uc3m | Universidad Carlos III de Madrid

Autor: Profesores EDA

Estructura de datos y algoritmos
Unit 2.2 – TAD Pila
Unit 2.3 – TAD Cola

Problema 1 – Inversa de una palabra.

Implementa una clase en Java que tenga un método que reciba una palabra y devuelva su inversa (por ejemplo: ROMA-> AMOR). Utiliza una pila de caracteres para implementar la solución.

Solución

```
package unit2.stack.exercises;

public interface IStackChar {


	public boolean isEmpty();
	public void push(Character elem);
	public Character pop();
	public Character top();
	public int getSize();

}

public class SNode {

	public Character elem;
	public SNode next;

	public SNode(Character e) {
		elem = e;
	}

}



public class SStackChar implements IStackChar {
	SNode peak = null;
	int size;


	public boolean isEmpty() {
		return peak == null;
	}
```

```java
	public void push(Character newElem) {
		SNode newNode = new SNode(newElem);
		newNode.next = peak;
		peak = newNode;
		size++;
	}


	public Character pop() {
		if (isEmpty()) {
			System.out.println("The stack is empty.");
			return null;
		}
		Character elem = peak.elem;
		peak = peak.next;
		size--;
		return elem;
	}

	public Character top() {
		if (isEmpty()) {
			System.out.println("The stack is empty.");
			return null;
		}
		return peak.elem;
	}

	public String toString() {
		String result = "[";
		for (SNode nodeIt = peak; nodeIt != null; nodeIt =
nodeIt.next) {
			if (result != "[") result = result + " ";
			result = result + nodeIt.elem.toString();
		}
		return result +"]";
	}

	public int getSize() {
		return size;
	}

}

public class TestInverse {

	public static void main(String args[]) {
		TestInverse obj=new TestInverse();
		String str="amor";
		System.out.println("The reverse of " + str + " is :" +
obj.reverse("AMOR"));
```

```
            //Let us to try with other words
            str="ABDZer93";
            System.out.println("The reverse of " + str + " is :" +
obj.reverse(str));

            str="inversa de la frase";
            System.out.println("The reverse of " + str + " is :" +
obj.reverse(str));

      }

      public String reverse(String sWord) {
            if (sWord==null) return null;

            //we create an object stack to save the characters
            SStackChar stackChar=new SStackChar();
            //each character is saved into a stack
            for (int i=0;i<sWord.length();i++) {
                  stackChar.push(sWord.charAt(i));
            }

            //now we can pop from the stack and store each character
into the string strInverse
            String strInverse="";
            while (!stackChar.isEmpty()) {
                  strInverse=strInverse+stackChar.pop();
            }
            return strInverse;
      }


}
```

Problema 2 – Paréntesis balanceados en expresión aritmética

Implementa en Java un programa (Class) que determine si los delimitadores (,),{,},[,] en una expresión aritmética (e.j. [(5+x)-(y+z)]) están equilibrados..

- Ejemplo de expresión correcta: ()(())\{([()])\}
- Ejemplo de expresión incorrecta: (\{[])\}

Utiliza una pila para implementar la solución. Considera las siguientes pistas:

- Si encontramos un símbolo de apertura **[**,(,**{** debemos apilarla.
- Si encontramos un símbolo de cierre **],},)** entonces consultamos el elemento que hay en la cima de pila. Si son de distinto tipo, podemos afirmar que la expresión no está balanceada. Si son del mismo tipo, debemos desapilar.
- La expresión estará balanceada si al terminar de leer la expresión la pila está vacía.

Solución:

```java
public class CheckerBalPar {
/**
* this method checks if the expression is balanced parenthesis
* sExpression can contain any type of character. We only treat the
parenthesis: (,),[,],{,}
* @param sExpression
* @return
*/
public boolean parChecker(String sExpression) {
      SStackChar stack = new SStackChar();
      boolean balanced = true;
      int index = 0;
            while (index < sExpression.length() && balanced) {
                  Character symbol = sExpression.charAt(index);
      if (symbol == '(' ||symbol == '[' ||symbol == '{') {
            stack.push(symbol);
      } else if (symbol == ')' ||symbol == ']' ||symbol == '}') {
            Character peak=stack.pop();
            if (stack.isEmpty())
                  balanced=false;
            else {
            if (symbol==')' && peak!='(') balanced=false;
            else if (symbol==']' && peak!='[') balanced=false;
            else if (symbol=='}' && peak!='{') balanced=false;
      }
} else {
//symbol is not a parenthesis, then we do not do anything.
}
index++;


if (balanced && stack.isEmpty()) return true;
else return false;
}

public static void main(String args[]) {
      //We are going to try with several expressions.
      CheckerBalPar obj=new CheckerBalPar();
      String str="()(()){([()])}";
      System.out.println("is balanced " + str + " ?: " +
obj.parChecker(str));

      str="([(3+5)*4]+10)";
      System.out.println("is balanced " + str + " ?: " +
obj.parChecker(str));

      str="([(3+5)*4]+10]";
      System.out.println("is balanced " + str + " ?: " +
obj.parChecker(str));

      str="({[])}";
      System.out.println("is balanced " + str + " ?: " +
```

```
obj.parChecker(str));
      str="()}";
      System.out.println("is balanced " + str + " ?: " +
obj.parChecker(str));
      }

}
```

Problema 3 – Josephus problem.

En la revuelta judía contra Roma, Josephus y 39 de sus compañeros se resistieron a los romanos en una cueva. Con la inminencia de la derrota, decidieron que preferirían morir antes que ser esclavos de los romanos. Decidieron organizarse en un círculo. Un hombre fue designado como el número uno, y procedieron en sentido horario matando a cada séptimo hombre (paso). Josephus fue entre otras cosas un consumado matemático; por lo que instantáneamente descubrió dónde debería sentarse para ser el último en irse. Pero cuando llegó el momento, en lugar de suicidarse, se unió al bando romano.

Implementar un método que averigüe en qué posición se debe sentar Josephus para no ser asesinado. La solución debe generalizarse para cualquier cantidad de soldados judíos y cualquier paso. En la solución se debe usar una cola de enteros (cada soldado se representa con un número del 1 al n).

En el siguiente vídeo, puedes encontrar una buena explicación de este problema.
https://www.youtube.com/watch?v=uCsD3ZGzMgE

solución.

```
package unit2.queue.exercises;

public interface IQueue {

      public boolean isEmpty();
      public void enqueue(Integer elem);
      public Integer dequeue();
      public Integer front();
      public int getSize();

}
public class SNode {

      public Integer elem;
      public SNode next;

      public SNode(Integer e) {
            elem = e;
      }

}

public class SQueue implements IQueue {
```

```java
    private SNode first;
    private SNode last;
    int size;

    public boolean isEmpty() {
        return first == null;
    }

    public void enqueue(Integer elem) {
        SNode node = new SNode(elem);
        if (isEmpty()) {
            first = node;
        } else {
            last.next = node;
        }
        last = node;
        size++;
    }

    public Integer dequeue() {
        if (isEmpty()) {
            System.out.println("Queue is empty!");
            return null;
        }
        Integer firstElem = first.elem;
        first = first.next;
        if (first == null) {
            last = null;
        }
        size--;
        return firstElem;
    }



    public Integer front() {
        if (isEmpty()) {
            System.out.println("Queue is empty!");
            return null;
        }
        return first.elem;
    }

    @Override
    public String toString() {
        String result = null;
        for (SNode nodeIt = first; nodeIt != null; nodeIt =
nodeIt.next) {
            if (result == null) {
                result = "[" + nodeIt.elem.toString() + "]";
            } else {
                result += "," + nodeIt.elem.toString();
```

```
                }
            }
            return result == null ? "empty" : result;
        }




        @Override
        public int getSize() {
            return size;
        }
    }

    public class Josephus {

        public static void main(String args[]) {
            solveJosephus(40,7);
            solveJosephus(50,4);
            solveJosephus(100,10);

        }

        public static void solveJosephus(int n, int k) {
            SQueue q = new SQueue();
            //saving the Jewish soldiers
            for (int i=1; i<=n; i++) {
                q.enqueue(i);
            }
            //kill
            while (q.getSize()>1) {
                int i=1; //count the number of soldiers
                //we have to skip k-1 soldiers
                while (i<k) {
                    //we remove it from the queue,
                    //but we save it at the end again
                    //in order to keep it
                    q.enqueue(q.dequeue());
                    i=i+1;
                }
                //we kill the k-th soldier
                q.dequeue();
            }

            int saved=q.front();
            System.out.print("For " + n + " soldiers and a step of " +
k + ", ");

            System.out.println("Josephus chose the place:"+ saved);
        }

    }
```