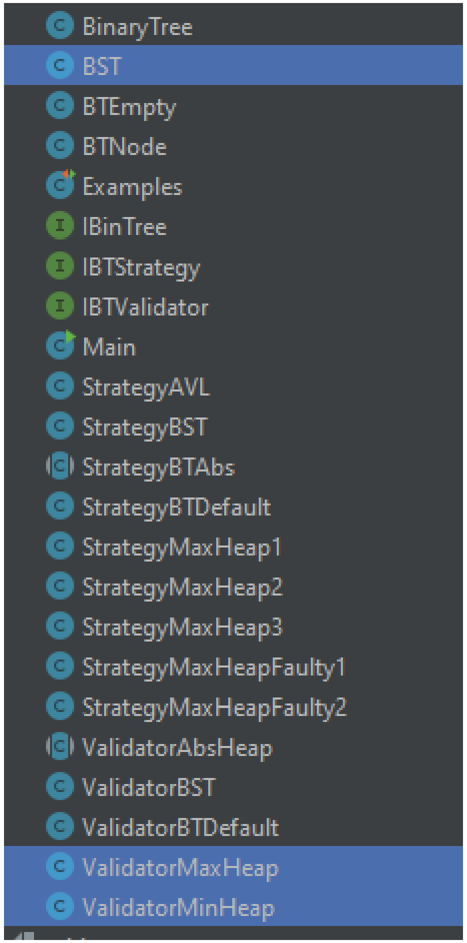
Lab 4 – Testing the Heap Validators & BST

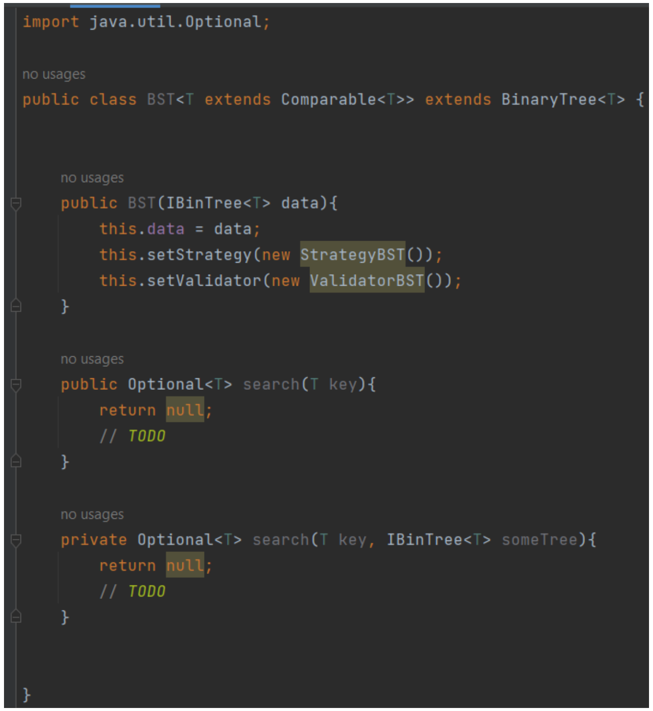
Profs. Ahrens, Sun – B24 - CS2102

Build off of the HW4 starter code

- Same code from lecture
  - Including StrategyBST and ValidatorBST
- Make 3 new classes
- ValidatorMaxHeap
  - Implements IBTValidator
- ValidatorMinHeap
  - Implements IBTValidator
- BST
  - Extends BinaryTree<T>
  - T extends Comparable<T>
- Copy the stubs from the lab instructions page

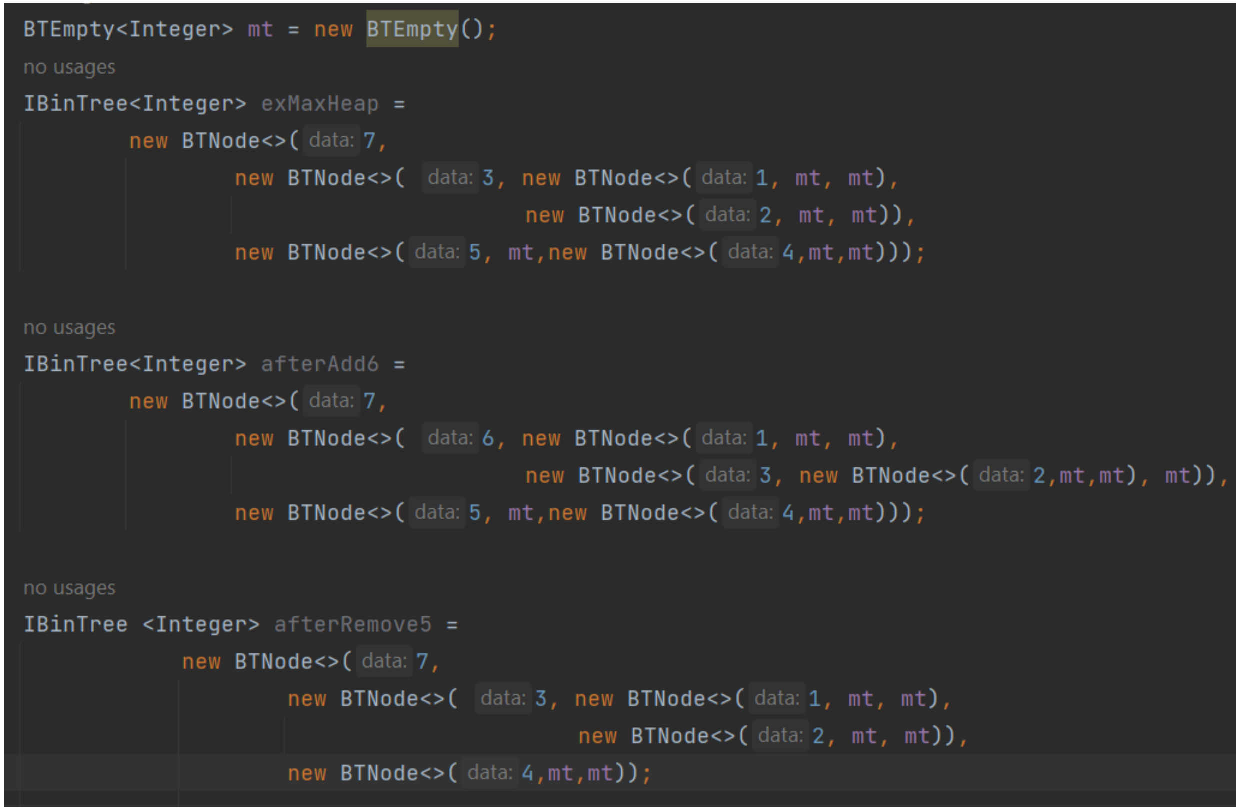


BinaryTree
BST
BTEmpty
BTNode
Examples
IBinTree
IBTStrategy
IBTValidator
Main
StrategyAVL
StrategyBST
StrategyBTAbs
StrategyBTDefault
StrategyMaxHeap1
StrategyMaxHeap2
StrategyMaxHeap3
StrategyMaxHeapFaulty1
StrategyMaxHeapFaulty2
ValidatorAbsHeap
ValidatorBST
ValidatorBTDefault
ValidatorMaxHeap
ValidatorMinHeap



```
import java.util.Optional;

no usages
public class BST<T extends Comparable<T>> extends BinaryTree<T> {


    no usages
    public BST(IBinTree<T> data){
        this.data = data;
        this.setStrategy(new StrategyBST());
        this.setValidator(new ValidatorBST());
    }

    no usages
    public Optional<T> search(T key){
        return null;
        // TODO
    }

    no usages
    private Optional<T> search(T key, IBinTree<T> someTree){
        return null;
        // TODO
    }


}
```

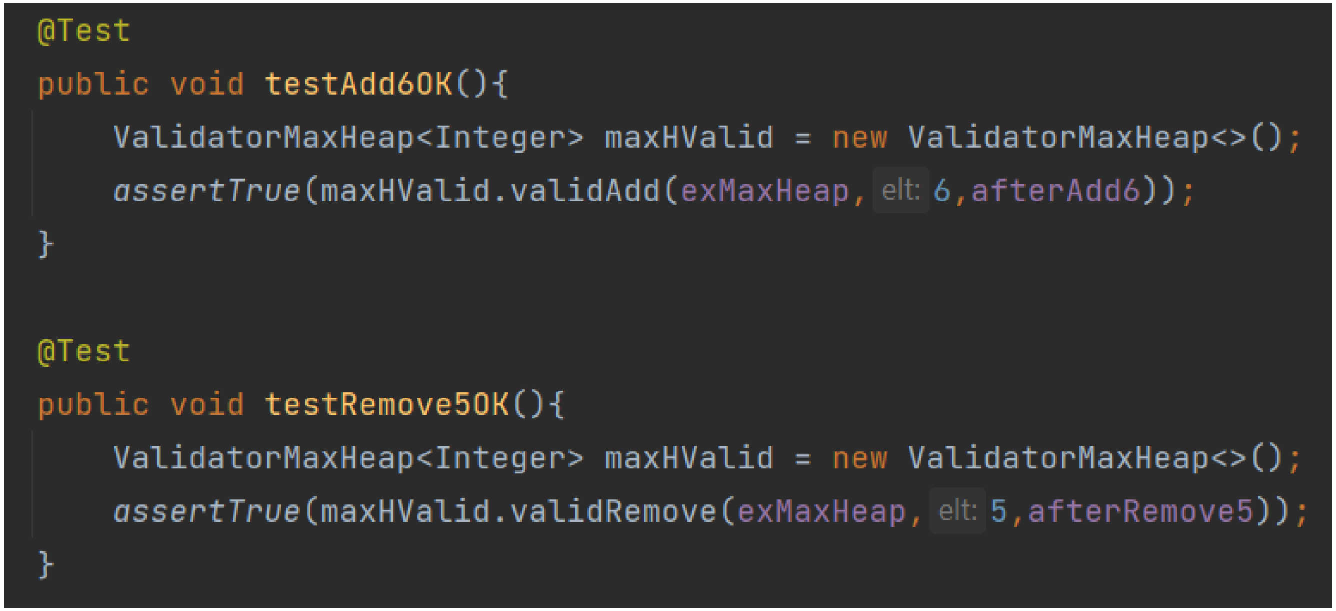
Make Example Data



```
BTEmpty<Integer> mt = new BTEmpty();
no usages
IBinTree<Integer> exMaxHeap =
        new BTNode<>( data: 7,
                new BTNode<>(  data: 3, new BTNode<>( data: 1, mt, mt),
                                      new BTNode<>( data: 2, mt, mt)),
                new BTNode<>( data: 5, mt,new BTNode<>( data: 4,mt,mt)));

no usages
IBinTree<Integer> afterAdd6 =
        new BTNode<>( data: 7,
                new BTNode<>(  data: 6, new BTNode<>( data: 1, mt, mt),
                                      new BTNode<>( data: 3, new BTNode<>( data: 2,mt,mt), mt)),
                new BTNode<>( data: 5, mt,new BTNode<>( data: 4,mt,mt)));

no usages
IBinTree <Integer> afterRemove5 =
            new BTNode<>( data: 7,
                    new BTNode<>(  data: 3, new BTNode<>( data: 1, mt, mt),
                                          new BTNode<>( data: 2, mt, mt)),
                    new BTNode<>( data: 4,mt,mt));
```

- Make a field to hold an empty heap with a short name (like mt)
- Make a valid MaxHeap IBinTree<Integer> to test with.
- Make a valid resulting MaxHeap after adding a specific element
- Make a valid resulting MaxHeap after removing a specific element

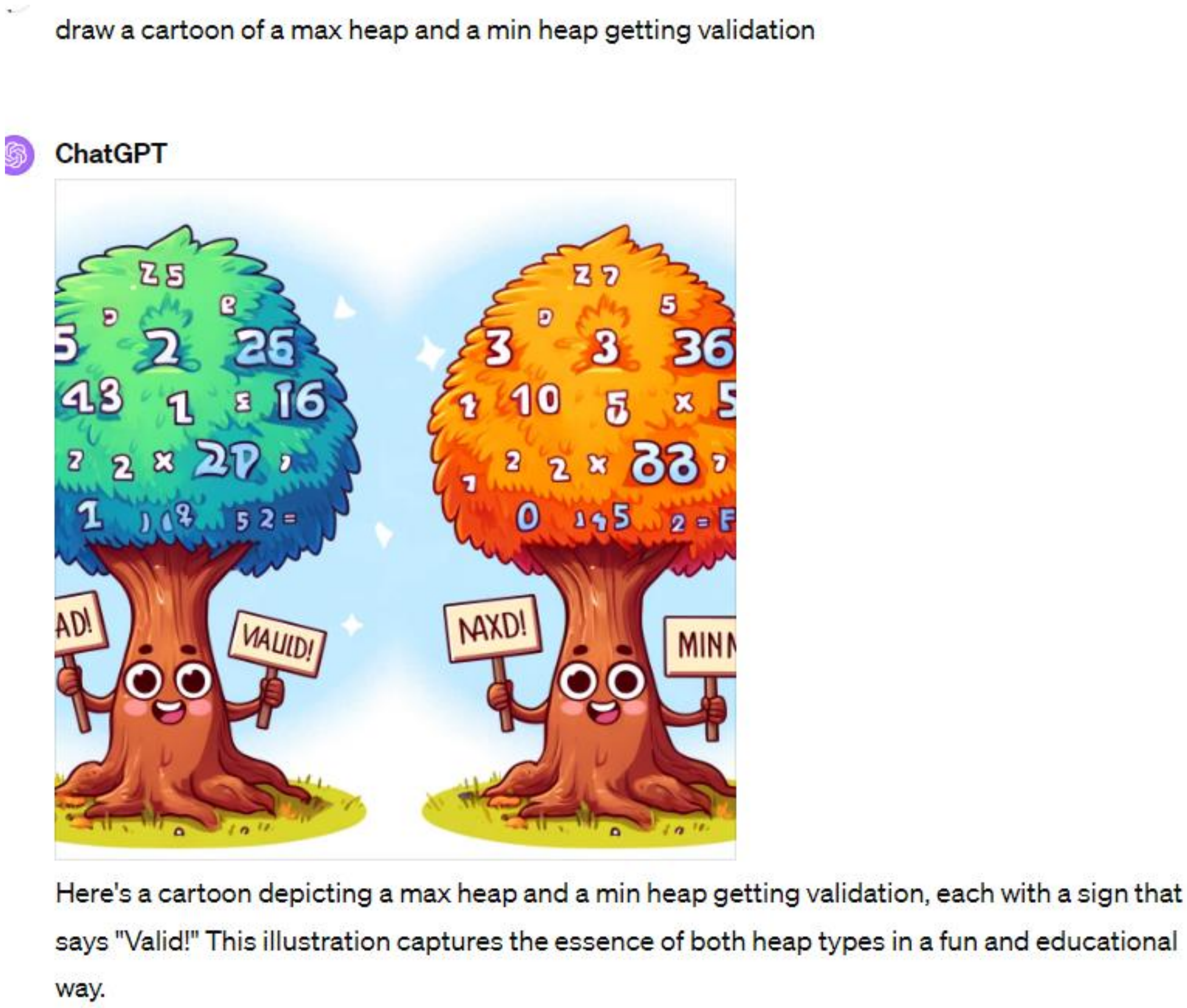
Make an assertTrue() test for each



```java
@Test
public void testAdd6OK(){
    ValidatorMaxHeap<Integer> maxHValid = new ValidatorMaxHeap<>();
    assertTrue(maxHValid.validAdd(exMaxHeap, elt: 6,afterAdd6));
}

@Test
public void testRemove5OK(){
    ValidatorMaxHeap<Integer> maxHValid = new ValidatorMaxHeap<>();
    assertTrue(maxHValid.validRemove(exMaxHeap, elt: 5,afterRemove5));
}
```

- Validate that the resulting heap, after adding 6, is ok
- Validate that the resulting heap, after removing 5, is ok
- Submit to autograder between each test

Check in with your lab leader if you need help!



draw a cartoon of a max heap and a min heap getting validation

ChatGPT

Here's a cartoon depicting a max heap and a min heap getting validation, each with a sign that says "Valid!" This illustration captures the essence of both heap types in a fun and educational way.

Make assertFalse() tests

- Think of a resulting tree that would fail one of the criteria
    - Loses an element
    - Adds extra elements
    - Not a maxheap
- The validator.validAdd()/validRemove() should produce false on such resulting trees
(its goal is to prevent bad adds/removes after all)

ValidateMinHeap & BST

- Test ValidateMinHeap on similar data
  - Not needed to get 50% coverage in lab
  - But part of getting 100% coverage for extra credit.
- Test BST's search()
  - Returns Optional.of(element) when it find it
  - Returns Optional.empty() when it does not find it
- If you don't finish these tests for lab, you should do so to test your solution for the HW itself