cphbusiness

COPENHAGEN BUSINESS ACADEMY

Algorithms and Data Structure-Day2

Today

- Efficiency of algorithms
  - Big O
- Classic Algorithms
  - Recursion
- Data Structures
  - Java collection framework
  - LinkedList
  - Binary Tree
  - Binary Search Tree
  - Hash Table/Hash map
  - Binary Search Tree
  - Tree map

Russian doll??

Recursion

- The *base case* returns a value without making any subsequent recursive calls. It does this for one or more special input values for which the function can be evaluated without recursion. For factorial(), the base case is $n = 1$.

- The *reduction step* is the central part of a recursive function. It relates the value of the function at one (or more) input values to the value of the function at one (or more) other input values. Furthermore, the sequence of input values values must *converge* to the base case.
  For factorial(), the value of n decreases by 1 for each call, so the sequence of input values converges to the base case.

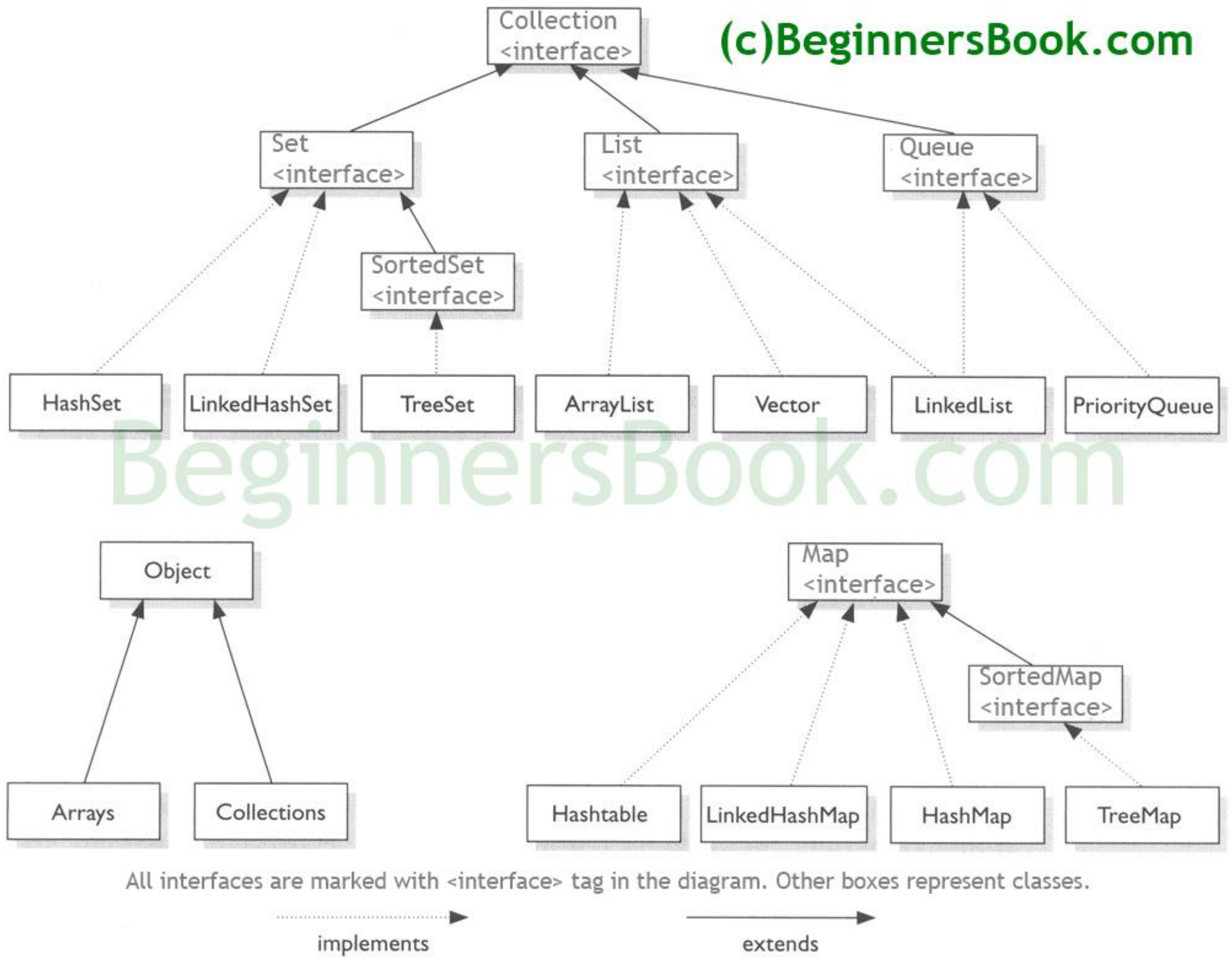
A recursive Java method

1. The method calls itself (”recursive call”)
2. The **recursive call** solves a smaller problem
3. The method determines whether a ”**base-case**” has been reached. If this is the case, the recursive call is not made
4. Sooner or later, the smaller problem will turn into the base case

Factorial n!

```
factorial(5)
    factorial(4)
        factorial(3)
            factorial(2)
                factorial(1)
                    return 1
                return 2*1 = 2
            return 3*2 = 6
        return 4*6 = 24
    return 5*24 = 120
```

Collection Framework



Linked List

- Demonstrations

Tree

- Extenstion of LinkedList
- Collection of nodes
- Fast searchable collection
- Characteristics of a tree
  - Root
  - Parent
  - Child
  - Leaf node
  - Siblings
  - Path
  - Depth
  - Height

Tree Traversal

- Printing all the nodes

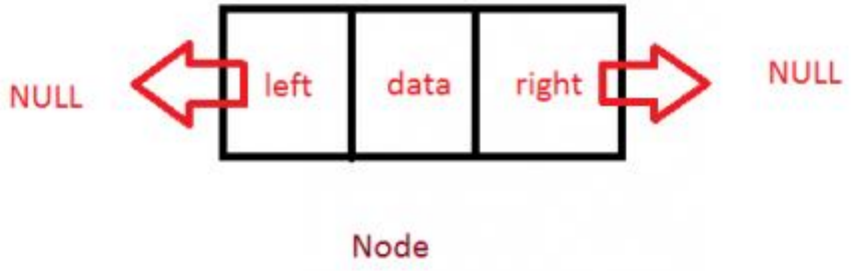
- The order of nodes we must traverse?
  - Preorder – read the parent before its children
  - Postorder – read the parent after its children
  - Inorder – read left, then parent, then right child
  - Inorder(node x)

```
  inorder(x.left)
  print(x)
  inorder(x.right)
```

Binary Tree

- Nodes with data and references to other nodes (two children
  - Left and right child

```
public class BinaryNode<T> {
    T element;
    BinaryNode left;
    BinaryNode right;
}
```



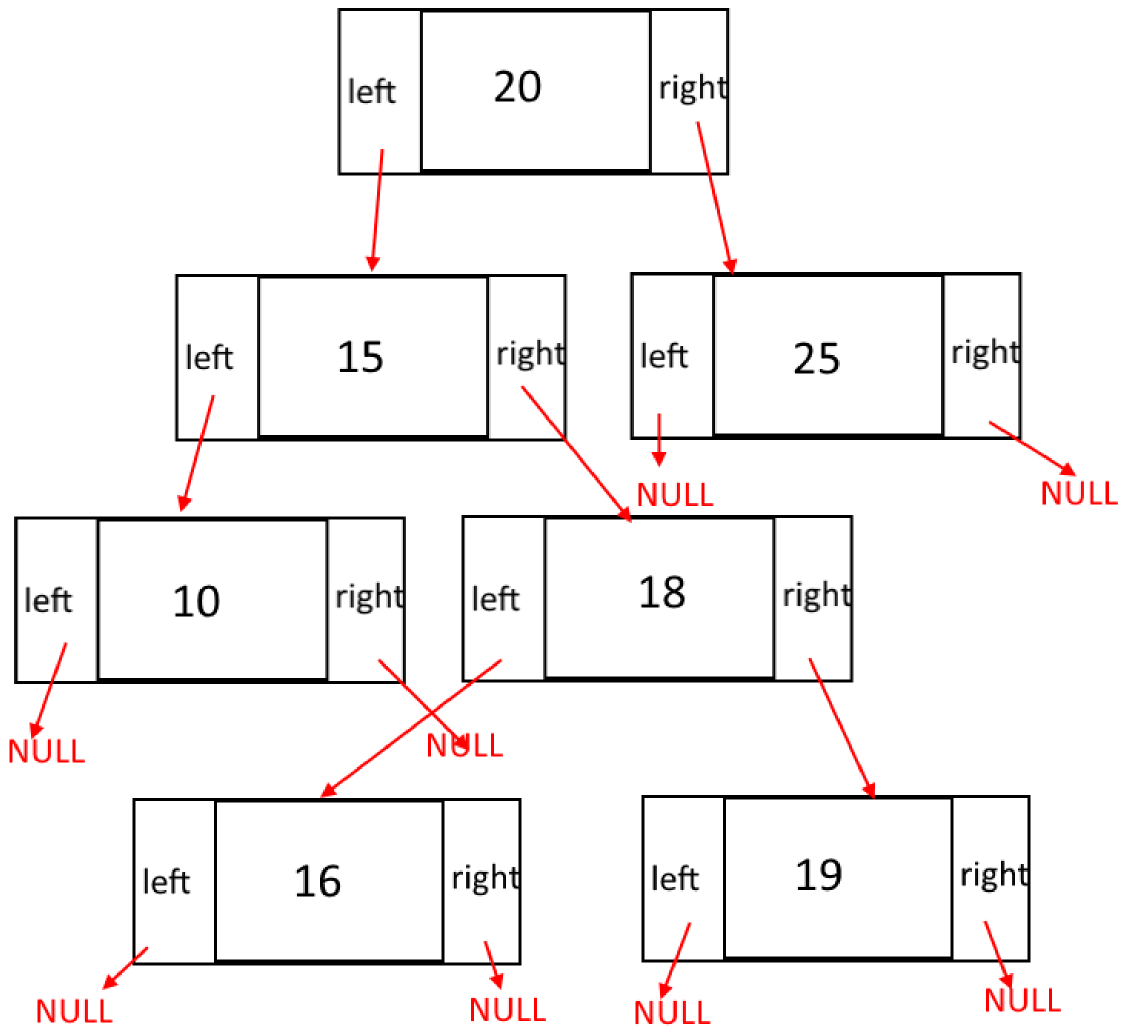
cphbusiness

Recursive Search algorithm

```
If the tree is empty
    element is not in the tree
else
   if root match the key
       element found
   else
       if key < root-value
           **search in the left sub tree      (repeat)**
       else
           **search in the right sub tree   (repeat)**
```

Binary Search Tree (BST)

- All nodes smaller than root is on the left
- All nodes greater than root is on the right
- Olog(n) for a search



Log (n)