UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Universidad del Perú, Decana de América

Facultad de Ingeniería de Sistemas e Informática

EJERCICIOS CON PILAS Y COLAS

GRUPO DE TRABAJO 3

ASIGNATURA: Estructura de Datos II

INTEGRANTES:

- ALEJO CARNICA, Bryan Martin 17200256
- CHÁVEZ SILUPÚ, Erick Alberto 17200267
- CUELLO APAZA, Alexander Gabriel 17200269
- CORDOVA SANDOVAL, Rafael Anthony 17200268

DOCENTE: Javier Antonio Prudencio Vidal.

ESCUELA ACADÉMICO-PROFESIONAL: Ingeniería de Software.

LIMA - PERÚ

2022

LÁMINA 16:

- Considere un archivo de entrada de puntajes de exámenes en orden ABC inverso:

```
 Yeilding       Janet       87
White           Steven      84
Todd            Kim         52
Tashev          Sylvia      95
...
```

- Escriba código para imprimir los puntajes del examen en orden ABC usando una pila.

  – ¿Qué pasa si queremos seguir procesando los exámenes después de imprimirlos?

CLASE ALUMNO

```
public class Alumno {
    private String apellido;
    private String nombre;
    private int nota;

    public Alumno(String apellido, String nombre, int nota) {
        this.apellido = apellido;
        this.nombre = nombre;
        this.nota = nota;
    }
    public String getApellido() {...3 lines }

    public String getNombre() {...3 lines }

    public int getNota() {...3 lines }

    public void setApellido(String apellido) {...3 lines }

    public void setNombre(String nombre) {...3 lines }

    public void setNota(int nota) {...3 lines }
}
```

CLASE ORDERABC2

```
import java.util.*;

public class OrderABC2 {
  public static void main(String[] args) {
    Stack<Alumno> pila = new Stack<>();
    pila.push(new Alumno("Yeilding","Janet",87));
    pila.push(new Alumno("White","Steven",84));
    pila.push(new Alumno("Todd","Kim",52));
    pila.push(new Alumno("Tashev","Sylvia",95));
    pila.push(new Alumno("Lujan","Jesus",60));
    pila.push(new Alumno("Llallico","Luis",69));
    pila.push(new Alumno("Cortez","Terror",33));
    pila.push(new Alumno("Cordero","Gg",99));
      while(!pila.empty()){
         Alumno apellido=pila.peek();
         Alumno nombre=pila.peek();
         Alumno nota=pila.peek();
         System.out.println(apellido.getApellido()+" "+nombre.getNombre()+" "+nota.getNota());
         pila.pop();
      }
    }
}
```

EJECUCIÓN

```
run:
Cordero Gg 99
Cortez Terror 33
Llallico Luis 69
Lujan Jesus 60
Tashev Sylvia 95
Todd Kim 52
White Steven 84
Yeilding Janet 87
BUILD SUCCESSFUL (total time: 0 seconds)
```

LÁMINA 22:

- Modifique nuestro programa de puntuación de exámenes para que lea las puntuaciones de los exámenes en una cola e imprima la cola.

  – A continuación, filtre los exámenes en los que el alumno obtuvo una puntuación de 100.

  – Luego realice su código anterior de invertir e imprimir a los estudiantes restantes.

    - ¿Qué pasa si queremos seguir procesando los exámenes después de imprimirlos?

CLASE ALUMNO

```
public class Alumno{

    private String nombre;
    private String apellido;
    private int nota;

    public Alumno(String nombre, String apellido, int nota) {...5 lines }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public int getNota() {
        return nota;
    }

    public void setNota(int nota) {
        this.nota = nota;
    }
}
```

MÉTODOS PARA LA CLASE

```
                                                                    }
 private static void mostrarColaSinOrdenar(Queue<Alumno>cola){
      System.out.println("\033[32m===========/COLA ORIGINAL NOTAS = 100/========\033[37m");
    for(Alumno i : cola){
        if(i.getNota()==100){
             System.out.println(i.getNombre()+"\t"+i.getNota());
                        }
                    }
                                                          }
private static void mostrarColaFiltrada(Queue<Alumno>cola){
    Queue <Alumno> colaAux = new LinkedList();
    for(Alumno i : cola){
        if(i.getNota()!=100){
            colaAux.add(new Alumno(i.getNombre(),i.getApellido(),i.getNota()));
                        }
                    }
    System.out.println("\033[32m==========/NOTAS RESTANTES /===========\033[37m");
    for(Alumno i : colaAux){
        System.out.println(i.getNombre()+"\t"+i.getNota());
                        }
                                                              }
 private static Queue<Alumno> mirror(Stack<Alumno>p){
 Queue <Alumno> aux = new LinkedList(p);
 Stack <Alumno> pila = new Stack();
    while (!aux.isEmpty()){
        pila.push(aux.remove());
                    }
    while (!pila.isEmpty()){
        aux.add(pila.pop());
                       }
    return aux;                 }
                        }
```

CLASE PRINCIPAL

```
public static void main(String[] args) throws FileNotFoundException {

  Stack<Alumno> pila = new Stack();

    pila.add(new Alumno("Zabrio","Perez",98));
    pila.add(new Alumno("Yaneth","Perez",100));
    pila.add(new Alumno("Xiomara","Perez",94));
    pila.add(new Alumno("Rodolfo","Perez",56));
    pila.add(new Alumno("Francis","Perez",100));
    pila.add(new Alumno("Ernesto","Perez",71));
    pila.add(new Alumno("Dario","Perez",13));
    pila.add(new Alumno("Carlos","Perez",100));
    pila.add(new Alumno("Bertha","Perez",67));
    pila.add(new Alumno("Arnol","Perez",86));
    pila.add(new Alumno("Arturo","Perez",100));

     Queue<Alumno> cola = mirror(pila);

     System.out.println("\033[32mNOMBRE\t\tAPELLIDO\tNOTA\033[37m");
     for(Alumno i: cola){
         System.out.println(i.getNombre()+"\t\t"+i.getApellido()+"\t\t"+i.getNota());
                        }
     mostrarColaSinOrdenar(cola);
      mostrarColaFiltrada(cola);
                                                                    }
```

EJECUCION DEL CODIGO .java

```
run:
NOMBRE          APELLIDO        NOTA
Arturo          Perez           100
Arnol           Perez           86
Bertha          Perez           67
Carlos          Perez           100
Dario           Perez           13
Ernesto         Perez           71
Francis         Perez           100
Rodolfo         Perez           56
Xiomara         Perez           94
Yaneth          Perez           100
Zabrio          Perez           98
===========/COLA ORIGINAL NOTAS = 100/========
Arturo  100
Carlos  100
Francis 100
Yaneth  100
==========/NOTAS RESTANTES /============
Arnol   86
Bertha  67
Dario   13
Ernesto 71
Rodolfo 56
Xiomara 94
Zabrio  98
BUILD SUCCESSFUL (total time: 0 seconds)
```

LÁMINA 23.1:

- Escriba un método `stutter` que acepte una cola de enteros como parámetro y reemplace cada elemento de la cola con dos copias de ese elemento.

  - `frente [1, 2, 3] atrás`
    se convierte en
    `frente [1, 1, 2, 2, 3, 3] atrás`

CLASE MAIN

```
public static void main(String[] args)
{
    int nro;
    Queue<Integer> Cola = new LinkedList <>();
    Scanner dato = new Scanner(System.in);

    System.out.println("Numero de datos de la cola: ");
    nro = dato.nextInt();

    while(nro > 0 )
    {
        System.out.println("Ingrese el número a guardar en la cola: ");
        Cola.add(dato.nextInt());
        nro--;
    }
    Stutter(Cola);
}
```

CLASE STUTTER

```
public static void Stutter(Queue<Integer> Cola)
{
    Queue<Integer> aux = new LinkedList<>();
    Queue<Integer> aux2 = new LinkedList<>();

    aux.addAll(Cola);
    aux2.addAll(Cola);

    Stack<Integer> pila = new Stack();

    while(!aux.isEmpty() && !aux2.isEmpty())
    {
        pila.push(aux.remove());
        pila.push(aux2.remove());
    }

    while(!pila.empty())
    {
        aux.add(pila.pop());
    }

    while(!aux.isEmpty())
    {
        pila.push(aux.remove());
    }

    while(!pila.empty())
    {
        aux.add(pila.pop());
    }

    System.out.println(aux);
}
```

EJECUCIÓN

```
run:
Numero de datos de la cola:
3
Ingrese el número a guardar en la cola:
5
Ingrese el número a guardar en la cola:
8
Ingrese el número a guardar en la cola:
2
[5, 5, 8, 8, 2, 2]
BUILD SUCCESSFUL (total time: 9 seconds)
```

LÁMINA 23.2:

- Escriba un método `mirror` que acepte una cola de cadenas como parámetro y agregue el contenido de la cola a sí mismo en orden inverso.

  - `frente [a, b, c] atrás`
    se convierte en
    `frente [a, b, c, c, b, a] atrás`

23

Clase Mirror

```
private static void mirror(Queue<String> p) {
    Queue<String> aux = new LinkedList();
    aux.addAll(p);
    Stack<String> pila = new Stack();
    while (!aux.isEmpty()) {
        pila.push(aux.remove());
    }
    while (!pila.isEmpty()) {
        aux.add(pila.pop());
    }
    p.addAll(aux);
    System.out.println(p);
}
```

Clase Main

```
public class Cadena {

    public static void main(String[] args) {
        Queue<String> cadena = new LinkedList();
        cadena.add("a");
        cadena.add("b");
        cadena.add("c");

        mirror(cadena);


    }
```

Ejecucion

```
run-single:
[a, b, c, c, b, a]
BUILD SUCCESSFUL (total time: 2 seconds)
```

LÁMINA 37:

1 - 2 ^ 3 ^ 3 - (4 + 5 * 6) * 7

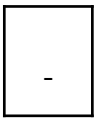
Símbolo	Off Stack Prioridad	On Stack Prioridad
+	1	1
-	1	1
*	2	2
/	2	2
^	4	3
(	5	0

Expresión Infija: 1-2^3^3-(4+5*6)*7

· Primero extraemos el 1 a la expresión postfija:

- Infija: -2^3^3-(4+5*6)*7
- Postfija: 1

· Agregamos el – a la pila:



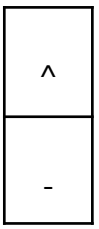
Pila

- Infija: 2^3^3-(4+5*6)*7
- Postfija: 1

· Extraemos el 2 a la expresión postfija:

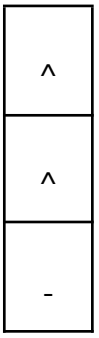
- Infija: ^3^3-(4+5*6)*7
- Postfija: 1 2

· Agregamos el ^ a la pila, como tiene mayor prioridad que el -, la pila queda de la siguiente manera:



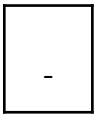
Pila

- Infija: 3^3-(4+5*6)*7
- Postfija: 1 2
· Extraemos el 3 a la expresión postfija:
- Infija: ^3-(4+5*6)*7 / Postfija: 1 2 3
· Agregamos el ^ a la pila y queda de la siguiente manera:



Pila

- Infija: 3-(4+5*6)*7
- Postfija: 1 2 3
· Extraemos el 3 a la expresión postfija:
- Infija: -(4+5*6)*7
- Postfija: 1 2 3 3
· Agregamos el - a la pila, que como tiene menor prioridad que ^, vaciamos la pila hasta llegar a un elemento de menor o igual prioridad que -, entonces la pila queda de la siguiente manera:

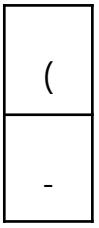


Pila

- Infija: (4+5*6)*7

- Postfija: 1 2 3 3 ^ ^ -

· Agregamos el (a la pila:



Pila

- Infija: 4+5*6)*7

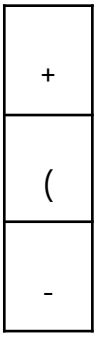
- Postfija: 1 2 3 3 ^ ^ -

· Extraemos el 4 a la expresión postfija:

- Infija: +5*6)*7

- Postfija: 1 2 3 3 ^ ^ - 4

· Agregamos el + a la pila:



Pila

- Infija: 5*6)*7

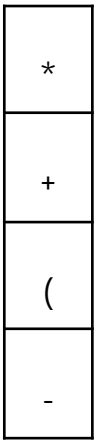
- Postfija: 1 2 3 3 ^ ^ - 4

· Extraemos el 5 a la expresión postfija:

- Infija: *6)*7

- Postfija: 1 2 3 3 ^ ^ - 4 5

· Agregamos el * a la pila:



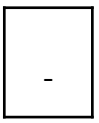
Pila

- Infija: 6)*7
- Postfija: 1 2 3 3 ^ ^ - 4 5

· Extraemos el 6 a la expresión postfija:

- Infija:)*7
- Postfija: 1 2 3 3 ^ ^ - 4 5 6

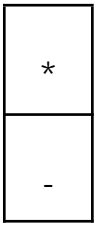
· Ahora como encontramos un), vaciamos la pila hasta encontrar un (en la pila, quedando de la siguiente manera:



Pila

- Infija: *7
- Postfija: 1 2 3 3 ^ ^ - 4 5 6 * +

· Agregamos el * a la pila:



Pila

- Infija: 7

- Postfija: 1 2 3 3 ^ ^ - 4 5 6 * +

· Finalmente extraemos el 7 a la expresión postfija y vaciamos la pila, teniendo la expresión postfija final:

- **Postfija: 1 2 3 3 ^ ^ - 4 5 6 * + 7 * -**

LÁMINA 41:

- ¿Cuál es el resultado de evaluar la siguiente expresión postfija(polaca inversa)?

1) 6 3 2 + *

A. 18

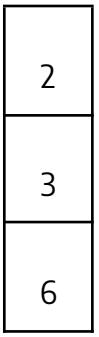
B. 36

C. 24

D. 11

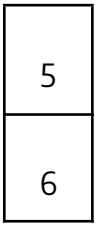
E. 30

- Primero agregamos los números 6 3 2 a la pila:



Pila

- Luego tenemos el operador +, entonces hacemos la operación 3 + 2 y agregamos el resultado a la pila:



Pila

Finalmente tenemos el operador *, haciendo la operación 6 * 5 tenemos como resultado de evaluar la expresión a la alternativa E.30

- Evaluar las sgtes. expresiones postfija y escribir su correspondiente expresion infija:

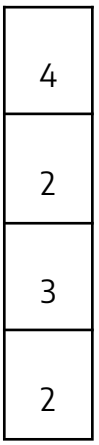
  2 3 2 4 * + * 1 2 3 4 ^ * +

  1 2 - 3 2 ^ 3 * 6 / + 2 5 ^ 1 -

41

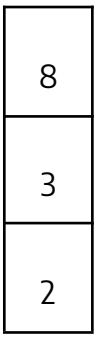
2) 2 3 2 4 * + *

- Primero agregamos los números 2 3 2 4 a la pila:



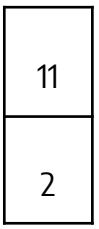
Pila

- Luego tenemos el operador *, entonces hacemos la operación 2 * 4 y agregamos el resultado a la pila:



Pila

- Ahora tenemos el operador +, hacemos la operación 3 + 8 y agregamos el resultado a la pila:

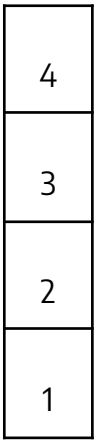


Pila

Finalmente tenemos el operador *, hacemos la operación 2 * 11 y obtenemos 22 como resultado de evaluar la expresión. Su expresión infija es: 2 * (3 + 2 * 4)

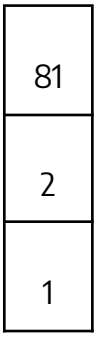
3) **1 2 3 4 ^ * +**

- Primero agregamos los números 1 2 3 4 a la pila:



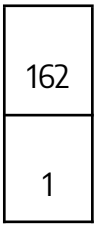
Pila

- Luego tenemos el operador ^ , entonces hacemos la operación 3 ^ 4 y agregamos el resultado a la pila:



Pila

- Ahora tenemos el operador *, hacemos la operación 2 * 81 y agregamos el resultado a la pila:

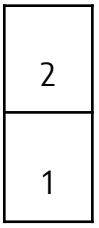


Pila

Finalmente tenemos el operador +, hacemos la operación 1 + 162 y obtenemos 163 como resultado de evaluar la expresión. Su expresión infija es: 1 + 2 * 3 ^ 4

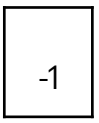
4) **1 2 – 3 2 ^ 3 * 6 / +**

- Primero agregamos los números 1 2 a la pila:



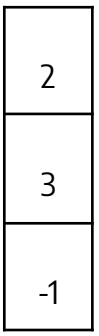
Pila

- Luego tenemos el operador -, hacemos la operación 1 – 2 y agregamos el resultado a la pila:



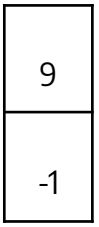
Pila

- Agregamos los números 3 2 a la pila:



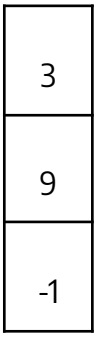
Pila

- Ahora tenemos el operador ^, hacemos la operación 3 ^ 2 y agregamos el resultado a la pila:



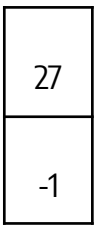
Pila

- Agregamos el número 3 a la pila:



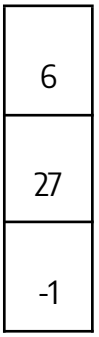
Pila

- Ahora tenemos el operador *, hacemos la operación 9 * 3 y agregamos el resultado a la pila:



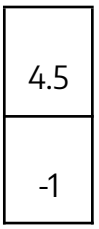
Pila

- Agregamos el número 6 a la pila:



Pila

- Ahora tenemos el operador /, hacemos la operación 27 / 6 y agregamos el resultado a la pila:

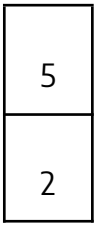


Pila

Finalmente tenemos el operador +, hacemos la operación -1 + 4.5 y obtenemos 3.5 como resultado de evaluar la expresión. Su expresión infija es: 1 - 2 + 3 ^ 2 * 3 / 6

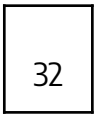
5) 2 5 ^ 1 –

- Primero agregamos los números 2 5 a la pila:



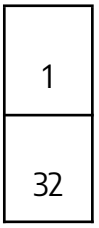
Pila

- Luego tenemos el operador ^, hacemos la operación 2 ^ 5 y agregamos el resultado a la pila:



Pila

- Agregamos el número 1 a la pila:



Pila

Finalmente tenemos el operador -, hacemos la operación 32 - 1 y obtenemos 31 como resultado de evaluar la expresión. Su expresión infija es: 2 ^ 5 - 1