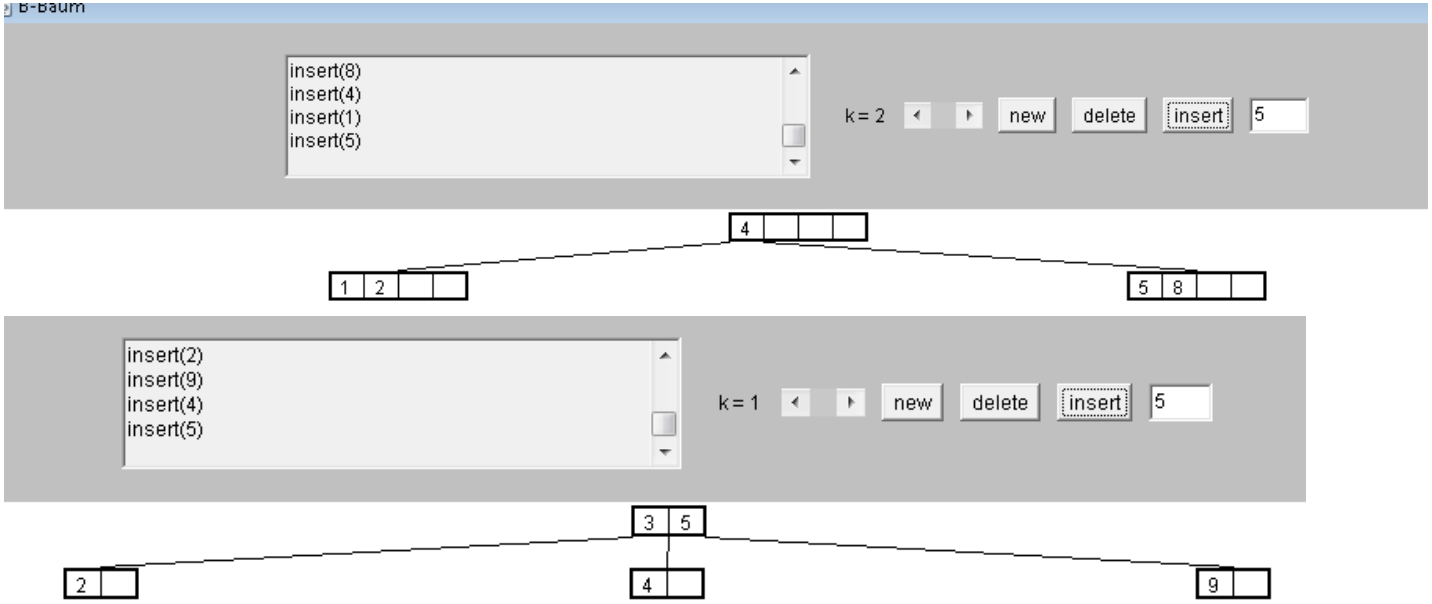
B-Trees (B-Bäume)

B does not stand for binary but for its "inventor" Rudolph Bayer



Main characteristics

1. each node (with exception of the root node) contains between k and 2k values.
2. each node (with exception of the root node) has i + 1 children, where i is the number of elements inside the node
3. the tree is sorted
    1. all elements in the right/left sub-tree are bigger/smaller
    2. the elements within a node are sorted as well
4. any path from the root to any leaf has the same length

B-trees and databases

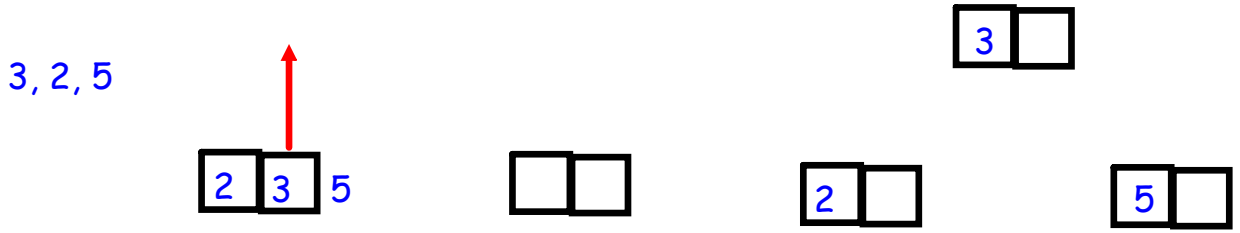
In databases B-trees are used to implement indices to quickly search inside databases.

The nodes of B-tree represent in a database the "pages". Pages are "storage areas" on the hard disk.

The access to these "storage areas" is much slower than the access to memory.

The index / tree structure is kept in fast memory in ordert o quickly navigate to the page in order to search or to insert values into this page. Only then the time consuming access on hard disk happens.

Rules for inserting elements into a B-tree



1. Insert element into node (mind ascending order)
2. Évaluate whether the node still has max. 2k element

    if yes: done

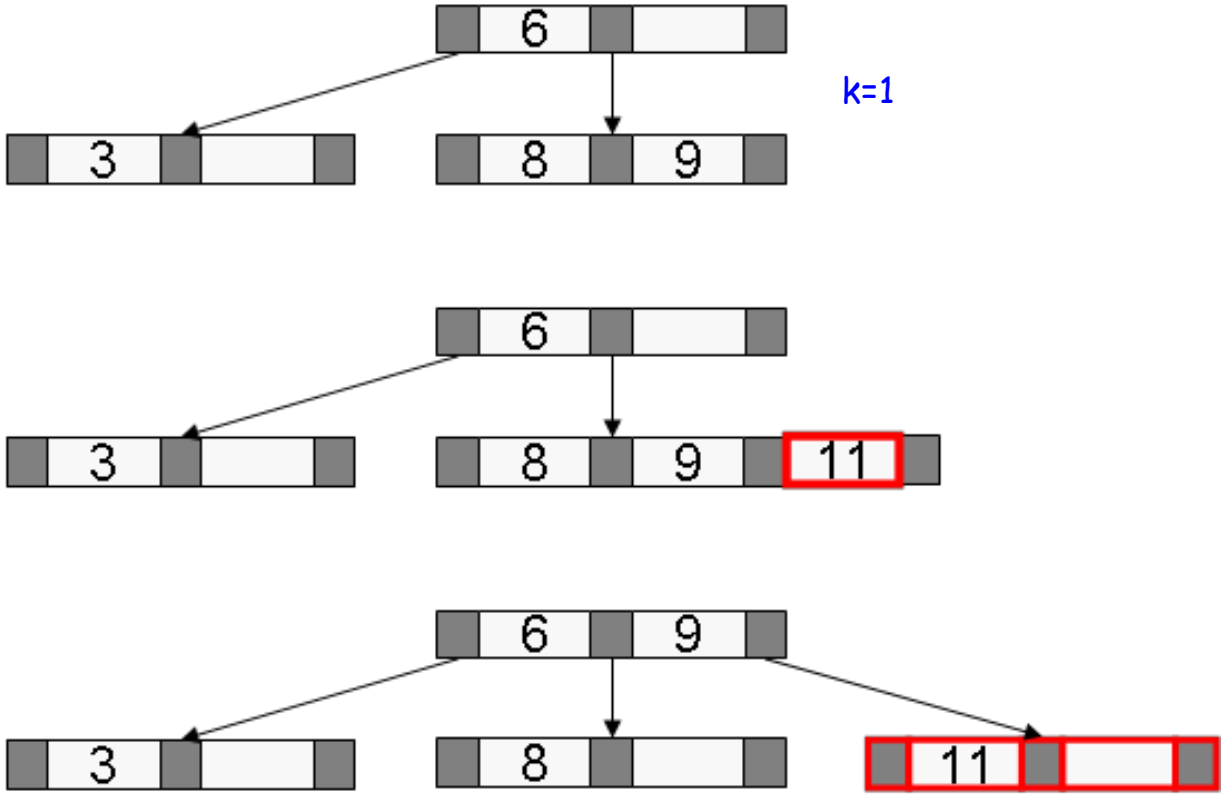
    if not: overflow treament

Overflow treatment

Element in the middle goes up to parent node (might happen recursively)

Element to the left of shifted element remains.

Element to the right is shifted to a new node (split)



Exercise:

tree: k=1

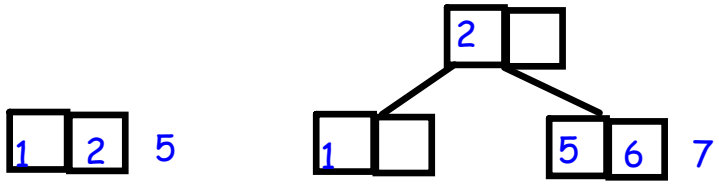
Insert: 1, 5, 2, 6, 7, 8, 3, 4

Overflow treatment

Element in the middle (9) goes up to parent node (might happen recursively)

Element to the left of shifted element (8) remains.

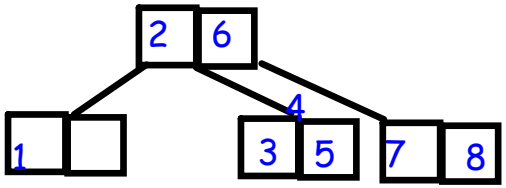
Element to the right (11) is shifted to a new node (split)

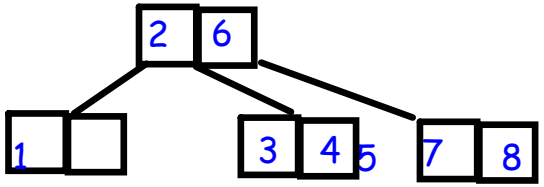


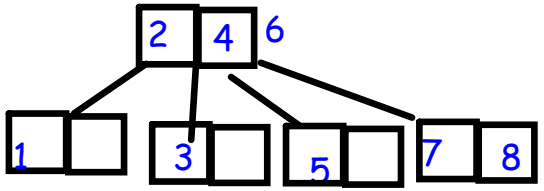
Exercise:

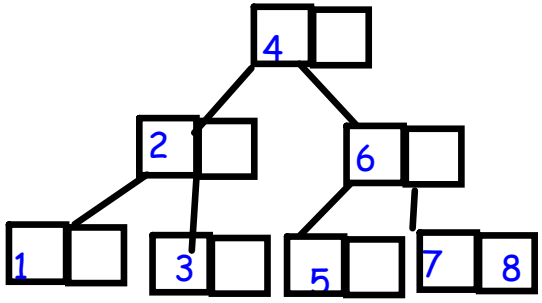
tree: k=1

Insert: 1, 5, 2, 6, 7, 8, 3, 4









Overflow treatment

Element in the middle (9) goes up to parent node (might happen r

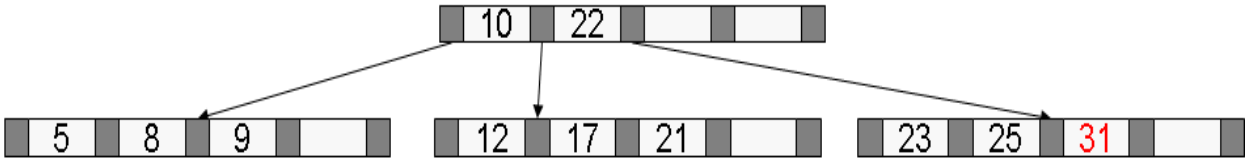
Element to the left of shifted element (8) remains.

Element to the right (11) is shifted to a new node (split)

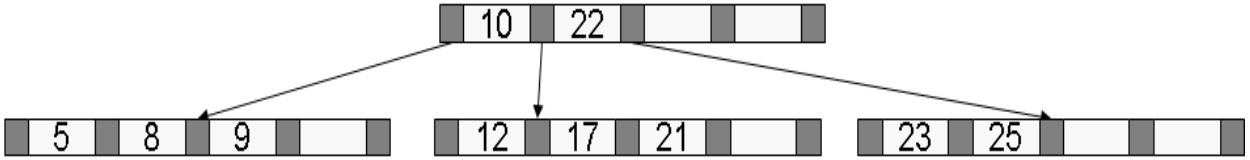
Delete elements from B-trees

Possible cases

- element to be deleted is in a leaf: delete element and handle underflow
- element to be deleted is in a node: delete element, replace it by the next one and underflow



k=2

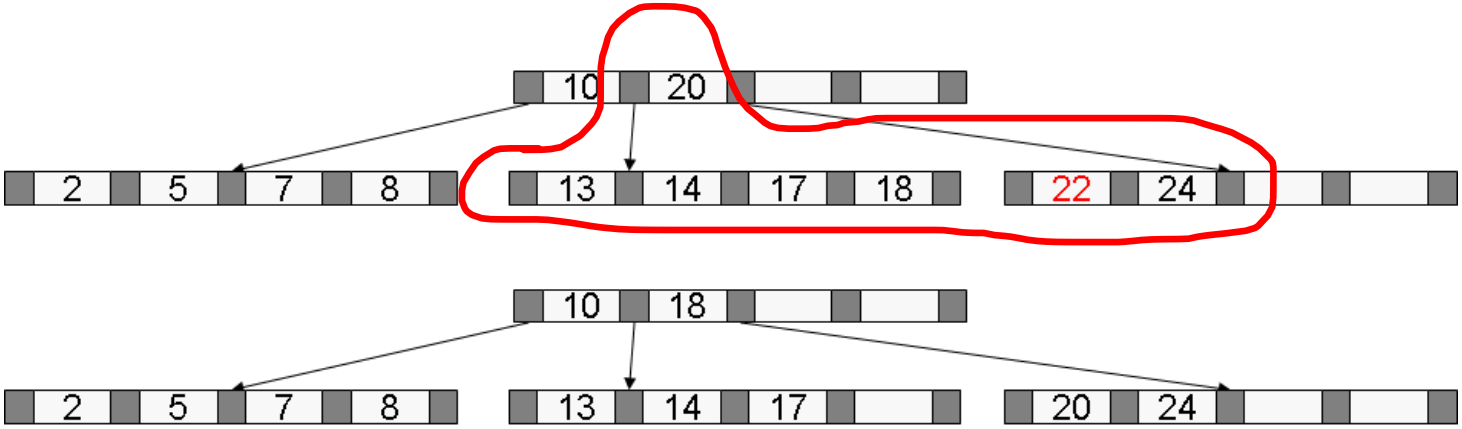


no underflow treatment needed, there still remain k elements in the node/leaf

Underflow treatment:

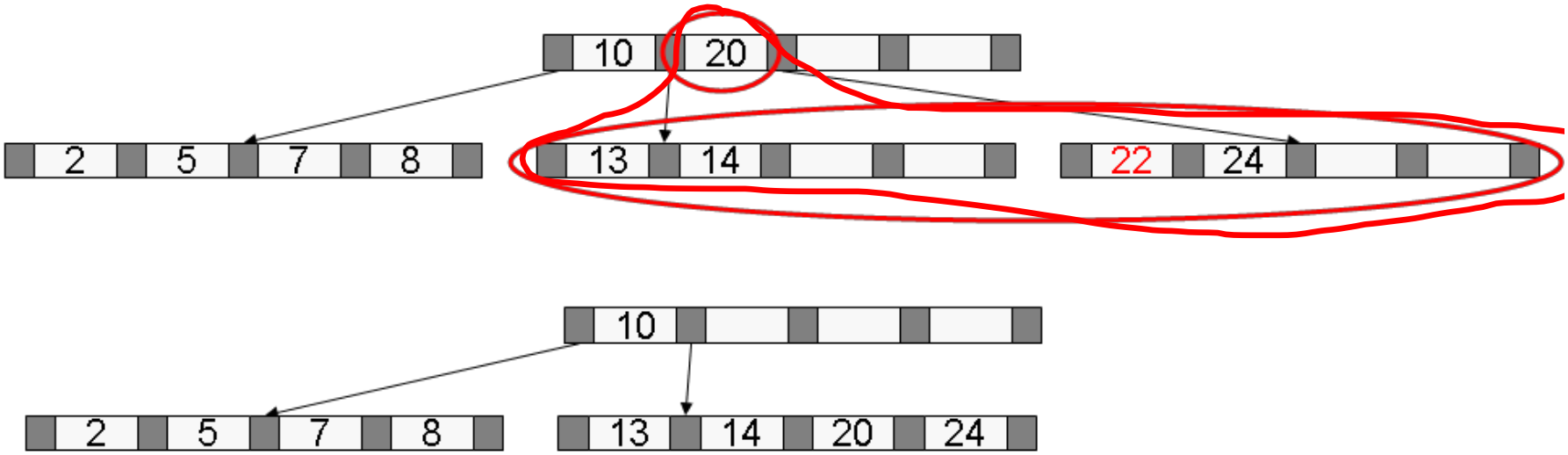
1. case: there are not sufficient elements in the node, but we can re-arrange the elements with the elements of the sibling node and the common parent element.

Underflow treatment by reorganization: make union of all remaining element including common parent element (here 13, 14, 17, 18, 20 and 24) and make the middle element (or element next to the middle here 18) to become the new parent element. Smaller elements go to the left node. Bigger elements go to the right node.



18

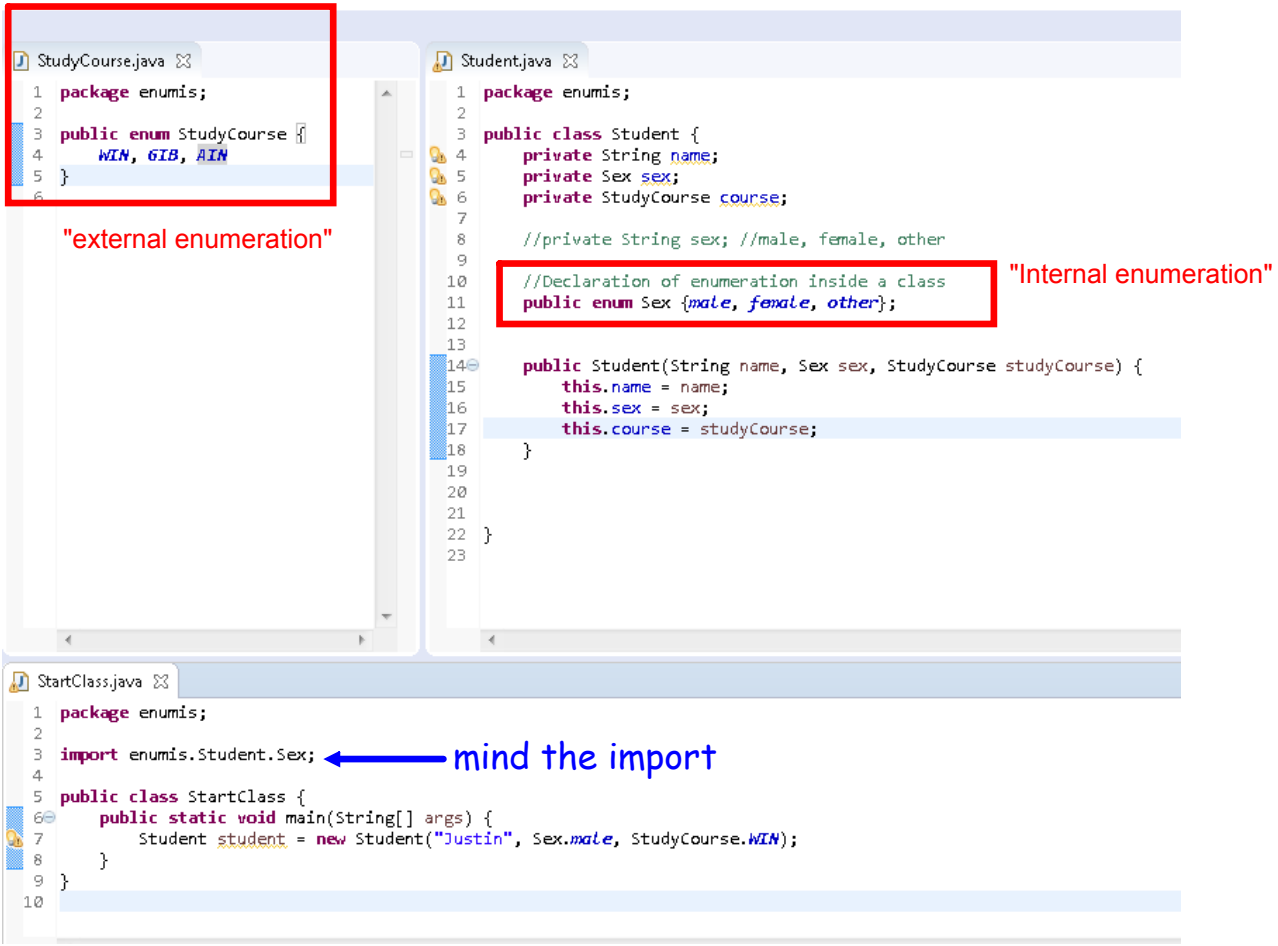
13 14 17 20 24



: If the node where the element was deleted, the sibling node (neigbour) and the common parent element do not have enough element for a reorganization, the two neigbour nodes (pages) collapse to one node that includes the common parent element (20).

This however, might cause a recursive underflow treatment.

Enumerations



Month.java

```
1  package enumis;
2
3  public enum Month {
4      Jan(31, "January"), Feb(28, "February"), Mar(31, "March");
5
6      private int numDays;
7      private String fullName;
8
9      private Month(int numberDays, String fName) {
10         this.numDays = numberDays;
11         this.fullName = fName;
12     }
13
14     public int getNumDays() {
15         return numDays;
16     }
17
18     public void setNumDays(int numDays) {
19         this.numDays = numDays;
20     }
21
22     public String getFullName() {
23         return fullName;
24     }
25
26     public void setFullName(String fullName) {
27         this.fullName = fullName;
28     }
```

mind the semicolon

an enumeration can have attributes, methods and constructors

StartClass2.java

```
1  package enumis;
2
3  public class StartClass2 {
4      public static void main(String[] args) {
5          for(Month month : Month.values()) {
6              System.out.println("The Month " + month.getFullName() + " has " + r
7          }
8      }
9  }
```

how to iterate over the enumeration values.

Month.java

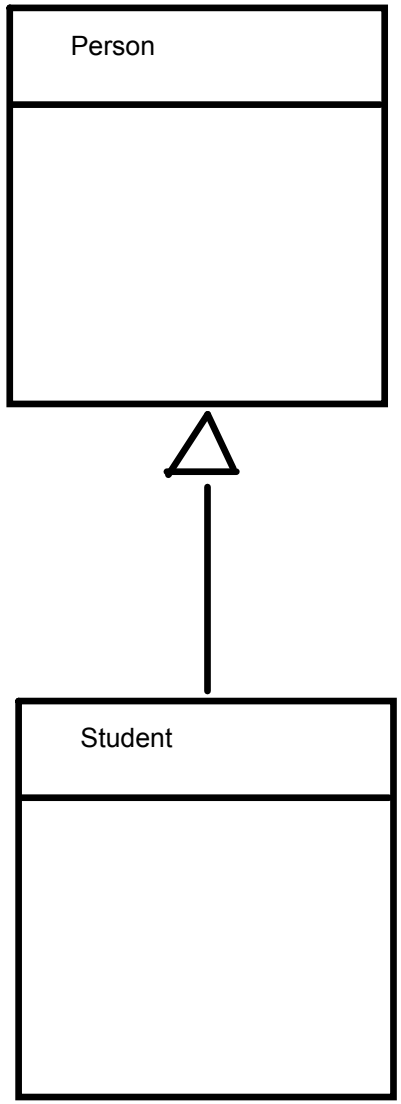
```
1  package enumis;
2
3  public enum Month {
4      Jan(31, "January") {
5          int getTemperature(){
6              return -8;
7          }
8      },
9
10     Feb(28, "February") {
11         int getTemperature() {
12             return 2;
13         }
14     },
15
16     Mar(31, "March") {
17
18         int getTemperature() {
19             return 5;
20         }
21     };
22
23     private int numDays;
24     private String fullName;
25
26     abstract int getTemperature();
27
28     private Month(int numberDays, String f
29         this.numDays = numberDays;
30         this.fullName = fName;
31     }
32
```

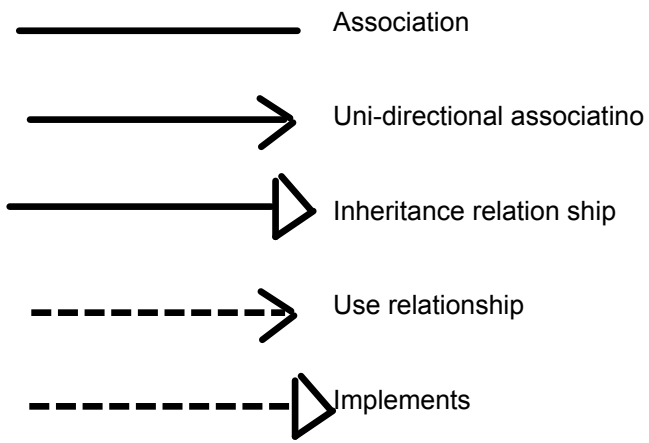
we even can add methods to each enum-values.

If we want to enforce that these methods are actually implemented, an abstract method has to be defined. Then it is possbile to iterate over all values and access the respective method.

```
package enumis;

public class StartClass2 {
    public static void main(String[] args) {
        for(Month month : Month.values()) {
            System.out.println("The Month " + month.getFullName() +
                    " has " + month.getNumDays() + " days." +
                    " and has the following temperature " + month.getTemperature());
        }
    }
}
```


Person
Student


Association
Uni-directional associatino
Inheritance relation ship
Use relationship
Implements