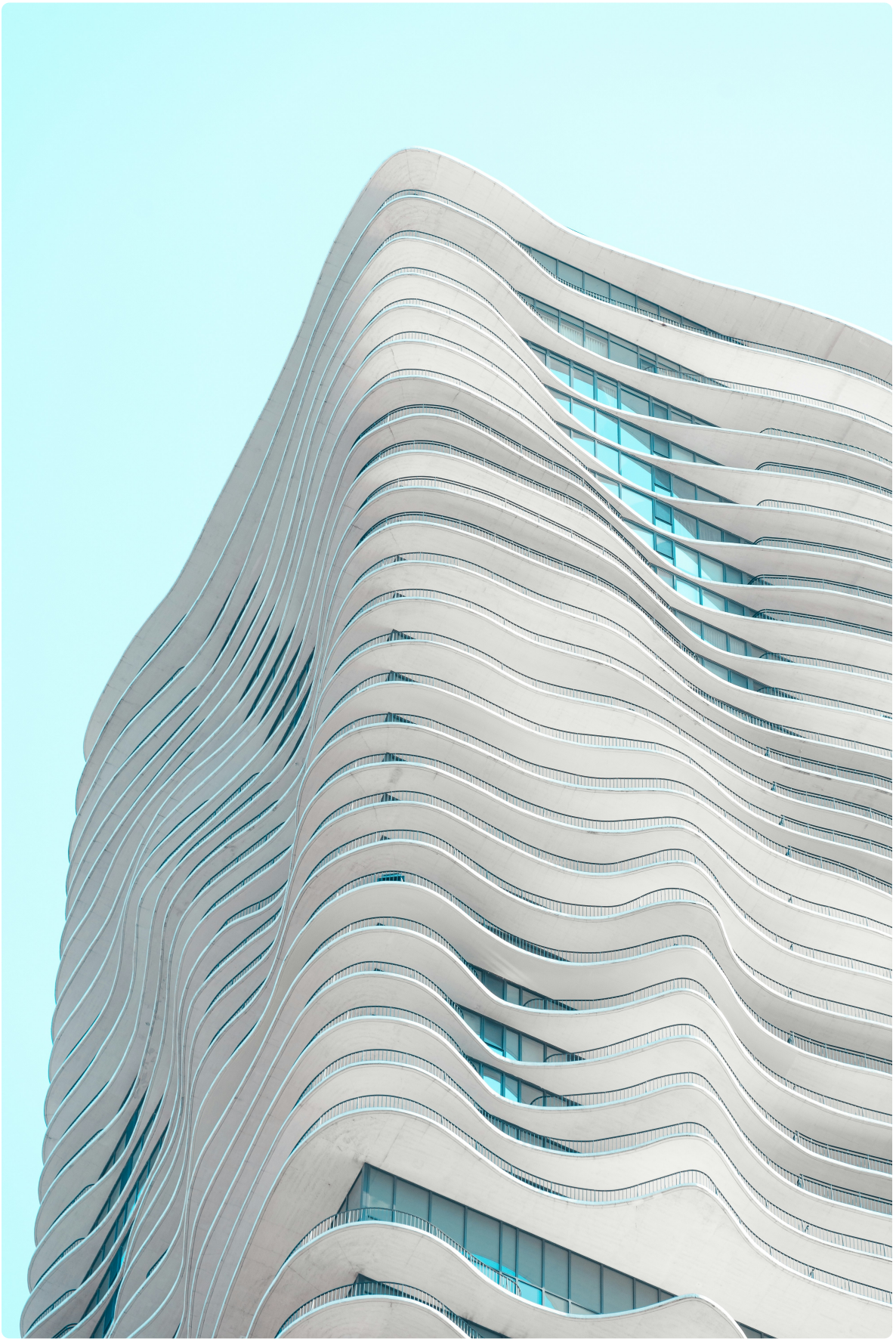
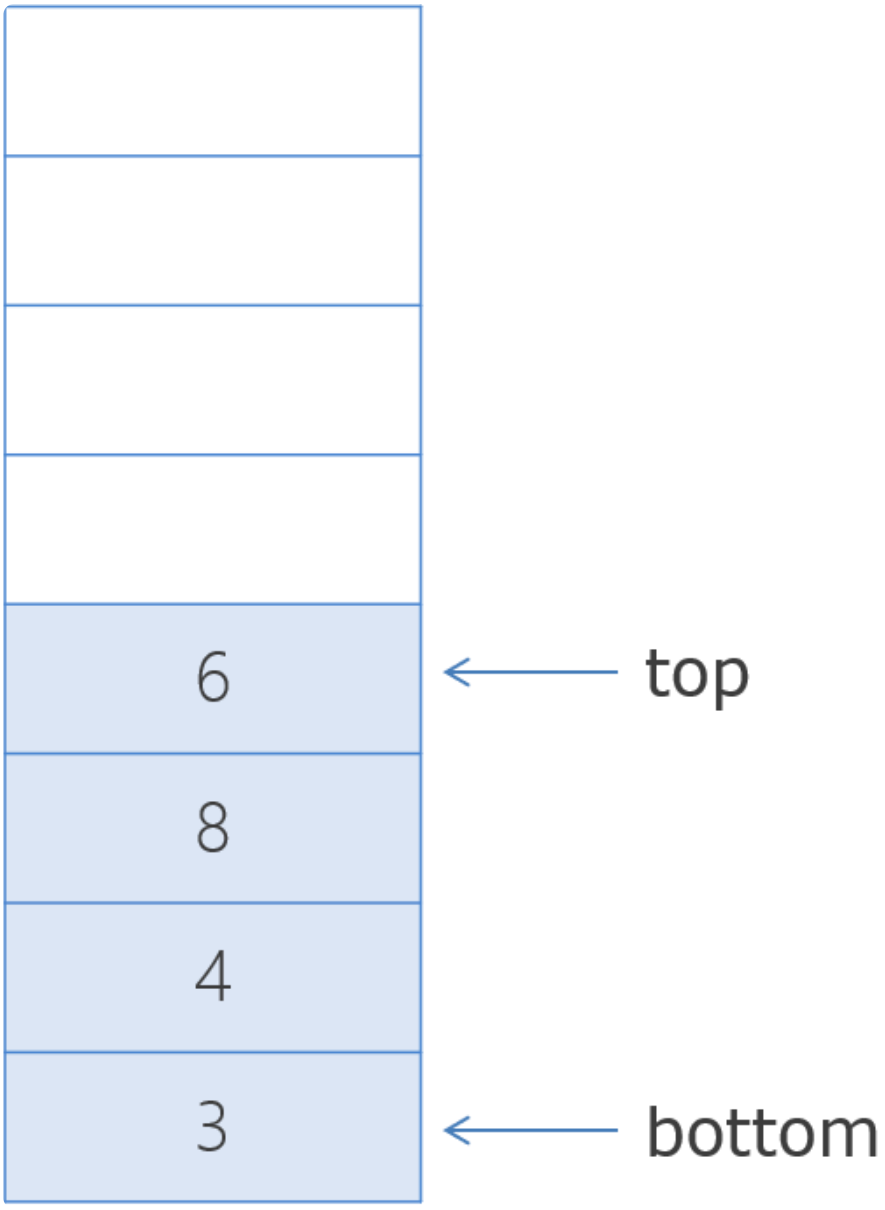


# 栈 Stacks

为了模拟各种任务情景，数据的储存介质也需要根据不同的逻辑来设计。而栈的推出则能够有效满足许多判断类型的任务。

## 简介

堆栈是一种特殊的列表，元素的插入和调出都在同一端，即同进同出。能用于插入与调出这一端称为顶端，而另一端则称为底部，底部不会发生任何操作。因此栈的基本逻辑是先进后出。




栈

## 插入与弹出

- push

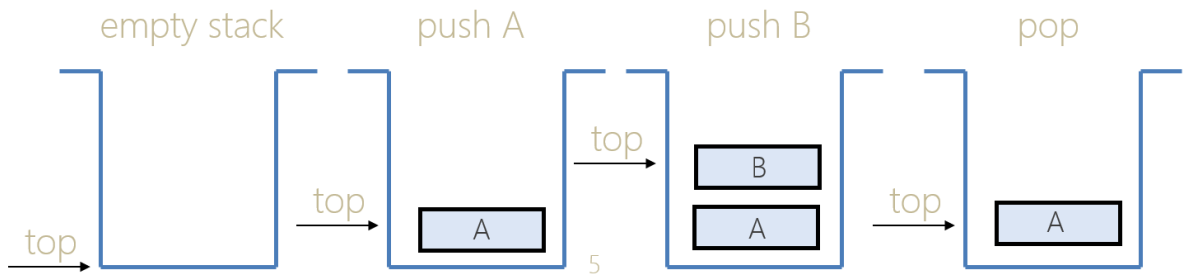
  向堆栈顶部添加元素

- pop

  删除堆栈顶部的元素

- top

  返回堆栈顶部的元素，但不删除该元素




插入与弹出

# 结构

Class: Stack

*Setters and getters are not listed.*

| Stack |
| --- |
| - values: Double[]<br>- top: int |
| + Stack(int size)<br>+ isEmpty(): boolean<br>+ isFull(): boolean<br>+ top(): Double<br>+ push(double x): Double<br>+ pop(): Double<br>+ displayStack():void |

栈的结构

# 代码实现

```java
/*Add some comments by yourself.*/
public class Stack {

    private Double[] values;
    private int top;
    public Stack(int size) {
        this.values = new Double[size];
        top = -1;
    }
    public boolean isEmpty() {
        return this.top < 0;
    }
    public boolean isFull() {
        return this.top == this.values.length - 1;
    }
    public Double top() {
        if(top < 0)
            return null;
        return this.values[top];
    }
    public Double push(double x) {
        if(isFull())
            return null;
        this.values[++top] = Double.valueOf(x);
        return top();
    }
    public Double pop() {
        if(top < 0)
            return null;
        return this.values[top --];
    }
    public void displayStack() {
        System.out.print("top -->");
        for(int i = this.top; i >= 0; i --)
            System.out.println("\t|\t " + String.format("%,
.4f", this.values[i].doubleValue()) + "\t|");
        System.out.println("\t+----------------------+");
```

```
    }
}
```