Maps

Thursday, April 16, 2020 7:14 AM

=> keys & values — generic.

=> operations:

- => get (object)
  - => returns val or null
- => put (object)
  - => returns last val or null
- => remove (object)
  - => returns val or null
- => containsKey(object)
- => containsValue(object)
- => clear, size

need to make sure argument is correct type

astro obj -> classification

ex

Map

Earth: planet
Ganymede: moon
Venus: planet
Sirius: star
Aldebaran: star
Andromeda: Galaxy
Pleiades: Star cluster
M67: Star cluster
pluto: dwarf planet
ceres: dwarf planet
Europa: moon

Different Views

=> entrySet: set of entries

→ entrySet: set of entries
→ keySet: set of keys
→ values: collection of values

→ why are these sets?

→ sets must have unique elements (317)
→ keys must be unique
→ these are identifiers
→ thus entries unique
→ values – not always distinct

what does the entrySet look like?

{ {Earth, planet}, {Ganymede: moon},
{Venus, planet}, {Sirius: star},
{Aldebaran: star}, {Andromeda, Galaxy
{Pleiades, star cluster}, {M67, sc},
{pluto: dwarf planet}, {ceres, dwarf},
{Europa: moon} }

→ sets have many operations, but what operations do we use for an entrySet?

→ size
→ contains(Object) → Entry<K, V>
→ ... (Object)

- contains(Object)
- remove(Object)
- iterator
- clear

*do not duplicate work
*already have size, remove, clear for map
→don't do extra work!

What does the key set look like?

{Earth, Ganymede, Venus, Sirius, Aldebaran, Andromeda, Pleiades, M67, Pluto, Ceres, Europa}

*uses map w/o problems
→no need to implement

What does values look like?

[planet, moon, planet, star, star, galaxy, star cluster, star cluster, dwarf p, dwarf p, moon]

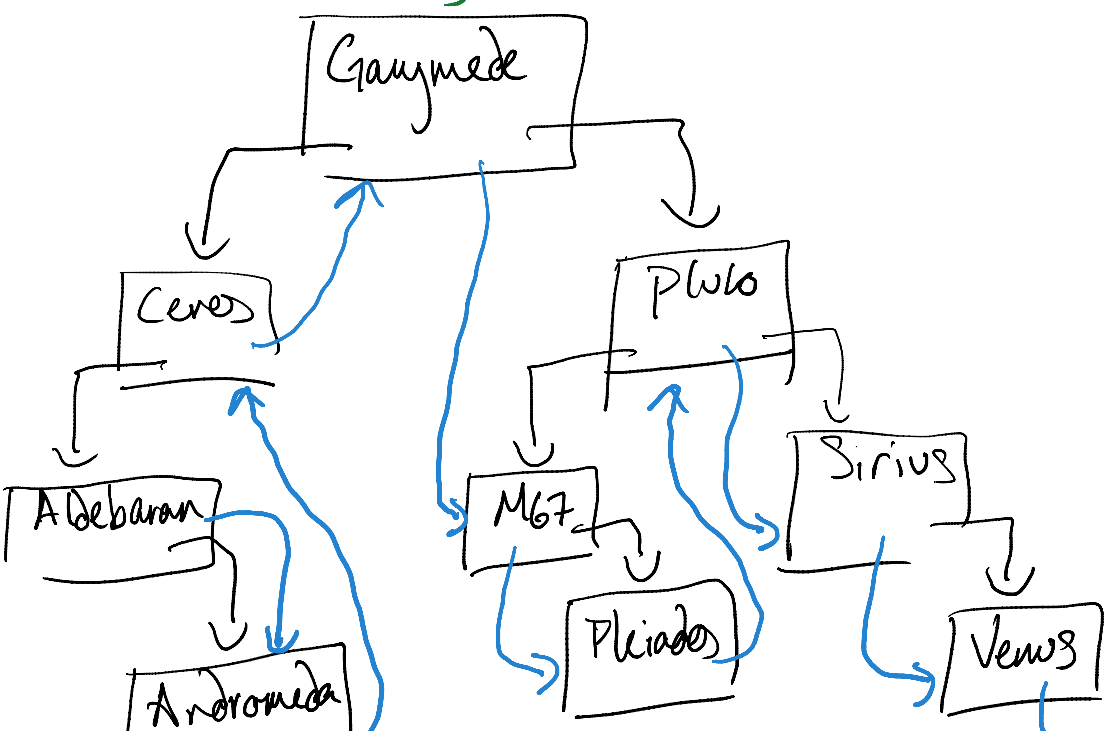
*has collection operations

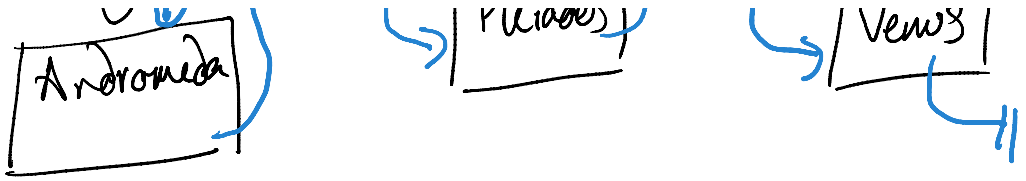
* has collection of - you ~~are~~ know
* Java implementation works - uses entrySet - don't have to write

* if you rewrite every remove, you are making extra work for yourself

→ recognize when you can avoid writing code!

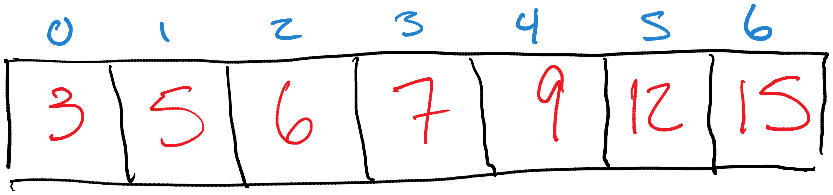
Threading ex

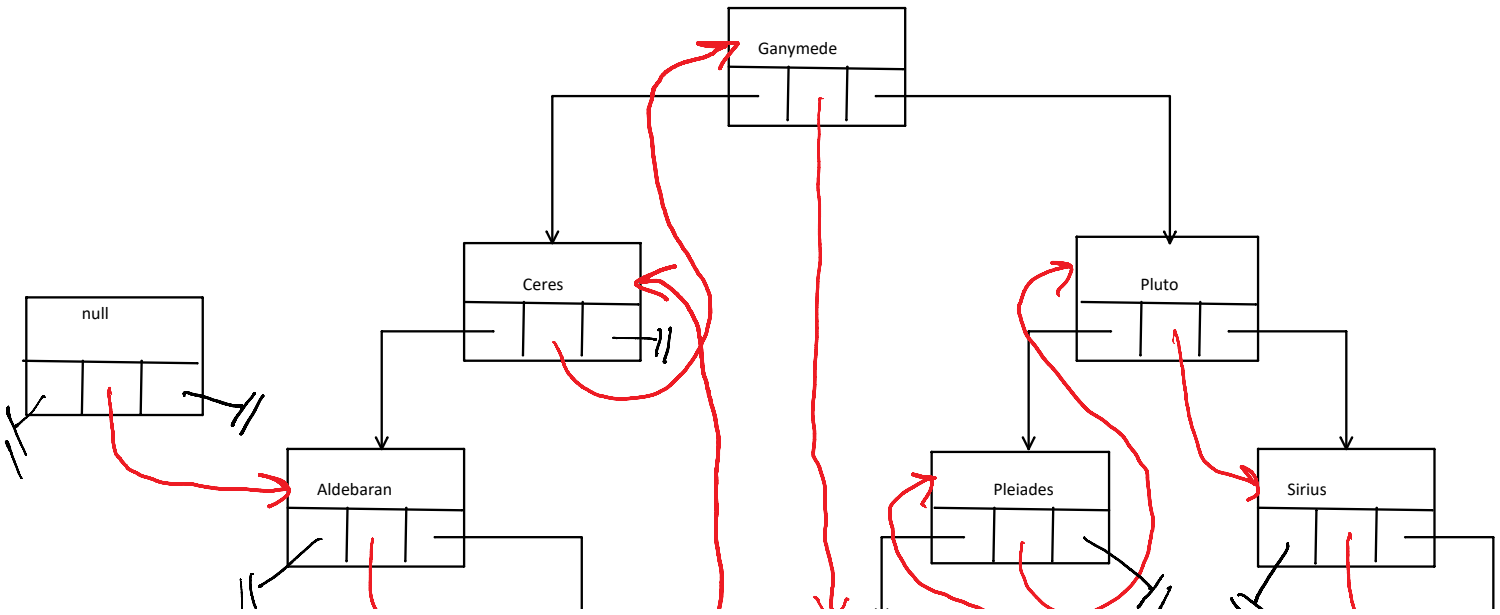


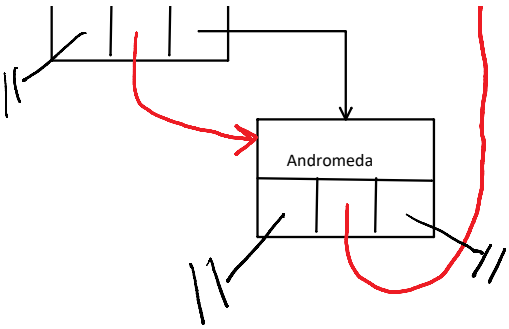


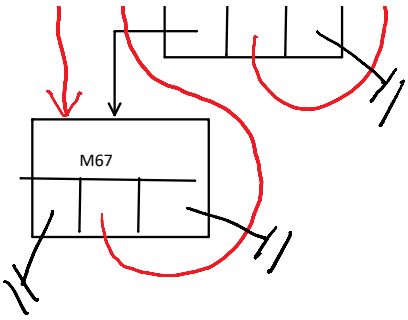
Binary Search

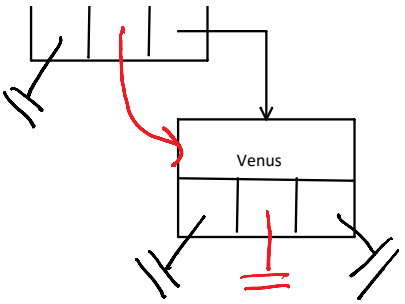
→ Given a low (inclusive) and a hi (exclusive)
→ find & compare mid point






Andromeda


M67


Venus