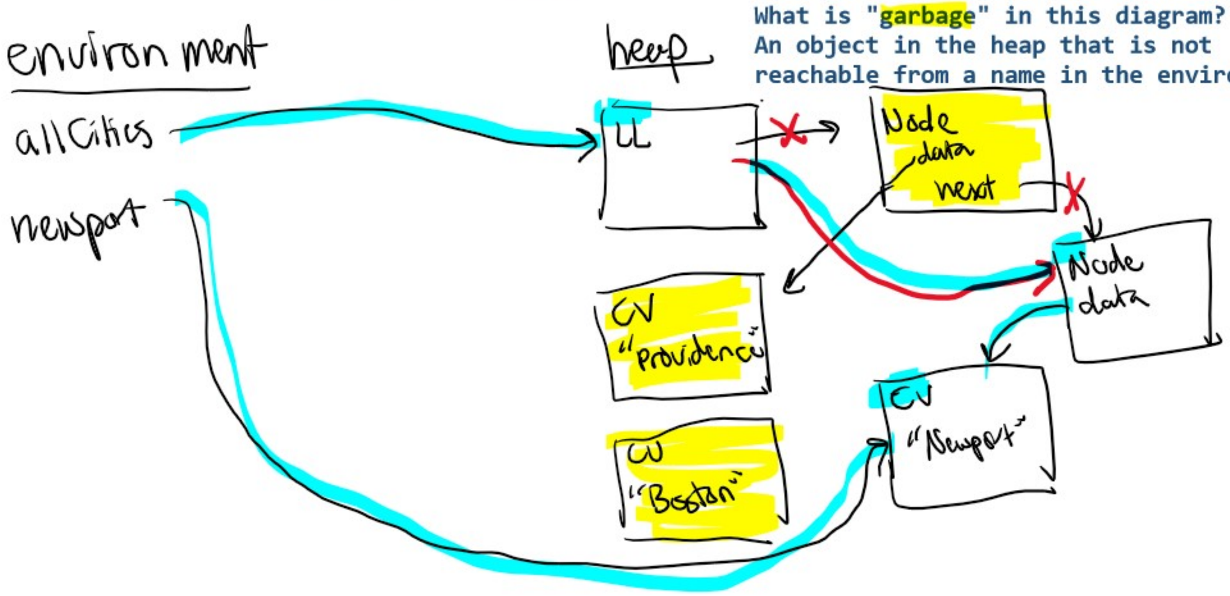
Garbage collection

Monday, October 31, 2022 1:05 PM

- So far, we've been operating under the assumption that we have infinite memory available to us as we run a program
- On a computer, the amount of memory you have is finite
- Programming languages like Java have technology under the hood that manages memory for you

```
public static void main(String[] args) {
  LinkedList<CityVertex> allCities = new LinkedList<>();
  allCities.add(new CityVertex("Providence"));
  new CityVertex("Boston");
  CityVertex newport = new CityVertex("Newport");
  allCities.add(newport);
  allCities.remove(0); // remove item at index 0
}
```

What is "garbage" in this diagram?
An object in the heap that is not reachable from a name in the environment

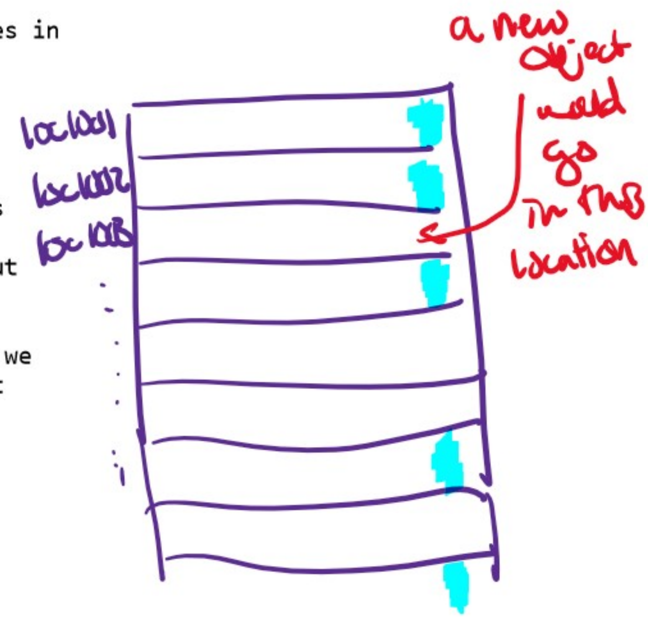


"Garbage collection" frees up memory space by giving the program back the memory locations of garbage objects
An operation that happens during runtime, managed by Java

Memory diagram looks like a graph: objects in the heap and names in the environment are the vertices, and references/pointers (the arrows) are the edges

To run garbage collection:
For each name in the environment, run DFS
If an object from the heap is marked visited by DFS, mark it as reachable in memory
DFS for a given name in the environment finishes once we run out vertices to check

After we've done this for all of the names in the environment, we can start creating objects at locations in memory that were not marked reachable



Code comparison

Monday, October 31, 2022 1:06 PM

posTemps is a temporary piece of data from the programmer perspective, but will not be considered garbage by Java

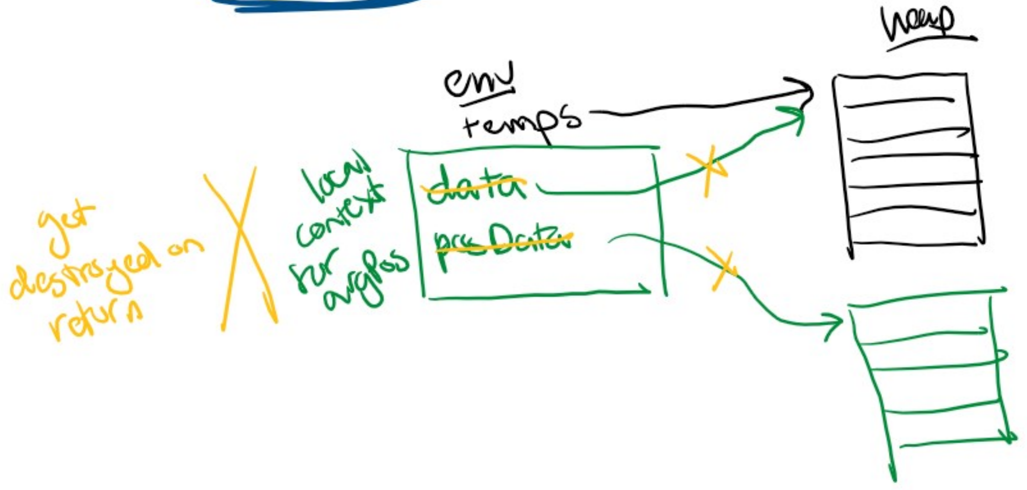
{67, 45, 66, 50}

```
public static void main(String[] args) {
   int[] temps = {67, 45, 0, 66, -21, 50};
   int[] posTemps = Arrays.stream(temps).filter(t -> t > 0).toArray();
   double avgTemp = Arrays.stream(posTemps).sum() / posTemps.length;
   System.out.println(avgTemp);
}
```

```
public static double avgPos(int[] data) {
   int[] posData = Arrays.stream(data).filter(d -> d > 0).toArray();
   return Arrays.stream(posData).sum() / posData.length;
}
```

will have "local context"

```
public static void main(String[] args) {
   int[] temps = {67, 45, 0, 66, -21, 50};
   System.out.println(avgPos(temps));
}
```


heap
env
temps
local context for avgPos
~~data~~
~~posData~~
get destroyed on return