package zad2;  
  
public class Main {  
 public static void main(String[] args){  
 Stack<Integer> stack = new Stack<>();  
  
 System.*out*.println("Początkowy rozmiar stosu: " + stack.size());  
  
 //Testowanie powiększania stosu i dodawania nowych elementów  
 stack.push(45);  
 stack.push(12);  
 stack.push(60);  
 stack.push(50);  
 stack.push(30);  
 stack.push(89);  
 stack.push(6456);  
 stack.push(23410);  
  
 //Operacja top, która tylko zwraca element  
 System.*out*.println("**\n**Zwracanie elementu na szczycie:");  
 System.*out*.println(stack.top());  
 System.*out*.println(stack.top());  
 System.*out*.println(stack.top());  
  
 //Testowanie usuwania elementów i zmniejszania jego wielkości  
 System.*out*.println("**\n**Operacje usuwania ze stosu:");  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
 System.*out*.println(stack.pop());  
  
 System.*out*.println("Próba wyciągnięcia elementu z pustego stosu: ");  
 try{  
 stack.pop();  
 } catch (EmptyStackException e){  
 System.*out*.println("Stos jest pusty!");  
 }  
 }  
}

package zad2;  
  
public interface IStack<T> {  
 boolean isEmpty();  
 T pop() throws EmptyStackException;  
 void push(T element);  
 void clear();  
 int size();  
 T top() throws EmptyStackException;  
}

package zad2;  
  
public class EmptyStackException extends RuntimeException{  
}

package zad2;  
  
public class Stack<T> implements IStack<T>{  
 private int topIndex;  
 private T[] array;  
  
 public Stack(){  
 topIndex = 0;  
 array = (T[])(new Object[10]);  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return topIndex == 0;  
 }  
  
 @Override  
 public T pop() throws EmptyStackException {  
 if(!isEmpty()){  
 if(size() <= array.length \* 0.25)  
 halfSize();  
  
 return array[--topIndex];  
 }  
  
 throw new EmptyStackException();  
 }  
  
 @Override  
 public void push(T element) {  
 if(size() >= array.length \* 0.75)  
 enlarge();  
  
 array[topIndex++] = element;  
 }  
  
 @Override  
 public int size() {  
 return topIndex;  
 }  
  
 @Override  
 public T top() throws EmptyStackException {  
 if(!isEmpty())  
 return array[topIndex - 1];  
  
 throw new EmptyStackException();  
 }  
  
 @Override  
 public void clear() {  
 for(int i = 0; i < topIndex; i++) {  
 array[i] = null;  
 }  
  
 topIndex = 0;  
 }  
  
 private void enlarge(){  
 T[] newArray = (T[])(new Object[array.length \* 2]);  
  
 for(int i = 0; i < size(); i++){  
 newArray[i] = array[i];  
 }  
  
 array = newArray;  
  
 //System.out.println("Powiększanie stosu, nowa wielkość: " + array.length);  
 }  
  
 private void halfSize(){  
 T[] newArray = (T[])(new Object[array.length / 2]);  
  
 for(int i = 0; i < size(); i++){  
 newArray[i] = array[i];  
 }  
  
 array = newArray;  
  
 //System.out.println("Pomniejszanie stosu, nowa wielkość: " + array.length);  
 }  
}

Wyjście programu:

Początkowy rozmiar stosu: 0

Zwracanie elementu na szczycie:

23410

23410

23410

Operacje usuwania ze stosu:

23410

6456

89

30

50

60

12

45

Próba wyciągnięcia elementu z pustego stosu:

Stos jest pusty!