

```

package LinearneStrukture;

import java.util.EmptyStackException;

/**
 * Implementacija Steka na statički način korišćenjem niza.
 * Stek je linearna struktura koja funkcioniše po
 * LIFO (Last In First Out) principu, odnosno poslednji element
 * koji je ubačen na stek se prvi izbacuje. Elementi se
 * ubacuju i izbacuju sa istog kraja strukture.
 *
 * @author Kostandin Dimitrijević, Strukture podataka i algoritmi, FON, 2010
 */
public class StatickiStek {
    // niz u kome se čuvaju elementi steka
    private int[] niz;

    // indeks poslednjeg ubačenog elementa
    private int vrh;

    // dimenzija niza - maksimalni broj elemenata na steku
    private int dimenzija;

    /**
     * Konstruktor. Inicijalizuje sve promenljive koje se koriste u implementaciji
     * @param kapacitet : maksimalni broj elemenata u steku
     */
    public StatickiStek(int kapacitet){
        dimenzija = kapacitet;
        niz = new int[dimenzija];
        vrh = -1;
    }

    /**
     * Vraća broj elemenata na steku
     * @return
     */
    private int brojElemenata(){
        return vrh + 1;
    }

    /**
     * Proverava da li je stek pun
     * @return true ako je stek pun. U suprotnom false
     */
    public boolean punStek(){
        // prvi način
        // if(vrh == dimenzija - 1)
        //     return true;
        // else
        //     return false;

        // drugi način
        return (vrh == dimenzija - 1);
    }
}

/**

```

```

    * Proverava da li je stek prazan.
    * @return true ako je stek prazan. U suprotnom false.
    */
    public boolean prazanStek(){
        return vrh == -1;
    }

    /**
     * Stavlja novi element na stek (ubacuje novi element)
     * @param element vrednost koja se ubacuje na stek
     * @return true ako je element ubačen, false ako nema mesta (stek je pun)
     */
    public boolean push(int element){
        if (punStek())
            return false;

        // prvi način
        vrh++; // ili vrh = vrh + 1; // ili vrh += 1; // ili ++vrh;
        niz[vrh] = element;
        // drugi način
        //niz[++vrh] = element;
        return true;
    }

    /**
     * Izbacuje poslednje ubačeni element (element na vrhu steka)
     * @return izbačeni element
     * @throws EmptyStackException Ako je stek prazan, baca se izuzetak.
     */
    public int pop() throws EmptyStackException{
        if (prazanStek())
            throw new EmptyStackException();

        // prvi način
        int element = niz[vrh];
        vrh--;
        return element;
        // drugi način
        //return niz[vrh--];
    }

    /**
     * Uzima element sa vrha steka bez izbacivanja
     * @return element sa vrha steka
     * @throws EmptyStackException Ako je stek prazan, baca se izuzetak.
     */
    public int peek() throws EmptyStackException{
        if (prazanStek())
            throw new EmptyStackException();

        return niz[vrh];
    }

    public static void main(String[] args) {
        StatickiStek ss = new StatickiStek(4);
        ss.push(234);
        ss.pop();
    }
}

```