Bài 20
Map & Tree

Môn học: PF-JAVA

Mục tiêu

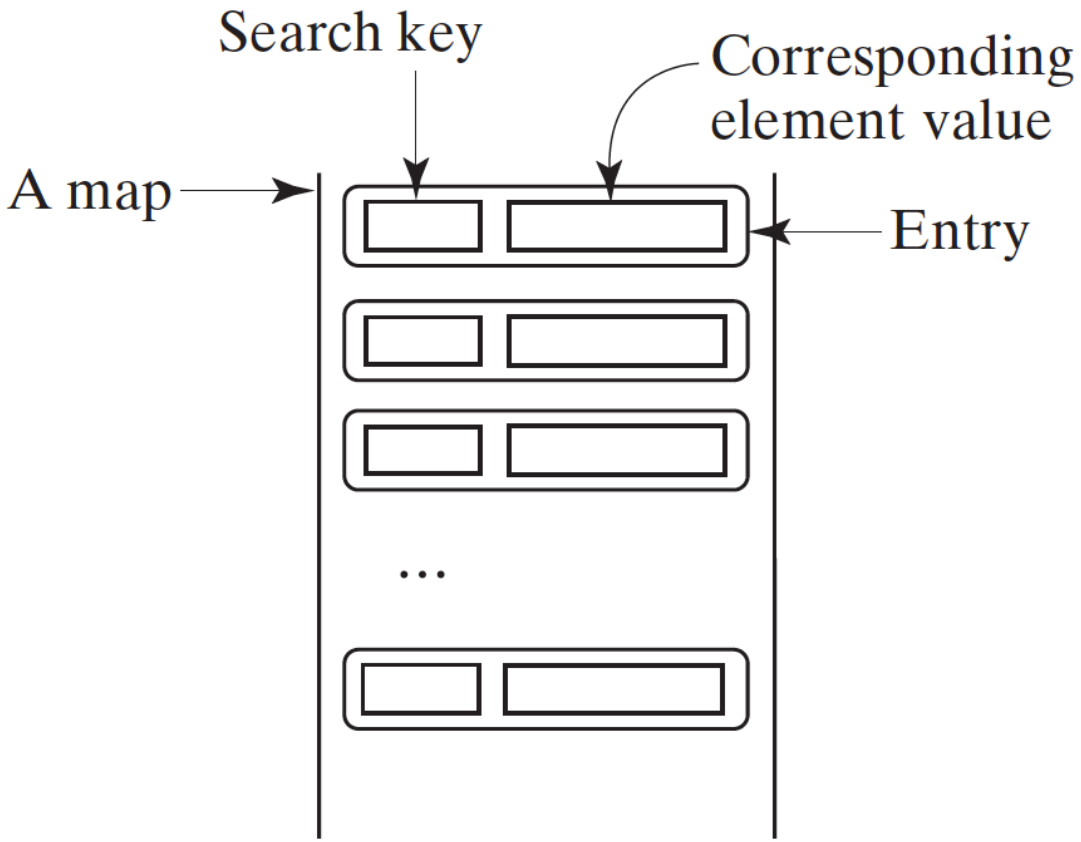
- Trình bày được cấu trúc dữ liệu Map
- Sử dụng được Map trong Java Collection Framework
- Trình bày được cấu trúc dữ liệu Tree
- Triển khai được cấu trúc dữ liệu Tree

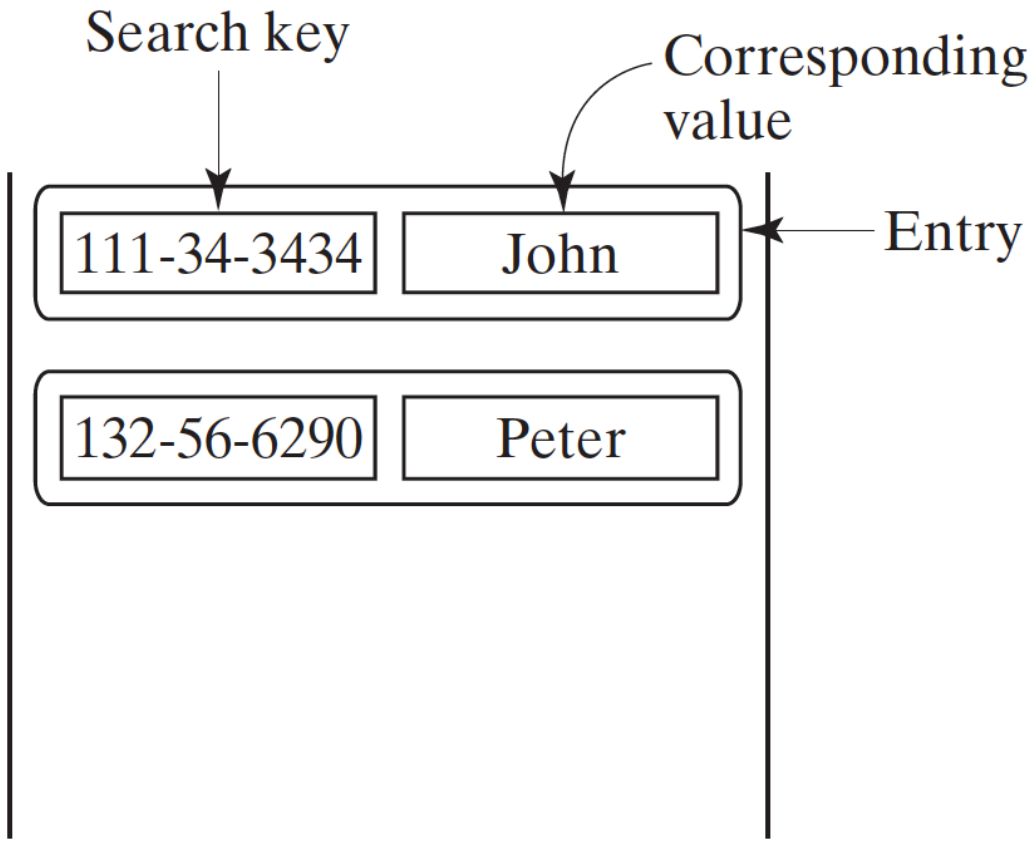
Map

Map

- Map là cấu trúc dữ liệu lưu trữ danh sách các cặp key/value
- Map cho phép thực hiện các hành động truy xuất, xoá và cập nhật các cặp key/value rất hiệu quả thông qua việc sử dụng key
- Map không cho phép 2 key trùng lặp
- Mỗi key tương ứng với một value
- Một cặp key-value được gọi là một Entry

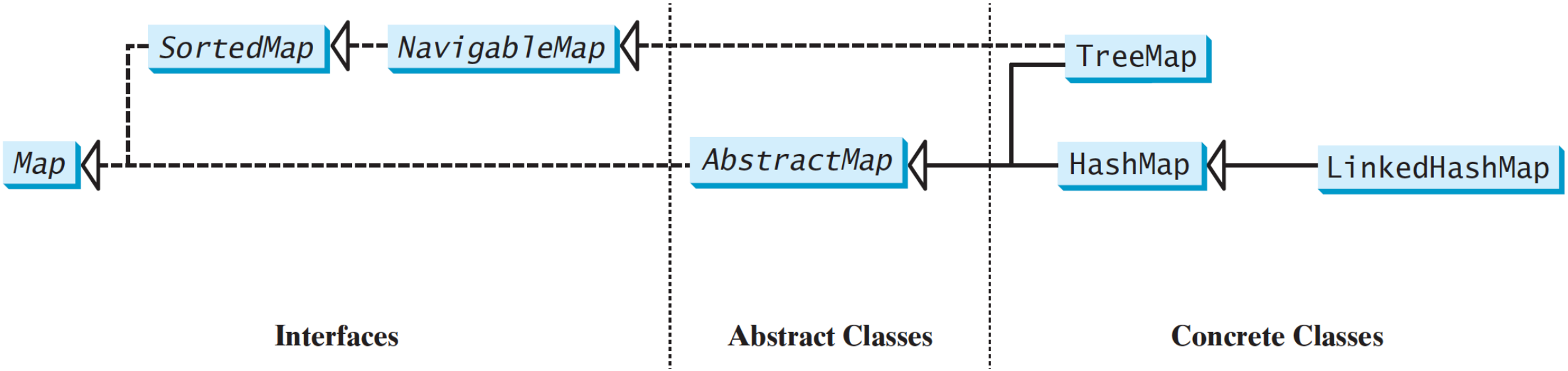
Map: Ví dụ


Search key
Corresponding element value
A map
Entry
...


Search key
Corresponding value
Entry
111-34-3434
John
132-56-6290
Peter

3 loại Map

- HashMap: Tối ưu cho các thao tác tìm kiếm, thêm và xoá
- LinkedHashMap: Kế thừa HashMap, hỗ trợ sắp xếp các entry
- TreeMap: Tối ưu cho thao tác duyệt qua các entry theo trật tự sắp xếp của các key



Interface Map và Entry

«interface»
java.util.Map<K,V>

```
+clear(): void
+containsKey(key: Object): boolean

+containsValue(value: Object): boolean

+entrySet(): Set<Map.Entry<K,V>>
+get(key: Object): V
+isEmpty(): boolean
+keySet(): Set<K>
+put(key: K, value: V): V
+putAll(m: Map<? extends K,? extends
 V>): void
+remove(key: Object): V
+size(): int
+values(): Collection<V>
```

«interface»
java.util.Map.Entry<K,V>

```
+getKey(): K
+getValue(): V
+setValue(value: V): void
```

HashMap: Ví dụ

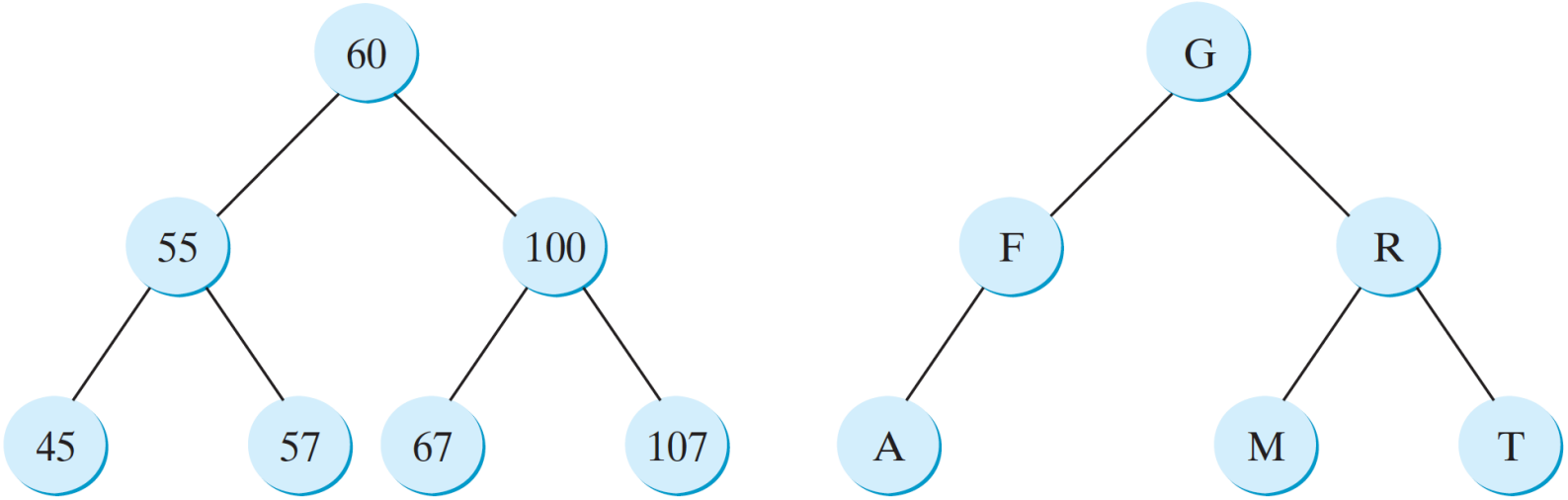
```
HashMap<String, Integer> customers = new HashMap<>();
customers.put("John", 30);
customers.put("Mike", 28);
customers.put("Bill", 32);
customers.put("Maria", 27);

Set<String> keys = customers.keySet();
for (String key: keys){
    System.out.println("Key: " + key + ": " + customers.get(key));
}
```

Tree

Binary Tree

- Tree lưu trữ dữ liệu trên các node
- Các node có mối quan hệ cha-con, node trên cùng được gọi là node gốc (root node)
- Binary Tree (Cây nhị phân) là cây mà mỗi node có 0, 1 hoặc 2 cây con (subtree)
- 2 cây con được gọi lần lượt là left-subtree (cây con trái) và right-subtree (cây con phải)

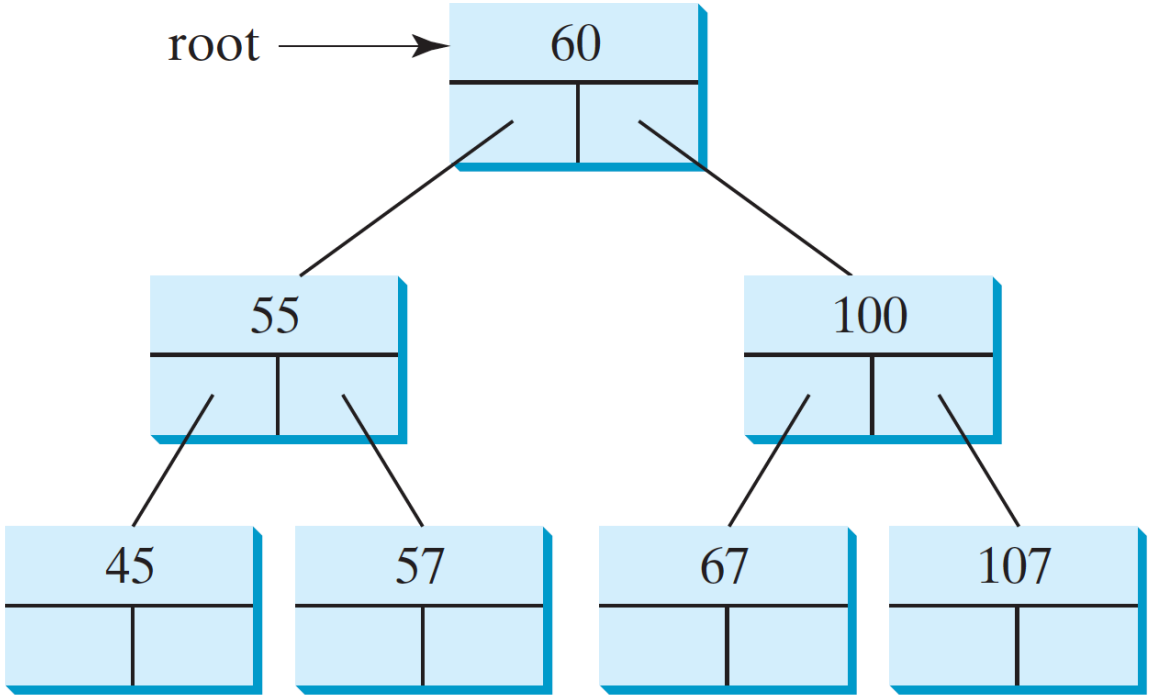


Các khái niệm

- Độ dài của đường đi (length of the path) là số lượng các cạnh
- Chiều sâu của node là độ dài của đường đi tính từ root node đến node đó
- Node anh em (sibling) là các node có cùng node cha
- Node không có node con thì gọi là node lá (leaf node)
- Chiều cao của cây là độ dài của đường đi từ node gốc đến node lá

Binary Search Tree (BST)

- Binary Search Tree (Cây tìm kiếm nhị phân) được biểu diễn bằng một tập các node liên kết với nhau
- Mỗi node chứa dữ liệu và 2 liên kết: 1 liên kết sang node con bên trái và. 1 liên kết sang node con bên phải



Triển khai BST

```
class TreeNode<E> {
    protected E element;
    protected TreeNode<E> left;
    protected TreeNode<E> right;
    public TreeNode(E e) {
        element = e;
    }
}
```

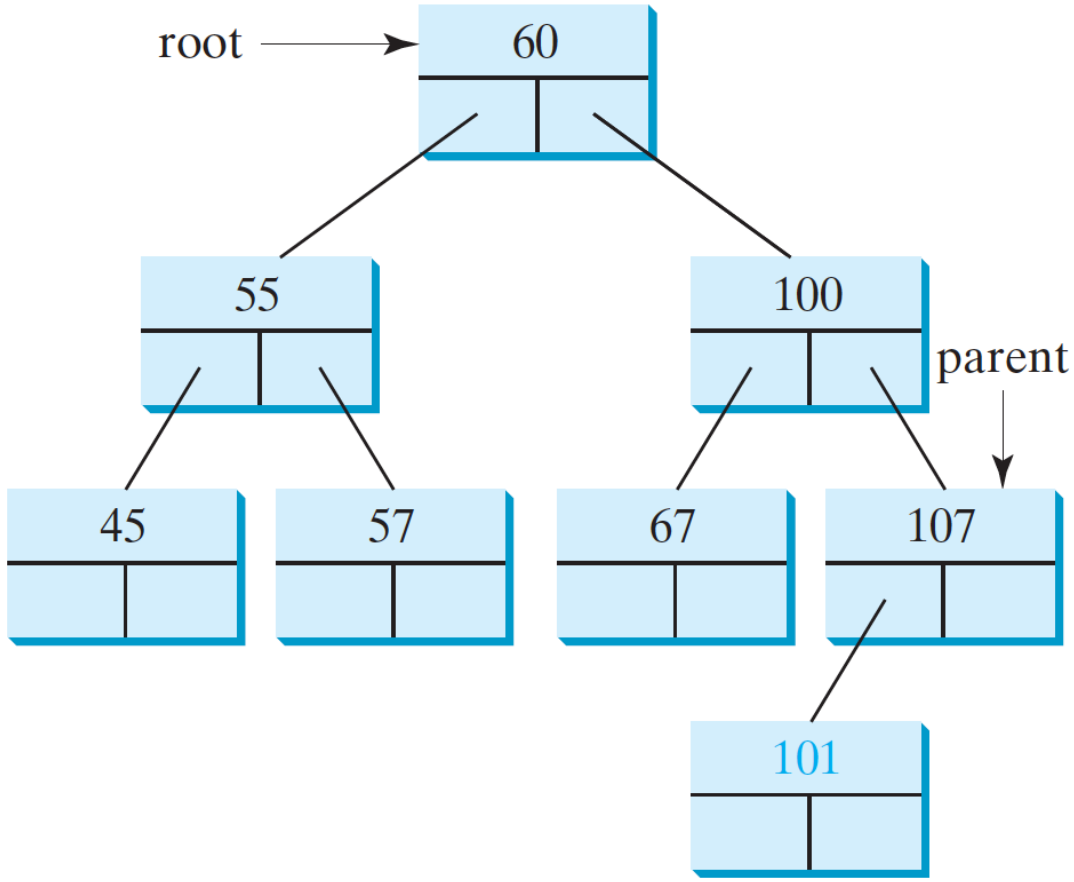
```
TreeNode<Integer> root = new
TreeNode<>(60);
root.left = new TreeNode<>(55);
root.right = new TreeNode<>(100);
```

Tìm kiếm trên BST

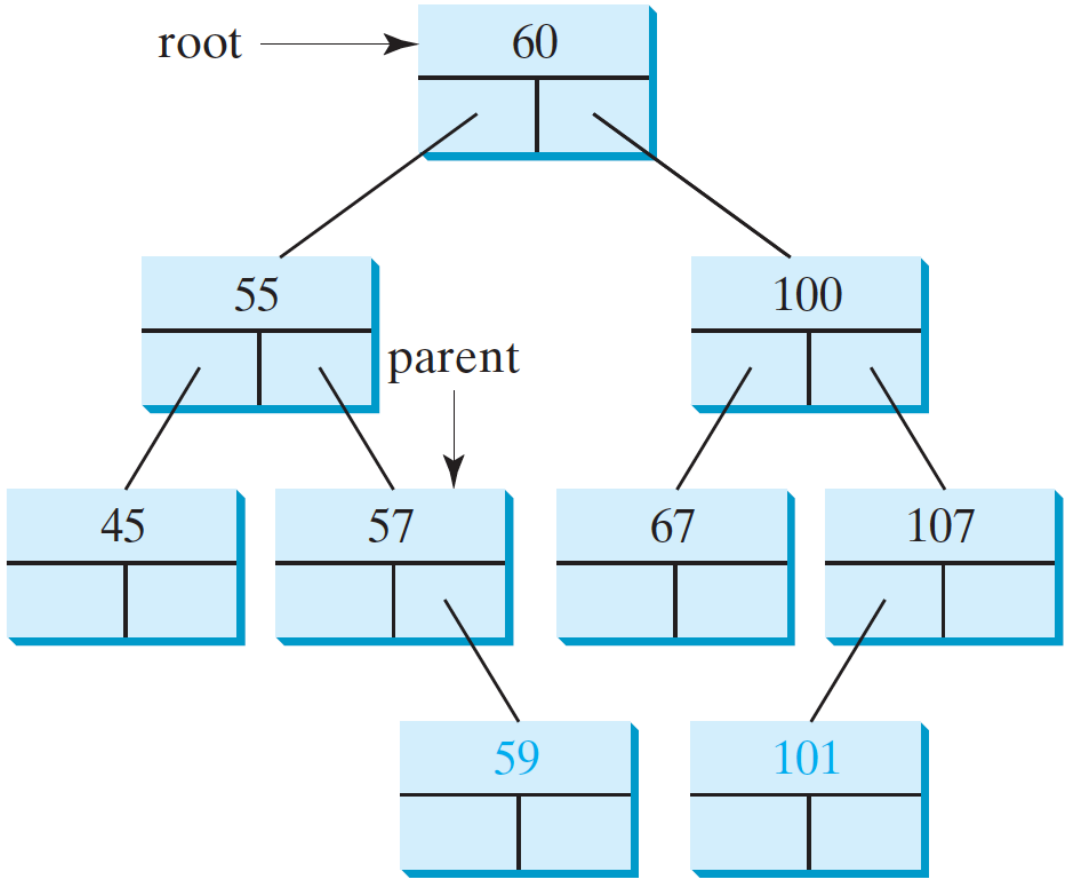
```
public boolean search(E element) {
    TreeNode<E> current = root;

    while (current != null) {
        if (element < current.element) {
            current = current.left;
        } else if (element > current.element) {
            current = current.right;
        } else
            return true;
    }
    return false;
}
```

Thêm phần tử vào BST



Chèn giá trị 101 vào BST

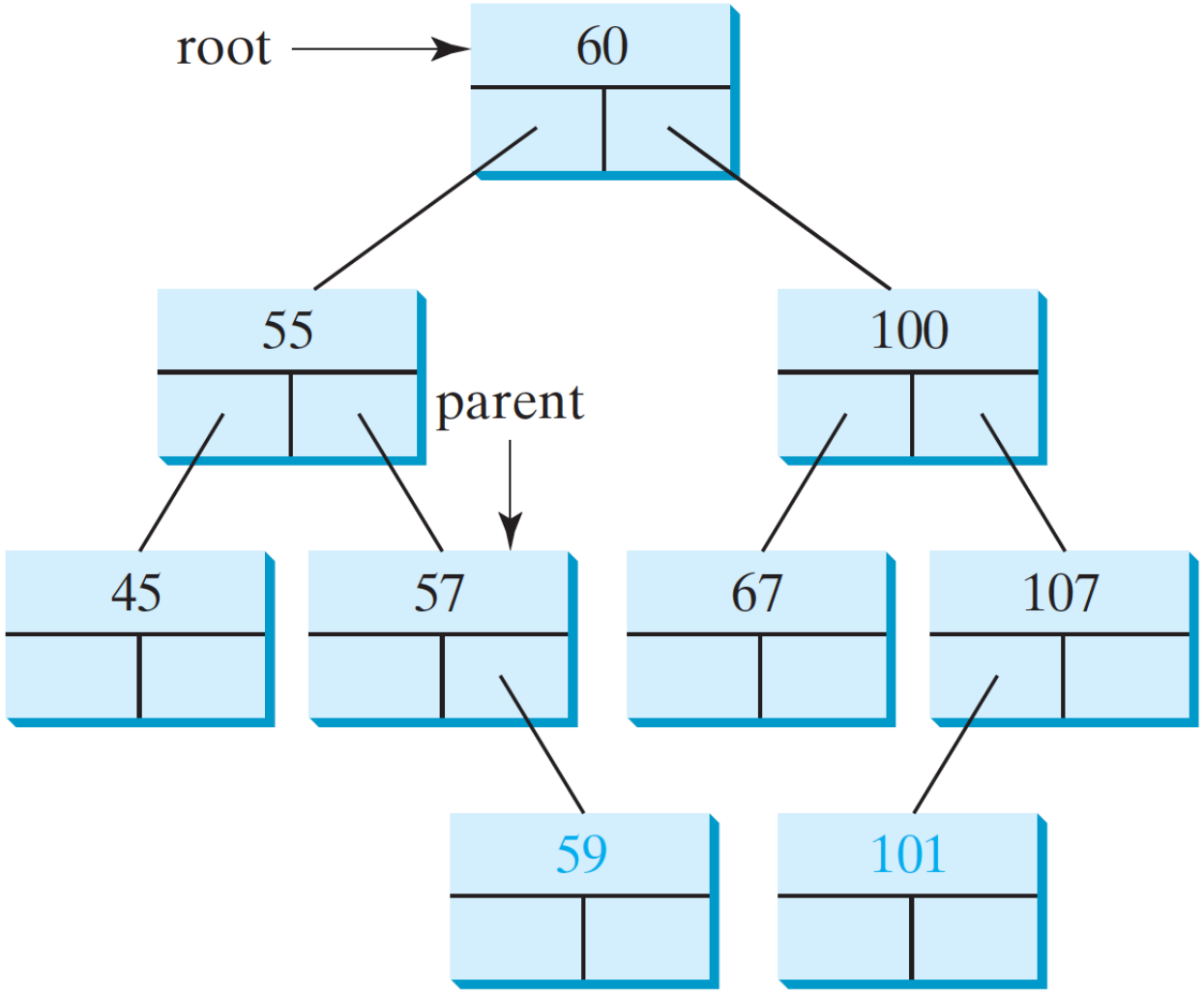


Chèn giá trị 59 vào BST

Duyệt cây

- Duyệt cây là thao tác đi qua từng node của cây, mỗi node được đi qua một lần duy nhất
- Có nhiều cách để duyệt cây, ví dụ:
  - Inorder: Duyệt theo thứ tự cây con trái->node hiện tại->cây con phải
  - Postorder: Duyệt theo thứ tự cây con trái->cây con phải->node hiện tại
  - Preorder: Duyệ theo thứ tự node hiện tại->cây con trái->cây con phải
  - Breath-first: Duyệt lần lượt theo từng level
  - …

Duyệt cây: ví dụ



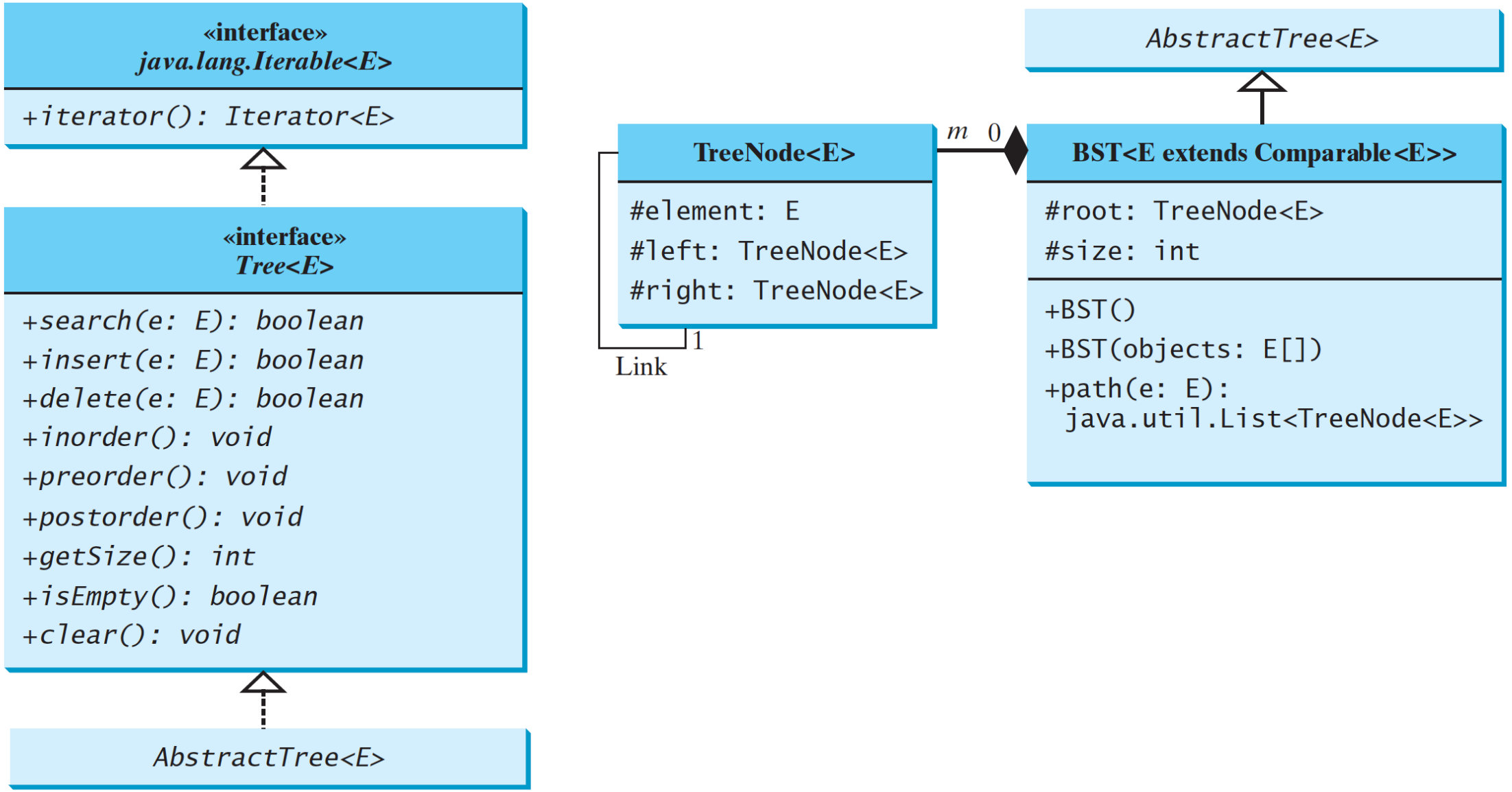
Inorder: 45 55 57 59 60 67 100 101 107

Postorder: 45. 59 57 55 67 101 107 100 60

Preorder: 60 55 45 57 59 100 67 107 101

Breath-first: 60 55 100 45 57 67 107 59 101

Triển khai BST



[Thực hành] Tạo HashMap, LinkedHashMap, TreeMap

[Thực hành] Cài đặt cây tìm kiếm nhị phân

[Bài tập] Tìm kiếm trong BST

[Bài tập] Postorder trong BST

[Bài tập] Preorder trong BST

[Bài tập] Xoá trong BST

[Bài tập] Occurrences of Words using MAP

Tổng kết

- Map lưu trữ dữ liệu theo từng cặp key/value
- Mỗi cặp key/value được gọi là một Entry
- Thao tác truy xuất sử dụng key có hiệu suất cao
- Java Collection Framework cung cấp 3 lớp Map là: HashMap, LinkedHashMap và TreeMap
- Tree lưu trữ dữ liệu theo các node có liên kết cha-con với nhau
- Cây nhị phân là cây mà mỗi node có thể có 0, 1 hoặc 2 cây con