CS310 – Advanced Data Structures and Algorithms

Class 13: Intro to Project 2
Inheritance Example from JDK

- HashMap extends AbstractMap, and so do six other JDK classes, so AbstractMap is the base class, the superclass.
- AbstractMap has lots of code that all these classes use.
- It offloads code from seven JDK classes, making them smaller and easier to support, and reduces code duplication.
- Note it’s not efficient for get and put: to do get(x), it looks through the entrySet for x.
- So many (all?) of these seven classes override get and put.
Intro to pa2

• We’ll work with the actual Java 8 HashMap code, with the TreeNode support removed to simplify it.

• TreeNode support: To cover the case of using a bad hash function, a bucket that has too many entries has its collision list converted to a tree structure. This code has been deleted.

• Also, many methods that AbstractMap can handle have been removed to further simplify the code, to HashMap1, provided in the setup project.

• This greatly simplified HashMap1 should be readable.
Intro to pa2

• First you’ll “seal up” HashMap1. Lots of its data structures are not private. Of course you can’t make everything private: you need to be able to call its public API, covered in the JDK Javadoc. You’ll end up with HashMap2.

• Then we would like to know if we can refactor HashMap to use a sealed-up container for the collision list instead of the open-coded linked list structure it has now, a list of Nodes.

• You will replace the list of Nodes with S&W’s SequentialSearchST, one for each bucket. Then the list will be hidden inside this “little map” for the bucket.
Intro to pa2

• Look at SeparateChainingHashST on pg. 465, a class like HashMap. It has very neat code because it is using a SequentialSearchST for each bucket’s collision list. Here is `get` from there:

```java
get(Key key) {
    return (Value) st[hash(key)].get(key);
}
```

• This says: hash the key, find the bucket, and it will have a collision list `st`, which is a little symbol table (i.e. Map) for the bucket. Just use the key to get the value from this `st`, and return it from the outer `get`.

• Wow, is that neat! Can we tame the mess in HashMap?

3/18/2020
But there’s a worry when you make code beautiful, esp. if it was written by real experts.

Is the code ugly because it will run faster that way?

We can draw the collision list structures in both cases and see there’s an additional reference at the start of list involved in using SequentialSearchSTs instead of the open-code list of Nodes.

Let’s find out if and how much the performance changes. Be suspicious if your code runs faster than the original HashMap!