Traversing a tree

Johan Montelius

KTH

HT22

traversing a tree

traversing a tree

Many ways of traversing a tree:

- depth-first: in-order, pre-order , post-order (*reversed Polish*)
- breath-first: one level at a time

traversing a tree

Many ways of traversing a tree:

- depth-first: in-order, pre-order , post-order (*reversed Polish*)
- breath-first: one level at a time

also called: in-fix, pre-fix and post-fix

depth-first in-order

```
public void inorder() {
  if (root != null)
      root.inorder()
}
```

```
public void inorder() {
  if (left != null)
    left.inorder()
  this.do_something();
  if (right != null)
     right.inorder()
}
```

an example

an example

Let's represent an arithmetic expression as a tree.

the fibonacci sequence

the fibonacci sequence

0,1,1,2,3,5,8,13,21 .....

the fibonacci sequence

0,1,1,2,3,5,8,13,21 .....

```
public static int fib(int n) {
  if (n == 1)
    return 0;
  if (n == 2)
    return 1;
  return ..... ;
}
```

an Iterator

```
Fib fib = new Fib();

Iterator<Integer> iter = fib.iterator();

Integer f1 = ite.next()
Integer f2 = ite.next()
Integer f3 = ite.next()
   :
```

an Iterator

```
Fib fib = new Fib();

Iterator<Integer> iter = fib.iterator();

Integer f1 = ite.next()
Integer f2 = ite.next()
Integer f3 = ite.next()
   :
```

```
import java.util.Iterator;
```

the Iterator

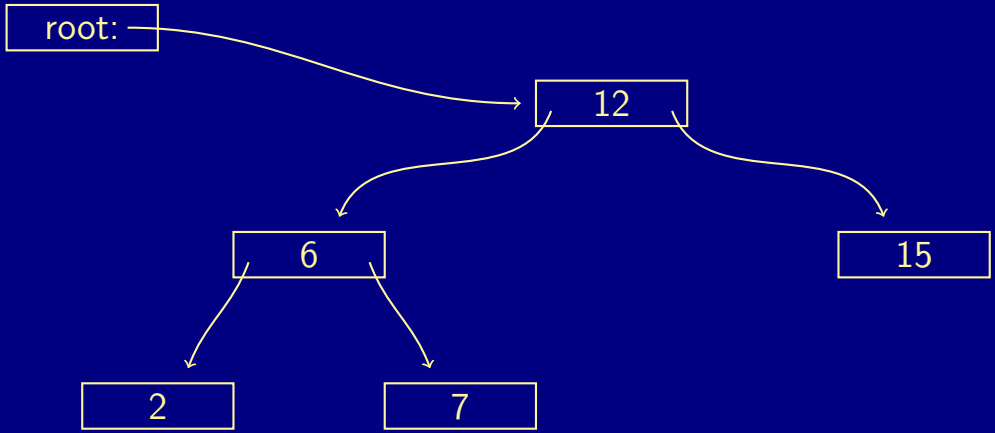
```
public class FibIterator implements Iterator<Integer>
        :
    @Override
    public boolean hasNext() {
      :
    }
    @Override
    public Integer next() {
       :
    }
    @Override
    public void remove() {
        :
    }
}
```

Iterable

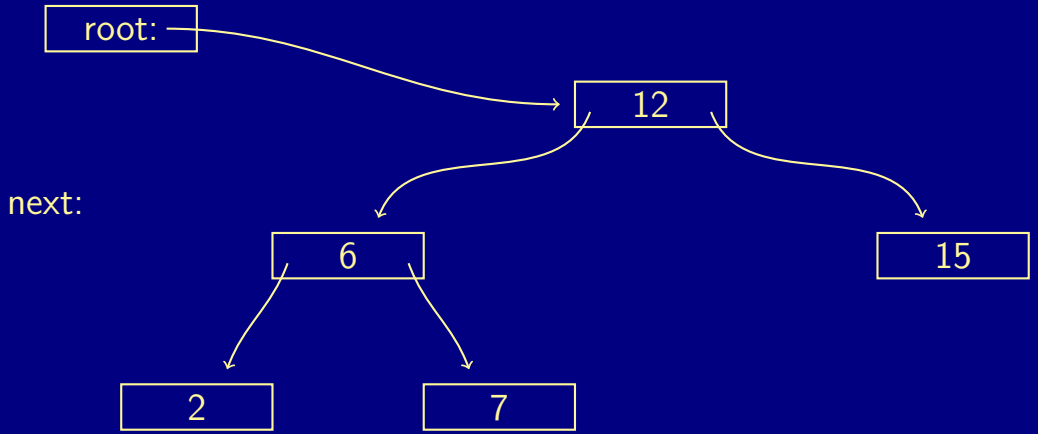
```
public class Fib implements Iterable<Integer> {
      :
    public Iterator<Integer> iterator() {
        return new FibIterator();
    }

    :
}
```

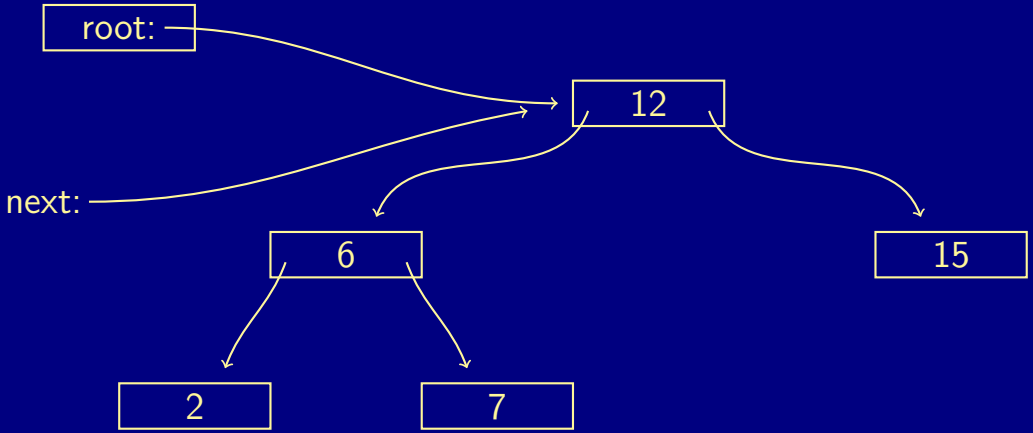
an Iterator for a tree



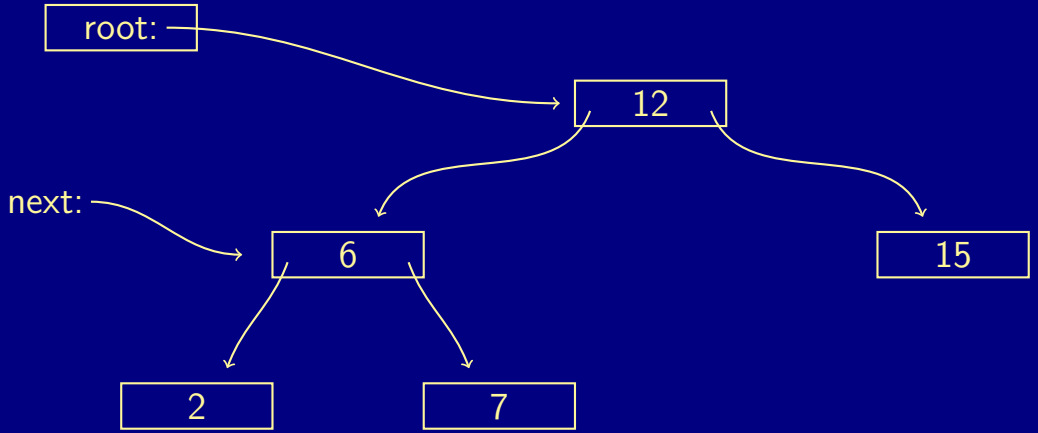
an Iterator for a tree



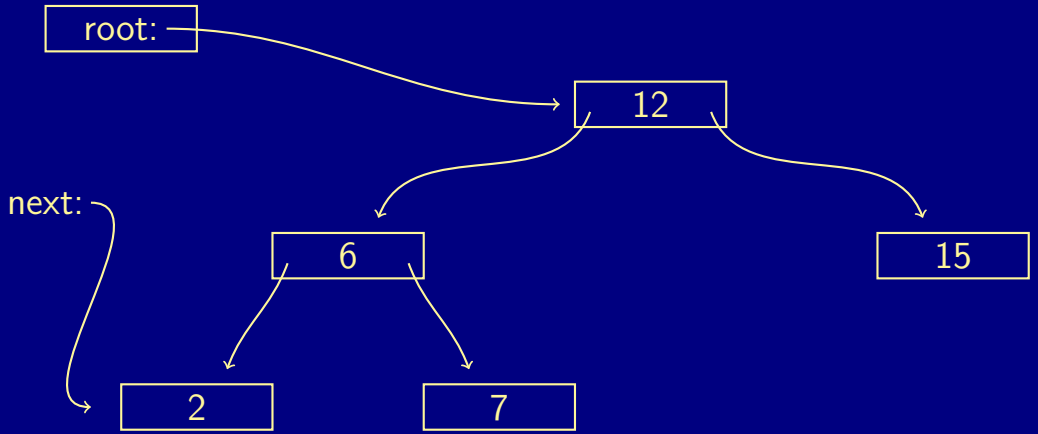
an Iterator for a tree



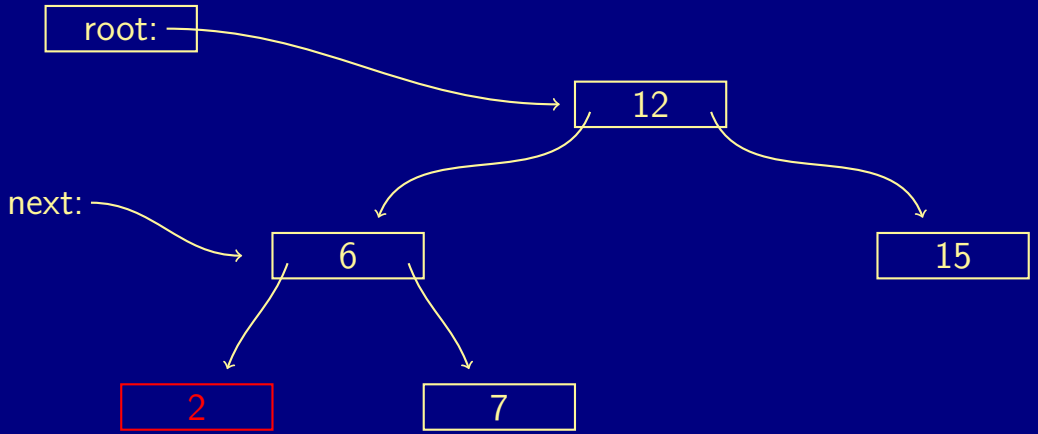
an Iterator for a tree



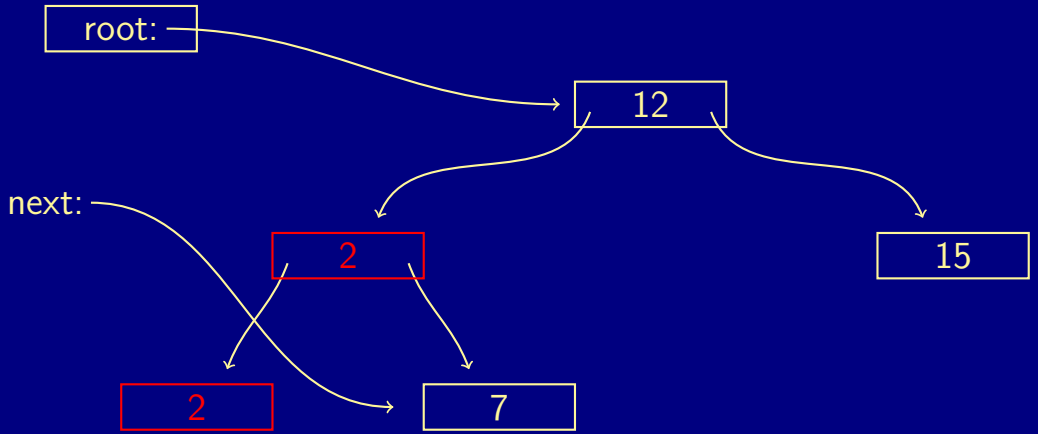
an Iterator for a tree



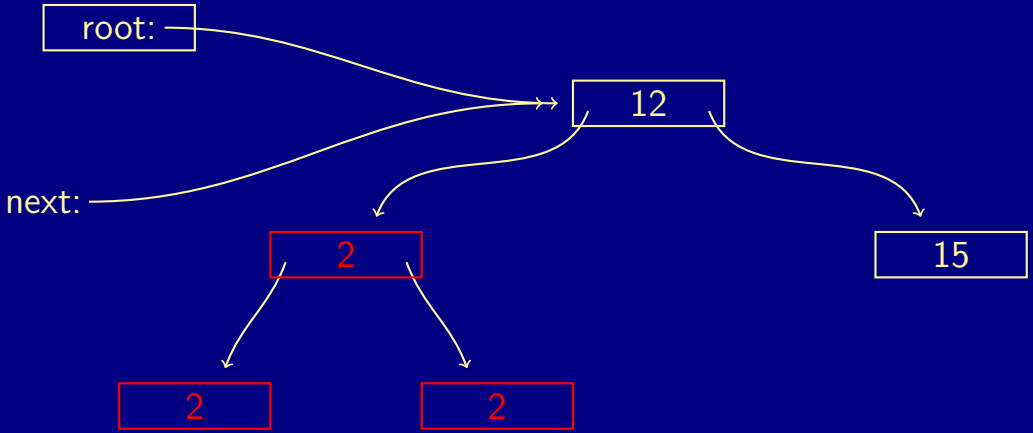
an Iterator for a tree



an Iterator for a tree



an Iterator for a tree



an Iterator for a tree

- You need to find your way back.
- Why not a stack?