```
package structure;

import structure.exceptions.StackException;
import structure.exceptions.VectorException;

public class Stack<T> {

	public Stack() throws StackException {
		init(DEFAULT_STACK_SIZE);
	}

	public Stack(int initSize) throws StackException {
		init(initSize);
	}

	// Copy constructor doesn't create copy of elements here.
	// It means that 2 stacks will have links to the same T objects.
	public Stack(Stack<T> stack) throws StackException {
		int size = stack.size();
		init(size);
		Vector<T> temp;
		try {
			temp = new Vector<T>(size);
			for (int i = 0; i < size; ++i)
```

```
                temp.set(i, stack.pop());
            for (int i = size - 1; i >= 0; --i)
            {
                stack.push((T)temp.get(i));
                push((T)temp.get(i));
            }
        } catch (VectorException e) {
            e.printStackTrace();
        }
    }

    public void push(T elem) throws StackException {
        if (idx == pool.size() - 1)
            try {
                pool.increaseSize(pool.size()*2);
            } catch (VectorException e) {
                e.printStackTrace();
            }
        pool.set(++idx, elem);
    }

    public T pop() throws StackException {
        T t = top();
        --idx;
```

```
        return t;
}

public T top() throws StackException {
        if (idx == -1)
                throw new StackException(new String("stack is empty"));
        T t = null;
        try {
                t = (T)pool.get(idx);
        } catch (VectorException e) {
                e.printStackTrace();
        }
        return t;
}

public int size() {
        return idx + 1; // pool - zero-based array
}

@Override
public String toString() {
        String str = null;
        try {
                str = new String("Stack(" + pool.size() + "):{ ");
```

```
                for (int i = 0; i < idx; ++i)
                        str += new String("{" + pool.get(i).toString() + "}, ");
                if (idx >= 0 && idx < pool.size())
                        str += new String("{" + pool.get(idx).toString() + "}");
                str += new String(" }");
        }
        catch (VectorException e) {
                e.printStackTrace();
        }
        return str;
}

protected void init(int size) throws StackException {
        idx = -1;
        try {
                pool = new Vector<T>(size);
        } catch (VectorException e) {
                throw new StackException(new String("Unable to create vector"));
        }
}

protected Vector<T> pool; // stores T[] array
protected int idx; // index
```

```
        private final int DEFAULT_STACK_SIZE = 10;

}
```