Stack Sınıfı (yığıt)

java.util
Class Stack<E>

```
java.lang.Object
  └java.util.AbstractCollection<E>
      └java.util.AbstractList<E>
          └java.util.Vector<E>
              └java.util.Stack<E>
```

Stack (yığıt) sınıfı nesnelerin LIFO (last-input-first-output) yapısıyla depolanmasını sağlayan bir veri tipidir. Bilgisayar uygulamalarında çok sık kullanılır. Üst üste konulmuş kutular gibidir. Yani gelen kutu en üste konur. Alttaki ya da aradaki bir kutuyu almak için, en üsttekinden başlayarak, istenen kutuya kadar üsttekileri sırayla almak gerekir.

Stack sınıfının, boş bir stack (yığıt) yaratan bir tek *kurucusu* vardır:

```
Stack()
```

Yığın ilk yaratıldığında boştur; ona öğeler push() metodu ile eklenir.

Stack Sınıfının Metotları

`boolean empty()`
Yığıtın boş olup olmadığını söyler. Yığıt boşsa *true* değerini verir..

`E peek()`
Yığıtın en üstündeki öğeyi değer olarak alır; ama onu yığıttan almaz, yerinde bırakır.

`E pop()`
Yığıtın en üstündeki öğeyi değer olarak alır ve onu yığıttan siler.

`E push(E item)`
Verilen nesneyi yığıtın üstüne koyar.

`int search(Object o)`
Verilen nesnenin yığıtta kaçıncı öğe olduğunu söyler. Saymaya en alttakinden 1 diye başlar.

Aşağıdaki program bir yığıt (stack) yaratıyor.

Örnek 1:

```
import java.util.*;

public class Stack01 {

	public static void main(String[] args) {
		Stack stack = new Stack();
		stack.push("Londra");
		stack.push("Moskova");
		stack.push("Ankara");
		stack.push("Paris");
		stack.push("Viyana");
		System.out.println(stack);
		System.out.println(stack.search("Ankara"));
		System.out.println(stack.peek());
		System.out.println(stack.pop());
		System.out.println(stack);
	}
}
	/*
	Çıktı:
	[Londra, Moskova, Ankara, Paris, Viyana]
	3
	Viyana
	Viyana
	[Londra, Moskova, Ankara, Paris]
	*/
```

Açıklamalar:

push() metodu ile yığıta 5 öğe giriliyor.

Search() metodu Ankara öğesinin yığıtta alttan üste doğru 3-üncü öğe olduğunu söylüyor.

peek() metodu ile yığıtın en üstündeki *Viyana* okunuyor.

pop() metodu ile yığıtın en üstündeki *Viyana* öğesini siliyor.

Aşağıdaki program bir yığıt yaratıyor.

Örnek 2:

```
import java.util.*;

public class LinkedList01 {
	private LinkedList list = new LinkedList();

	public void push(Object v) {
		list.addFirst(v);
	}

	public Object top() {
		return list.getFirst();
	}

	public Object pop() {
		return list.removeFirst();
```

```
	}

	public static void main(String[] args) {
		LinkedList01 stack = new LinkedList01();
		for (int i = 0; i < 10; i++)
			stack.push(new Integer(i));
		for (int i = 0; i < 10; i++)
			System.out.print("  " + stack.pop());
	}
}
	/*
	Çıktı:
	  9  8  7  6  5  4  3  2  1  0
	*/
```

Açıklamalar:

Önceki örnekten farklı olarak, burada liste Stack sınıfı ile değil, LinkedList sınıfı ile yaratıldığı için, *push()* ve *pop()* metotlarını ayrıca tanımlamak gerekmiştir. Tanımlanan `push()` metodu for döngüsü ile listeye *0, 1, 2, ,3, 4, 5, 6, 7, 8, 9* öğelerini giriyor. Sonra, tanımlanan `pop()` metodu for döngüsü ile listenin sonundan, sırasıyla `9 8 7 6 5 4 3 2 1 0` öğelerini siliyor.

Aşağıdaki program boş bir yığıt yaratıyor. Kullanıcı istediği verileri ekliyor ve siliyor.

Örnek 2:

```
import java.util.*;
import java.util.Stack;
import java.util.Scanner;
import java.util.Iterator;

public class LinkedList01 {

	public static void stackTest{
		Stack<Integer> integerStack = new Stack<Integer>();

		while (true) {

			System.out.println("*****************************");
			System.out.println("Lütfen seçiniz:");
			System.out.println("1 -- Yığıta bir tamsayı ekle");
			System.out.println("2 -- Yığıtın üstündeki öğeyi sil");
			System.out.println("3 -- Yığıtı yazdır");
			System.out.println("4 -- Çık");

			Scanner in = new Scanner(System.in);
			Integer input = in.nextInt();

			System.out.println("Seçiminiz: " + input);

			System.out.println("*****************************");

			if (input == 1) {
				System.out.println("Bir tamsayı giriniz: ");
				Scanner pushIn = new Scanner(System.in);
```

```
                        Integer toPush = pushIn.nextInt();
                        integerStack.push(toPush);
                        System.out.println("Girilen: \"" + toPush
                                    + "\" push() ile girildi.");
                  } else if (input == 2) {
                        System.out.println("Öğe: \"" + integerStack.pop()
                                    + "\" Yığıtın üstünden alındı");
                  } else if (input == 3) {
                        System.out.println("Yaz");
            System.out.println("Uzunluk:=" + integerStack.size());

                        for (int i = integerStack.size(); i > 0; i--) {
            System.out.println("****\t" + integerStack.elementAt(i - 1)
                                          + "*******");
                        }

                  } else if (input == 4) {
                        System.out.println("Hoşça kal!");
                        System.exit(0);
                  } else {
                        System.out.println("Geçersiz seçim");
                  }
            }
      }

      public static void main(String args[]) {
            runIntegerStackTest();
      }
}
```

Örnek 3:

Aşağıdaki program `Stack` içinde 33 sayısını arayıp bulmaktadır. Bunun için

boolean contains(Obj)

metodunu kullanmaktadır. Bu metot bir koleksiyonda aranan bir öğenin olup olmadığını bildirir. Metot `boolean` değer alır: Aranan öğe bulunursa `true`, değilse `false` değerini alır.

```
import java.util.*;
public class LinkedList01 {
      public static void main(String[] args) {
            Stack st = new Stack();
            st.add("11");
            st.add("22");
            st.add("33");
            st.add("44");
            st.add("55");
            st.add("66");
            st.add("77");
```

```
            st.add("88");
            st.add("99");
            st.add("100");
            if (st.contains("33"))
                  System.out.println("Aranan öğe bulundu!");
      }
}
```

```
/*
Çıktı:
 Aranan öğe bulundu!
*/
```