EXAMPLE FOR WORKING W/ HASHMAPS

```
HashMap<Integer, String> offices = new HashMap<Integer, String>();
```

KEY

```
offices.put(210, "Helena");
offices.put(255, "Sun");
```

VALUE

Programmer perspective:

- Each key can only map to one value in the HashMap
- For all operations (get, put, containsKey, …), Java calls hashCode() on the key to get an integer value (the "hash code")—if keys have the same hash code, they will map to the same value
- Java has already has a hashCode() for built-in types (Integer, String, …) If you are making your own class, you should write your own hashCode() method (just like equals())

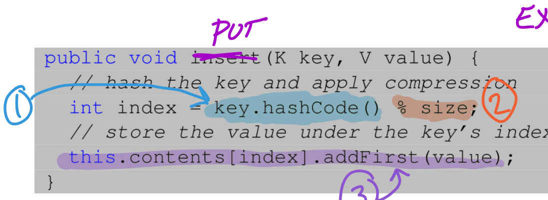
IMPLEMENTATION PERSPECTIVE

Example: what if we want to add some elements:
put(250, "A");
put(255, "B");
put(230, "C");

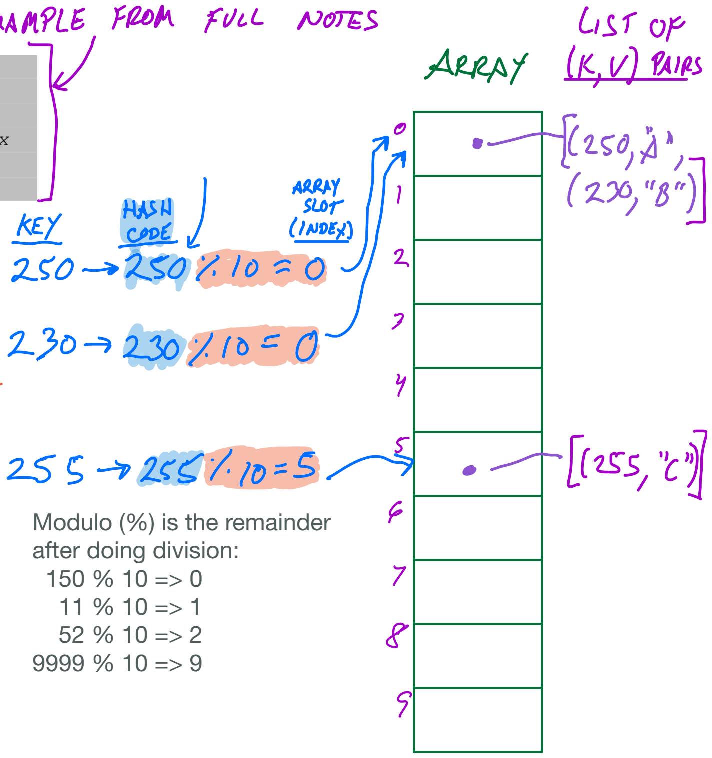
What happens inside the hash table?
(ie, hidden from the programmer)

INSIDE A HASH TABLE: INSERTI VALUES (PUT)

PUT



```
public void insert(K key, V value) {
  // hash the key and apply compression
  int index = key.hashCode() % size;
  // store the value under the key's index
  this.contents[index].addFirst(value);
}
```



1. Use hashCode() to get a unique hash value for this key. hashCode() always returns an int. (If the key is an Integer, hashCode() just returns the integer itself—like you see here.)

2. Since there are more possible hash values than array slots, we "compress" the hash value to pick or a slot for it in the array. Usually we use modulo, ie: hash % size

   Compression means that more than one key may occupy to the same array slot, even when their hashCodes are different values

3. Why does this work? Each array slot contains a list or (key, value) pairs that were mapped to that slot.

   To add an element to the hash map, put() (also called insert()) adds the new element to this list.

Modulo (%) is the remainder after doing division:
150 % 10 => 0
11 % 10 => 1
52 % 10 => 2
9999 % 10 => 9

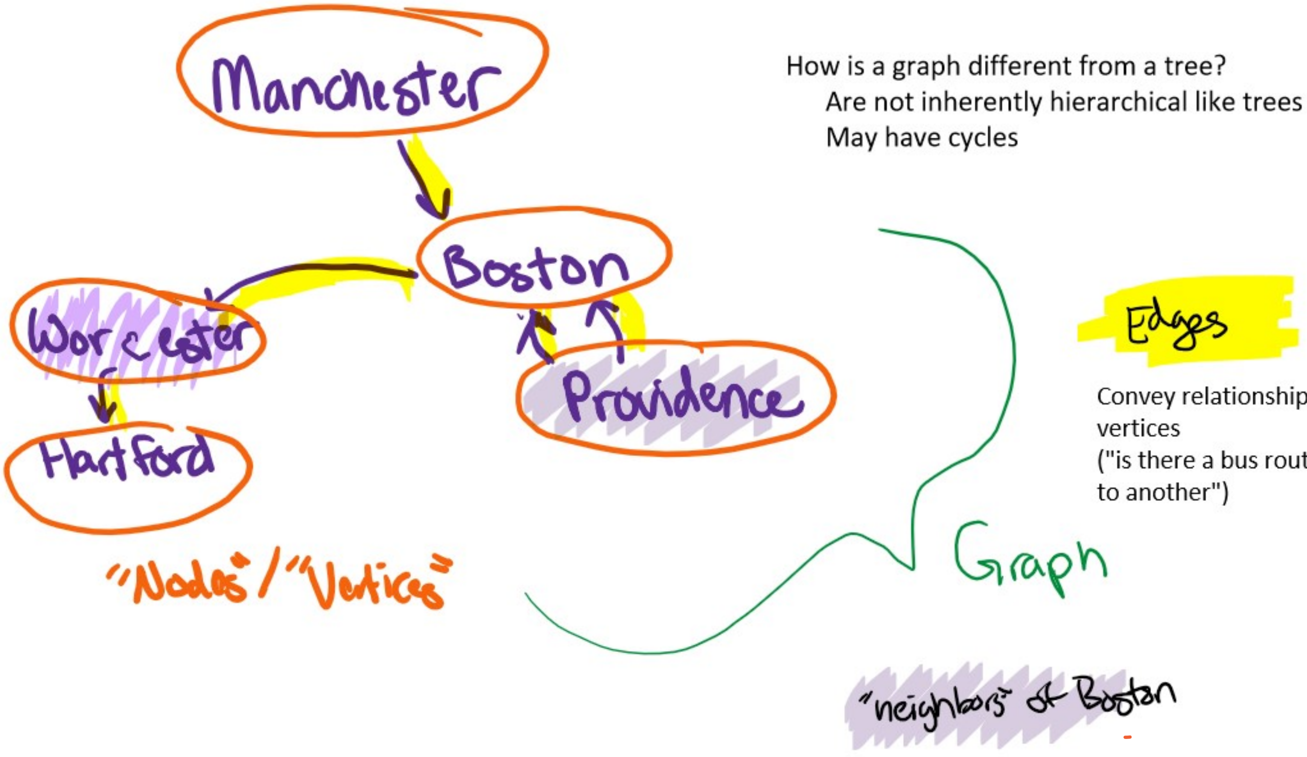
For an example with get(), see the full typed notes from lecture 16

Graphs

Monday, October 17, 2022 1:36 PM

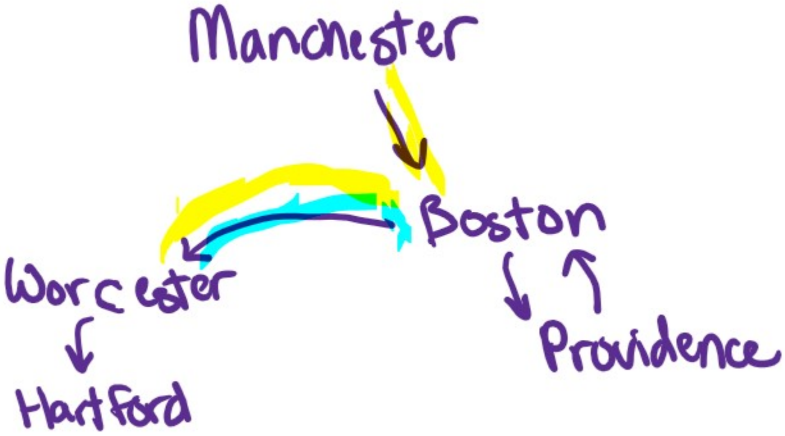
How is a graph different from a tree?

- Are not inherently hierarchical like trees are
- May have cycles



Convey relationships between vertices
("is there a bus route from one city to another")

One question we can ask about this graph is if we can get from Manchester to Worcester by following the bus routes (edges)



One way to conceptualize the answer is recursively -- we ask our neighbors if they can get to the destination, and if so, we know we can get to the destination (that is, answering if we can get from Manchester to Worcester boils down to asking if we can get from Boston to Worcester). More on this in the typed up notes.

HERE'S ONE WAY WE CAN REPRESENT VERTICES AND EDGES IN A GRAPH — WE'LL BUILD MORE OF THIS IN THE NEXT LECTURES.

```
public class CityVertex {
   LinkedList<CityVertex> toCities;
   String name;

   public CityVertex(String nm) {
       this.name = nm;
       this.toCities = new LinkedList<CityVertex>();
   }

   public void addEdge(CityVertex toVertex) {
       this.toCities.add(toVertex);
   }

   public String toString() {
       String retstring = "City " + this.name + " goes to { ";
       for (CityVertex toCity : this.toCities) {
           retstring += toCity.name + " ";
       }
       retstring += "}";
       return retstring;
   }


}
```

```java
public class TestCityVertex {
   public static void main(String [] args) {
       CityVertex man = new CityVertex("Manchester");
       CityVertex bos = new CityVertex("Boston");
       CityVertex pvd = new CityVertex("Providence");
       CityVertex wos = new CityVertex("Worcester");
       CityVertex har = new CityVertex("Hartford");



   }
}
```