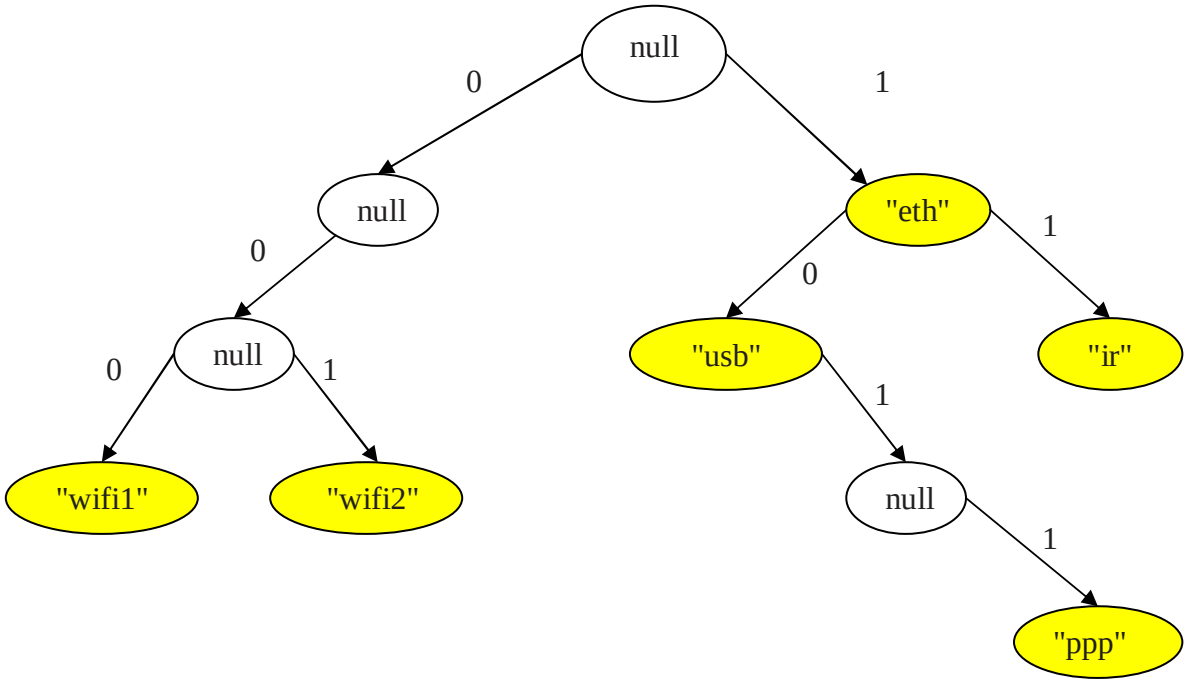
Assignment No. 3

The main purpose of this assignment is to learn a new data structure and practice Design By Contract, both in implementation and proof.

A *trie* or *radix* is a tree-based data structure useful for storing a collection of key-value pairs such as a dictionary. The key of each node in this tree can be determined by traversing the path from the root to the node.

The figure bellow illustrates a trie that stores the bit string keys[1]: 1, 11, 10, 1011, 000, 001. In this binary tree a move to the left denotes a 0 and a move to the right denotes a 1. Thus, when searching for a key $a = a_0a_1a_2...a_n$ we go left at a node of depth i if $a_1 = 0$ and right if $a_1 = 1$. Not all the nodes in the tree represent key-value pairs. Some nodes represent only unique prefixes common to several keys ("0", "00" and "101") and are essential for establishing a path to other nodes. In the figure below, only the yellow nodes represent key-value pairs. One should note that there is no need to actually store the keys in the nodes, since they can be inferred by traversing the path from the root to the node (but there is a need to define that a node represents a key).



Your assignment is to design a Java class that implements a routing table using a trie. A *routing table* is an object that supports one command and one query (in practice, there is another command, *delete*, that you do not have to define in this exercise):

- *public void add(int address, int n, String value)* – adds a key-value pair to the table. The key is a 0-1 string of up to 32 bits, specified using an integer *address* and an integer *n* that specifies the

[1]Please note that a trie need not be binary. Any alphabet is possible and the degree of nodes is the size of the alphabet.

length of the string; the key consists of the *n* high-order bits of *address*; the value is a string (in practice, the value would be a routing decision, but here we'll use a string)

- public String *route(int address) throws java.net.NoRouteToHostException;* Returns the value associated with the **longest prefix in the routing table** that matches the argument. If no prefix matches the given address, an exception is thrown.

For example, the routing table above was created by the commands (we use here binary notation, which is not legal syntax in Java, to make the presentation simpler, and we use 8-bit rather than 32-bit integers):

```
t = new RoutingTable();
t.add(10000000b,1,"eth");
t.add(11000000b,2,"ir");
t.add(10000000b,2,"usb");
t.add(10110000b,4,"ppp");
t.add(00000000b,3,"wifi1");
t.add(00100000b,3,"wifi2");
```

On this object, the query **`t.route(01000000b)`** should throw an exception, and the query **`t.route(10101111b)`** should return **`"usb"`**.

Carefully define the contract, implement the class, define the representation invariant, and prove the class correctness.