UCLouvain

icteam

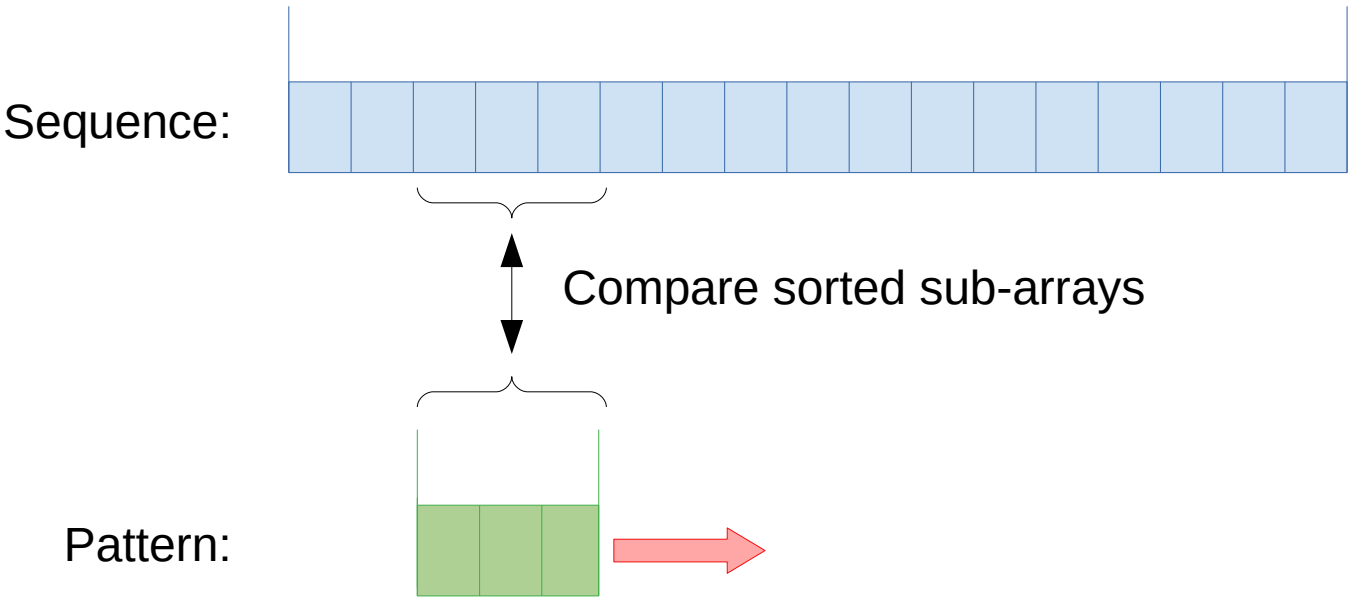
(1) Quiz – Find pattern

```java
public class FindPattern {
    /**
     * In this task you must find the first occurrence of a pattern in a
     * sequence.
     * This occurrence might also be **in a different order** than in
     * the pattern.
     *
     * For example, let:
     *      pattern = [1, 1, 3, 5]
     *      sequence = [3, 1, 2, 5, 1, 3, 1, 5, 1]
     *
     * Starting at index 3, we have the sub-sequence [5, 1, 3, 1] which
     * is the pattern reordered. Thus your method must return 3.
     * Note that the pattern also appears at index 4, but you must
     * return the first occurrence.
     * If the pattern is not in the sequence, you must return -1.
     */
}
```

UCLouvain

icteam

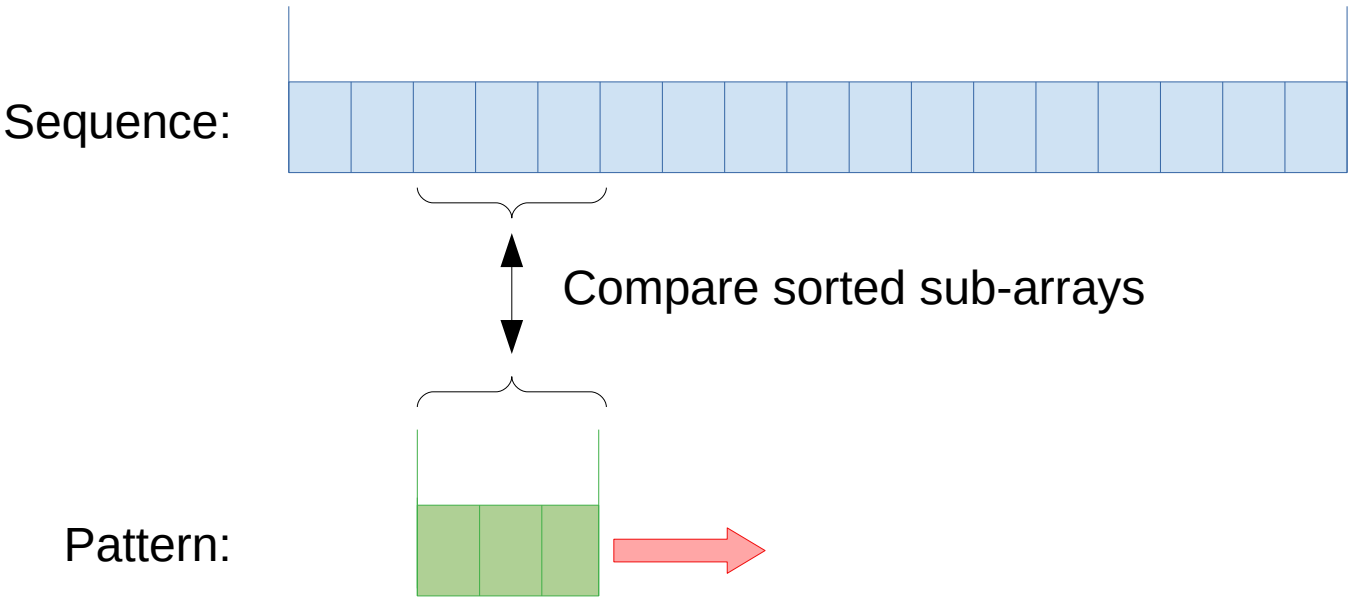
Version 1



UCLouvain

icteam

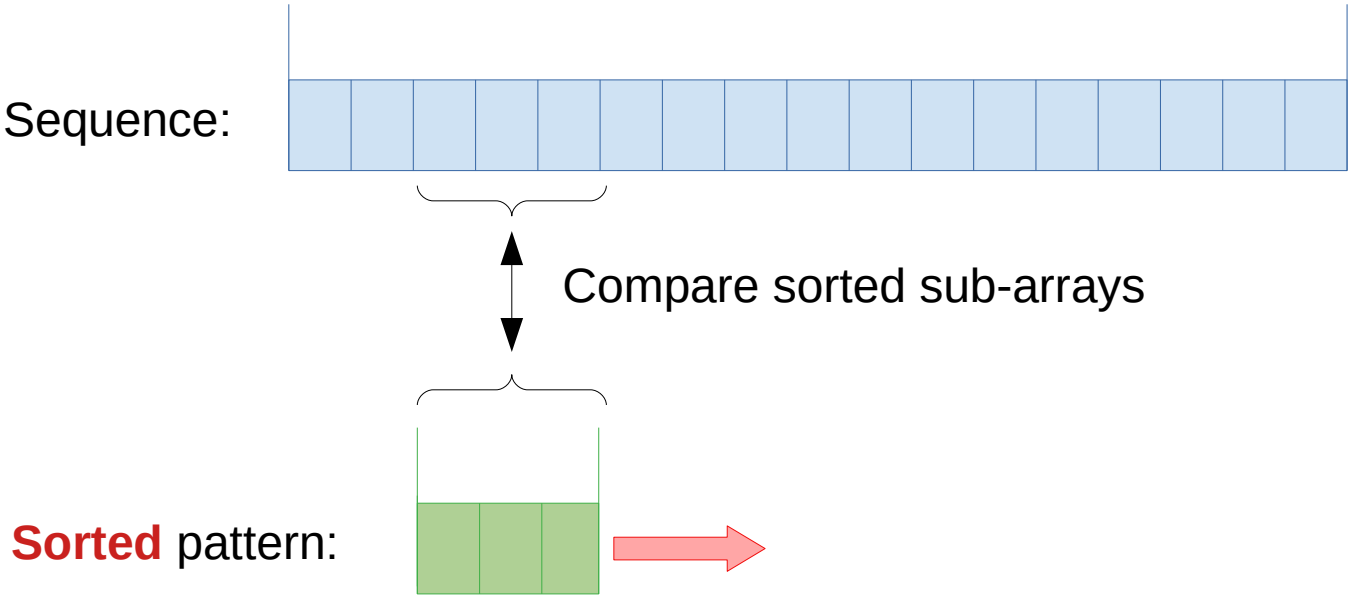
Version 1



We must:

1) Sort sub-arrays: `int[] sort(int[] sequence, int start, int length)`
2) Compare sub-arrays: `boolean compare(int[] a, int[] b)`

Version 2 – Minor optimization



The pattern can be sorted only once!

UCLouvain

icteam

Version 3 – Without sorting

The complexity of the internal loop depends on the length of the pattern!

Could this be avoided?

```
public class FindPattern {
    /**
     * ...
     * @param pattern The pattern to look for
     * @param sequence The sequence to look in. Each element of the sequence is
     *        in the interval [0, 15]
     * @return The index of the first occurrence of the pattern in the
     *         sequence or -1 if the pattern is not in the sequence
     */
}
```

UCLouvain

icteam

Version 3 – Without sorting

The complexity of the internal loop depends on the length of the pattern!

Could this be avoided?

```
public class FindPattern {
    /**
     * ...
     * @param pattern The pattern to look for
     * @param sequence The sequence to look in. Each element of the sequence is
     *        in the interval [0, 15]
     * @return The index of the first occurrence of the pattern in the
     *         sequence or -1 if the pattern is not in the sequence
     */
}
```

Let's take advantage of this [0, 15] range!

UCLouvain

icteam

Remember the “Anagram” exercise (module 1)?

S A N T A

S A T A N

Remember the “Anagram” exercise (module 1)?

S A N T A

S A T A N

Signature

```
A : 2
B : 0
C : 0
D : 0
E : 0
...
N : 1
...
S : 1
T : 1
...
```

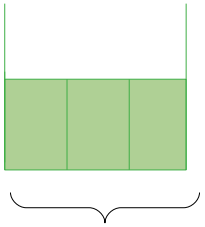
We can compare signatures of length 16!

UCLouvain

icteam

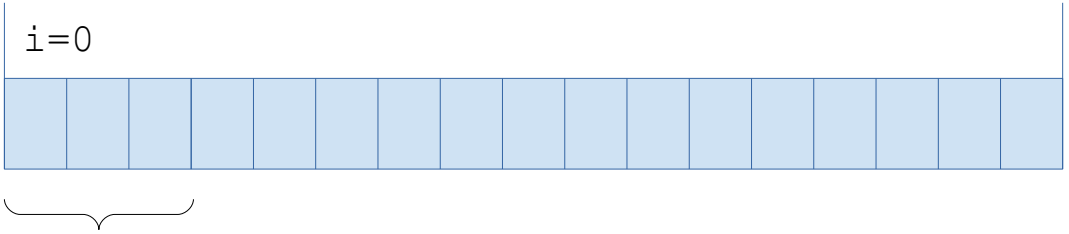
Efficient update of the signature of the sub-sequence

(1) Pattern signature:
(size = 16)



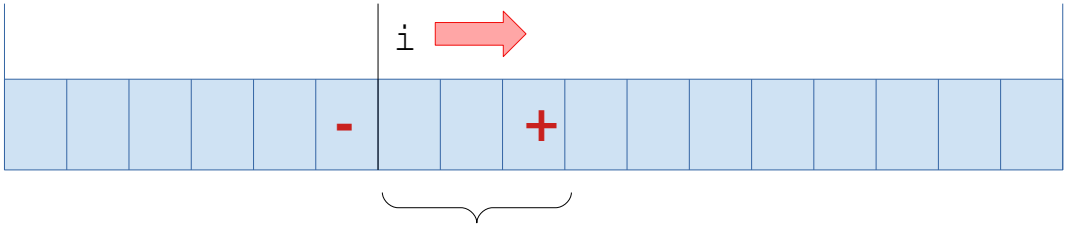
`int[] patternSignature`

(2) Initial sequence signature:



`int[] sequenceSignature`

(3) Update:



UCLouvain

icteam

(2) Quiz – Merge sorted lists, no duplicates

```java
public class MergeSortedLinkedListDuplicate {
    /**
     * You receive two linked lists containing elements in increasing order.
     * You are asked to return a new linked list that contains the
     * elements of both linked lists in increasing order but without duplicates.
     * The input linked lists must remain unchanged.
     * Moreover, the final linkedList must not contain duplicate values
     * That is, instead of 1-1-2-5, you must return 1-2-5.
     *
     * The complexity of your method must be in O(n+m)
     */
    public static Node merge(Node list1, Node list2) {
        // TODO
    }
}
```

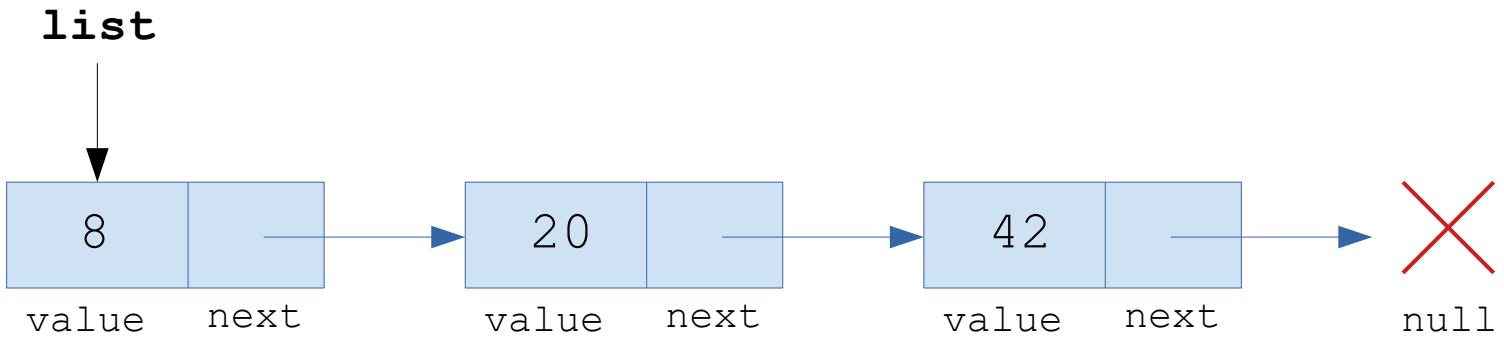
We must stick to the data structure defined by class “Node”

UCLouvain

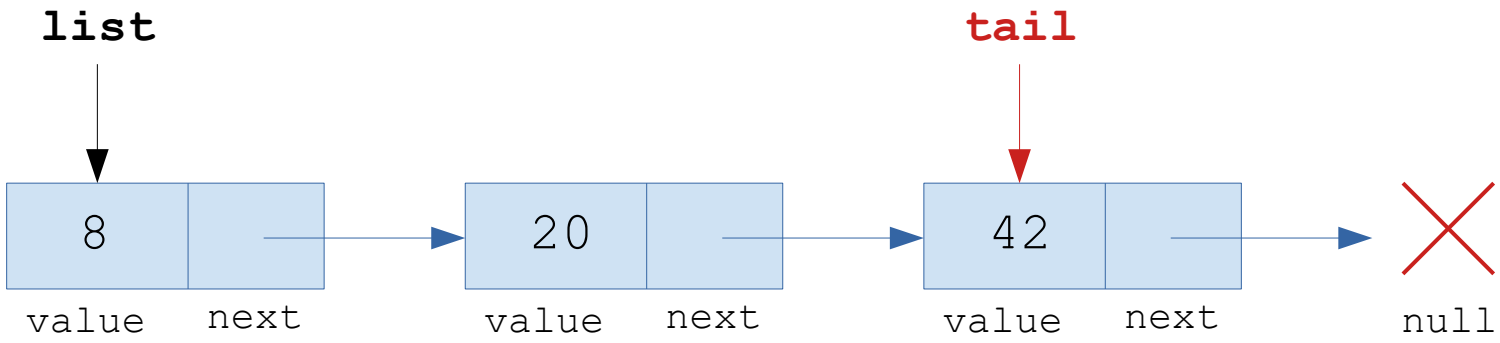
icteam

```
static class Node {
    int value;
    Node next;

    public Node(int value, Node next) {
        this.value = value;
        this.next = next;
    }
}
```



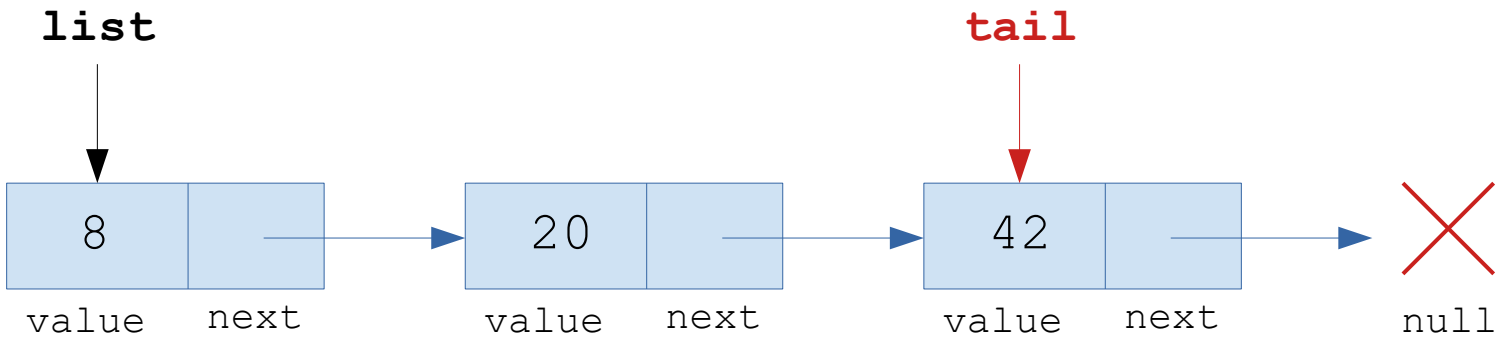
Appending without duplicate, in constant time



UCLouvain

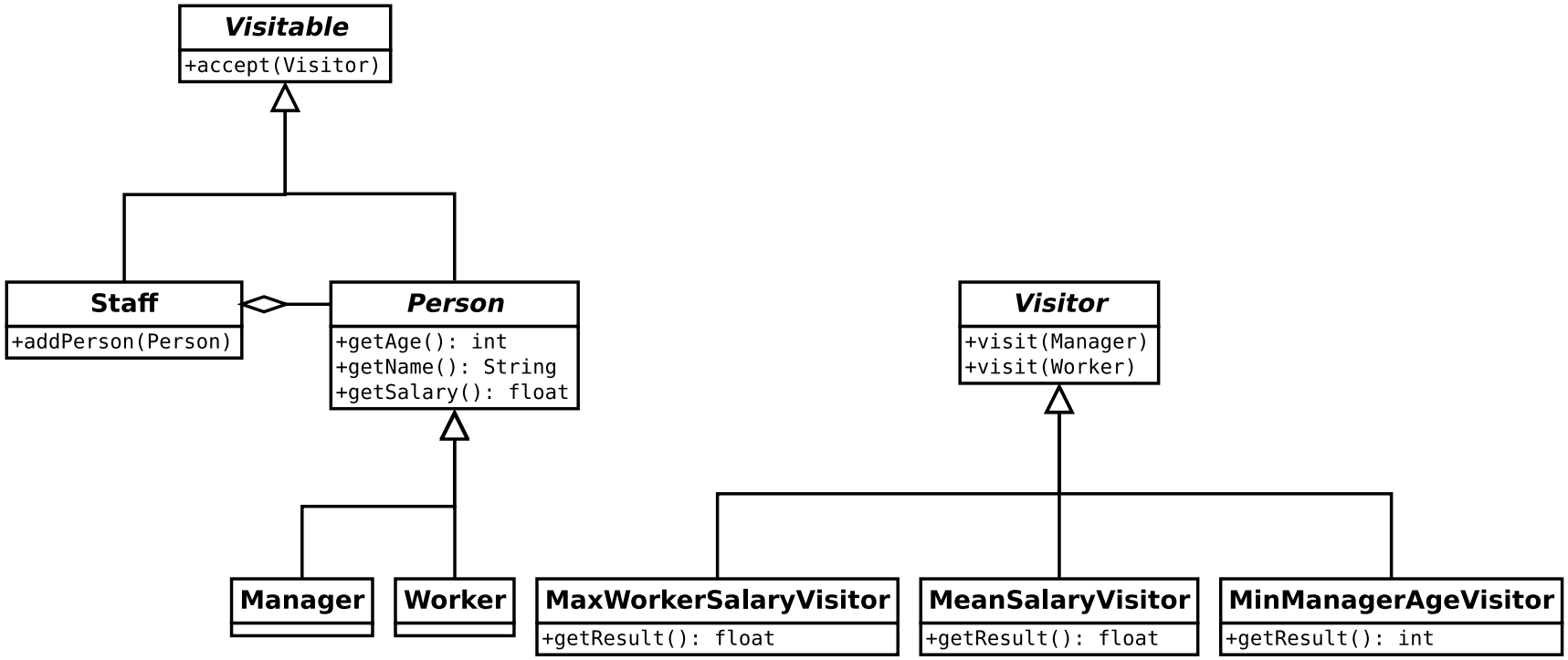
icteam

Appending without duplicate, in constant time

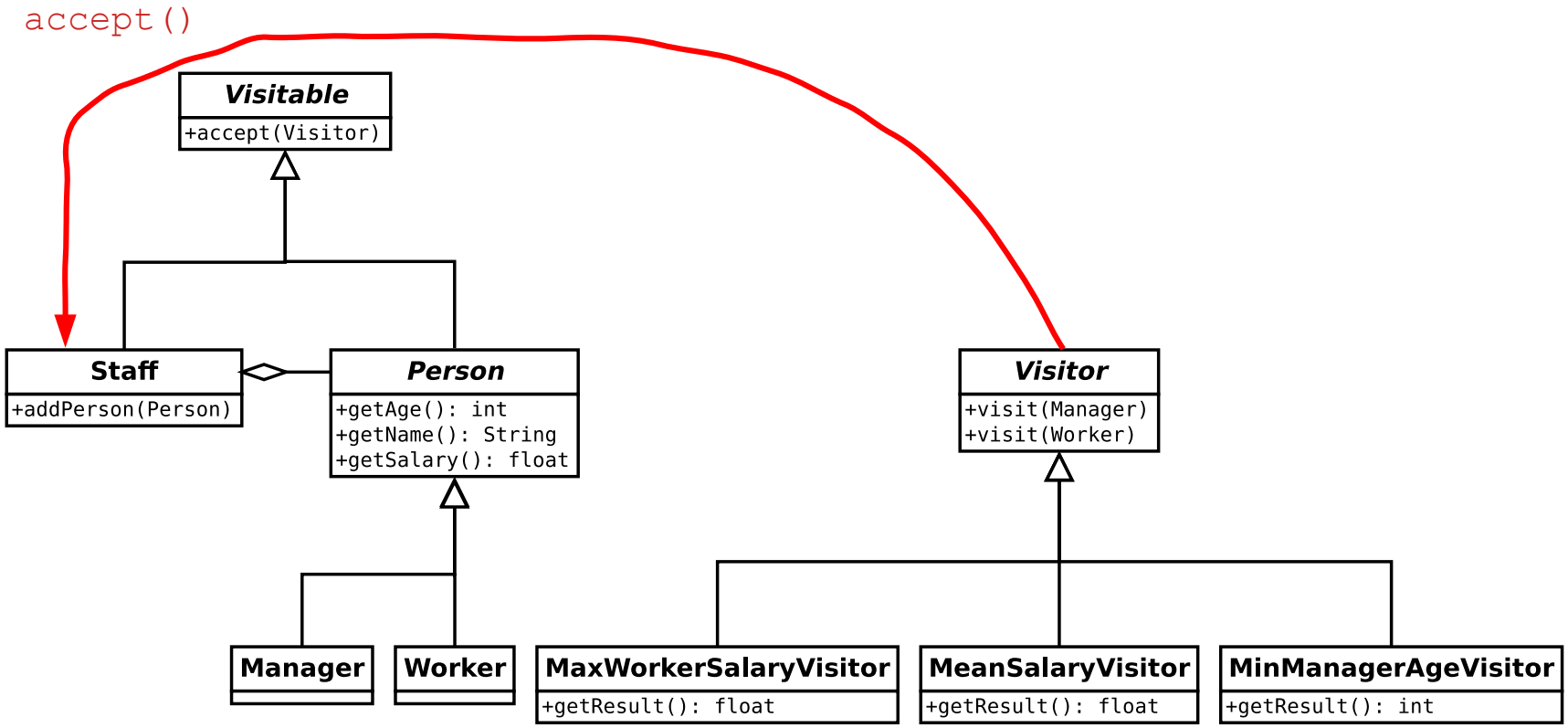



```
public void append(int value) {
    if (list == null) {
        list = new Node(value, null);
        tail = list;
    } else if (value != tail.value) {
        Node node = new Node(value, null);
        tail.next = node;
        tail = node;
    }
}
```

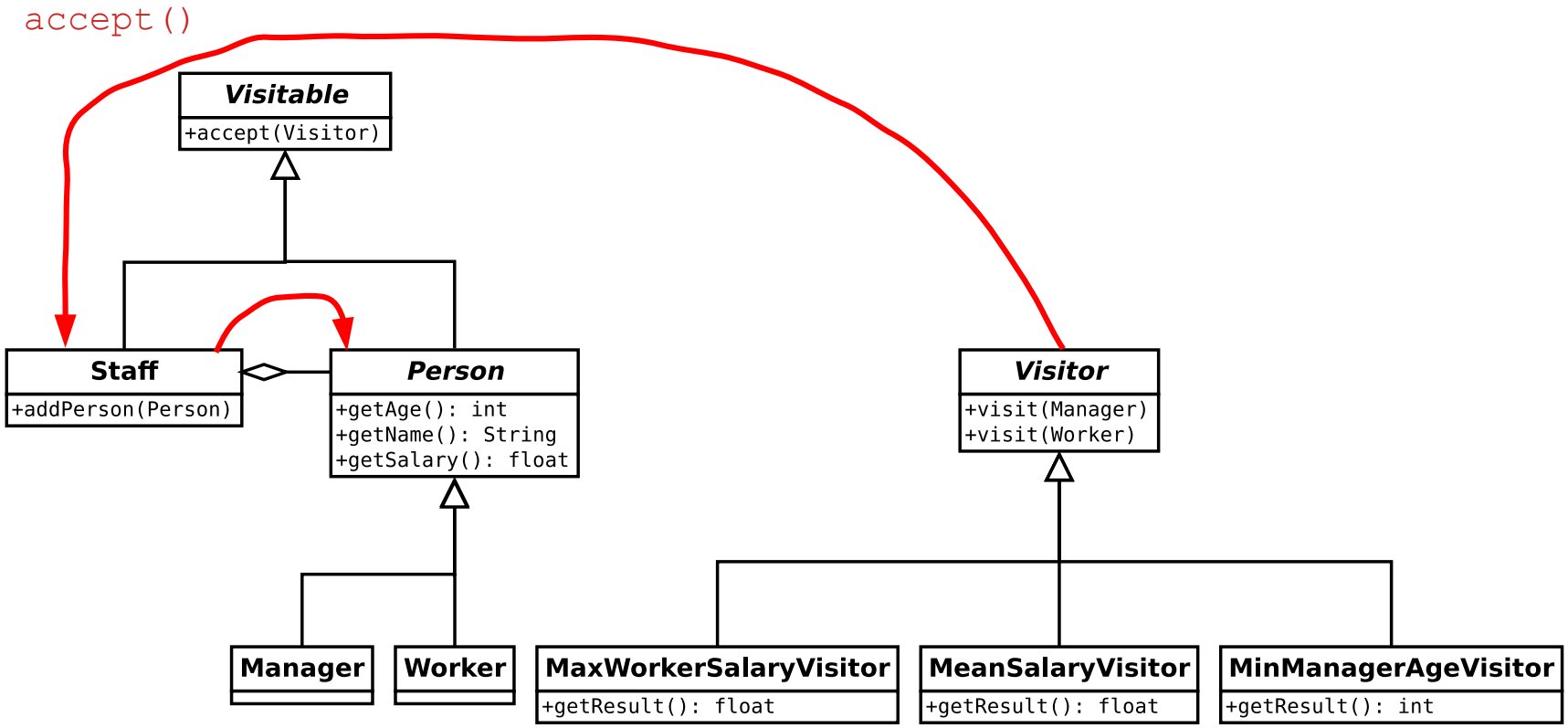
(3) Staff visitor



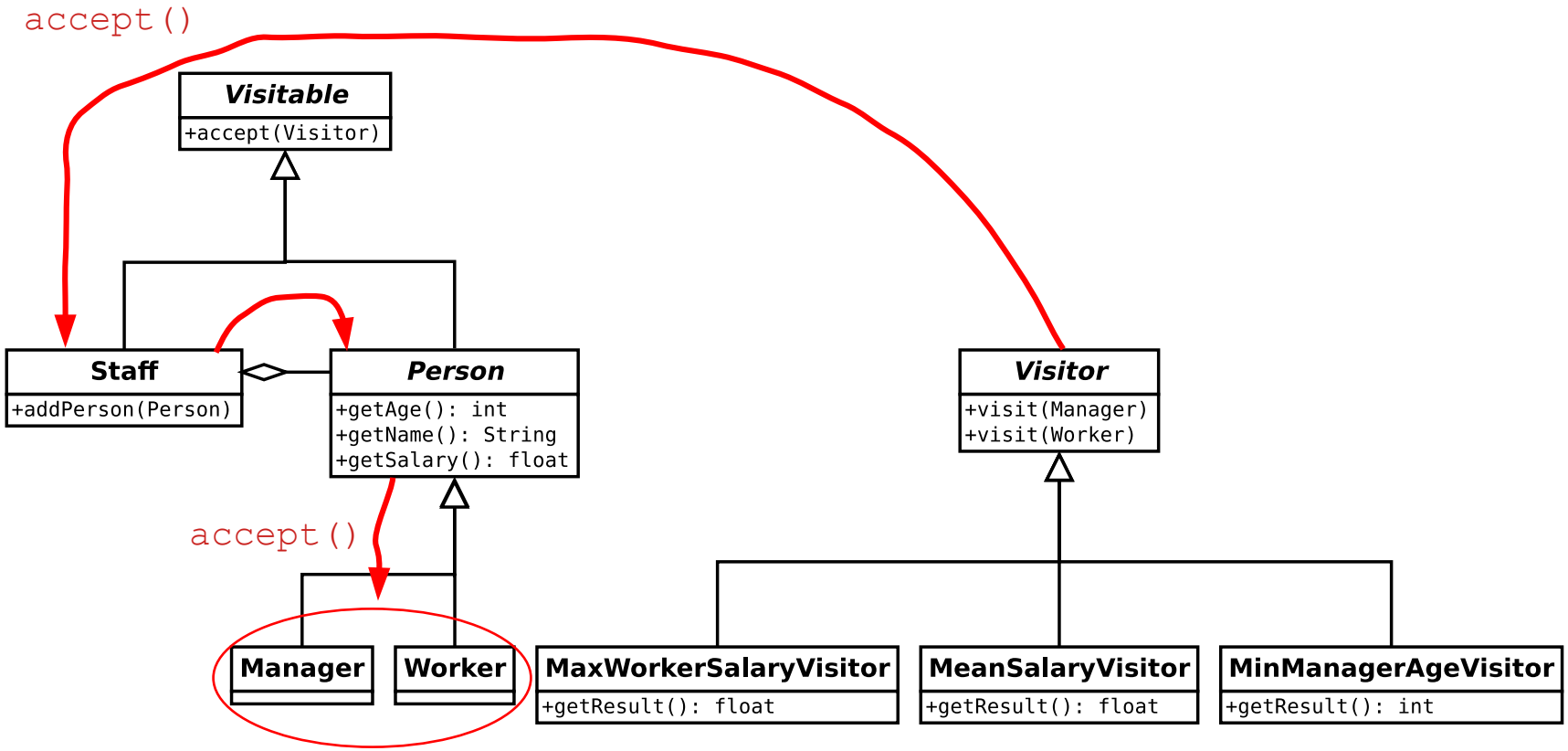
(3) Staff visitor



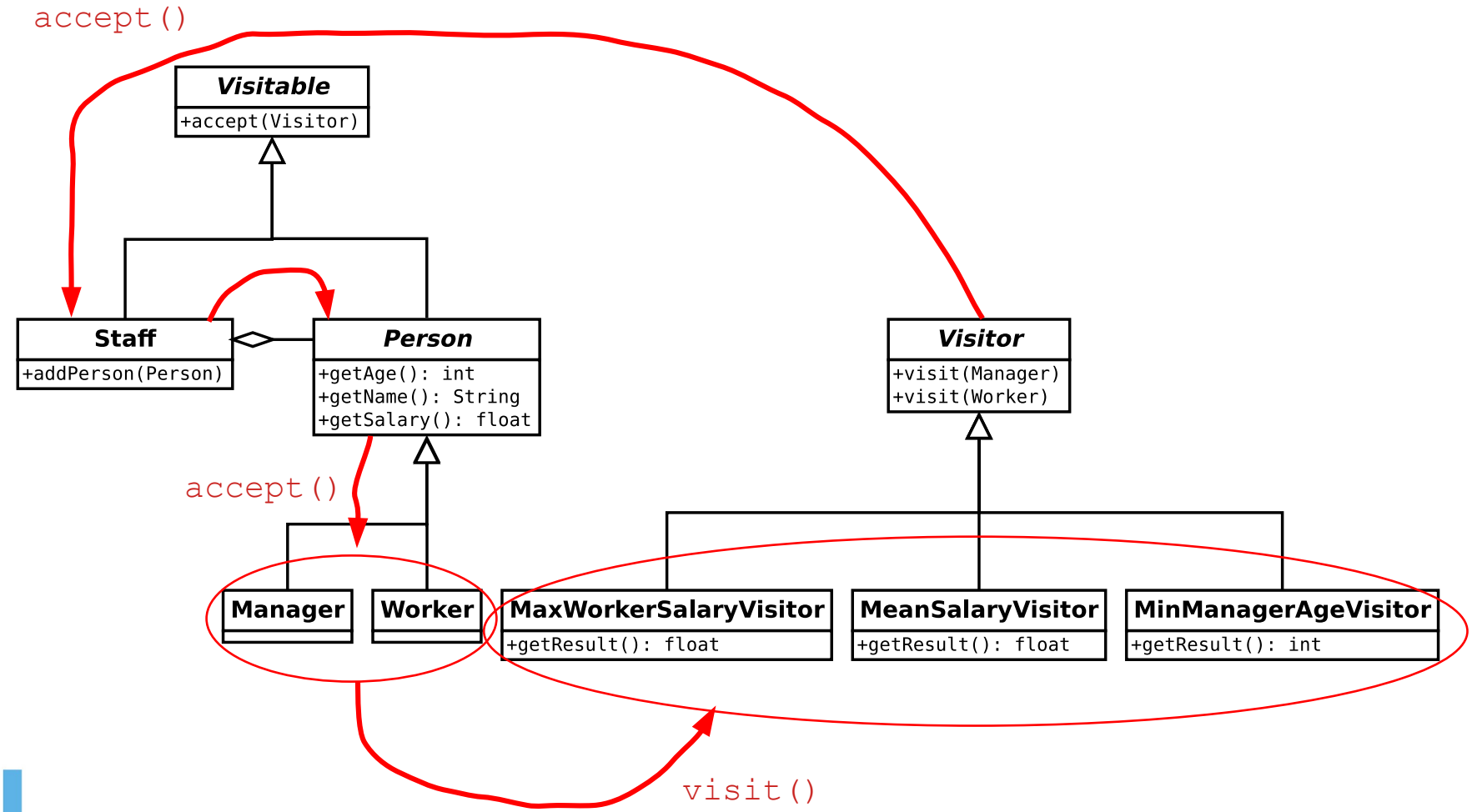
(3) Staff visitor



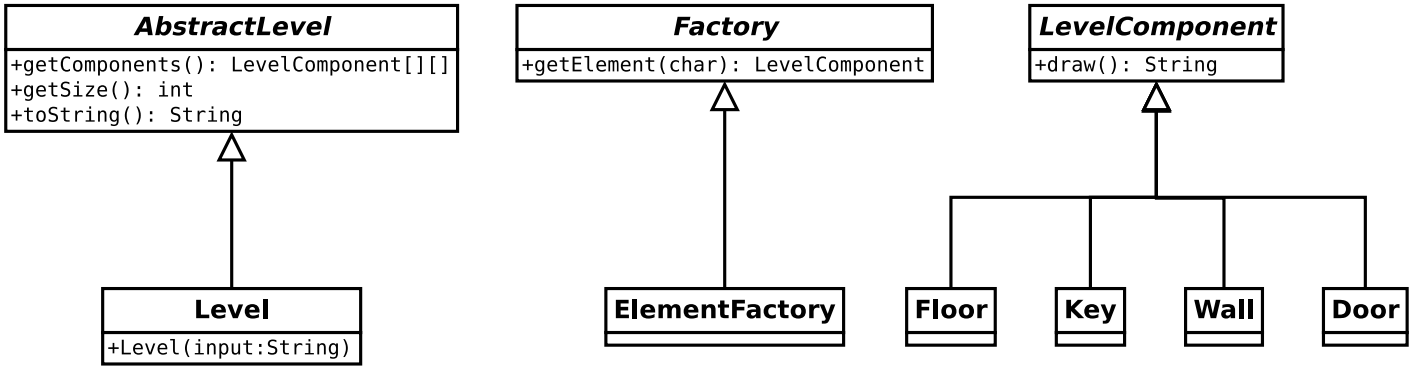
(3) Staff visitor



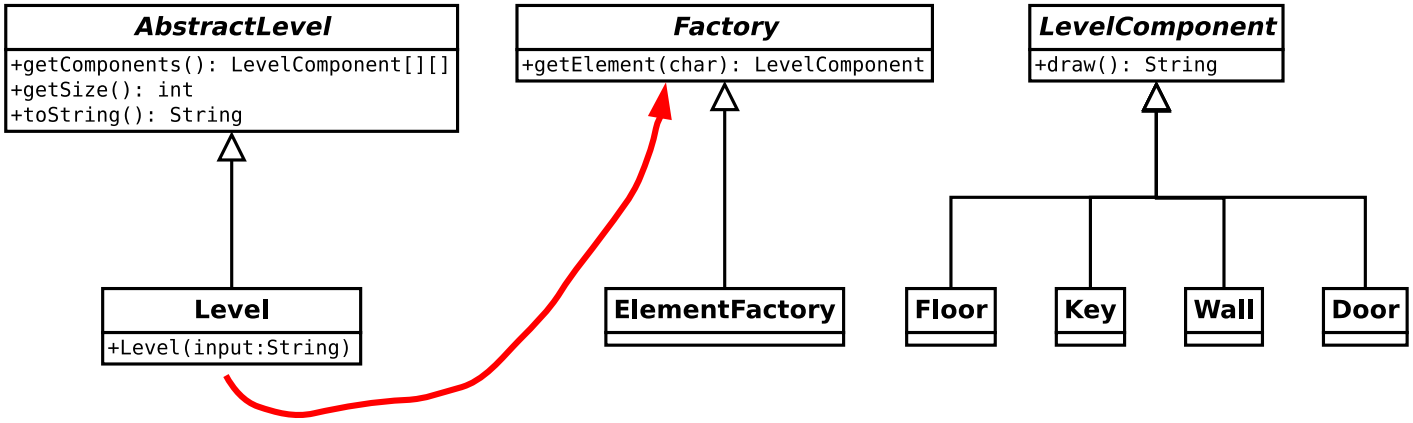
(3) Staff visitor



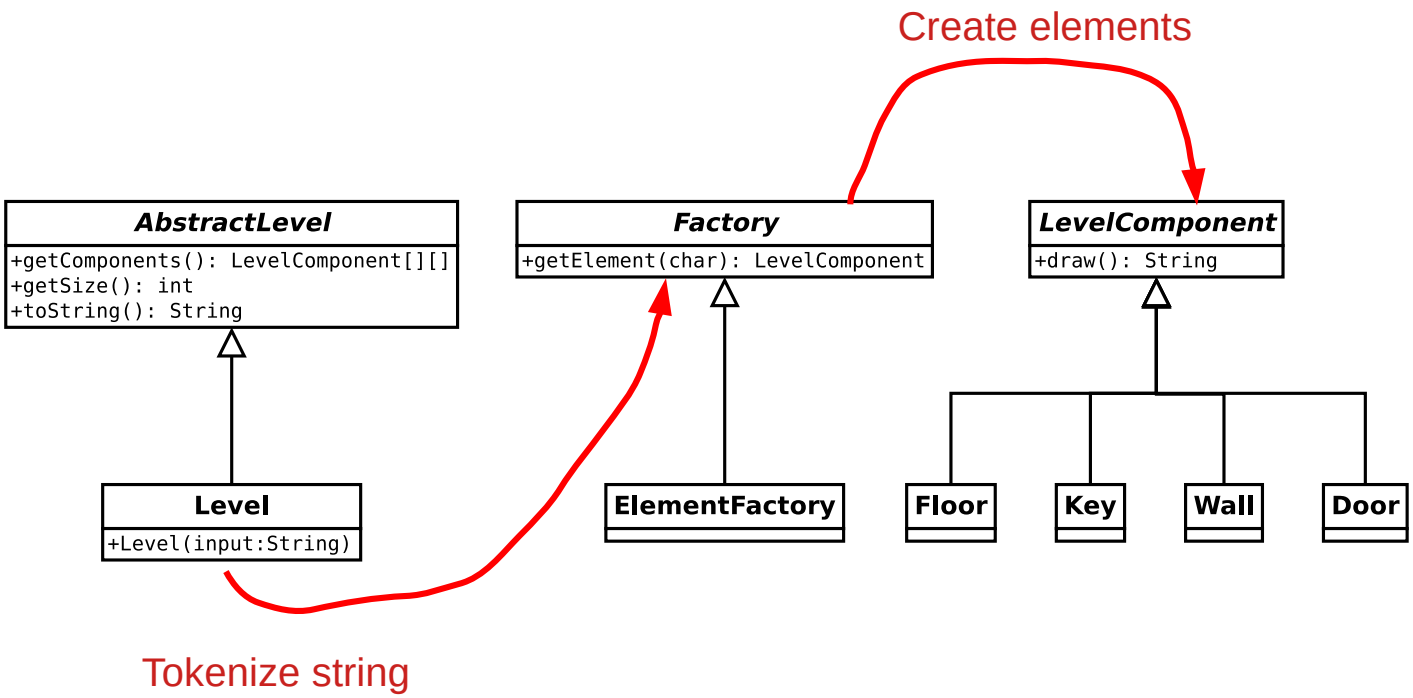
(4) Level generator (Factory)



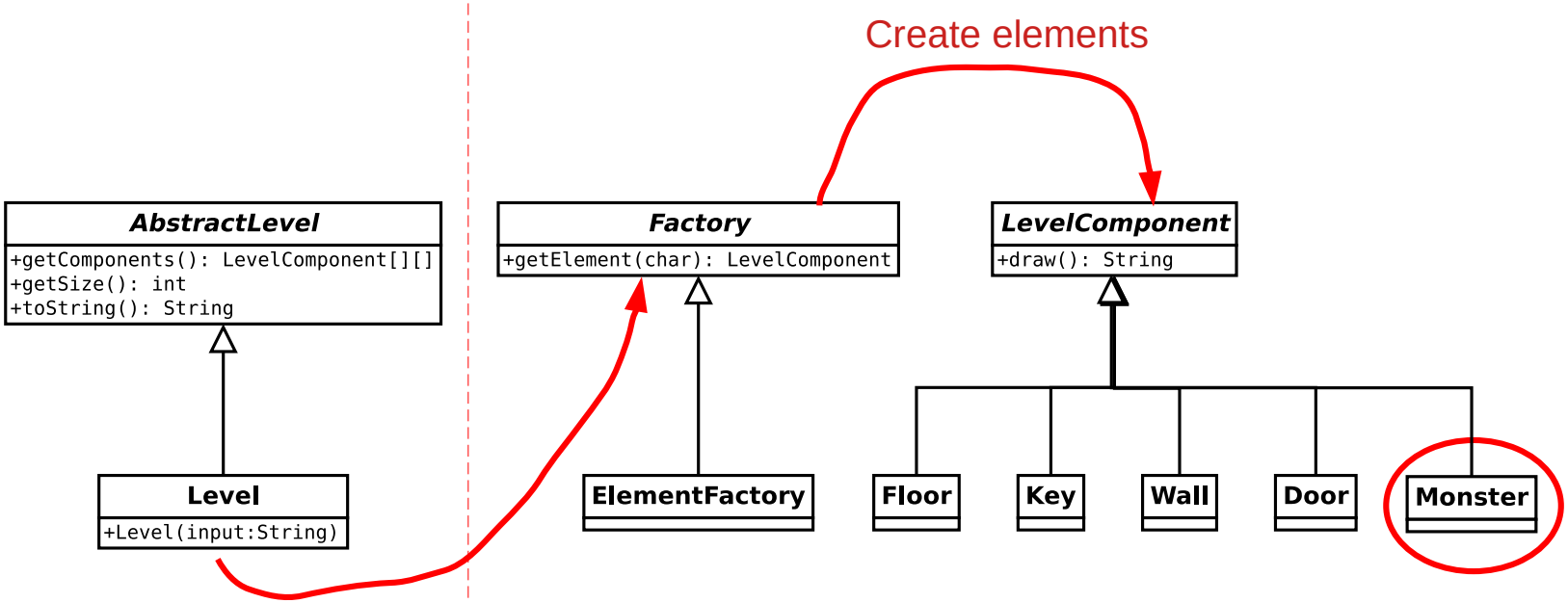
(4) Level generator (Factory)



(4) Level generator (Factory)



(4) Level generator (Factory)



UCLouvain

icteam

Singleton

```
public class ElementFactory extends Factory {

    private static ElementFactory instance = null;

    public static ElementFactory getInstance() {

        if (instance == null) {
            instance = new ElementFactory();
        }

        return instance;
    }
}
```

Only create heavyweight objects once!