

204700

Data Structure and Programming Languages

Jakarin Chawachat

From: <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/index.htm>

solution

```
public class Triangle {
    double height = 1;
    double base = 1;
    Triangle() { }
    Triangle(double h, double b) {
        height = h;
        base = b;
    }
    void setHeight(double h) {height = h;}
    void setBase(double b) {base = b;}

    void findArea() {
        System.out.println("Area Of Triangle=" + 0.5 * height * base);
    }
}
```

```
public class Counter {
```

```
    int myCount = 0;  
    static int ourCount = 0;
```

Fields

```
    void increment(){  
        myCount++;  
        ourCount++;  
    }
```

Method

```
    public static void main(String[] arguments){
```

```
        Counter counter1 = new Counter();
```

```
        Counter counter2 = new Counter();
```

```
        counter1.increment();
```

```
        counter1.increment();
```

```
        counter2.increment();
```

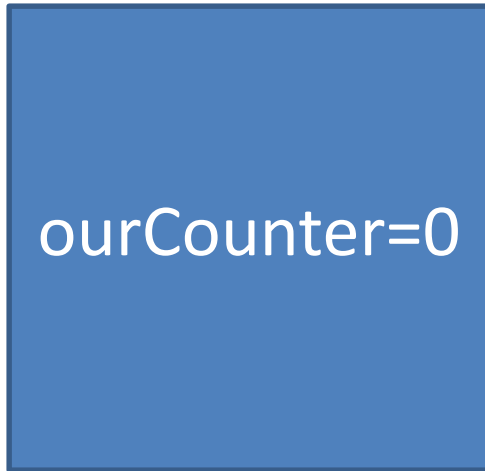
```
        System.out.println("Counter 1:" + counter1.myCount + "  
" + counter1.ourCount);
```

```
        System.out.println("Counter 2:" + counter2.myCount + "  
" + counter2.ourCount);
```

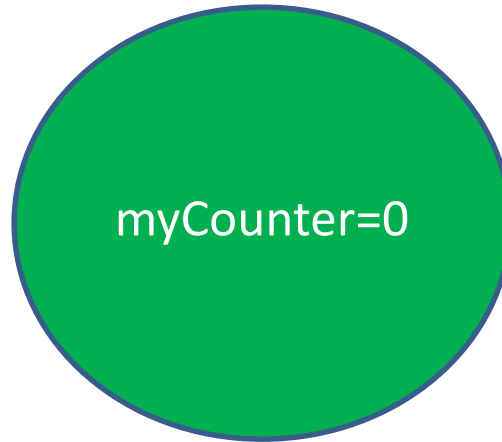
```
    }
```

```
}
```

Class Counter

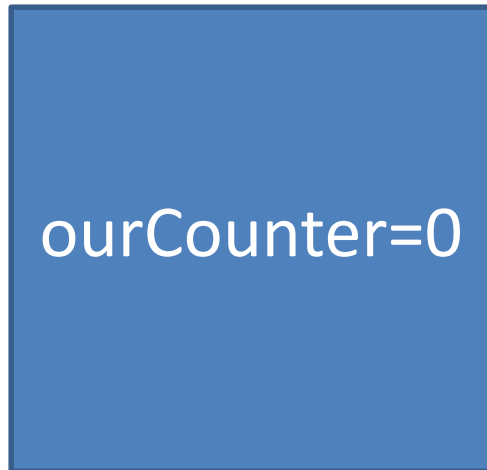


Object Counter

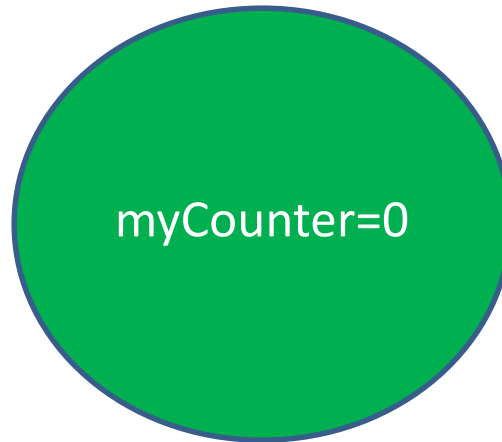


```
Counter counter1 = new Counter();
```

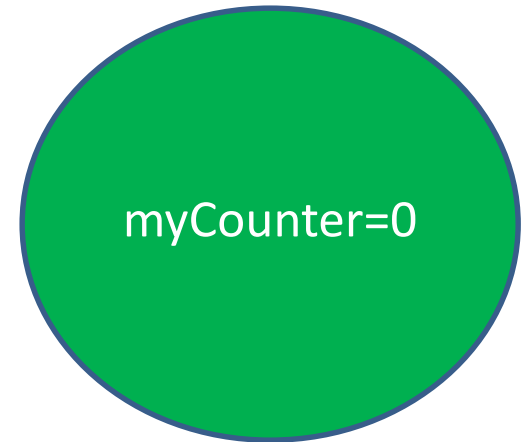
Class Counter



Object Counter

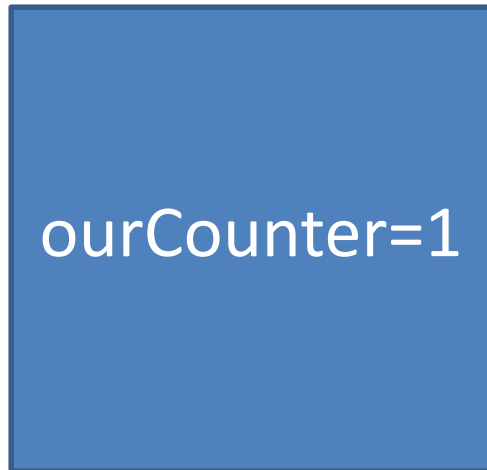


Object Counter

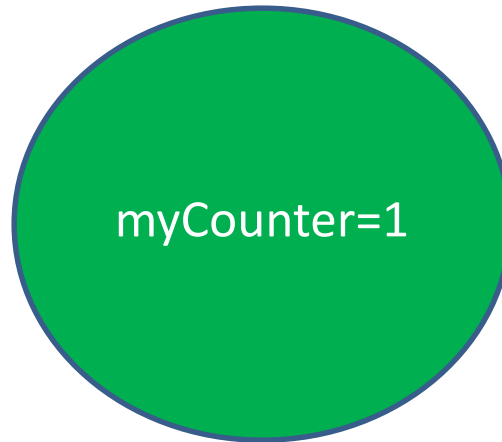


```
Counter counter1 = new Counter();  
Counter counter2 = new Counter();
```

Class Counter



Object Counter

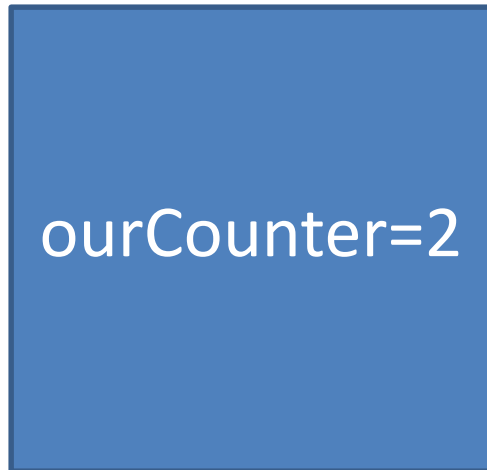


Object Counter

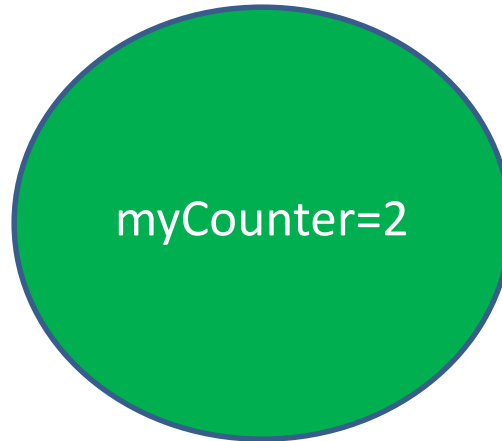


```
Counter counter1 = new Counter();  
Counter counter2 = new Counter();  
counter1.increment();
```

Class Counter



Object Counter

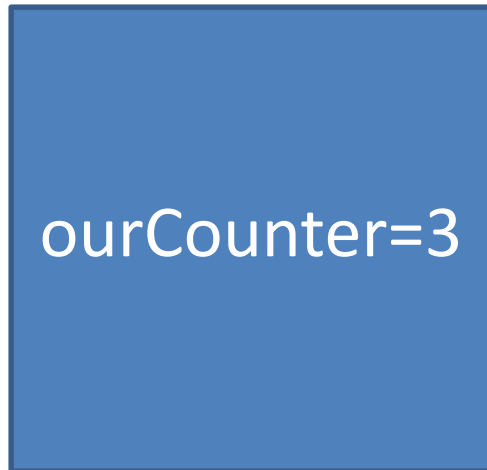


Object Counter

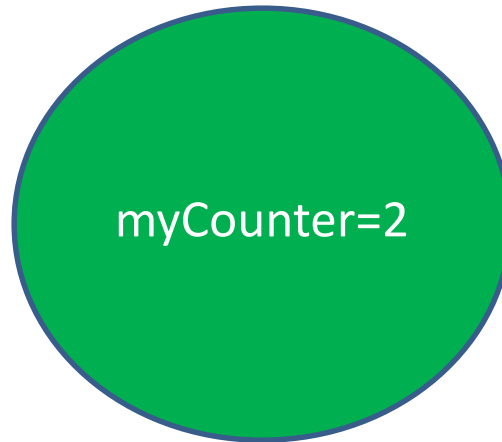


```
Counter counter1 = new Counter();  
Counter counter2 = new Counter();  
counter1.increment();  
counter1.increment();
```

Class Counter



Object Counter



Object Counter



```
Counter counter1 = new Counter();  
Counter counter2 = new Counter();  
counter1.increment();  
counter1.increment();  
counter2.increment();
```

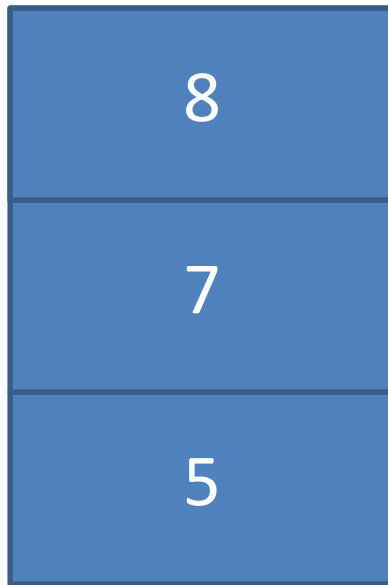

STACK

Stack

Stack เป็นโครงสร้างข้อมูลแบบเชิงเส้น ที่มีการใส่ข้อมูลเข้าและข้อมูลออกเพียงด้านเดียว

ดังนั้นข้อมูลที่เข้าไปอยู่ใน **stack** ก่อนจะออกจาก **stack** หลังข้อมูลที่เข้าไปใน **stack** ที่หลัง นั่นคือการที่เข้าทีหลังแต่ออกก่อน

Last In First Out: LIFO



Push(5)

Push(7)

Push(2)

Pop()

Push(8)

Stack Operation

- `push()` เพิ่มข้อมูลเข้า `stack`
- `pop()` นำข้อมูลออก `stack`
- `isEmpty()` ตรวจสอบว่า `stack` ว่างหรือไม่
- `getTop()` สอบถามตัวบนสุดของ `stack`

Stack: Array

ข้อจำกัดของ **array** คือจำนวนสมาชิกจำกัดตามขนาดของ **array**

ตัวอย่าง **array size 8**



ดังนั้นตอนเริ่มต้น เราจะกำหนดว่า **stack** ว่างว่าอย่างไร

top = -1

การสร้าง stack

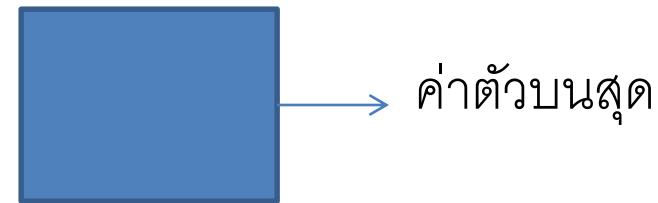
- เราจะสร้างเป็น **class** สมมติชื่อว่า **MyStack**
- ตอนเริ่มต้นเป็นการกำหนดค่าต่างๆ
 - สร้าง **array** ตามขนาดที่กำหนด และ กำหนดว่าตอนนี้ **stack** ว่าง

```
public class MyStack {  
    int max=10;  
    char arr[] = new char[max];  
    int top=-1;  
}
```

Method:getTop

- `getTop` เป็น `method` ที่สอบถามว่าตัวบนสุดของ `stack` มีค่าเป็นอะไร
- ตัวบนสุดคือตัวไหน

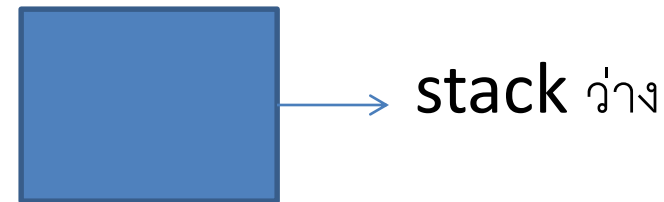
```
char getTop(){  
    char ans = '\0';  
    if(top != -1){  
        ans = arr[top];  
    }  
    return ans;  
}
```



Method:isEmpty

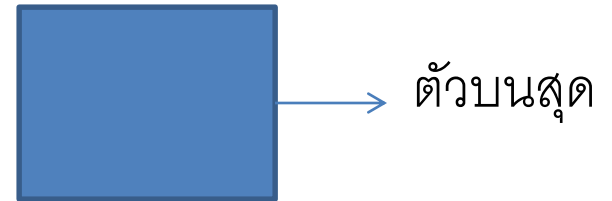
- isEmpty เป็น method ที่คืนค่าว่าขณะนี้ stack ว่างหรือไม่
- stack ว่างมีลักษณะเป็นอย่างไร

```
boolean isEmpty(){  
    if(top>=0){  
        return false;  
    } else {  
        return true;  
    }  
}
```



Method:pop

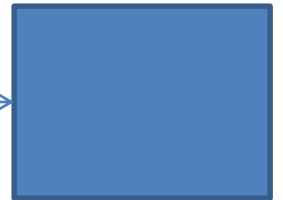
```
char pop(){
    char ans= '\0';
    if(top!=-1){
        ans=arr[top];
        top=top-1;
    }
    return ans;
}
```



Method:push

```
void push(char newItem){  
    if(top+1==max){  
        System.out.println("Stack Overflow");  
    } else {  
        top=top+1;  
        arr[top] = newItem;  
    }  
}
```

ข้อมูลใหม่



Method:main

```
public static void main(String[] args) {  
    MyStack s1 = new MyStack();  
    s1.push('5');  
    s1.push('6');  
    s1.push('9');  
    System.out.println(s1.getTop());  
    System.out.println(s1.getTop());    9  
    s1.pop();                            9  
    System.out.println(s1.getTop());    6  
    System.out.println(s1.isEmpty());   false  
}
```

Reverse Number(5_1)

- Input: 1 3 4 9 8 4
- Output: 4 8 9 4 3 1

Decimal to Binary(5_2)

- Input: 8
- Output:1 0 0 0

Checking for Balance Braces(5_3)

- Input : char[] s={'{','{','}','}','{','}'};=>{{}}{}
- Output: true