Дисциплина: *СД и ООП* **Тема:** *Стек*
Специалност: *КН 1 курс, 3 и 4 групи* **Дата**: *17.03.2010 г.*

Задача 1: Пример за използване на родов стек **Stack<E>** - виж http://java.sun.com/javase/6/docs/api/: клас **IntegersPrinter** за генериране на случаен брой от случайни цели числа и извеждане на числата в ред обратен на получаването им:

```
import java.util.Stack;

public class IntegersPrinter {

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

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

		int n = (int)(Math.random() * 100) % 10 +1;
		int number;

		for(int i = 0; i < n; i++) {
			number = (int)(Math.random() * 100);
			integerStack.push(number   );
			System.out.println("push: " + number);
		}

		while( !integerStack.isEmpty() ) {
			number = integerStack.pop();
			System.out.println("pop: " + number);
		}

		System.out.println("Done!");
	}
}
```

Задача 2: Пример за използване на родов стек **Stack<E>**: клас **TextPrinter** за въвеждане на текст и извеждане на последните n (случайно число) реда от него:

```
import java.util.Stack;

public class TextPrinter {

	public void print() {
		Stack<String> stringStack = new Stack<String>();
		Scanner s = new Scanner(System.in);

		int n = (int)(Math.random() * 100) % 10 +1;
		System.out.println("Reads text and prints last " + n +" (or less) lines:");

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

		String line = s.nextLine();
		while(!line.equals("")) {
			stringStack.push(line);
			line = s.nextLine();
		}

		String result = "";
		for(int i = 1; i <= n; i++)
			if( !stringStack.isEmpty() ) {
				line = stringStack.pop();
				result = line + '\n' + result;
			}
			else break;
```

```
                System.out.println("Printing last lines (" + n +" or less):");
                System.out.println(result);

                System.out.println("Done!");
        }
}
```

Задача 3: Реализиран е стек с елементи от тип **char**, съгласно спецификацията:

```
import java.util.EmptyStackException;

class FullStackException extends RuntimeException {…}

public class StackOfChars {
        //Представяне на стек

        //Конструктори
        public StackOfChars() {…} //Създава празен стек
        public StackOfChars(int n) {…} //Създава празен стек с капацитет n

        //Методи
        public boolean isEmpty() {…} //Проверява дали текущия стек е празен
        public boolean isFull() {…} //Проверява дали текущия стек е пълен
        public void push(char  x) throws FullStackException  {…} //Включва x в текущия стек
        public char  pop() throws EmptyStackException {…} //Изключва елемента на върха на текущия стек
        public char  top() throws EmptyStackException {…} //Връща елемента на върха на текущия стек
        public int size() {…} //Връща броя на елементите на текущия стек
        public int capacity() {…} //Връща капацитета на текущия стек
        public void clear() {…} //Променя текущия стек на празен
}
```

Пример за използване на клас **StackOfChars**: клас **PalindromChecker** за проверка дали дума принадлежи на езика $L=\{\omega@O(\omega) \mid \omega \in \{a,b\}^*\}$:

```
public class PalindromChecker {

        public void check(String str) {
                StackOfChars stack = new StackOfChars(str.length());
                int step = 1;
                boolean flag = true;
                for(int i = 0; i < str.length(); i++) {
                        char s = str.charAt(i);
                        if(step == 1) {
                                if(s == 'a' || s == 'b') stack.push(s);
                                else if(s == '@') step = 2;
                                else{
                                        System.out.println("Error: "+s+" at "+(i+1));
                                        flag = false;
                                        break;
                                }
                        }//if
                        else { // step = 2
                                if( stack.isEmpty() ) {
                                        System.out.println("Error: missing symbols before @");
                                        flag = false;
                                        break;
                                }
                                else if(s != 'a' && s != 'b') {
                                        System.out.println("Error: "+s+" at "+(i+1));
                                        flag = false;
                                        break;
                                }
                                else {
                                        char c = stack.pop();
```

```
                                        if(c != s) {
                                                System.out.println("Error: "+s+" at "+(i+1));
                                                flag = false;
                                                break;
                                        }
                                }
                        }//else
                }//for

                if(flag && step == 1) {
                        System.out.println("Error: missing symbol @");
                        flag = false;
                }

                if(flag) {
                        if( !stack.isEmpty() )
                                System.out.println("Error: missing symbols after @");
                        else System.out.println("Yes");
                }
        }//method
}//class
```

Задача 4: Реализиран е стек с елементи от тип **int**, съгласно спецификацията:

```
import java.util.EmptyStackException;

class FullStackException extends RuntimeException {…}

public class StackOfIntegers {
        //Представяне на стек

        //Конструктори
        public StackOfIntegers() {…} //Създава празен стек
        public StackOfIntegers(int n) {…} //Създава празен стек с капацитет n

        //Методи
        public boolean isEmpty() {…} //Проверява дали текущия стек е празен
        public boolean isFull() {…} //Проверява дали текущия стек е пълен
        public void push(int  x) throws FullStackException  {…} //Включва x в текущия стек
        public int  pop() throws EmptyStackException {…} //Изключва елемента на върха на текущия стек
        public int  top() throws EmptyStackException {…} //Връща елемента на върха на текущия стек
        public int size() {…} //Връща броя на елементите на текущия стек
        public int capacity() {…} //Връща капацитета на текущия стек
        public void clear() {…} //Променя текущия стек на празен
}
```

Пример за използване на клас **StackOfIntegers**: клас **AkermanEvaluater** за пресмятане на стойността на функцията на Акерман за дадени стойности на променливите. Дефиницията на функцията на Акерман е:

$$A(m,n)=\begin{cases} n+1, \text{ако } m=0 \\ A(m-1,1), \text{ако } m\neq 0 \text{ и } n=0 \\ A(m-1,A(m,n-1)), \text{ако } m\neq 0 \text{ и } n\neq 0 \end{cases}$$

За някои стойности на m има формули, които може да се използват при тестването:

$A(0,n)=n+1$ $\quad\quad$ $A(3,n)=2^{n+3}-3$

$A(1,n)=n+2$

$A(2,n)=2n+3$ $\quad\quad$ $A(4,n)=2^{2^{\cdot^{\cdot^{2}}}}-3$, n+2 вложени степени

```
import java.util.*;

public class AkermanEvaluater {

	public int evaluate(int m,int n) {
		StackOfIntegers intStack = new StackOfIntegers();
		int initM = m, initN = n;
		intStack.push(m);
		intStack.push(n);
		 while(true) {
			try{
				n = intStack.pop();
				m = intStack.pop();
				if(m != 0) {
					if(n != 0) {
						intStack.push(m - 1);
						intStack.push(m);
						intStack.push(n - 1);
					}
					else {
						intStack.push(m - 1);
						intStack.push(1);
					}
				}
				else intStack.push(n + 1);
			}//try
			 catch(EmptyStackException e) { return n; }
			 catch(FullStackException e) {
				throw new ArithmeticException("Impossible computing the value!");
			 }
		}//while
	}//method
}//class
```

Да се промени обработката на изключението **FullStackException**, като се удвои капацитетът на стека и пресмятането започне отначало (с първоначалните стойности на параметрите initM и initN).

Задача 5: Пример за използване на клас **StackOfIntegers**: клас **PostfixExpressionEvaluater** за пресмятане на стойността на аритметичен израз в обратен полски запис с операнди – цели числа без знак и операции: +, -, *, /:

Решение: Следващата таблица съдържа аритметични изрази в *инфиксен запис* и съответното им представяне в *обратен полски запис*:

Инфиксен запис	**Обратен полски запис**
`2+(15-12)*17`	`2 15 12 -17 *+`
B	B
A+B	A B +
A+B+C	A B +C +
A+B*C	A B C *+
A*(B+C)	A B C +*
A/B*C	A B /C *

Алгоритъм за пресмятане на стойността на аритметичен израз в обратен полски запис:

1. Създаване на празен стек
2. За всеки елемент на израза се извършва следното:
   - Ако елементът е число:
     - Включване на числото в стека
   - Иначе:
     - Ако елементът е операция:

Четене и изключване на десния операнд от стека
Четене и изключване на левия операнд от стека
Прилагане на операцията върху операндите
Включване на резултата в стека

Иначе:

Има грешка в израза

3.Четене и изключване на число от стека

4.Проверка:

Ако стекът не е празен:

Има грешка в израза

Иначе:

Последното прочетено число е стойността на израза

Пример: Пресмятане на стойността на 2 15 12 -17 *+:

```
2
2,  15
2,  15,  12
2,  3                //3=15-12
2,  3,  17
2,  51               //51=3*17

53                   //53=2+51
```

```
import java.util.*;

public class PostfixExpressionEvaluater {
        //Data for all objects
        private static final String operators = "+-*/";

        //Private method
        private static boolean isOperator(char arg) {
                return operators.indexOf(arg) != -1;
        }

        //Public method
        public int evaluate(String str) throws ArithmeticException {
                StackOfIntegers stack = new StackOfIntegers(str.length());
                for(int i = 0; i < str.length(); i++) {
                        try {
                                char c = str.charAt(i);
                                int number = -1;
                                if(c >= '0' && c <= '9') {
                                        number = 0;
                                        while(c != ' ') {
                                                number = number * 10 + (c - '0');
                                                i++;
                                                c = str.charAt(i);
                                        }
                                        stack.push(number);
                                        System.out.println("Pushed constant " + number);
                                }
                                else if (isOperator(c)) {
                                        int x,y,answer=0;
                                        y = stack.pop();
                                        System.out.println("Popped the right operand " + y + " of the " + c);
                                        x = stack.pop();
                                        System.out.println("Popped the left operand " + x + " of the " + c);
                                        switch (c) {
                                                case '+':  answer = x + y;
                                                           break;
                                                case '-':  answer = x - y;
                                                           break;
```

```
                                        case '*':  answer = x * y;
                                                   break;
                                        case '/':  answer = x / y;
                                                   break;
                                }
                                stack.push(answer);
                                System.out.println("Evaluated " + c +" and pushed "+ answer);
                        }
                        else
                                throw new ArithmeticException("Invalid operator found: " + c);
                }catch(EmptyStackException e) {
                        throw new ArithmeticException("Not enough numbers in expression!");}
        }//for

        if(stack.isEmpty())
                throw new ArithmeticException("No expression provided!");
        int value = stack.pop();
        System.out.println("Popped " + value + " at end of expression.");
        if (!stack.isEmpty())
                throw new ArithmeticException("Not enough operators for all the numbers!");
        return value;
    }//method
}//class
```

Задачи за упражнение:

1.Да се реализира клас **StringReverser** за получаване на обратния низ на даден низ.

2.Да се реализира клас за проверка дали дума принадлежи на езика $L=\{a^m b^{2m+1} \mid m \geq 0\}$.

3.Да се реализира клас за проверка дали дума принадлежи на езика $L=\{a^m b^n \mid m > n \geq 0\}$.

4.Да се реализира клас **PostfixToInfixConvertor** за преобразуване на аритметичен израз в обратен полски запис с операнди - цели числа без знак и операции: +, -, *, / в аритметичен израз в инфиксен запис.

Упътване: Използвайте алгоритъма за пресмятане на стойността на аритметичен израз в обратен полски запис, както е показано на примера:

Пример: Преобразуване на 2 15 12 -17 *+:
2
2, 15
2, 15, 12
2, (15-12)
2, (15-12), 17
2, ((15-12)*17)
(2+((15-12)*17))

5.Да се реализира клас **InfixToPostfixConvertor** за преобразуване на аритметичен израз в инфиксен запис с операнди - цели числа без знак и операции: +, -, *, / в аритметичен израз в обратен полски запис.