



Object-Oriented Programming with Java



Recitation No. 7

The Routing Table Example

```
public class RoutingTable {  
    /**  
     * Adds a key-value pair to the routing table, where the key is a binary string  
     * specified by the n high-order bits of address  
     **/  
    public void add(int address, int n, String value) {...}  
  
    /**  
     * @return the value associated with the longest prefix in the routing  
     * table that matches the argument.  
     * @throws NoRouteToHostException if no prefix matches address  
     **/  
    public String route(int address) throws NoRouteToHostException {...}  
}
```

Abstraction Function

$$A : RT \rightarrow 2^{\{(b_1 \dots b_n, s) \mid b_i \in \{0,1\}, 1 \leq n \leq 32, s \text{ is a string} \neq \text{null}\}}$$

where:

$$RT = \{rt \mid rt \text{ is a } \textit{RoutingTable} \text{ state}\}$$

No clues about the implementation

Example:

$$A(rt) = \{(000, \textit{wifi1}), (001, \textit{wifi2}), (1, \textit{eth}), (11, \textit{ir})\}$$

The Contract

■ public void add(int address, int n, String value):

- Requires:

$(1 \leq n \leq 32) \ \&\& \ (value \neq null)$

- Ensures:

if $\exists(a_1 \dots a_n, s) \in A(old)$:

$A(new) = A(old) / \{(a_1 \dots a_n, s)\} \cup \{(a_1 \dots a_n, value)\}$

otherwise:

$A(new) = A(old) \cup \{(a_1 \dots a_n, value)\}$

where:

a_1, \dots, a_n are the n high-order bits of address

The Contract (cont.)

■ public String route(int address)

throws NoRouteToHostException:

- Requires: nothing

- Ensures:

1. $A(new) = A(old)$

2. if a *NoRouteToHostException* is thrown:

$$\forall(b_1 \dots b_n, s) \in A(this) : b_1 \dots b_n \neq a_1 \dots a_n$$

otherwise:

$$\exists(a_1 \dots a_n, returned_value) \in A(this) \ \&\&$$
$$\forall(b_1 \dots b_{m>n}, s) \in A(this) : b_1 \dots b_m \neq a_1 \dots a_m$$

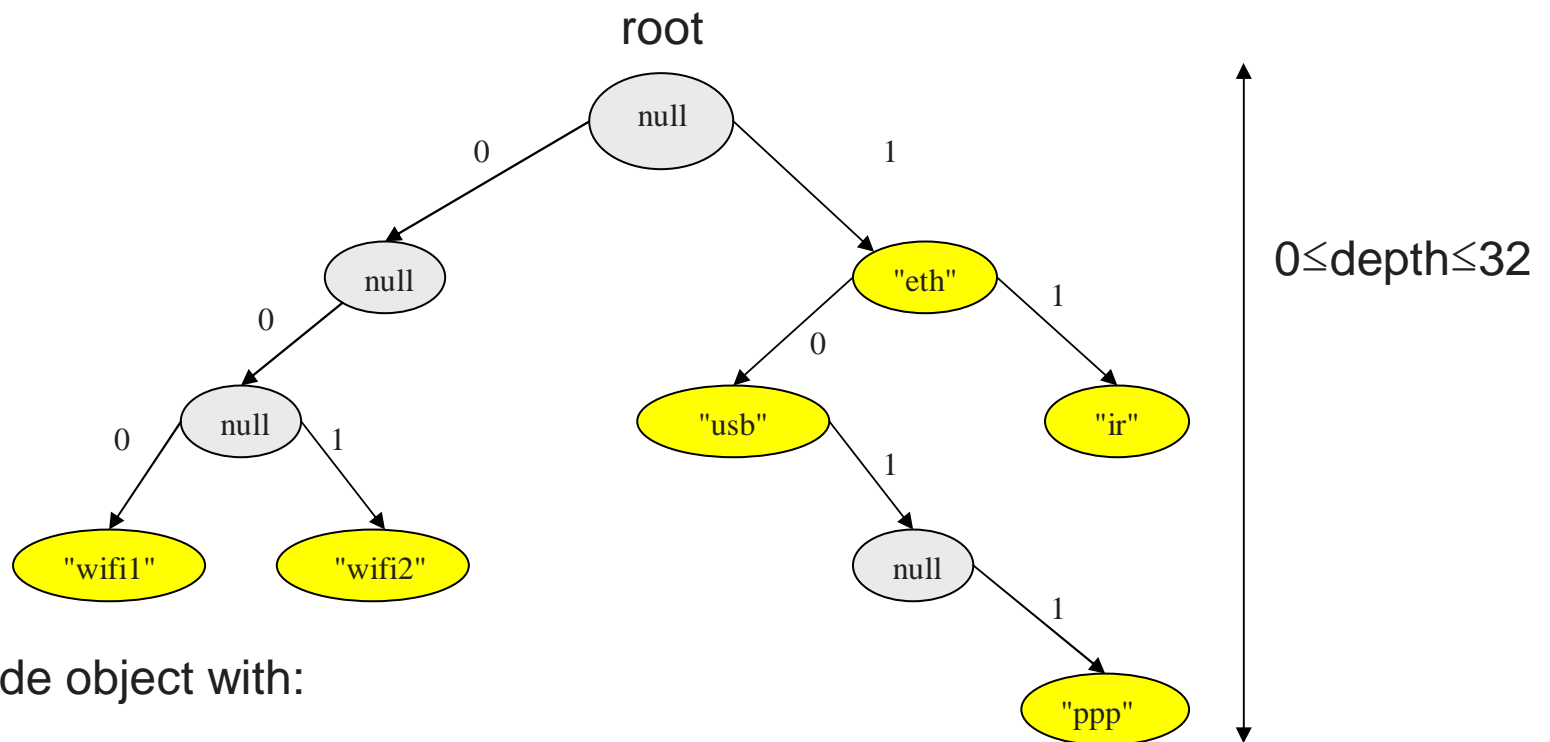
where:

$a_1 \dots a_{32}$ is the binary representation of address

A query

Implementation

- Based on a binary trie:



is a Node object with:

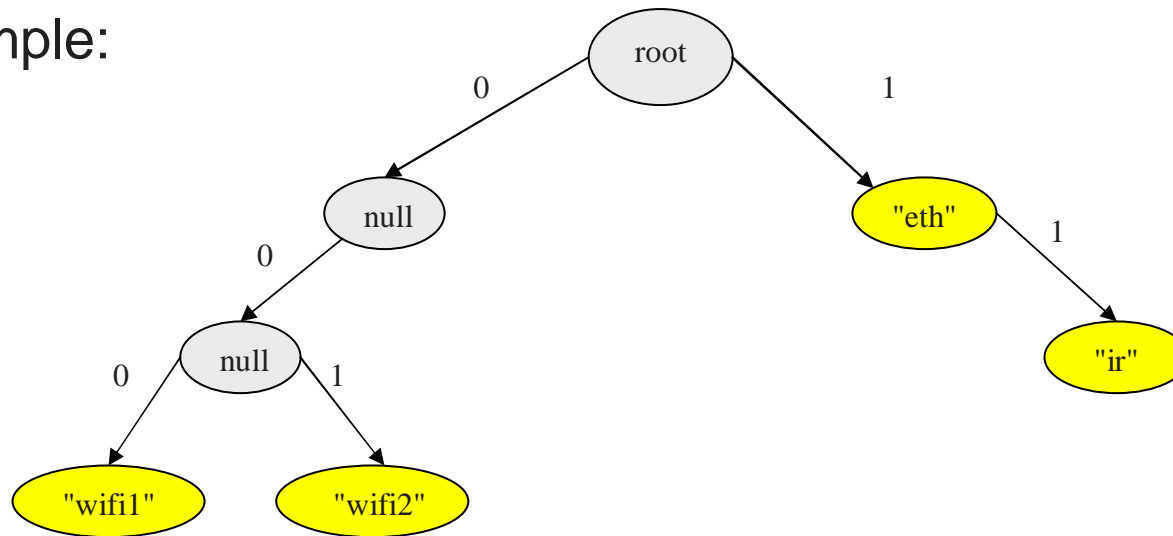
String value (can be null)

Node one,zero (can be null)

Representation Invariant

1. $root \neq null$
2. $0 \leq depth(trie) \leq 32$
3. $A(this) = \{(b_1 \dots b_n, s) \mid root \xrightarrow{b_1 \dots b_n} v_n \text{ with value } s \neq null \text{ in a trie}\}$

Example:



$$A(this) = \{(000, wif1), (001, wif2), (1, eth), (11, ir)\}$$

Class Correctness Proof

■ Invariant Verification at initial state:

- | | |
|--|----------------------------|
| 1. <code>root = new Node()</code> | (field initialization) |
| 2. <code>root != null</code> | (1) ■ |
| 3. <code>(root.zero == null) and (root.one == null)</code> | (1, Node's initialization) |
| 4. <code>depth(trie) == 0</code> | (3) ■ |
| 5. <code>A(this) = ∅</code> | (4) ■ |
| 6. All three invariants are satisfied at initial state | (2,4,5) |

Class Correctness Proof (cont.)

■ add:

add(address, n, value) extracts the n high-order bits of address: $a_1 \dots a_n = \text{key}[0, \dots, n-1]$.
The method tries to traverse a path: $\text{root} = \text{node}_0 \xrightarrow{\text{key}[0, \dots, i < n]} \text{node}_{i+1 \leq n} \in A(\text{old})$
such that: $i = n - 1$ or $\text{node}_i \xrightarrow{\text{key}[i]} \text{null}$

Then:

if $i = n - 1$: $\text{node}_n.\text{value} = \text{value}$, that is $(\text{key}[0, \dots, n - 1], s)$ is replaced with $(\text{key}[0, \dots, n - 1], \text{value})$



post-cond. $A(\text{new}) = A(\text{old}) / \{(a_1 \dots a_n, s)\} \cup \{(a_1 \dots a_n, \text{value})\}$ and inv. 3 are satisfied

otherwise: the path $\text{node}_i \xrightarrow{\text{key}[i, \dots, n-1]} \text{node}_n$ is added, where:

$\forall i + 1 \leq j \leq n - 1 \text{ node}_j.\text{value} = \text{null} \text{ and } \text{node}_n.\text{value} = \text{value}$



post condition $A(\text{new}) = A(\text{old}) \cup \{(a_1 \dots a_n, \text{value})\}$ and invariant 3 are satisfied

Class Correctness Proof (cont.)

■ add (cont.):

- the method does not modify the reference to root \Rightarrow
root \neq null and invariant 1 is satisfied
- $1 \leq n \leq 32$ (pre-condition) \Rightarrow
a path no longer than 32 nodes (excluding the root) is added \Rightarrow
 $1 < \text{depth}(\text{trie}) \leq 32$ and invariant 2 is satisfied

Class Correctness Proof (cont.)

■ route:

- the method does not modify the reference to root \Rightarrow
root \neq null and invariant 1 is satisfied
- The trie is unchanged \Rightarrow
A(new)=A(old), $0 \leq \text{depth}(\text{trie}) \leq 32 \Rightarrow$
post condition 1 and invariants 2 and 3 are satisfied

Class Correctness Proof (cont.)

■ route (cont.):

The method represents address as a 32-bit array: $a_1 \dots a_{32} = \text{key}[0, \dots, 32]$.

If a `NoRouteToHostException` is thrown:

$\forall 1 \leq n \leq 32: \exists \text{root} \xrightarrow{\text{key}[0, \dots, n-1]} \text{node}_n \Rightarrow \text{node}_n.\text{value} = \text{null}$ (code)



$\neg \exists (b_1 \dots b_n, s) \in A(\text{this}): b_1 \dots b_n = \text{key}[0, \dots, n-1]$ (invariant 3)



$\forall (b_1 \dots b_n, s) \in A: b_1 \dots b_n \neq a_1 \dots a_n$



post condition 2 is satisfied

Class Correctness Proof (cont.)

■ route (cont.):

Otherwise:

$returned_value \neq null$ (code)



1. $\exists 1 \leq n \leq 32: root \xrightarrow{key[0, \dots, n-1]} node_n$ and $node_n.value = returned_value \neq null$

2. $n = 32$ or $node_n \xrightarrow{key[n]} null$ (code)



$\exists n: (key[0, \dots, n-1], returned_value) \in A(this) \ \& \ \&$

$\forall (b_1 \dots b_{m>n}, s) \in A: b_1 \dots b_n \neq key[0, \dots, m-1]$



post condition 2 is satisfied

Class Correctness Proof (cont.)

- Other Code Verification:

root is private.



No other code can change the invariant.

The Contract – Different Format

```
/**  
 * @pre (1 <= n <= 32) && (value != null)  
 * @post get(address,n) == value  
 */  
public void add(int address, int n, String value)
```

```
/**  
 * @pre (1 <= n <= 32)  
 * @post nothing  
 */  
public String get(int address, int n)
```

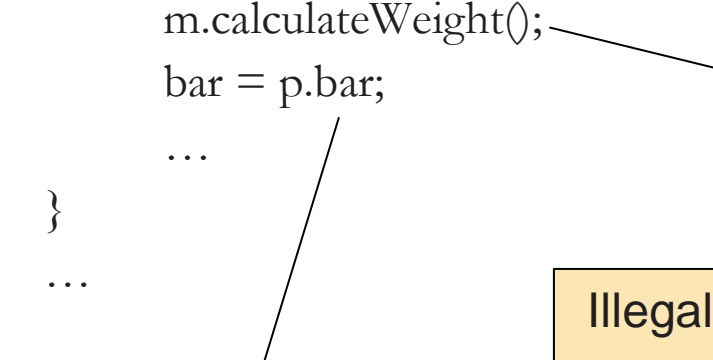
The Contract – Different Format

```
/**
 * @pre nothing
 * @post if an exception is thrown:  $\forall 1 \leq i \leq 32: \text{get}(\text{address}, i) == \text{null}$ 
 * else,  $\exists 1 \leq i \leq 32: \text{get}(\text{address}, i) = \text{returned\_value} \ \&\& \ \forall i < j \leq 32: \text{get}(\text{address}, j) == \text{null}$ 
 **/
public String route(int address) throws
    java.net.NoRouteToHostException
```


Visibility

```
package A;  
  
public class Molecule {  
    ...  
    protected void calculateWeight() {  
        ...  
    }  
    ...  
}
```

```
package B;  
  
public class Protein extends Molecule {  
    private int bar = 5;  
    void foo(Protein p, Molecule m) {  
        calculateWeight();  
        p.calculateWeight();  
        m.calculateWeight();  
        bar = p.bar;  
        ...  
    }  
    ...  
}
```



Exceptions

```
int i=1, j=1;
try {
    i++;
    j--;
    if (i/j > 1)
        i++;
} catch(ArithmeticException e) {
    System.out.println(1);
} catch(Exception e) {
    System.out.println(2);
} finally {
    System.out.println(3);
}
```

The output is:

1
3

By-Value/Reference Arguments

```
public class Test {  
    private static class Value { int v = 1; }  
  
    public static void main(String[] args) {  
        Test test = new Test();  
        int v = 2;  
        Value value = new Value();  
        value.v = 3;  
        foo(value, v);  
        System.out.println(value.v + " " + v);  
    }  
  
    private static void foo(Value value, int v) {  
        v = 4;  
        value.v = 5;  
        value = new Value();  
        System.out.println(value.v + " " + v);  
    }  
}
```

The output is:

1 4

5 2

Interfaces

```
public interface Foo {  
    public void bar()  
        throws Exception;  
}
```

Compilation Error:
"Unhandled exception type Exception"

```
public class FooImpl implements Foo {  
    public void bar() {  
        System.out.println("An exception is not thrown");  
    }  
  
    public static void main(String args[]) {  
        Foo foo = new FooImpl();  
        foo.bar();  
    }  
}
```

Interfaces

```
public interface Foo {  
    public void bar()  
        throws Exception;  
}
```

Output:
No exception is thrown

```
public class FooImpl implements Foo {  
    public void bar() {  
        System.out.println("No exception is thrown");  
    }  
}
```

```
public static void main(String args[]) {  
    FooImpl foo = new FooImpl();  
    foo.bar();  
}  
}
```