PA 6 due Today
PA 7 tomorrow → due in 2 weeks

Binary Search Tree (BST)

```
class BST<K, V> {          → implements Map<K, V>
  Node<K, V> root;
  BST() (this.root = null);
  BST(Node<K, V> root) { this.root = root; }

private   V get(Node<K, V> node, K key) {
base case   if (node == null) { //throw error }   ] not found
            if (node.key.equals(key)) {            ] found
              return node.value;
            }
recursive   if (node.key > key) {
case          return get(node.left, key);          ] greater
            }
            else {
              return get(node.right, key);         ] lesser
            }
          }

public    V get(Key key) {
            return this.get(root, key);
          }
}
```

```
class Node<K,V> {
  K key;
  V value;
  Node<K,V> left;
  Node<K,V> right;
  public Node(K key, V value,
              Node<K,V> left,
              Node<K,V> right) {
    this.key = key;
    this.value = value;
    this.left = left;
    this.right = right;
  }
}
```

Where is the get() method broken?

> doesn't work

How can we fix the get() method to work with Objects?

Interface → compare()
↳ Comparator/also on Object
↳ Comparable
  ↳ compareTo( )
    ↳ <0 less than
      0 equal
      >0 greater than

public String implements Comparable

What error should we throw in get() if the key isn't found?

→ NoSuchElementException / ElementNotFoundException

What would the code that uses get() look like to prevent the program crashing if the key is missing?

```
try {
   tree.get(2);
}
catch (NoSuchElementException e) {
   // do we care? → error message
                    correction
}
```

```
boolean find(E toFind) {
  Comparable comp = (Comparable) toFind;
  while loop
    if (comp.compareTo( — ) == 0) {
       return true;
  }
  }
  return false;
}
```

runtime error?

```
<E> extends Comparable>
class Test <E extends Comparable> {
   toFind
   if( toFind.compareTo(  ) == 0) {
   }
}
```

```
main( )
  Test <String> t1 = new ....
  Test <MyStudent> t2 = new ...
```

↳ compiler error if MyStudent does Not implement Comparable

Assume the key and value are identical for this example:

Trace the path for get(4)
How many nodes does it touch?

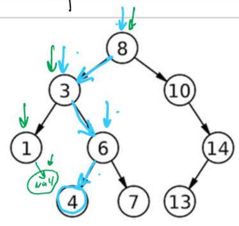
4 nodes

Trace the path for get(2)
How many nodes does it touch?

3 nodes

What happens when the node isn't found?

throws exception



left < smaller
right > larger

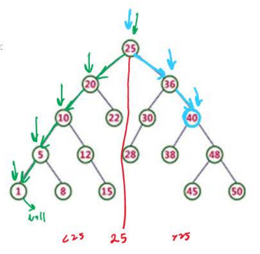
Assume the key and value are identical for this example:

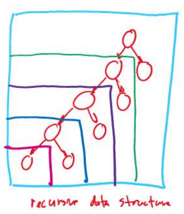
Trace the path for get(40)
How many nodes does it touch?

3 nodes

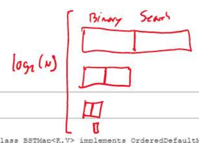
Trace the path for get(4)
How many nodes does it touch?

5 Nodes
throws exception





recursive data structure

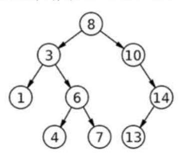


What order does printAllElement() traverse the tree?

```
void printAllElements(Node<K, N> n) {
  if (n == null ) return;
① System.out.println(n.key);
③→ printAllElements(n.left);
  printAllElements(n.right);   ②
}

void printAllElement() {
  printAllElements(this.root);
}
```

What's the ② post, ① pre, ③ in-order traversal of this tree?



Traversal
① pre-order 8 3 1 6 4 7 10 14 13
② post-order 1 4 7 6 3 13 14 10 8
③ in-order 1 3 4 6 7 8 10 13 14
↳ sorted values

```
class BSTMap<K,V> implements OrderedDefaultMap<K,V>{
  Node<K, V> root;
  int size;
  Comparator<K> comparator;
  ...
  Node<K, V> set(Node<K, V> node, K key, V value) {
    if (node == null) {
      this.size += 1;
      return new Node<K, V>(key, value, null, null);
    }
    int comp = this.comparator.compare(node.key, key);
    if (comp < 0) {
      node.right = this.set(node.right, key, value);
      return node;
    } else if (comp > 0) {
      node.left = this.set(node.left, key, value);
      return node;
    } else {
      node.value = value;
      return node;
    }
  }

  @Override
  public void set(K key, V value) {
    if (key == null) {
      throw new IllegalArgumentException();
    }
    this.root = this.set(this.root, key, value);
  }
}
```

Use the picture on the left and assume the key and value are identical:

set("5", 5);
set("11", 11);
set("15", 15);
set("12", 12);

What is the picture after calling the above set() methods?

What is the run-time for a Binary Tree

set()
Worst Case

What conditions make up the worst case for set()?

Best Case:

What conditions make up the best case for set()?

get()
Worst Case

What conditions make up the worst case for get()?

Best Case:

What conditions make up the best case for get()?

printAllElements()
Worst Case

Best Case:

What conditions make up the best case for set()?