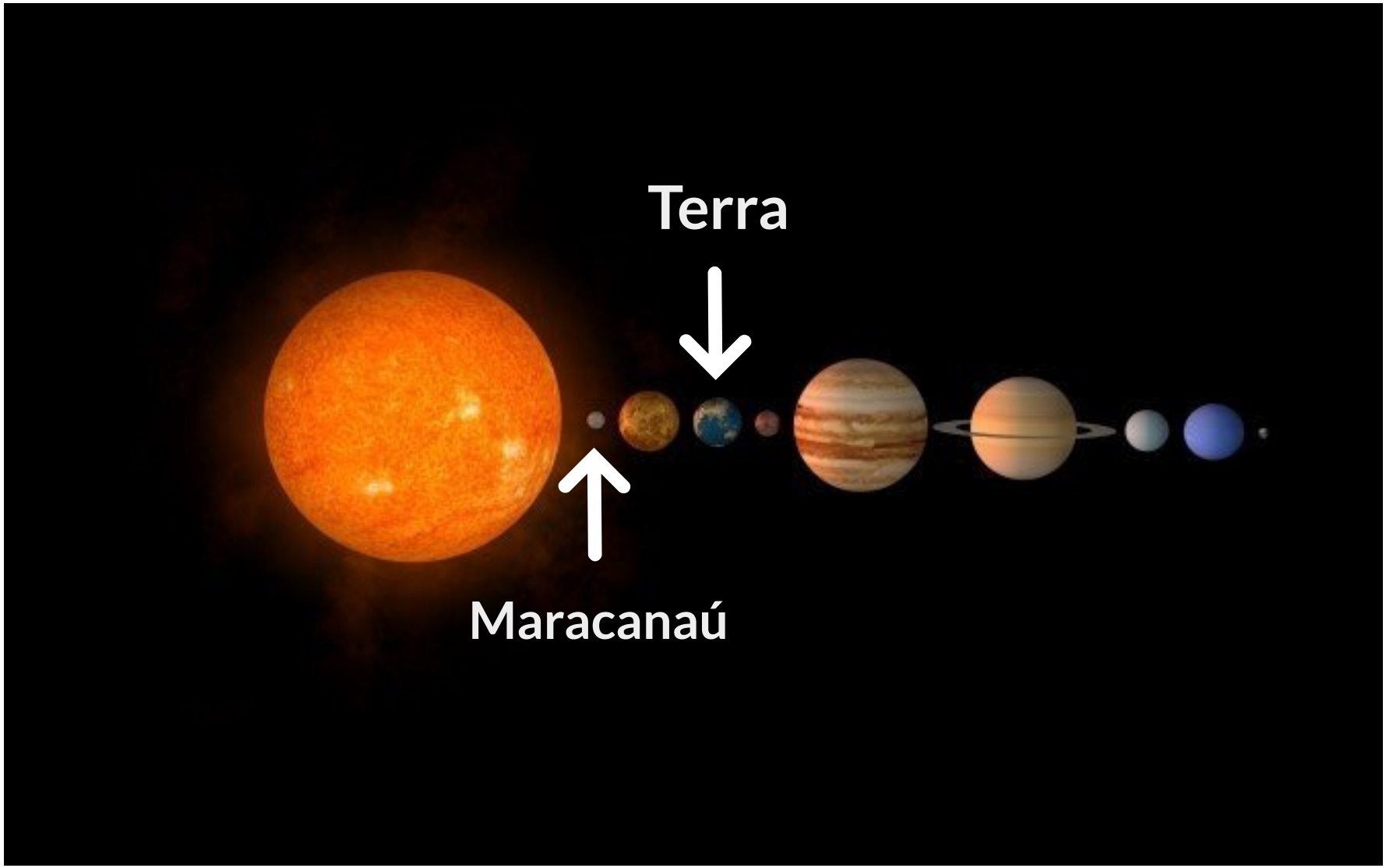
Ruby

Internals

Alisson Bruno

@alissonbrunosa


Terra
Maracanaú

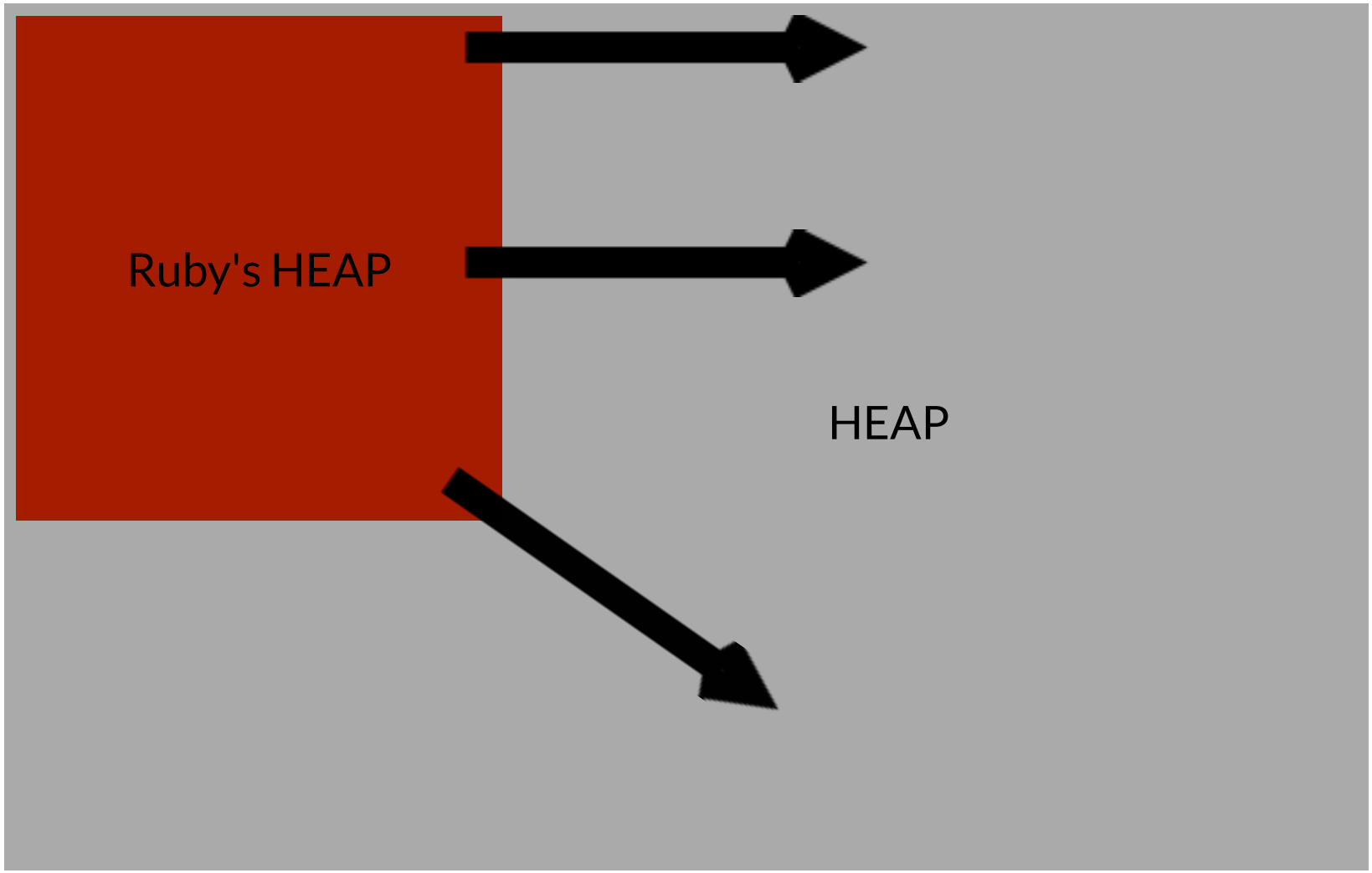
Resultados
Digitais

GC

Short history

- Allocate memory
- Identify garbage
- Reclaim memory

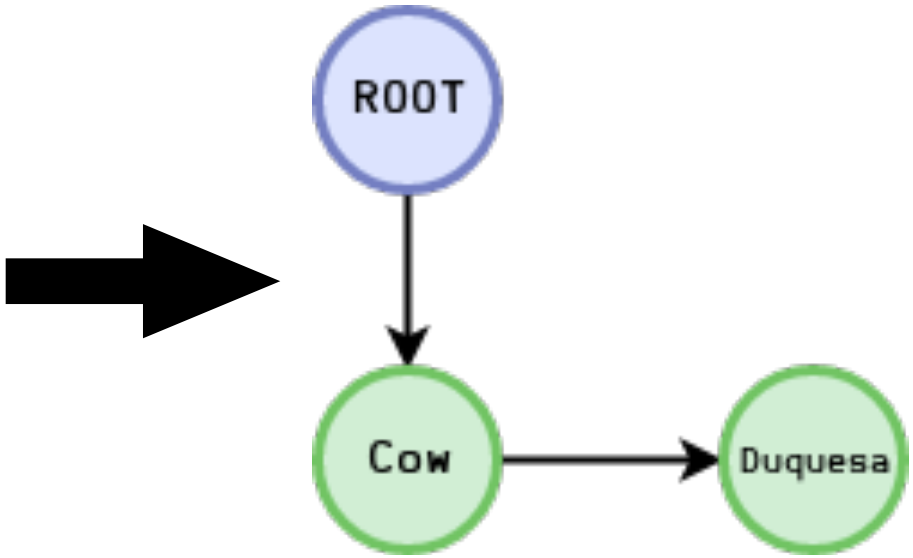
Ruby's memory



Collection

```ruby
class Cow
  def initialize(name)
    @name = name
  end
end
```

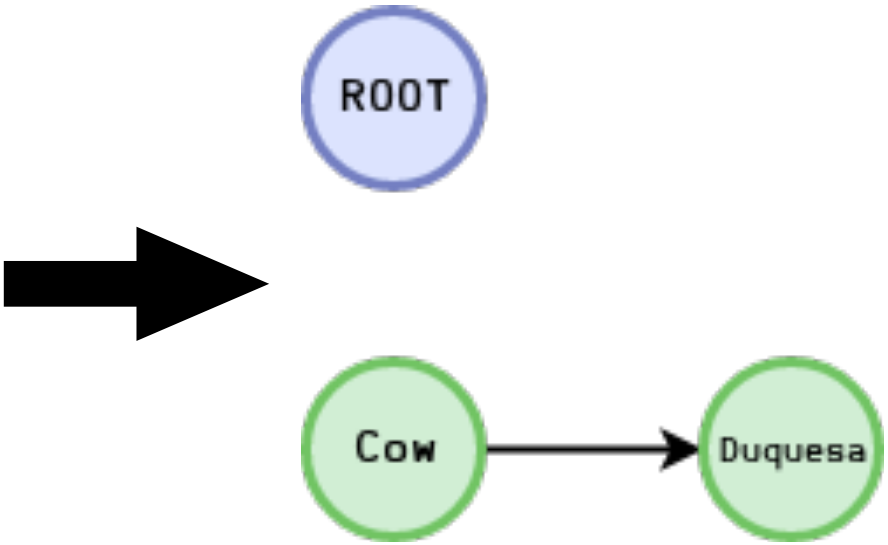
```ruby
cow = Cow.new("Duquesa")
```



```ruby
class Cow
  def initialize(name)
    @name = name
  end
end
```

```ruby
cow = Cow.new("Duquesa")
```

```ruby
cow = nil
```



Collection algorithms

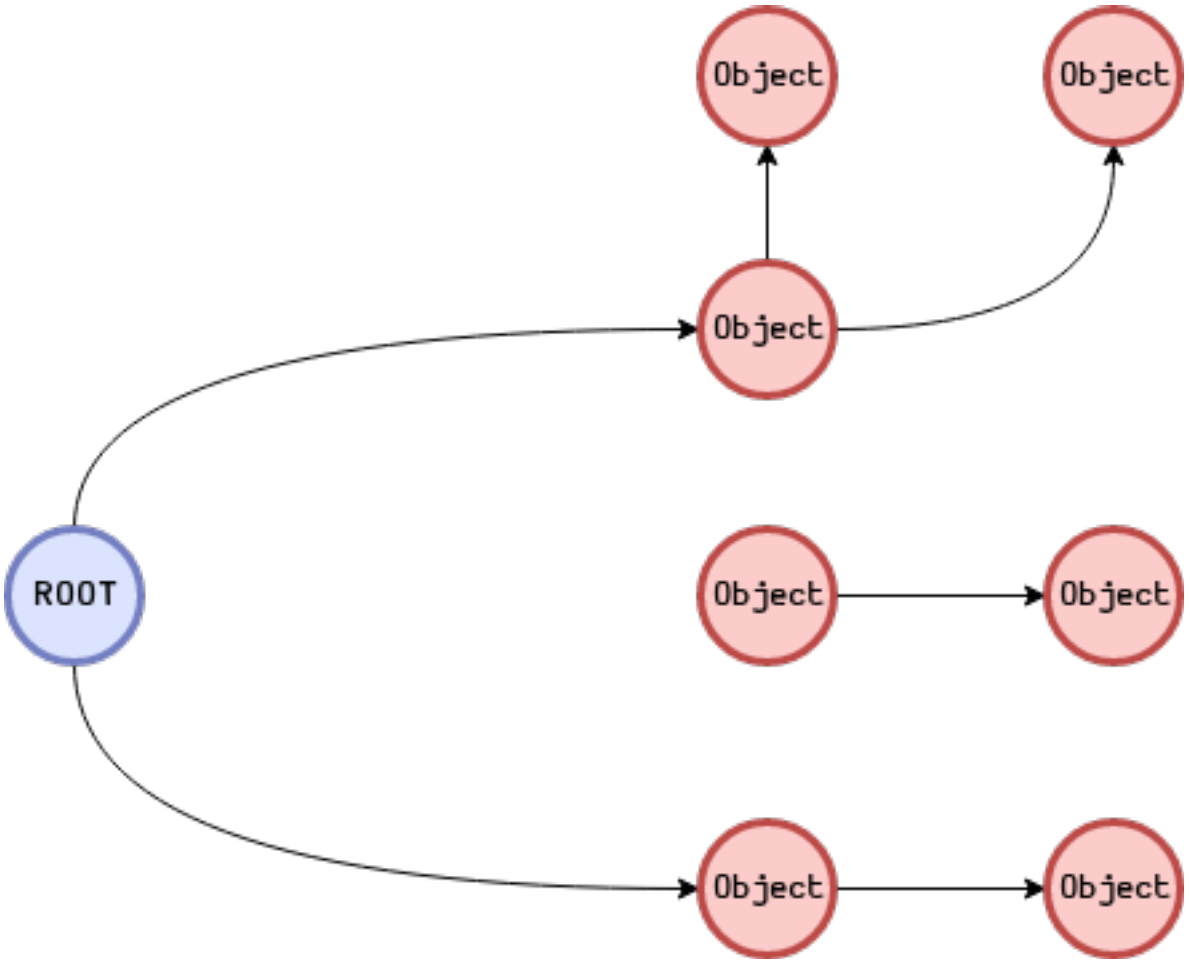
Mark & Sweep

Generational

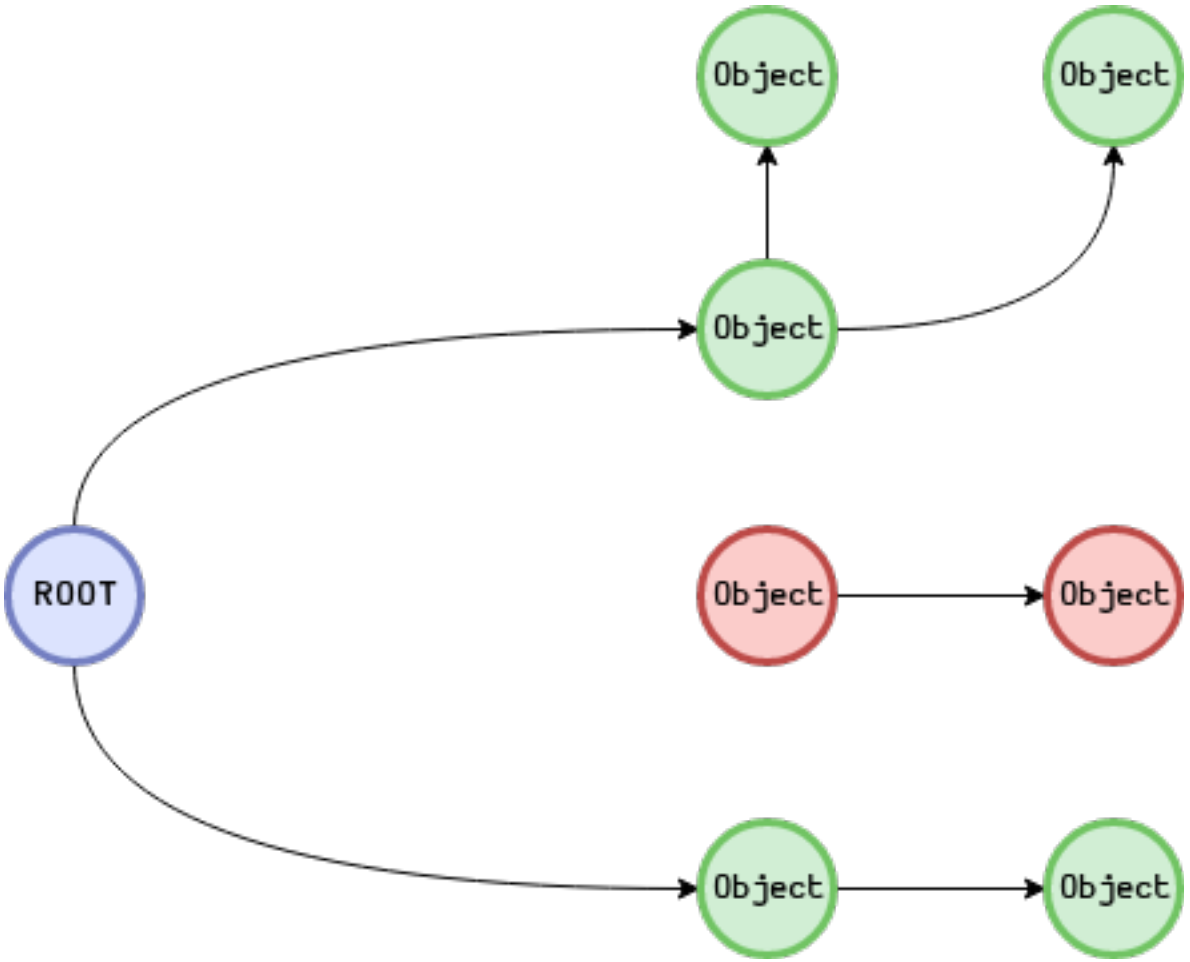
Incremental

Mark & Sweep

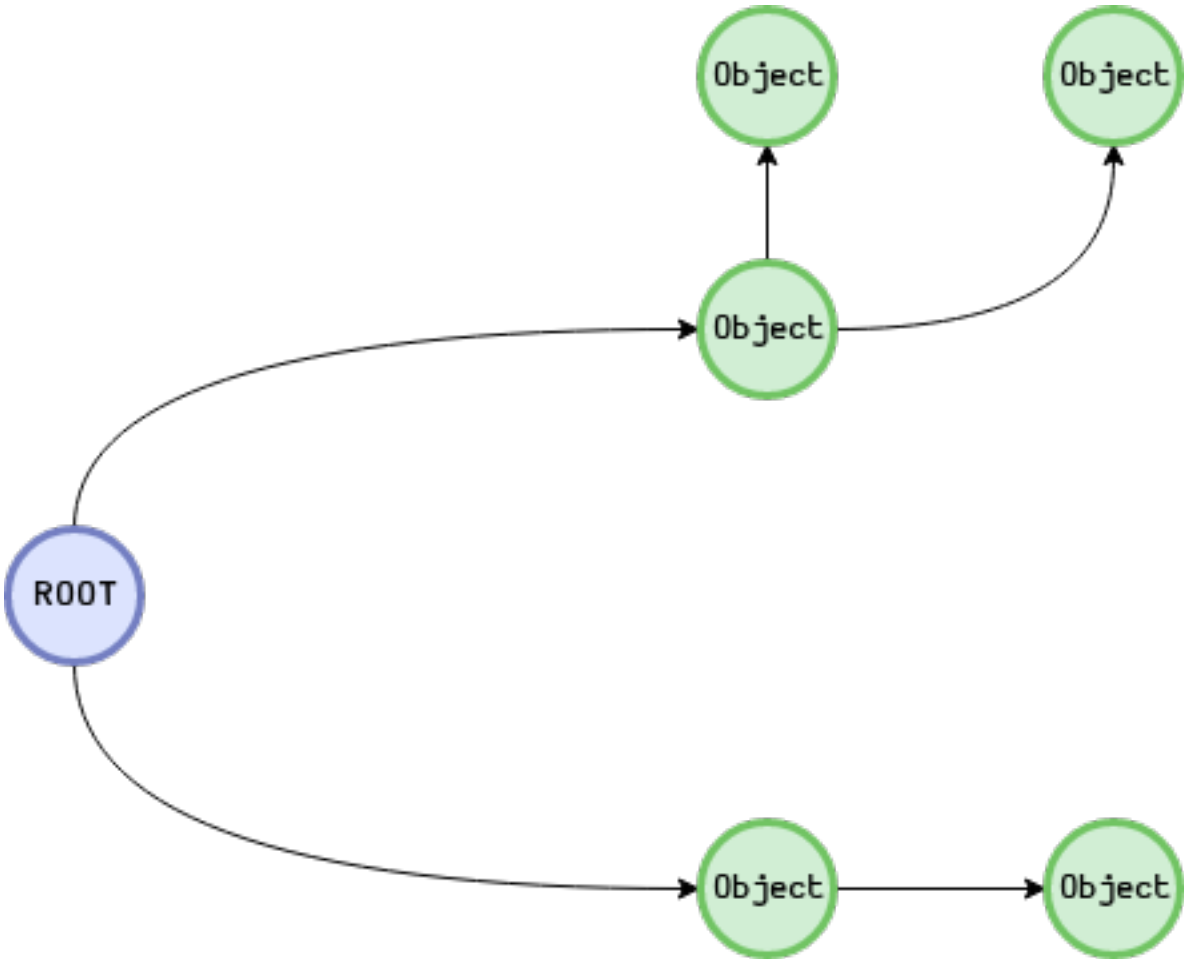
Mark& Sweep



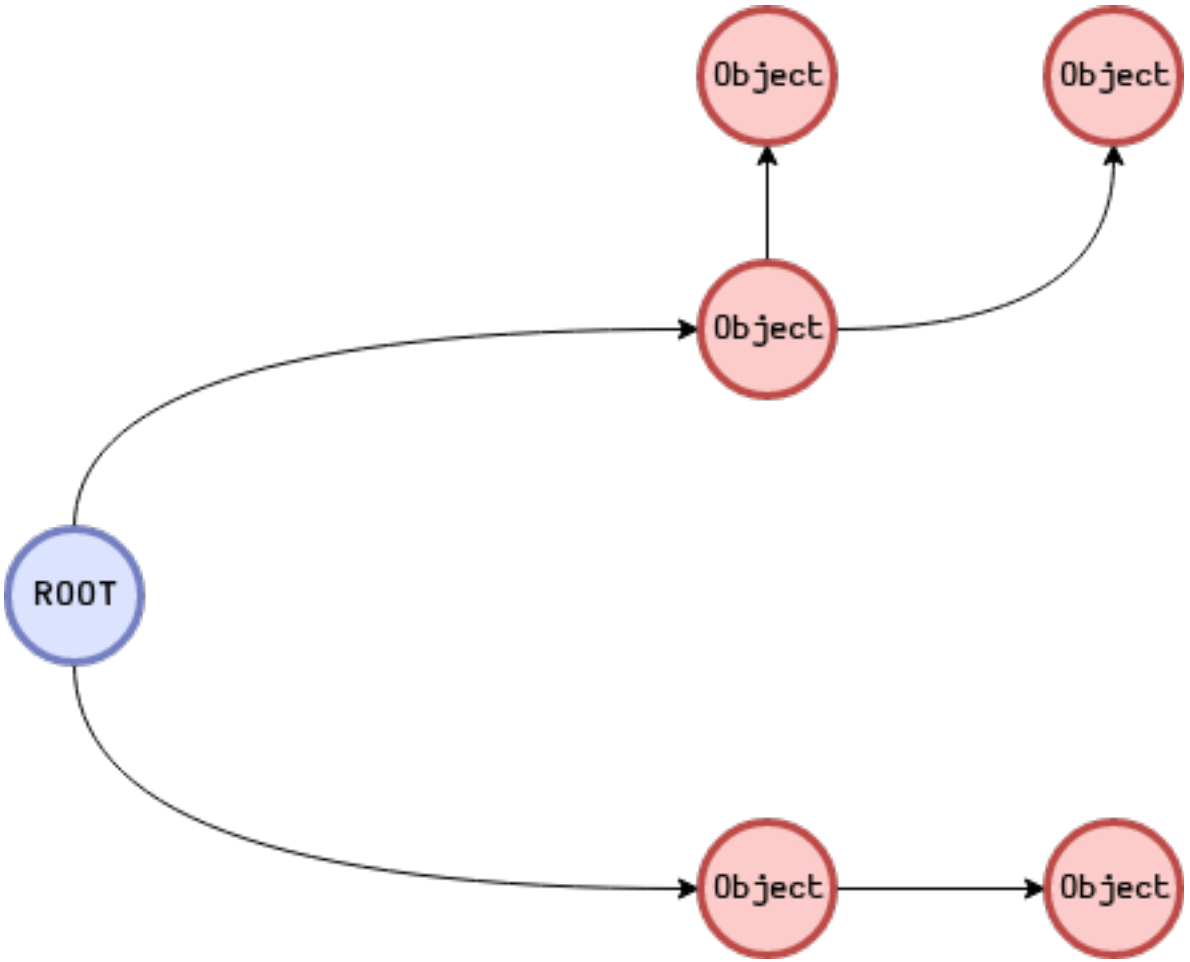
Mark



Sweep



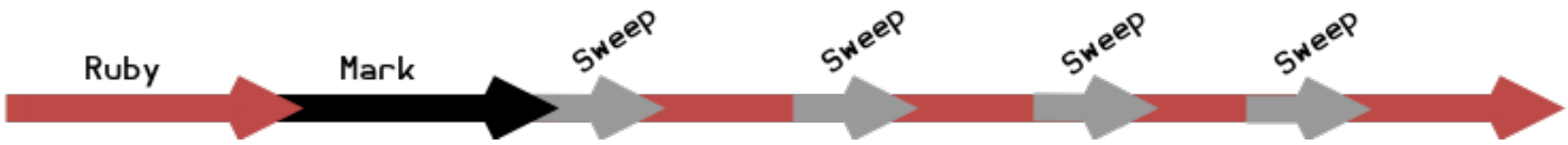
Unmark



Mark & Sweep

Low throughput
Easy implementation
It needs to stop the world

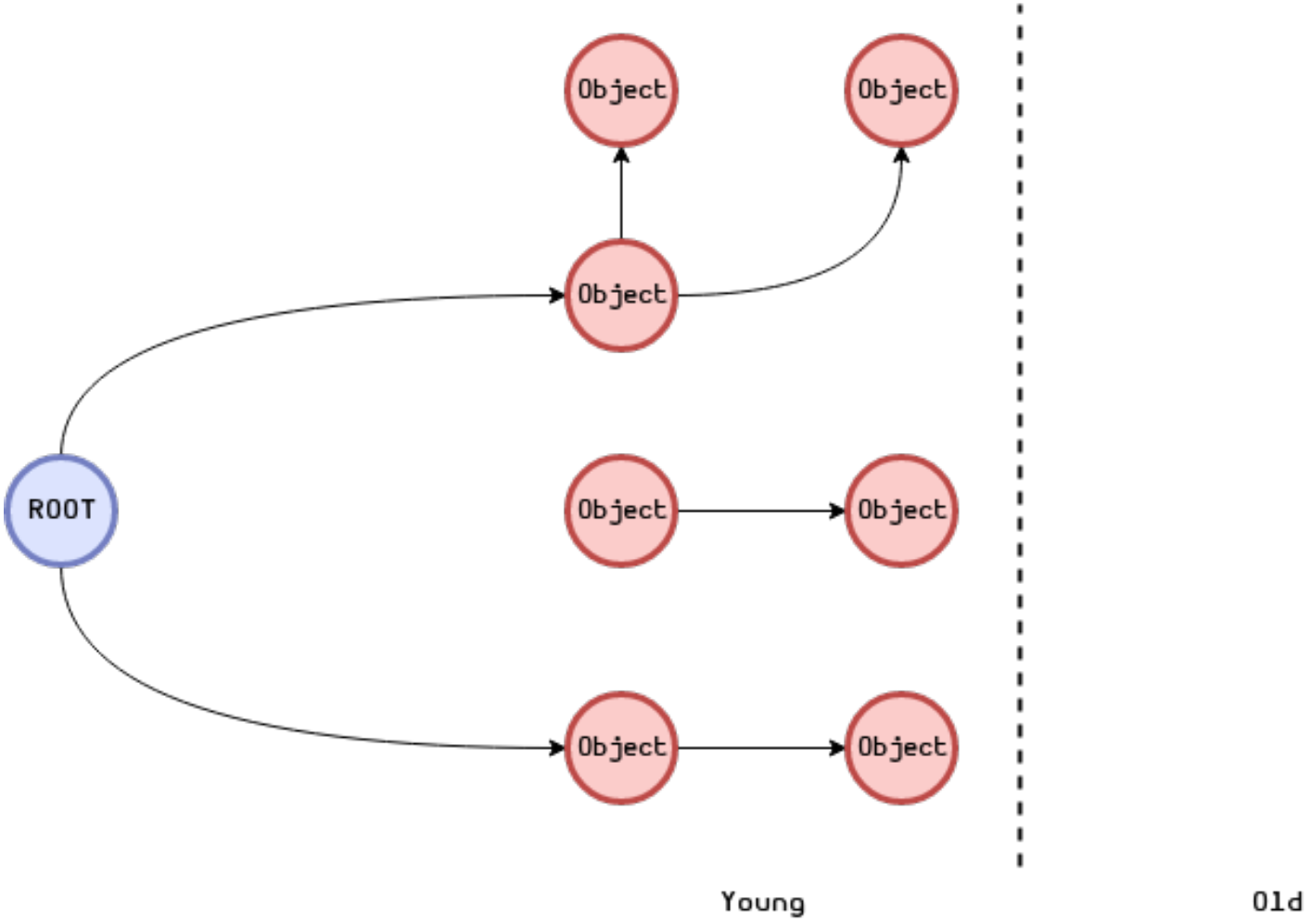
Mark & Lazy Sweep



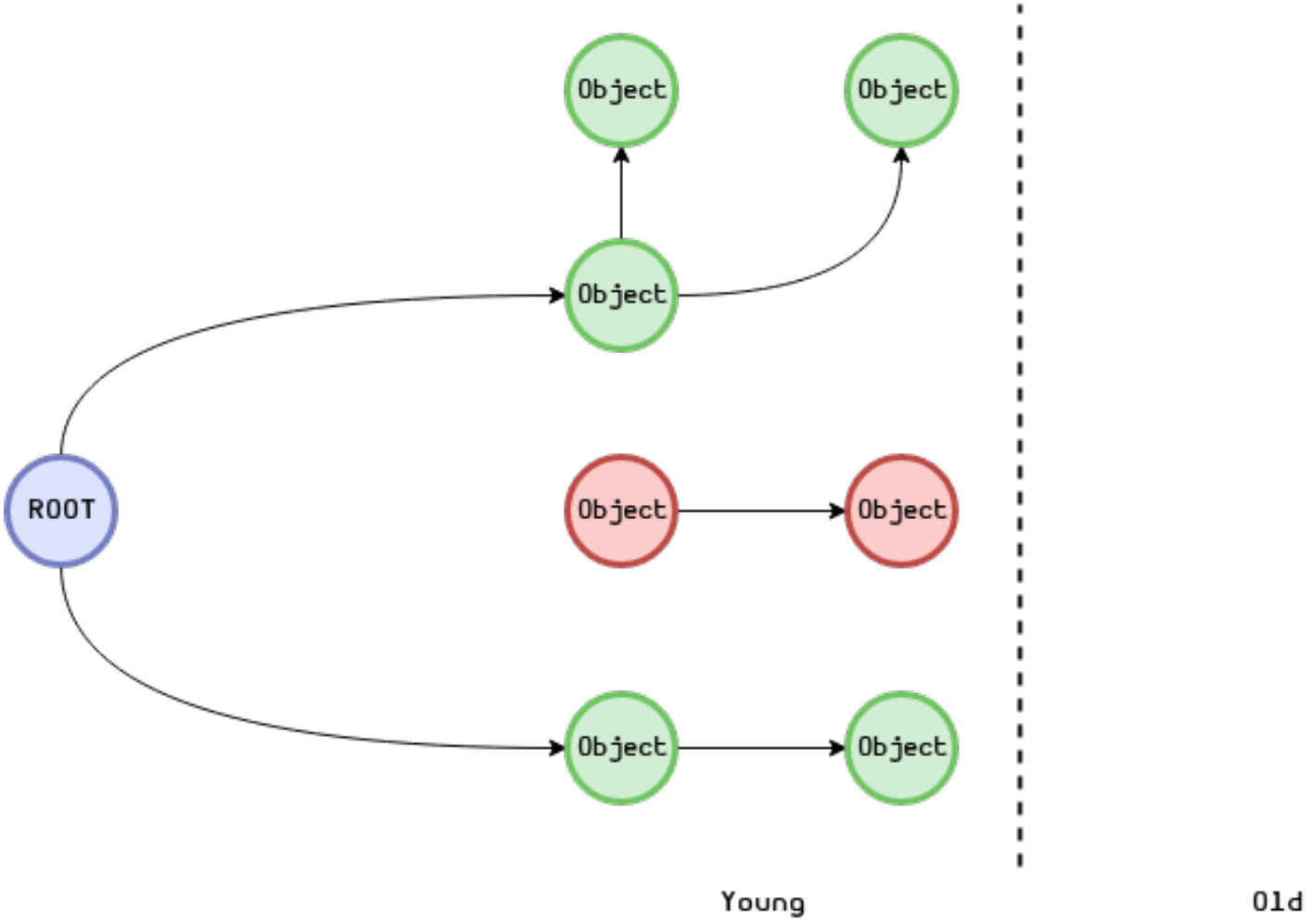
Generational

Divide objects into two generations

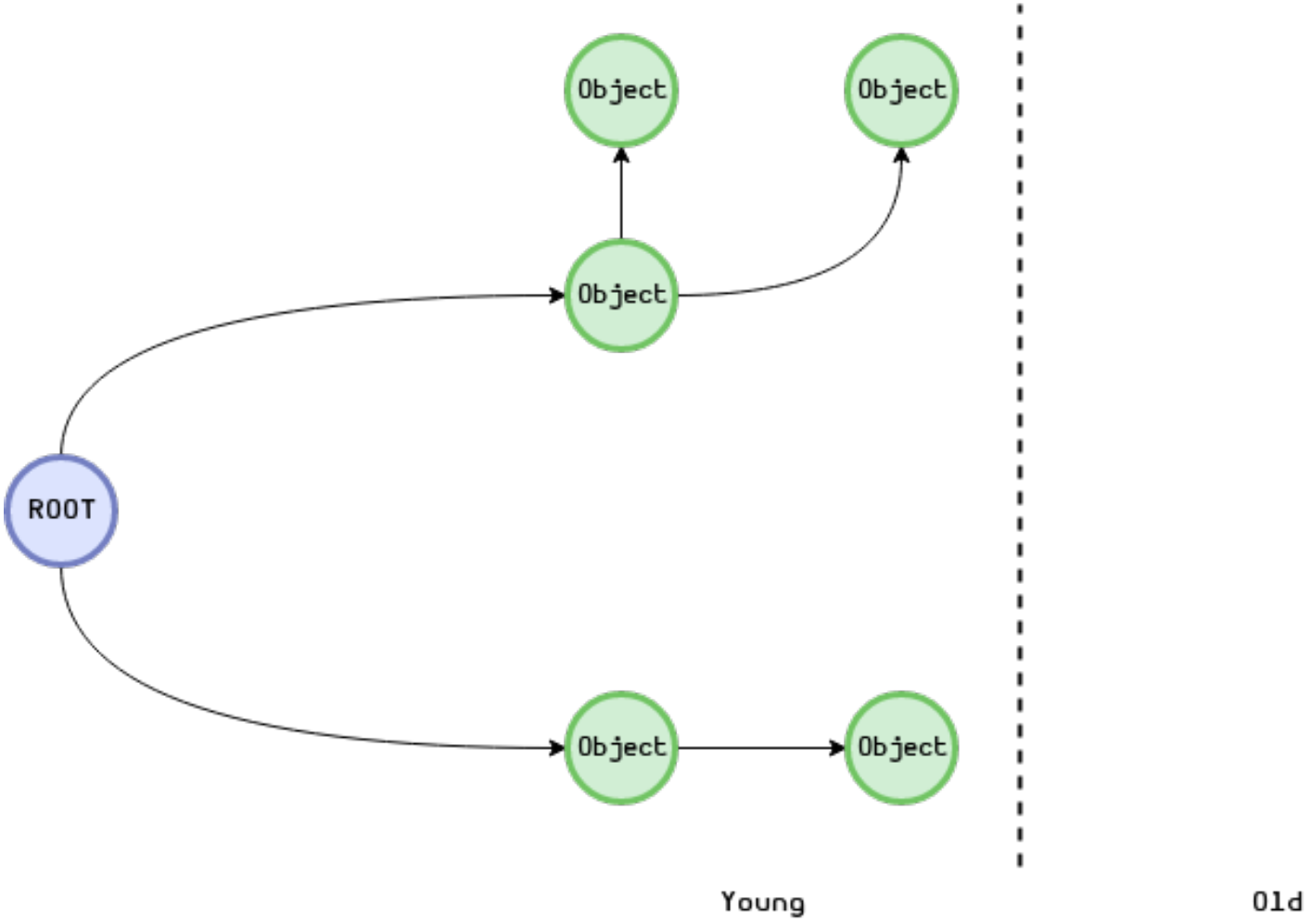
Generational



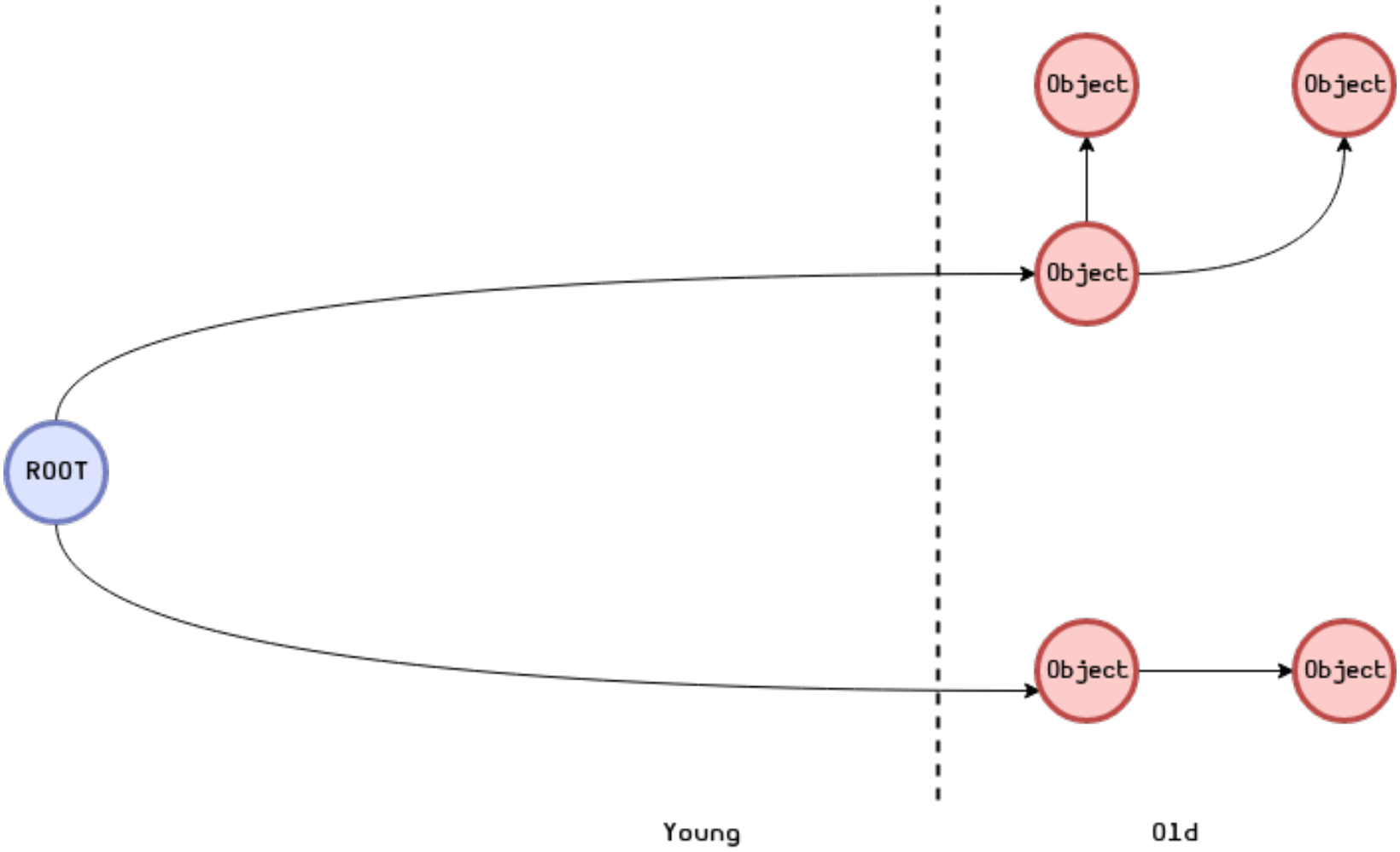
Mark


Object
Object
Object
ROOT
Object
Object
Object
Object
Young
Old

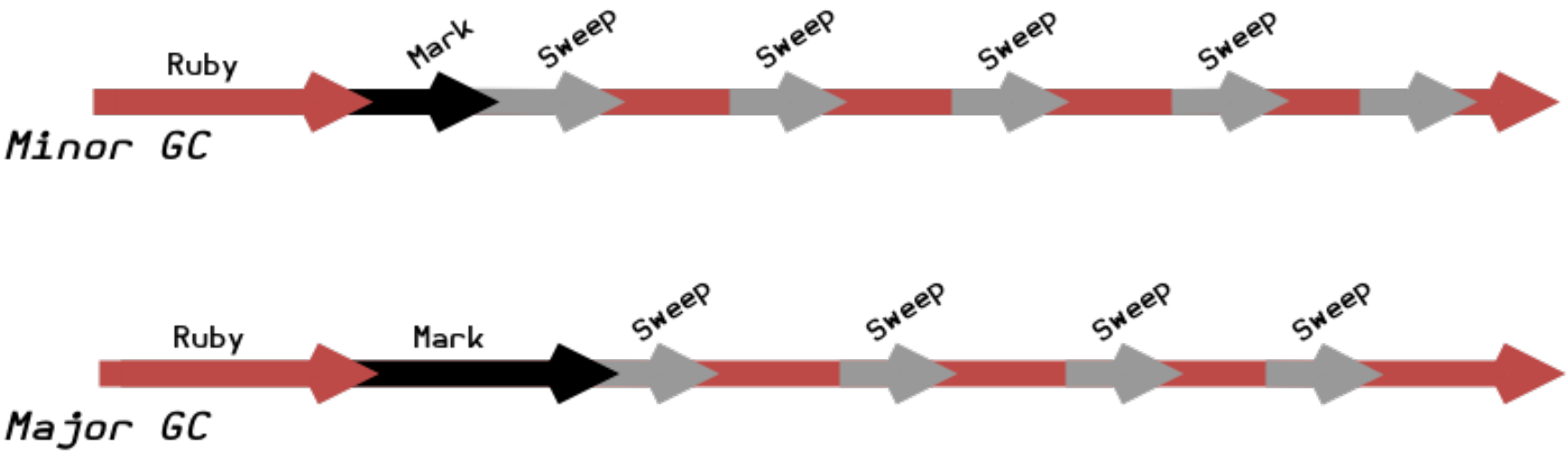
Sweep



Objects were moved to the old generation



RGenGC


Ruby
Mark
Sweep
Sweep
Sweep
Sweep
Minor GC
Ruby
Mark
Sweep
Sweep
Sweep
Sweep
Major GC

Faster than M&S

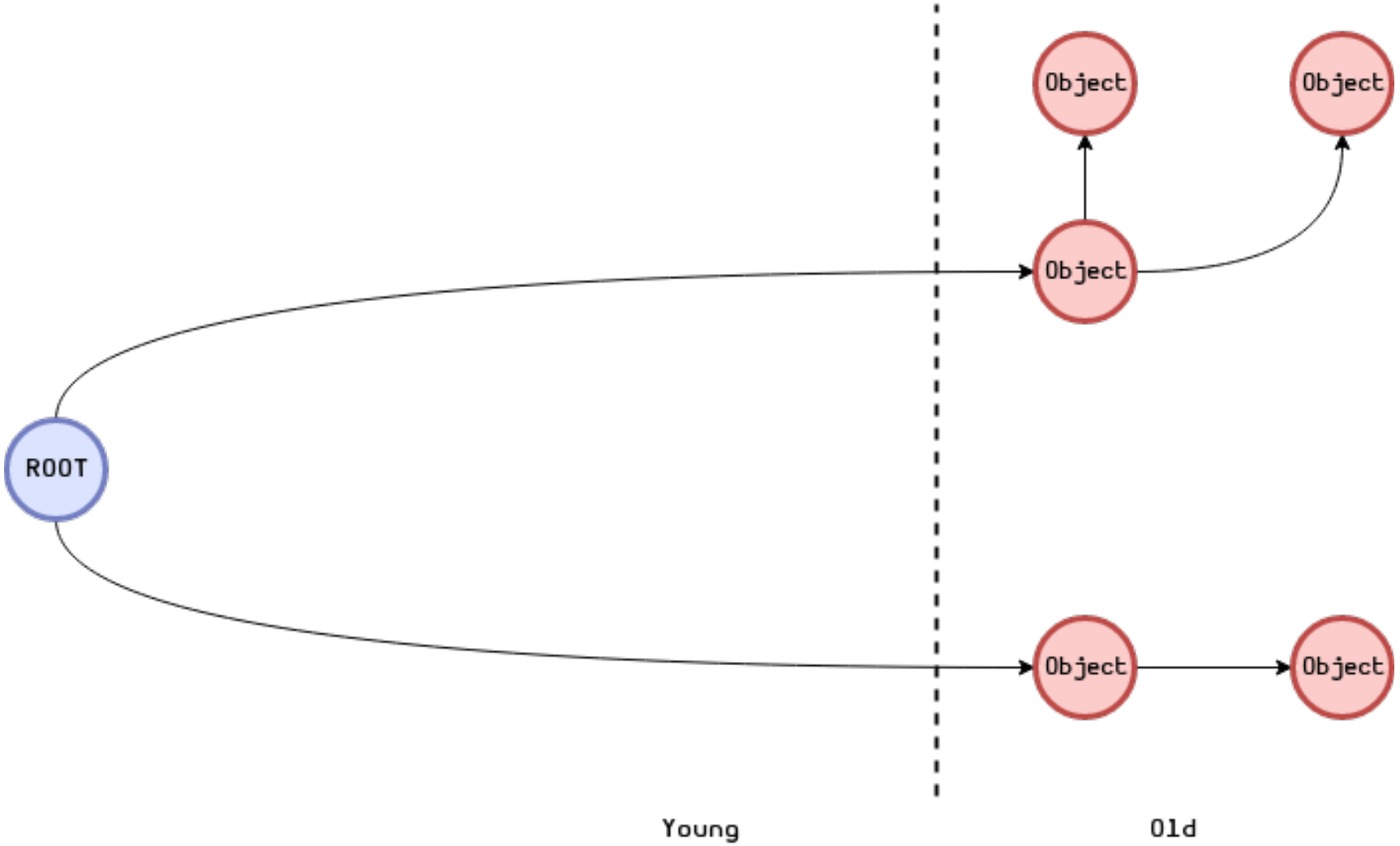
High throughput

Complex implementation

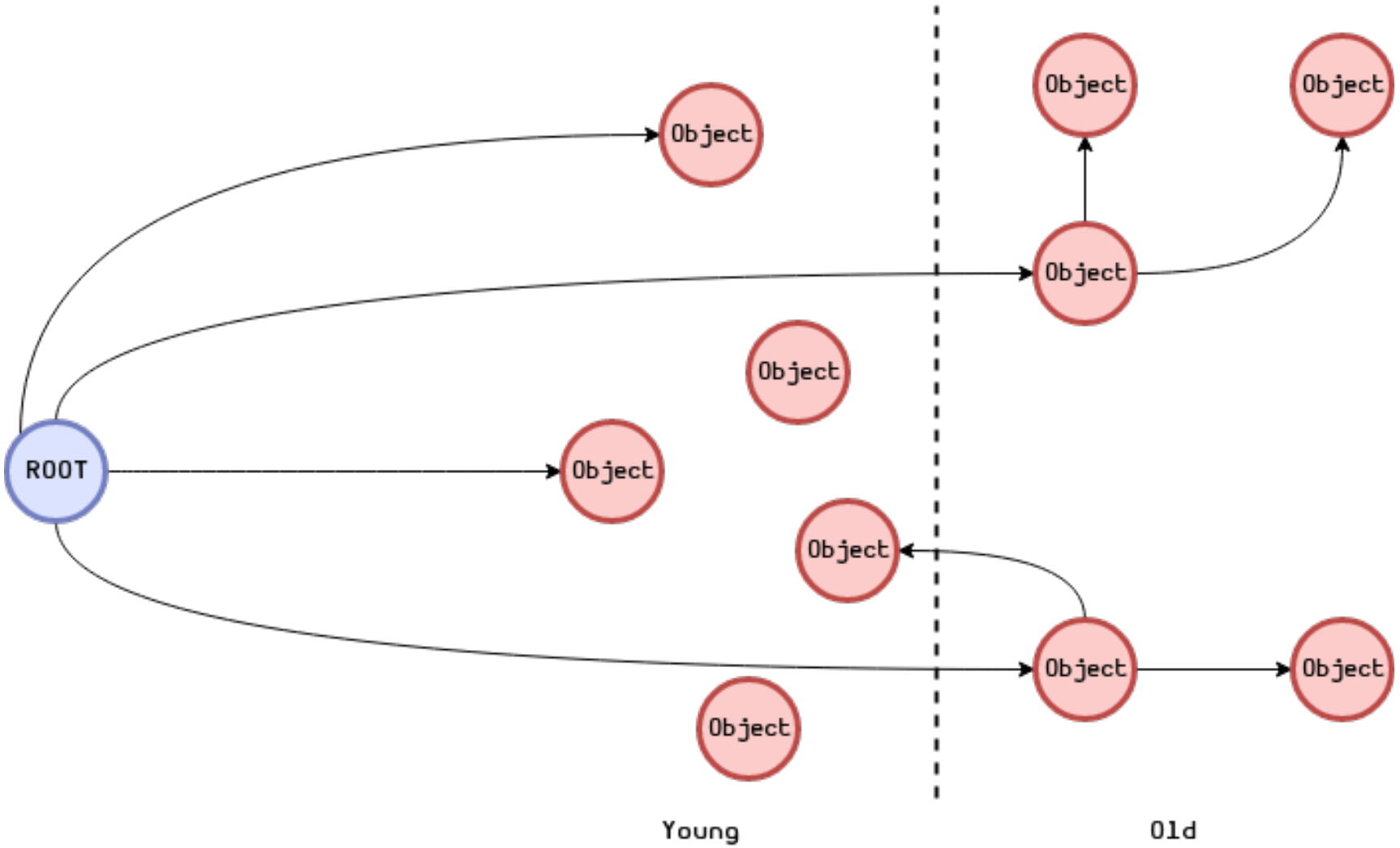
It still needs to stop the world

There's a bug 

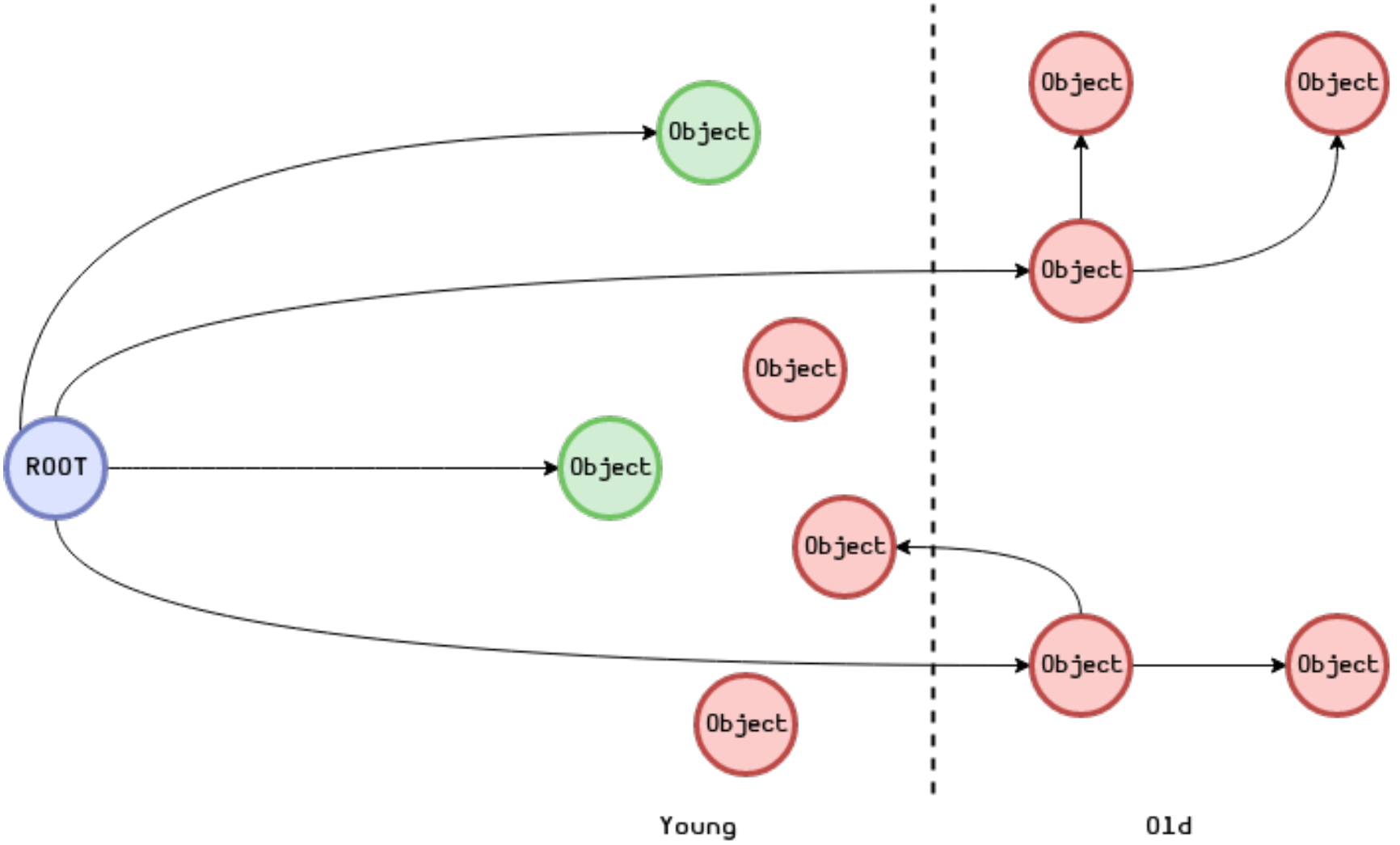
Scenario



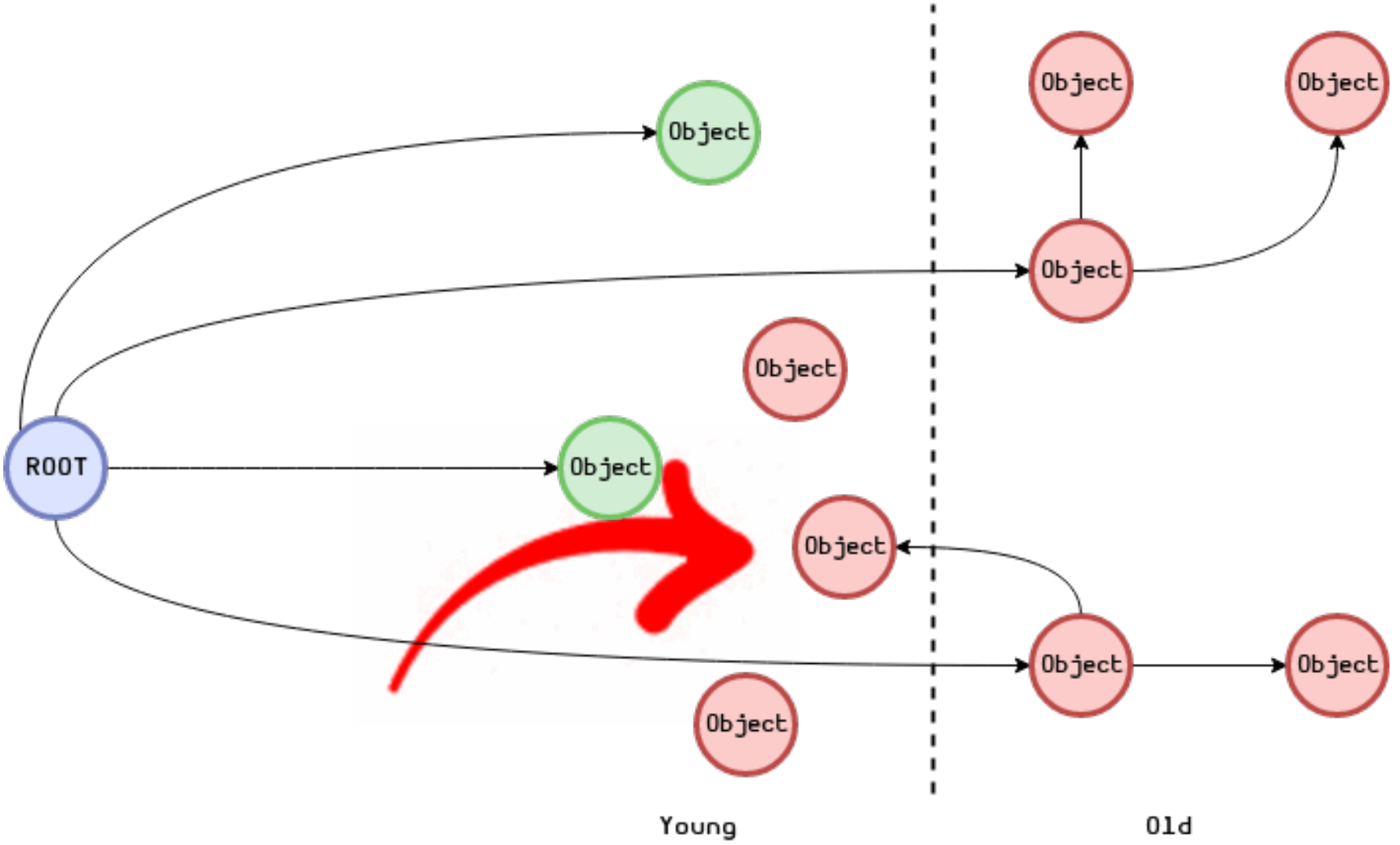
Scenario


Object
Object
Object
Object
Object
ROOT
Object
Object
Object
Object
Young
Old

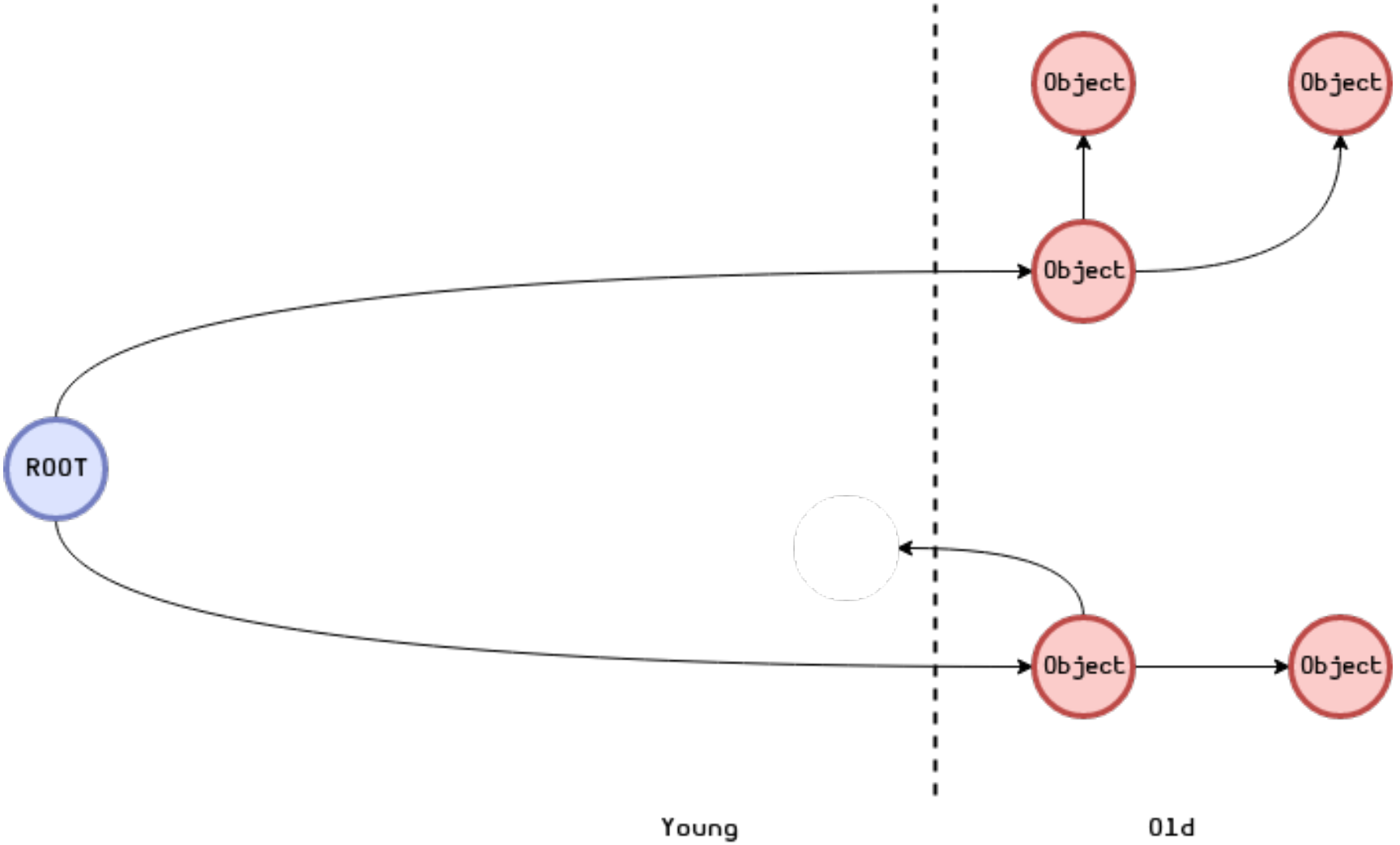
Mark phase


Object
Object
Object
Object
Object
ROOT
Object
Object
Object
Object
Object
Young
Old

Mark phase

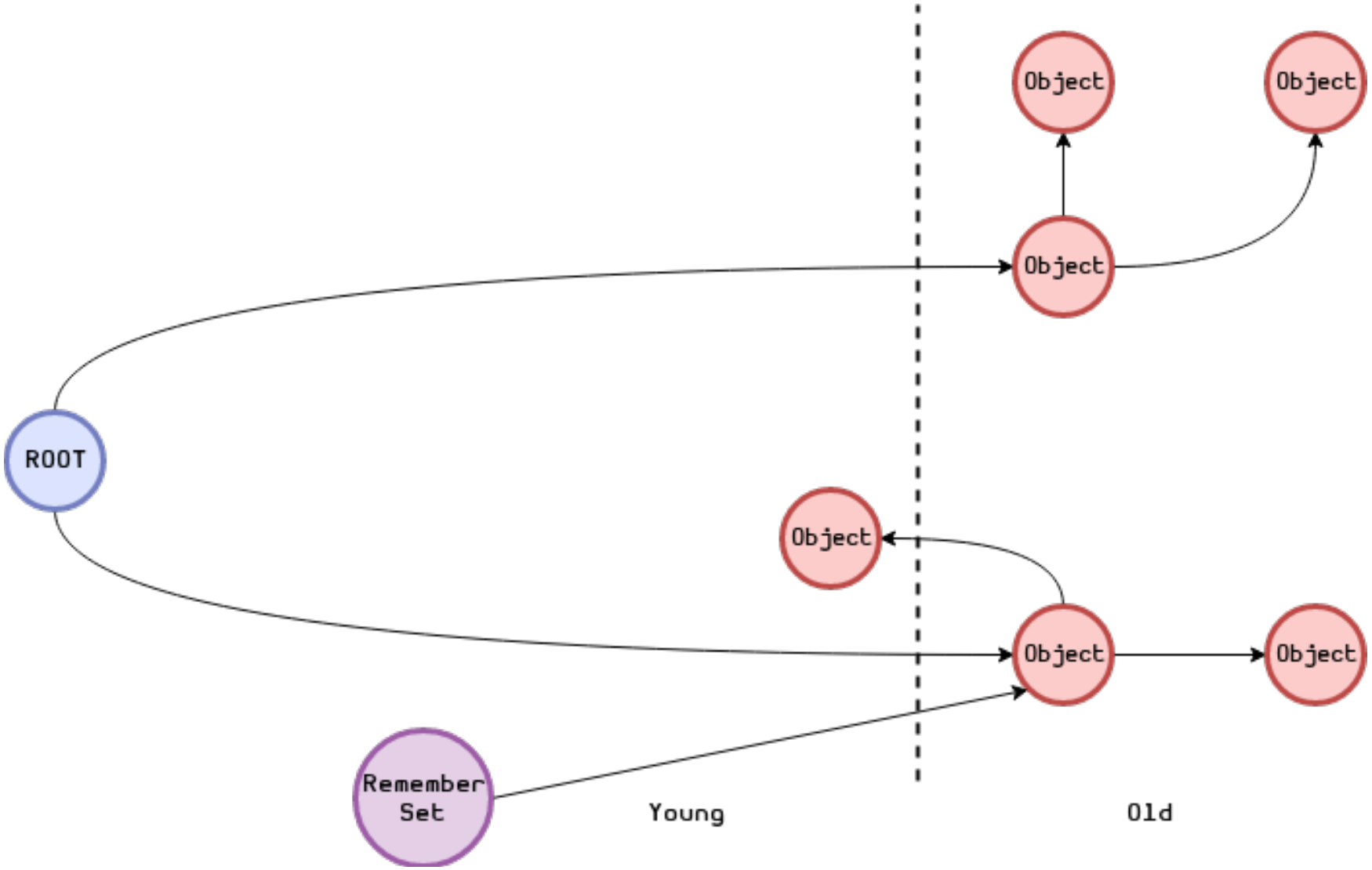

Object
Object
Object
Object
Object
ROOT
Object
Object
Object
Object
Object
Young
Old

Sweep phase

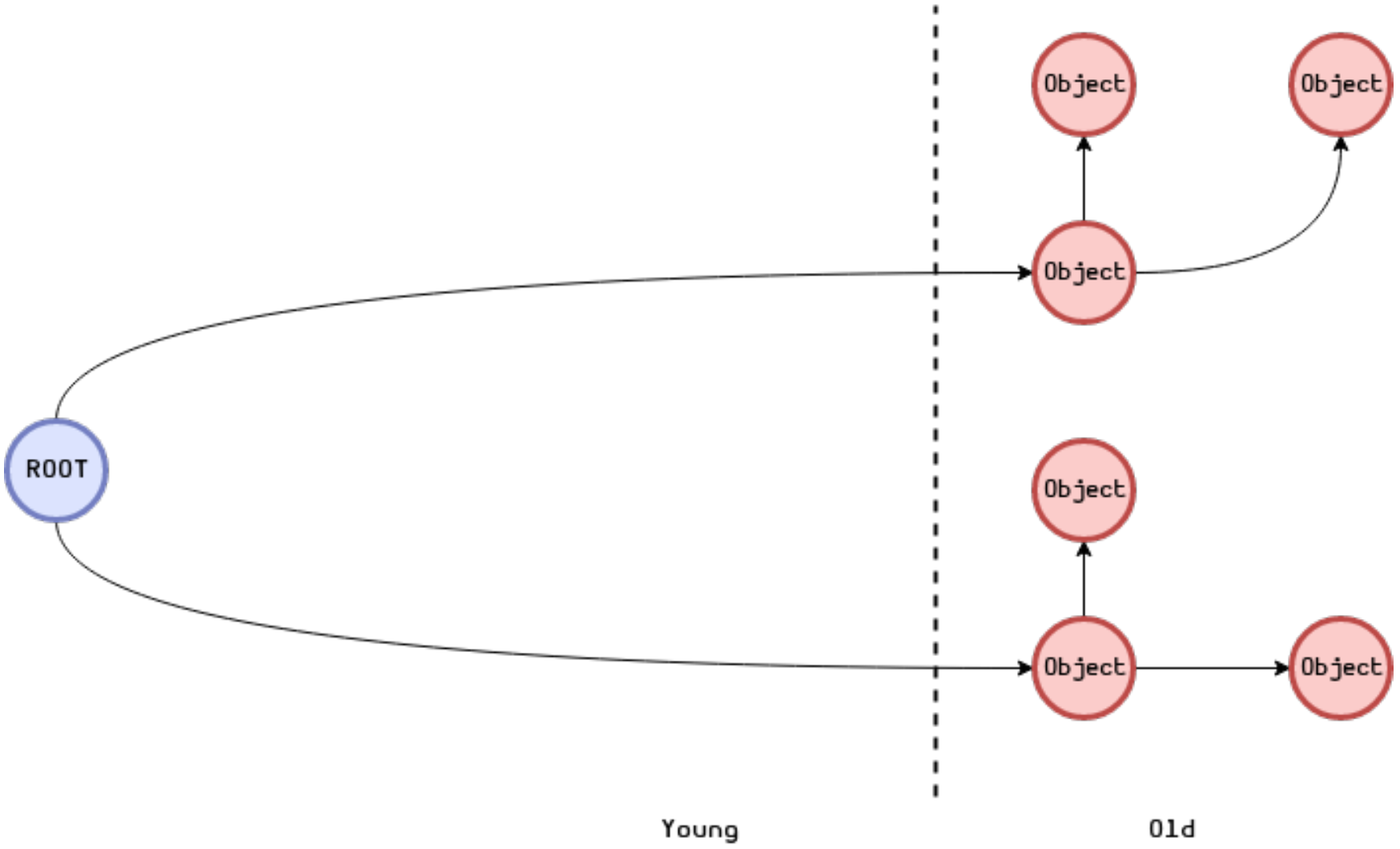


Write Barriers

Remembered Set

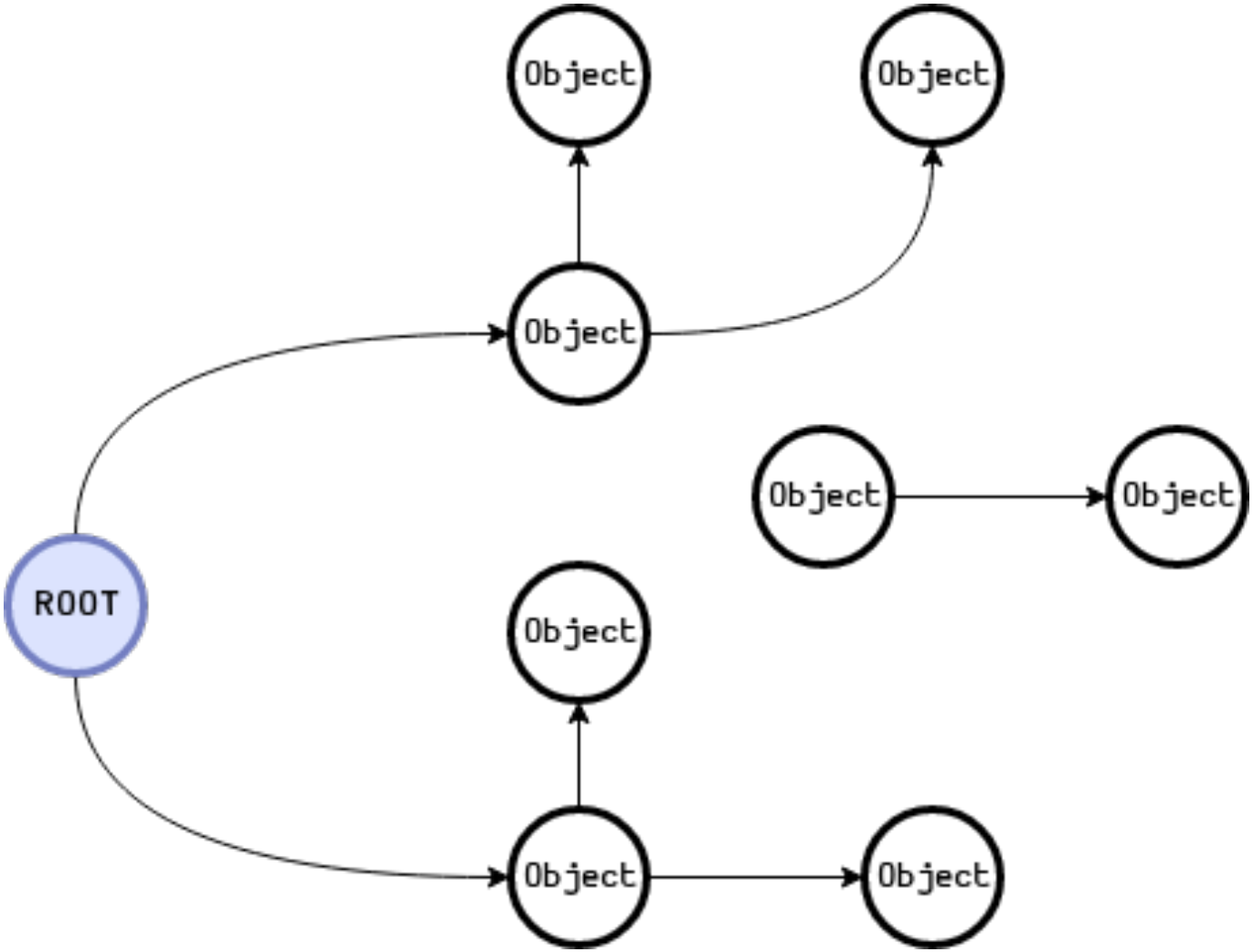


After GC

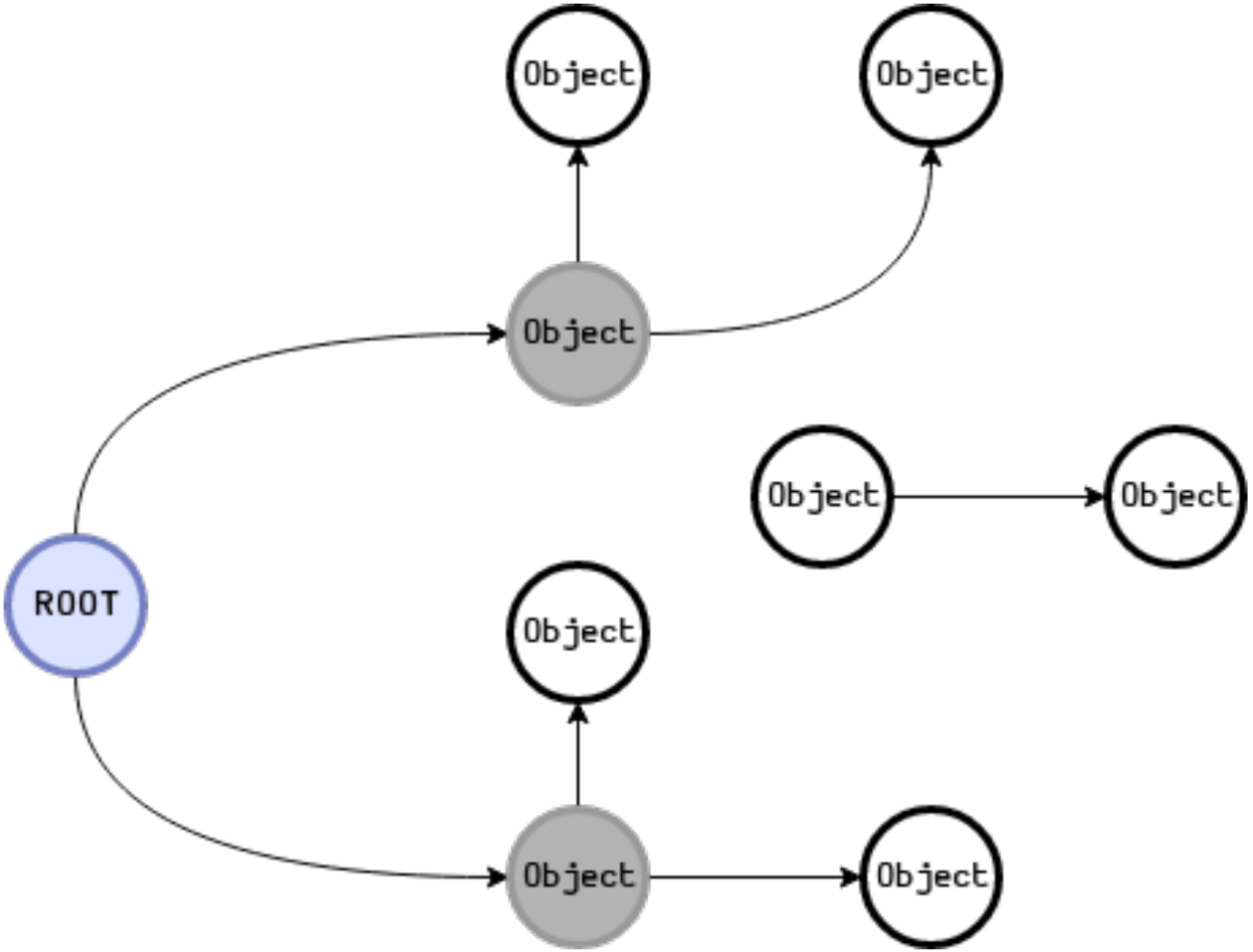


Incremental

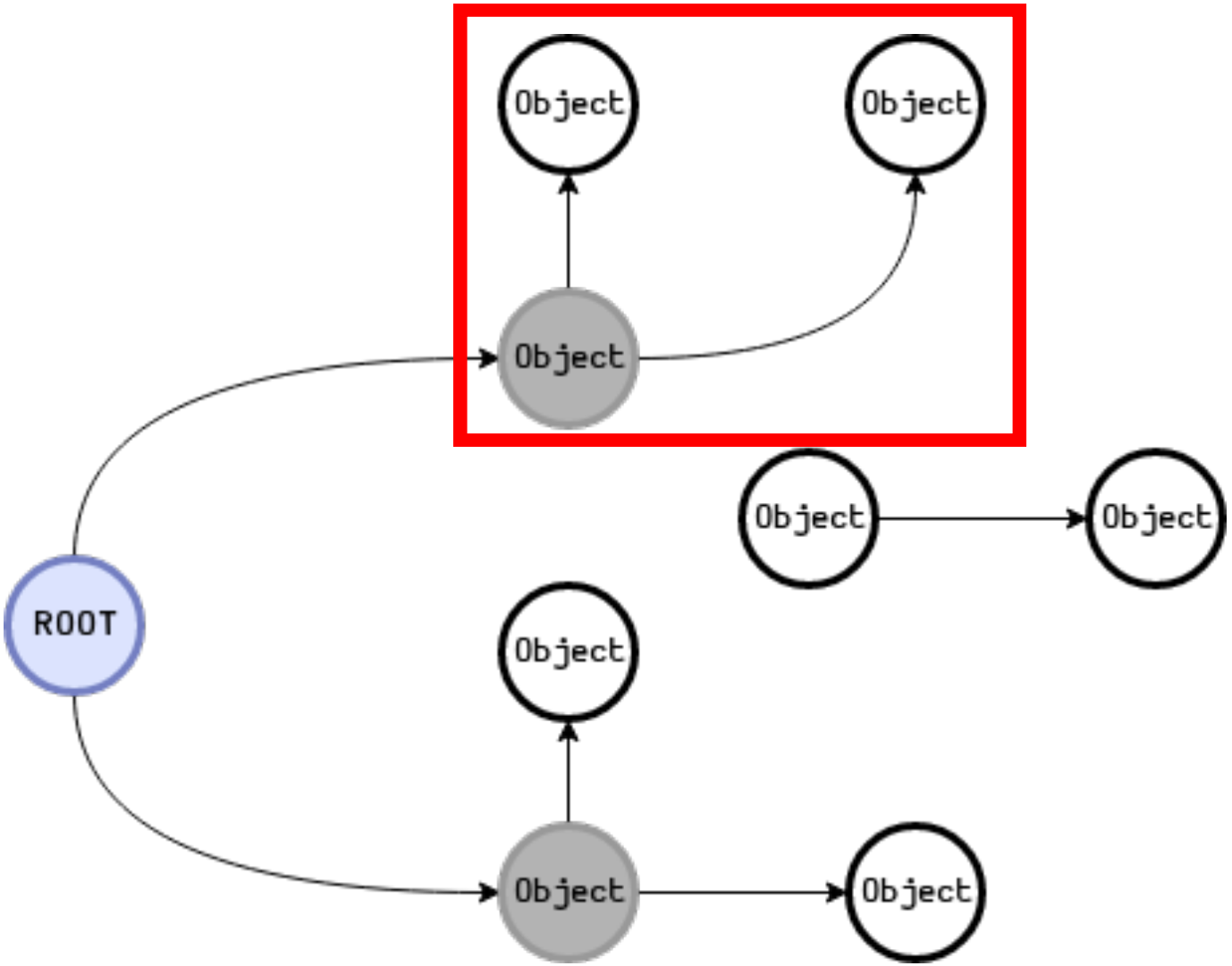
White phase


Object
Object
Object
Object
Object
ROOT
Object
Object
Object

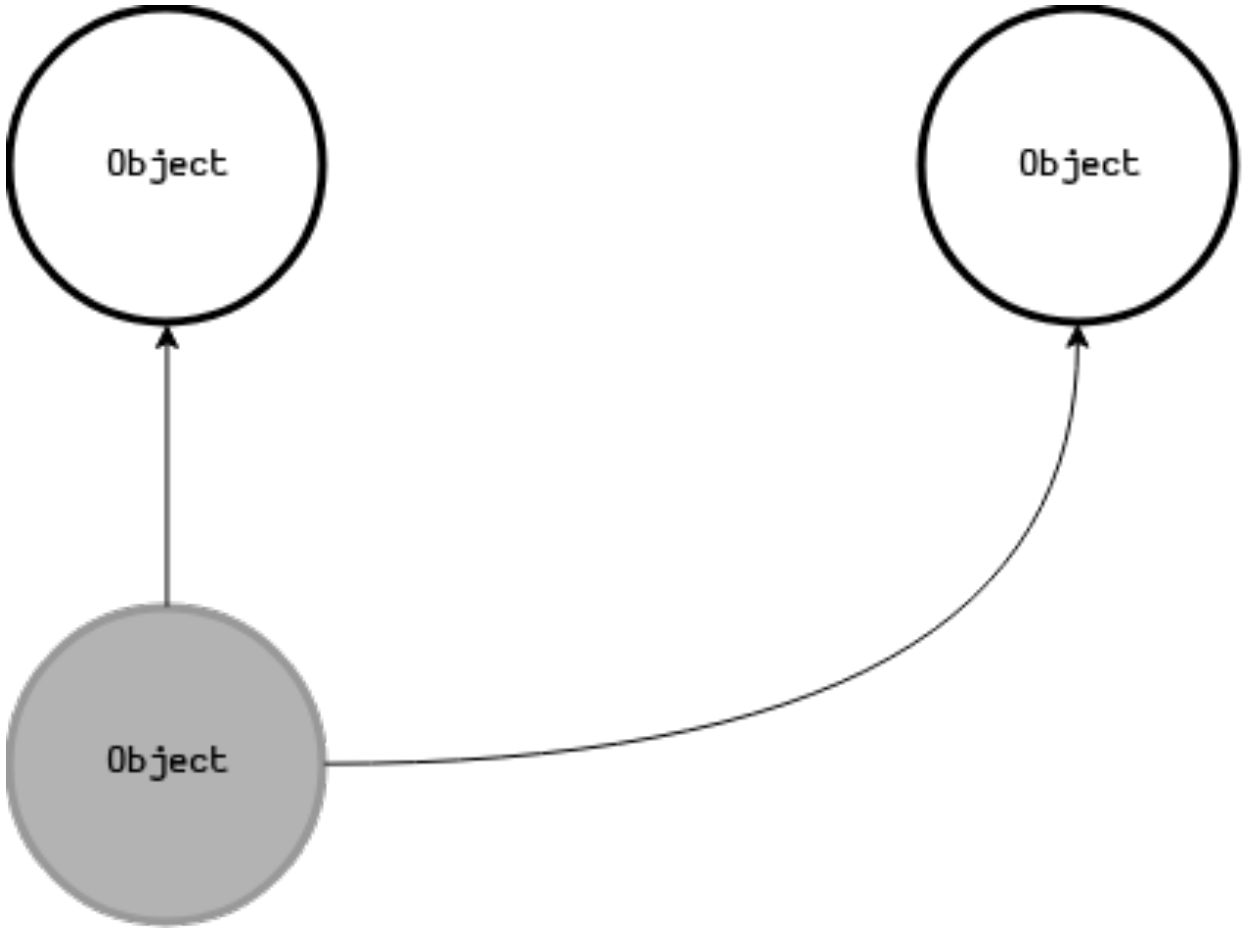
Grey phase


Object
Object
Object
Object
Object
ROOT
Object
Object
Object

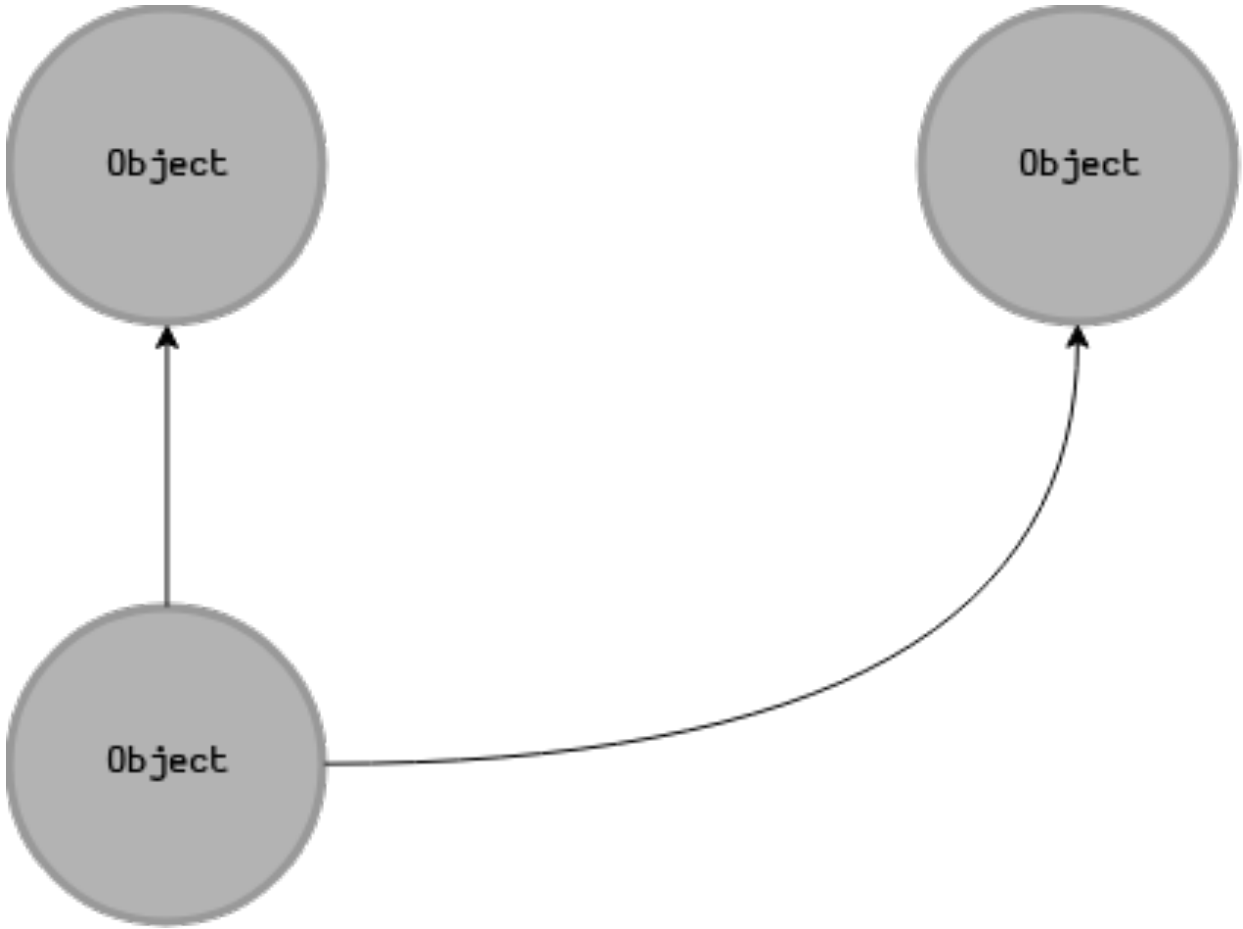
Grey phase


Object
Object
Object
Object
Object
ROOT
Object
Object
Object

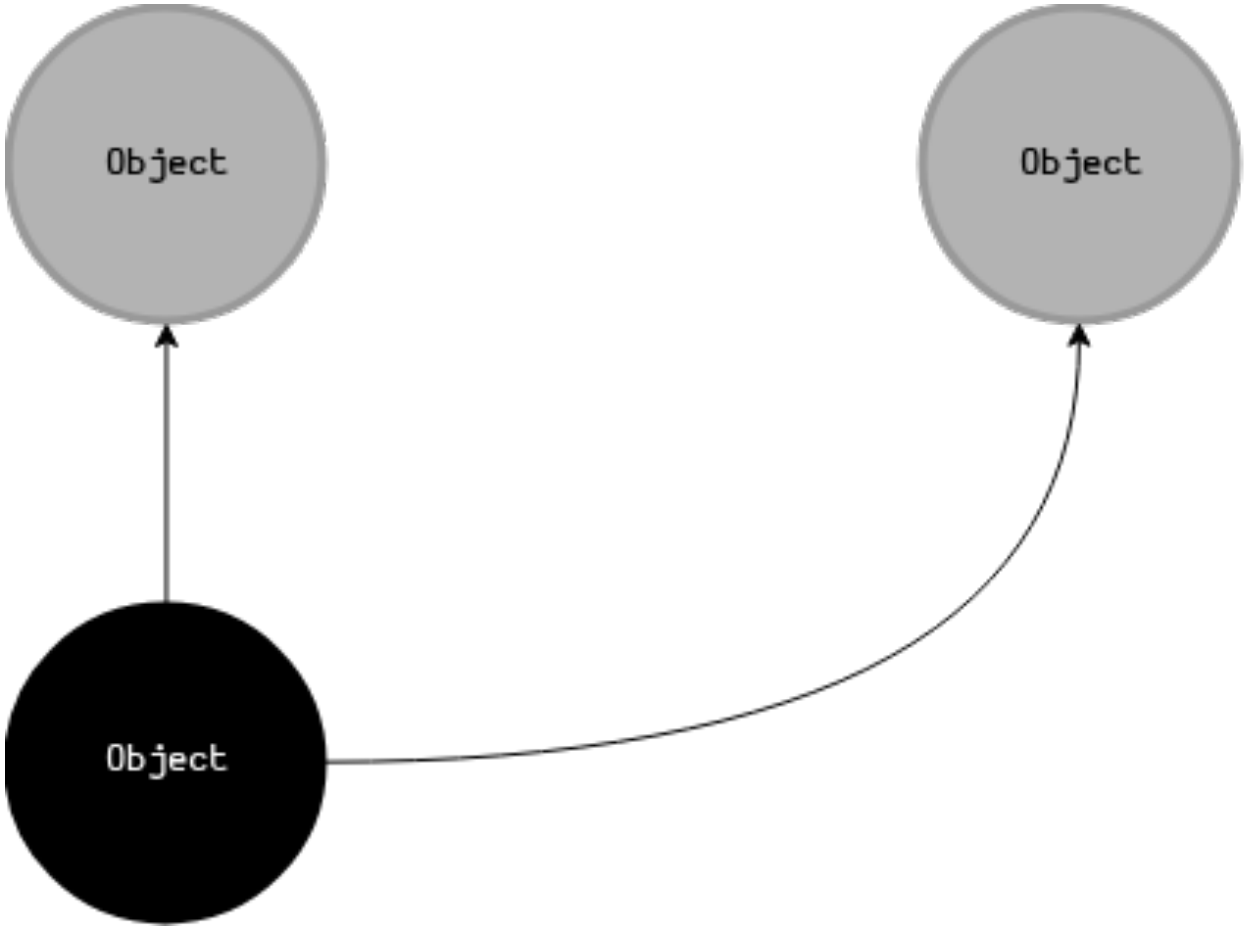
Grey phase



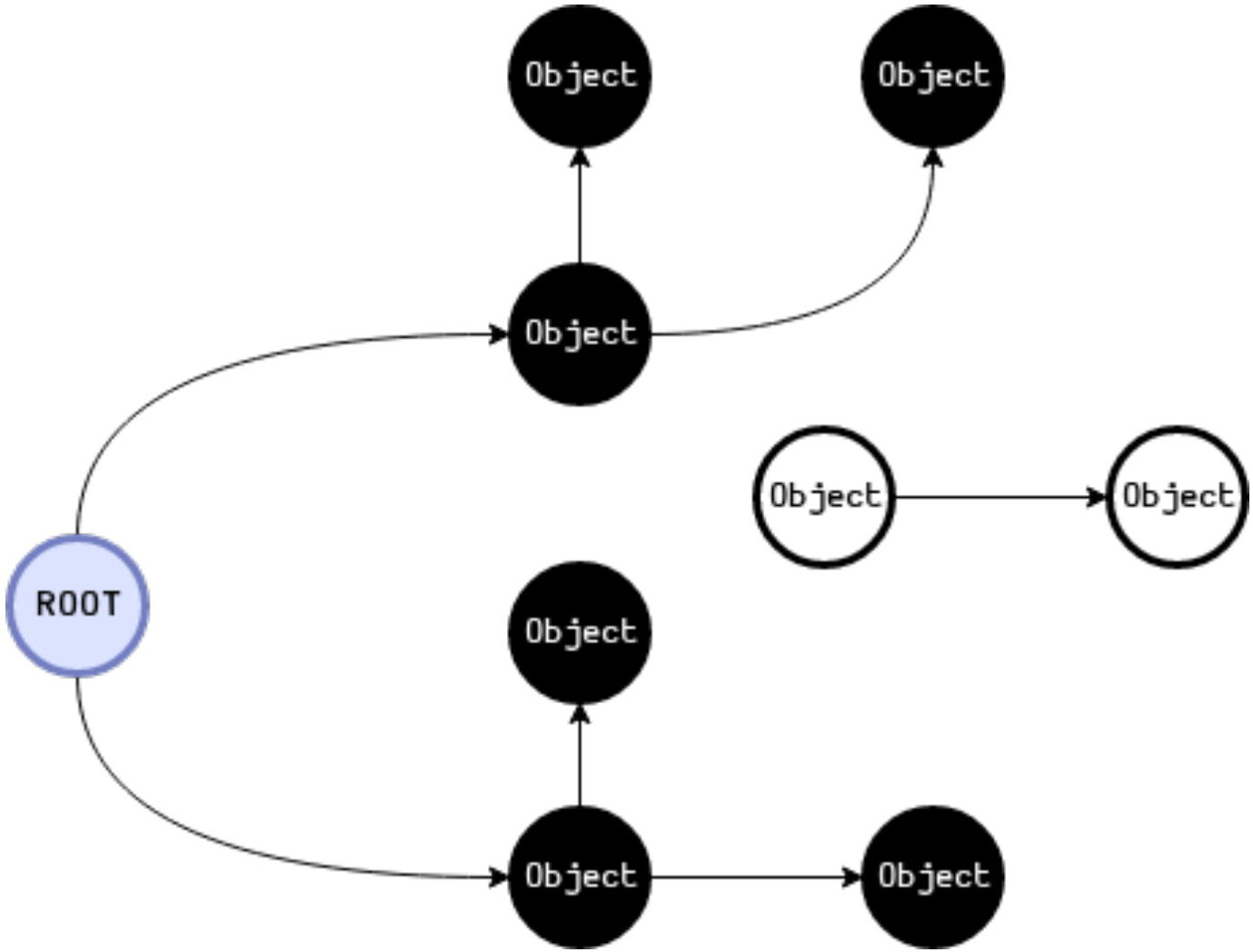
Grey phase



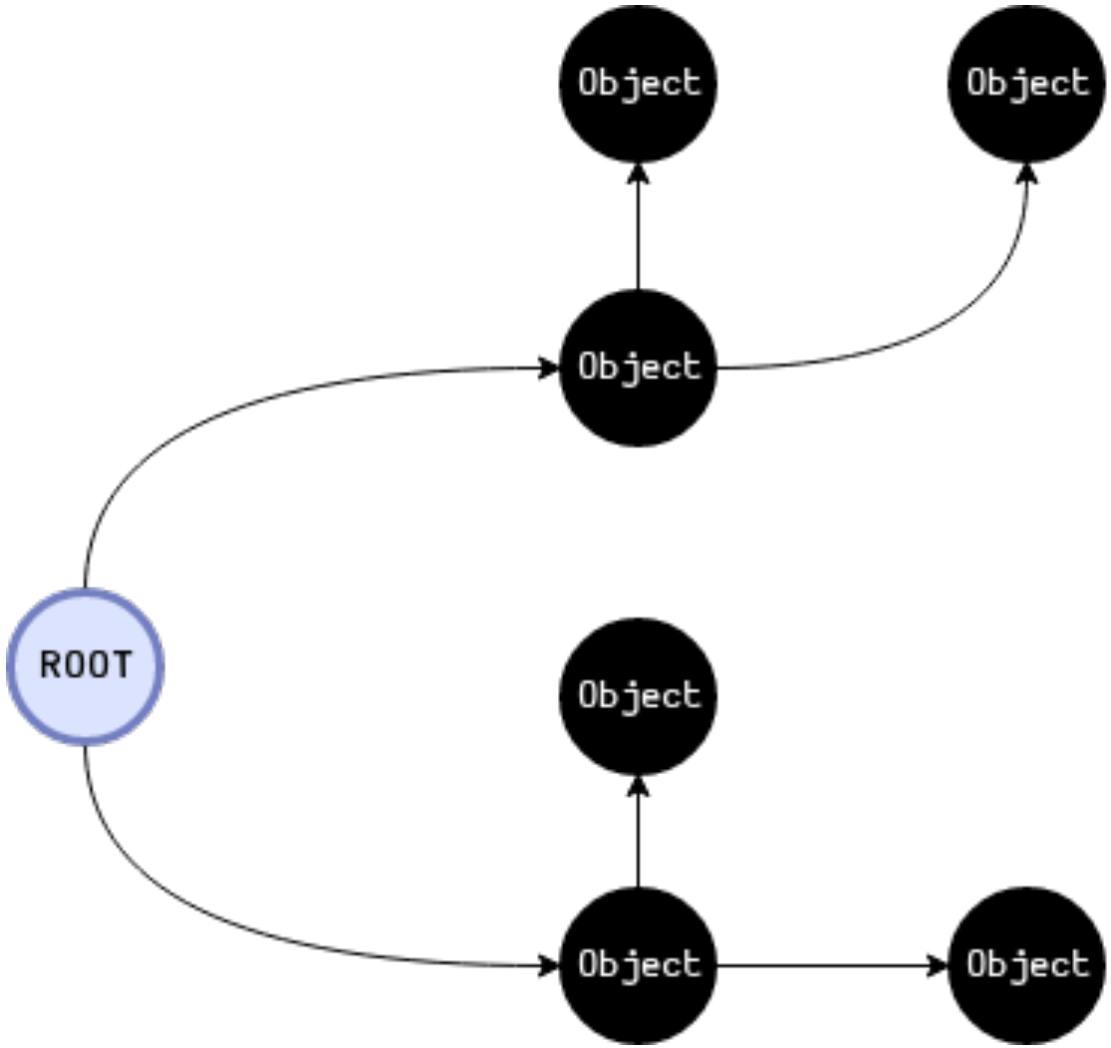
Black phase



Final result

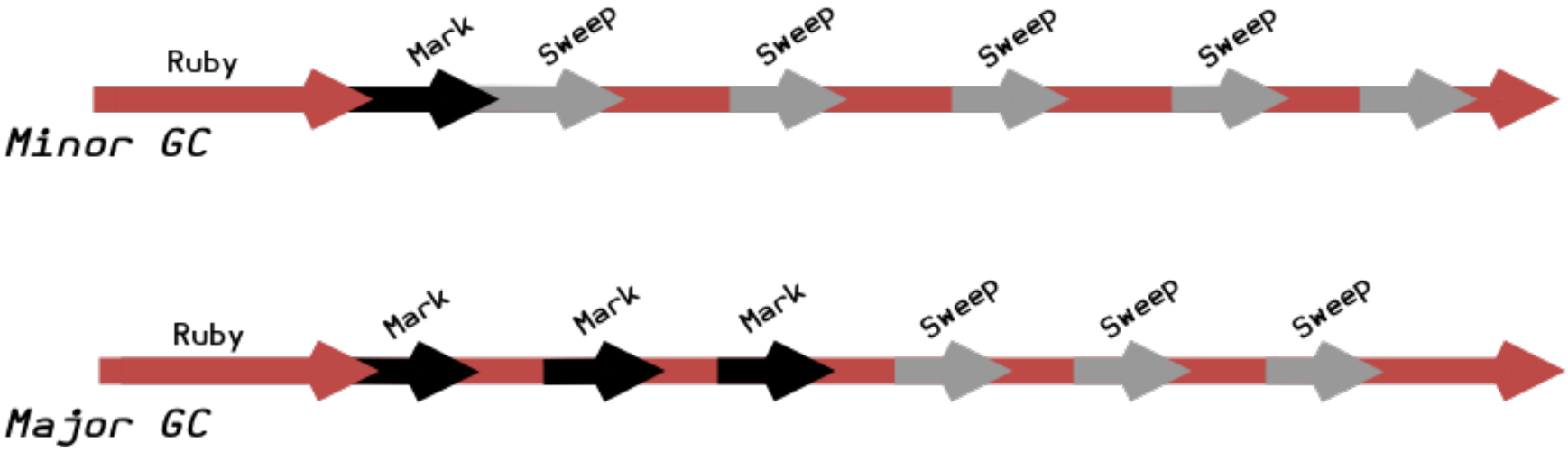


Sweep phase



High throughput
Short pause time

RIncGC


Ruby
Mark
Sweep
Sweep
Sweep
Sweep
Minor GC
Ruby
Mark
Mark
Mark
Sweep
Sweep
Sweep
Major GC

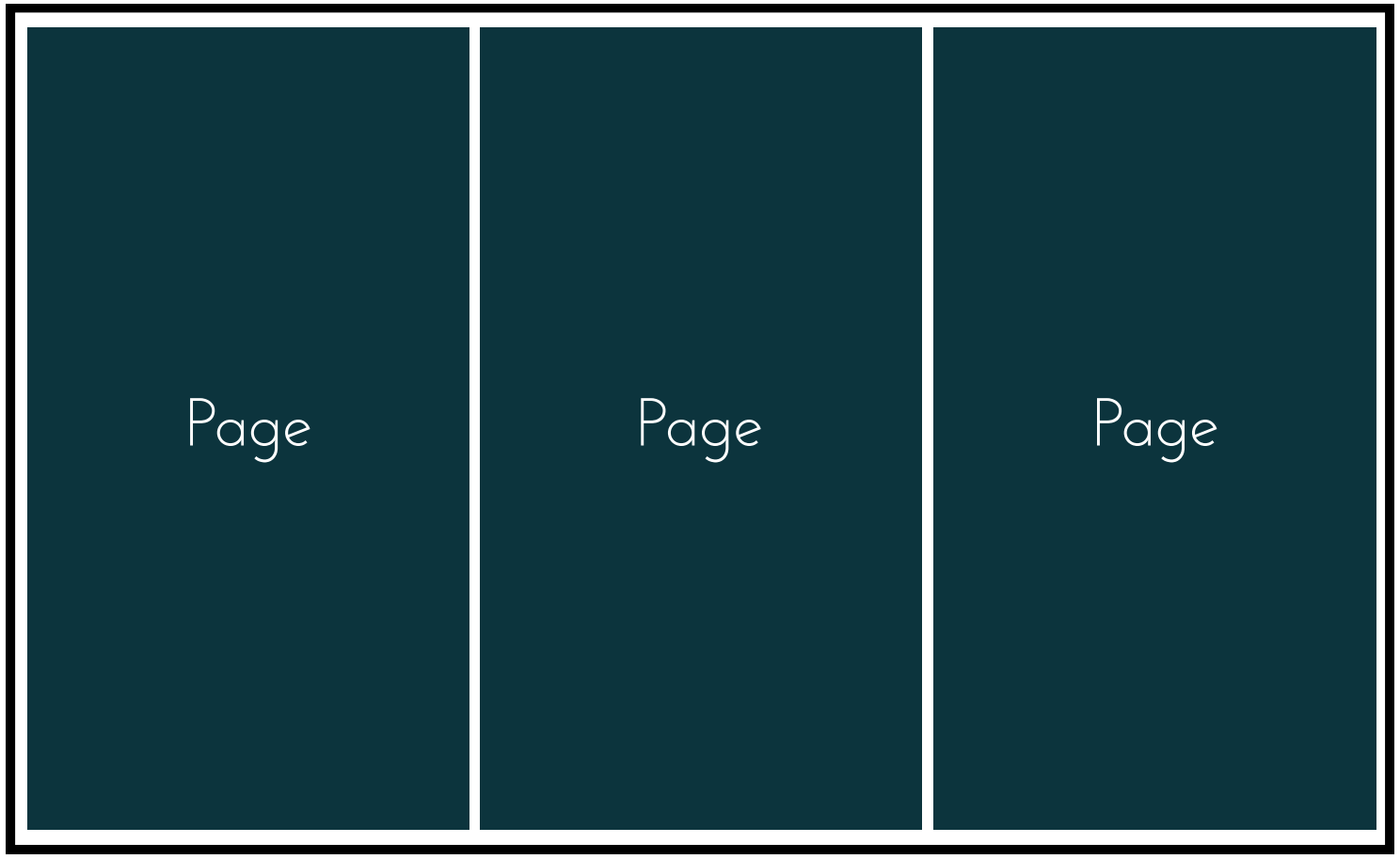
Allocation

```ruby
class Cow
  def initialize(name)
    @name = name
  end
end
```

```ruby
cow = Cow.new("Duquesa")
```

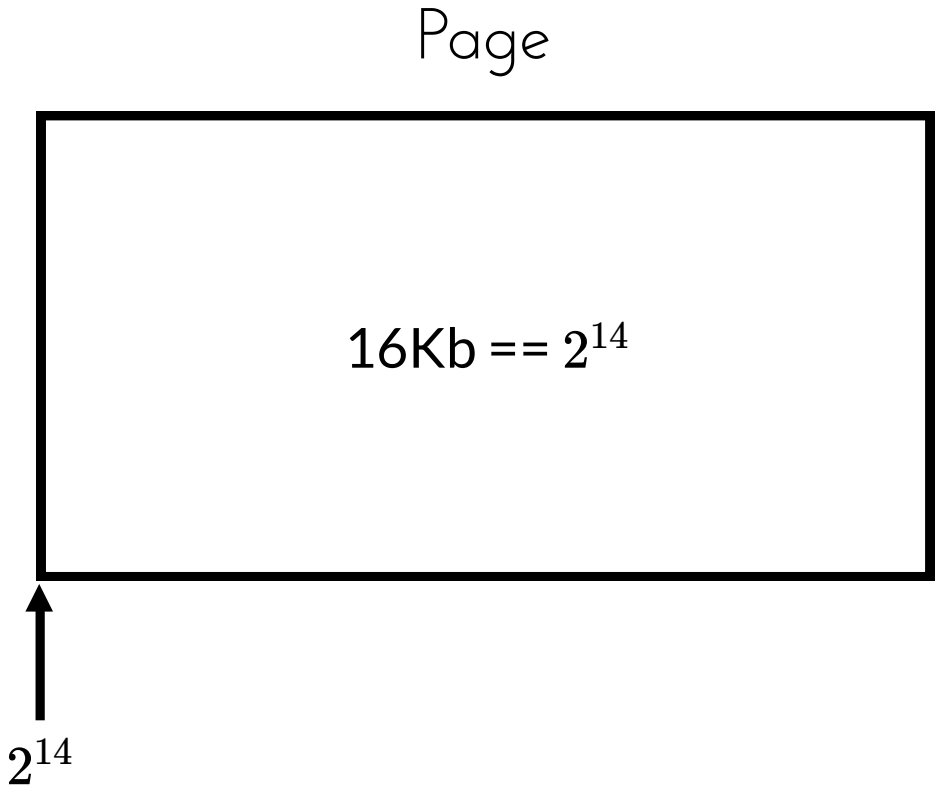
What happens?

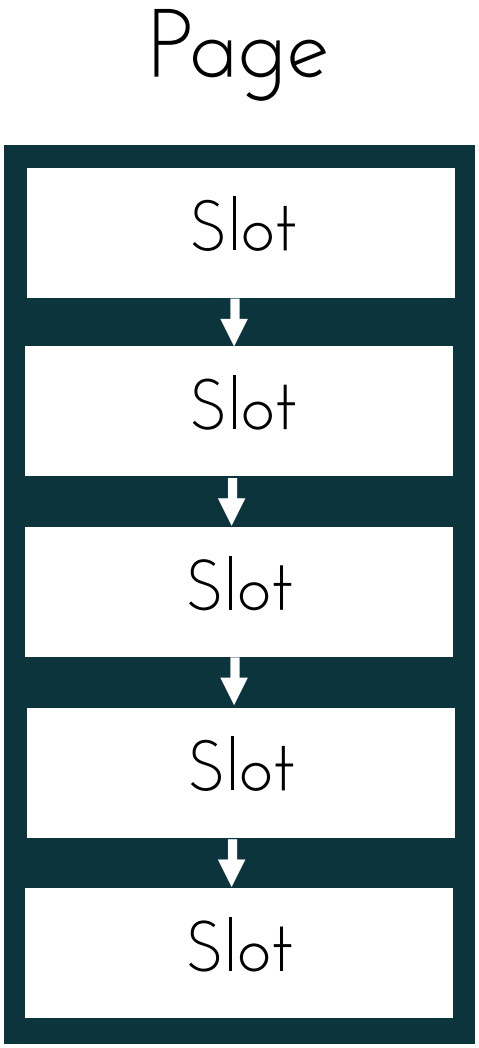
Ruby does not call malloc every single time


Heap
Page
Page
Page

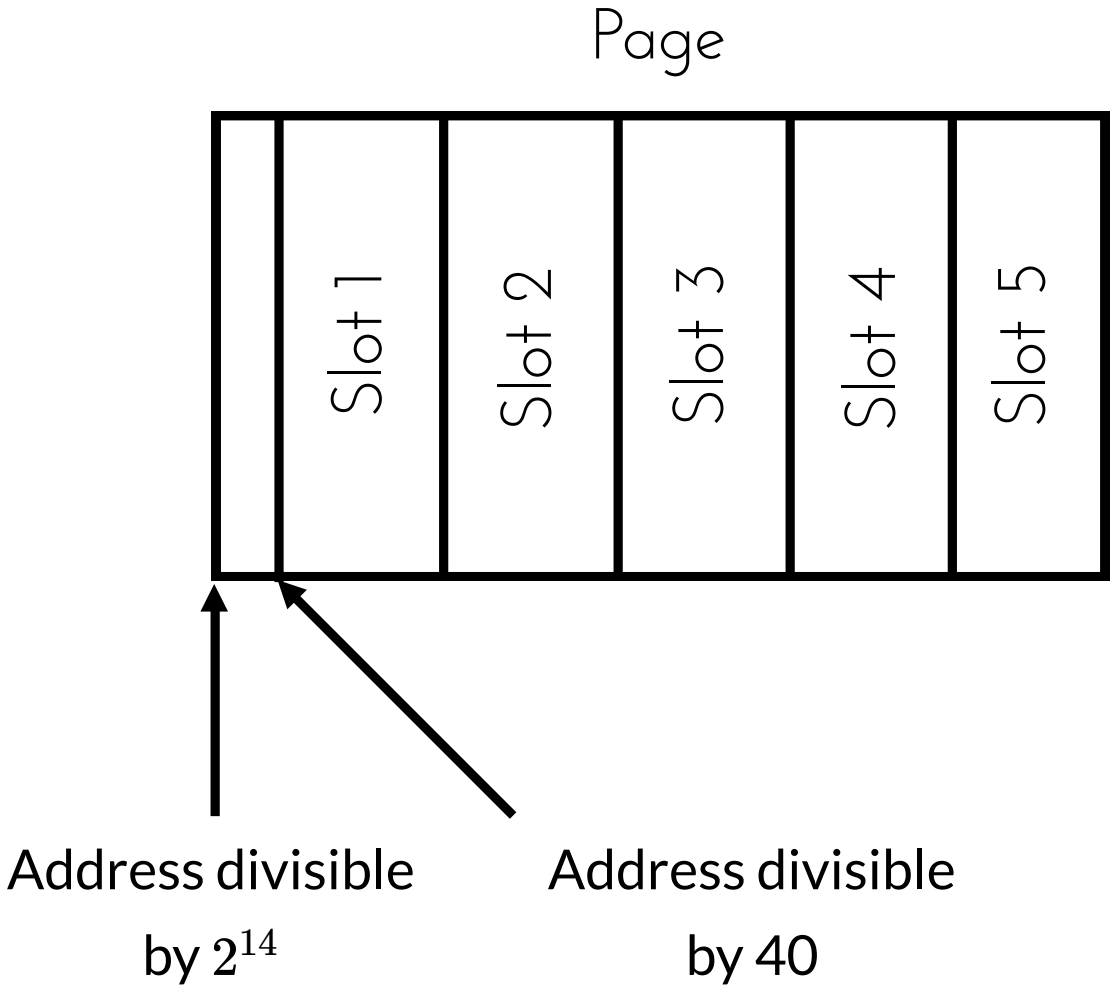
Pages are 16Kb size

Pages are allocated with an aligned malloc


Page
16Kb == 2^{14}
2^{14}


Page
Slot
Slot
Slot
Slot
Slot

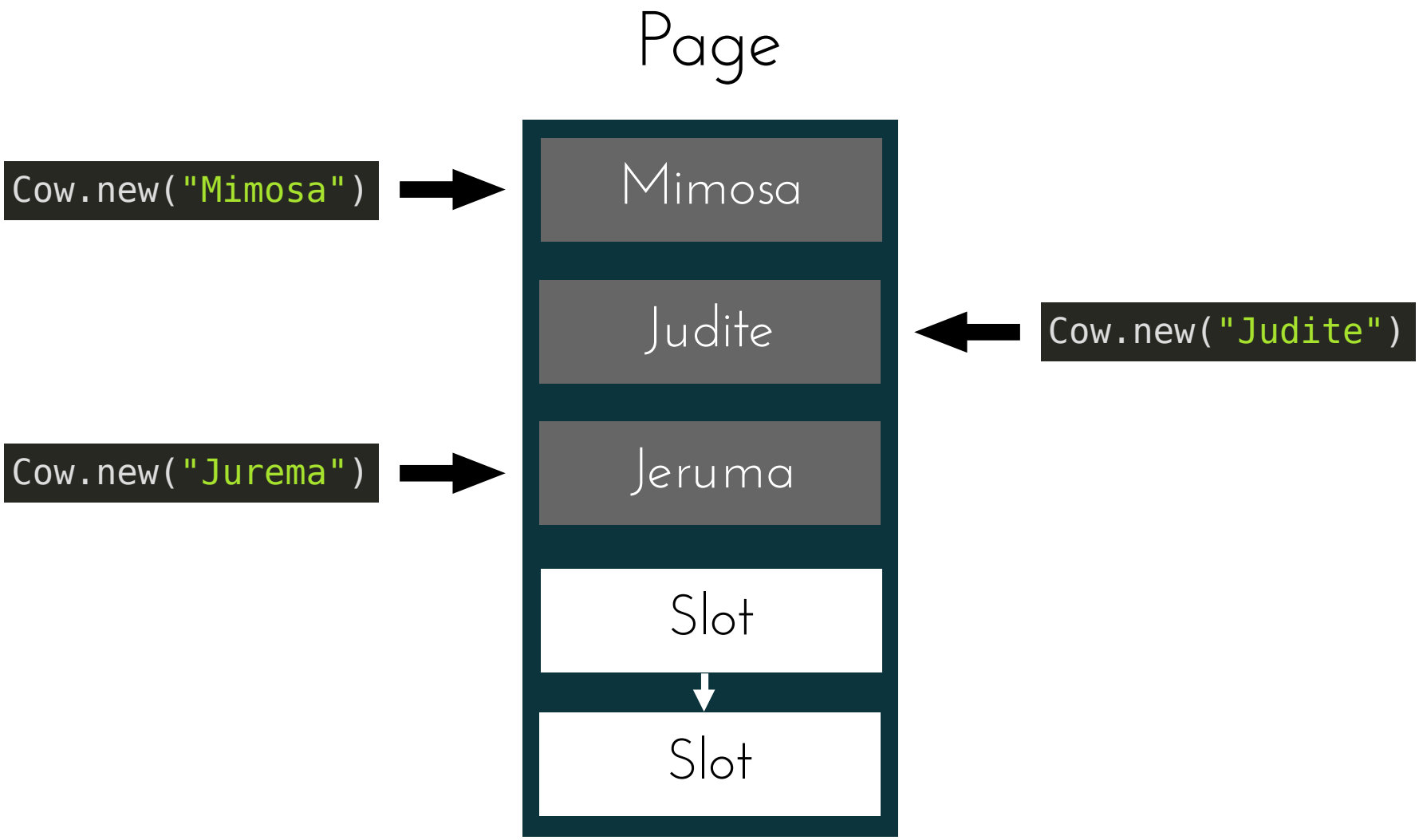
Slots are 40 bytes size


Page
Slot 1
Slot 2
Slot 3
Slot 4
Slot 5
Address divisible by 2^{14}
Address divisible by 40

RVALUE

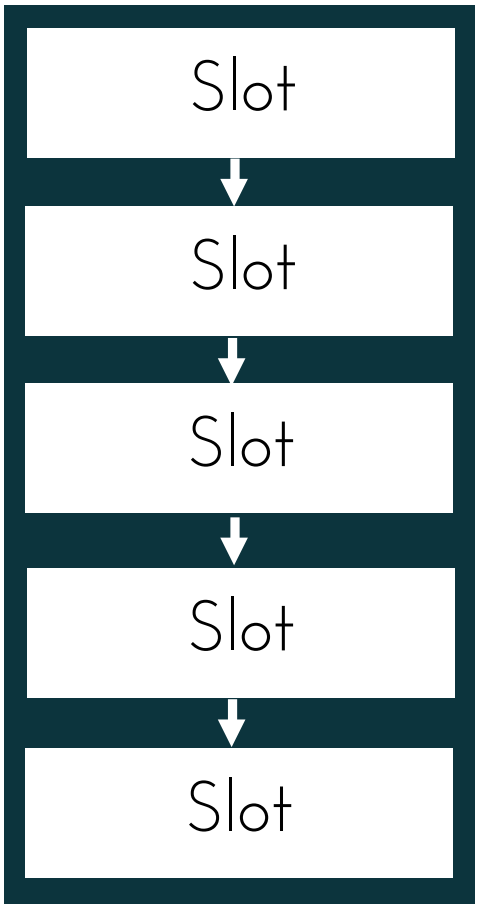
```
typedef struct RVALUE {
  union {
    struct {
      VALUE  flags;    /* always 0 for freed obj */
      struct RVALUE *next;
    } free;
    struct RBasic       basic;
    struct RObject      object;
    struct RClass       klass;
    struct RFloat       flonum;
    struct RString      string;
    struct RArray       array;
    struct RRegexp      regexp;
    struct RHash        hash;
    struct RData        data;
    struct RTypedData   typeddata;
    struct RStruct      rstruct;
    struct RBignum      bignum;
    struct RFile        file;
    struct RMatch       match;
    struct RRational    rational;
    struct RComplex     complex;
    .
    .
  } as;
} RVALUE;
```

10.000 Slots


Page
Cow.new("Mimosa")
Mimosa
Judite
Cow.new("Judite")
Cow.new("Jurema")
Jeruma
Slot
Slot

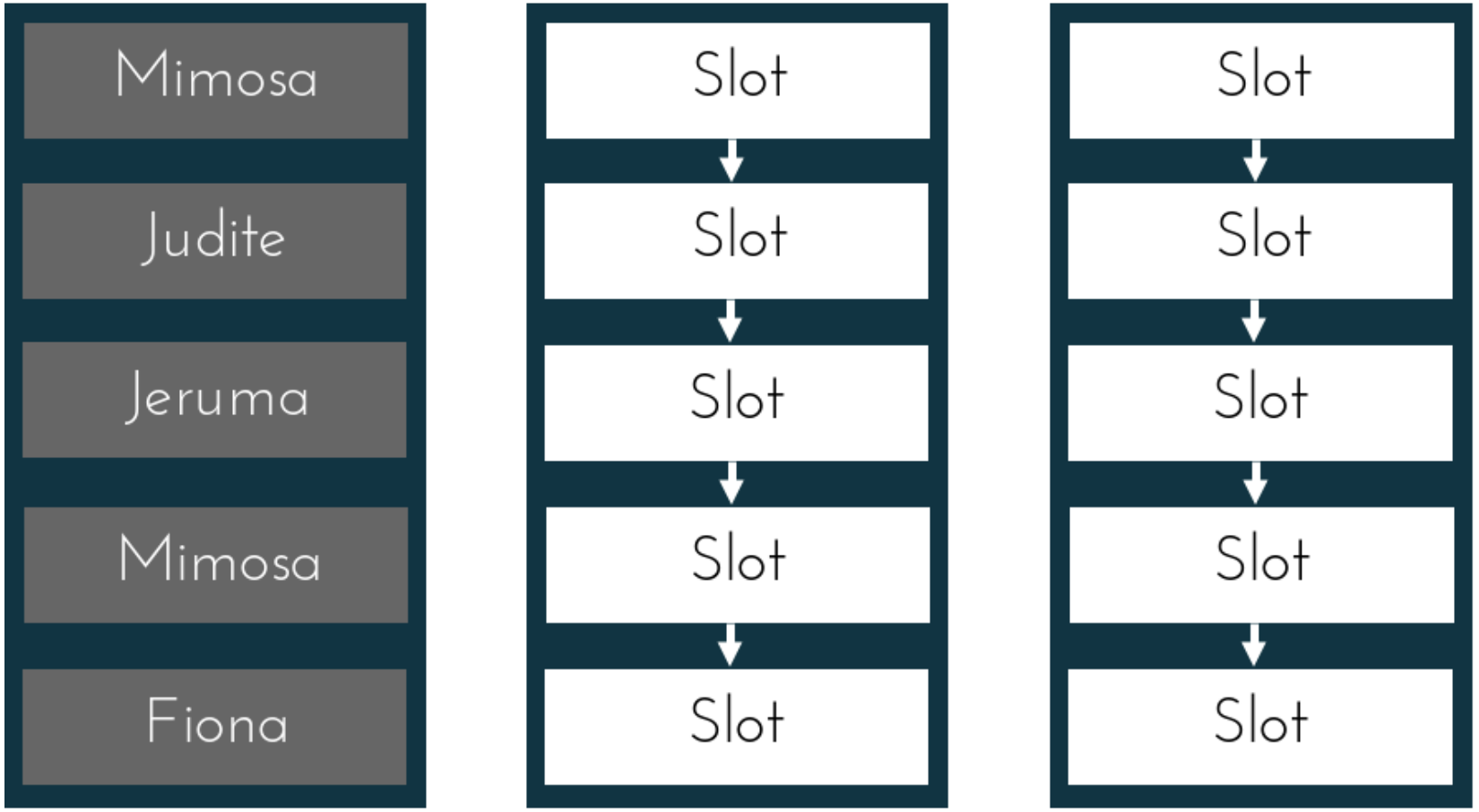
Objects are 40 bytes size

Empty page



Full Page

Full Page


Mimosa
Judite
Jeruma
Mimosa
Fiona
Slot
Slot
Slot
Slot
Slot
Slot
Slot
Slot
Slot
Slot

Not every object is allocated

Tagged pointers

Symbol

Fixnum

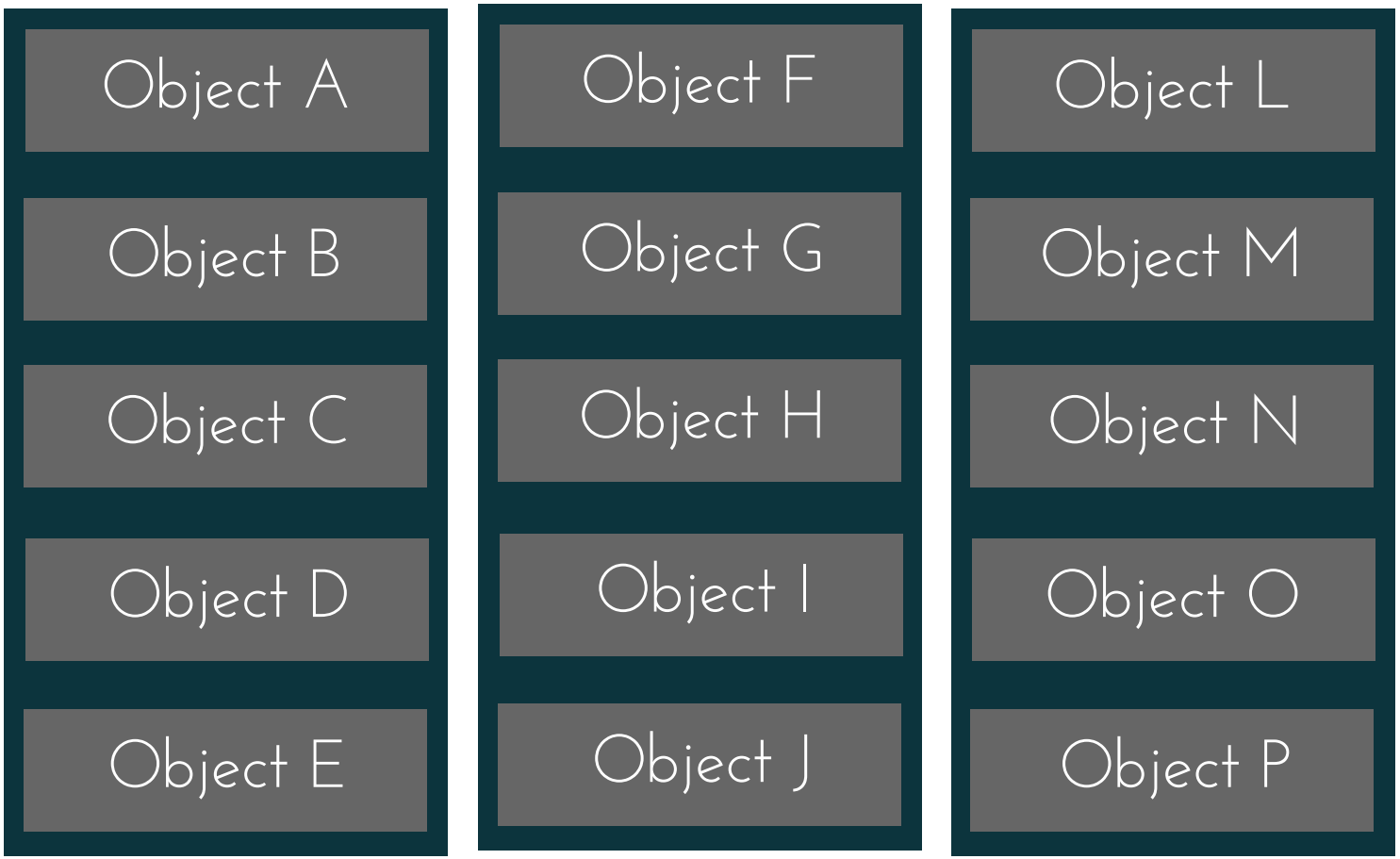
Float

False

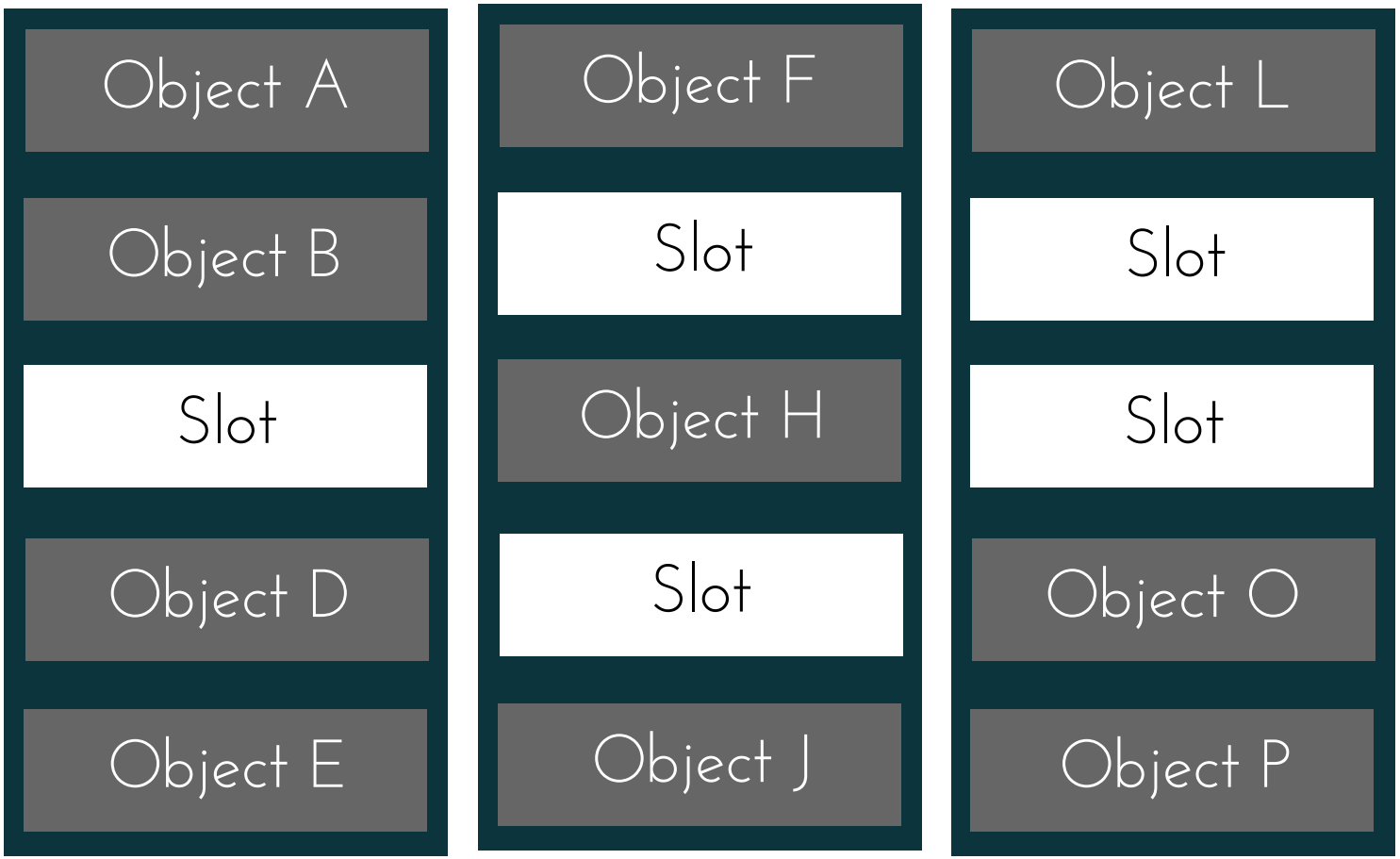
True

Nil

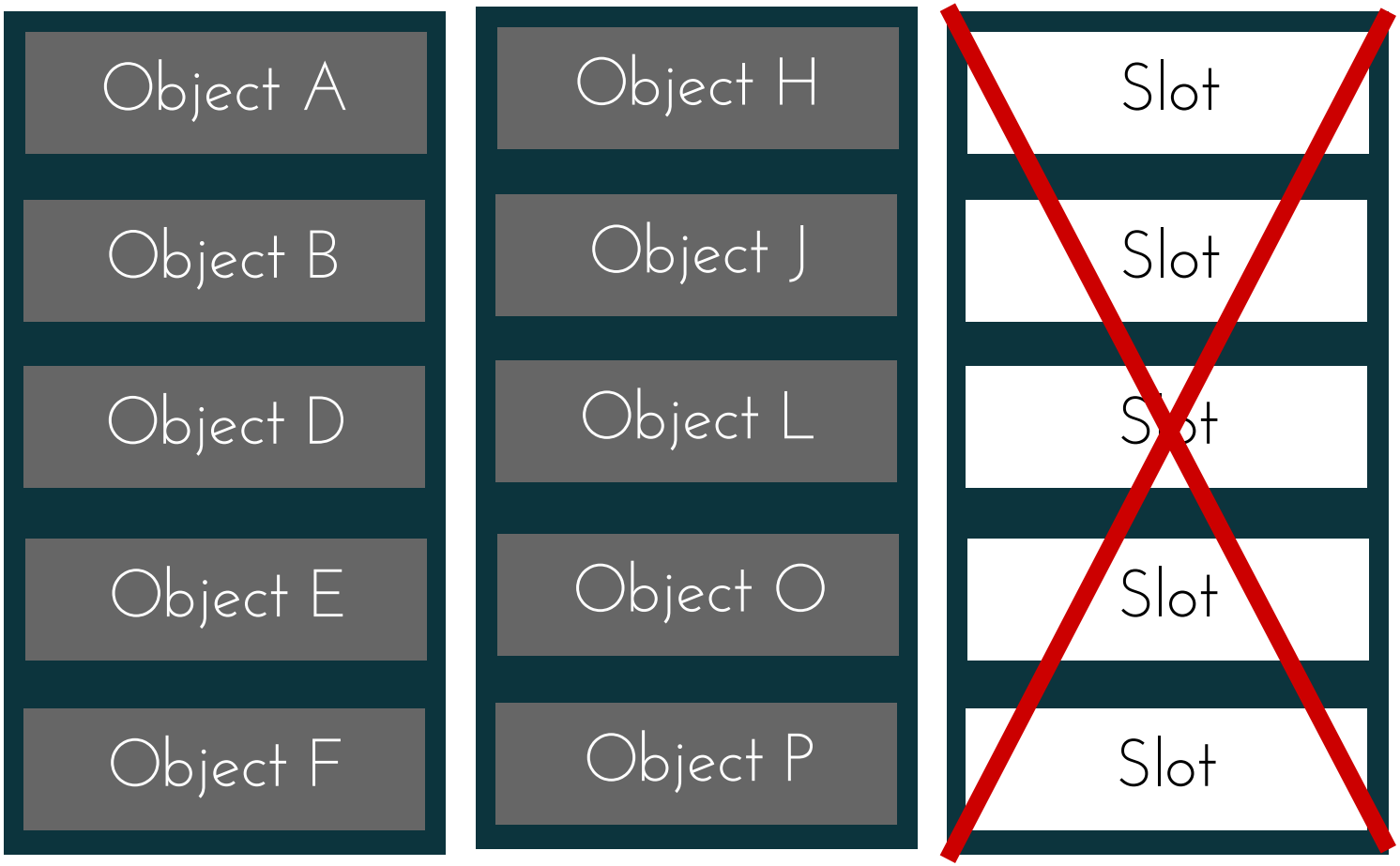
Memory fragmentation



Memory fragmentation



Memory fragmentation



References

Toward 'more' efficient Ruby 2.1

Methods of Memory Management in MRI

Incremental Garbage Collection in Ruby 2.2

Ruby Under a Microscope: An Illustrated Guide to Ruby Internals

That's all folks

Questions?

Thank you!

@alissonbrunosa