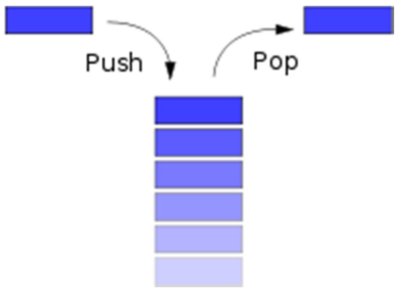
Stack & Map

1 Basic mechanism of a stack



(image source: Wikipedia)

A stack uses two operations: push and pop.

“**Push**” puts an item on the stack.
“**Pop**” takes the last item away from the stack.

The principle is therefore **FILO** (= first in, last out).

Compare this to a queue (“Warteschlange”) where the principle is FIFO (=first in, first out).

2 When do we use a stack?

There are different scenarios where using a stack is useful:

- ➔ **Navigation**: if you want to navigate through a set of positions, the stack will help you memorise your last position. You can navigate back. So we can use a stack in an adventure game, for example. (we will see such an example later)
- ➔ **User guiding**: You might use a stack to keep track of all the windows / forms a user has opened in a GUI. Therefore you will show the last window first.

3 Using a stack in Java

Java implements its own Stack class, which is part of the collections.

4 Maps

Another interesting data structure is the map. Here we have a **key-value pair** which is added to a data structure. So a key is mapped to a value.
Java uses a Map Interface, which is then implemented in various collection classes.

A typical example is the **HashMap** class:

5 Exercises with HashMap

a) Implement a **contacts application**, with the name as key and the telephone number as value.

   In your class you should have these methods:

   ```
   public void enterNumber(String name, String number)
   ```

   and

   ```
   public String lookupNumber(String name)
   ```

   The methods should use the **put** and **get** methods of the HashMap class to implement their functionality.
   Print out all the numbers in your telephone book.

b) What happens if I add two identical names to my contacts?