Renaissance
nearpod

Nombre completo

Jesus Talat Otero Hernandez

Nombre (optional)

Jesus Talat

Únete a la lección →

Recordar Tema de Listas

Nodo Lista Simple

```java
public class Nodo {
    private int dato;
    private Nodo siguiente;

    public Nodo(int dato) {
        this.dato = dato;
        this.siguiente = null;
    }

    public int getDato() { return dato; }
    public Nodo getSiguiente() { return siguiente; }
    public void setSiguiente(Nodo siguiente) { this.siguiente = siguiente; }
}
```

Lista Simple

```java
public class ListaSimple {
    private Nodo cabeza;

    public ListaSimple() {
        cabeza = null;
    }

    public void insertarInicio(int dato) {
        Nodo nuevo = new Nodo(dato);
        nuevo.setSiguiente(cabeza);
        cabeza = nuevo;
    }

    public void mostrar() {
        Nodo actual = cabeza;
        while (actual != null) {
            System.out.println(actual.getDato());
            actual = actual.getSiguiente();
        }
    }
}
```

Pregunta 1 / 10

¿Cuál es la principal función de la clase Nodo en una lista simple en Java?

- A. Almacenar un elemento y enlazarlo con el siguiente.
- B. Eliminar elementos de la lista.
- C. Ordenar los elementos de la lista.
- D. Almacenar múltiples elementos en un solo Nodo.

Pregunta 2 / 10

¿Qué atributo debe tener la clase Nodo para referenciar al siguiente Nodo en la lista?

- A. Un atributo de tipo String llamado 'siguiente'.
- B. Un atributo de tipo entero llamado 'siguiente'.
- C. Un atributo de tipo Nodo llamado 'siguiente'. ✓
- D. Un atributo de tipo booleano llamado 'siguiente'.

Pregunta 3 / 10

¿Qué tipo de acceso se recomienda para los atributos de la clase Nodo?

- A. Sin especificador de acceso.
- B. Protegido.
- C. Privado. ✓
- D. Público.

Pregunta 4 / 10

¿Cómo se puede crear un nuevo Nodo en Java?

- A. Declarando un Nodo sin inicializarlo.
- B. No se puede crear un Nodo en Java.
- C. Usando el operador 'new' para crear una instancia de Nodo. ✓
- D. Usando un método estático para crear un Nodo.

Pregunta 5 / 10

¿Qué método se podría implementar en la clase Nodo para obtener el valor almacenado?

- A. Un método llamado 'eliminarValor'.
- B. Un método llamado 'setValor'.
- C. Un método llamado 'getValor'. ✓
- D. Un método llamado 'mostrarValor'.

Pregunta 6 / 10

¿Qué se necesita para enlazar un nuevo Nodo al final de una lista simple?

- A. Agregar el nuevo Nodo directamente al inicio.
- B. No se necesita hacer nada, el nuevo Nodo se agrega automáticamente.
- C. Recorrer la lista hasta el último Nodo. ✓
- D. Eliminar el último Nodo antes de agregar el nuevo.

Pregunta 7 / 10

¿Cuál es una desventaja de usar listas simples en comparación con listas dobles?

- A. No permite almacenar elementos duplicados.
- B. Es más difícil de implementar.
- C. Ocupa más memoria que una lista doble.
- D. No se puede acceder a los elementos en ambas direcciones. ✓

Pregunta 8 / 10

¿Qué se debe hacer al eliminar un Nodo de una lista simple?

- A. Eliminar todos los Nodos de la lista.
- B. Eliminar el Nodo sin ajustar las referencias.
- C. Ajustar las referencias de los Nodos adyacentes. ✓
- D. No se puede eliminar un Nodo de una lista simple.

Pregunta 9 / 10

¿Qué constructor se recomienda para la clase Nodo?

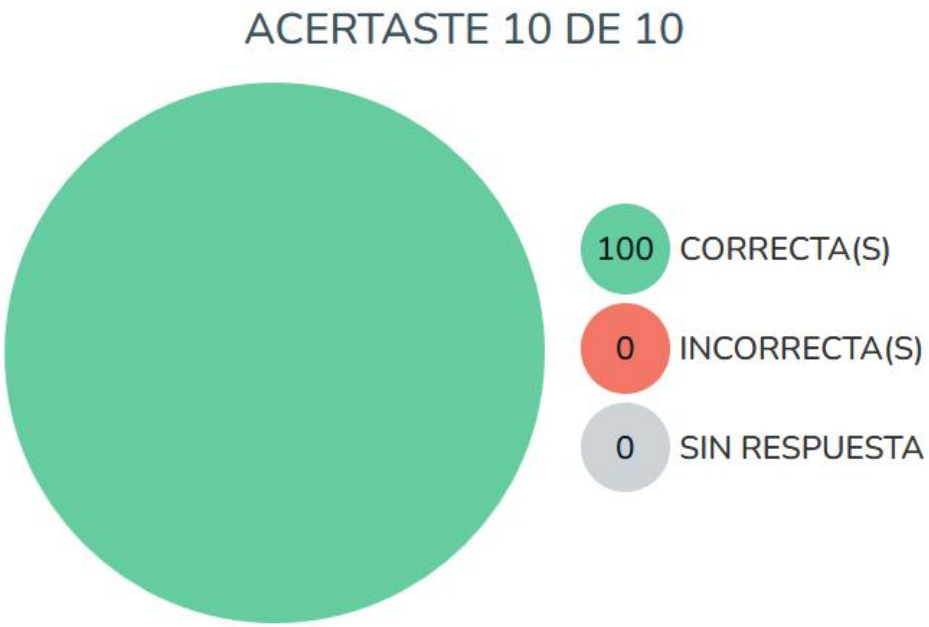
- A. Un constructor que acepte solo un valor entero.
- B. Un constructor que no acepte parámetros.
- C. Un constructor que acepte un valor y establezca 'siguiente' como null. ✓
- D. Un constructor que acepte un valor y un Nodo como parámetros.

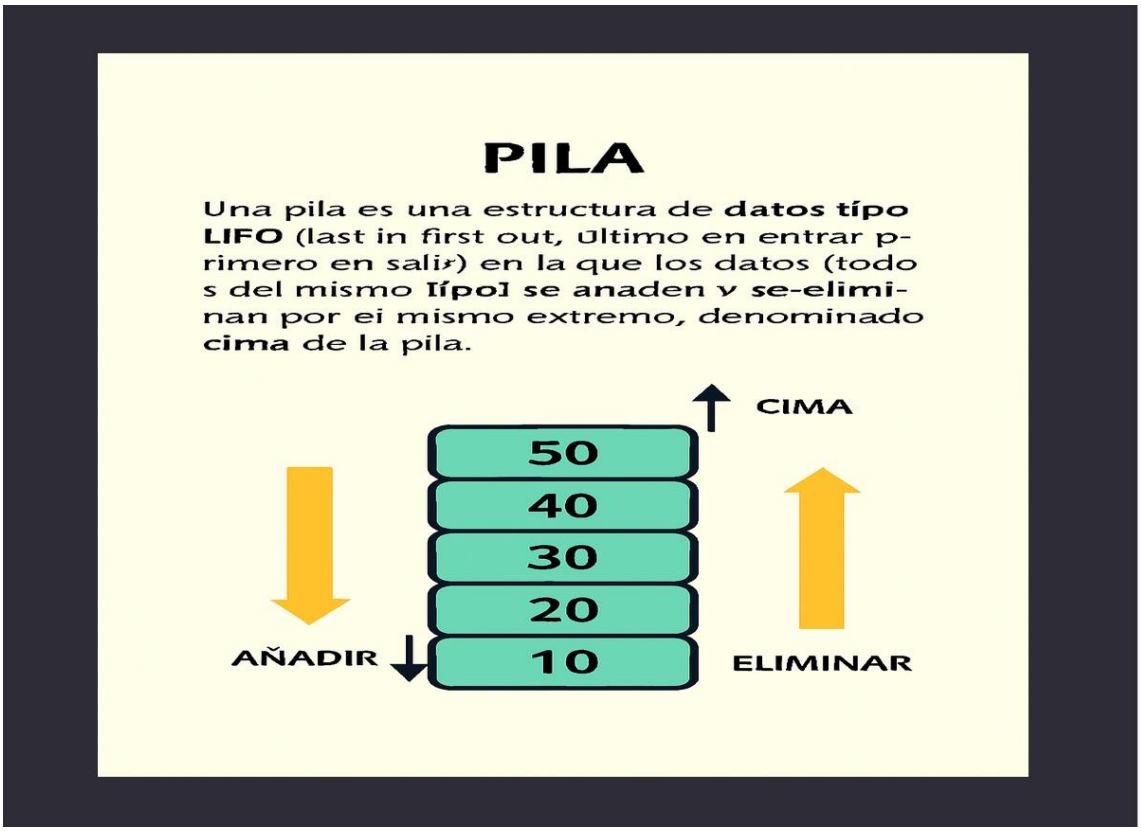
Pregunta 10 / 10

¿Qué tipo de estructura de datos es una lista simple?

- A. Estructura de datos jerárquica.
- B. Estructura de datos estática.
- C. Estructura de datos lineal. ✓
- D. Estructura de datos no lineal.

Listas y Pilas





⚙ 2. Operaciones Básicas

Operación	Descripción	Método
push(e)	Inserta un elemento en la cima.	`pila.push("A");`
pop()	Elimina el elemento superior.	`pila.pop();`
peek()	Devuelve el elemento superior sin eliminarlo.	`pila.peek();`
isEmpty()	Verifica si está vacía.	`pila.isEmpty();`

Implementación con Arreglos

```
public class PilaArreglo {
    private int[] pila;
    private int tope;
    private int capacidad;
```

```
public PilaArreglo(int tamaño) {
    capacidad = tamaño;
    pila = new int[capacidad];
    tope = -1;
}

// Insertar elemento
public void push(int dato) {
    if (tope == capacidad - 1) {
        System.out.println("Desbordamiento: la pila está llena.");
    } else {
        pila[++tope] = dato;
        System.out.println("Insertado: " + dato);
    }
}

// Eliminar elemento
public int pop() {
    if (isEmpty()) {
        System.out.println("Subdesbordamiento: pila vacía.");
        return -1;
    }
    return pila[tope--];
}
```

```
// Ver elemento superior
public int peek() {
    if (isEmpty()) return -1;
    return pila[tope];
}

// Verificar si está vacía
public boolean isEmpty() {
    return tope == -1;
}

// Mostrar contenido
public void mostrar() {
    System.out.print("Pila: ");
    for (int i = 0; i <= tope; i++) {
        System.out.print(pila[i] + " ");
    }
    System.out.println();
}
```

Implementación con ArrayList

```java
import java.util.ArrayList;

class Pila<T> {
    private ArrayList<T> elementos = new ArrayList<>();

    public void push(T valor) {
        elementos.add(valor);
    }

    public T pop() {
        if (isEmpty()) return null;
        return elementos.remove(elementos.size() - 1);
    }

    public T peek() {
        if (isEmpty()) return null;
        return elementos.get(elementos.size() - 1);
    }

    public boolean isEmpty() {
        return elementos.isEmpty();
    }
}
```

Aplicaciones En Programación

1. Evaluación de expresiones matemáticas

- Se usan pilas para evaluar expresiones en **notación postfija (RPN)** o **infija**.
- Ejemplo: convertir `3 + 4 * 2` en postfija y evaluarla usando una pila.

2. Conversión entre notaciones

- De **infija a postfija** o **prefija**, utilizando pilas para operadores y operandos.

3. Verificación de paréntesis balanceados

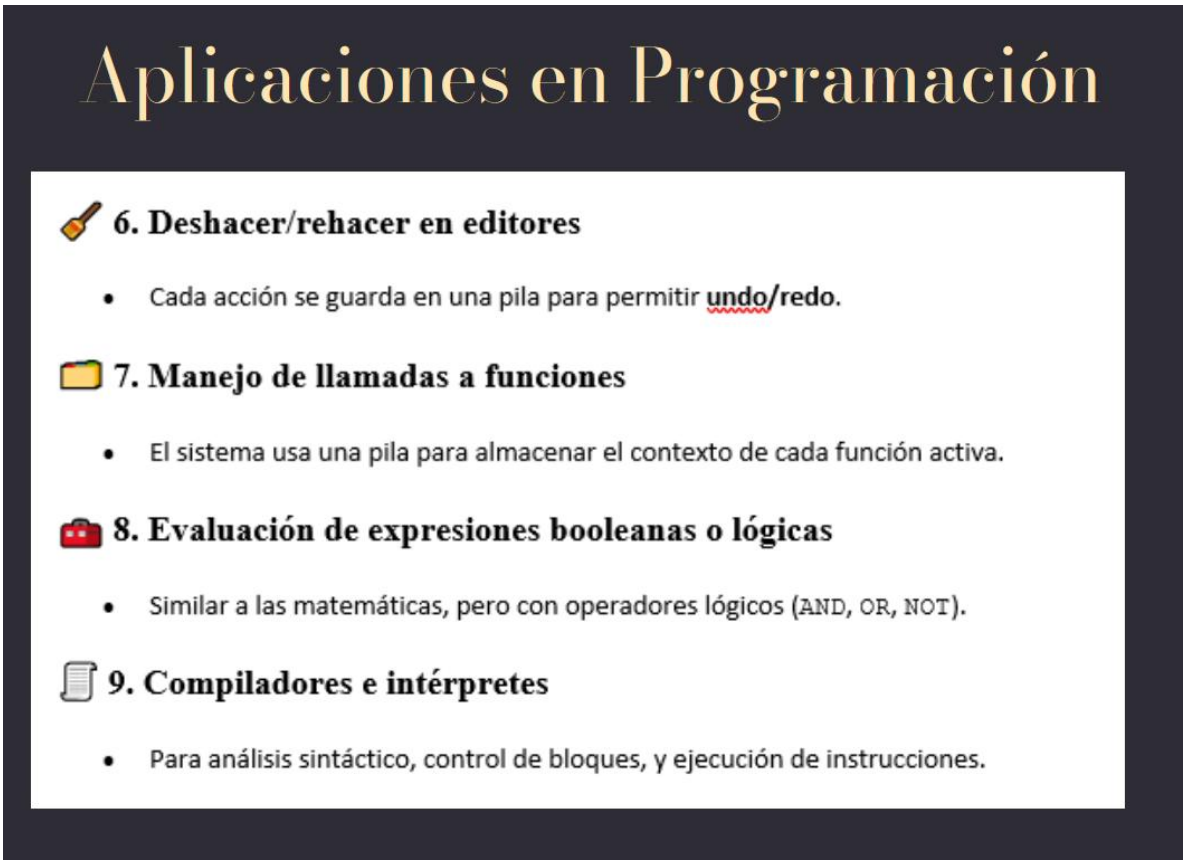
- Se apilan los símbolos de apertura y se desapilan al encontrar cierres.
- Útil en compiladores y editores de código.

4. Recursividad

- Cada llamada recursiva se apila en la **pila de ejecución**.
- Ejemplo: funciones como factorial, Fibonacci, recorrido DFS.

5. Algoritmos de búsqueda en grafos (DFS)

- El recorrido en profundidad (Depth-First Search) usa una pila para explorar nodos.



Pregunta 1 / 13

¿Cuál es la principal desventaja de usar un arreglo para implementar una pila?

A. Su complejidad

B. Que no permite duplicados.

C. Que no permite recorrer sus elementos.

D. Que tiene tamaño fijo y puede desbordarse.

Pregunta 2 / 13

Completa la condición para evitar el desbordamiento de la pila:

```
public void push(int dato) {
    if (tope == pila.length - 1) {
        System.out.println("Desbordamiento");
    } else {
        pila[++tope] = _____;
    }
}
```

- A. pila
- B. dato ✓
- C. capacidad
- D. tope

Pregunta 3 / 13

¿Qué ocurre si intentas hacer push() en una pila llena?

- A. Se elimina automáticamente el primer elemento.
- B. Lanza un error de subdesbordamiento (underflow).
- C. Lanza un error de desbordamiento (overflow). ✓
- D. Se redimensiona automáticamente el arreglo

Pregunta 4 / 13

Completa el método main para insertar y mostrar elementos:

```
public static void main(String[] args) {
    Pila p = new Pila(3);
    p.push(10);
    p.push(20);
    p.push(30);
    p._____();
}
```

- A. peek()
- B. mostrar() ✓
- C. pop()
- D. size()

Pregunta 5 / 13

¿Cuál es la función del método push() en una pila?

- A. Quitar el elemento del tope.
- B. Mostrar todos los elementos.
- C. Insertar un nuevo elemento en la cima. ✓
- D. Verificar si la pila está vacía.

Pregunta 6 / 13

Completa el código para eliminar el último elemento insertado:

```
public int pop() {
    if (isEmpty()) {
        System.out.println("Pila vacía");
        return -1;
    }
    return pila[_____--];
}
```

- A. 0
- B. pila.length
- C. tope ✓
- D. dato

Pregunta 7 / 13

¿Qué devuelve el método peek()?

- A. El primer elemento de la pila.
- B. El tamaño total de la pila.
- C. El elemento superior sin eliminarlo. ✓
- D. Todos los elementos en orden inverso.

Pregunta 8 / 13

Completa la instrucción para obtener el elemento superior sin eliminarlo:

```
public int peek() {
    if (isEmpty()) return -1;
    return pila[_____];
}
```

- A. pila.length - 1
- B. tope ✓
- C. 0
- D. capacidad

Pregunta 9 / 13

¿Qué método se utiliza para eliminar el elemento superior de la pila?

- A. pop() ✓
- B. peek()
- C. push()
- D. remove()

Pregunta 10 / 13

Completa la condición que verifica si la pila está vacía:

```
public boolean isEmpty() {
    return _____ == -1;
}
```

- A. pila.length
- B. capacidad
- C. tope ✓
- D. pila[0]

Pregunta 11 / 13

Completa la línea para declarar el arreglo y el tope de la pila:

- A. cima
- B. tope
- C. indice
- D. ultimo

Pregunta 12 / 13

Completa el constructor de la clase para inicializar la pila y el tope:

```
public Pila(int tamaño) {
    pila = new int[tamaño];
    _____ = -1;
}
```

- A. capacidad
- B. tope
- C. contador
- D. valor

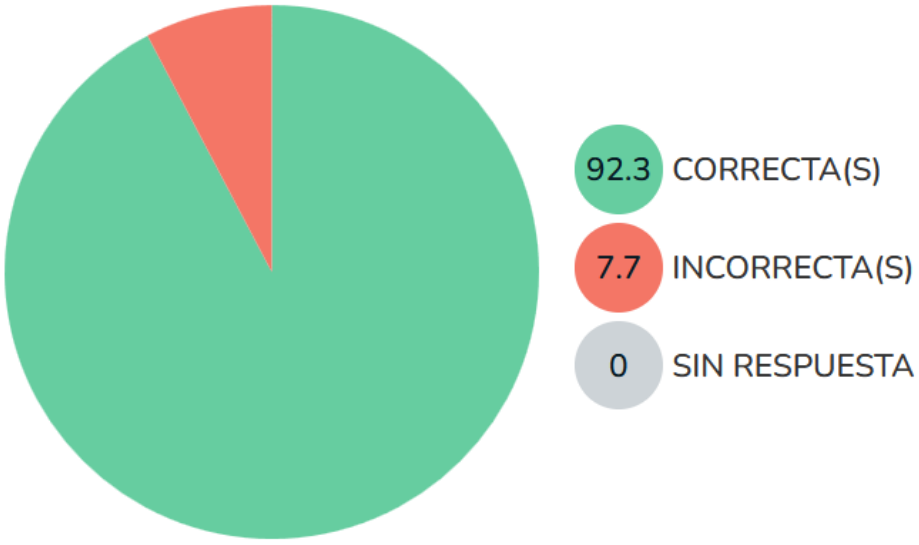
Pregunta 13 / 13

¿Qué instrucción evita errores antes de eliminar un elemento?

- A. if (isEmpty())
- B. if (pila == null)
- C. if (tope > 0)
- D. if (pila.length == 0)

Quiz de Pilas

ACERTASTE 12 DE 13




Congratulations!
1
Diana Mabel García M...
10062 Points
2
Jesus Talat Otero Hern...
9994 Points
3
Luis Angel Valencia Ja...
9860 Points