CS314H CSB Midterm 1 Review

Topics

1. OOP
    - motivation for OOP (code reuse, facilitate change)
    - Encapsulation, inheritance, subtype polymorphism
    - Static/dynamic type checking
    - Static/Dynamic binding
    - Overriding and overloading

2. Java
    - Object class
    - equals() and hashCode()
    - interface and abstract class
    - HashSet and HashMap
    - Generics (Just know how to use it. Nothing crazy)

3. Hashing
    - Separate chaining
    - Open addressing
    - Pros and cons
    - Good/bad hash functions
    - Variations like hashing with 2 tables

4. Algorithm analysis
    - Time complexity, space complexity
    - Worst case, average case

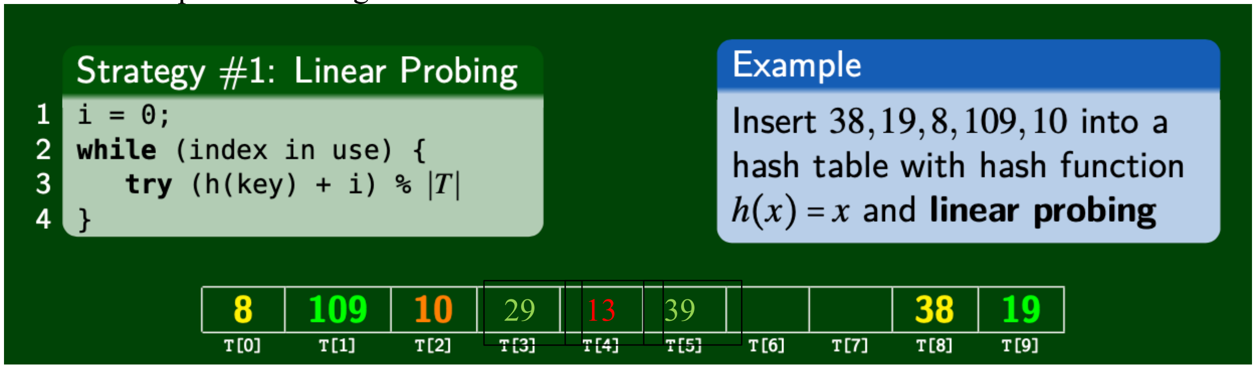
5. Testing
    - Whitebox testing and blackbox testing

1. (True/False). Why or Why not? Justify your answer. Your design choice does not have to be the same as how Java works as long as you provide good reason.

   - An interface can extend an abstract class
   - An interface can extend an interface
   - An abstract class can extend a concrete class

2. Review Kitchen sink note's subclass method overriding question. (Q1 & Q2)

3. Deletion in Open addressing



Strategy #1: Linear Probing

```
i = 0;
while (index in use) {
    try (h(key) + i) % |T|
}
```

Example

Insert 38, 19, 8, 109, 10 into a hash table with hash function $h(x) = x$ and **linear probing**

T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]	T[8]	T[9]
8	109	10	29	13	39			38	19

38, 19, 8, 109, 10 and then 29, 13, and 39 are inserted.

How to delete 19?
Compare lazy delete with extra boolean value (tombstone) vs Hantao's idea vs Jason's idea.

Hantao's idea: Delete 19 and probe everybody till the next empty slot (end of the primary cluster) and shift keys mapped to 9 (So 109 goes to T[9], 29 goes to T[1], 39 goes to T[3]).

Jason's idea: Delete 19 and find the last inserted item that has the same hash value (in this case it is 39) and send 39 to T[9] in place of 19.

Compare time complexity of insert and delete for each case.
Compare space complexity for each case.
Any other pros and cons…?
Which one is better cache performance? (Preserving the locality of data is often preferred than jumping back and forth in memory because modern CPU works more efficiently with contiguous blocks of memory. When data is accessed sequentially, the CPU cache can load nearby elements in a single cache line. This makes subsequent accesses faster because the required data is already present in the cache.)

4. Testing Square

I've been thinking a lot about Squares, because that's just who I am. I recently found the `Shape` interface, and since all the other shapes in my codebase implement this interface, I figured my Squares should implement it too. The `Shape` interface is defined as follows:

```
public interface Shape {
    // gets the shape's width
    public double getWidth();

    // sets the shape's width
    public void setWidth(double width);

    // gets the shape's height
    public double getHeight();

    // sets the shape's heigh
    public void setHeight(double height);

    // gets an array of the shape's dimensions (for most implementors this will be [width, height])
    public double[] getDimensions();

    // sets the dimensions of the shape
    public void setDimensions(double[] dimensions)

    // calculate's the shape's area
    public double getArea();

    // sets the shape's area keeping its dimension ratios constant
    public void setArea(double area);

    // calculates the shape's perimeter
    public double getPerimeter();
    // sets the shape's perimeter keeping its dimension ratios constant

    public void setPerimeter(double);

}
```

You haven't seen any of the code for my Square, you only know it implements the Shape interface. How would you test my Square implementation? Be as thorough as you can.

Make sure to fully describe your tests. For each test, indicate its purpose, as well as what bugs you would be hoping to uncover with it.

5. Is the following implementation of hashCode() legal?
   public int hashCode() { return 0; } If so, describe the effect of using it. If not, explain why.

6. Which of the following would be O(n) for a random search miss, using linear probing? Is it any different for search hit?

Case 1. All keys hash to the same index.
Case 2. All keys hash to different indices.
Case 3. All keys hash to an odd-numbered index.
Case 4. All keys hash to different odd-numbered indices.