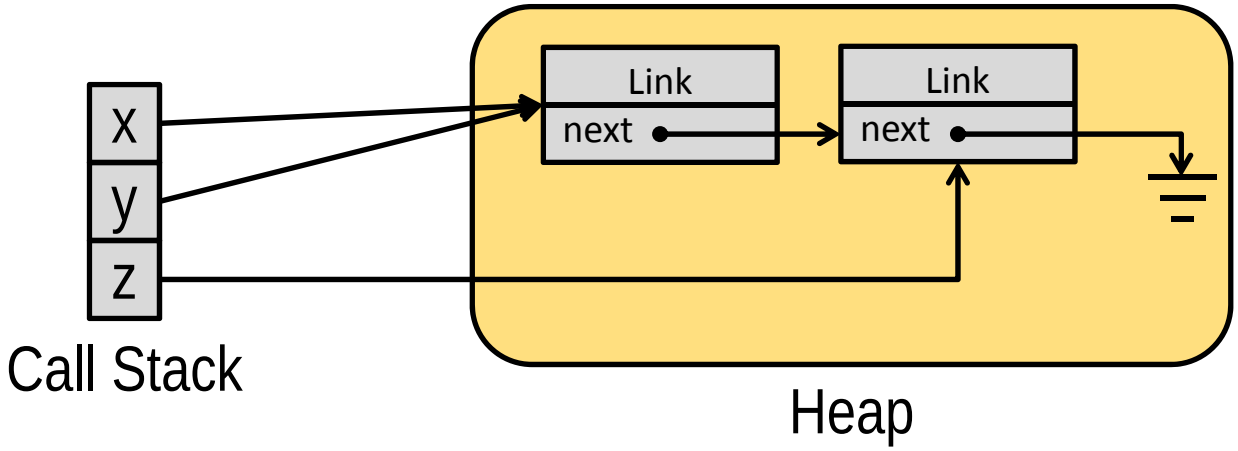
Myths & Realities Performance Impact of Garbage Collection

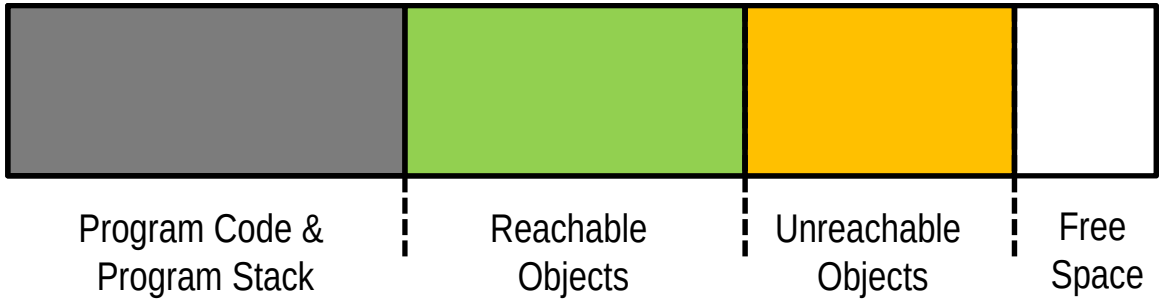
Garbage Collection (recap)



- Notes:
  - Variables x,y and z are **references**
  - Variables x and y **point** to same object
  - Two instances of Link exist in **heap**

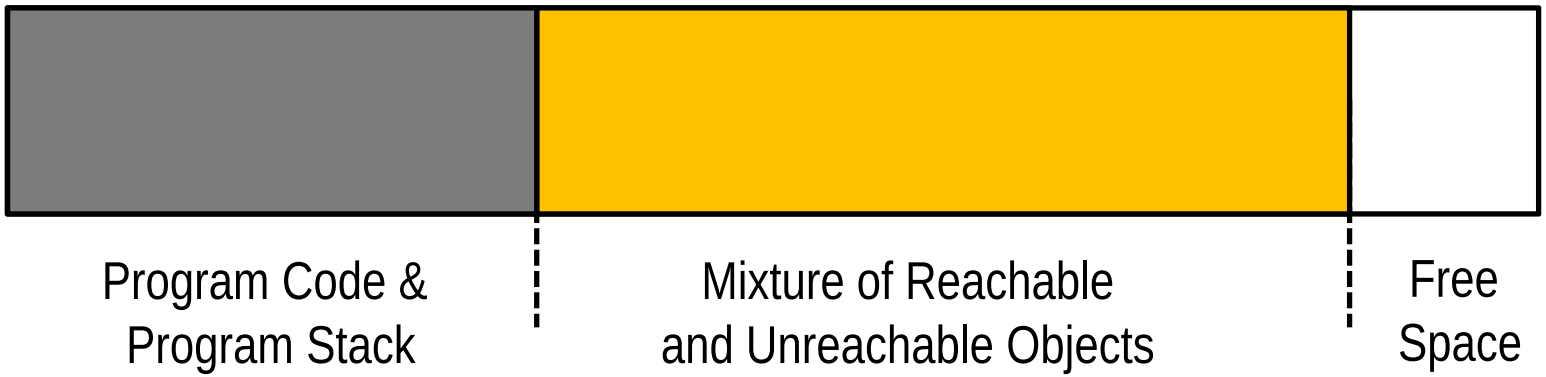
Garbage Collection (recap)

- A rough breakdown of memory usage for a running program:



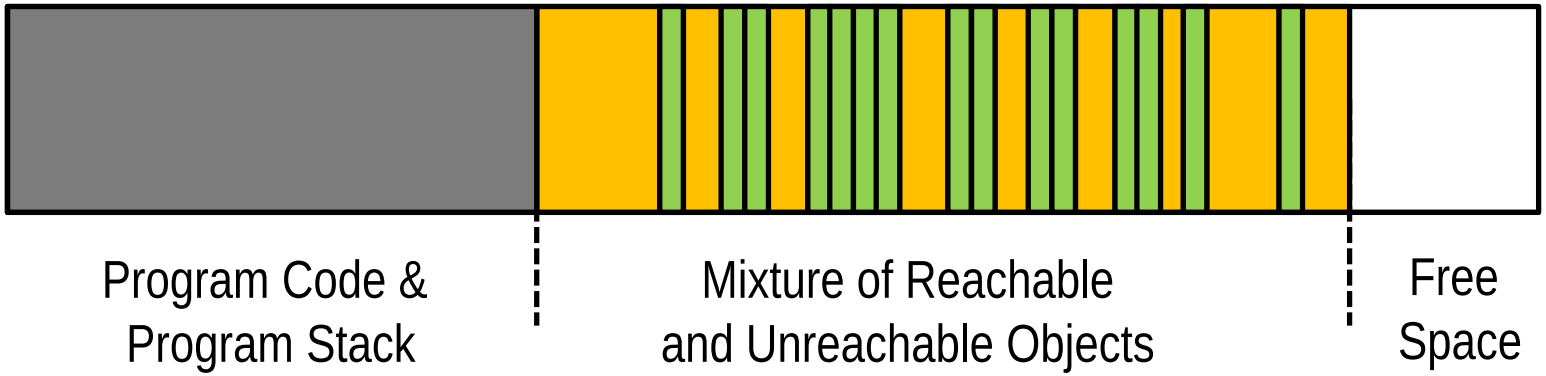
  - A running program has a **finite** amount of memory storage it can use
  - When memory is exhausted, program **halts** with OutOfMemory exception
  - Want to make most **efficient** use of memory …

Garbage Collection (Mark ‘n Sweep)

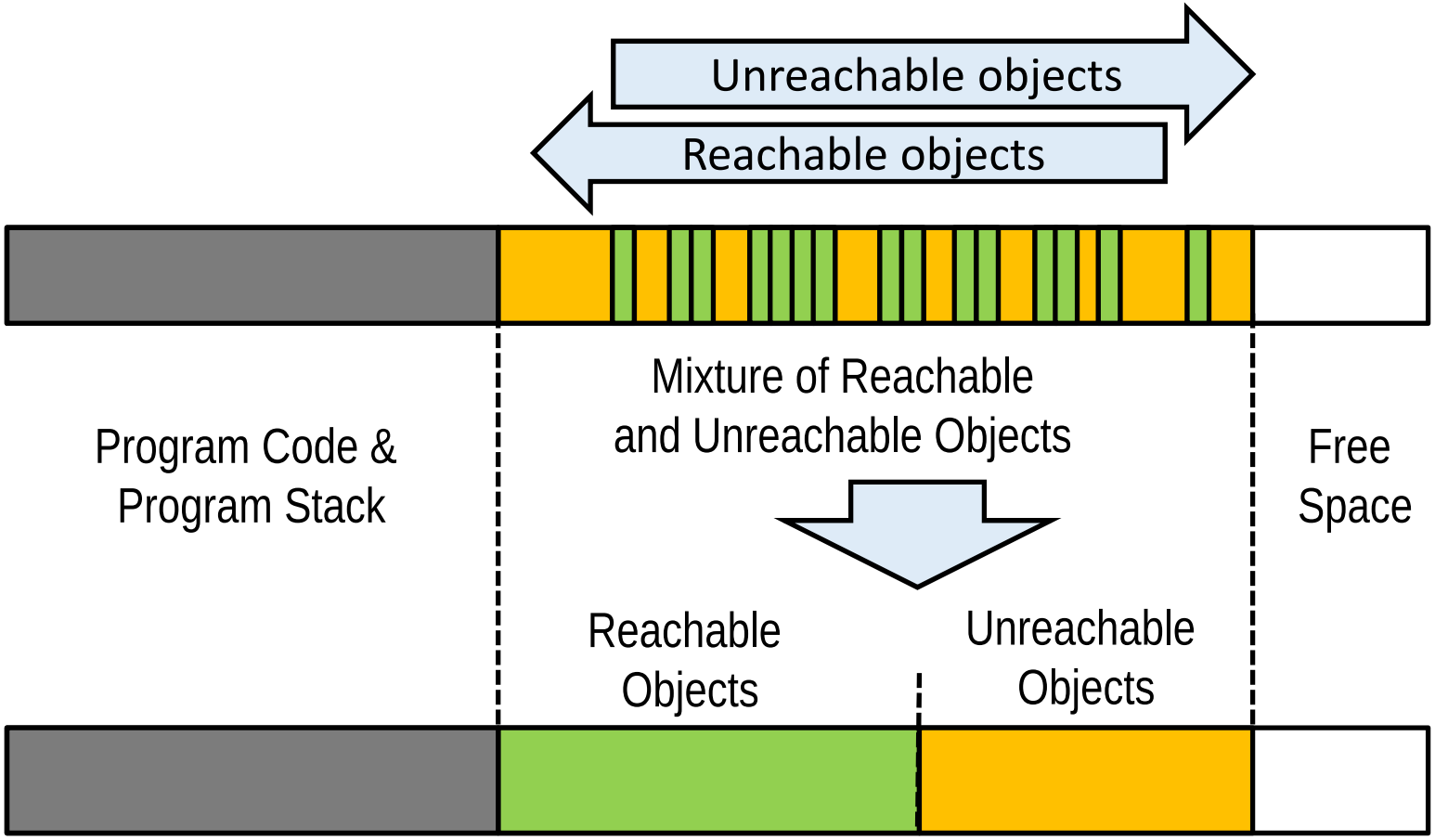


- Notes:
  - During execution, unreachable objects are **mixed up** with reachable objects
  - Must first **identify** unreachable objects, then we can reclaim them
  - **Basic algorithm** for this is called “mark and sweep”
  - Example of “whole heap” algorithm

Mark ‘n Sweep: **Marking Phase**

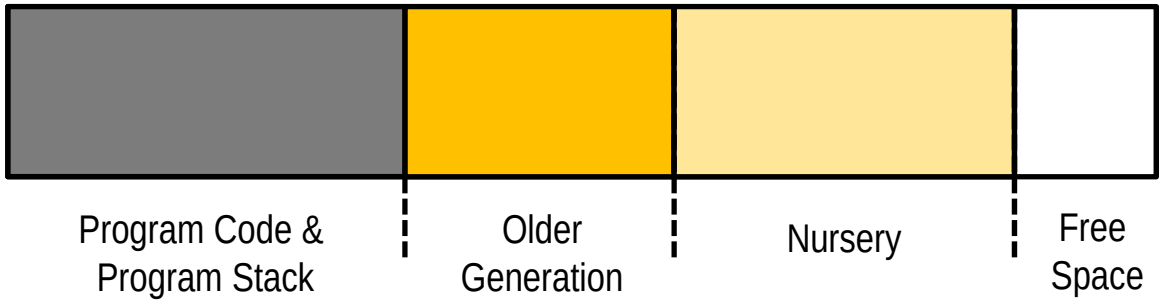


- Notes:
  - Reachable objects are “marked” by **traversing** from object “roots”
  - Could use e.g. **depth-first search** for this
  - Roots are **local** variables and **static** variables



- Marked objects are “swept” to the left
- Unmarked objects are “swept” to the right
- Then can reclaim the unmarked objects

Generational Collection



- **Generational Hypothesis**: *young objects die quickly; old objects live on*
- "Young" objects allocated to **nursery**
- "Old" objects moved into **"older generation(s)"**
- **Frequent** collection of nursery
- **Infrequent** collection of older generation(s)

Some findings

	alloc MB	alloc: min	% GC SemiSpace	% Nur srv	Source Field (p)					Target Object (o = *p)				
					% Read %			Focus		% Read %			Focus	
					Nur	Mat	Imm	Nur	Mat	Nur	Mat	Imm	Nur	Mat
_202_jess	261	17:1	63	1	29	44	27	0.4	69	18	62	20	0.2	97
_228_jack	231	17:1	53	3	25	39	36	0.1	6	21	50	28	0.1	7
_205_raytrace	135	8:1	46	2	19	75	6	0.3	48	18	78	4	0.3	49
_227_mtrt	142	7:1	51	5	20	75	6	0.3	21	19	77	5	0.3	21
_213_javac	185	7:1	29	23	30	46	24	0.4	2	25	55	21	0.3	3
_201_compress	99	6:1	8	0	97	0	3	11.0	3	61	39	0	6.9	712
pseudojbb	216	5:1	21	32	16	59	25	0.2	2	14	72	15	0.1	2
_209_db	82	4:1	11	9	5	69	26	0.3	49	1	89	9	0.1	63
_222_mpegaudio	3	1:1	0	–	–	–	–	–	–	–	–	–	–	–

- **FINDING:** *contiguous allocation benefits locality*
- **FINDING:** *most locality benefits for young objects*
- **FINDING:** *generational collectors generally better than whole heap*