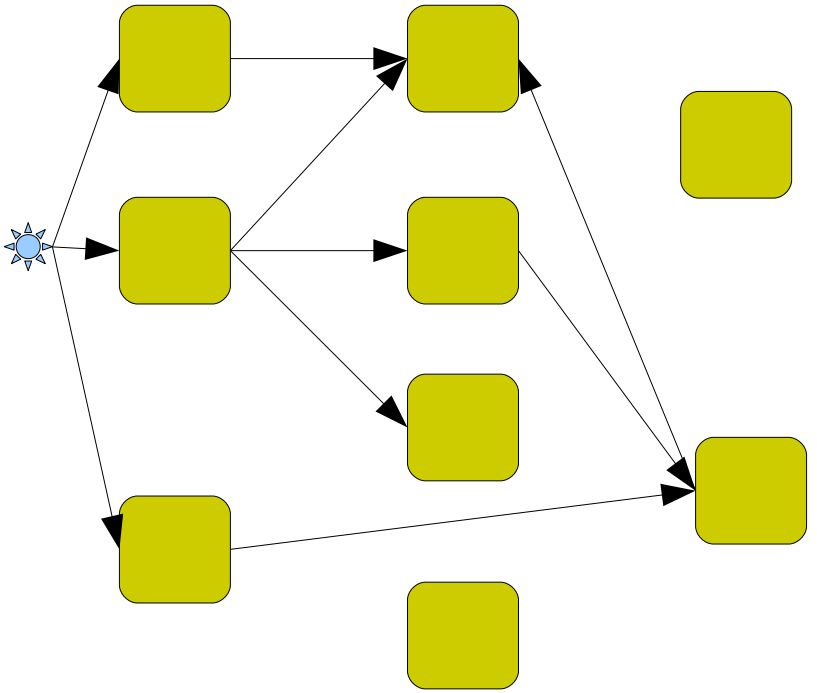
Look Ma, MT-GC in Smalltalk!

Javier Burroni

gera

MarkAndCompactGC

```
collect
  self
    ...
    followAll;
    setNewPositions;
    compact;
    ...
```



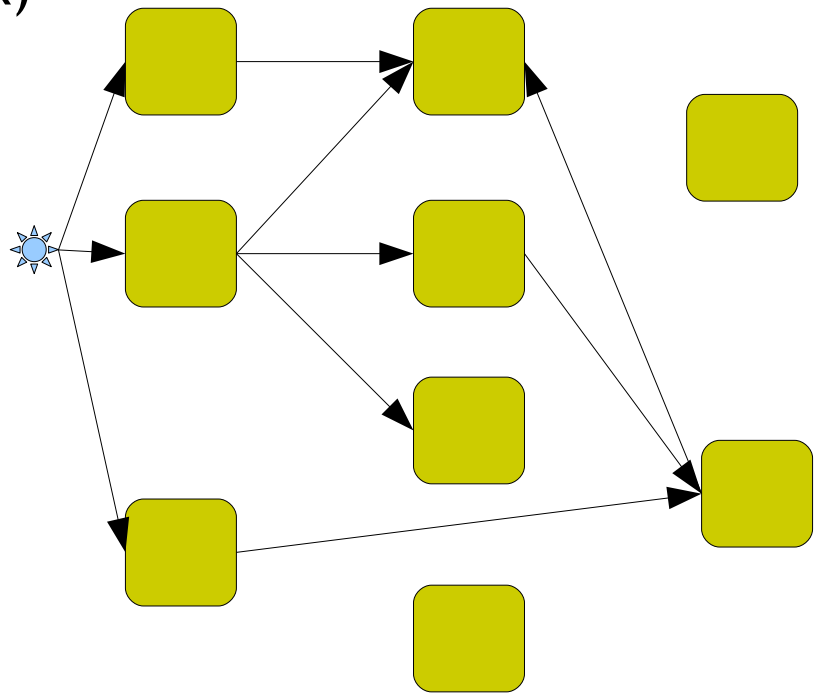
MarkAndCompactGC

#follow (mark)

```
collect
  self
    …
    followAll;
    setNewPositions;
    compact;
    …
```

now MT!

now MT!

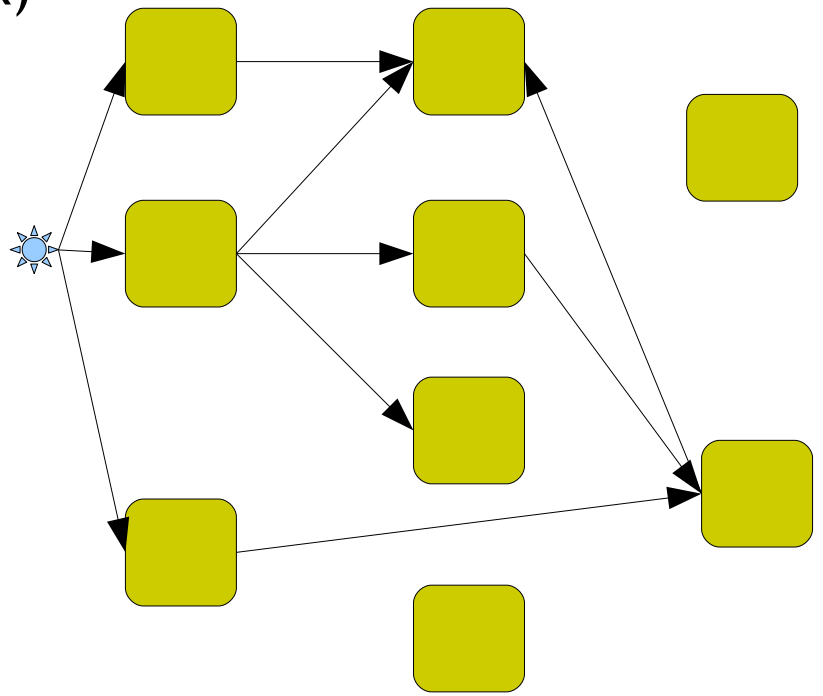


- Stop the World GC
  - All VM threads stop
  - Many threads take care of GC
  - Lock-free algorithms (as possible)

MarkAndCompactGC

#follow (mark)

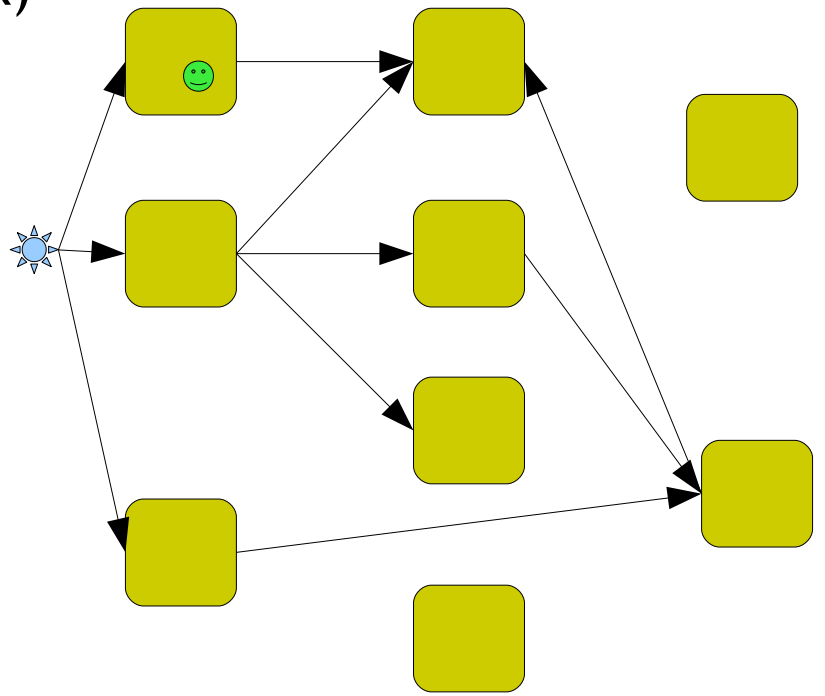
```
followWellKnownObjects
  self
    follow: self wellKnownRoots
    count: self wellKnownRootsSize
    startingAt: 1
```



MarkAndCompactGC

#follow (mark)

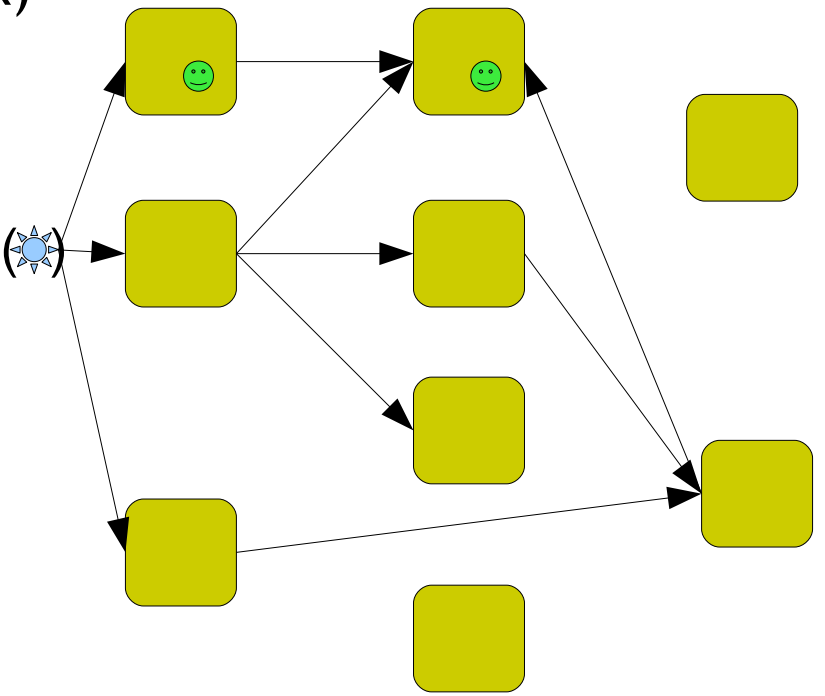
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
           object _hasBeenSeenInSpace ifFalse: [
              object _threadWith: objects at: index
              self
                 follow: object
                 count: object size
                 startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

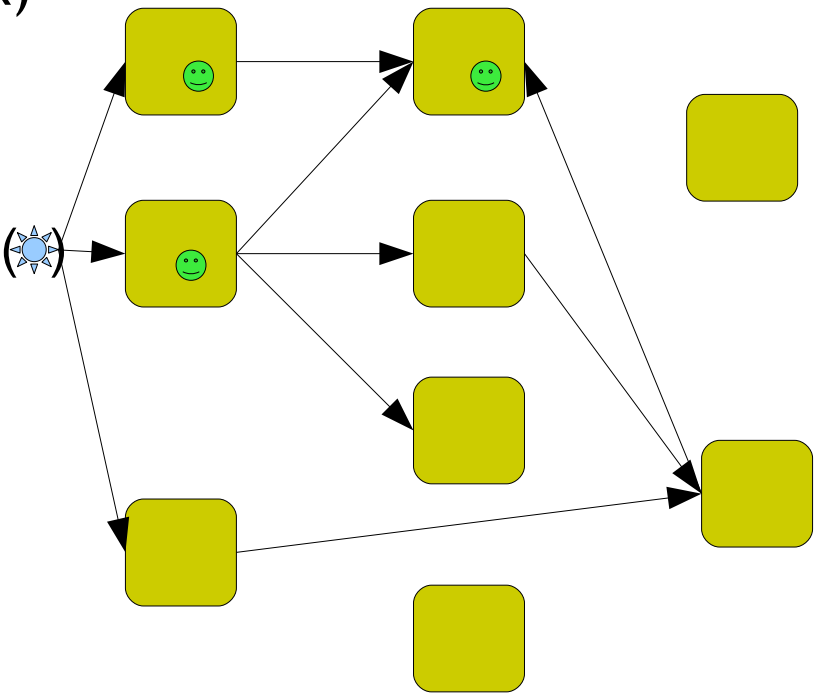
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
           object _hasBeenSeenInSpace ifFalse: [
              object _threadWith: objects at: index
              self
                 follow: object
                 count: object size
                 startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

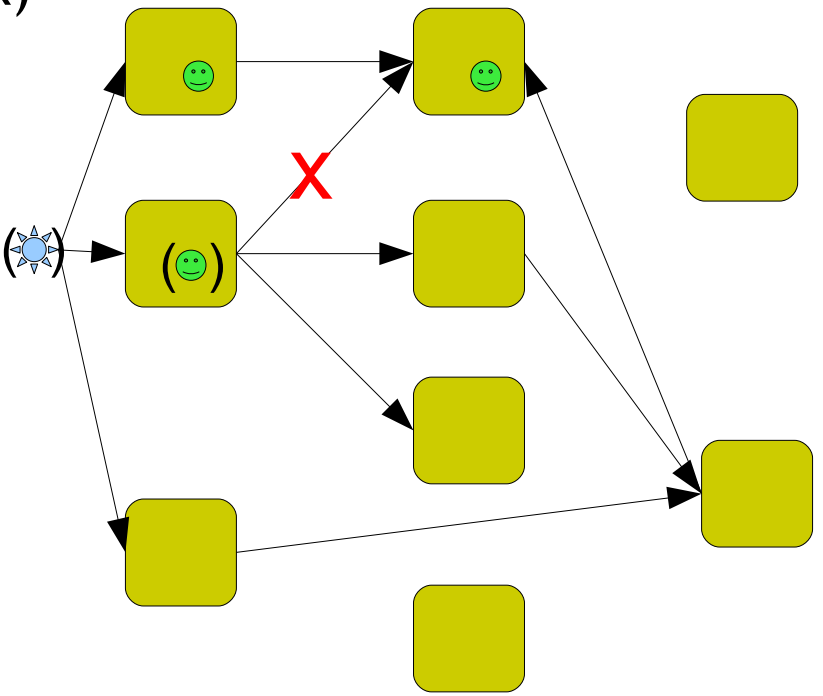
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
    object := objects at: index.
    object _isSmallInteger ifFalse: [ ...
          object _hasBeenSeenInSpace ifFalse: [
            object _threadWith: objects at: index
            self
              follow: object
              count: object size
              startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

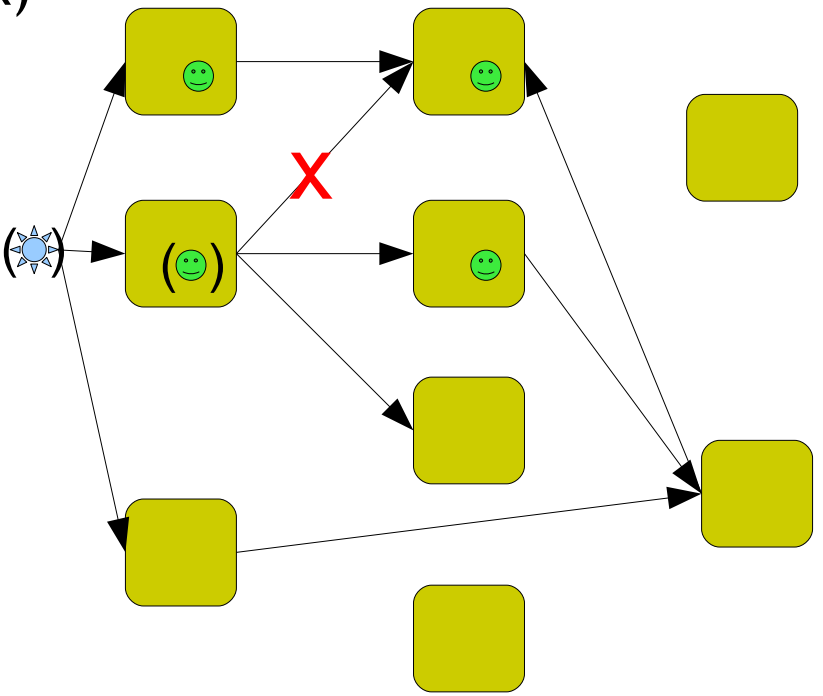
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
          object _hasBeenSeenInSpace ifFalse: [
             object _threadWith: objects at: index
             self
                follow: object
                count: object size
                startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

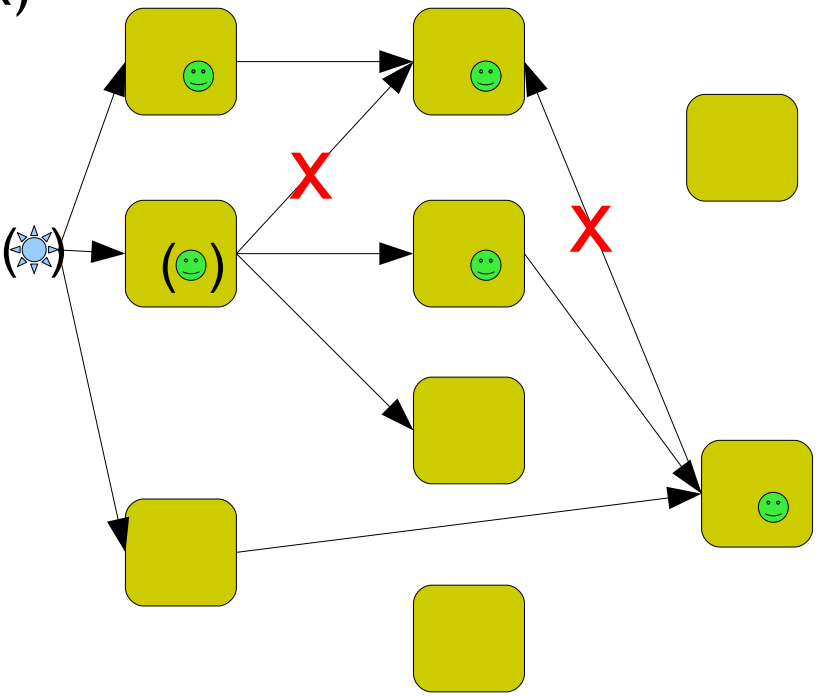
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
           object _hasBeenSeenInSpace ifFalse: [
              object _threadWith: objects at: index
              self
                 follow: object
                 count: object size
                 startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

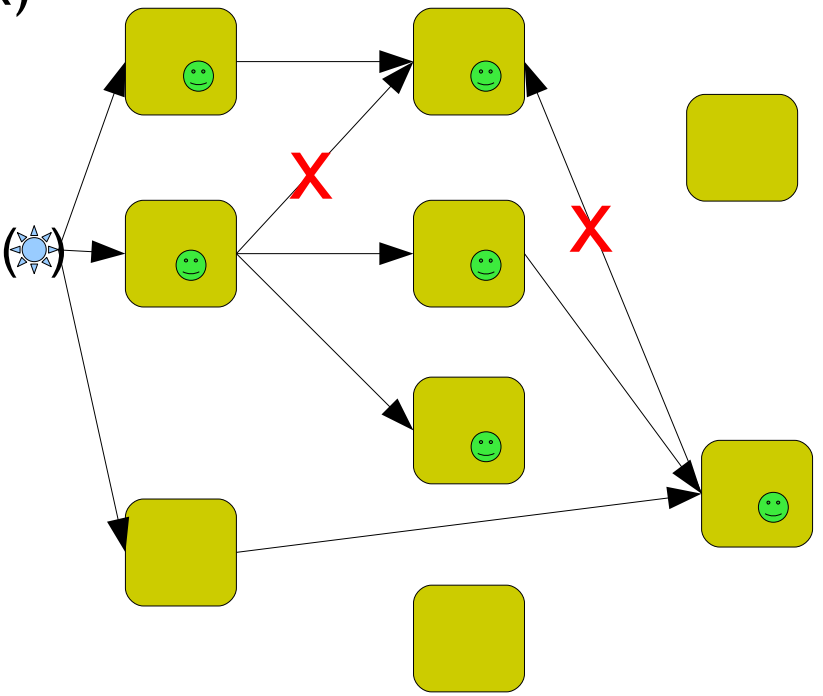
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
          object _hasBeenSeenInSpace ifFalse: [
             object _threadWith: objects at: index
             self
                follow: object
                count: object size
                startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

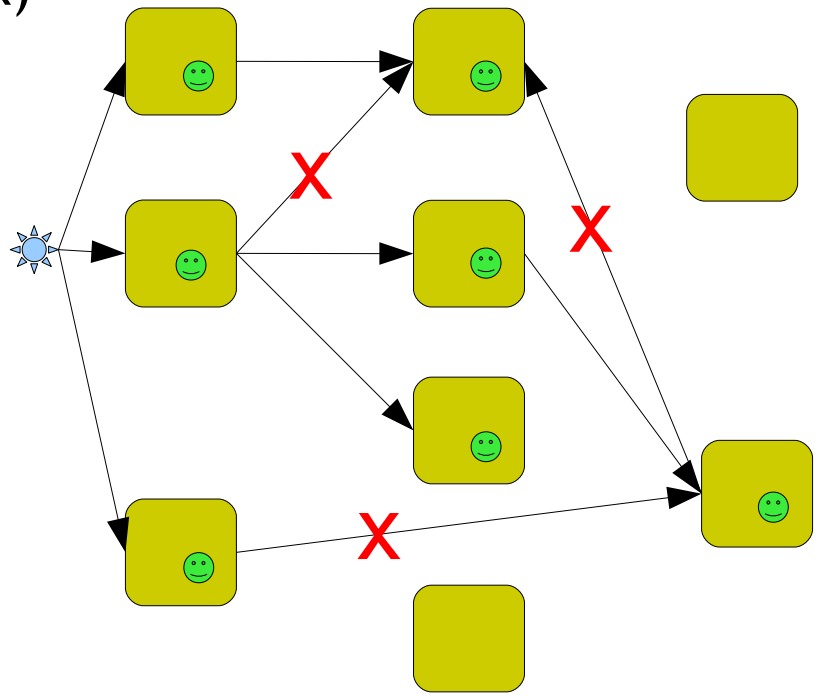
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
           object _hasBeenSeenInSpace ifFalse: [
              object _threadWith: objects at: index
              self
                 follow: object
                 count: object size
                 startingAt: 1]]].
```



MarkAndCompactGC

#follow (mark)

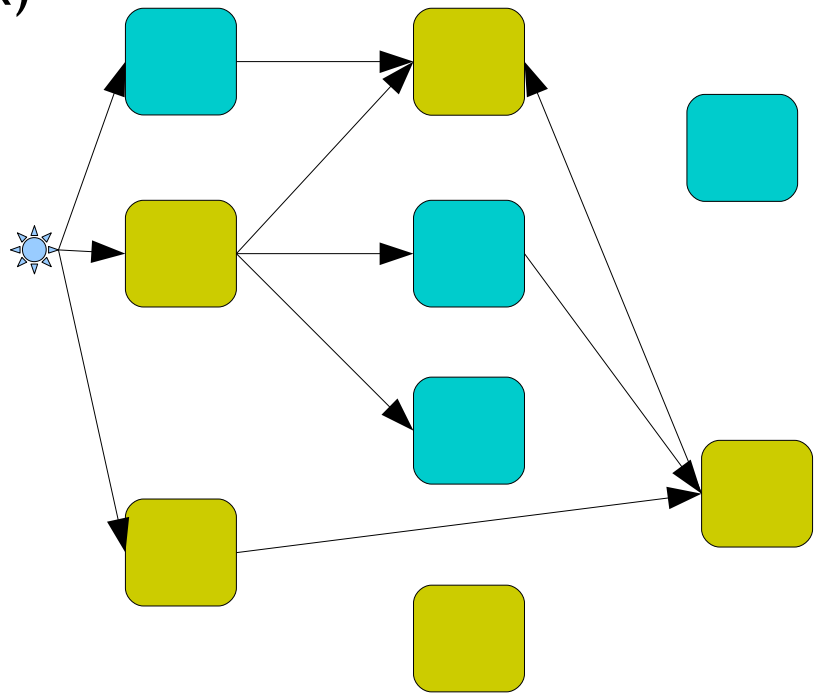
```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
     object := objects at: index.
     object _isSmallInteger ifFalse: [ ...
           object _hasBeenSeenInSpace ifFalse: [
              object _threadWith: objects at: index
              self
                 follow: object
                 count: object size
                 startingAt: 1]]].
```



MTMarkAndCompactGC

#follow (mark)

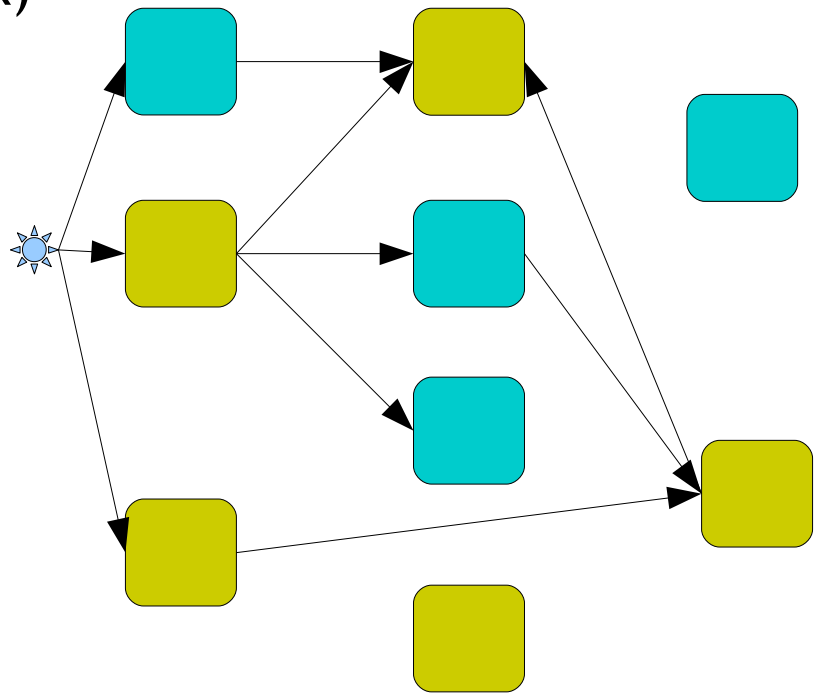
```
idOf: object
  ^(object _oop bitShift: -4) \\ followers size.
```



MTMarkAndCompactGC

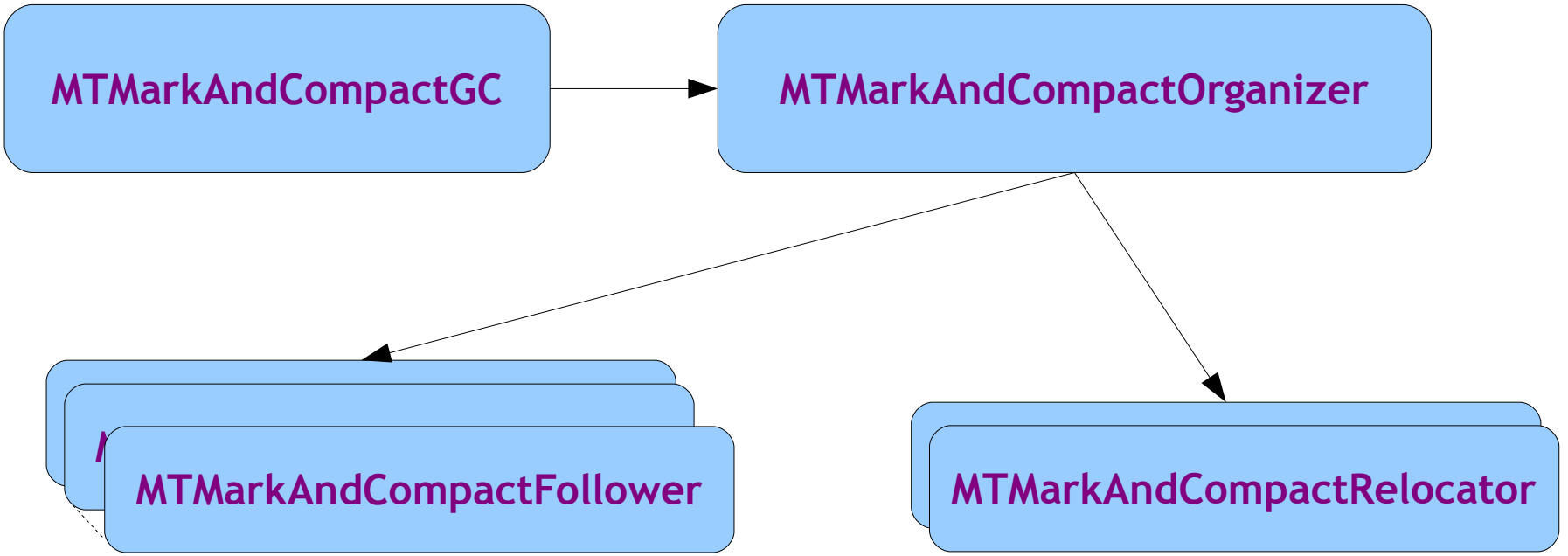
#follow (mark)

```
follow: objects count: size startingAt: base
  base to: size to: [:index | object |
    object := objects at: index.
    object _isSmallInteger ifFalse: [
        organizer
            follow: object
            from: objects
            at: index]]
```



MTMarkAndCompactGC

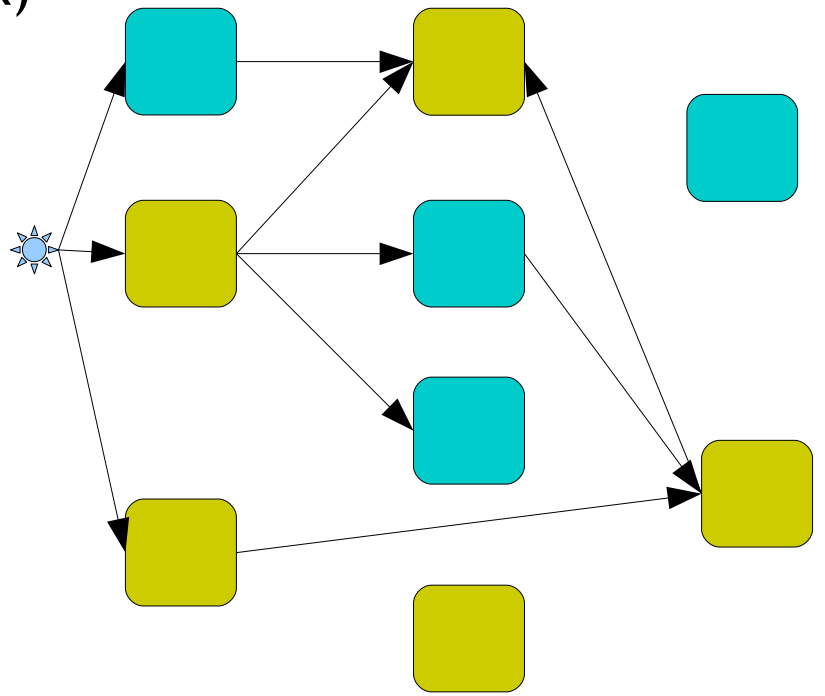
#follow (mark)



MTMarkAndCompactOrganizer

#follow (mark)

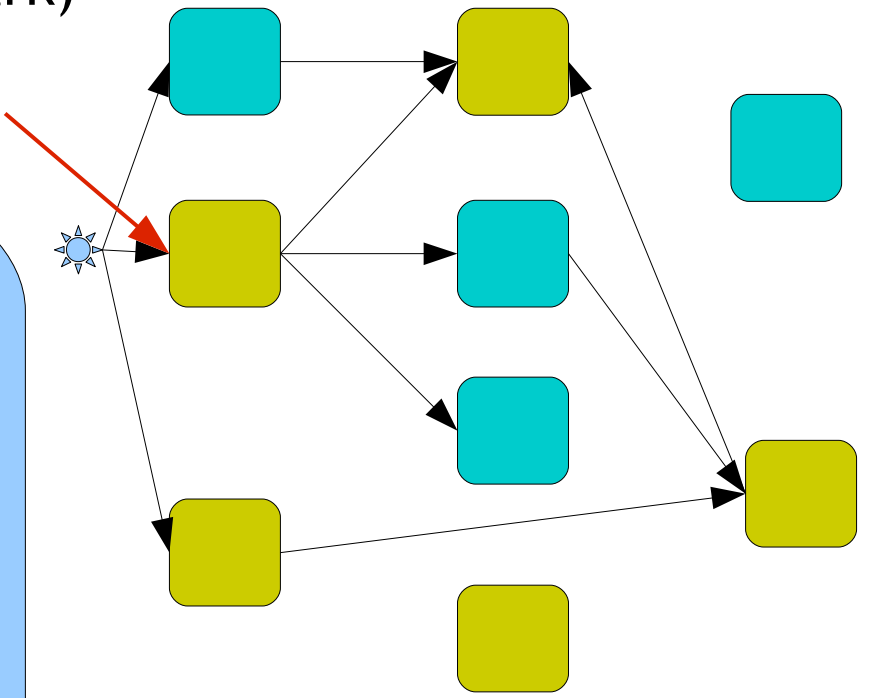
```
follow: object from: reference at: index
  | id follower |
  id := organizer idOf: object.
  follower := followers _basicAt: id + 1.
  follower
      queue: object
      from: reference
      at: index]
```



MTMarkAndCompactFollower

#follow (mark)

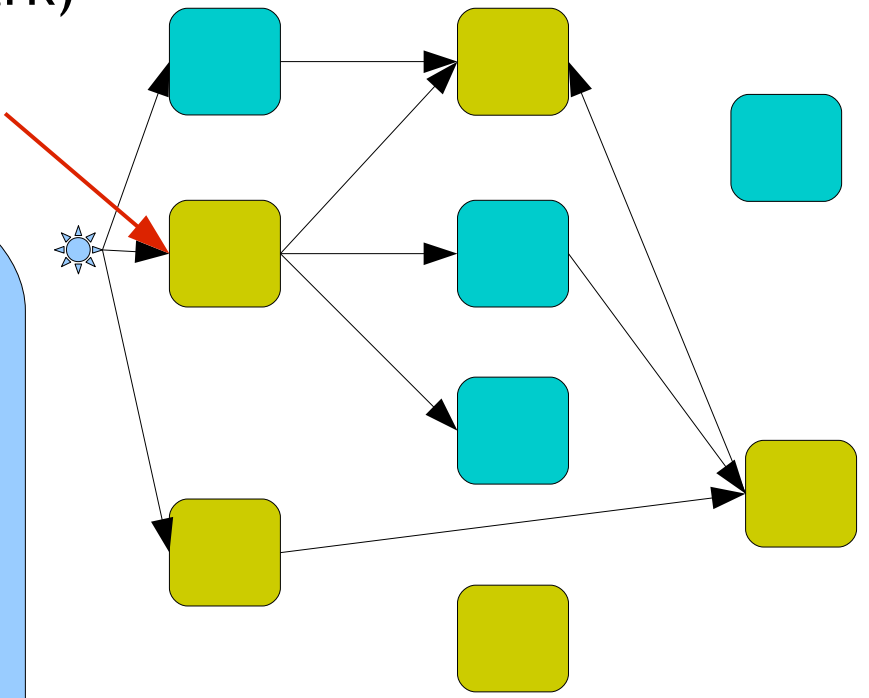
```
run
    | object |
    [queue waitNotEmpty] whileTrue: [

        [queue isEmpty] whileFalse: [
            reference := queue popFirst.
            self
                markAndFollow: object
                threadWith: reference]].
```



MTMarkAndCompactFollower

#follow (mark)

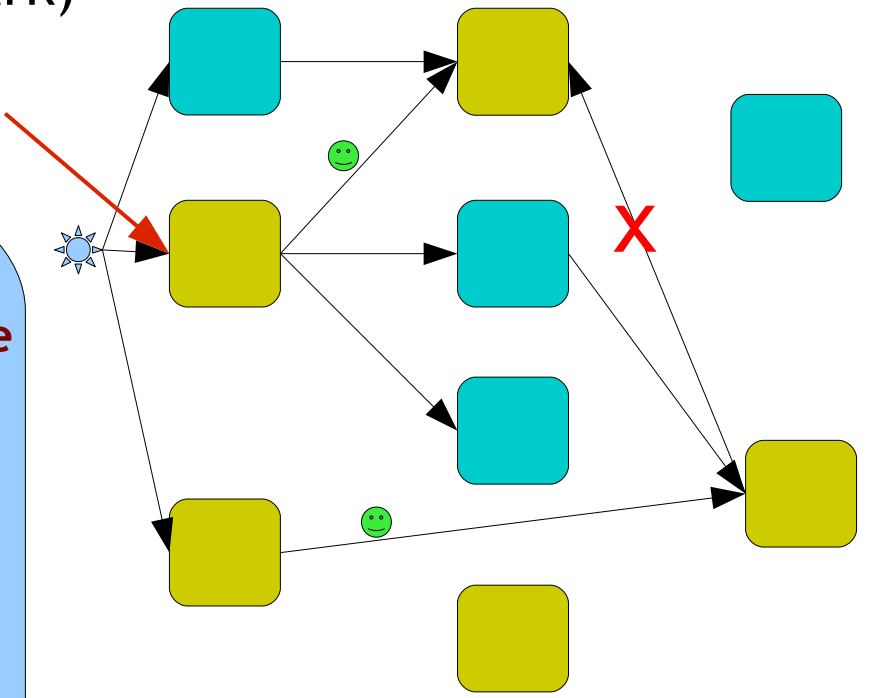
```
run
   | object |
   [queue waitNotEmpty] whileTrue: [

      [queue isEmpty] whileFalse: [
            reference := queue popFirst.
            self
               markAndFollow: object
               threadWith: reference]].
```



MTMarkAndCompactFollower

#follow (mark)

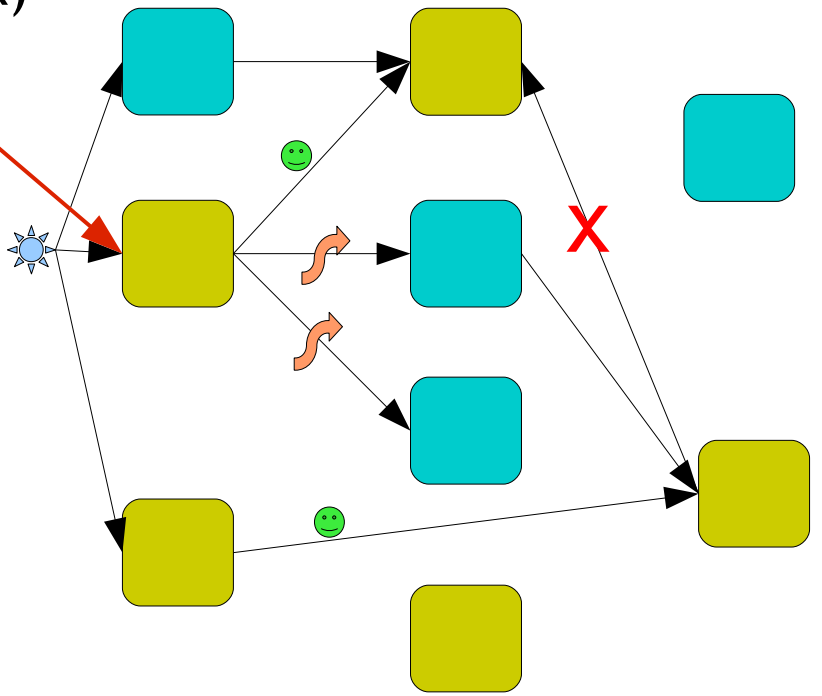
```
markAndFollow: object threadWith: reference
   object _hasBeenSeenInSpace ifFalse: [
      1 to: object size do: [:index |
        self
              follow: (object _basicAt: index)
              from: object
              at: index]].
   object _threadWith: reference.
```



MTMarkAndCompactFollower

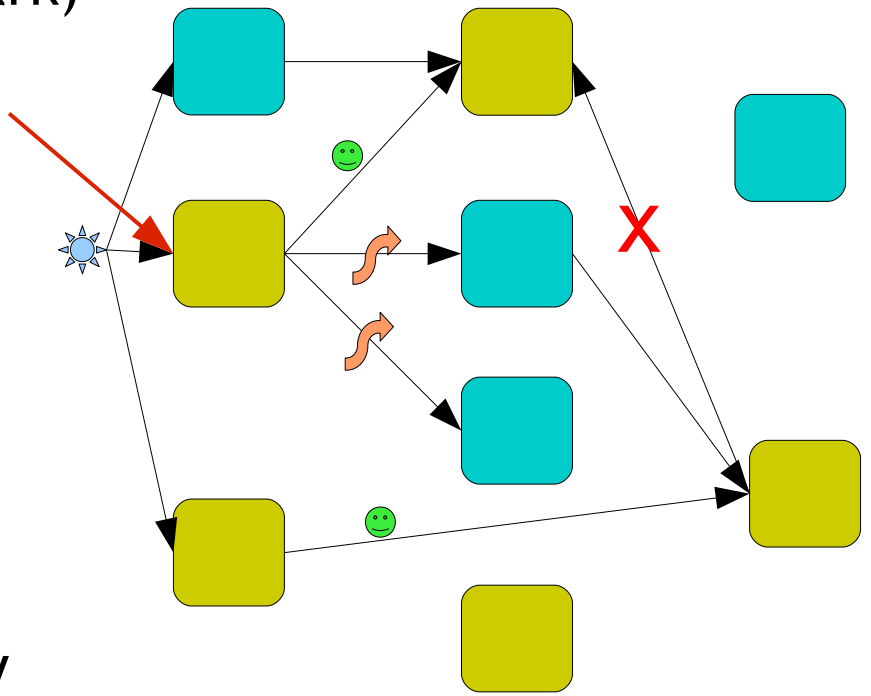
#follow (mark)

```
follow: object from: reference at: index
  | id follower |
  id := organizer idOf: object.
  Id == self id
    IfTrue: [
      self markAndThread: object
        from: reference]
    IfFalse: [
      organizer queue: object...]
```



MTMarkAndCompactFollower

#follow (mark)



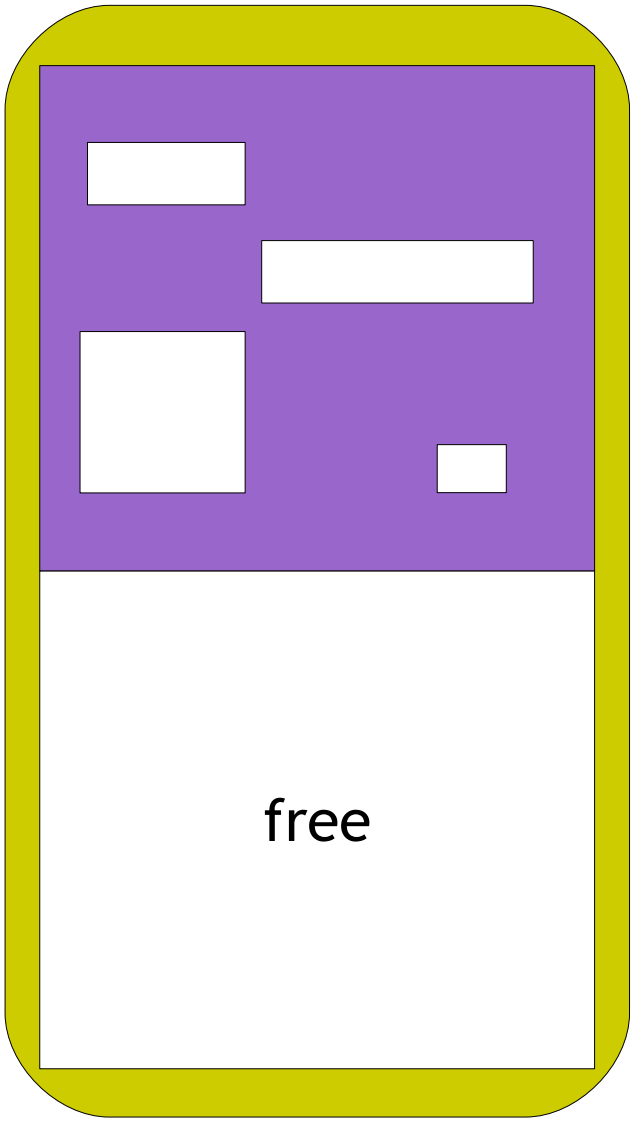
Slower with more threads

- All references are queued
- Queues have to grow
- Queueing more than necessary
- Ephemerons overhead

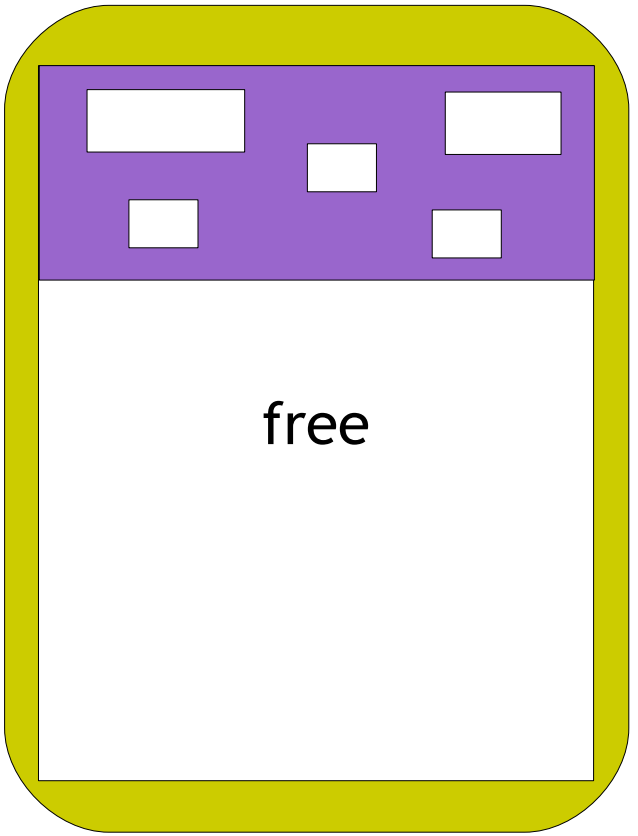
MarkAndCompactGC

#setNewPosition:

old



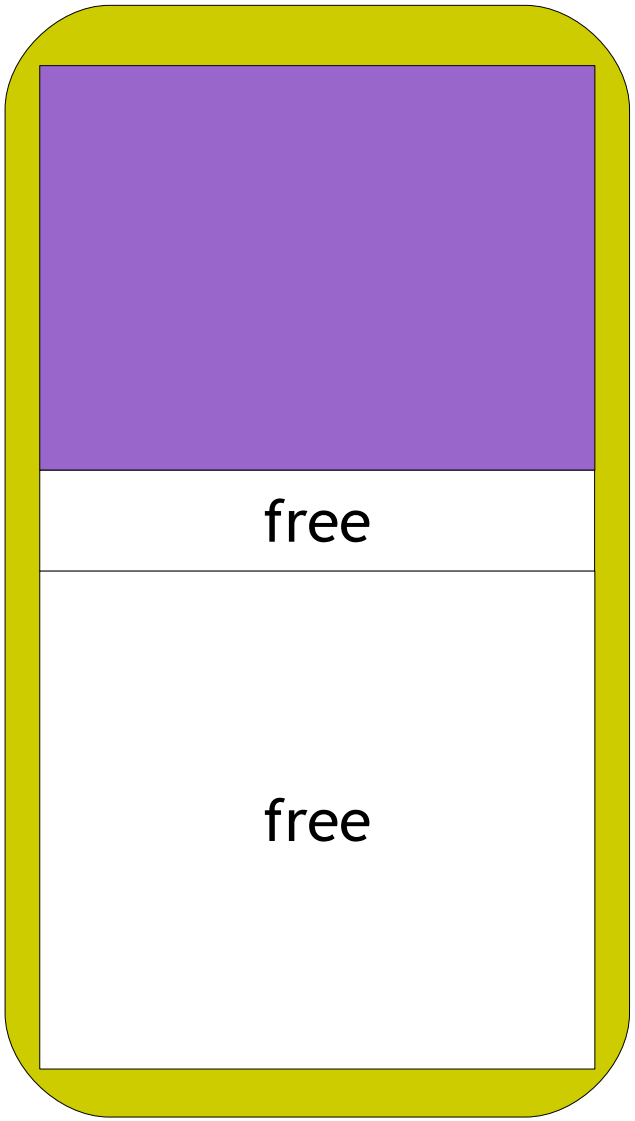
new



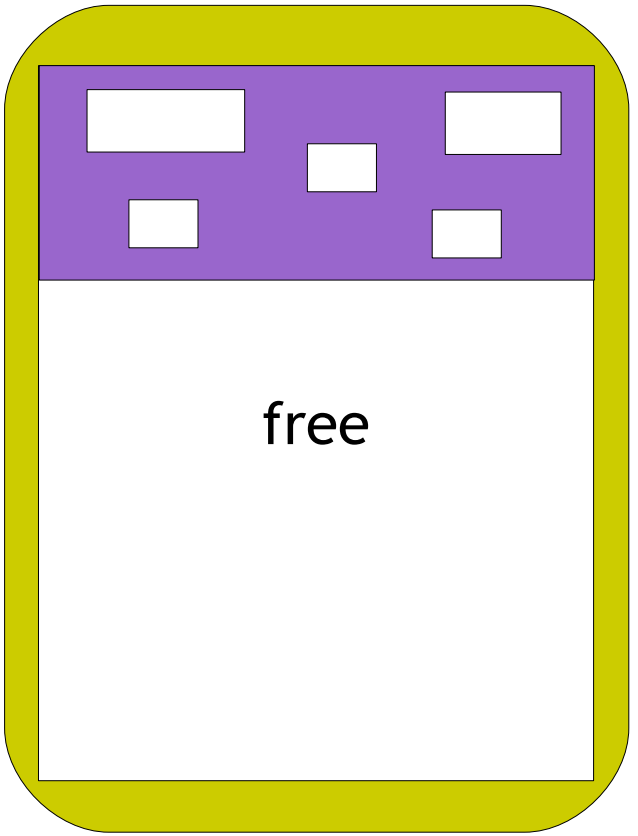
MarkAndCompactGC

#setNewPosition:

old

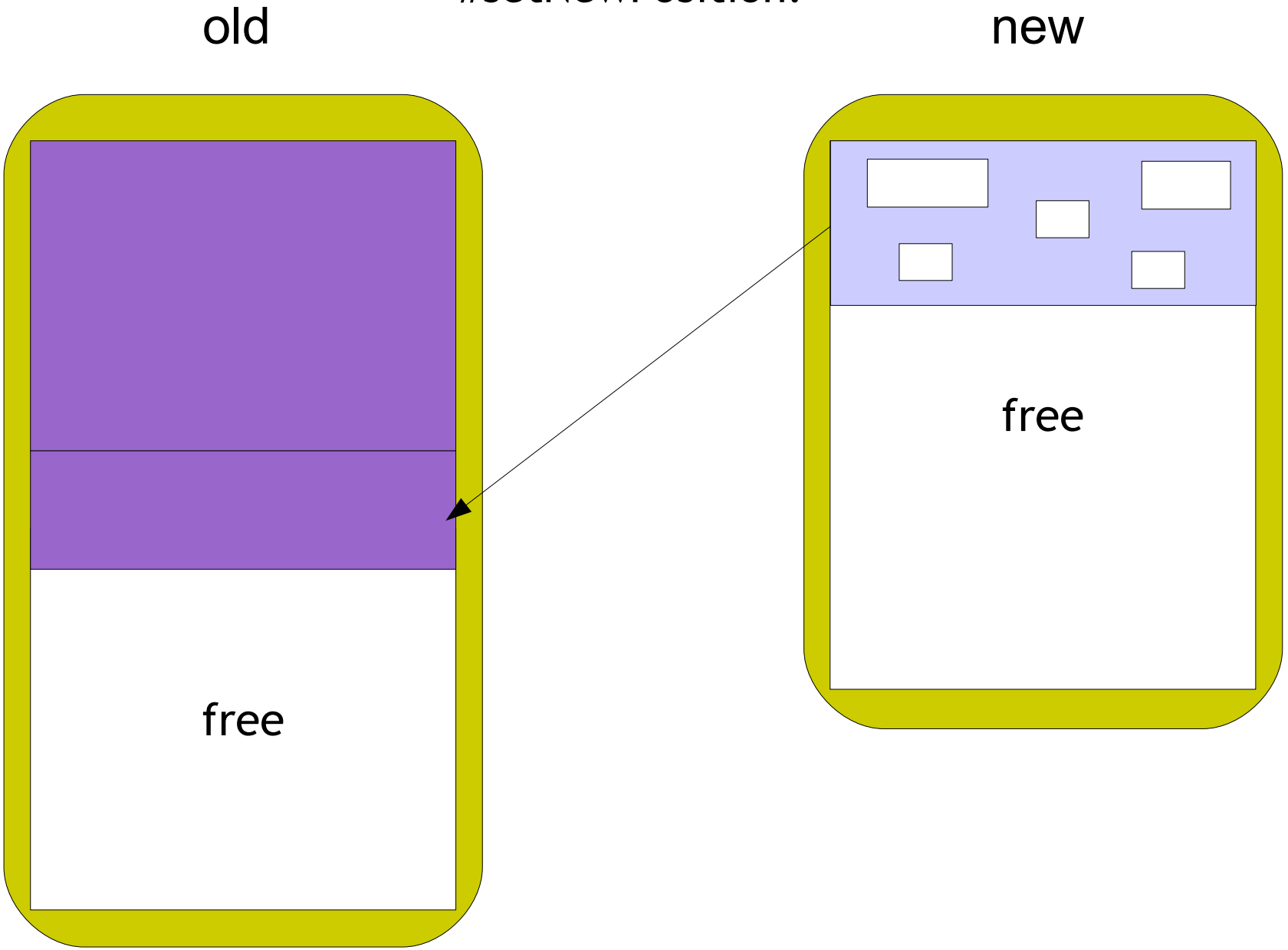


new



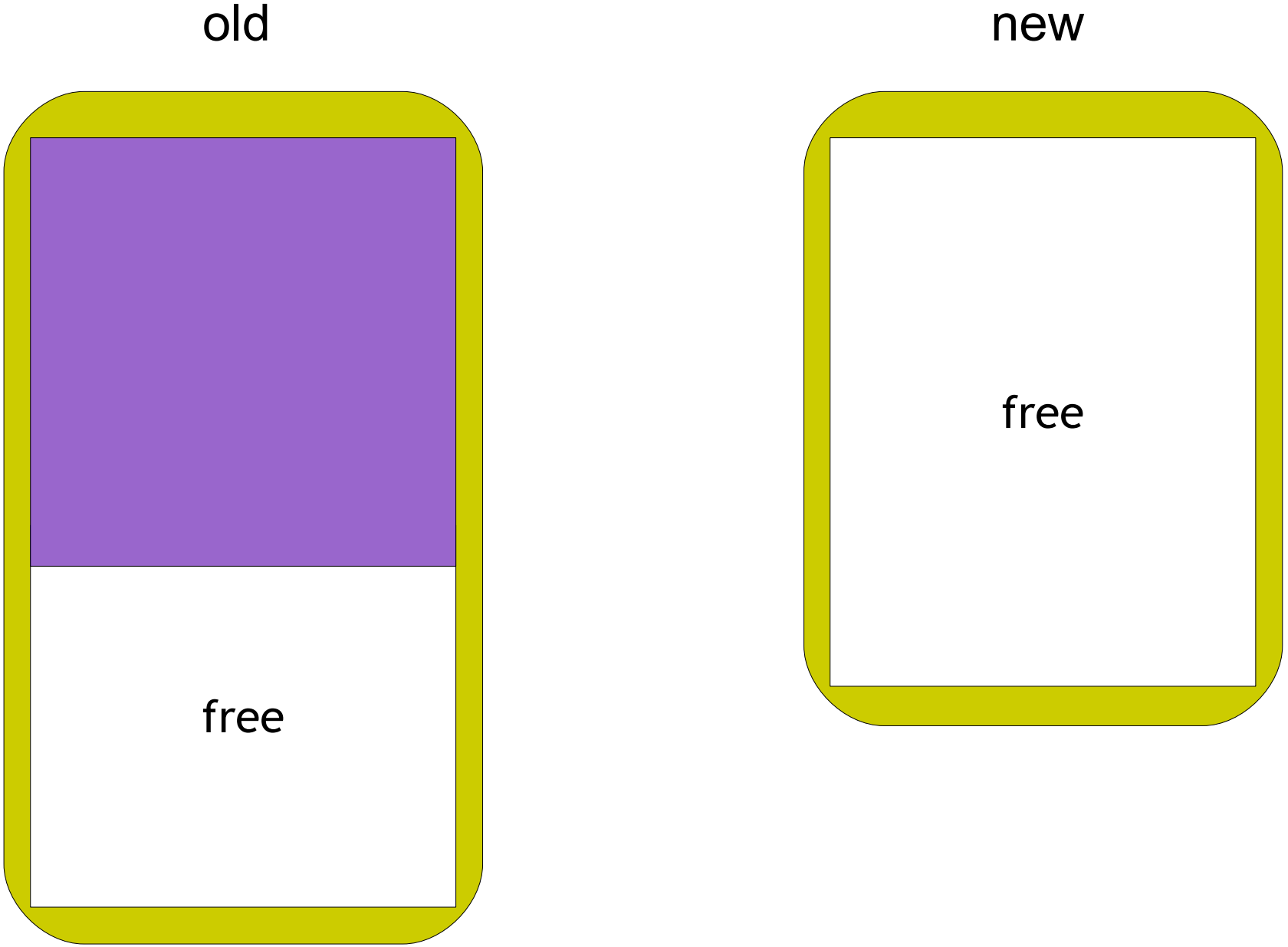
MarkAndCompactGC

#setNewPosition:



MarkAndCompactGC

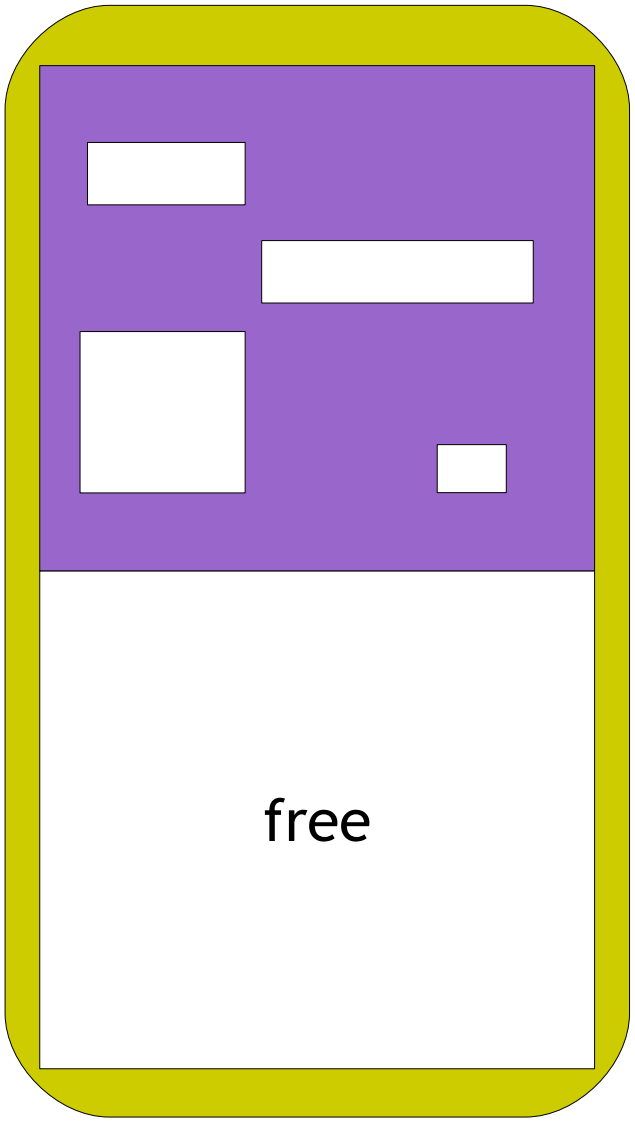
#setNewPosition:



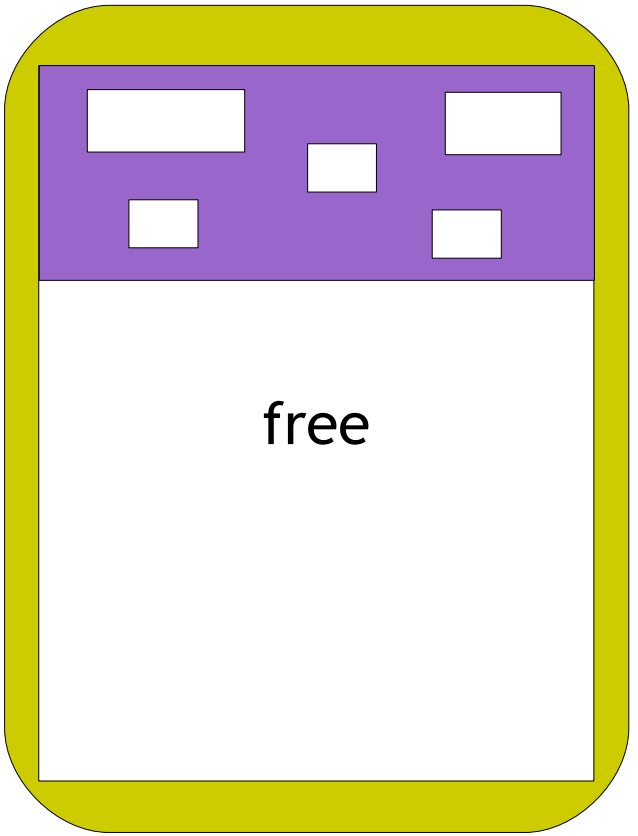
MTMarkAndCompactGC

#setNewPosition:

old



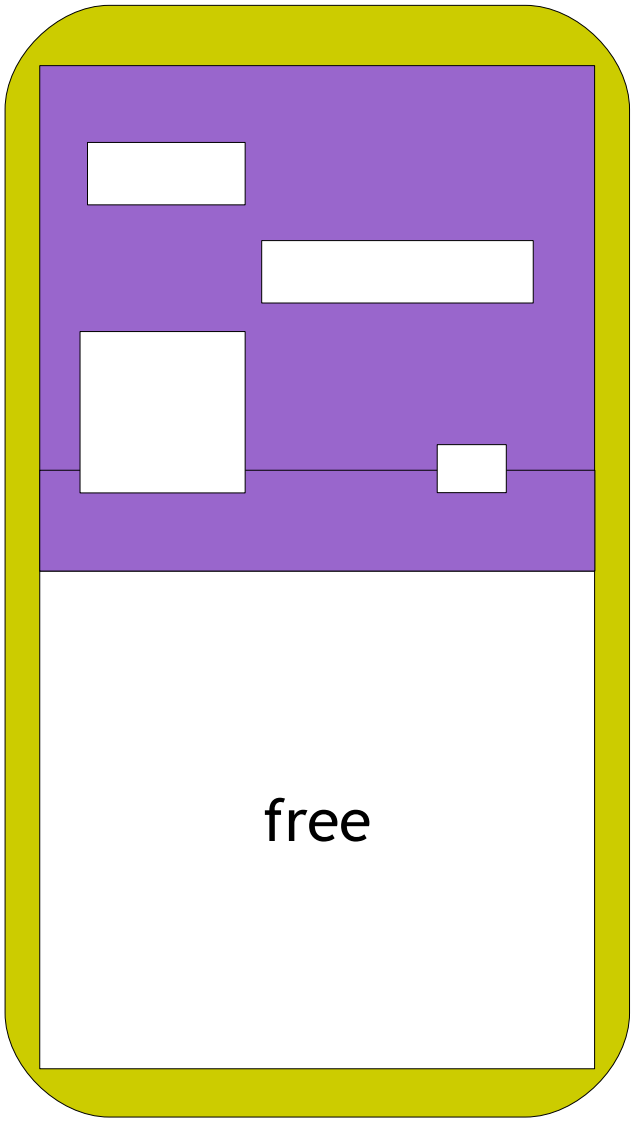
new



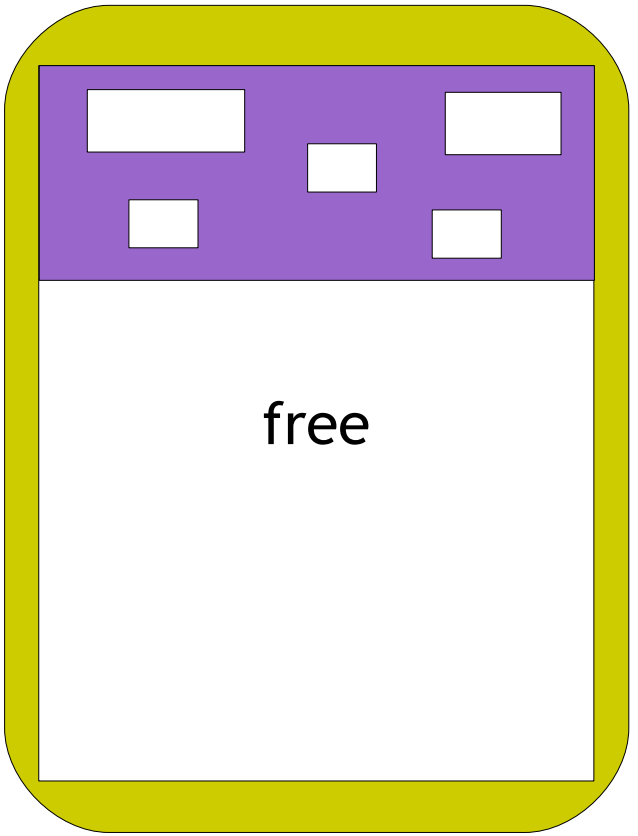
MTMarkAndCompactGC

#setNewPosition:

old

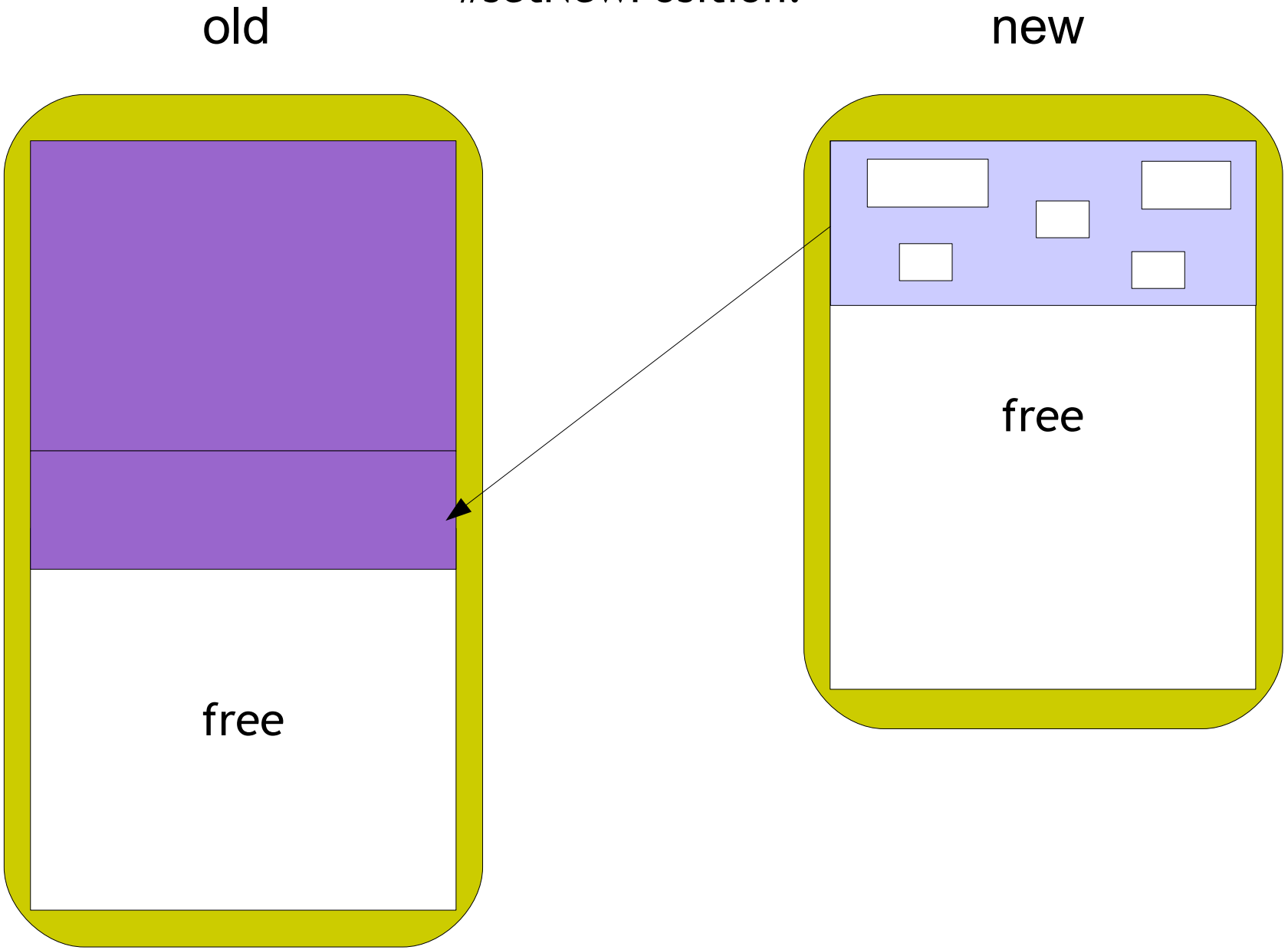


new



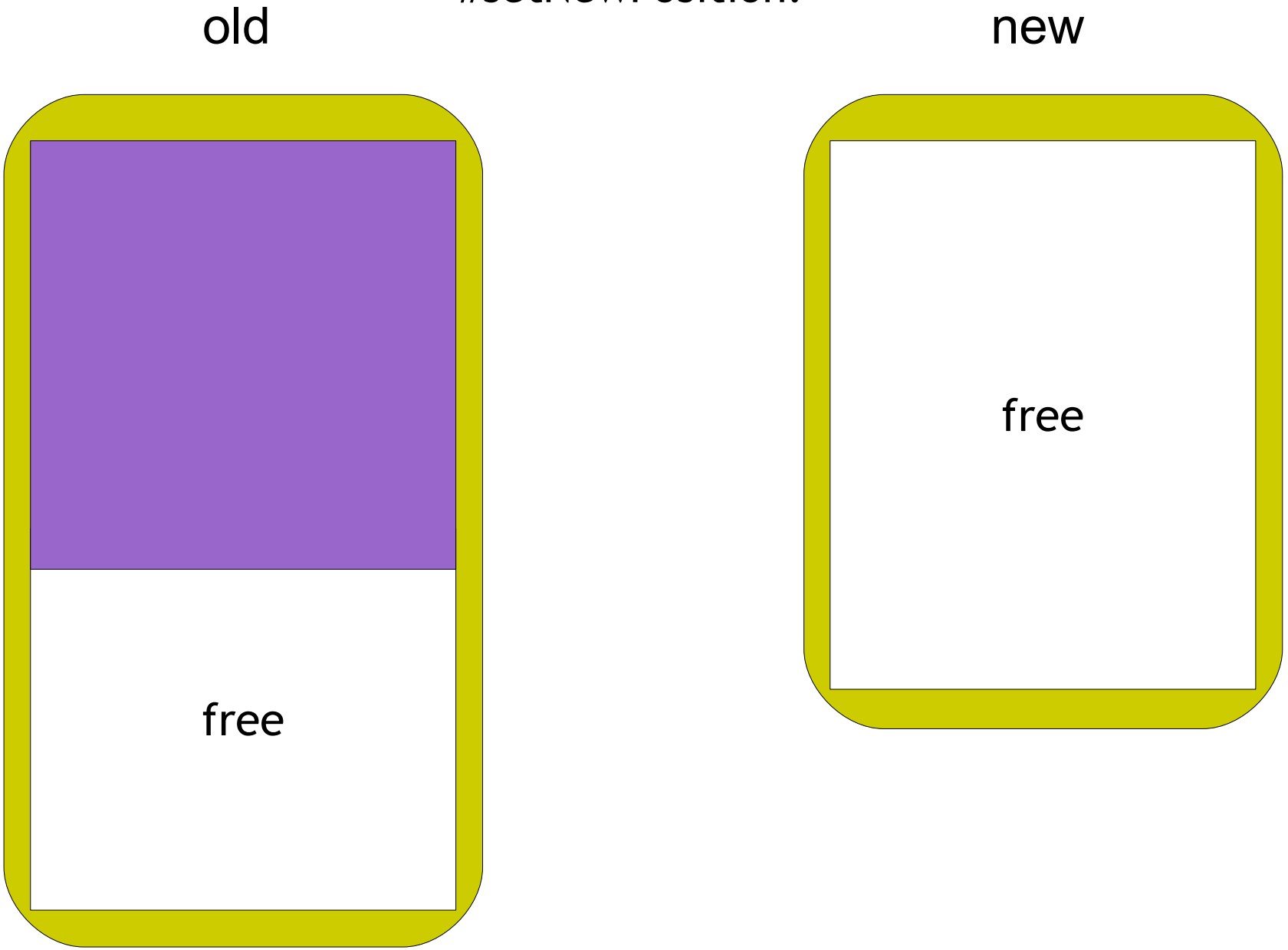
MTMarkAndCompactGC

#setNewPosition:



MTMarkAndCompactGC

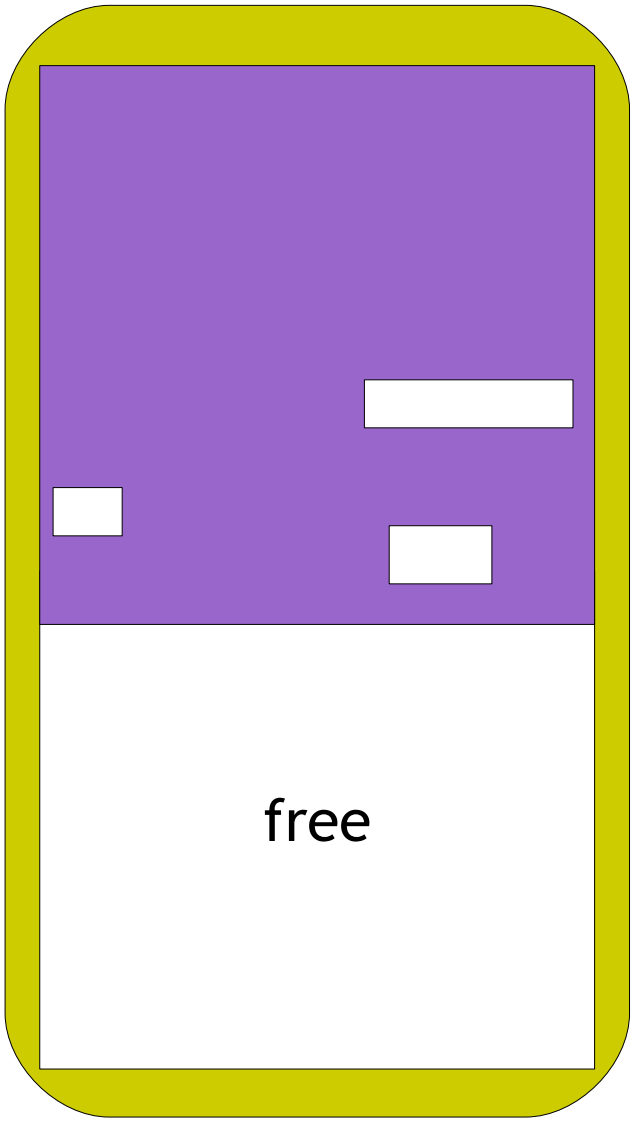
#setNewPosition:



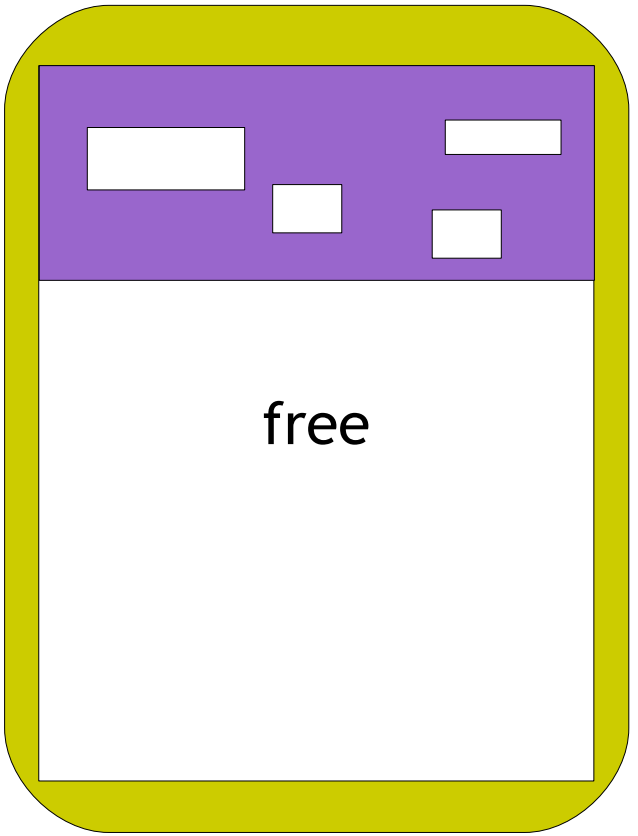
MTMarkAndCompactGC

#setNewPosition:

old

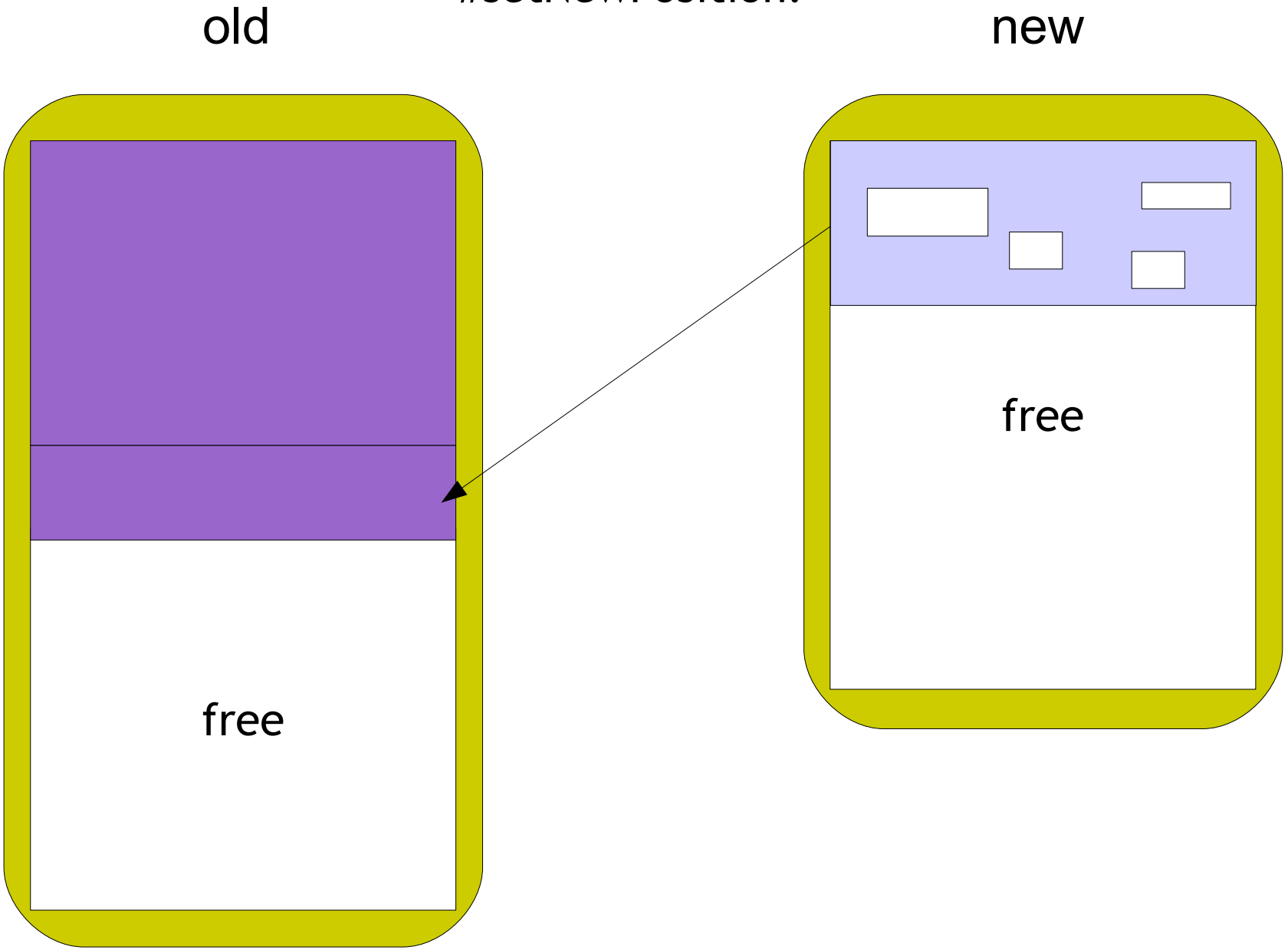


new



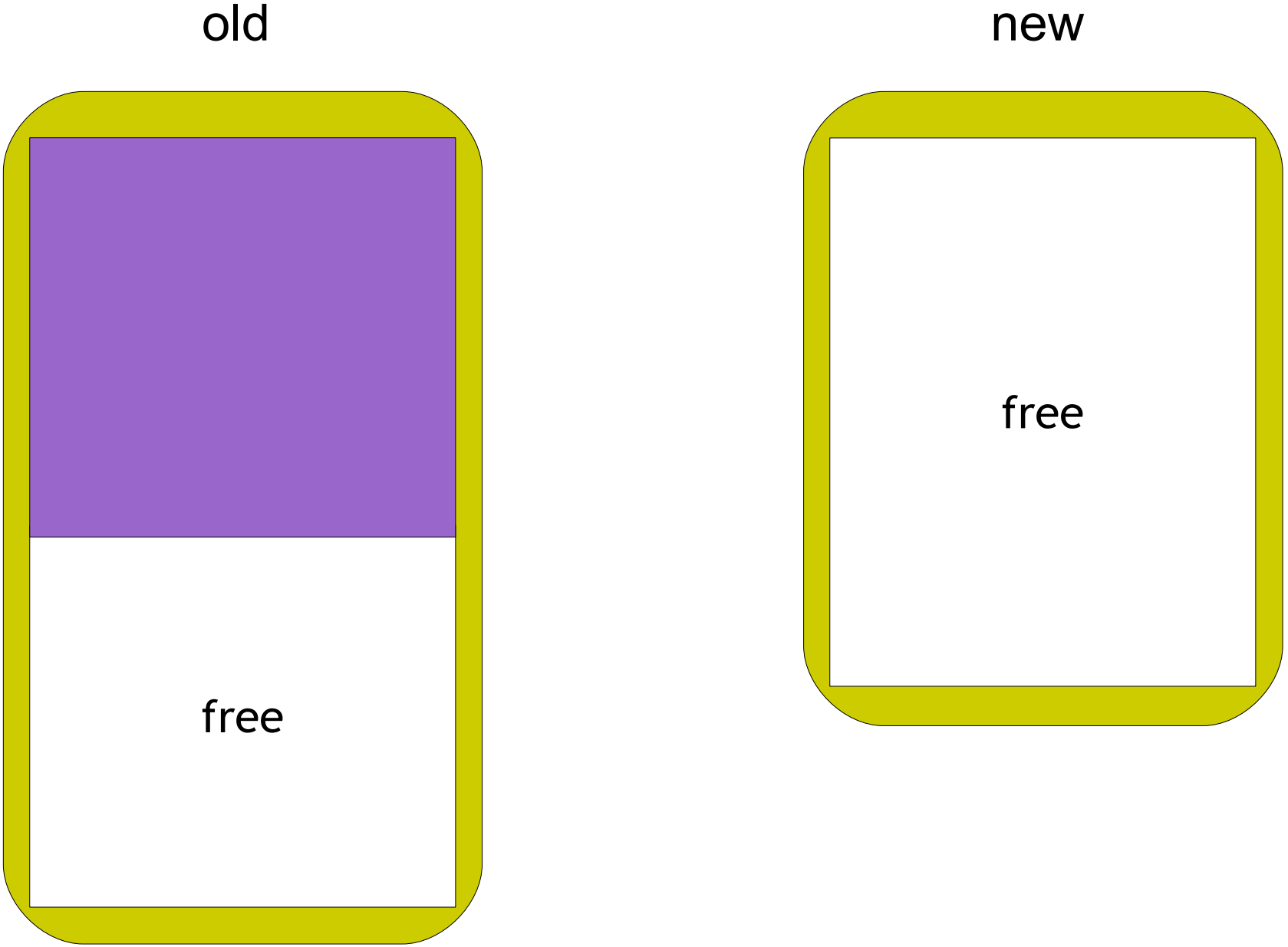
MTMarkAndCompactGC

#setNewPosition:



MTMarkAndCompactGC

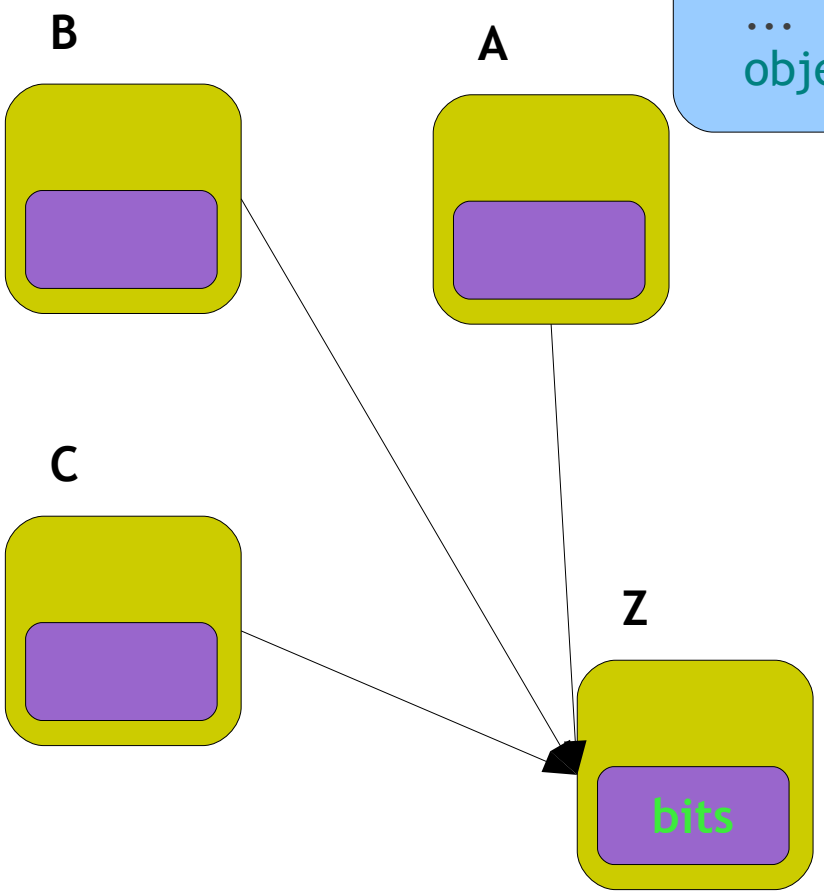
#setNewPosition:



MarkAndCompactGC

#setNewPosition:

```
follow: root count: size startingAt: base
  ...
  object _threadWith: reference at: oldIndex.
```

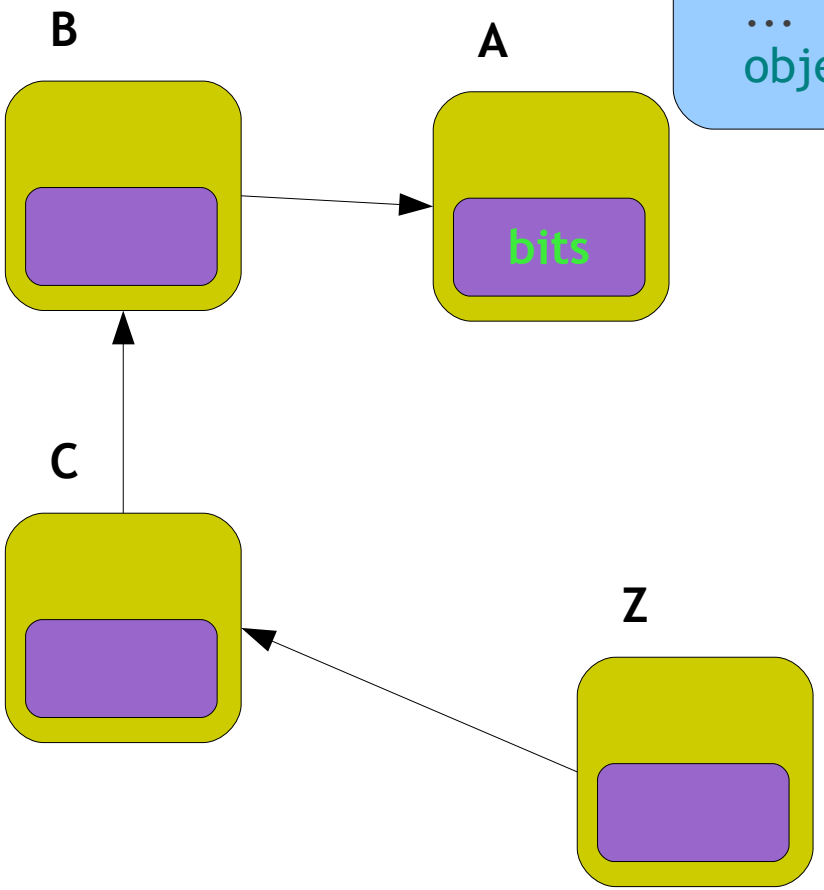


Morris, F. L. 1978. A time-and space-efficient garbage compaction algorithm. Communications of the ACM. 21, 8, 662-665.

MarkAndCompactGC

#setNewPosition:

```
follow: root count: size startingAt: base
  ...
  object _threadWith: reference at: oldIndex.
```

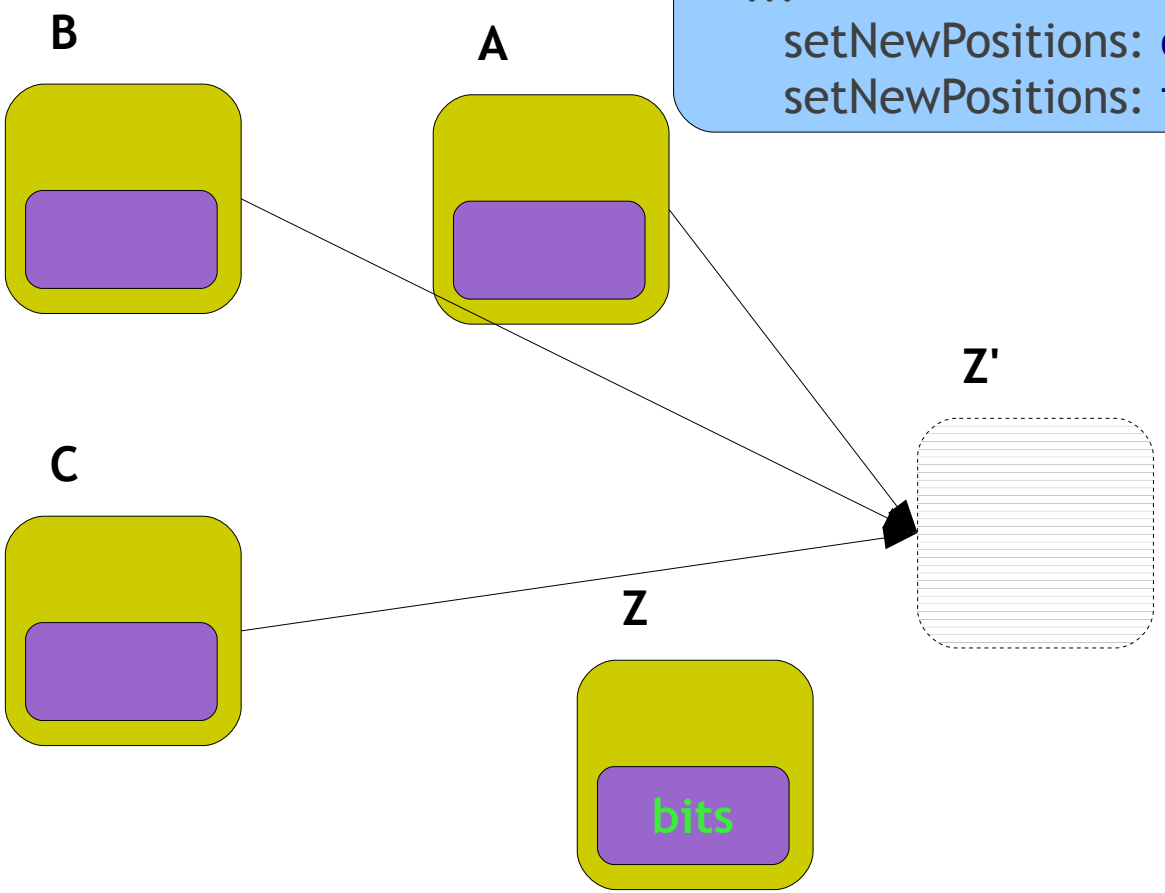


Morris, F. L. 1978. A time-and space-efficient garbage compaction algorithm. Communications of the ACM. 21, 8, 662-665.

MarkAndCompactGC

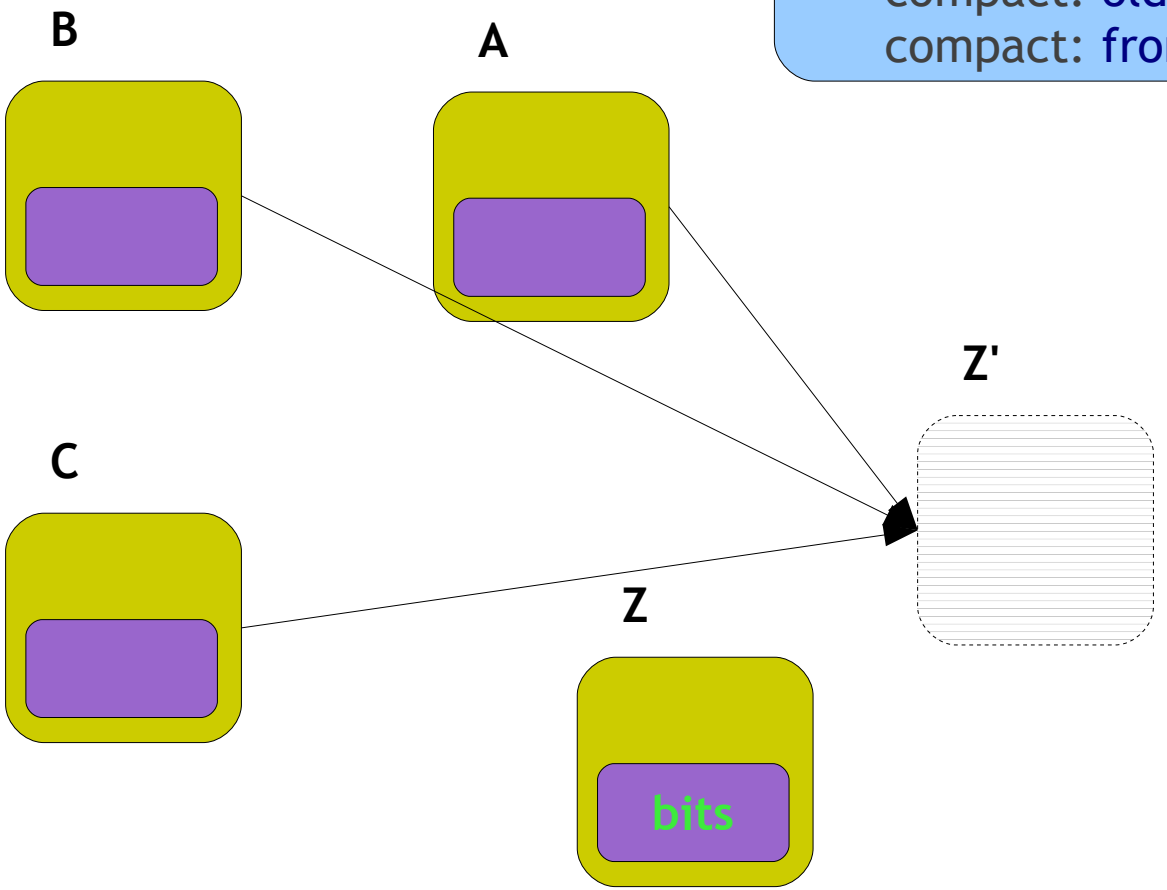
#setNewPosition:

```
collect
    ...
    setNewPositions: oldSpace;
    setNewPositions: fromSpace;
```



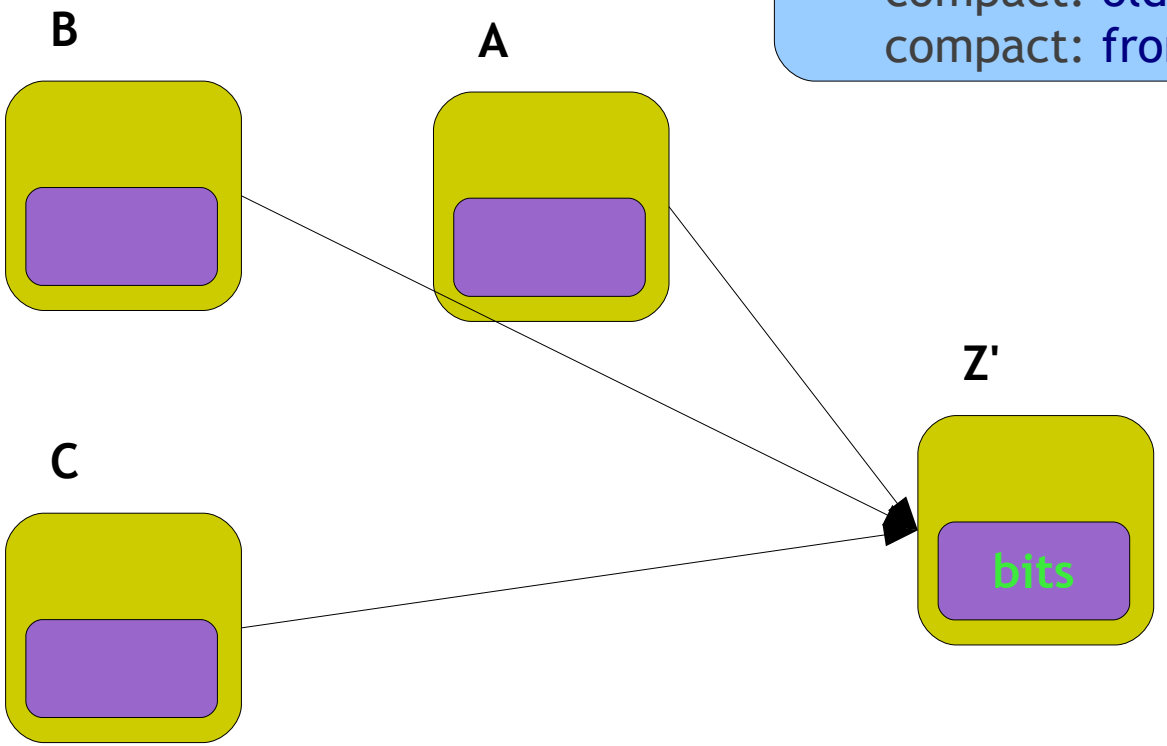
MarkAndCompactGC

```
collect
  ...
    compact: oldSpace;
    compact: fromSpace;
```



MarkAndCompactGC

```
collect
  ...
    compact: oldSpace;
    compact: fromSpace;
```



MarkAndCompactGC

#setNewPosition:

```
setNewPositions: space
  self
    seenObjectsFrom: space base
    to: space nextFree
    do: [:object :headerSize | | newPosition nextReference reference headerBits |
      newPosition := auxSpace nextFree + headerSize.
      reference := object _headerBits _unrotate.

        headerBits := reference _basicAt: 1.
        reference _basicAt: 1 put: newPosition _toObject.



      object _basicAt: -1 put: headerBits; _beSeenInSpace.
      auxSpace nextFree: newPosition + object _byteSize]
```

MarkAndCompactGC

#setNewPosition:

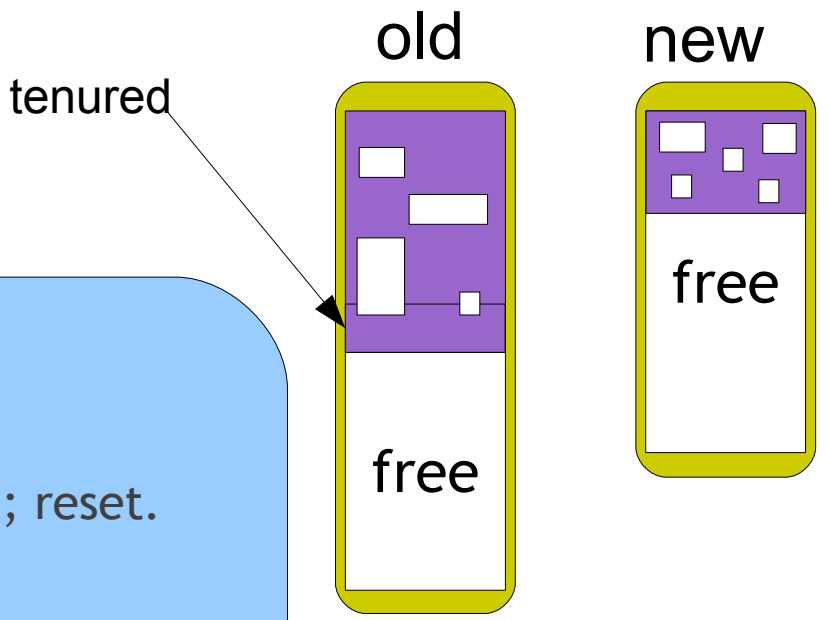
- Unthread references

```
setNewPositions: space
  self
    seenObjectsFrom: space base
    to: space nextFree
    do: [:object :headerSize | | newPosition nextReference reference headerBits |
      newPosition := auxSpace nextFree + headerSize.
      reference := object _headerBits _unrotate.
      [
        headerBits := reference _basicAt: 1.
        reference _basicAt: 1 put: newPosition _toObject.
        nextReference := headerBits _unrotate.
        nextReference _isSmallInteger]
        whileFalse: [reference := nextReference].
      object _basicAt: -1 put: headerBits; _beSeenInSpace.
      auxSpace nextFree: newPosition + object _byteSize]
```

MTMarkAndCompactGC

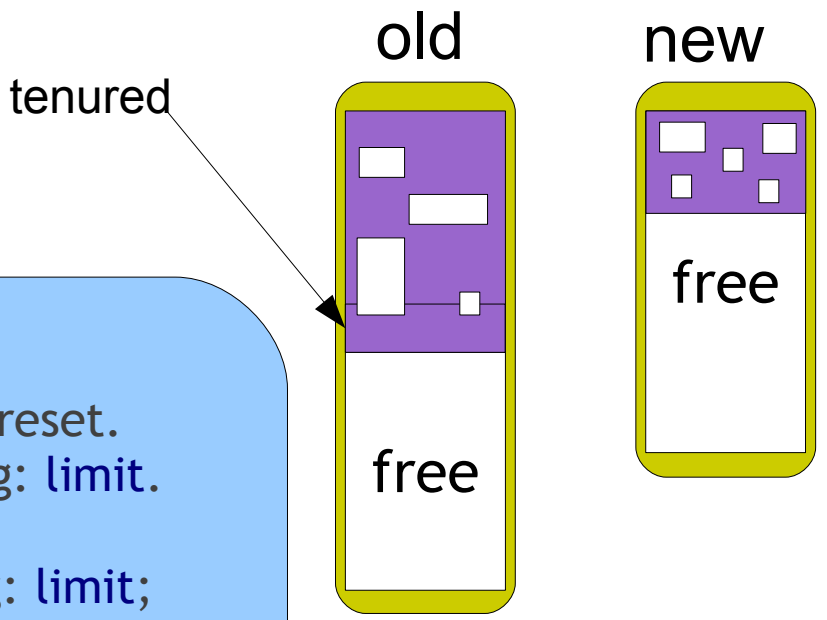
#setNewPosition:

```
setNewPositions
   remainingOld base: remainingOld base + tenured; reset.
   worker1 repositionFrom: old to: newOld.
   worker2
      repositionFrom: fromSpace to: remainingOld;
      waitForOutput.
   worker1 waitForOutput.
```



MTMarkAndCompactGC

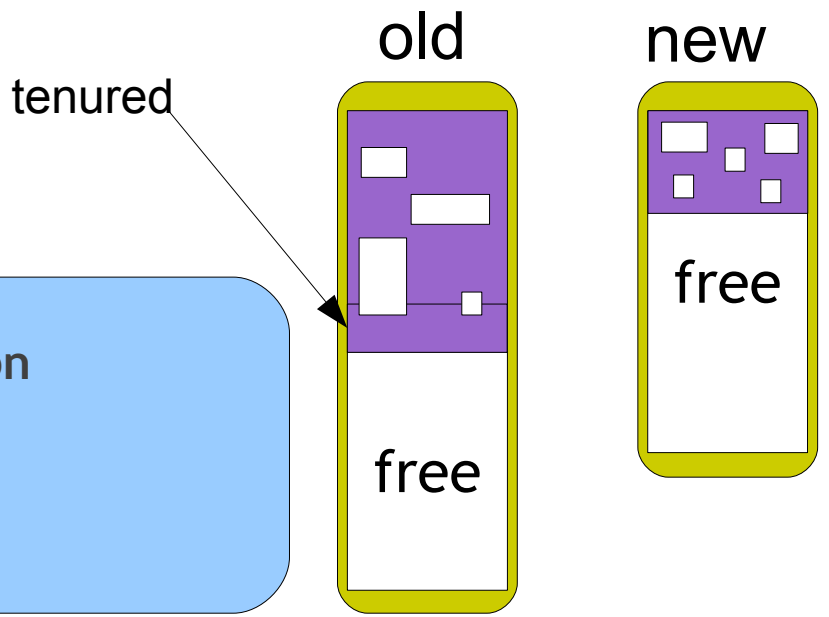
#setNewPosition:




```
setNewPositions
  remainingOld base: remainingOld base + tenured; reset.
  worker1 repositionFrom: old1h to: newOld limiting: limit.
  worker2
    repositionFrom: old2h to: remainingOld limiting: limit;
    waitForOutput.
  worker2
    repositionFrom: fromSpace to: remainingOld limiting: limit;
    waitForOutput.
  worker1 waitForOutput.
```

MTMarkAndCompactGC

#setNewPosition:




```
repositionFrom: src to: dst limiting: maxReposition
   ...
   self loadRepositionBlock.
   inbox signal
```

```
run
   [inbox wait] whileTrue: [
      source seenObjectsDo: repositionBlock.
      outbox signal]
```

```
[[self doDemo]
  on: Error do: [self makeJocks].
  Audience wows] whileTrue.
```

```
[Audience hasQuestions] whileTrue: [
    self answer: Audience nextQuestion].

Audience do: [:you | self thank: you].

self returnTo: Audience
```