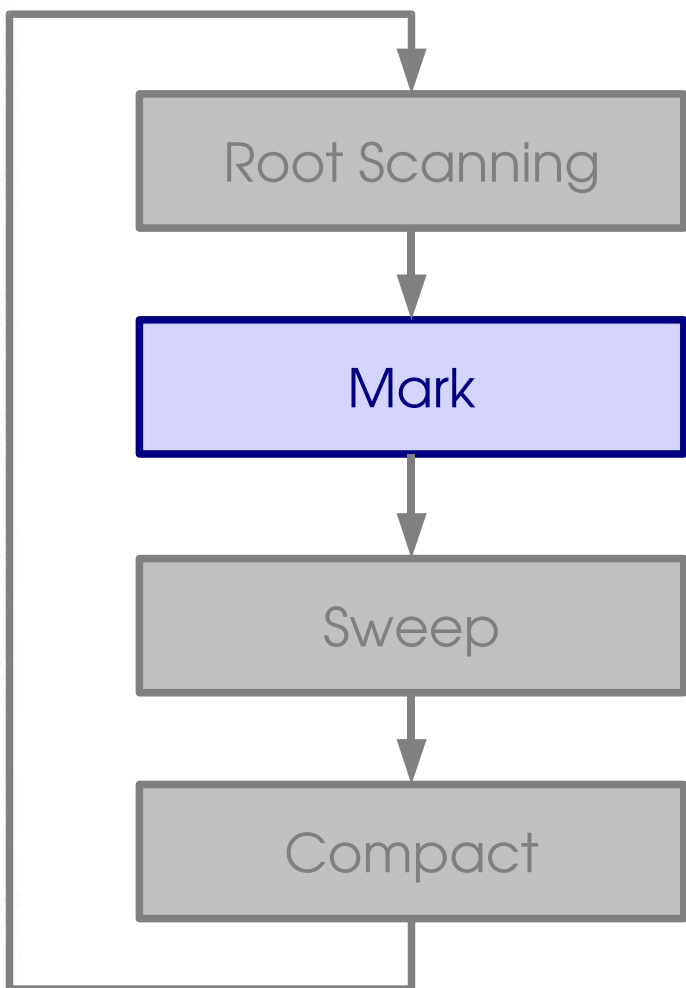
Classic Mark & Sweep GC



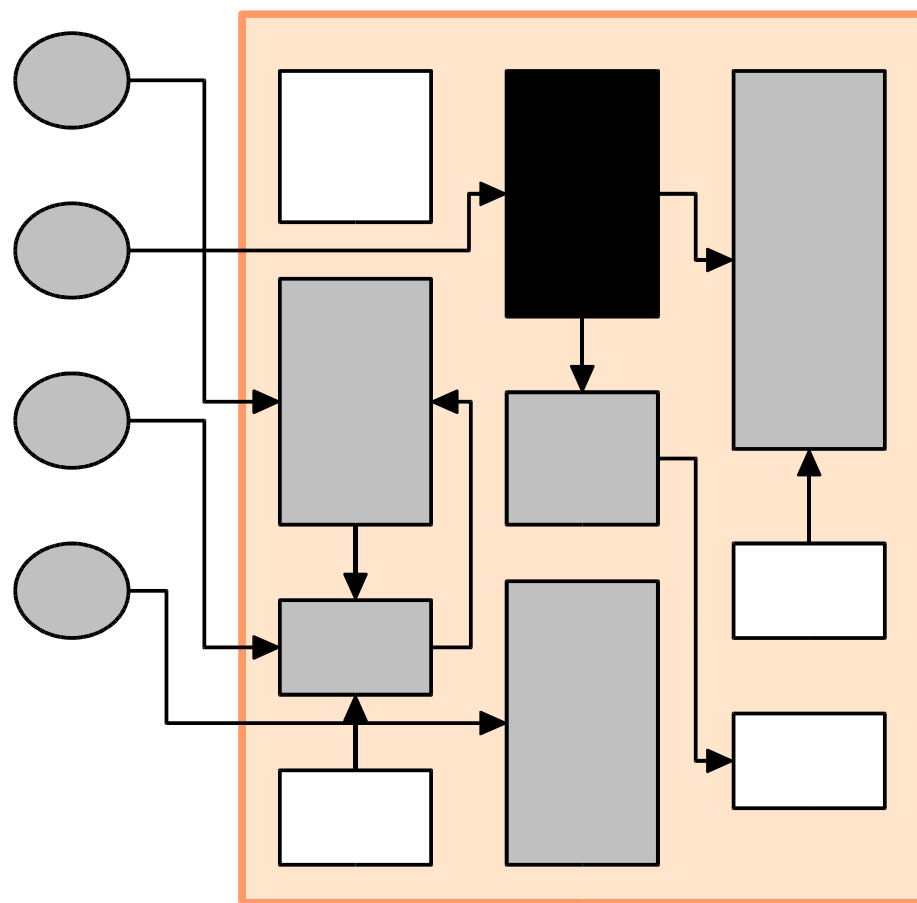
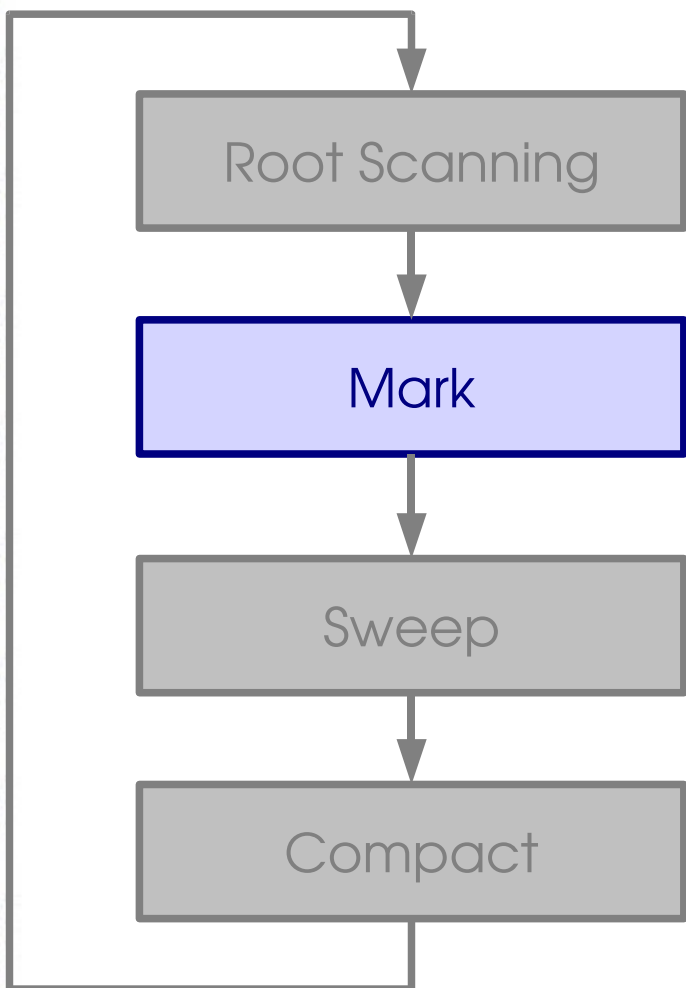
Classic Mark & Sweep GC



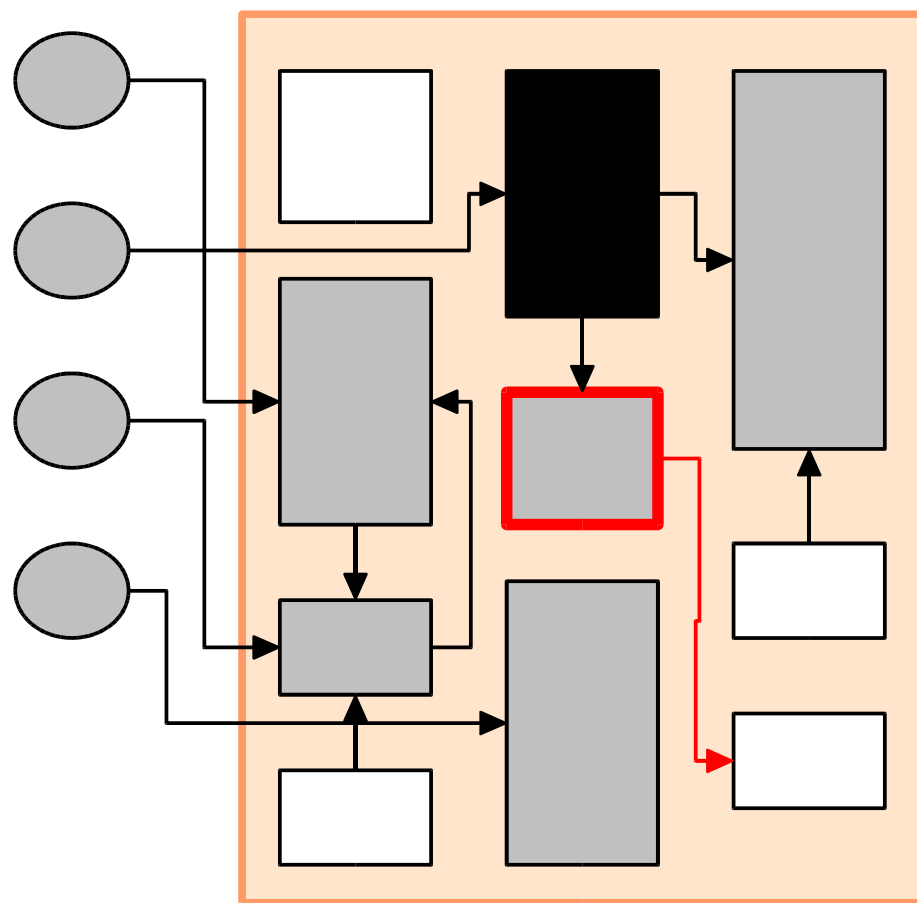
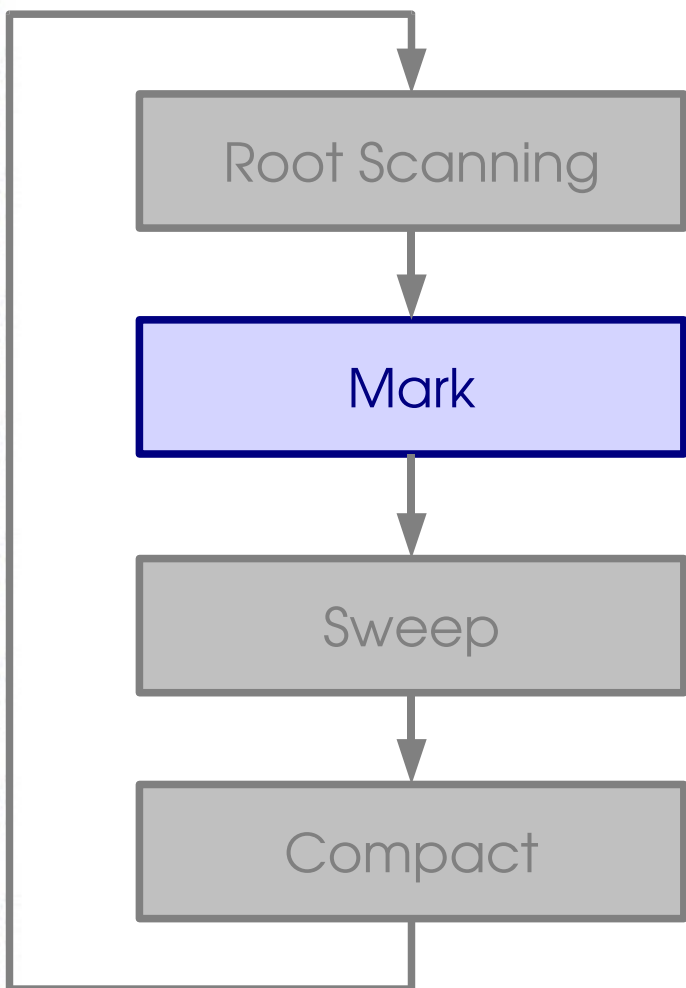
Classic Mark & Sweep GC



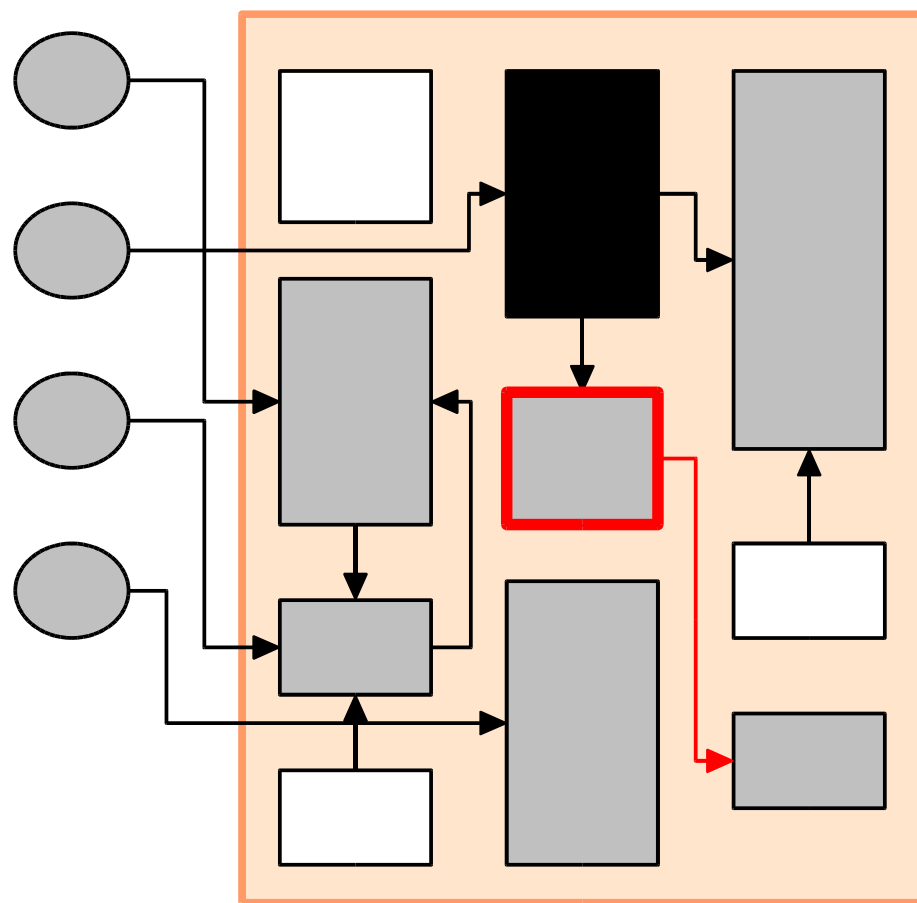
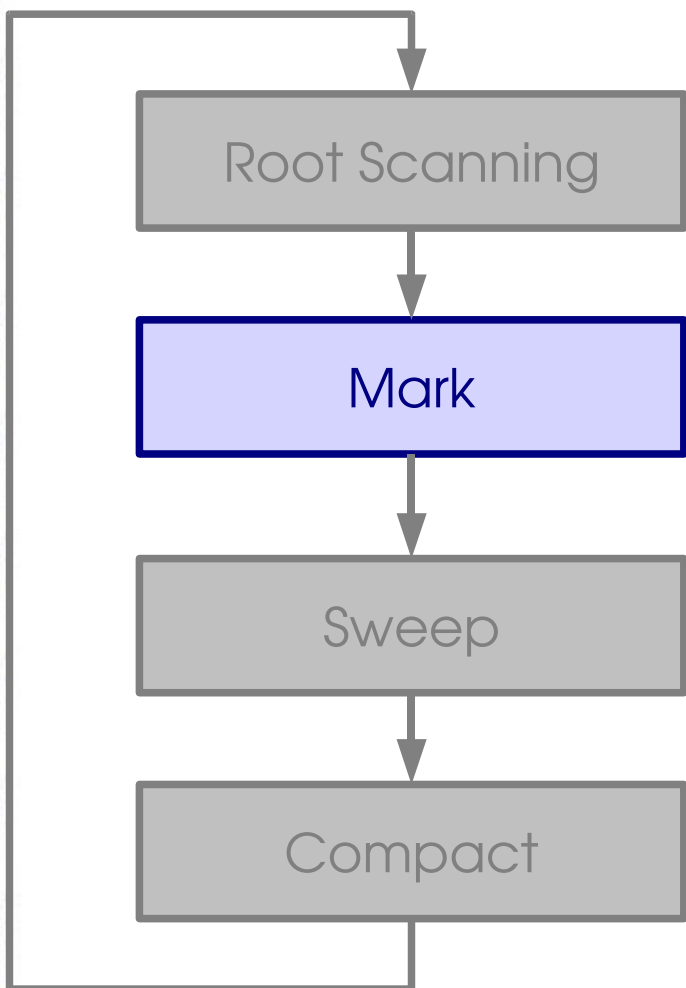
Classic Mark & Sweep GC



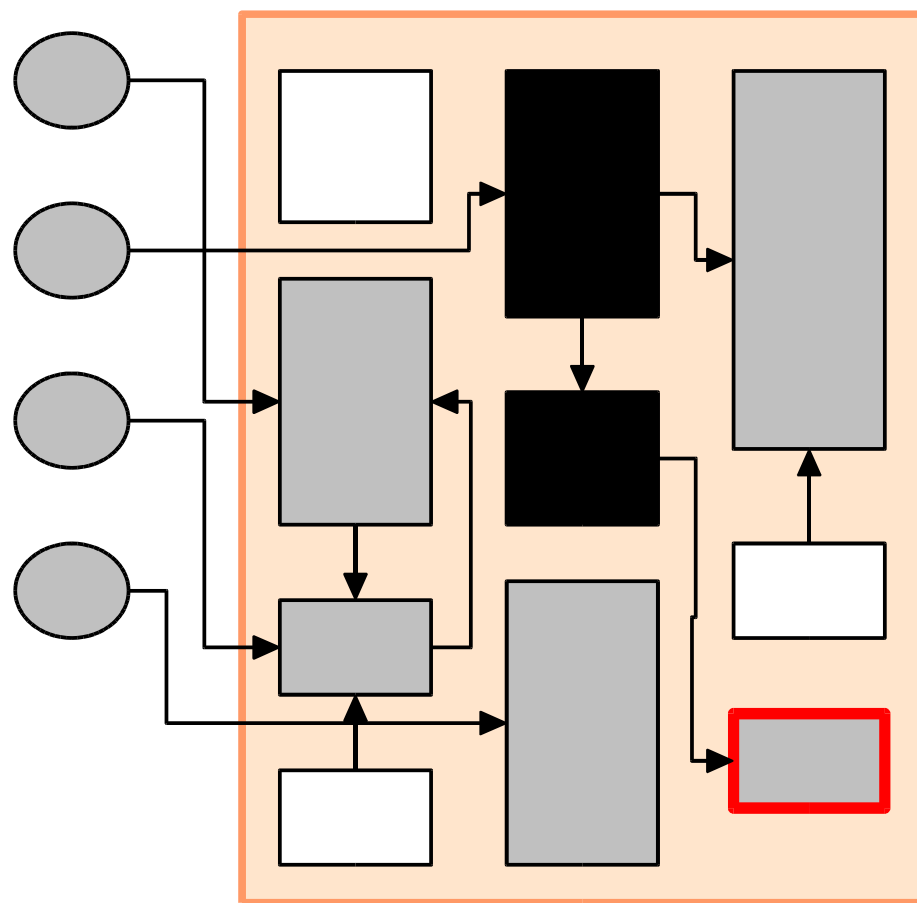
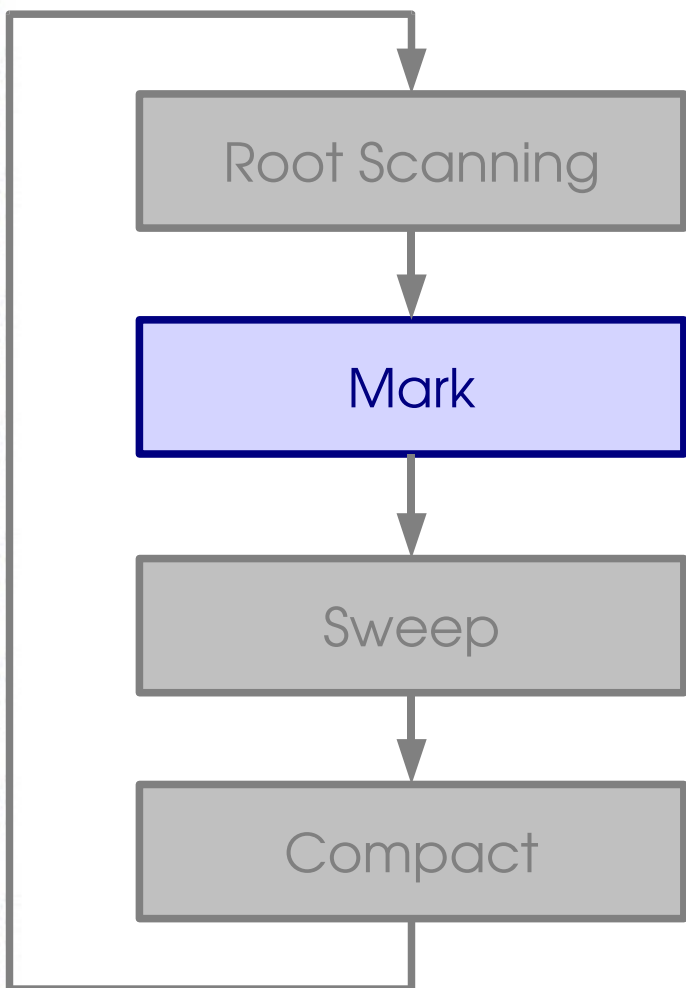
Classic Mark & Sweep GC



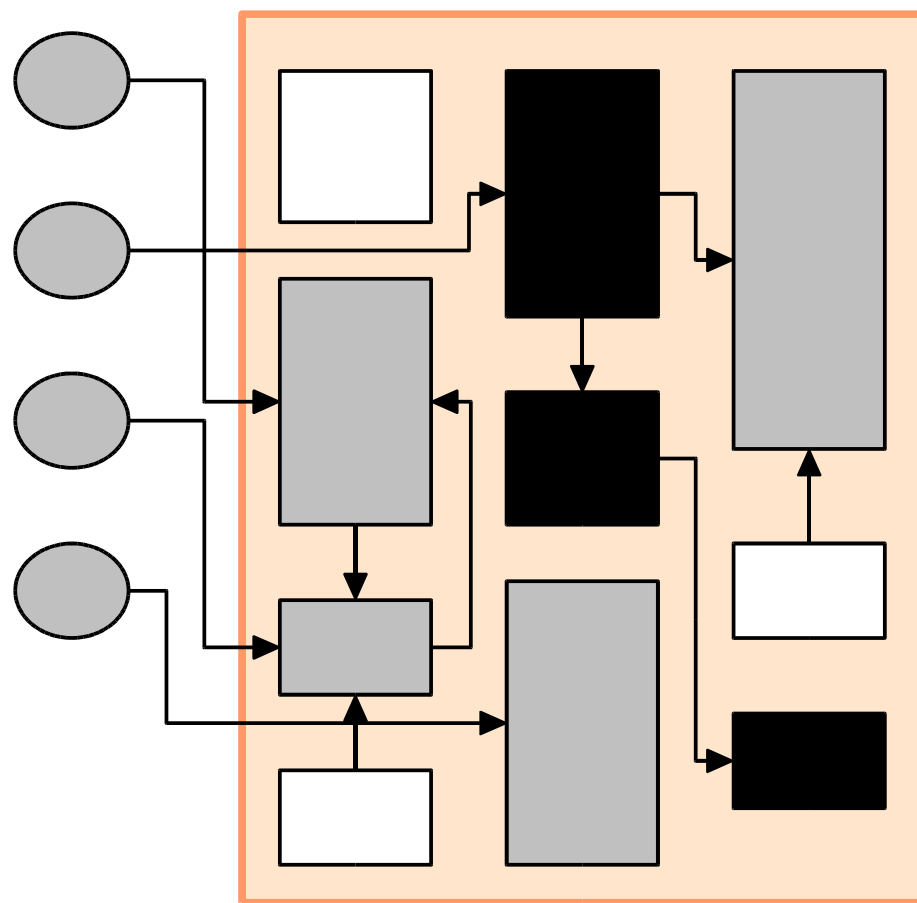
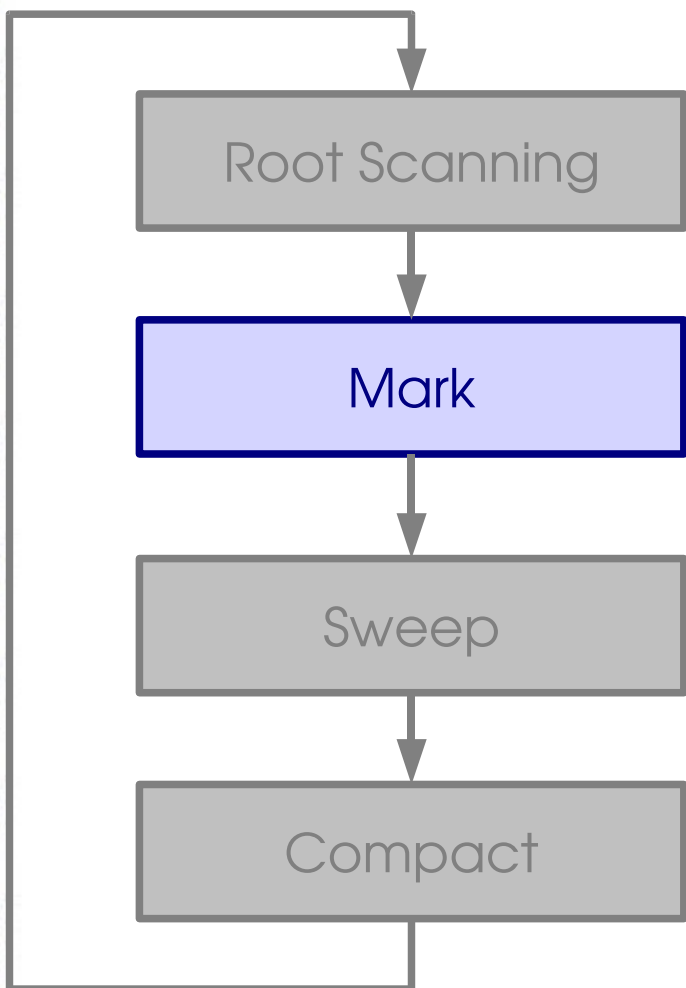
Classic Mark & Sweep GC



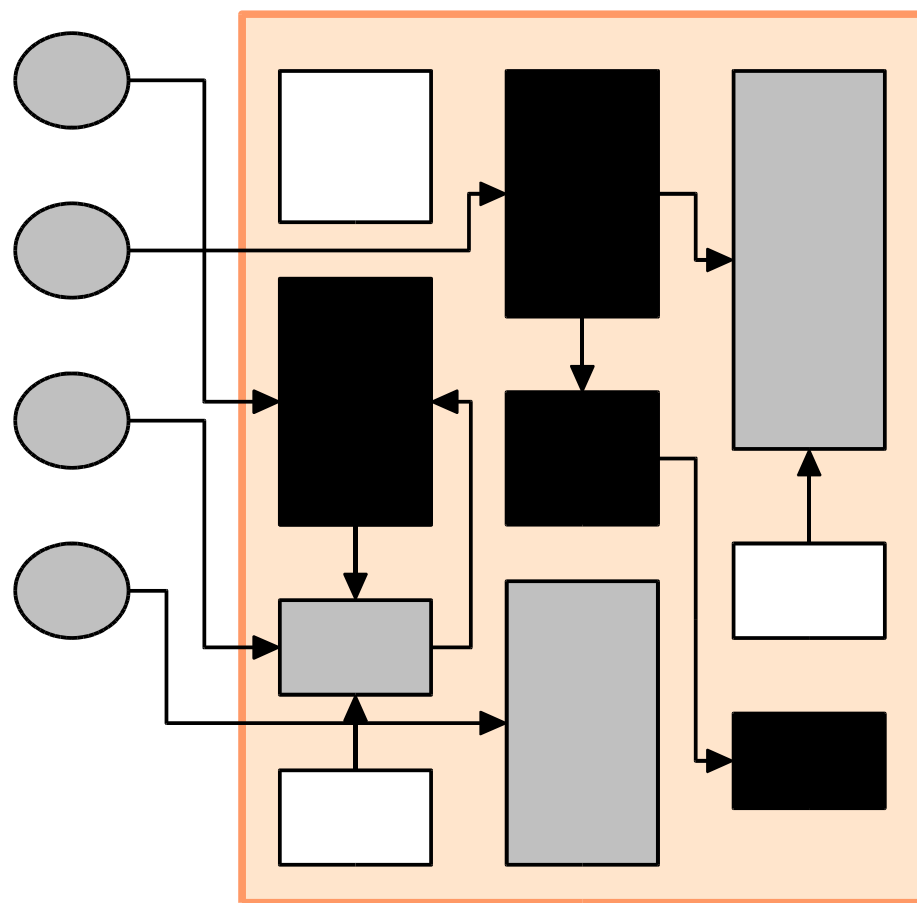
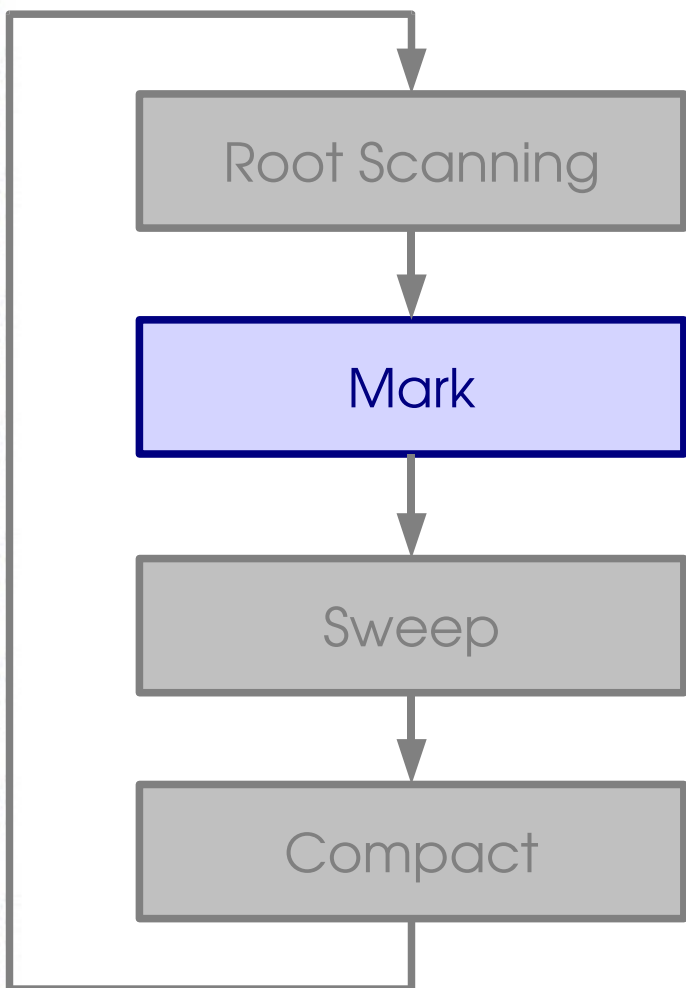
Classic Mark & Sweep GC



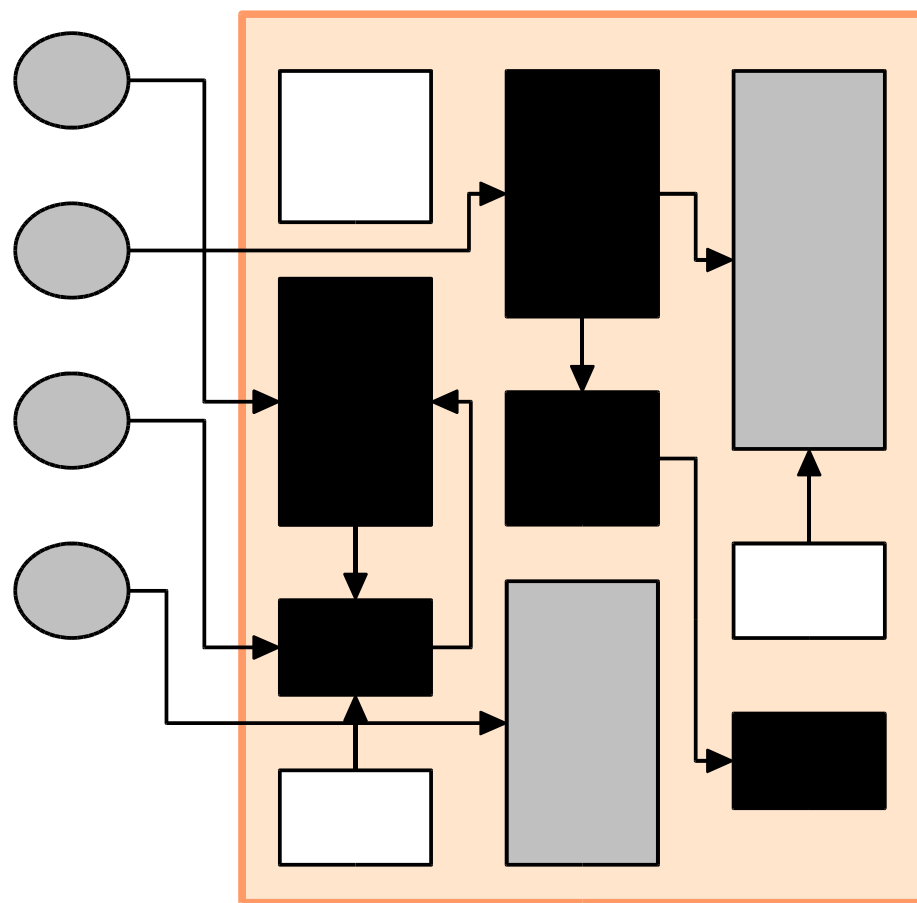
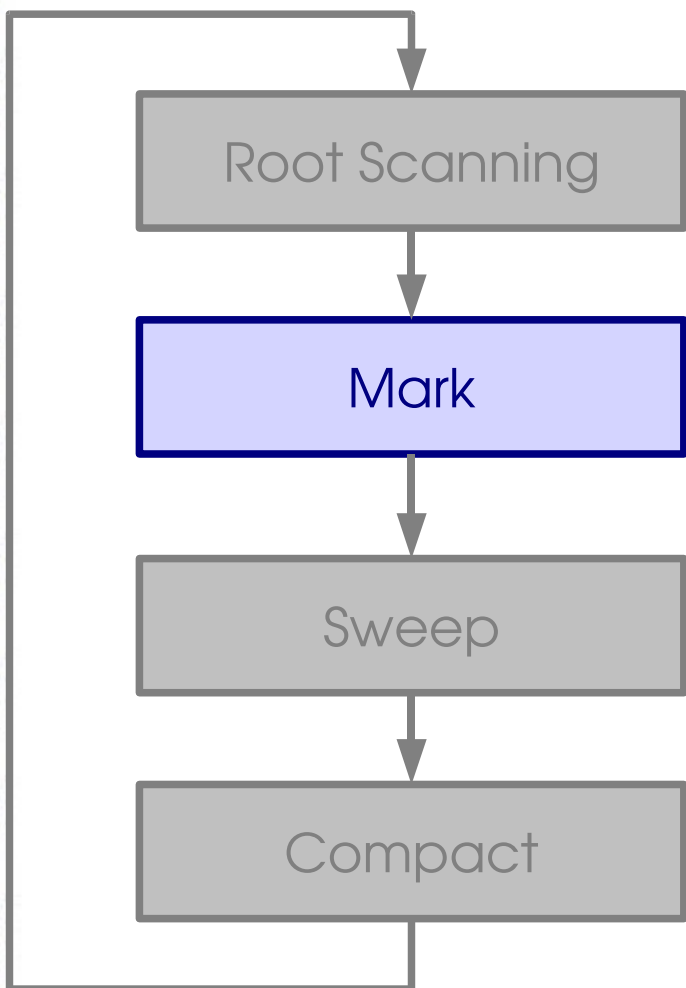
Classic Mark & Sweep GC



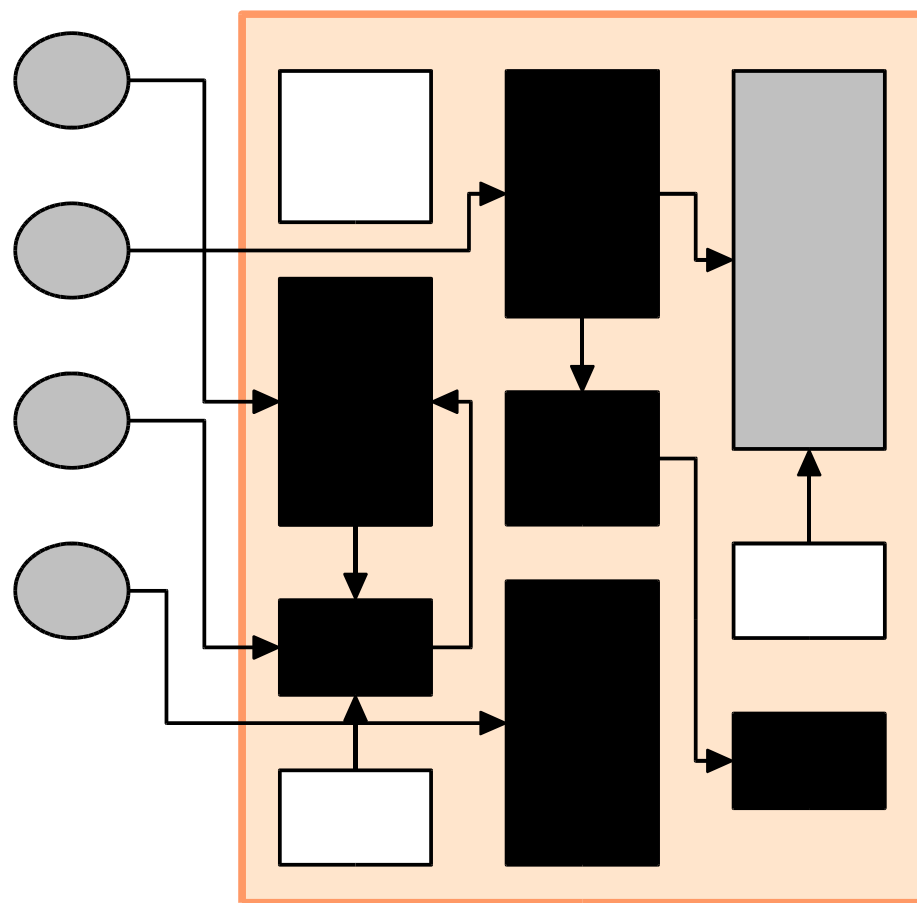
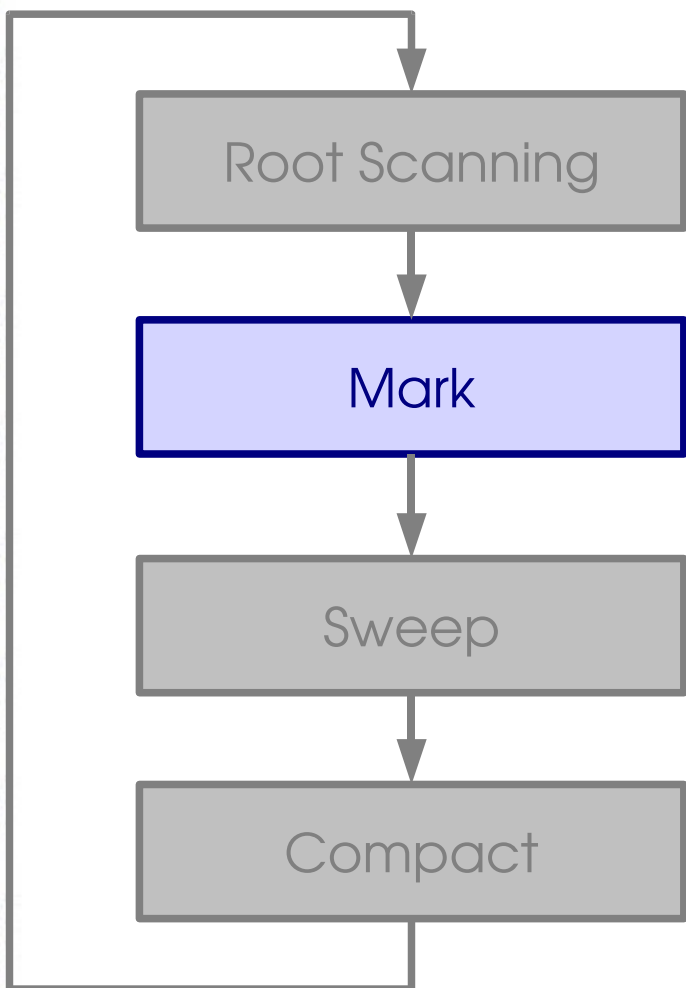
Classic Mark & Sweep GC



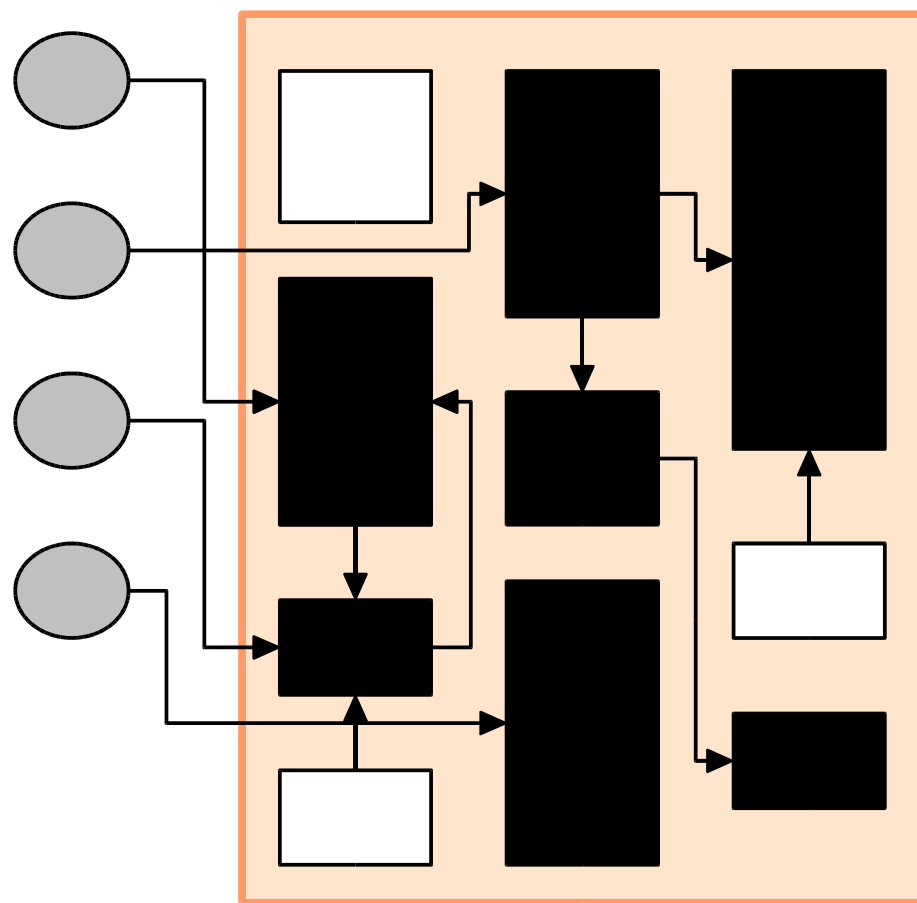
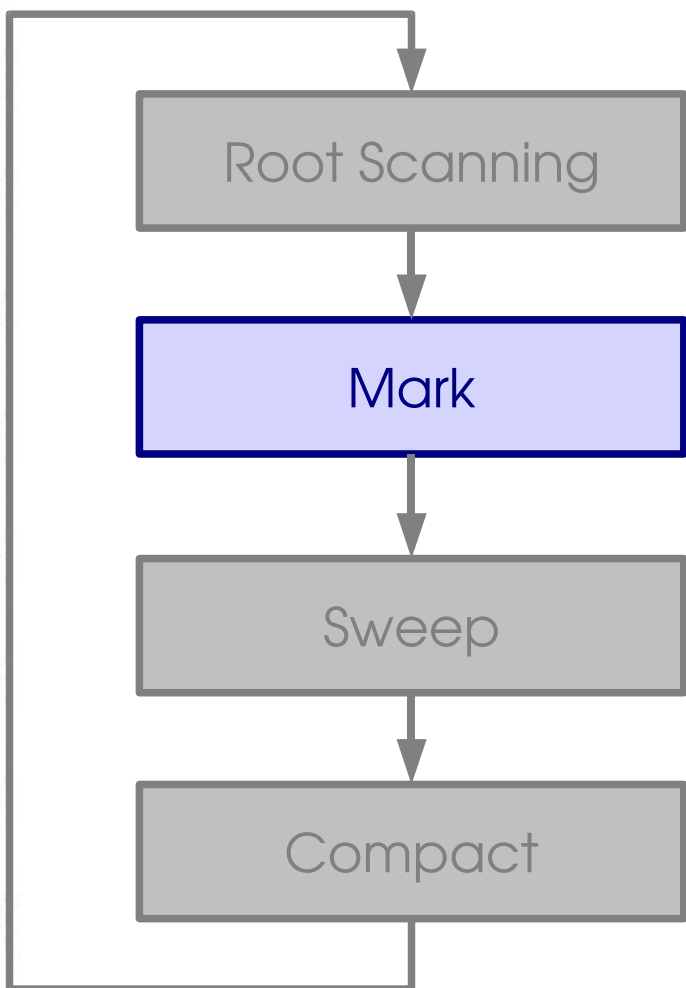
Classic Mark & Sweep GC



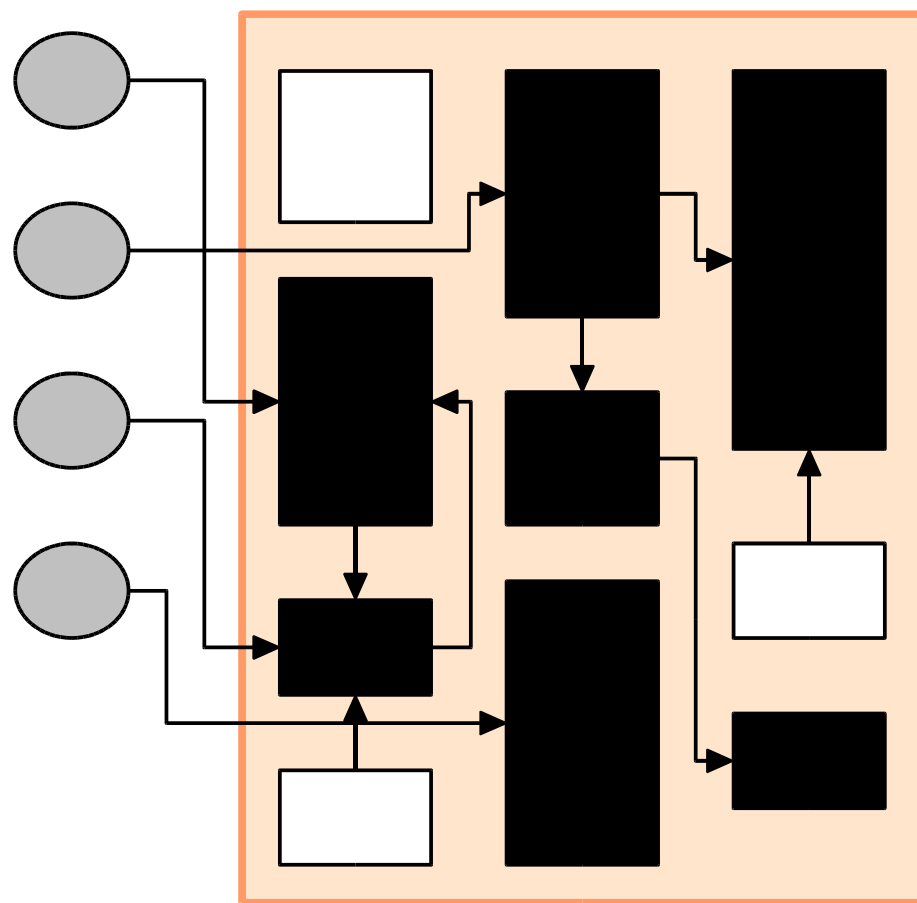
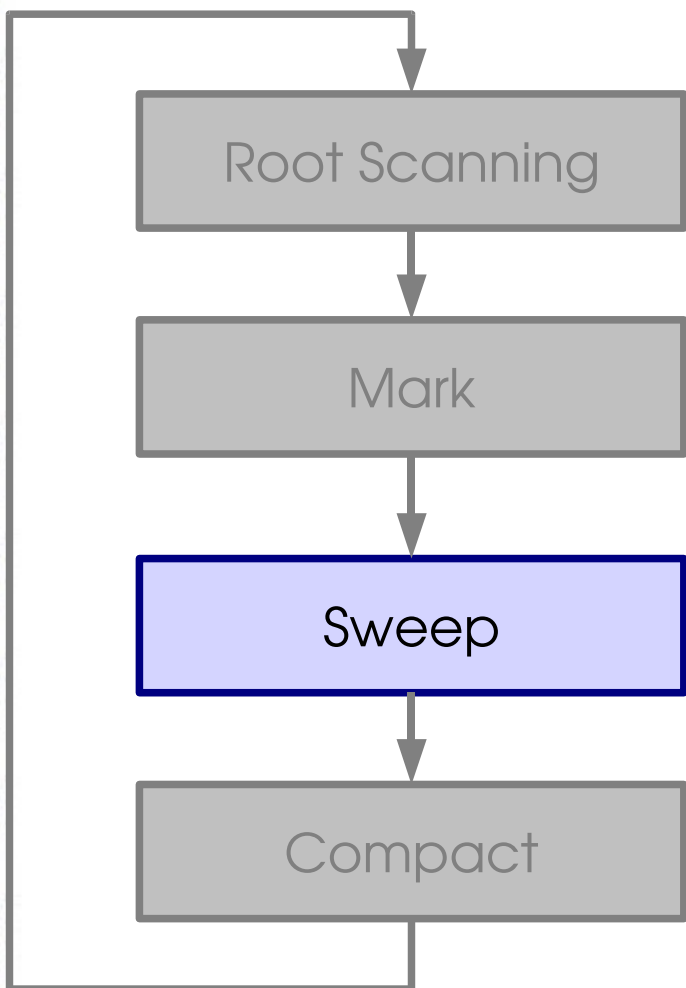
Classic Mark & Sweep GC



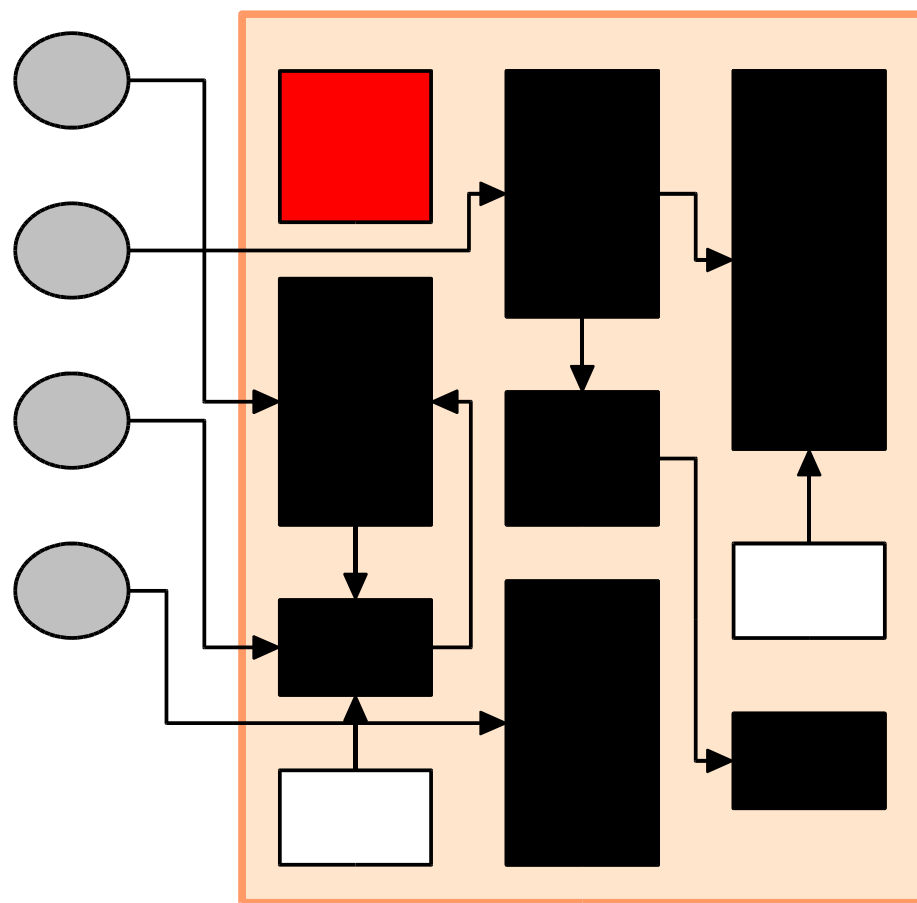
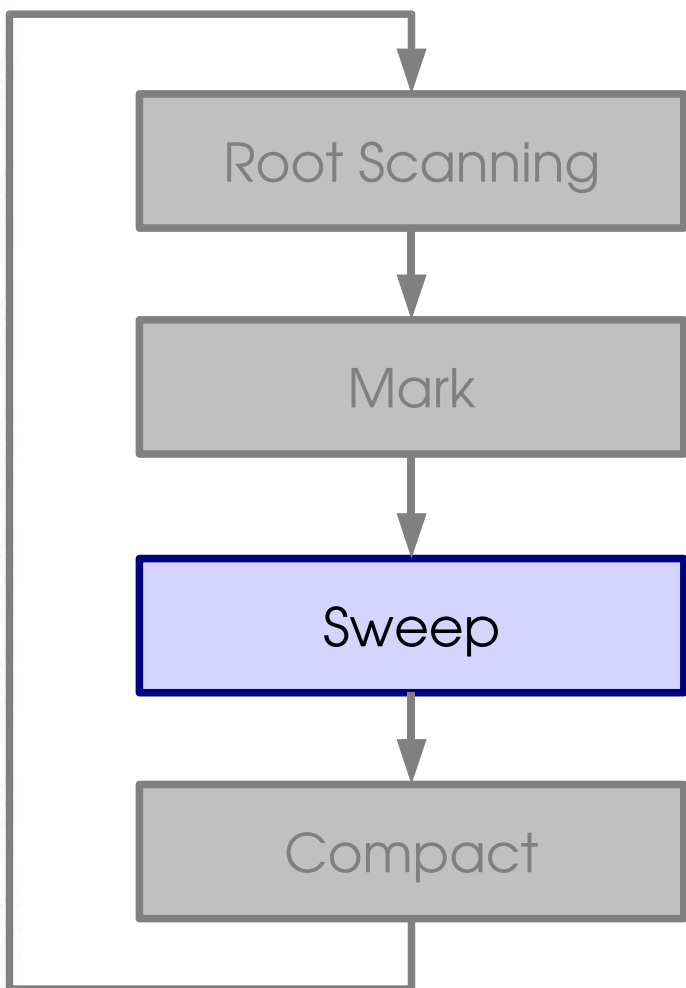
Classic Mark & Sweep GC



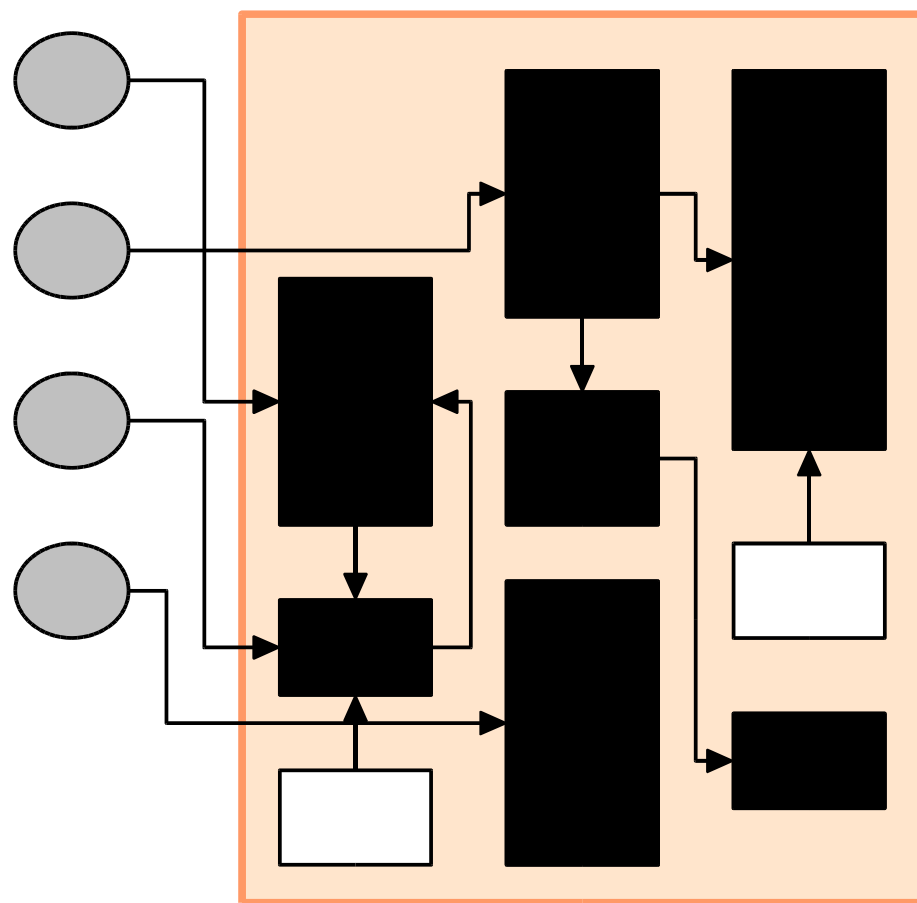
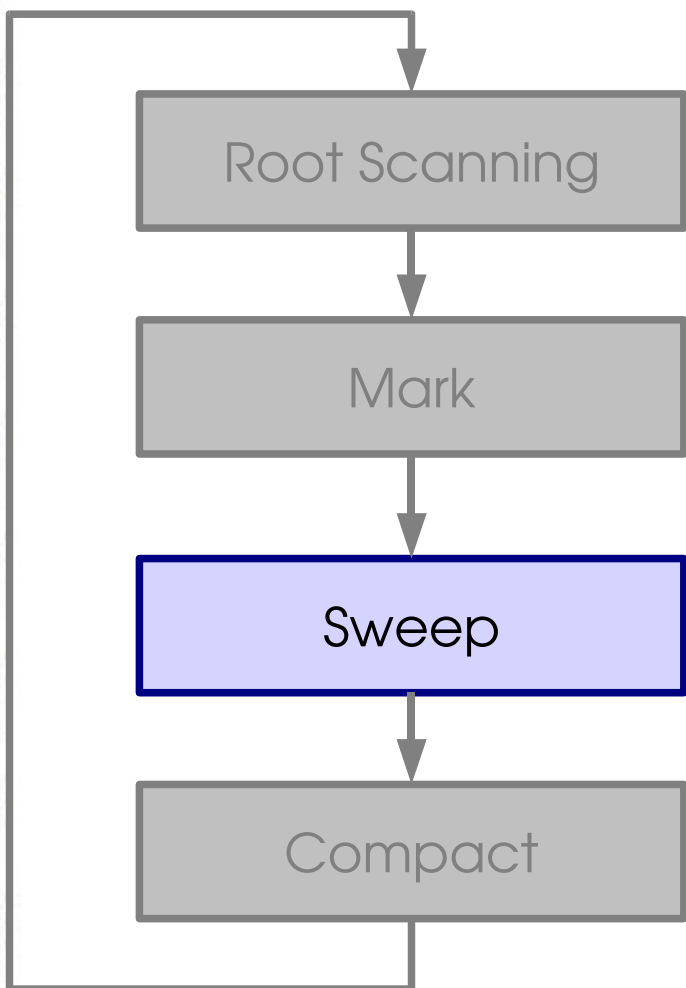
Classic Mark & Sweep GC



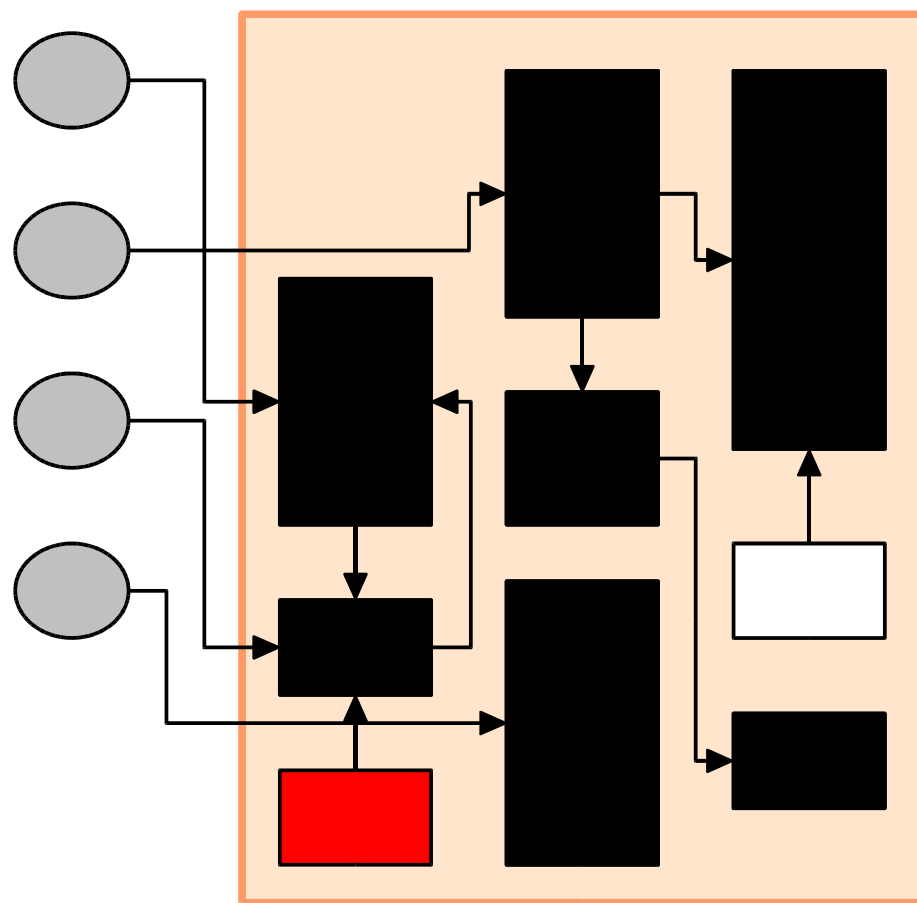
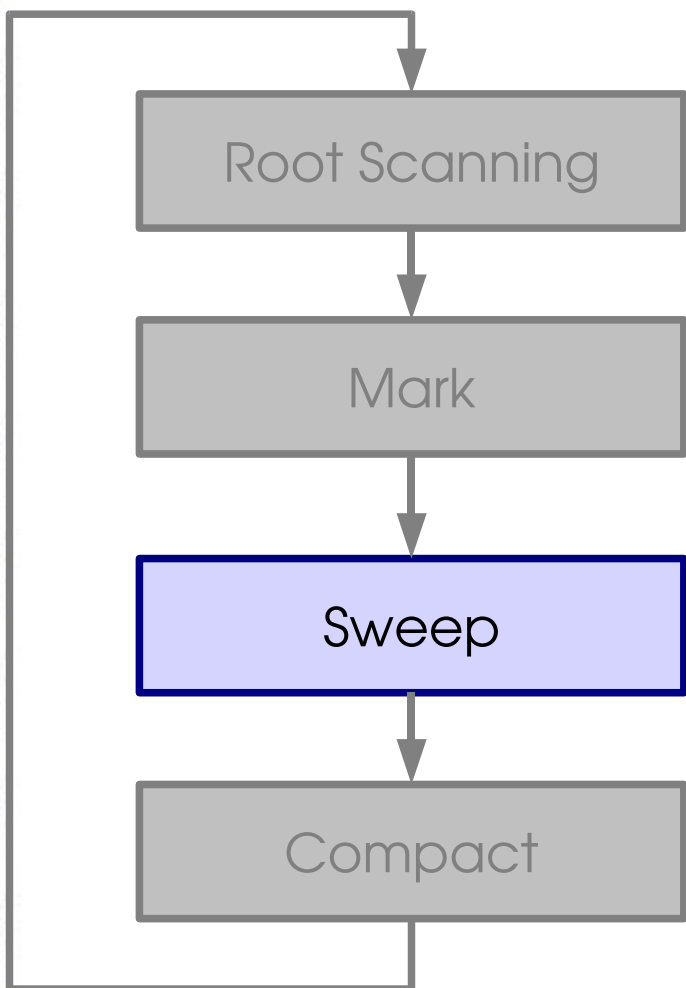
Classic Mark & Sweep GC



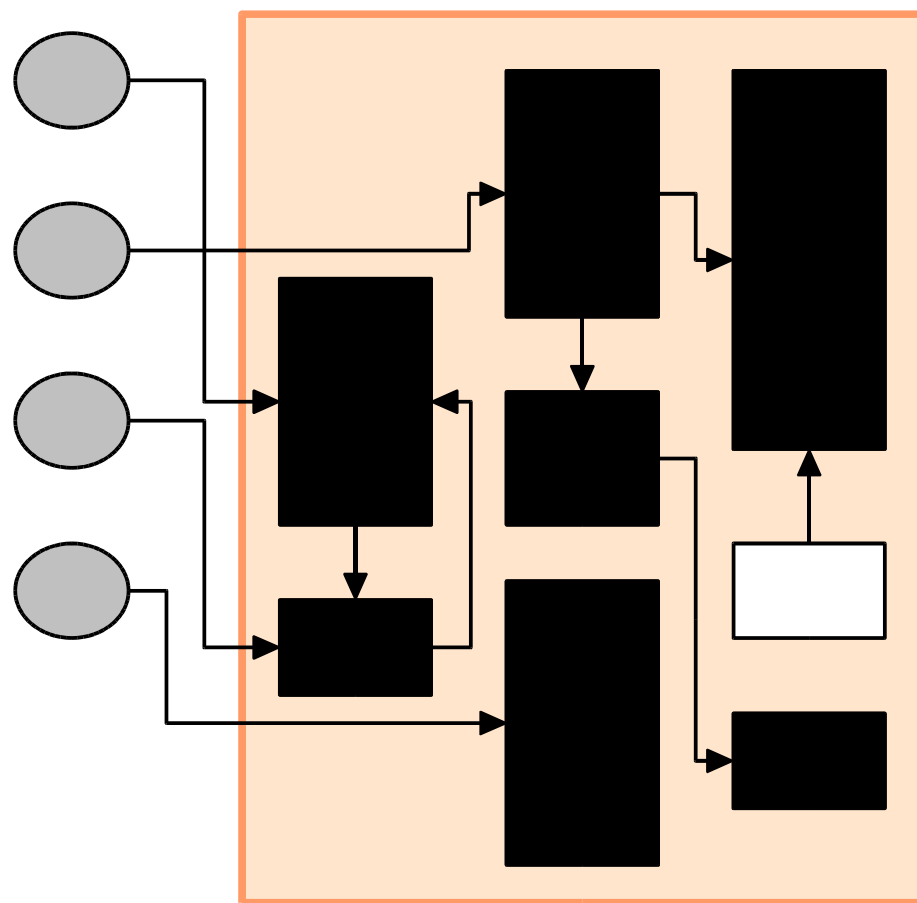
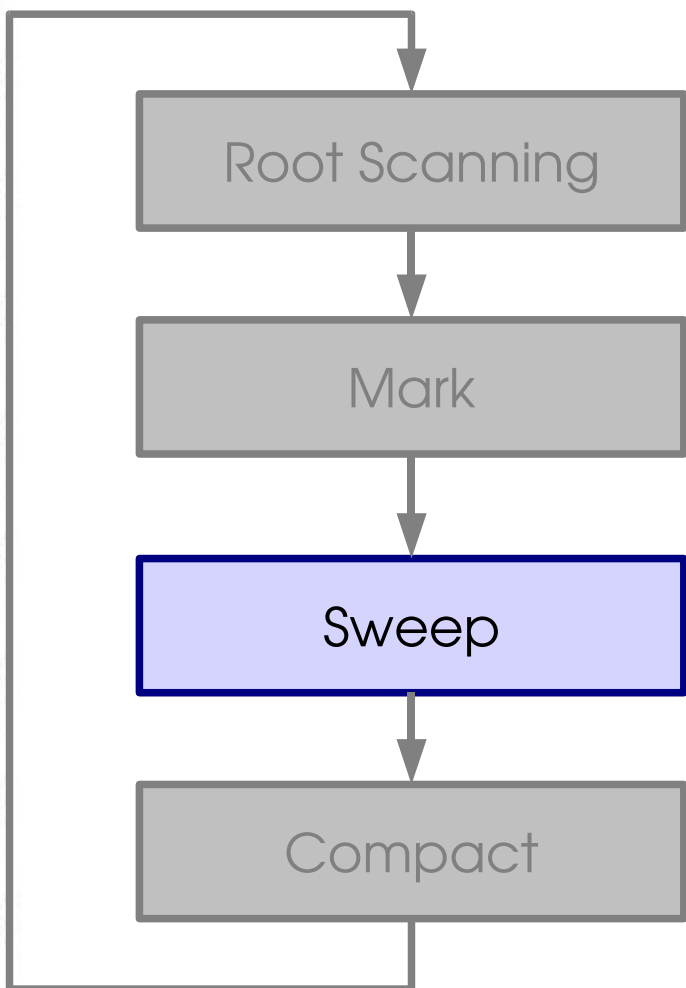
Classic Mark & Sweep GC



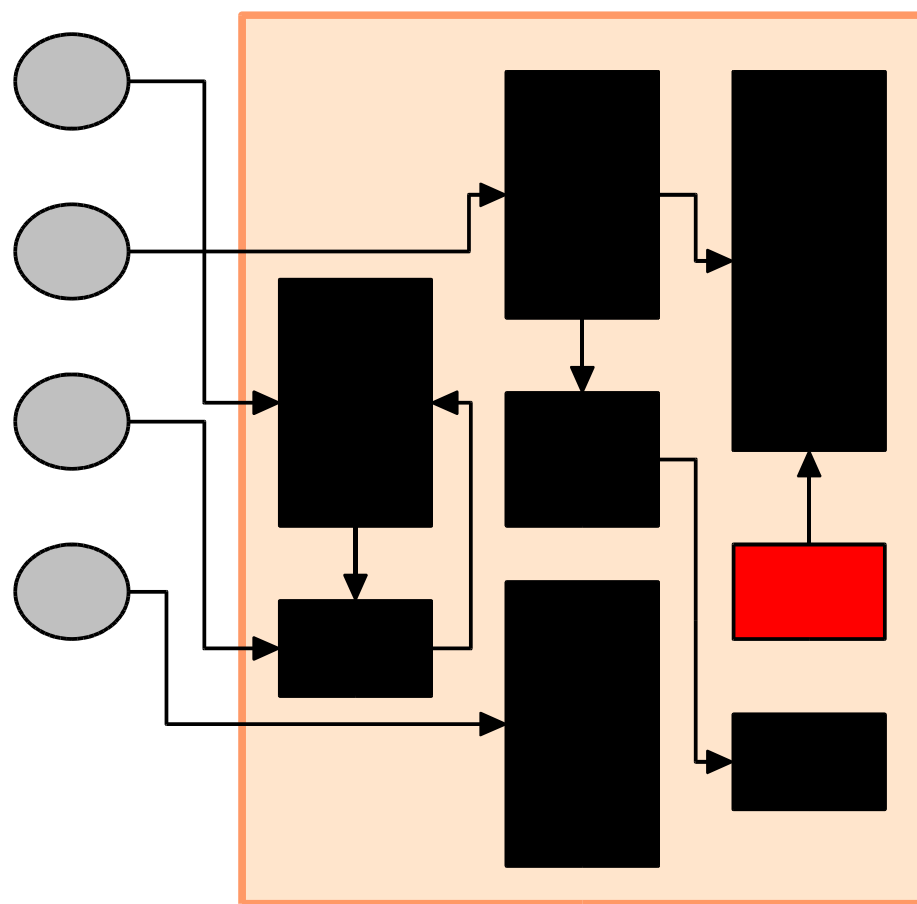
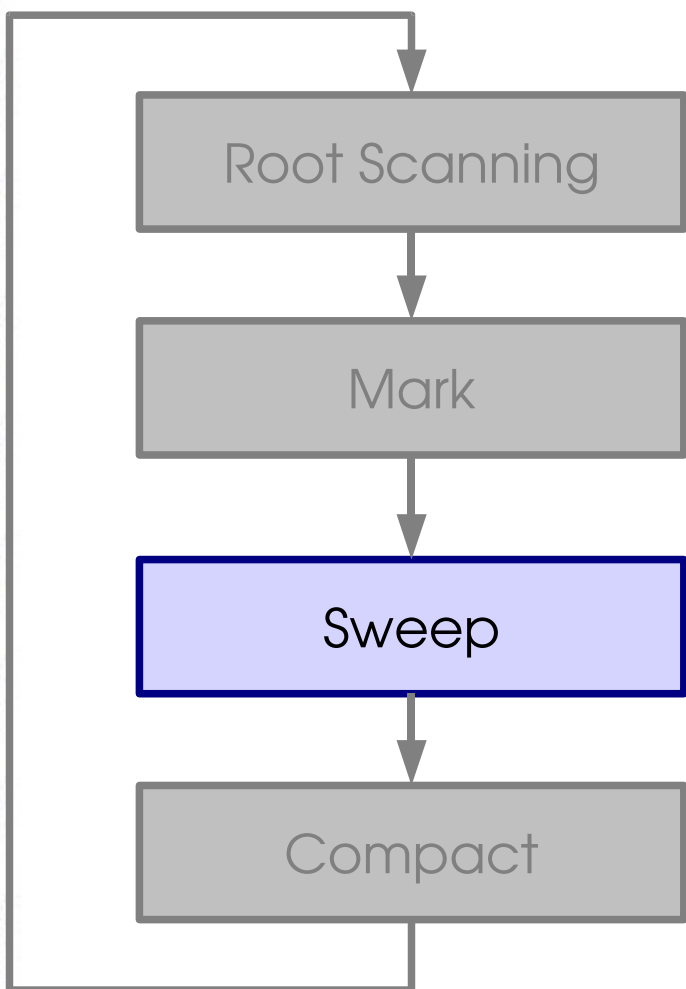
Classic Mark & Sweep GC



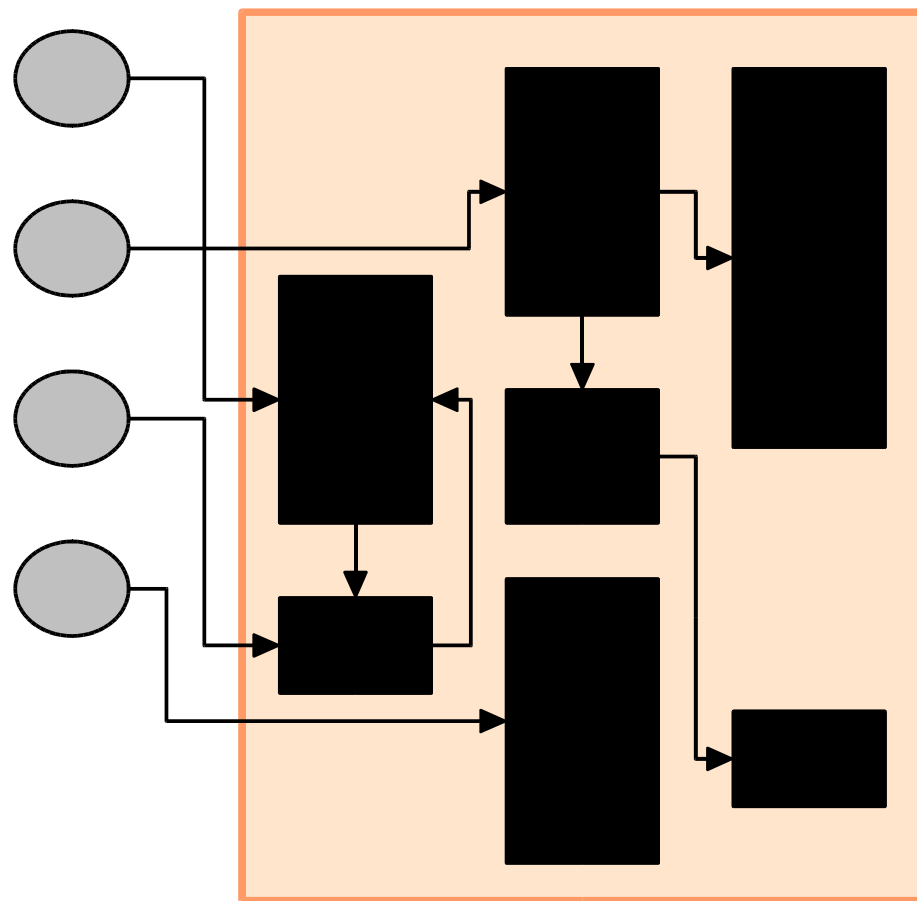
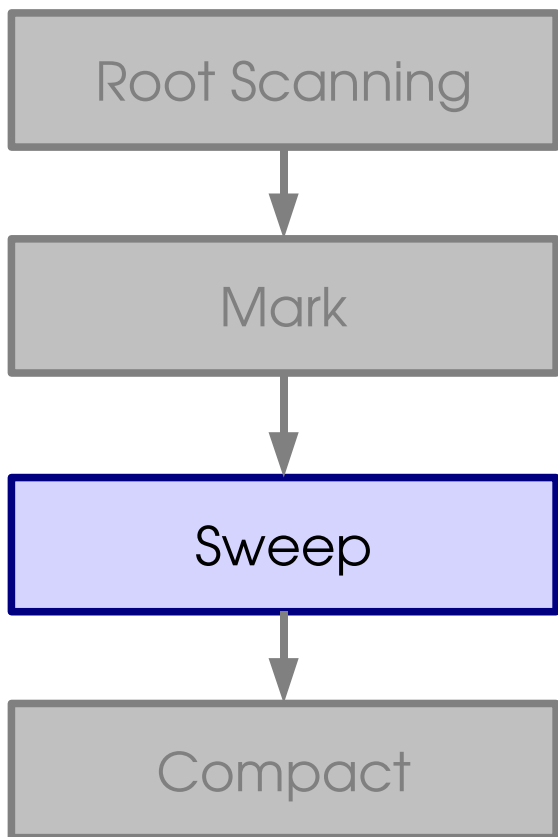
Classic Mark & Sweep GC



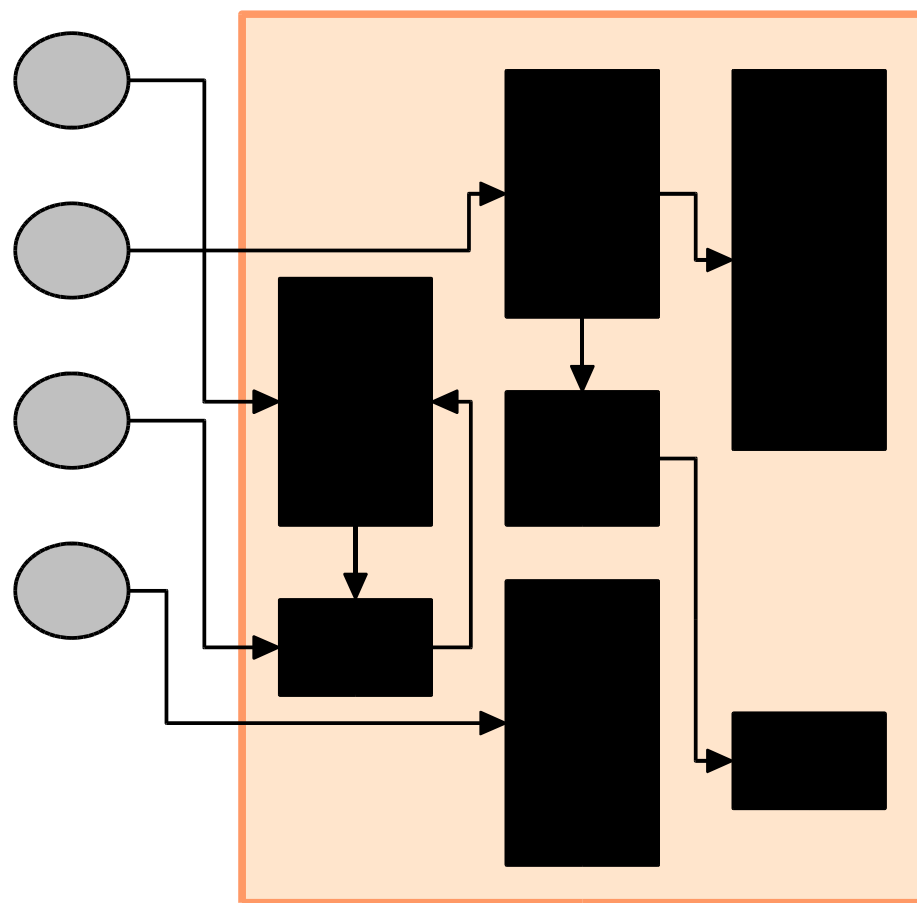
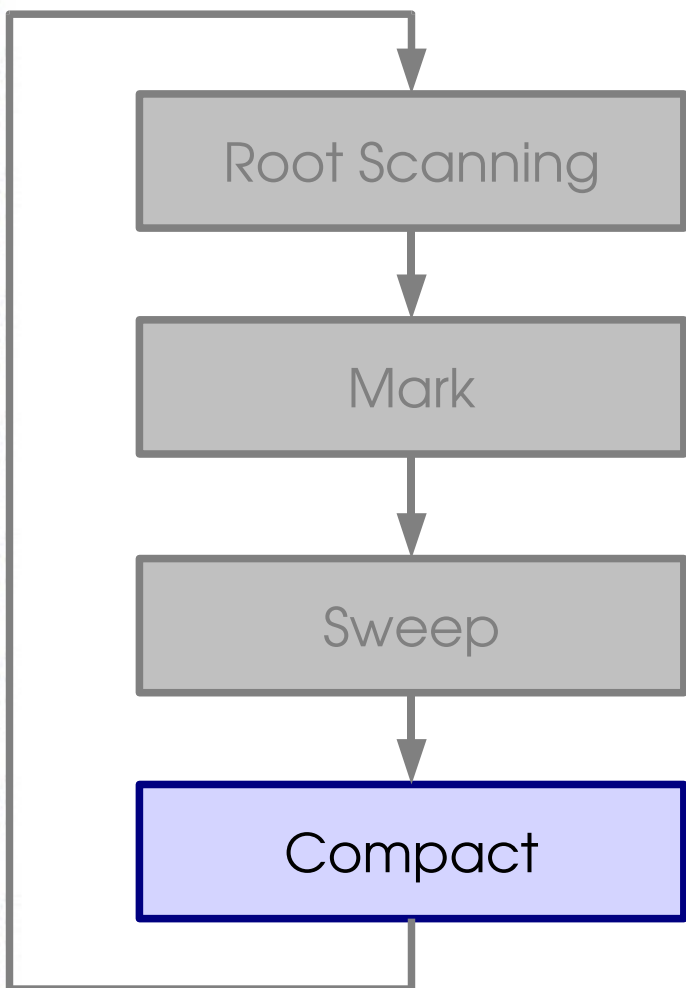
Classic Mark & Sweep GC



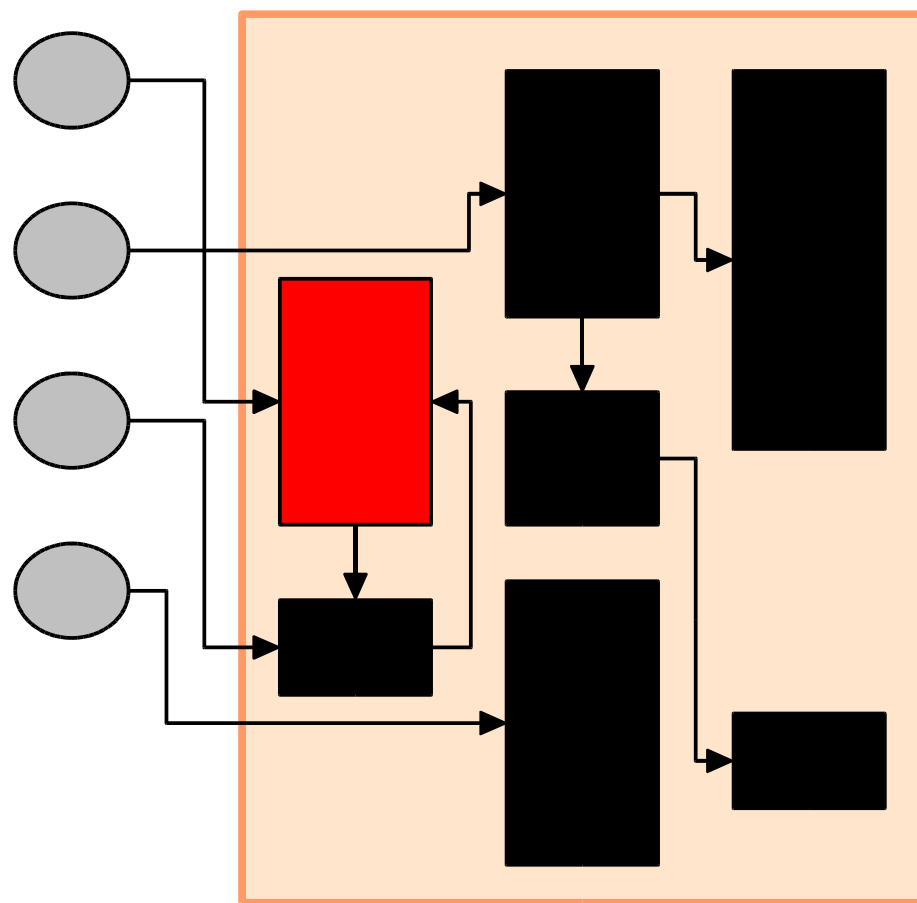
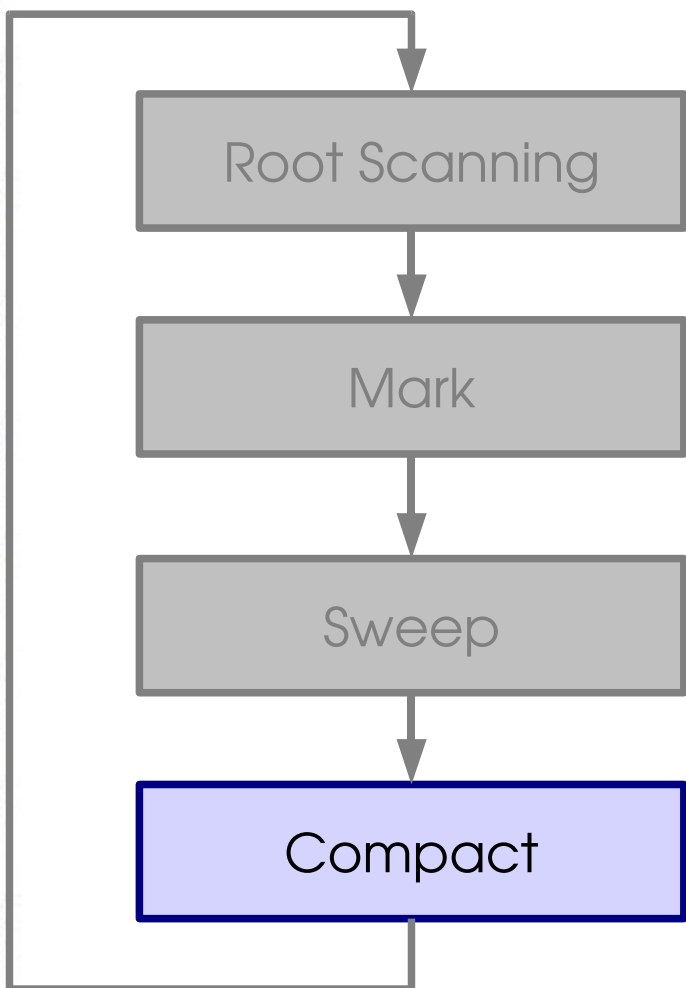
Classic Mark & Sweep GC



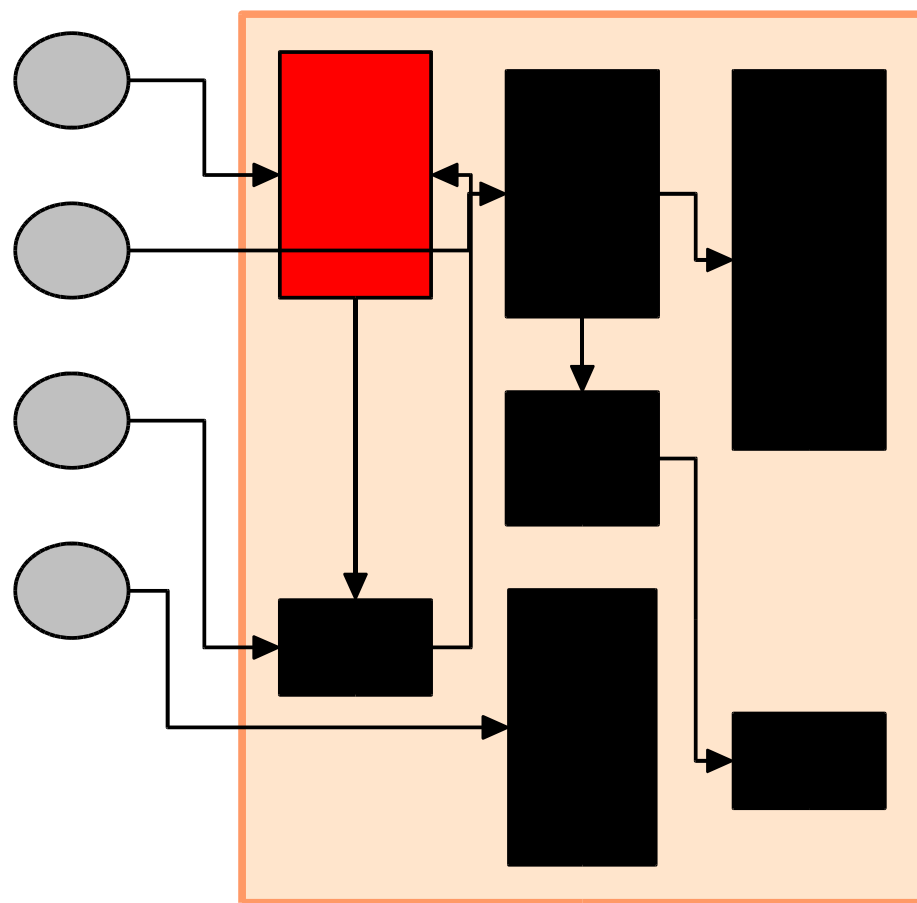
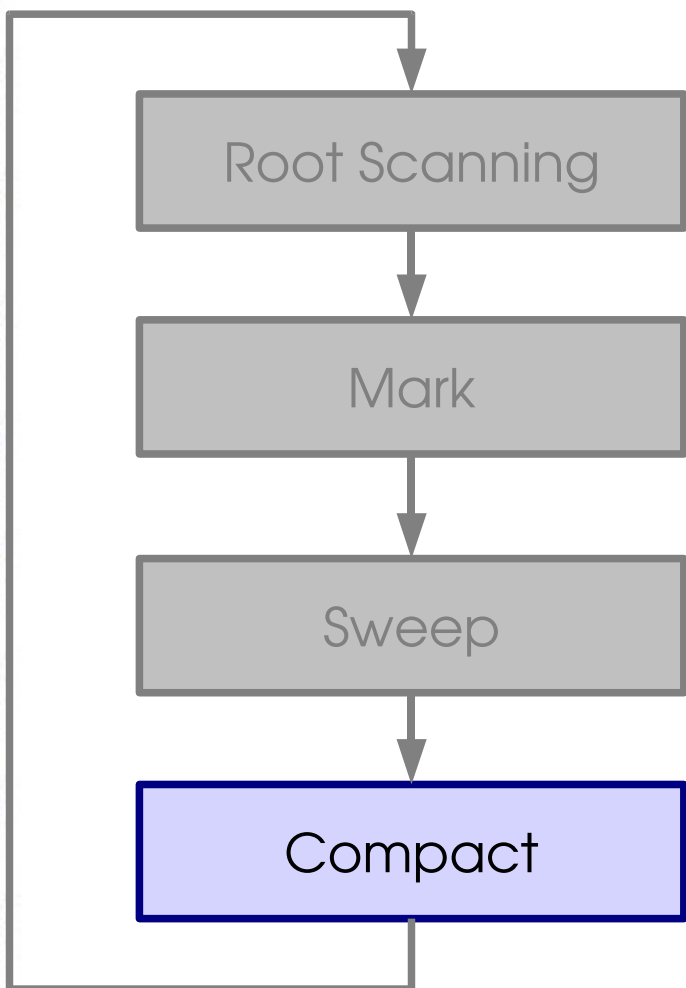
Classic Mark & Sweep GC



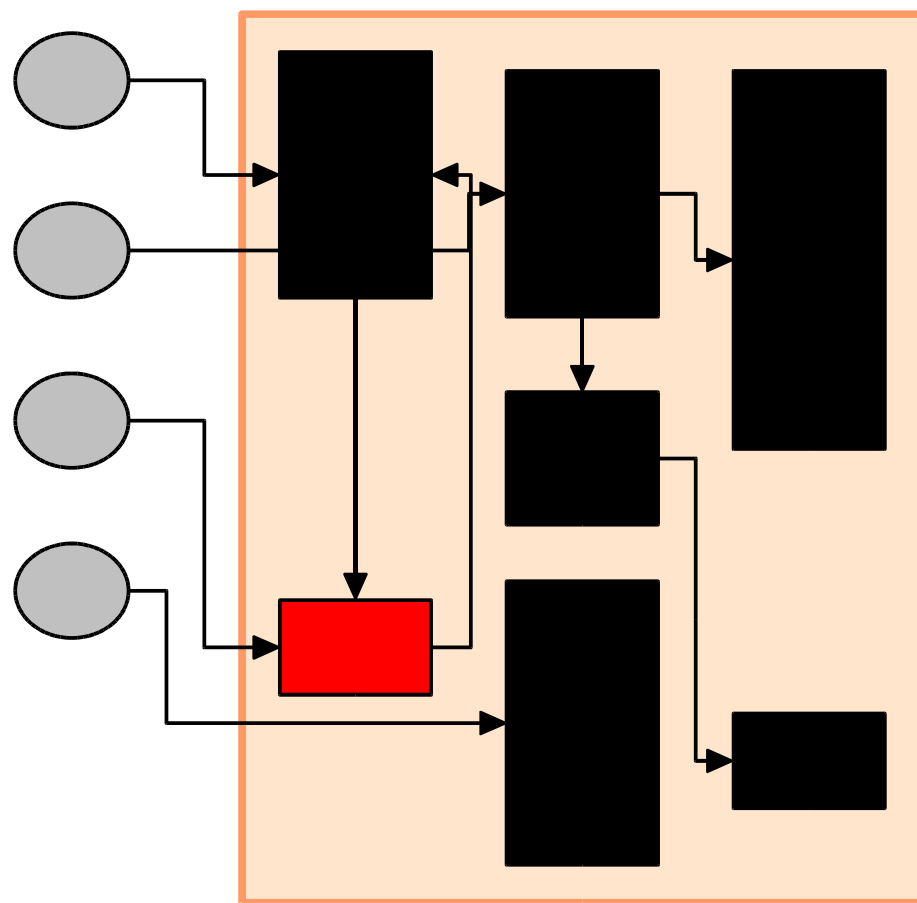
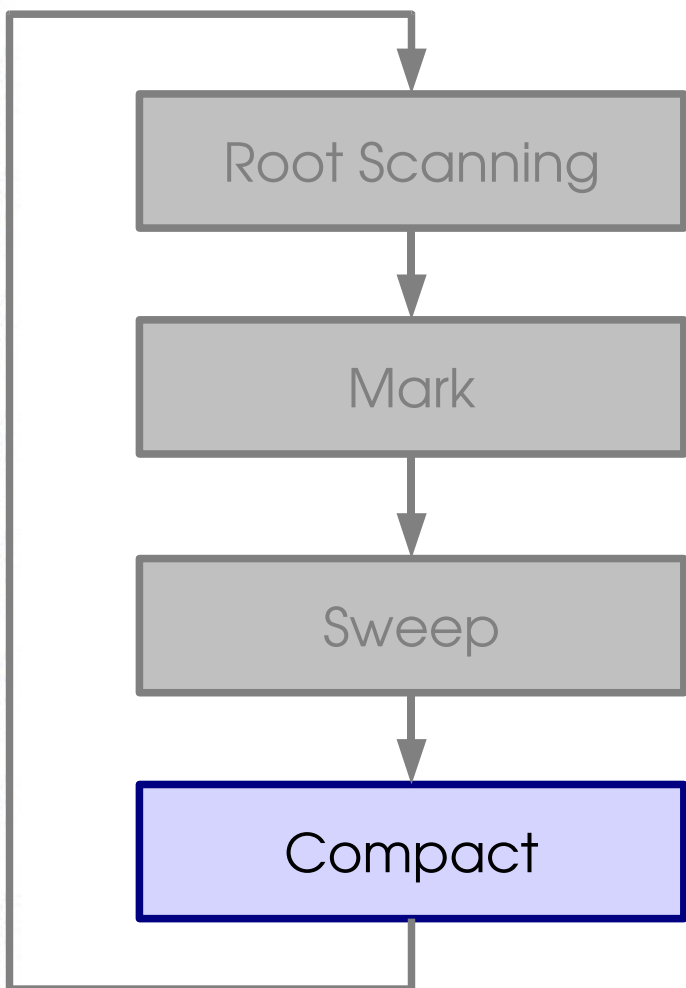
Classic Mark & Sweep GC



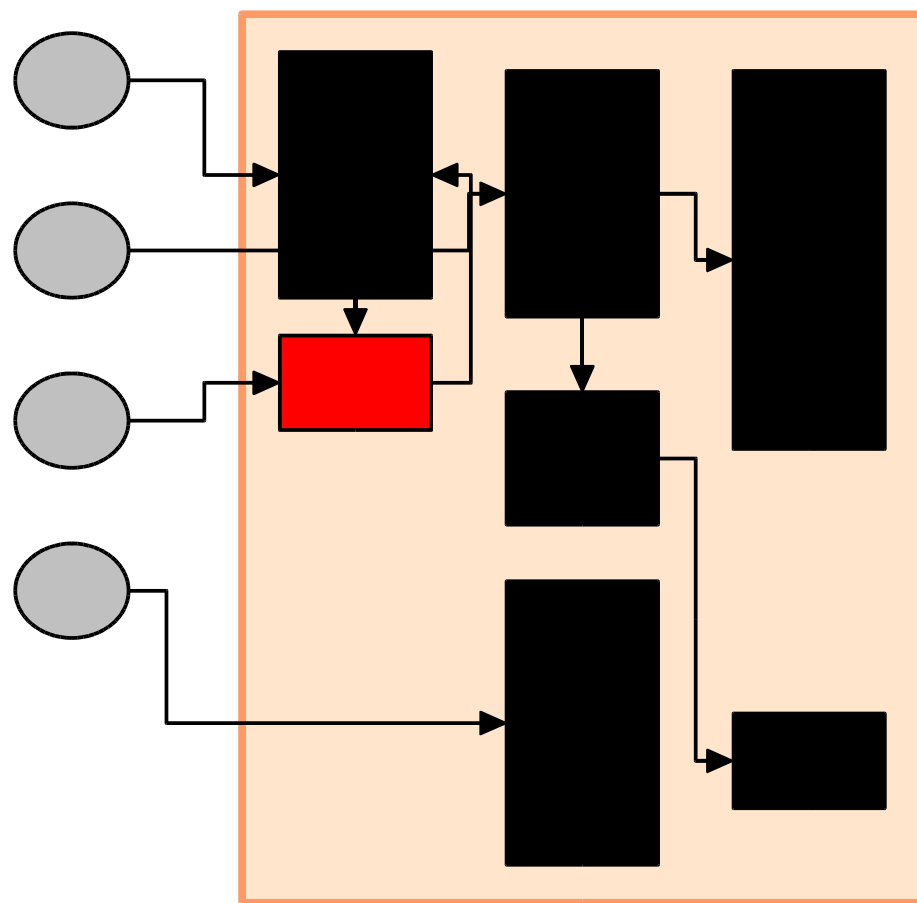
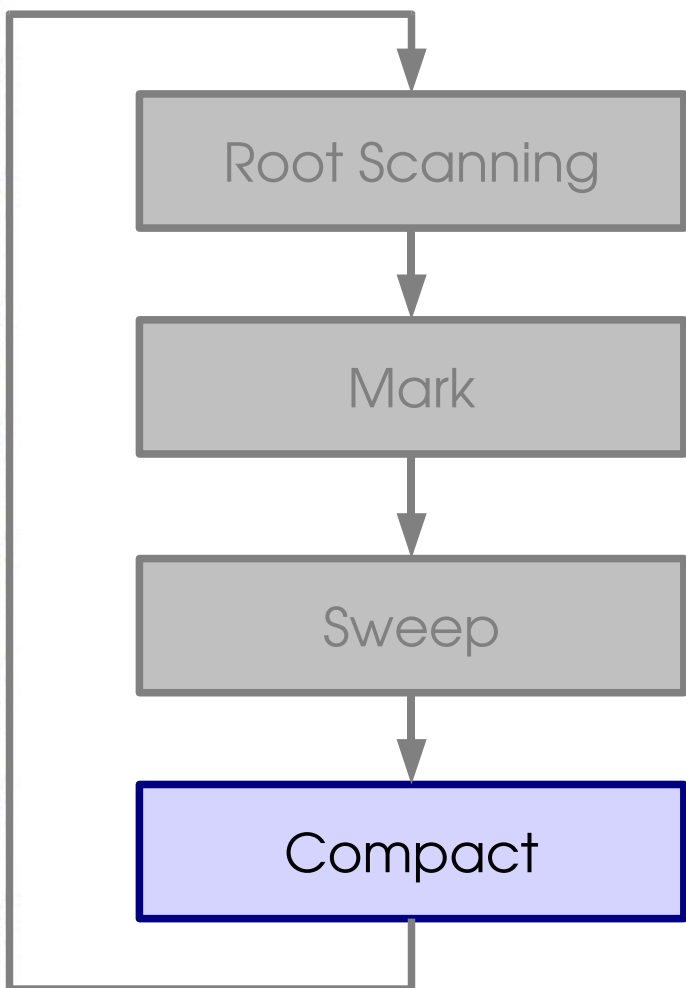
Classic Mark & Sweep GC



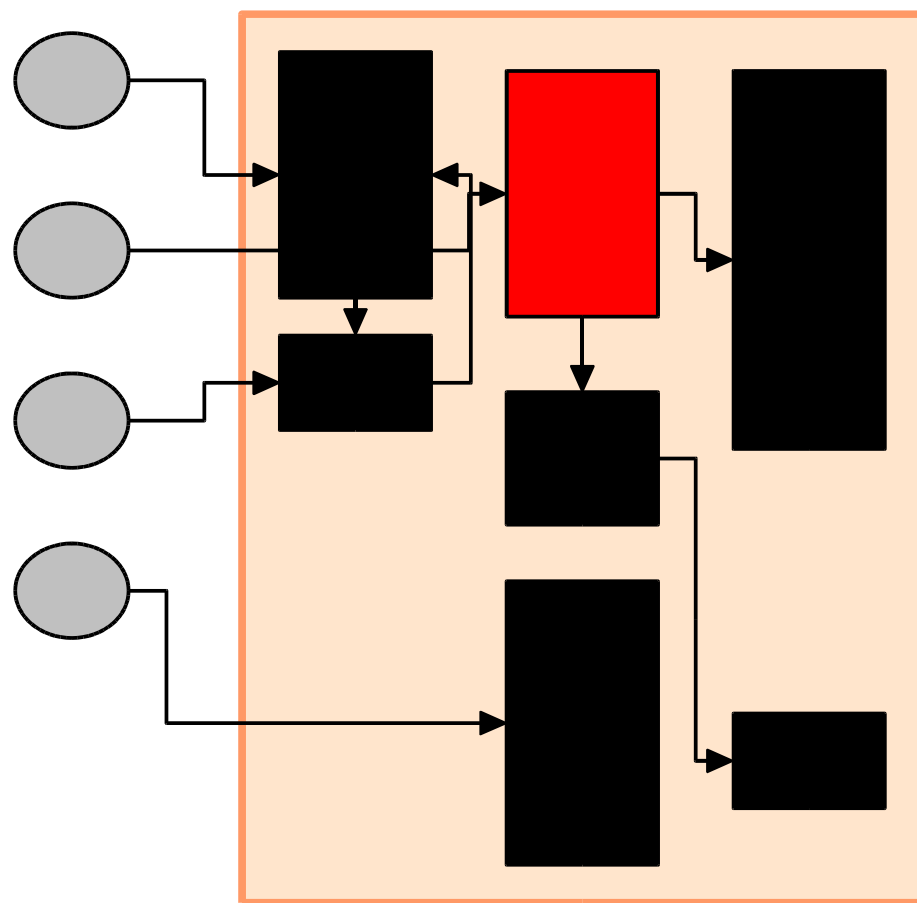
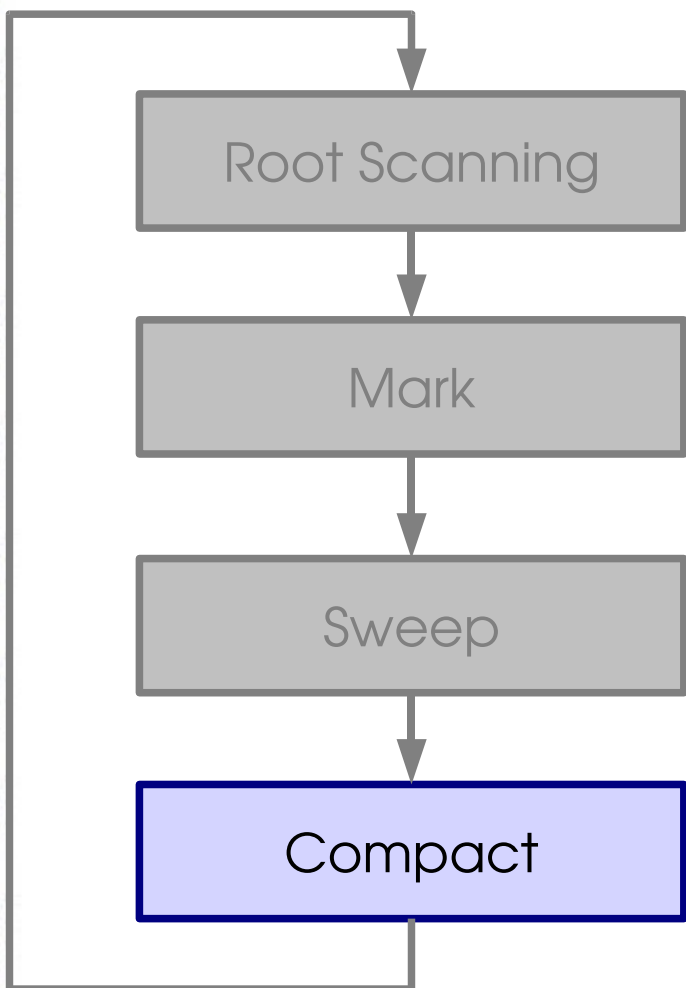
Classic Mark & Sweep GC



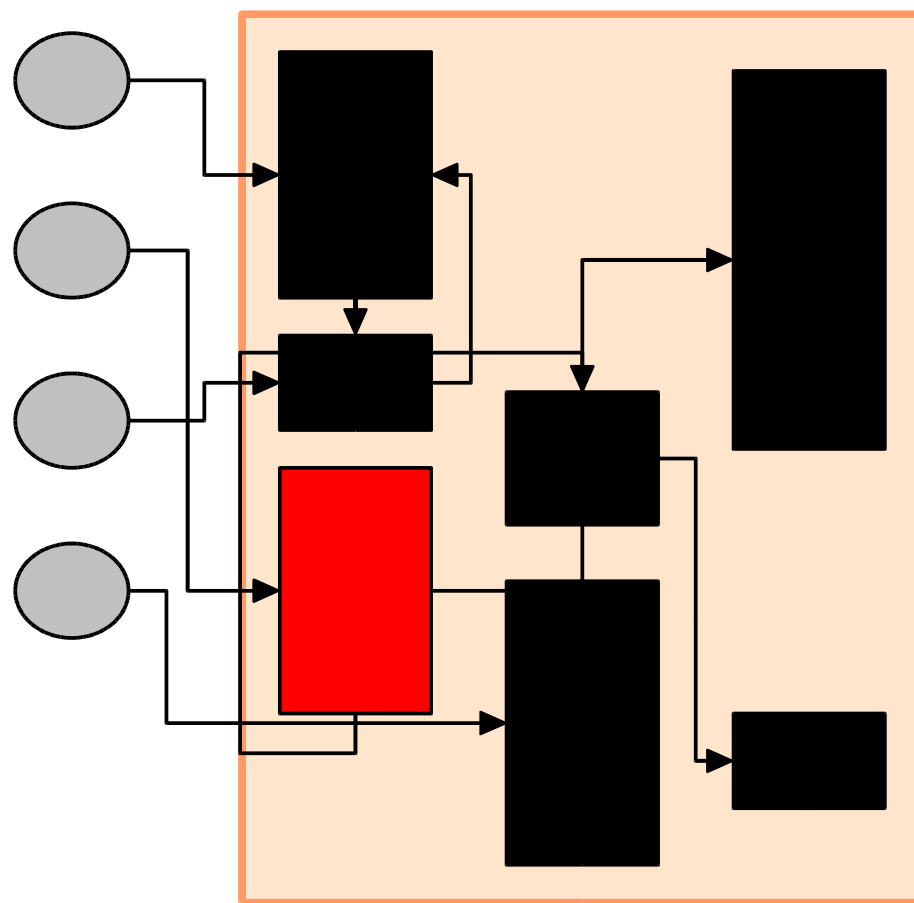
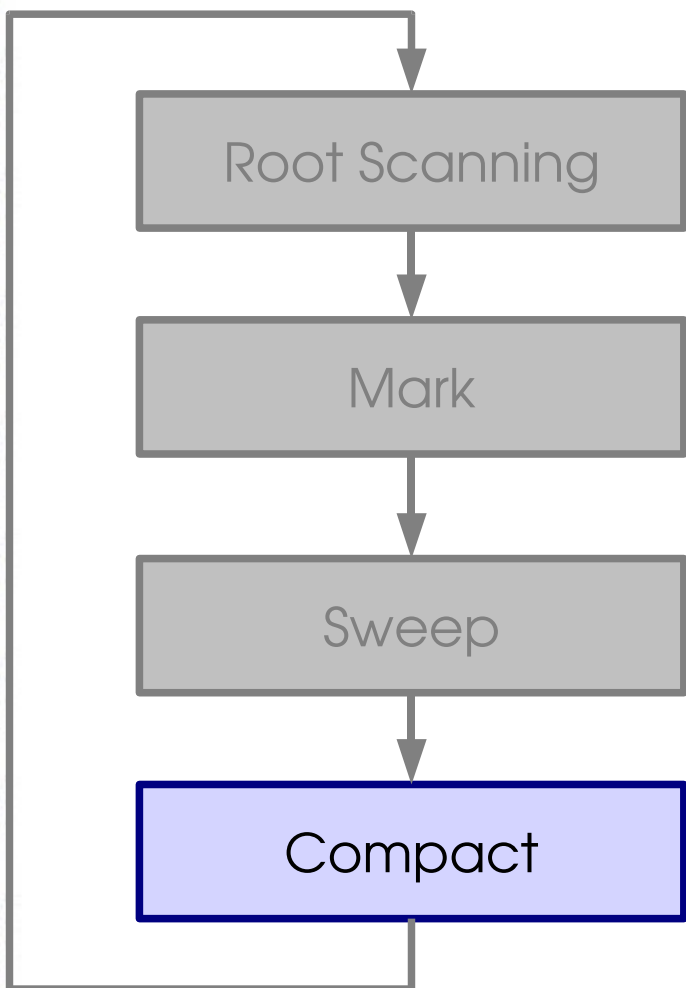
Classic Mark & Sweep GC



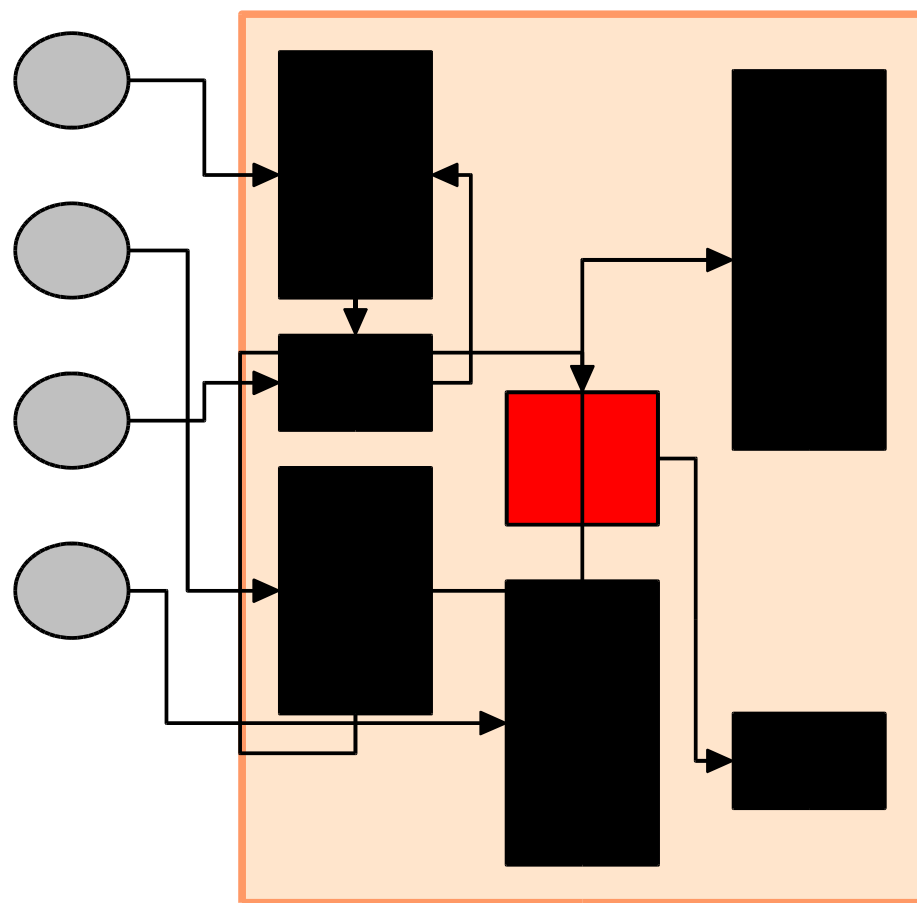
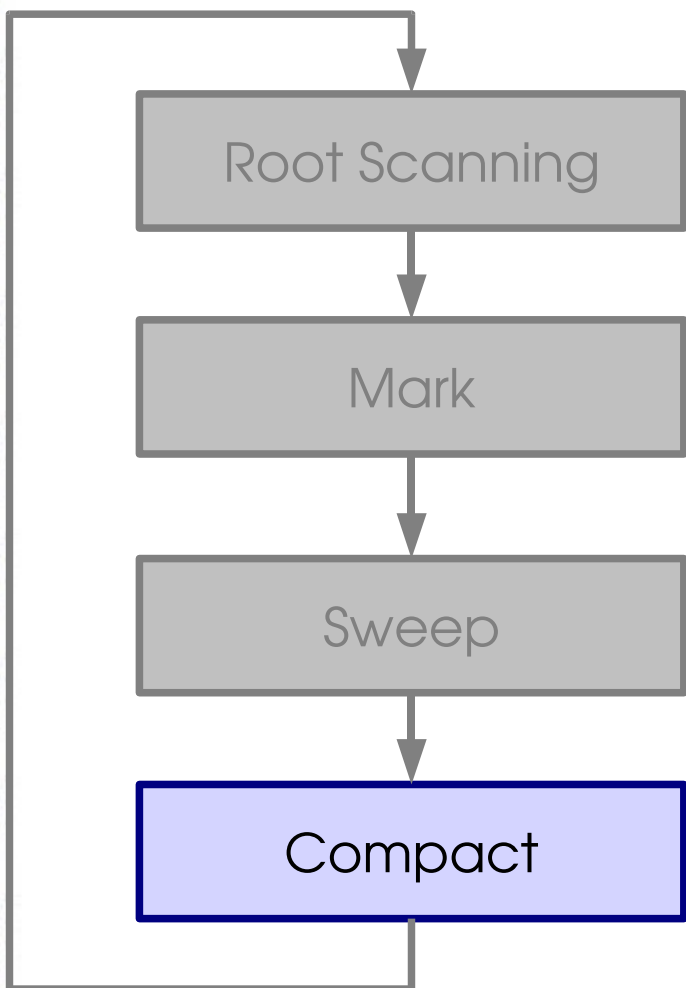
Classic Mark & Sweep GC



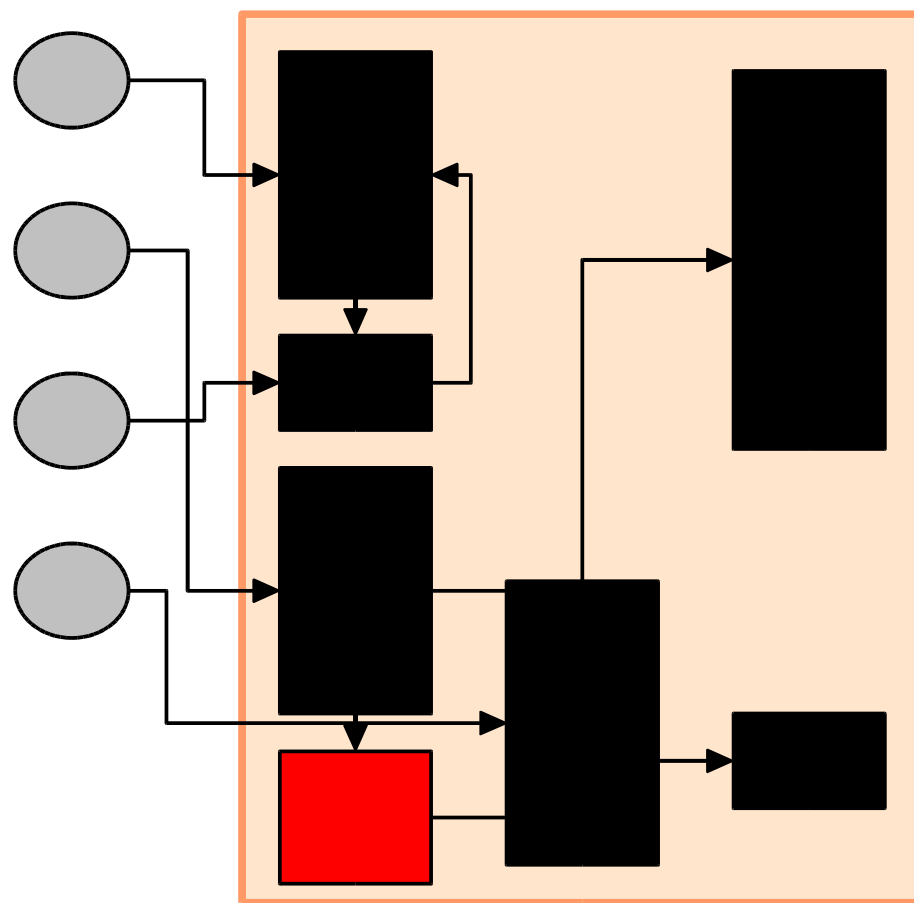
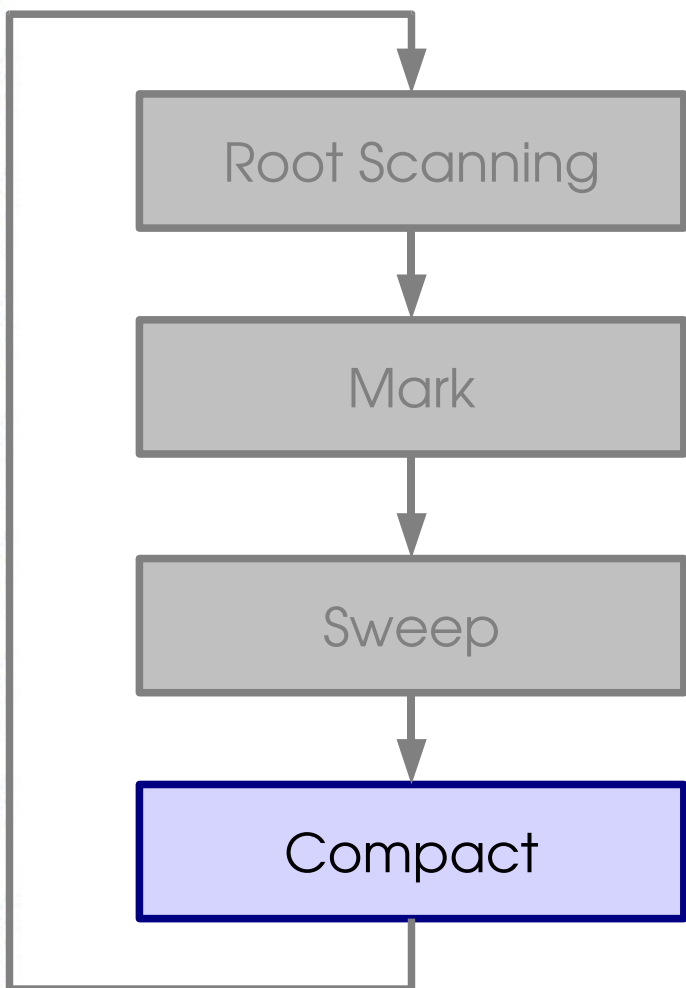
Classic Mark & Sweep GC



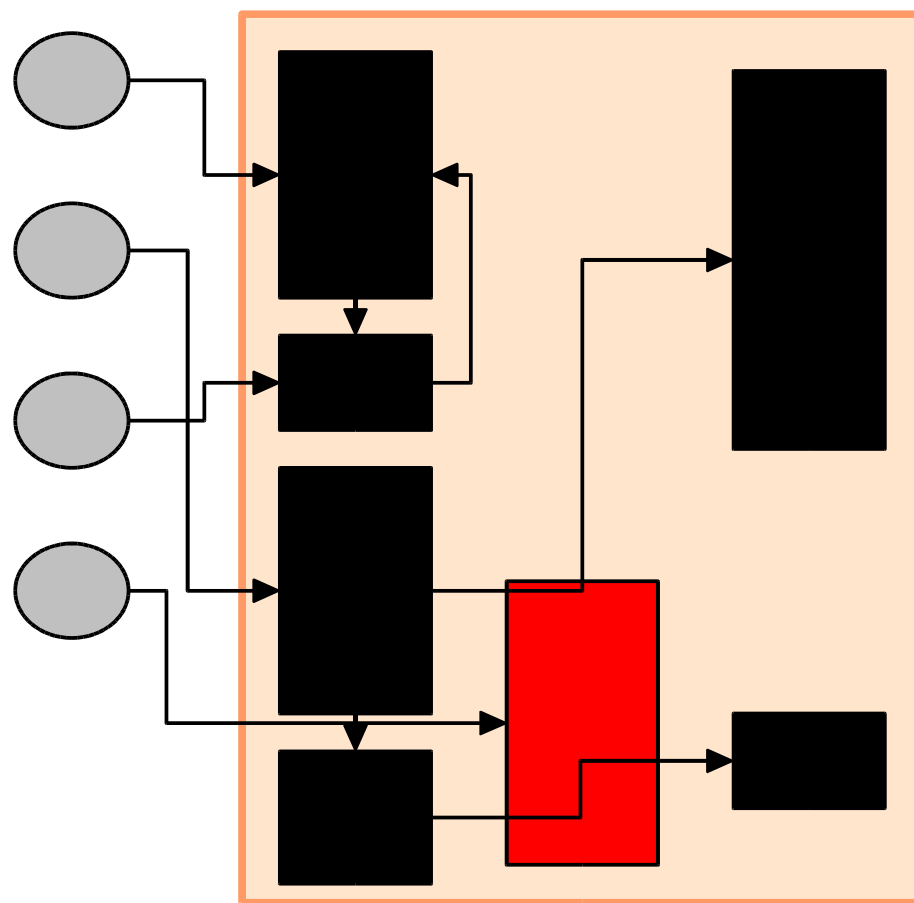
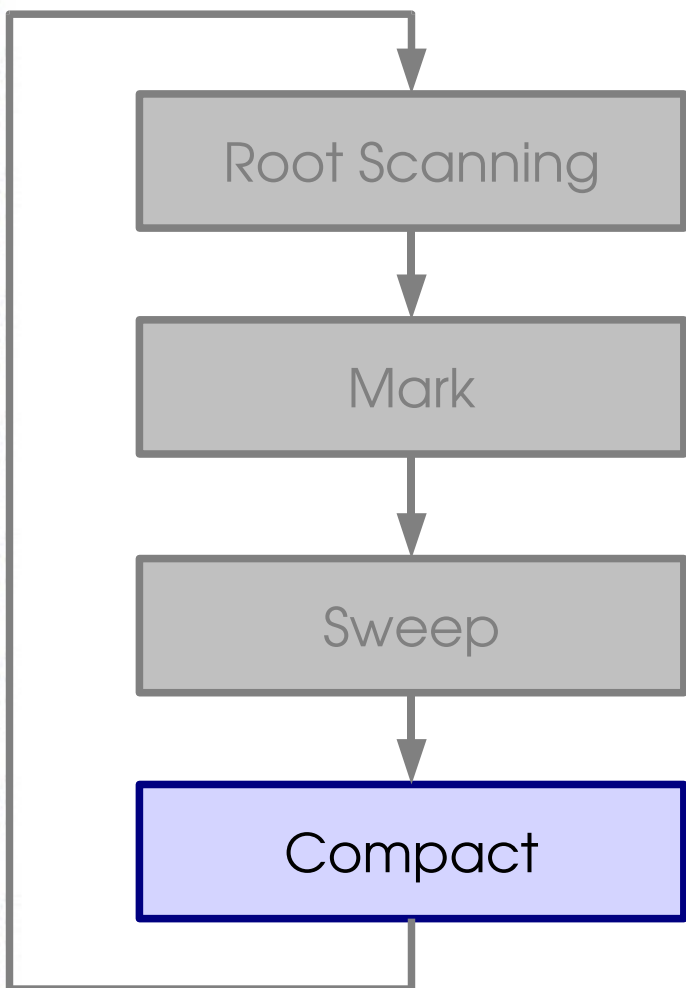
Classic Mark & Sweep GC



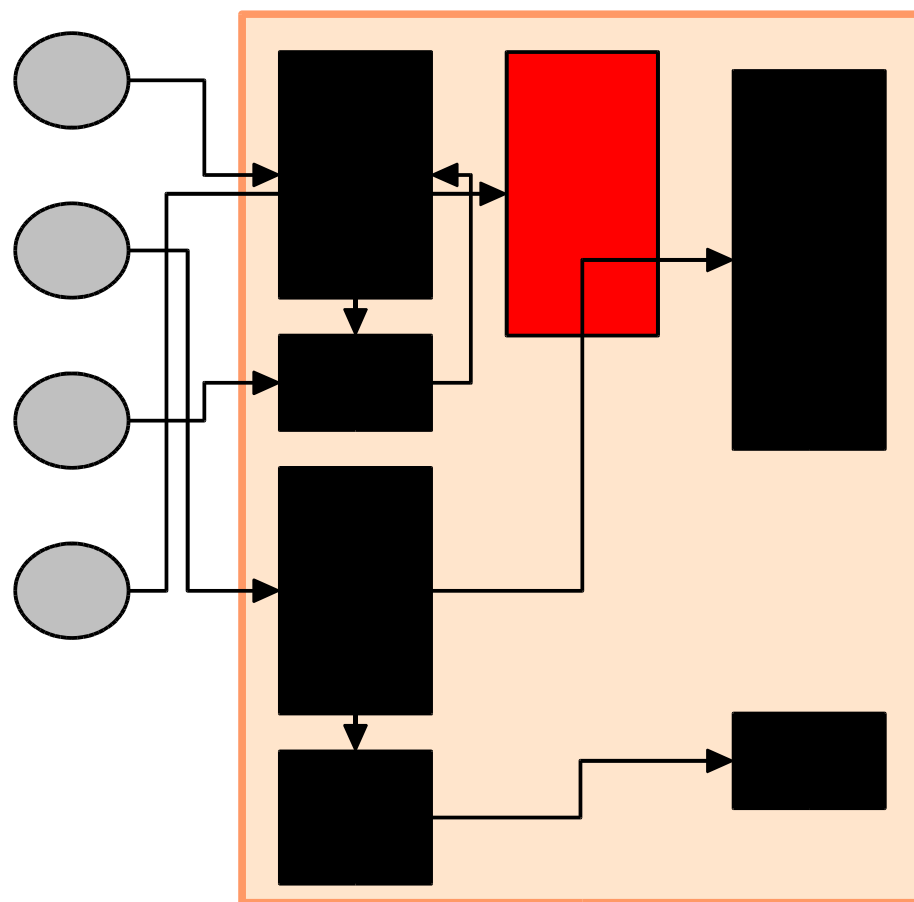
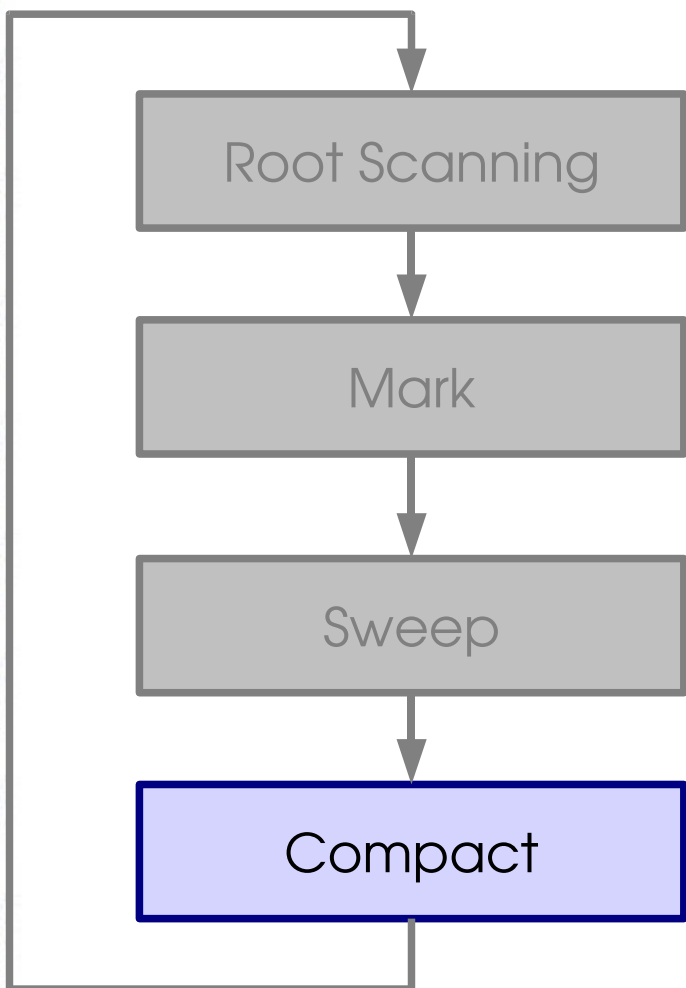
Classic Mark & Sweep GC



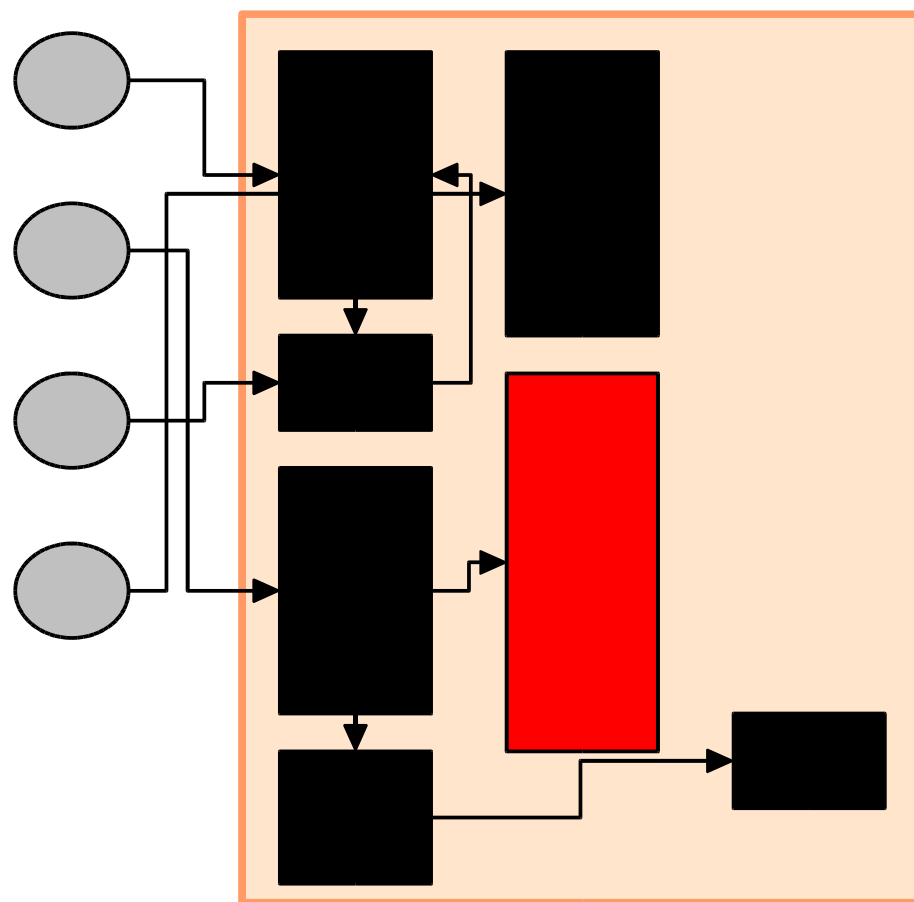
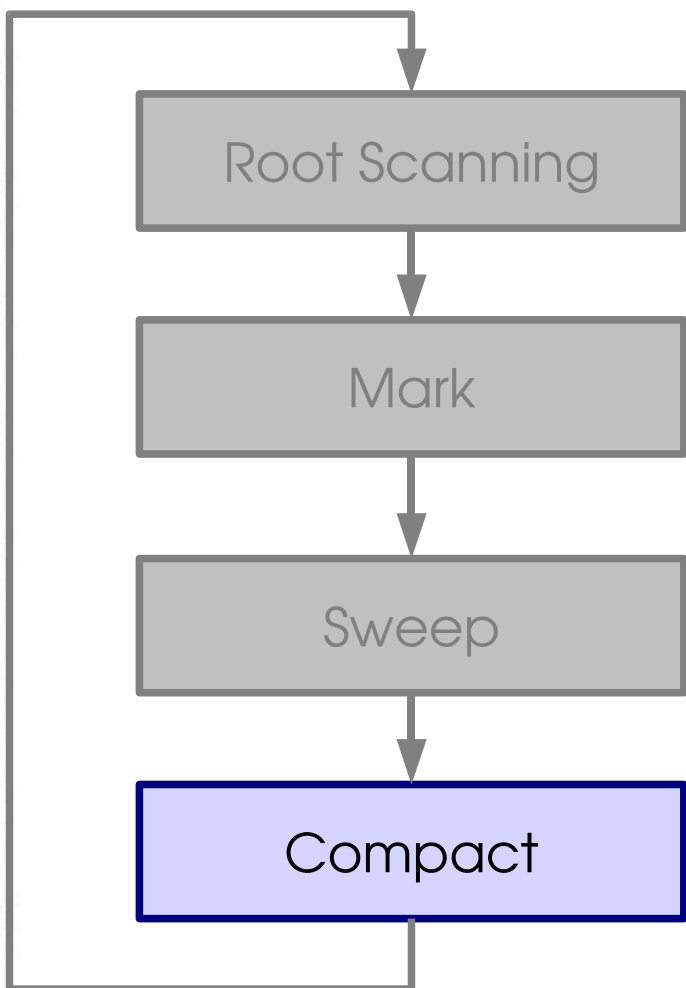
Classic Mark & Sweep GC



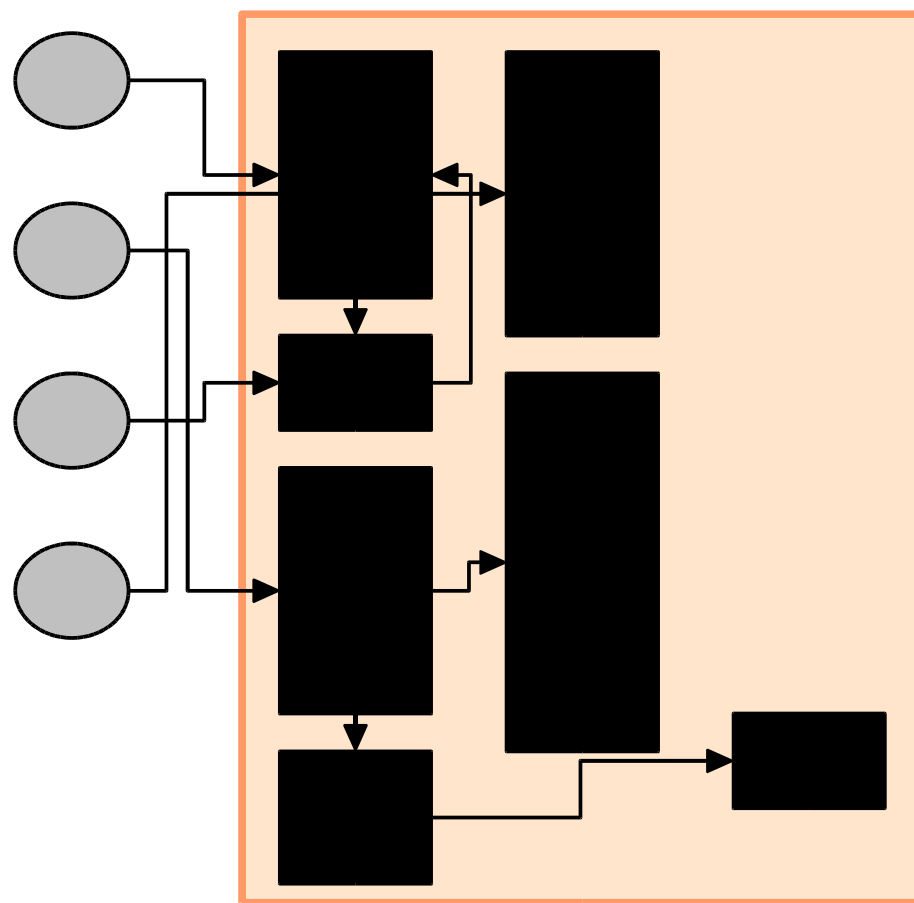
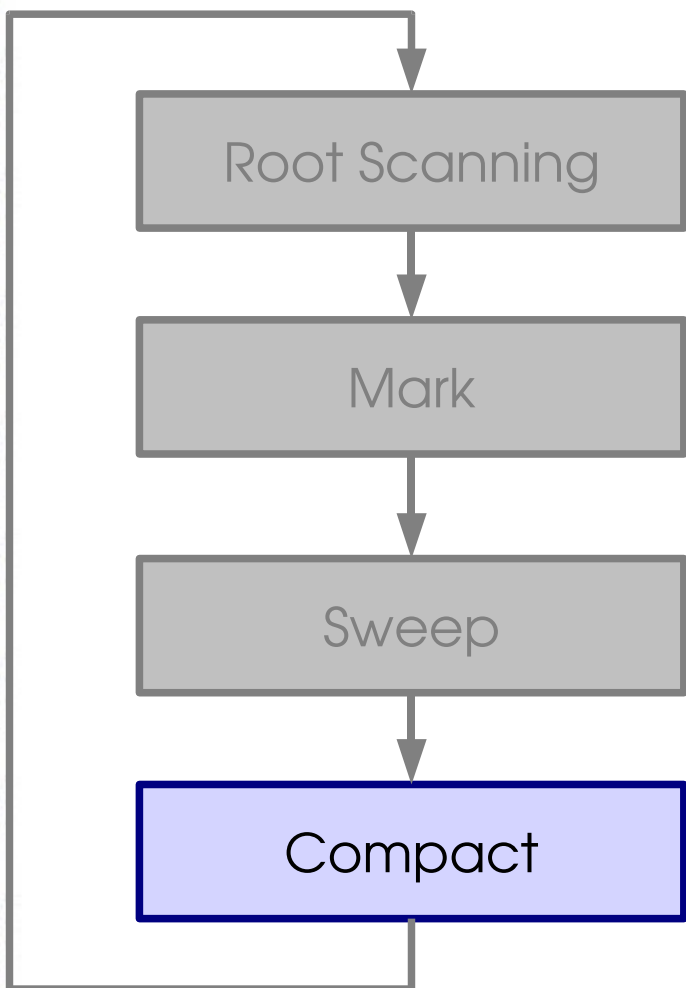
Classic Mark & Sweep GC



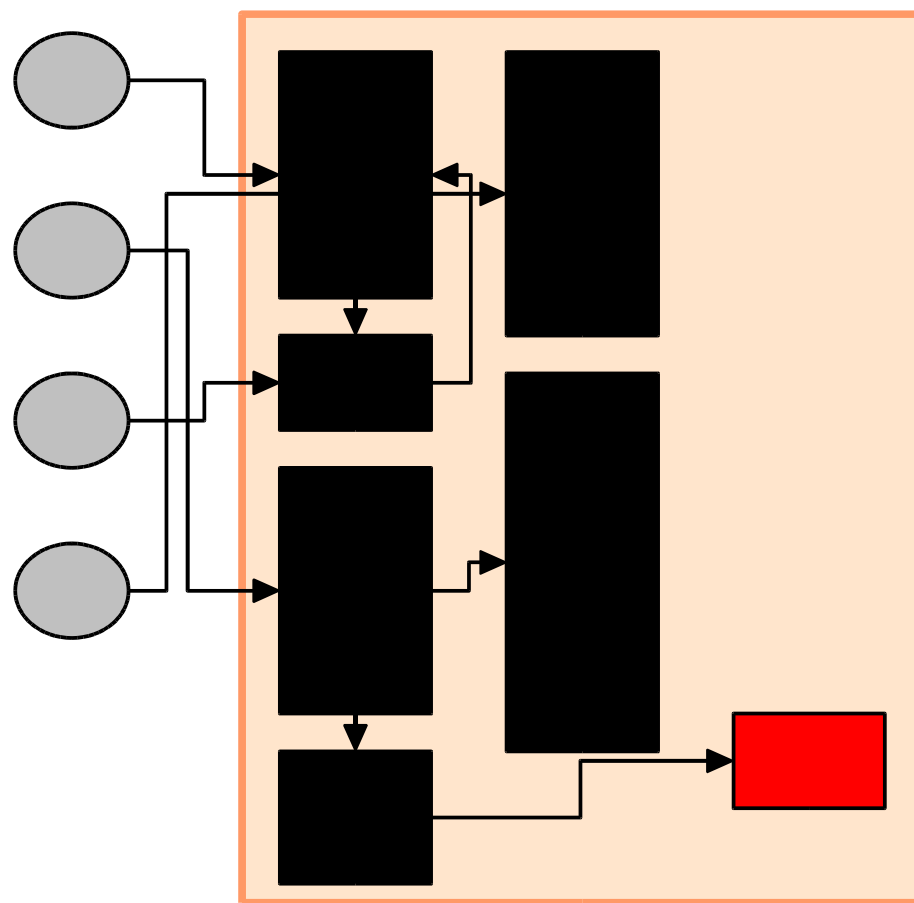
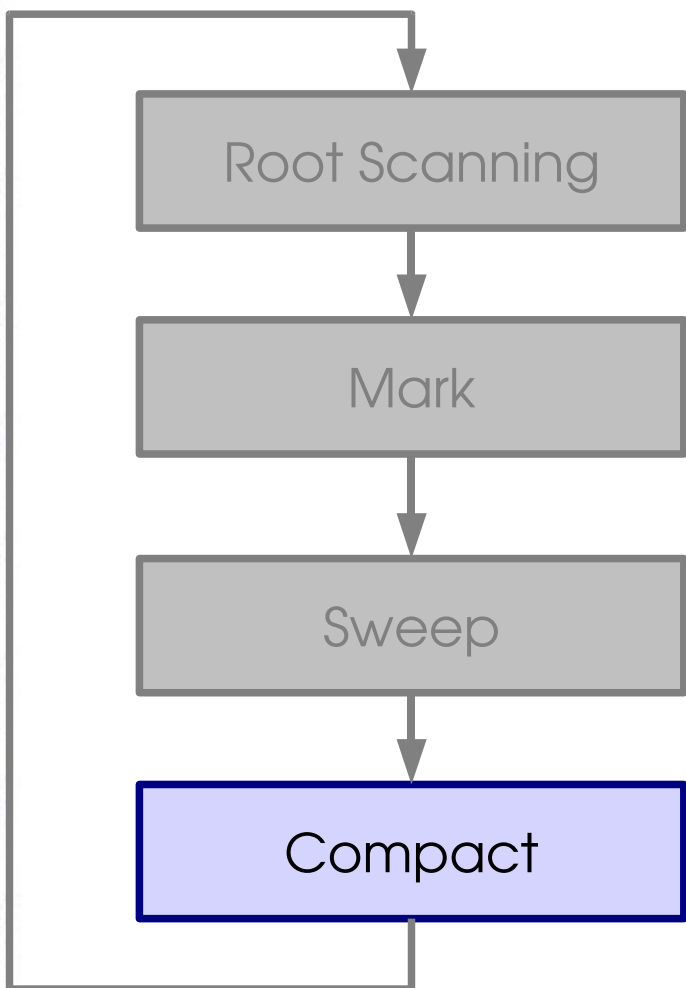
Classic Mark & Sweep GC



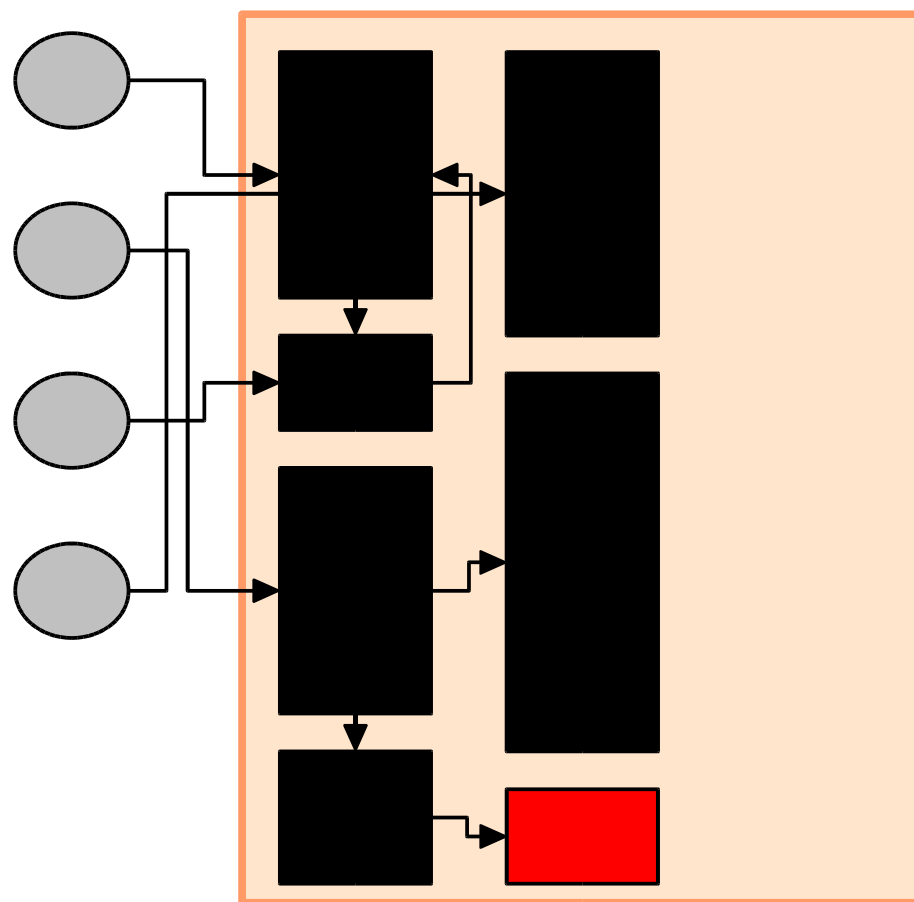
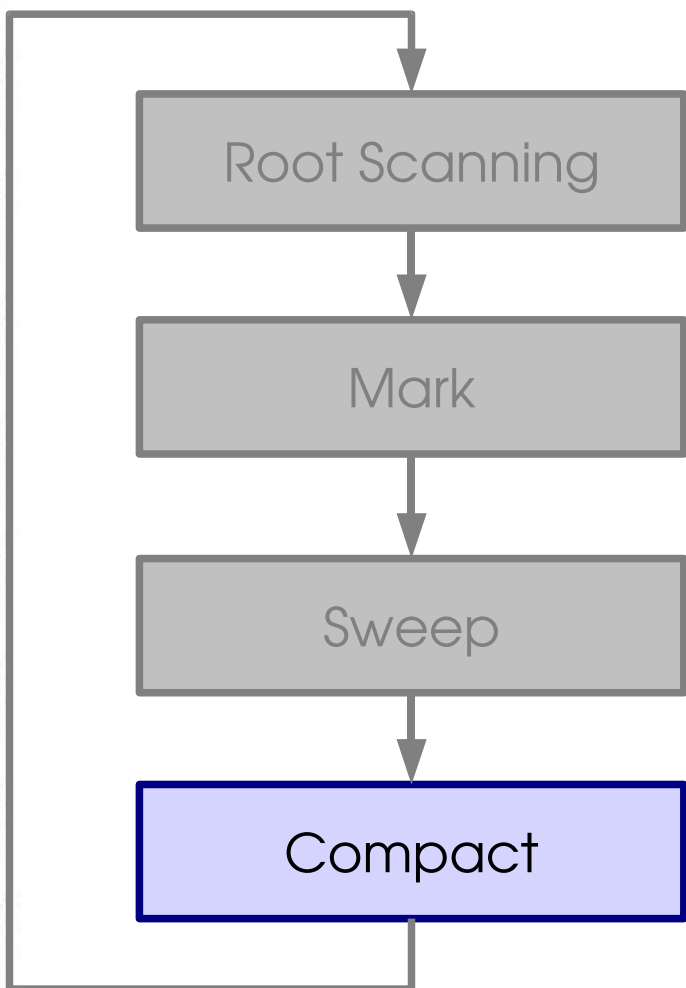
Classic Mark & Sweep GC



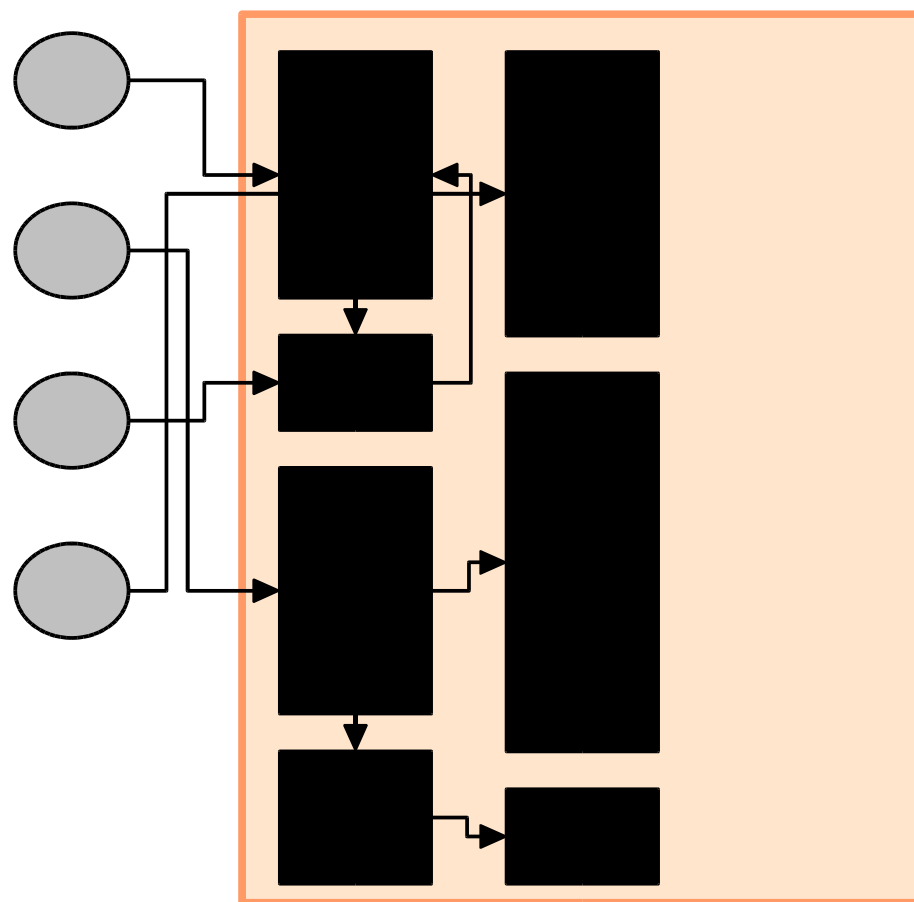
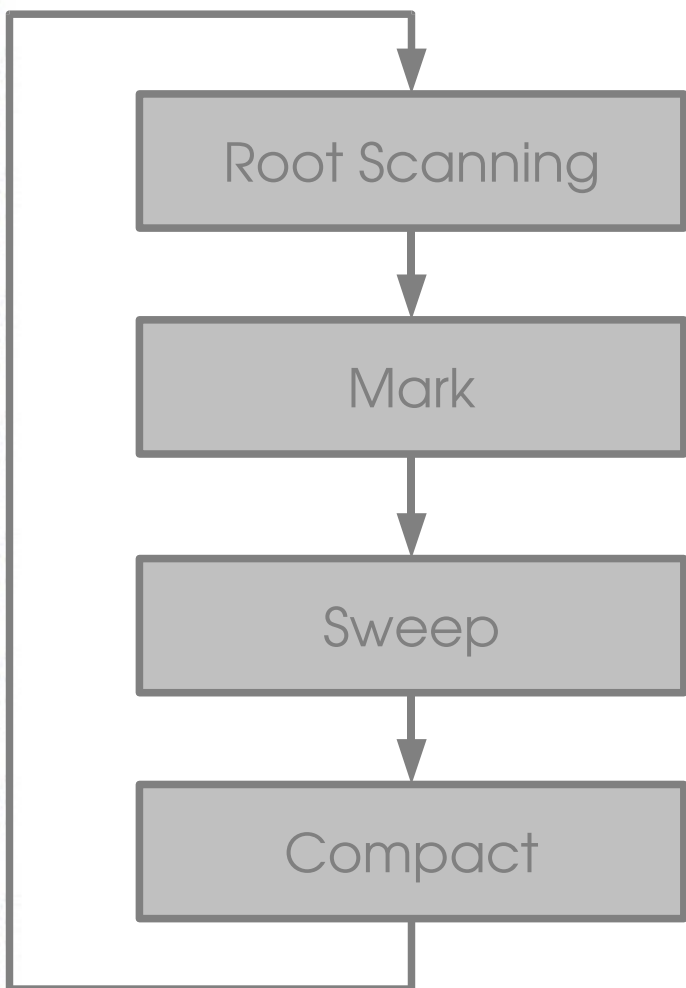
Classic Mark & Sweep GC



Classic Mark & Sweep GC



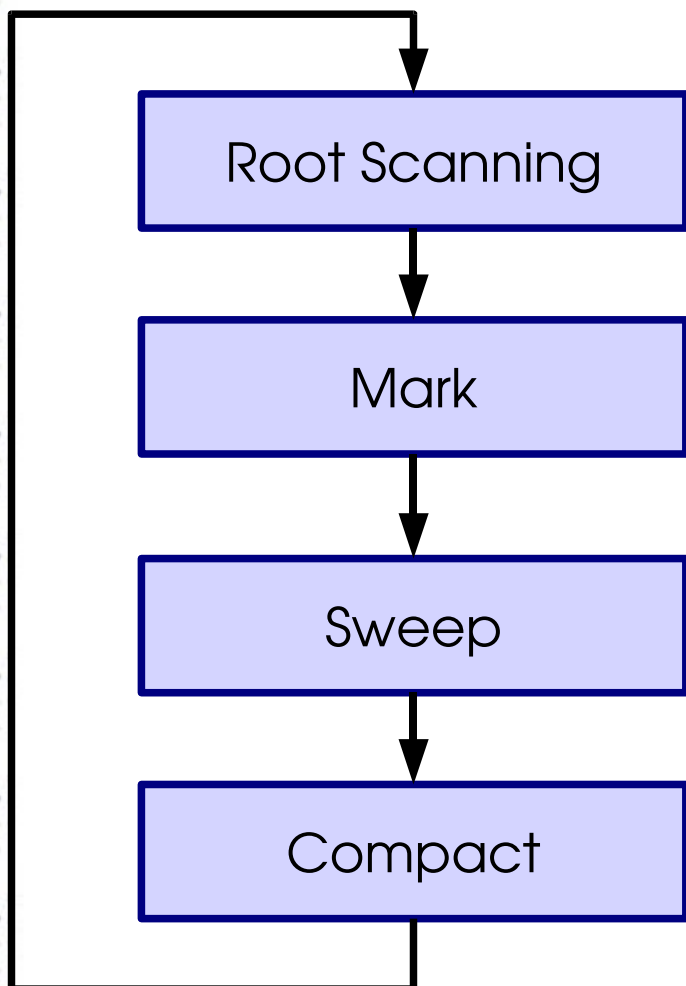
Classic Mark & Sweep GC



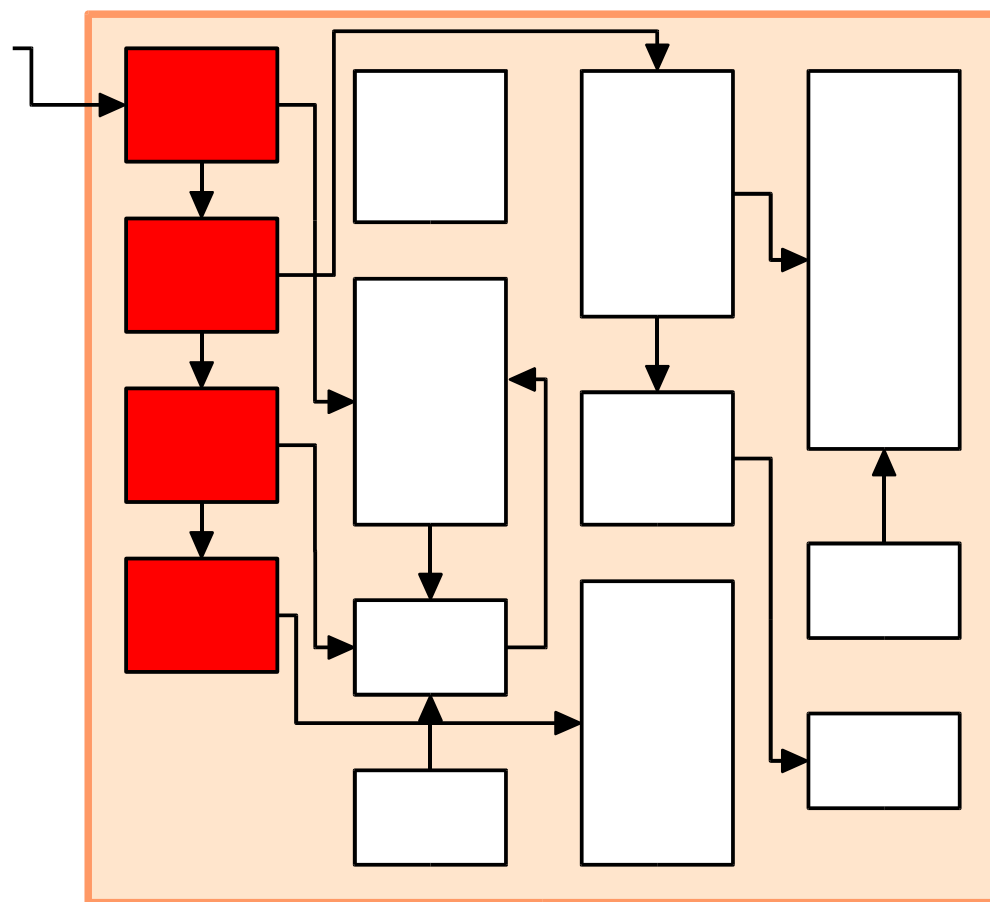
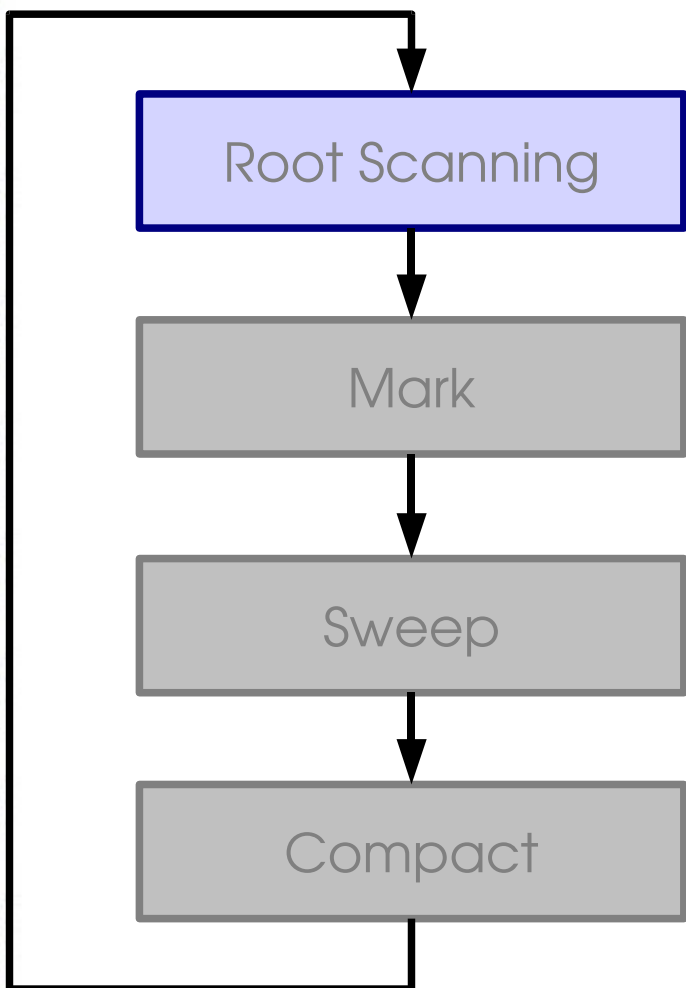
Classic Mark & Sweep GC

➔ Causes long and unpredictable pauses!

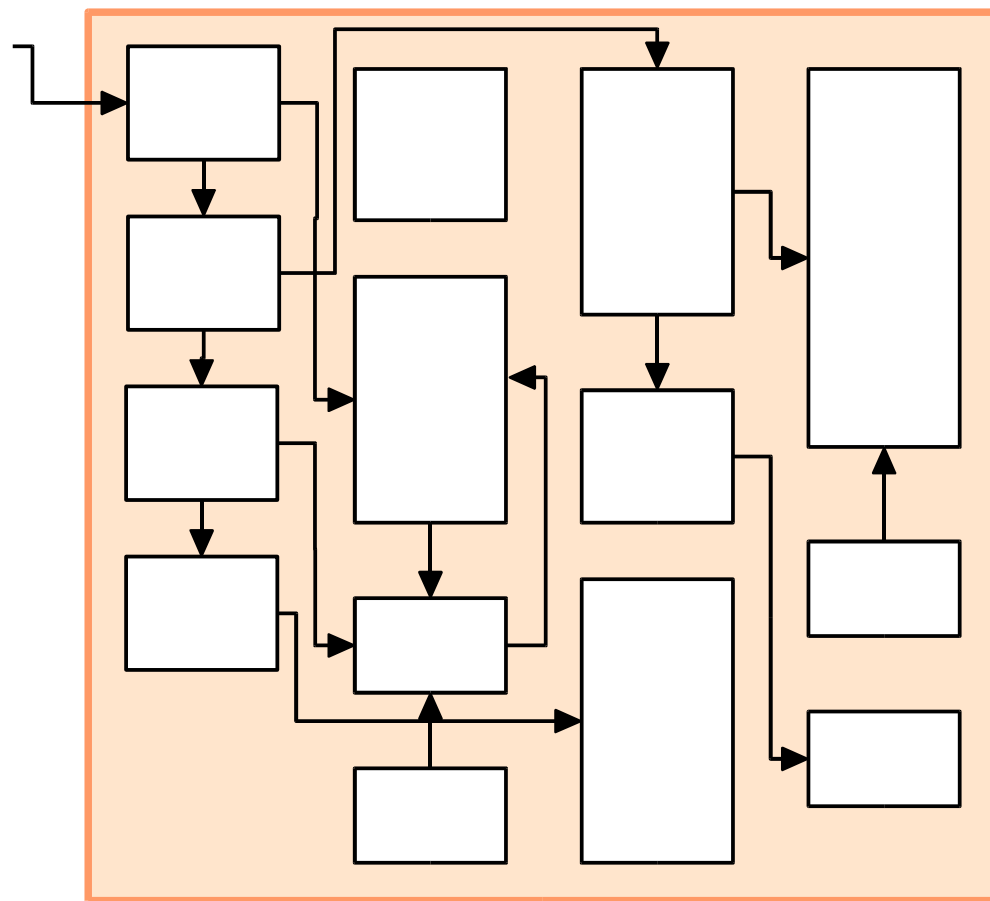
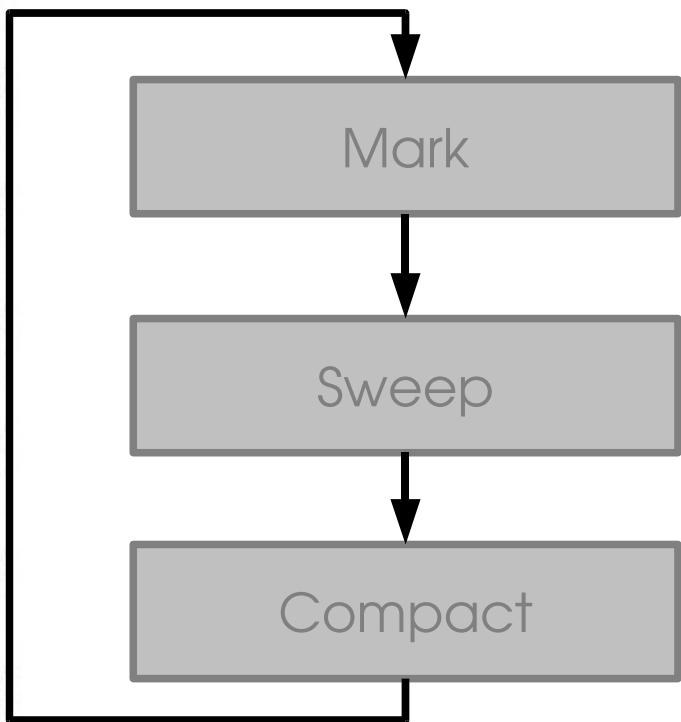
JamaicaVM realtime GC



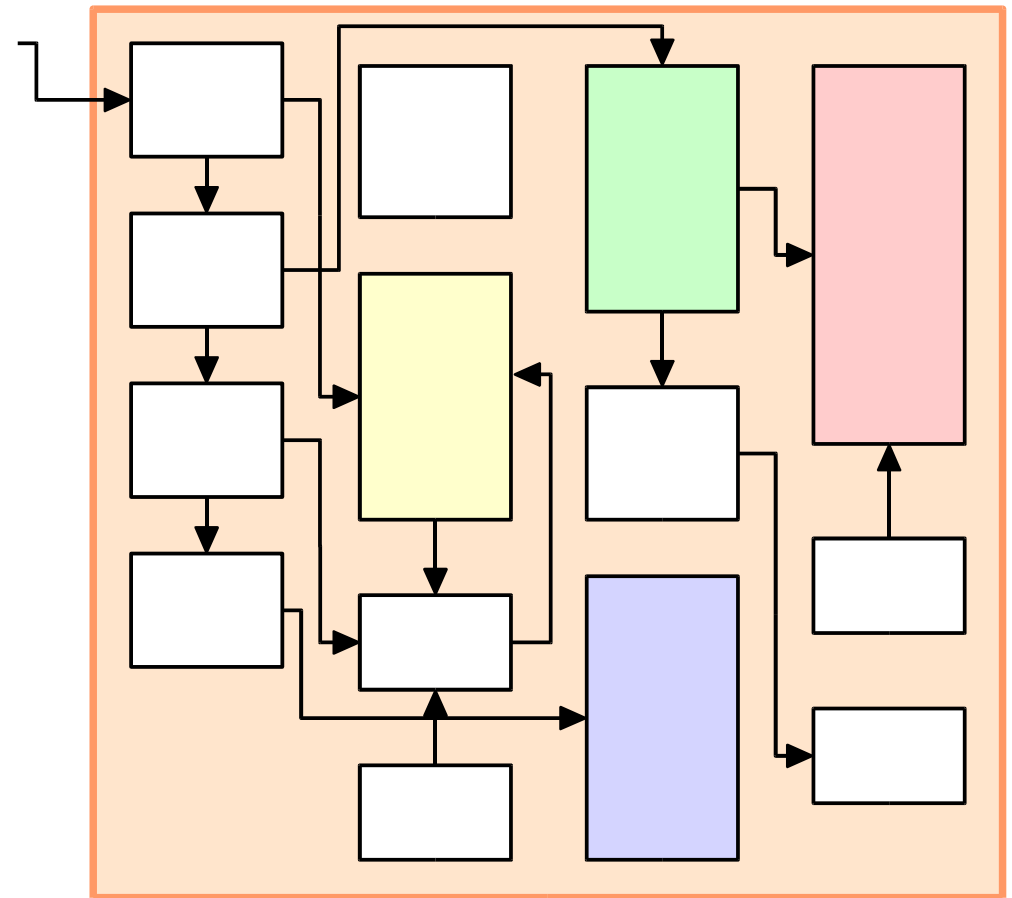
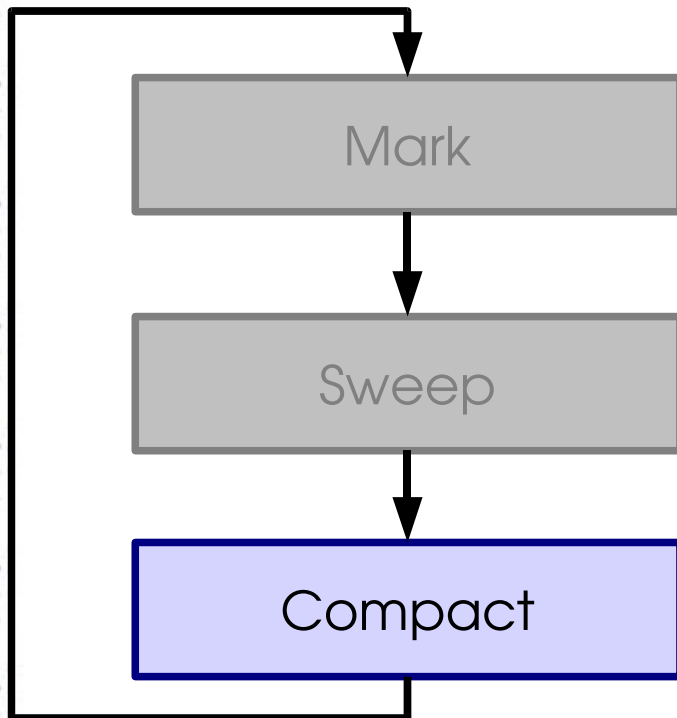
JamaicaVM realtime GC



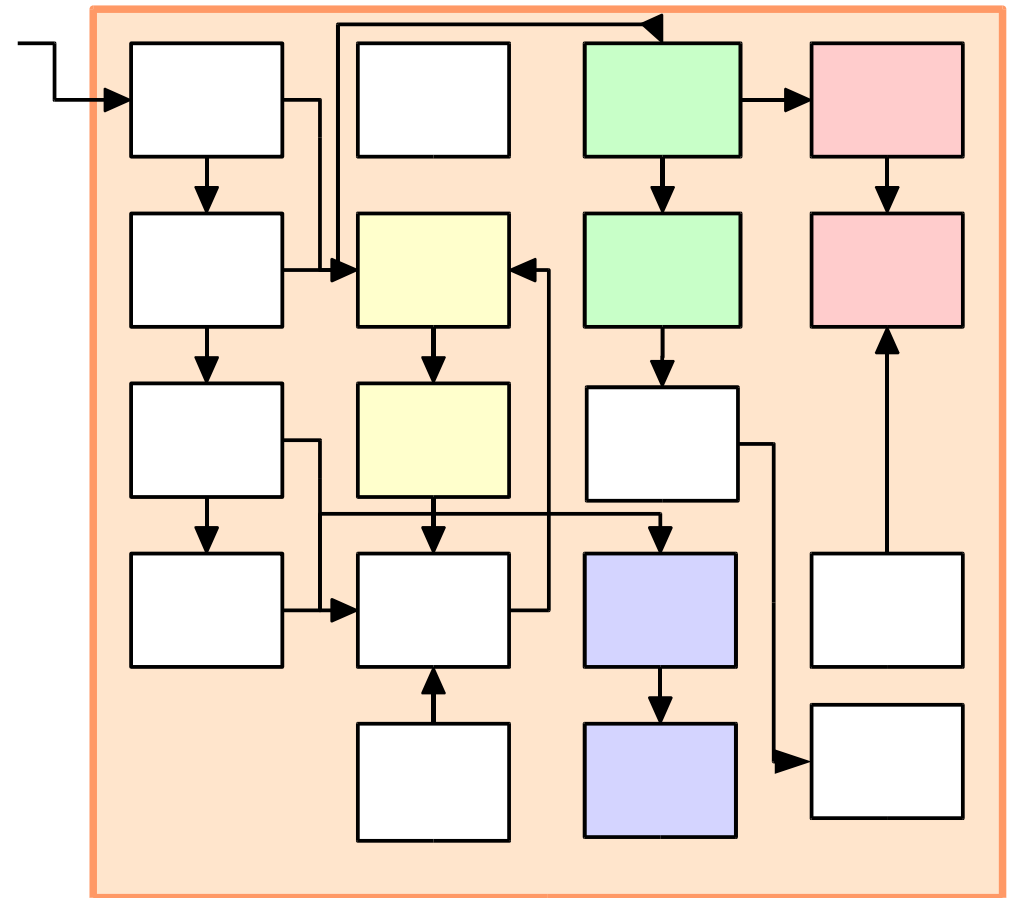
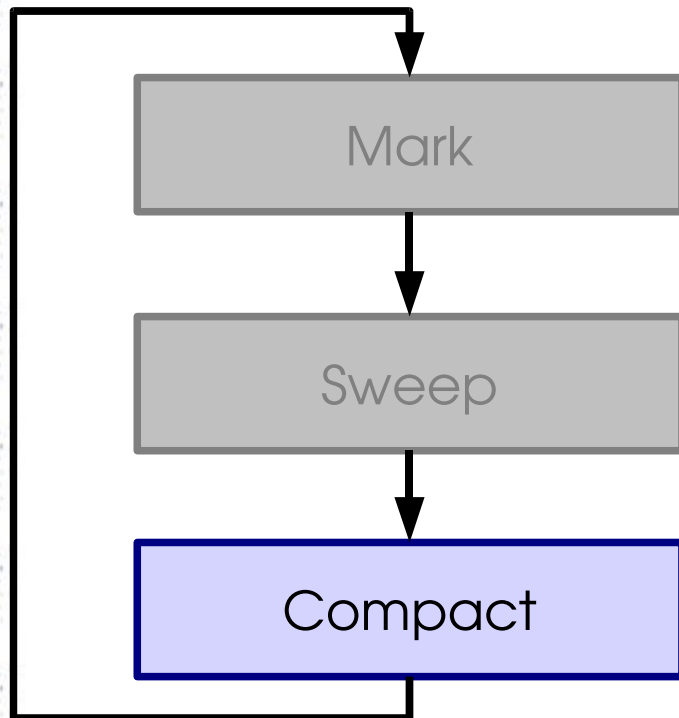
JamaicaVM realtime GC



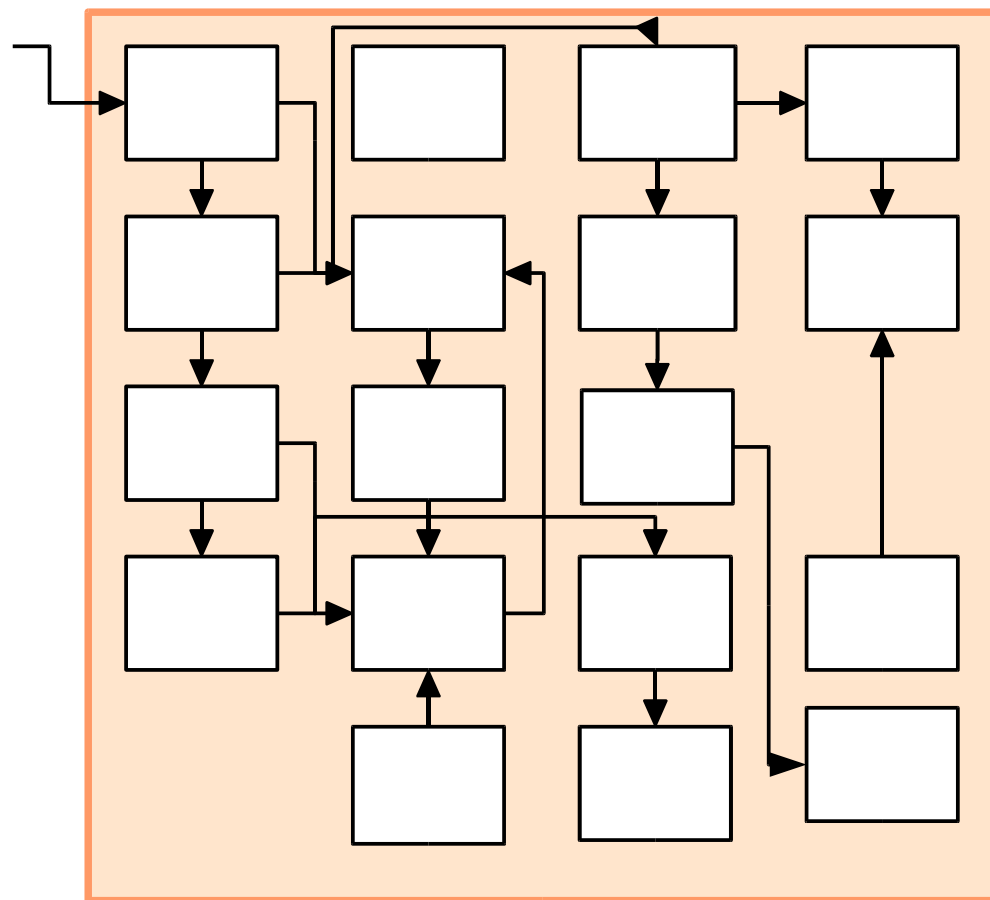
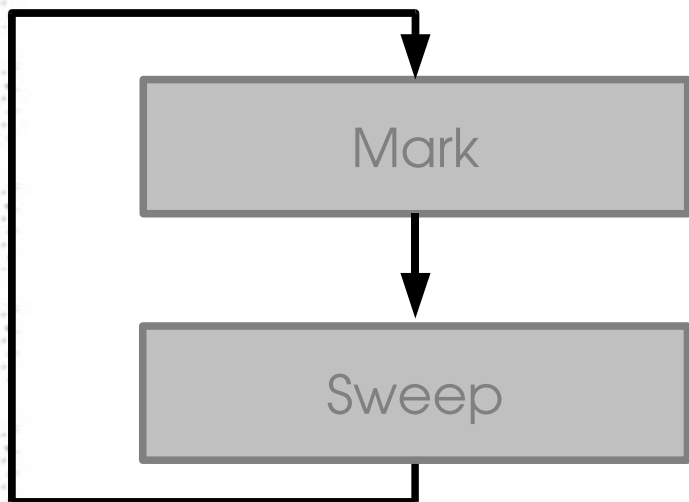
JamaicaVM realtime GC



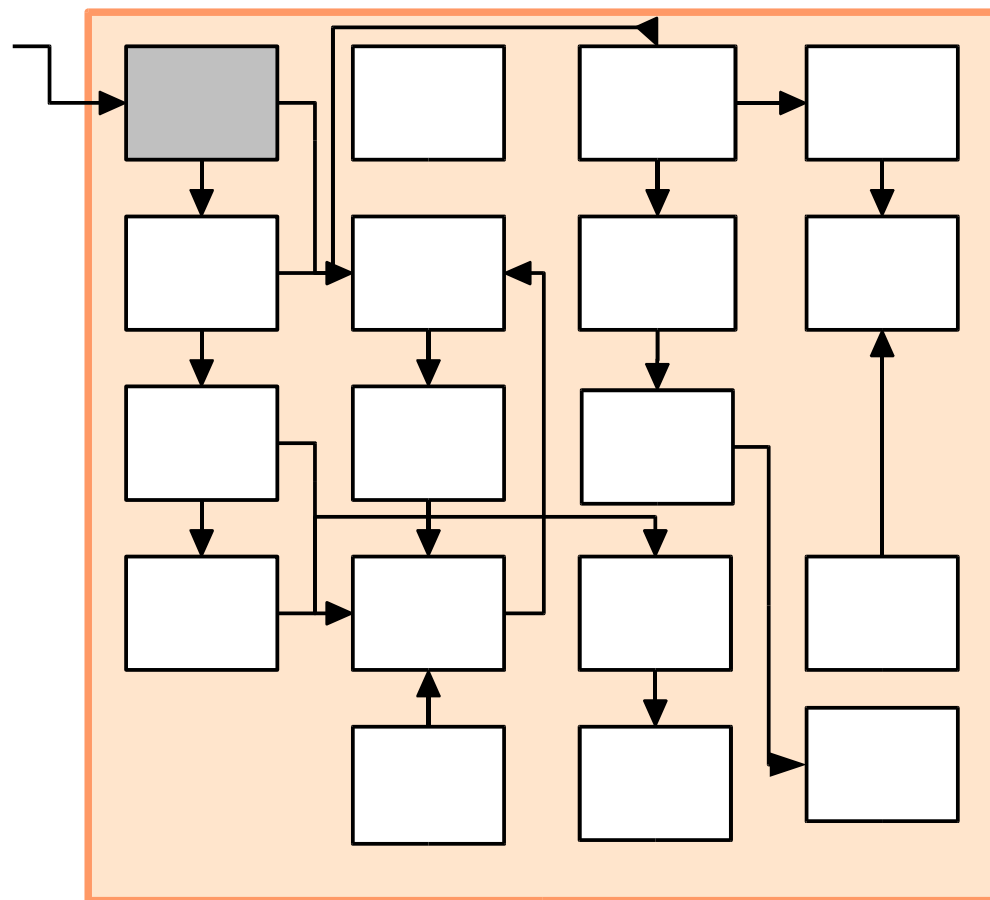
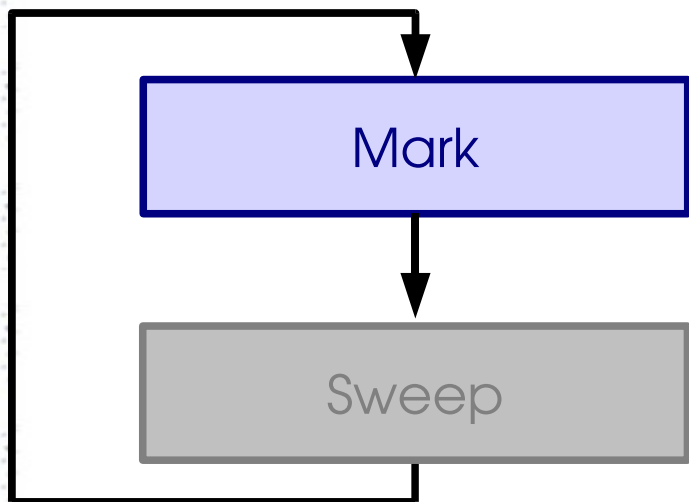
JamaicaVM realtime GC



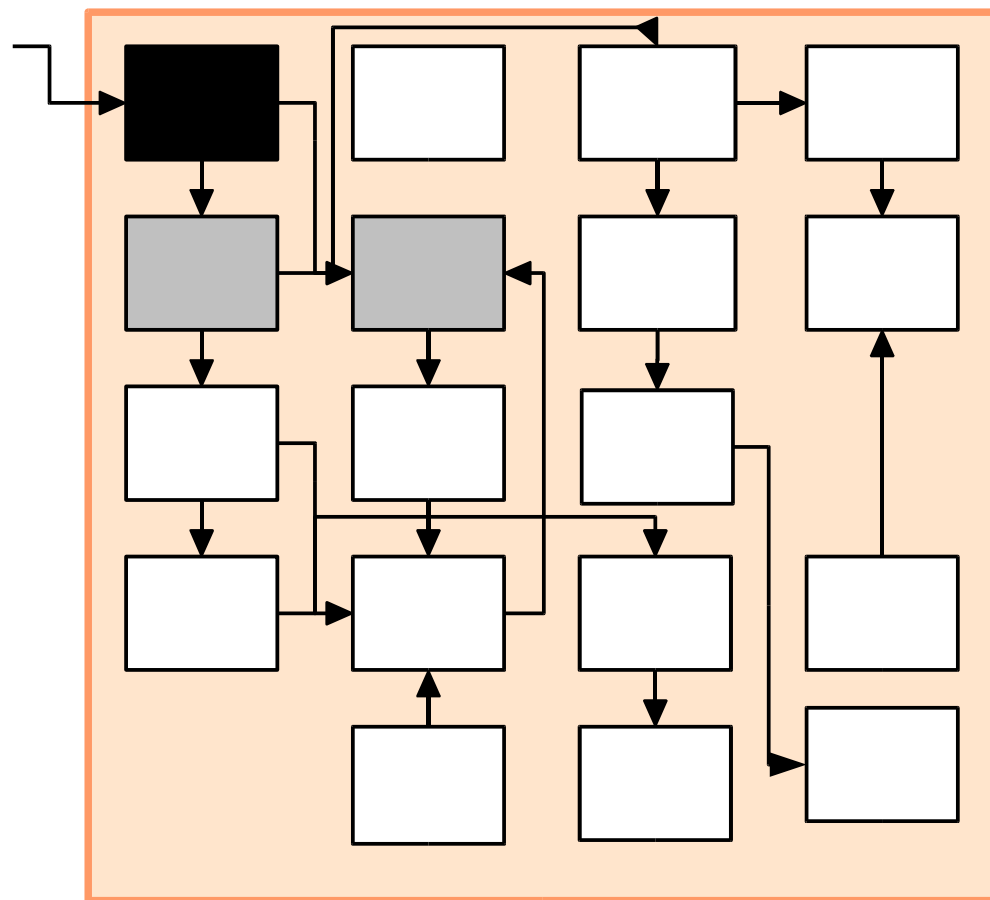
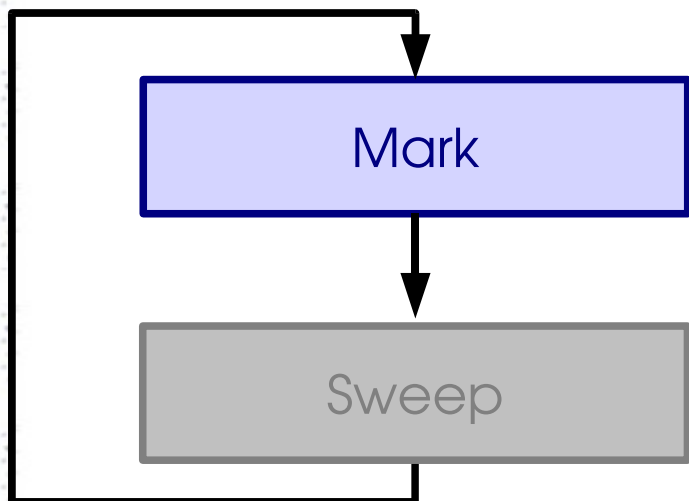
JamaicaVM realtime GC



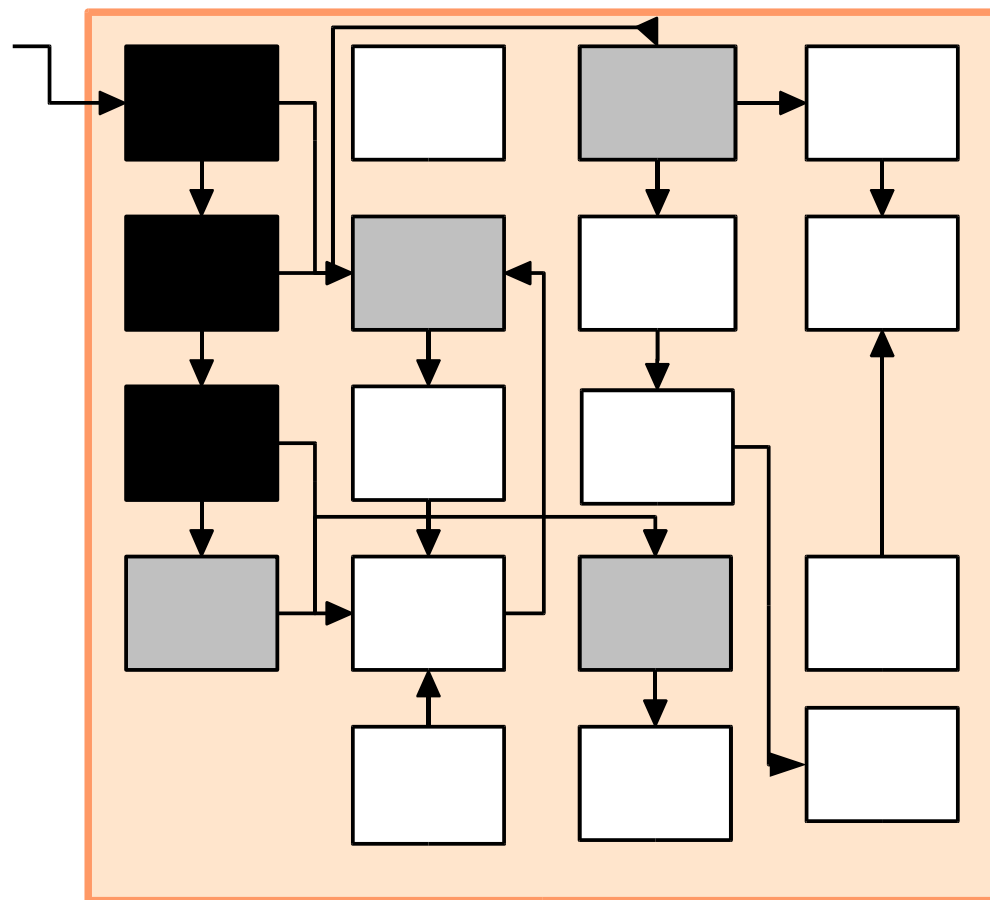
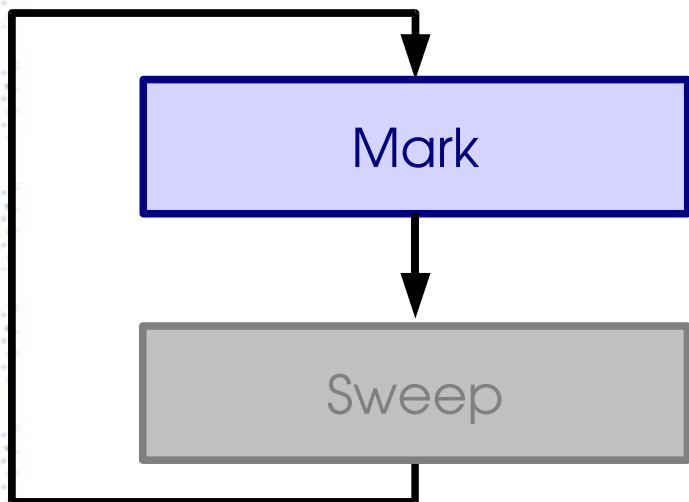
JamaicaVM realtime GC



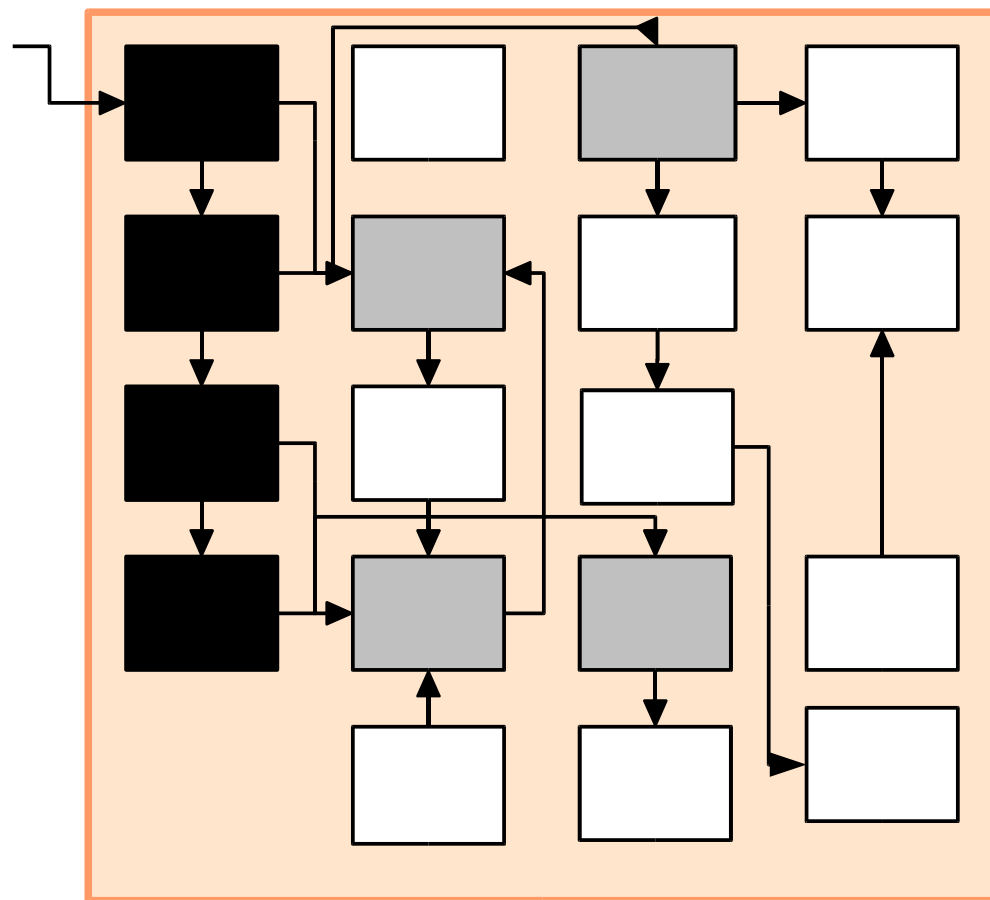
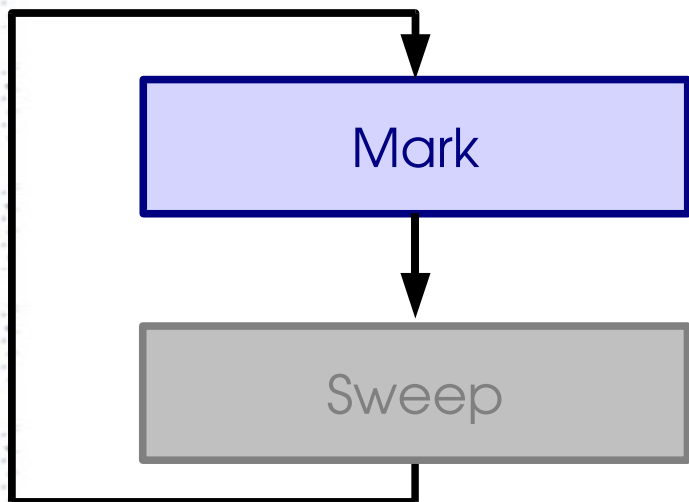
JamaicaVM realtime GC



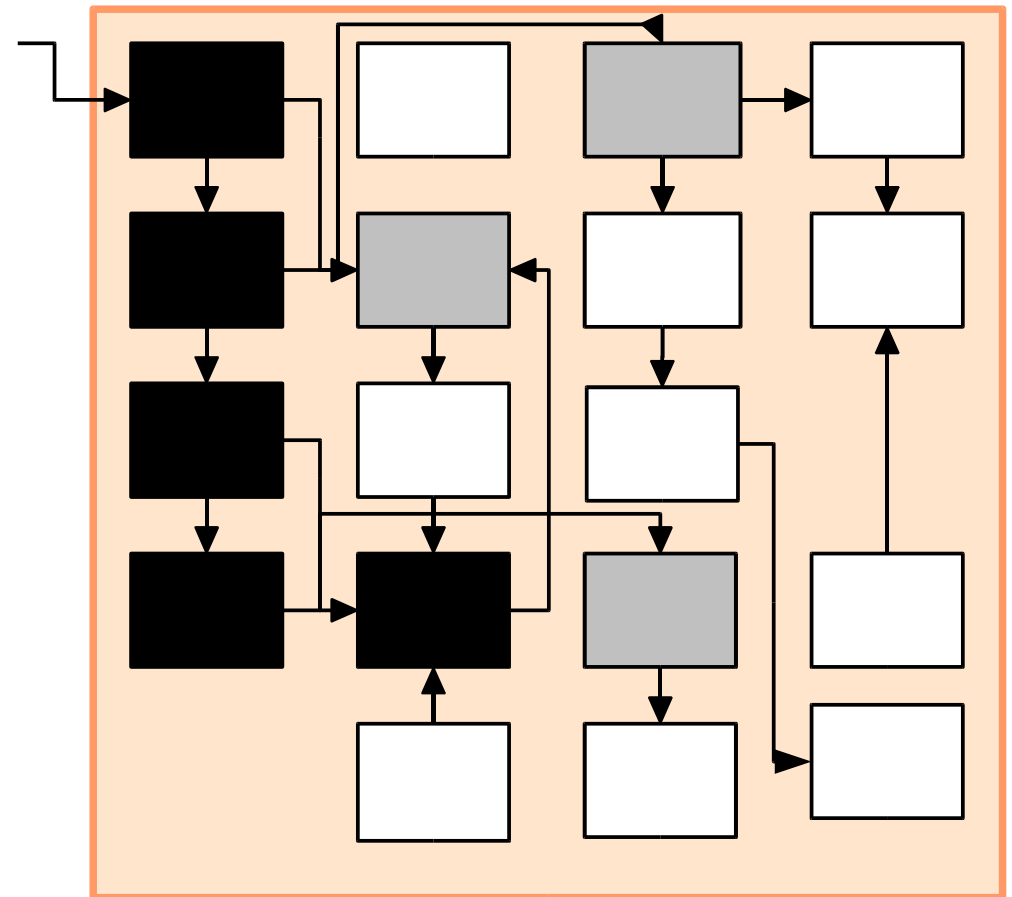
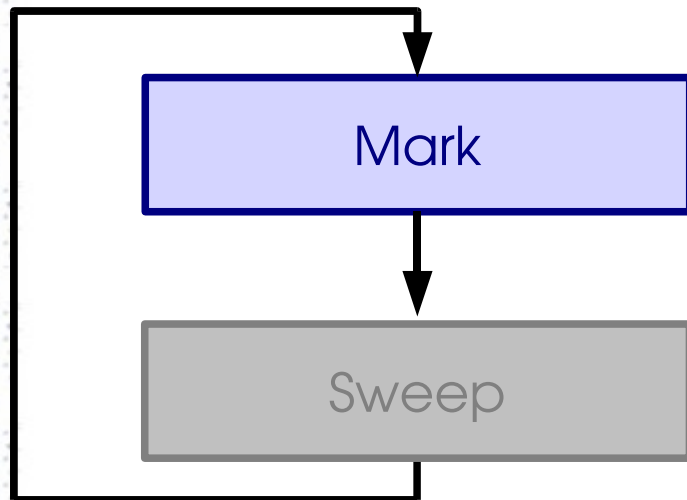
JamaicaVM realtime GC



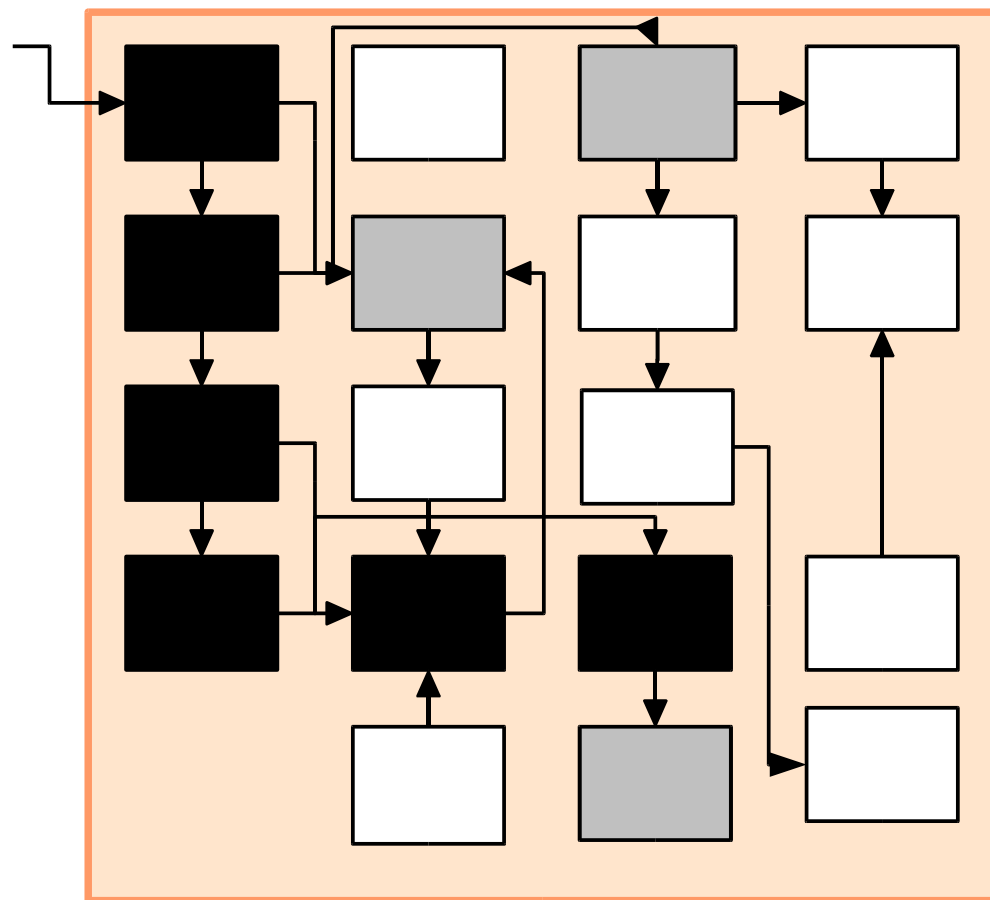
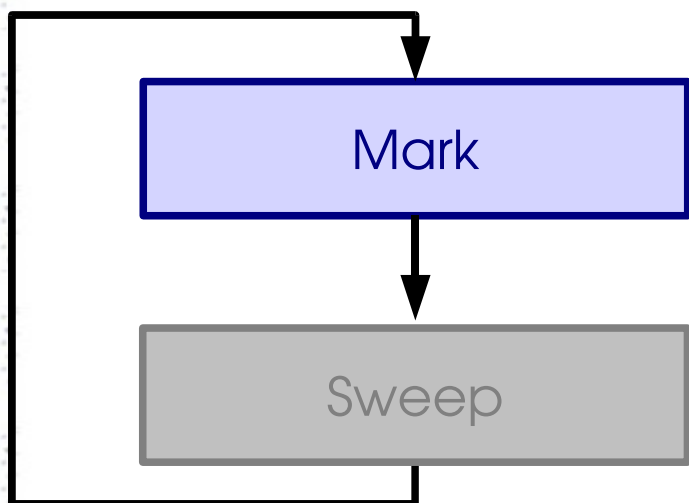
JamaicaVM realtime GC



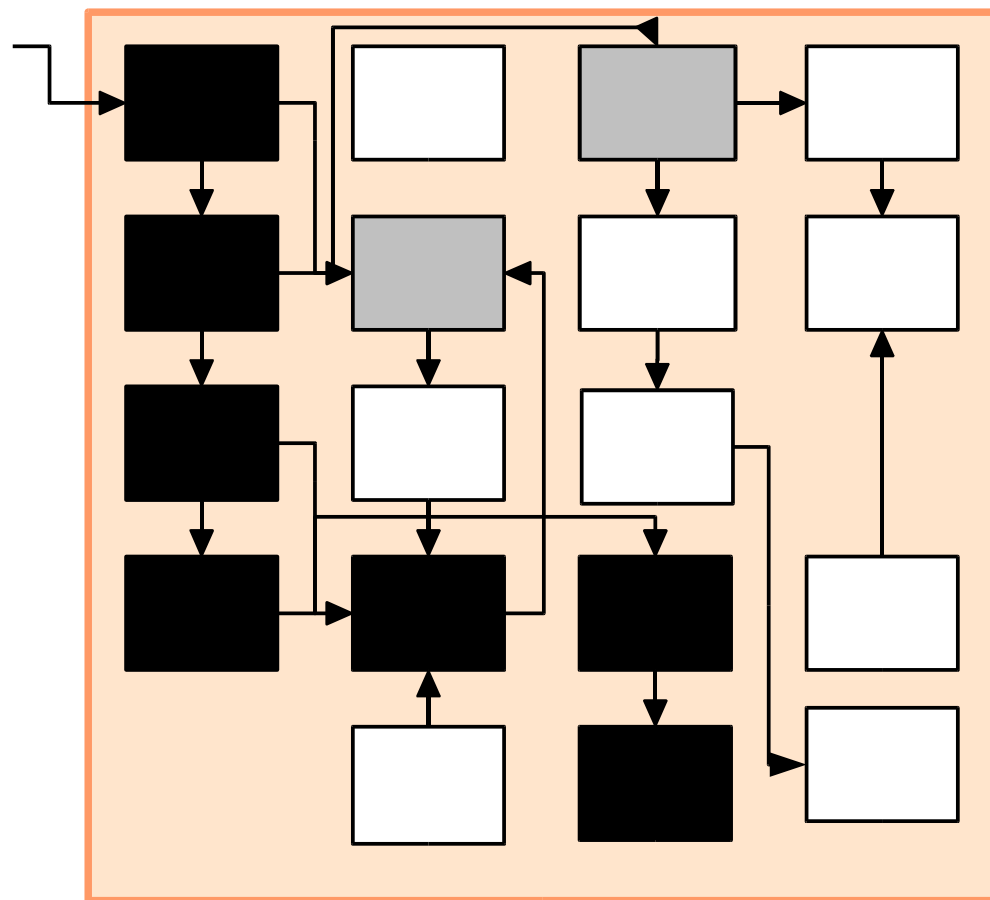
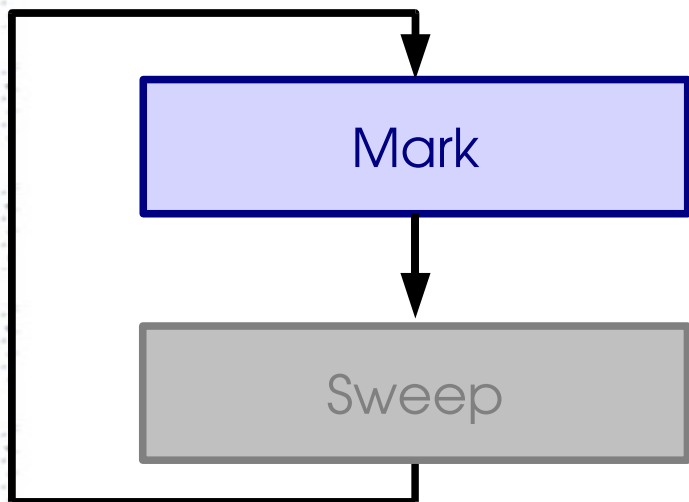
JamaicaVM realtime GC



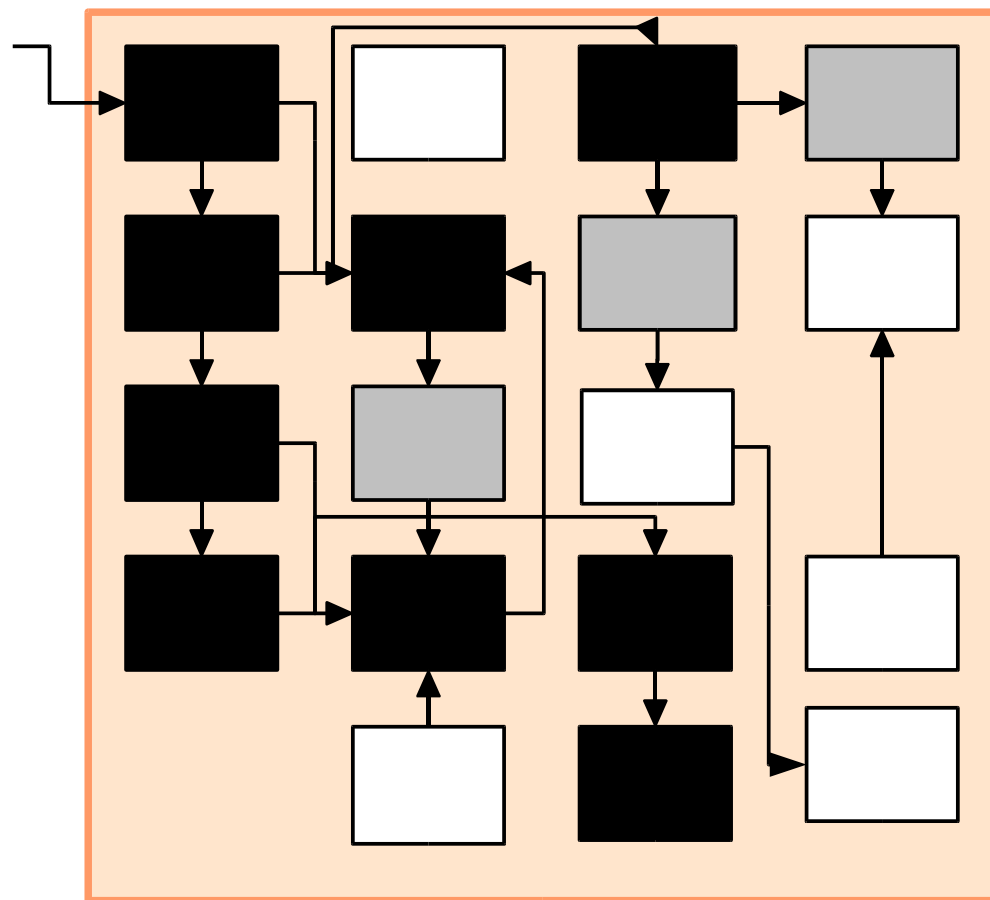
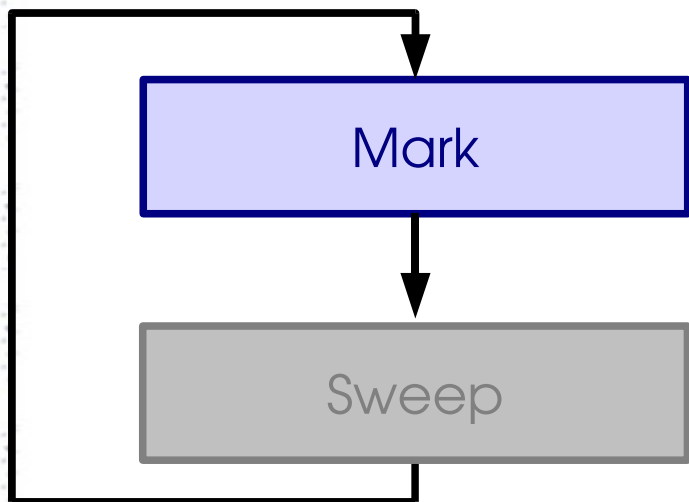
JamaicaVM realtime GC



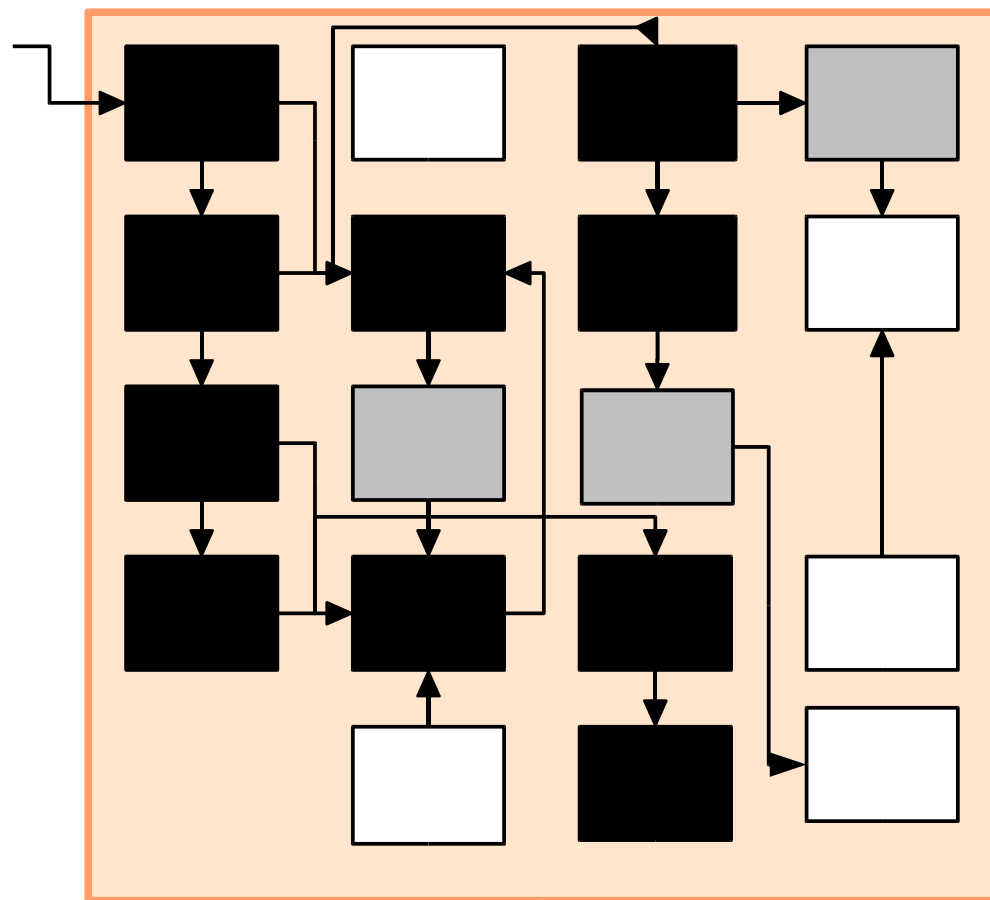
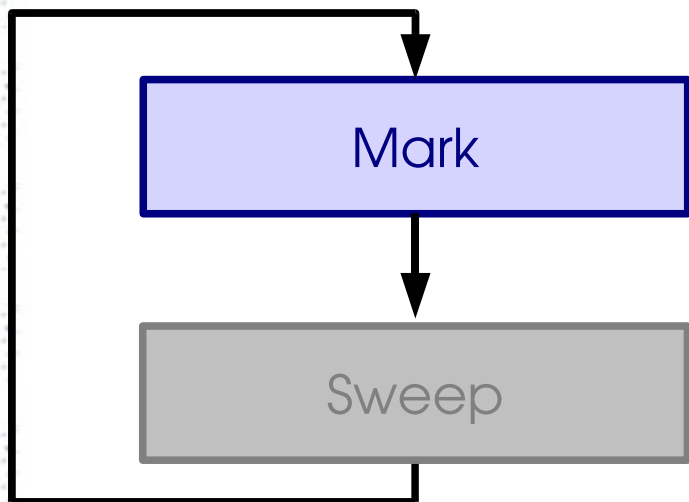
JamaicaVM realtime GC



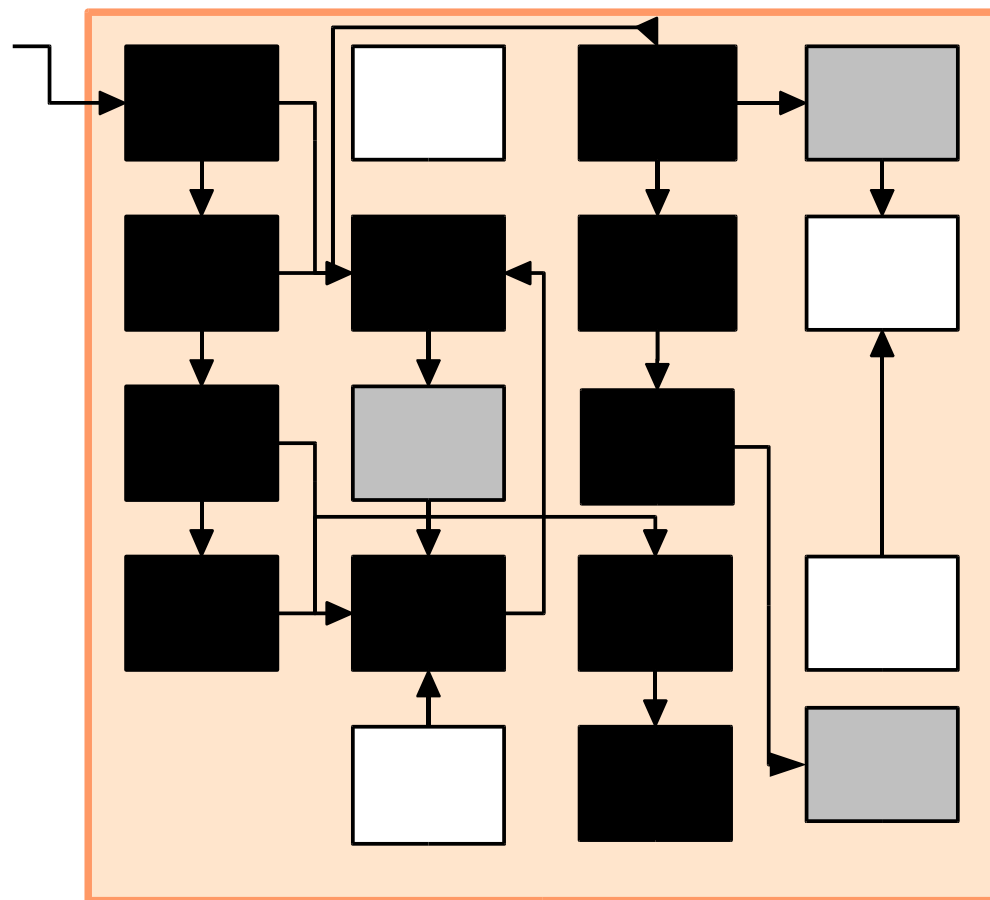
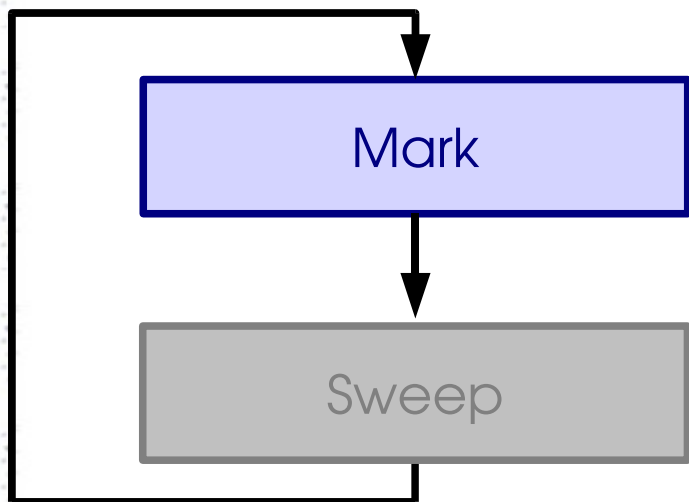
JamaicaVM realtime GC



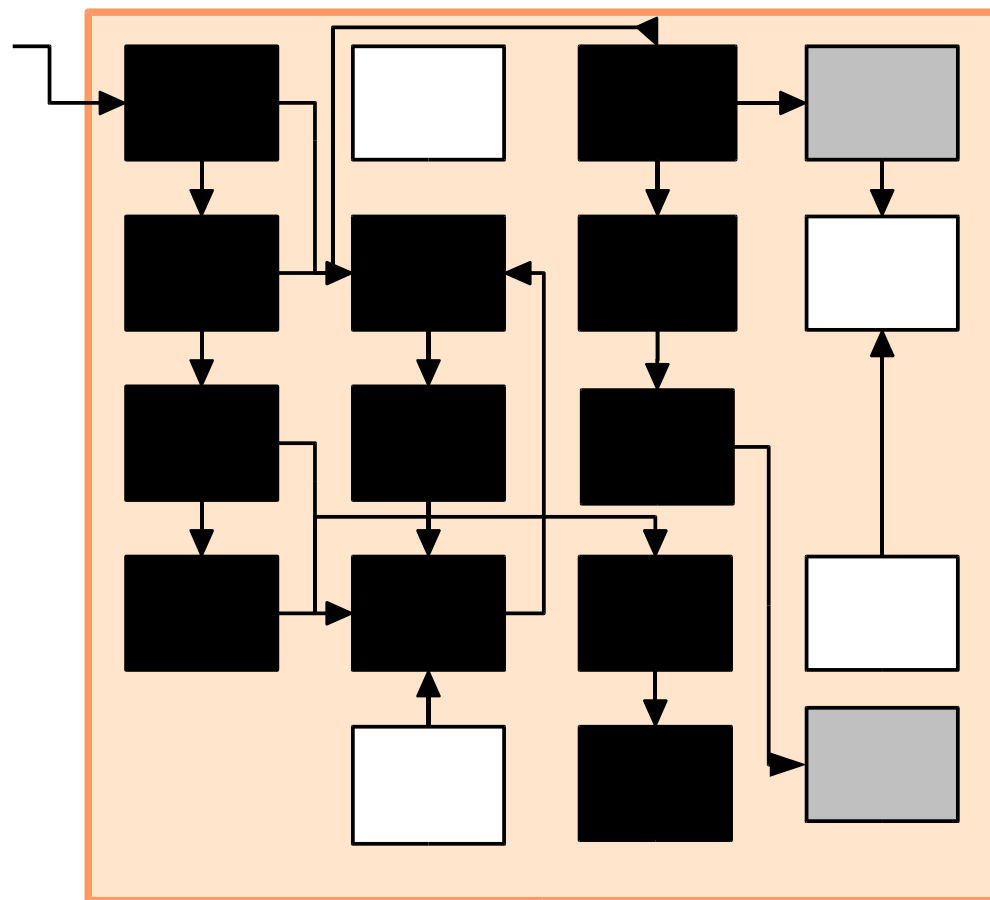
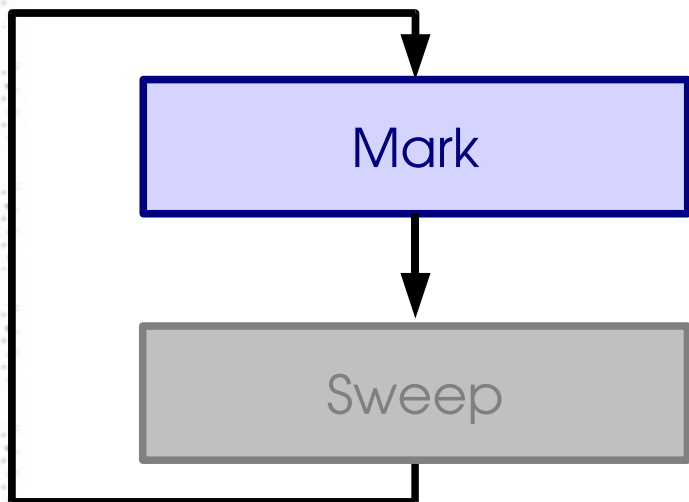
JamaicaVM realtime GC



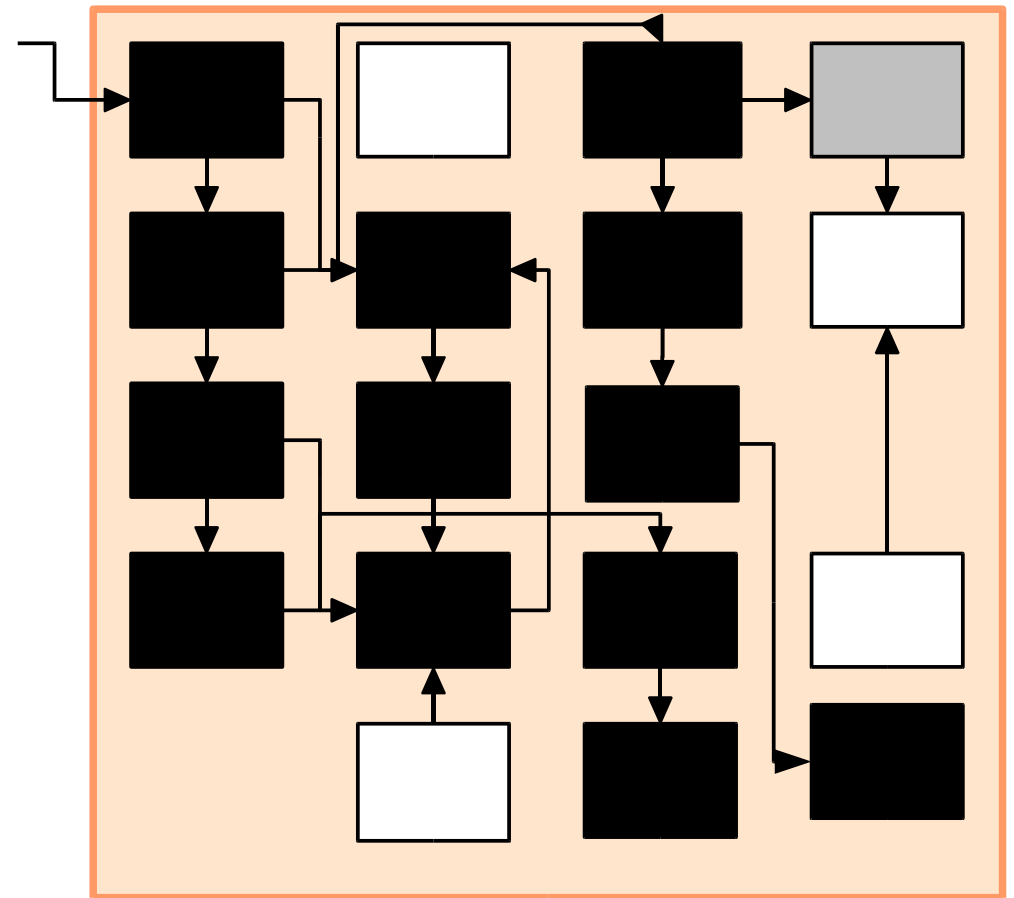
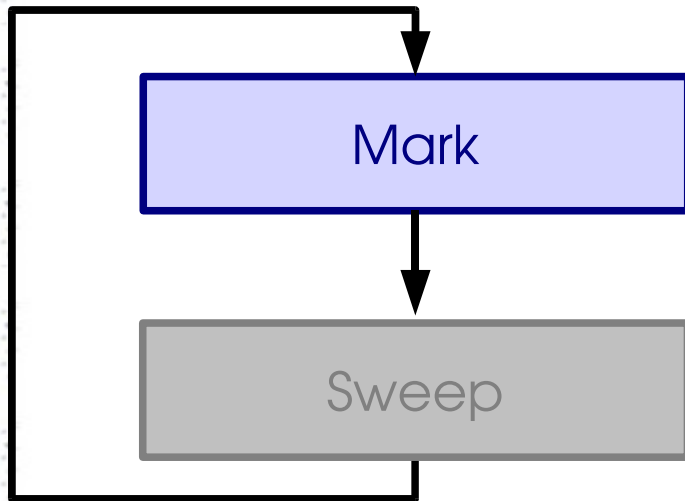
JamaicaVM realtime GC



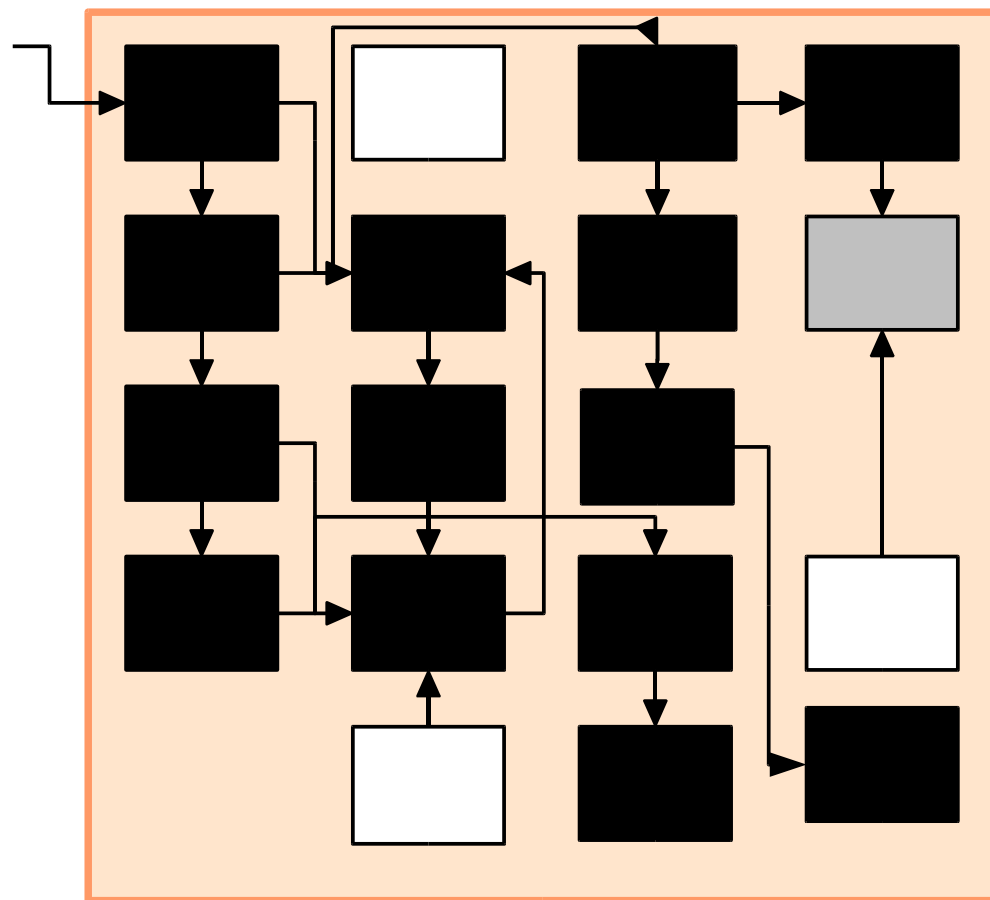
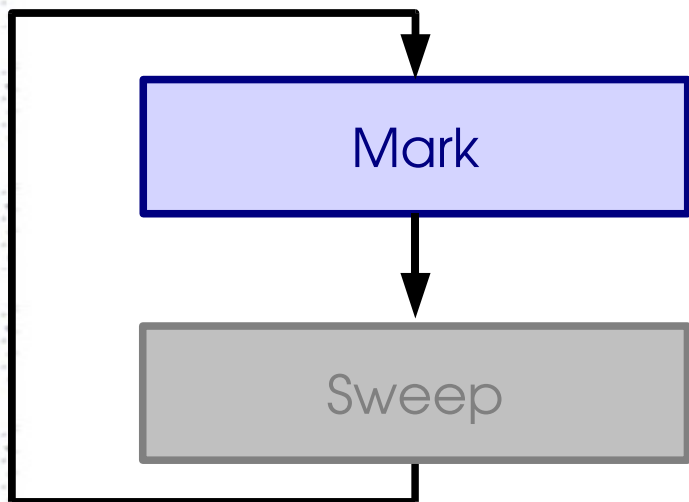
JamaicaVM realtime GC



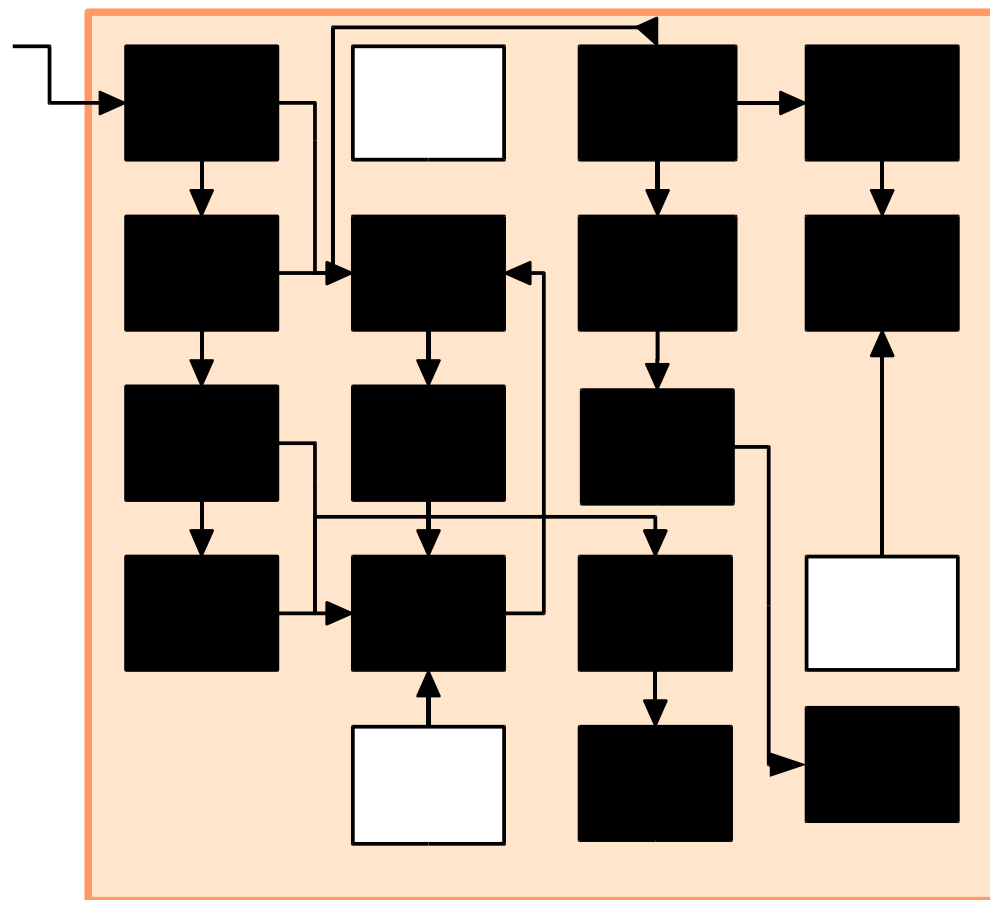
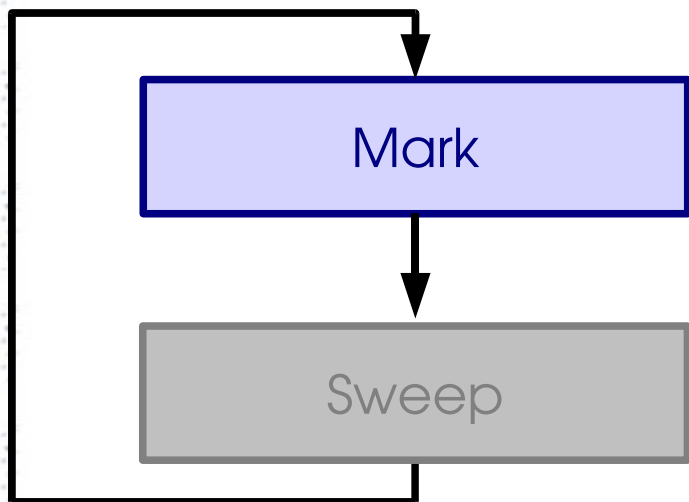
JamaicaVM realtime GC



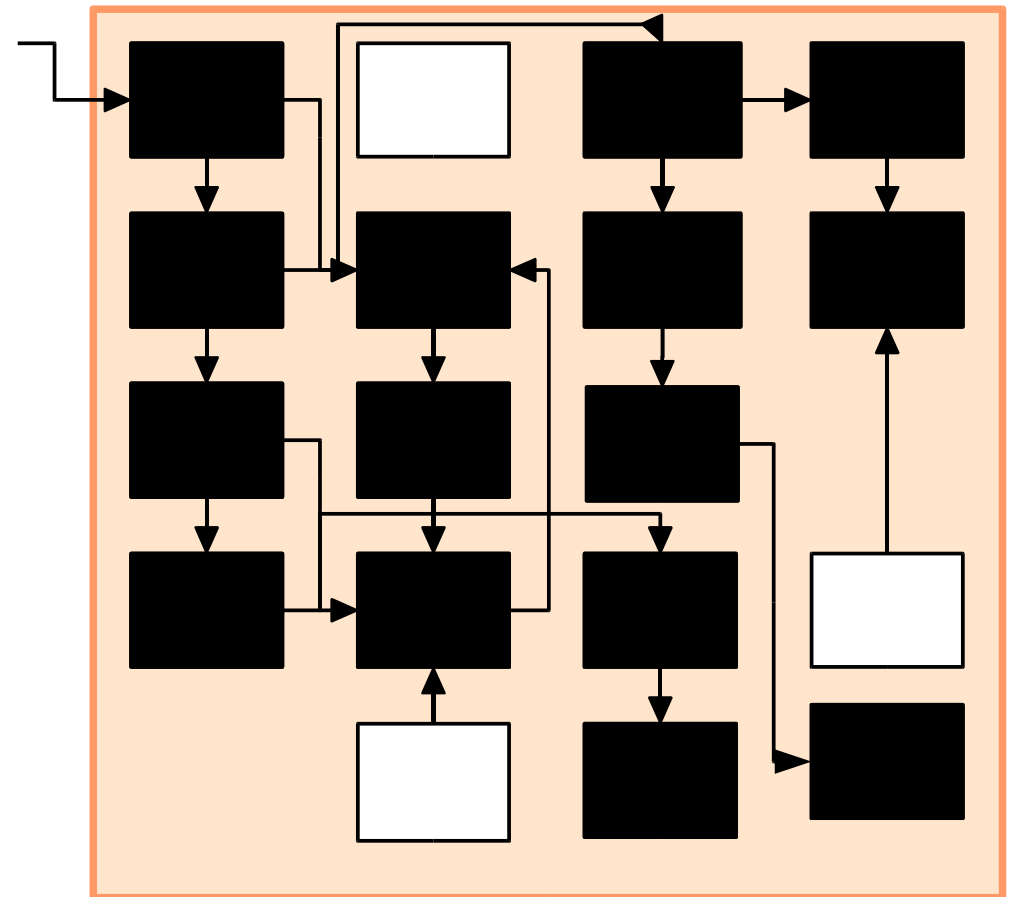
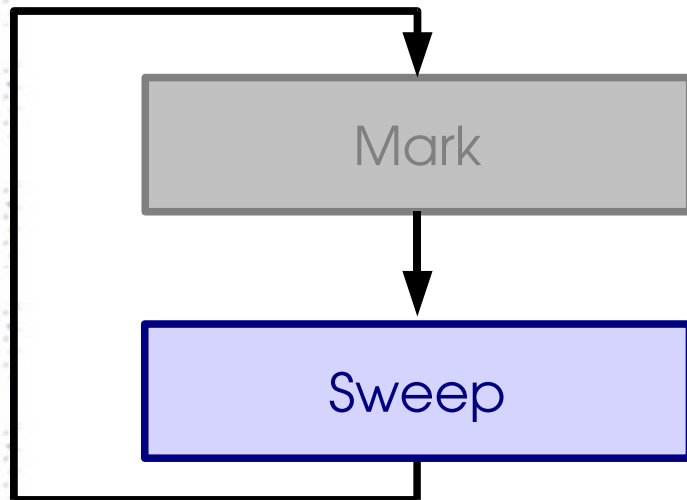
JamaicaVM realtime GC



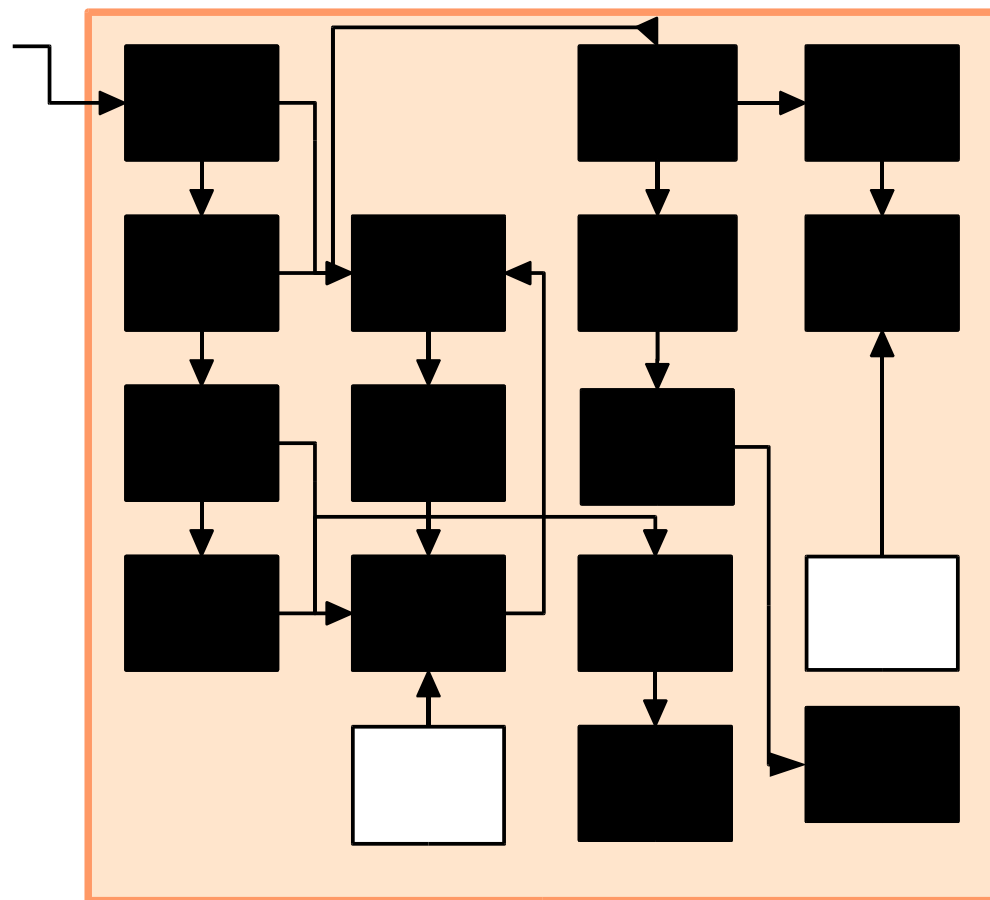
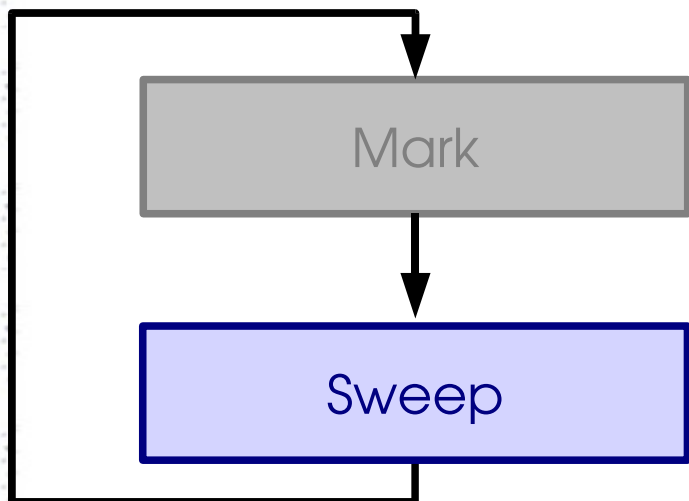
JamaicaVM realtime GC



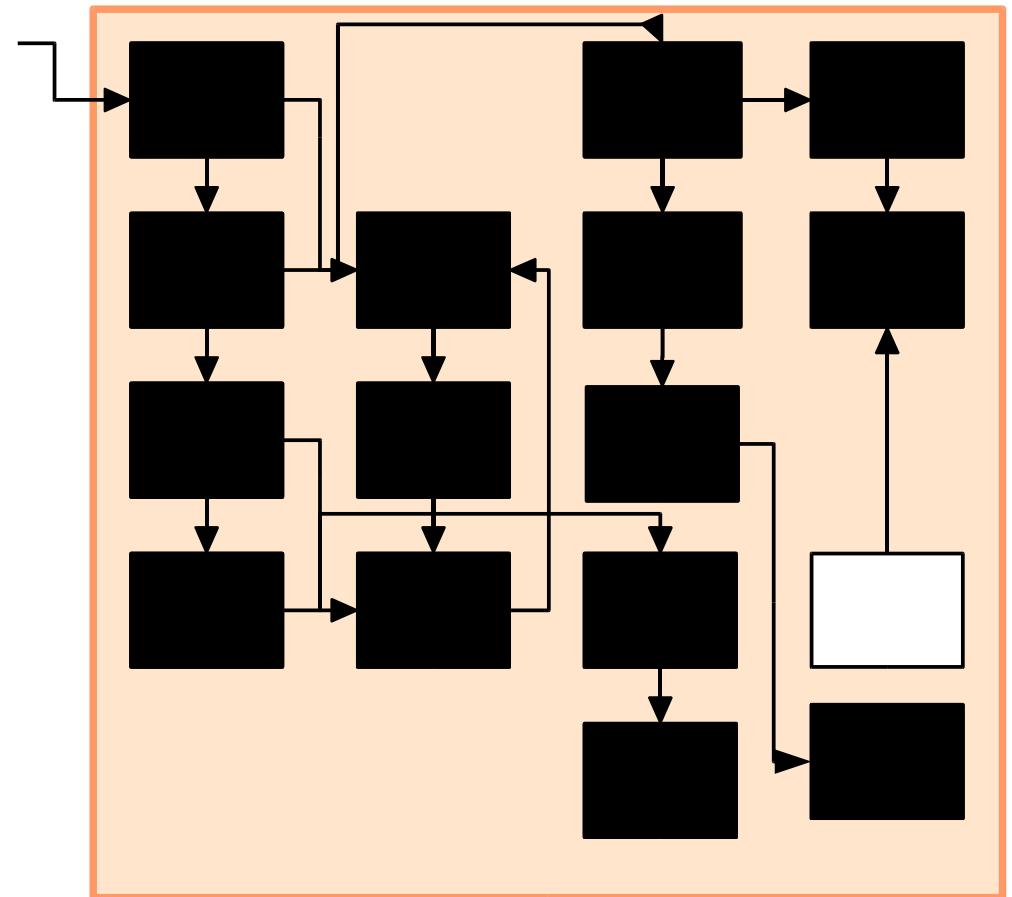
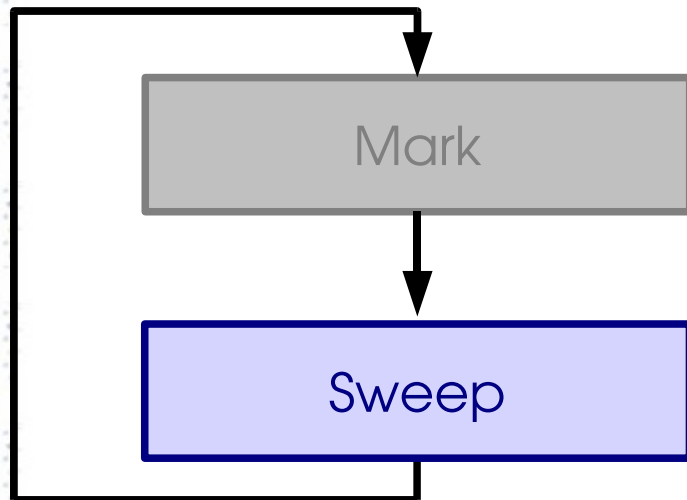
JamaicaVM realtime GC



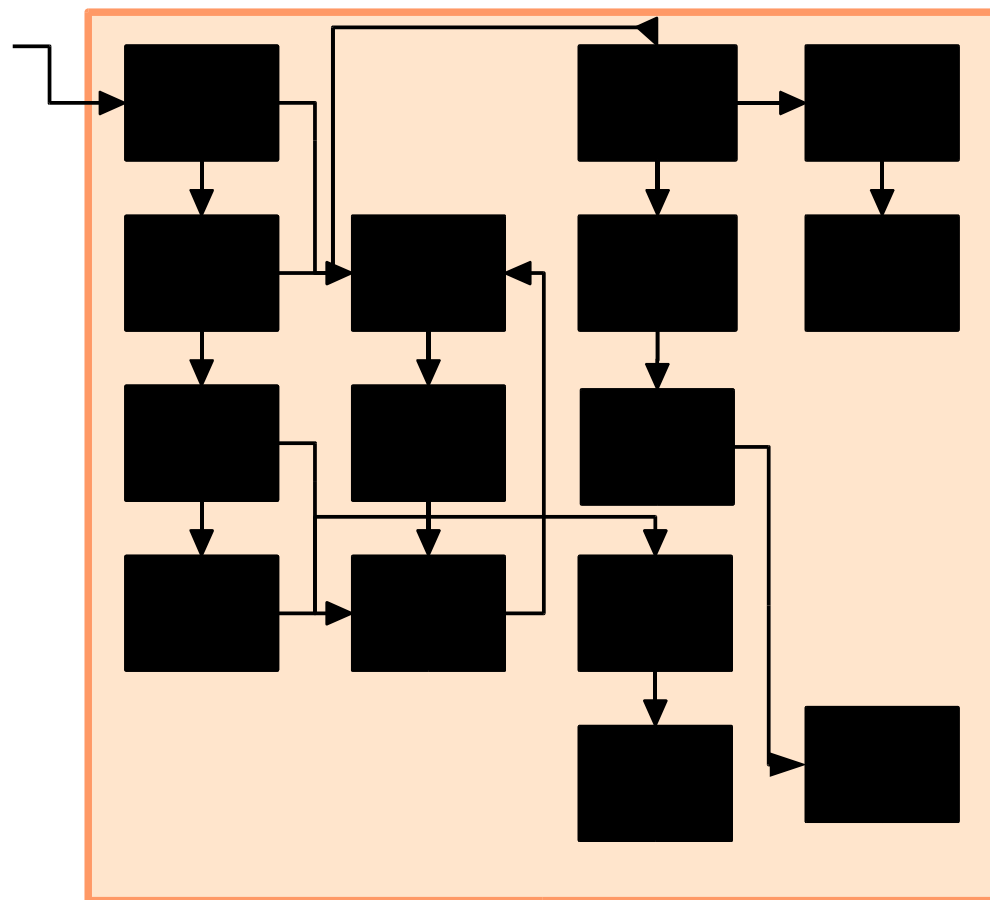
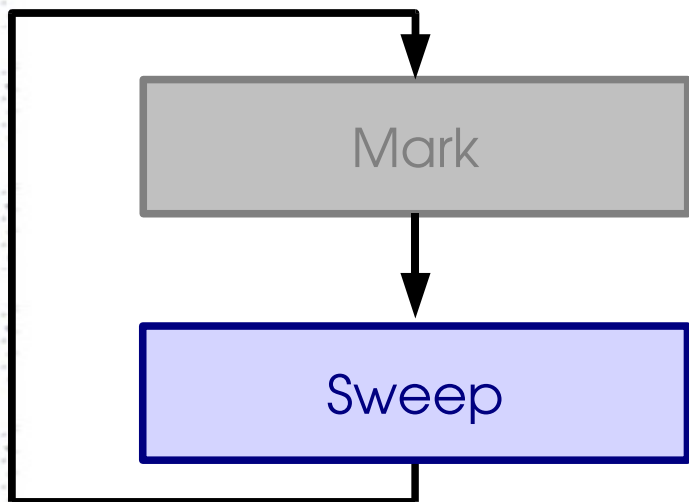
JamaicaVM realtime GC



JamaicaVM realtime GC



JamaicaVM realtime GC



JamaicaVM realtime GC

Features

- very uniform atomic GC work units: marking or sweeping a single 32 byte block
- typically 1 μ sec WCET for GC work unit
- GC cycle linear in amount of allocated memory a (max. $2 \cdot a$ GC work units)

GC activation

GC performs work on allocation only

- allocating thread pays by some units of GC work for each block allocated
- allocation rate does not need to be determined

If reachable memory is limited, there is an

- upper bound for GC work on allocation, and a
- guarantee for GC to reclaim memory fast enough

Upper bound on GC work

On allocation of one unit of memory

- perform $P=1/(1-a)$ units of GC work (a = fraction of allocated memory)
- GC cycle is linear in a
- GC cycle finishes if $\sum P(a)$ exceeds a

If reachable memory is bounded $r \leq r_{max}$

- if GC cycle starts at $a > r_{max}$, the cycle will reclaim garbage $g \geq r_{max} - a$

➔ there is an a_{max}

➔ there is a P_{max}

Limitations of earlier GC implementation

Only one non-expandable heap space allowed

- heap size needs to be set by user
- there is no good default

Write-barrier performance is critical

- required to inform GC on changes in graph
- marked objects stored in linked list for fast access

Old Heap Layout

Blocks of 32 bytes each

0	1	2	3	4	5	
						...
Colour (1 word each)						
colour blk 0	colour blk 1	colour blk 2	colour blk 3	colour blk 4	colour blk	...
						...
References (1 bit per word, i.e., one byte per block)						
refs blk 0	refs blk 1	refs blk 2	refs blk 3	refs blk 4	refs blk !	

Old Heap Layout

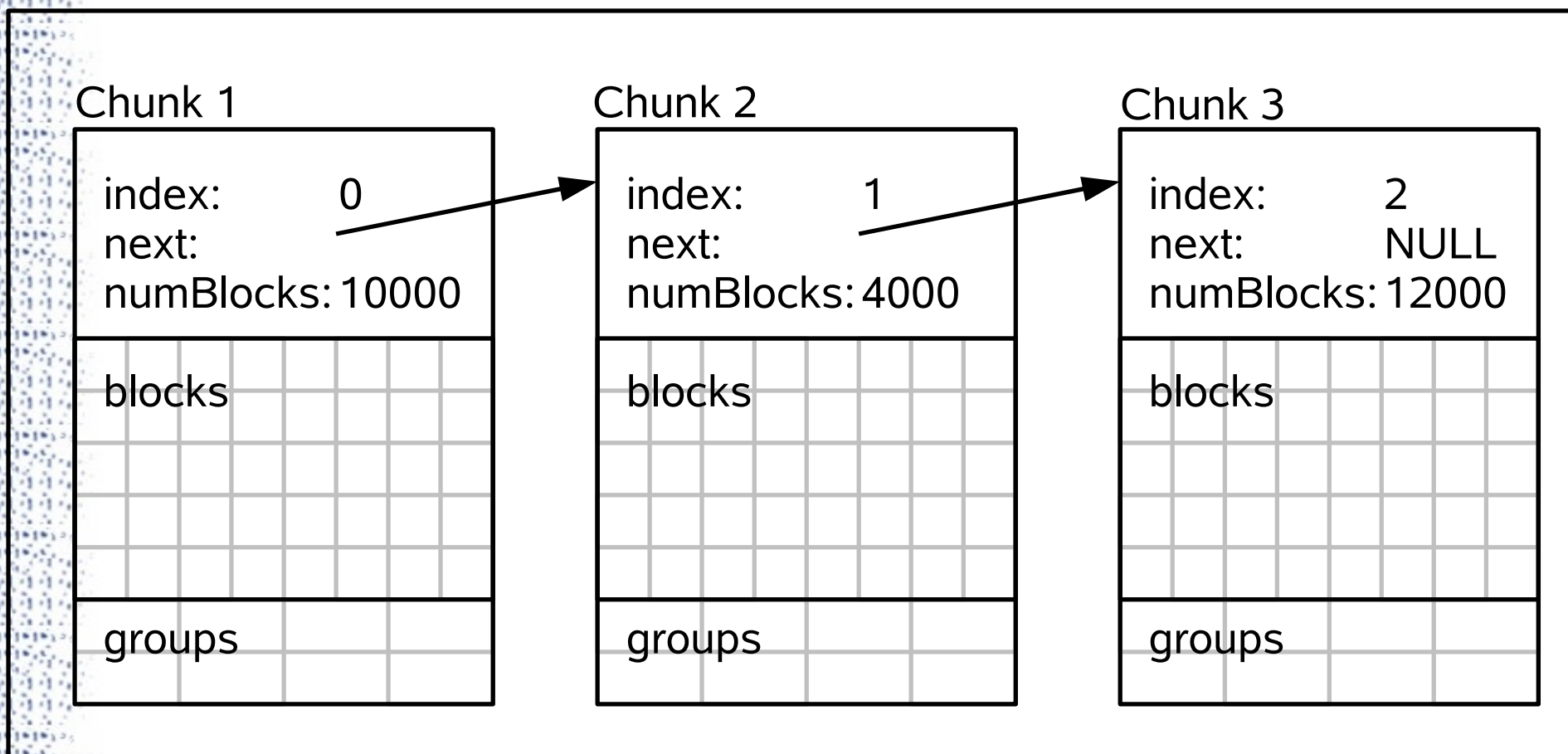
Disadvantages

- not expandable
- 12.5% overhead for colour array
- inefficient write-barrier

New heap layout should be

- expandable
- use less memory for marking
- enable efficient write barrier

New Heap Layout: List of Chunks



New Write Barrier / shading code

Two different blocks are distinguished

- “head” blocks (= Java objects)
- “inner” blocks

Head blocks

- must permit a very efficient write-barrier
- use first word for link in list of marked blocks

Inner blocks

- references not modified by Java code
- marking only required by GC mark phase code

JamaicaVM object layout

Block 1 (Header)

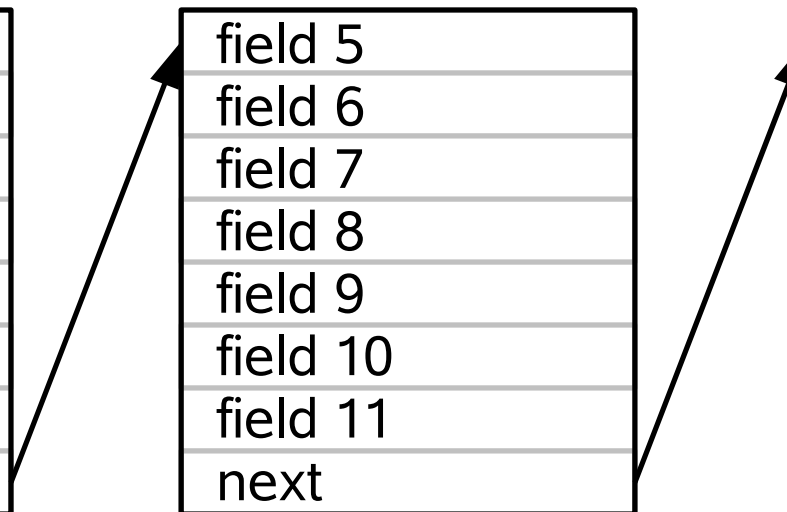
head color
type
monitor
field 1
field 2
field 3
field 4
next

Block 2

field 5
field 6
field 7
field 8
field 9
field 10
field 11
next

Block 3

field 12
field 13
field 14
field 15
—
—
—
—



JamaicaVM shading code

For head blocks:

```
if ((r != NULL) && (r->colour == white)
    {
        r->colour = grey_list;
        grey_list = r;
    }
```

- ➔ Very efficient.
- ➔ Space overhead one word per object

JamaicaVM shading code

For inner blocks:

- requires determination of chunk that contains block
- a group structure contains flags for a group of blocks
- 2 bits encode the colours *white*, *grey* and *black*
- all groups that contain at least one grey block are stored in linked list

➔ **Very space efficient.**

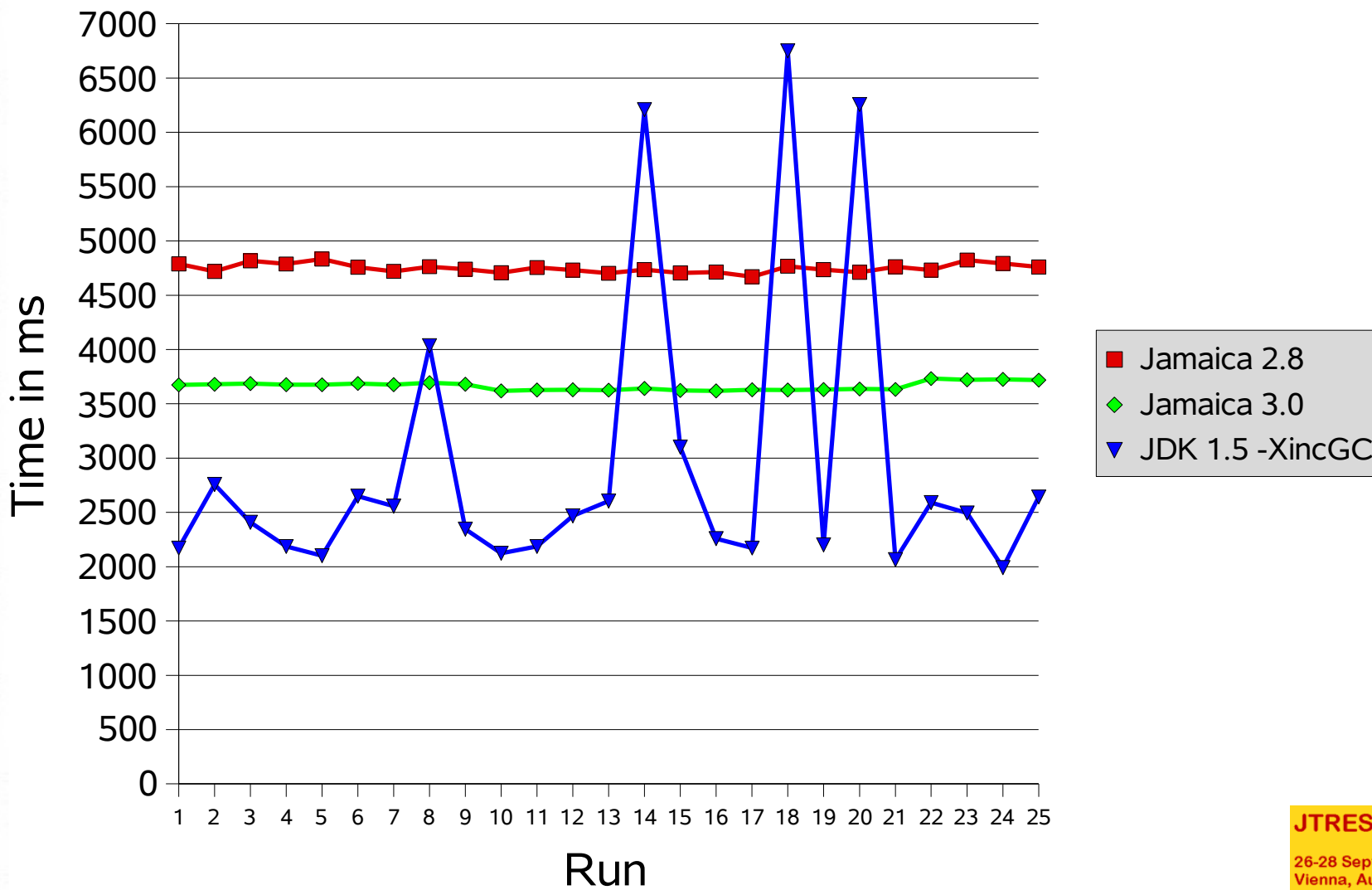
➔ **Higher but acceptable execution time**

Sweep Phase

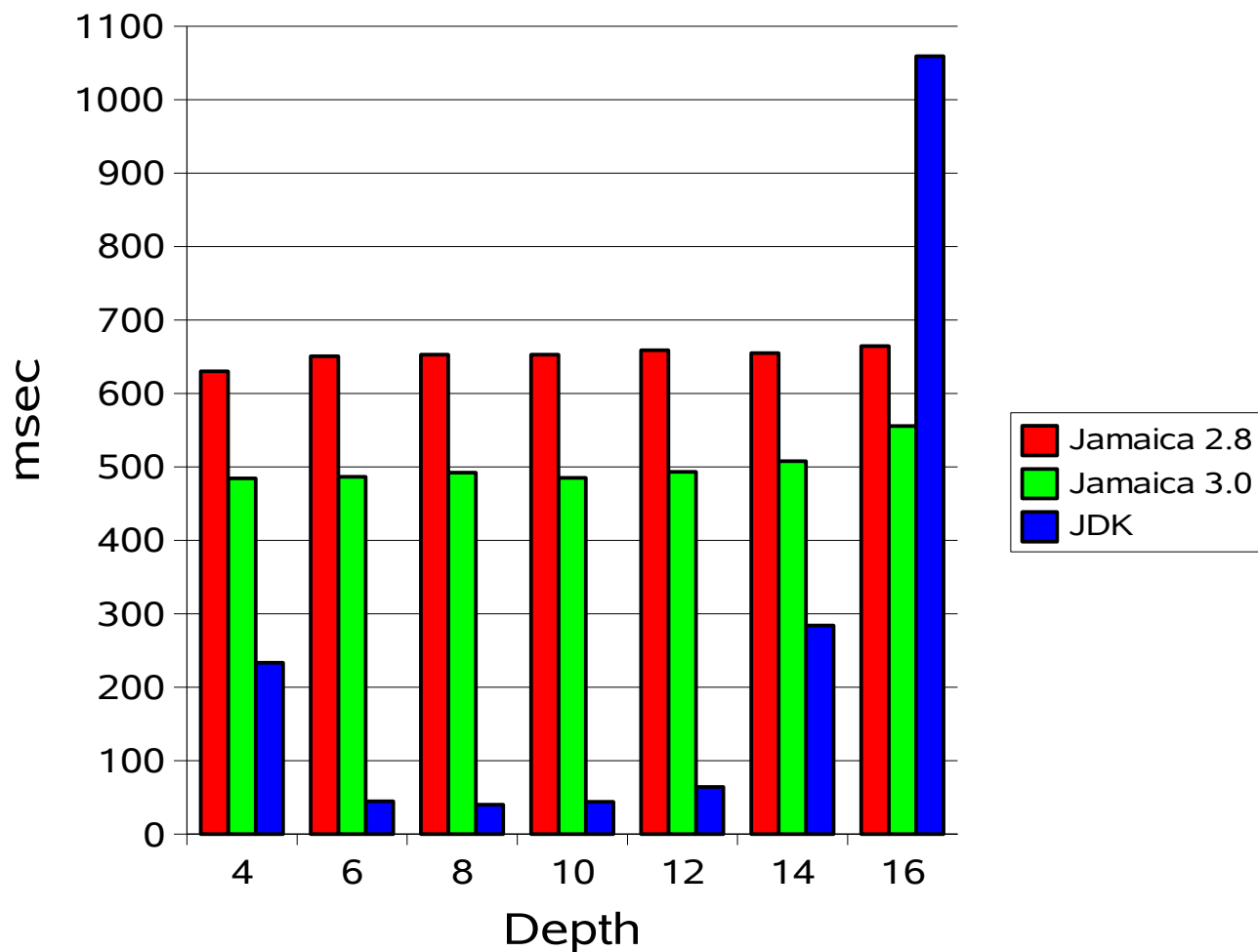
Responsible for 50% of the GC work

- no more touches blocks, only regards flags
- atomically switches meaning of *black* and *white*
- very significant performance improvement on small cache systems

Performance Comparison: GCBench



Performance Comparison: GCBench



Conclusion

Realtime Garbage Collection at a reasonable cost is possible.

In non-trivial applications, the complexity of manual memory management quickly exceeds that of a realtime garbage collector.

If robustness is desired, realtime GC brings a well tested, reused memory management.