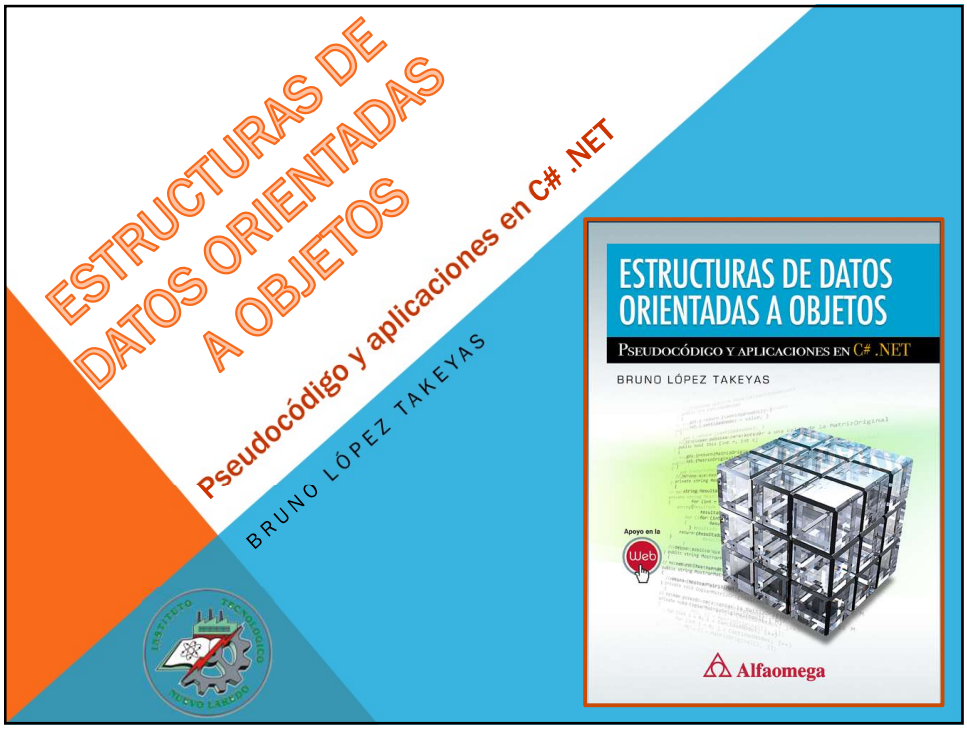
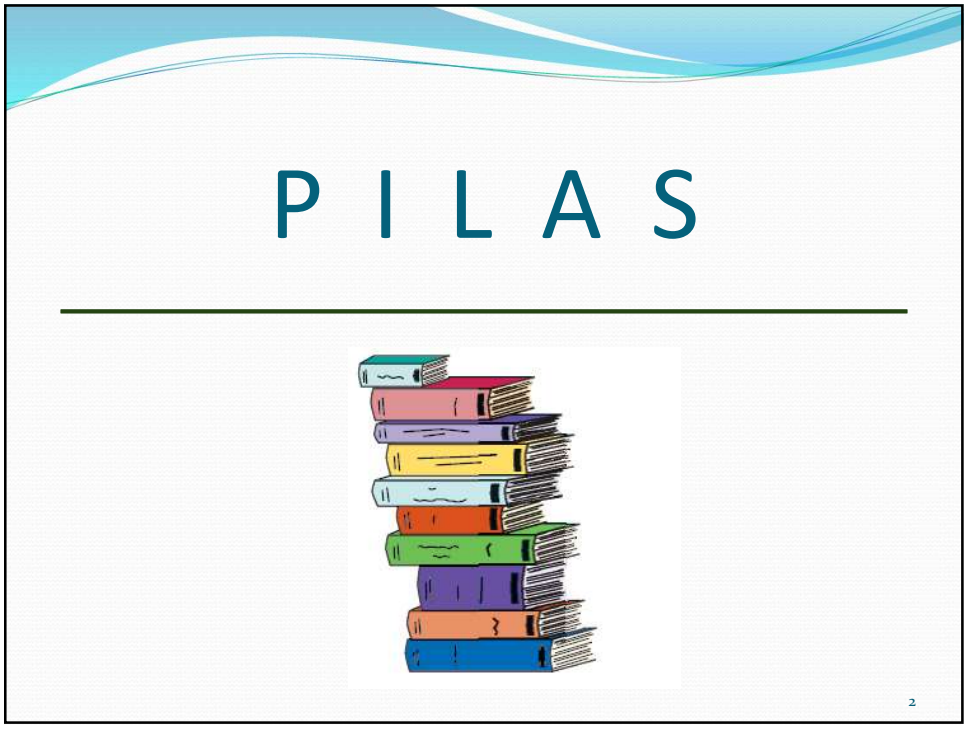

ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS
Pseudocódigo y aplicaciones en C# .NET
BRUNO LÓPEZ TAKEYAS
ESTRUCTURAS DE DATOS ORIENTADAS A OBJETOS
PSEUDOCÓDIGO Y APLICACIONES EN C# .NET
BRUNO LÓPEZ TAKEYAS
Alfaomega


P I L A S
2

Preguntas detonadoras

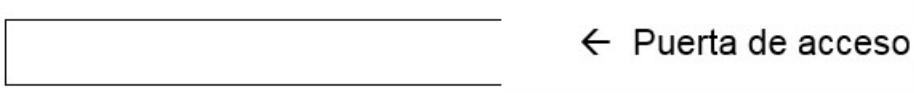
- ¿Qué es una pila?
- ¿Qué características distinguen a una pila?
- ¿Cómo se representa gráficamente una pila?
- ¿Qué operaciones se pueden realizar en una pila?
- ¿Qué situaciones de la vida cotidiana se pueden modelar con pilas?
- ¿Existen pilas estáticas?
- ¿ … y dinámicas?
- ¿Cómo se diseña un modelo orientado a objetos con pilas?

3

Pilas

Es una estructura de datos lineal que solamente tiene una puerta de acceso por la que se insertan y eliminan los datos.

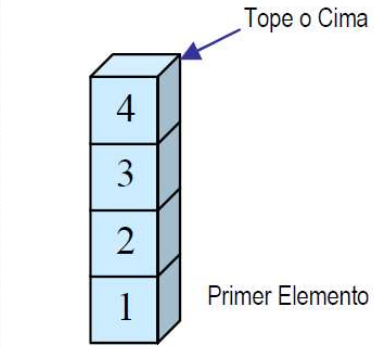
Los datos se insertan uno detrás de otro.



4

Representación gráfica de una pila

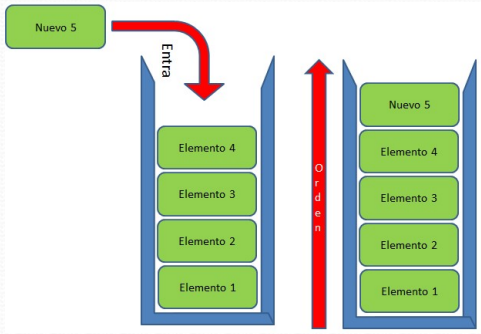
- *La pila tiene un apuntador inicial llamado Top (Tope o Cima) que señala al dato más cercano a la puerta de acceso*



5

Comportamiento de una pila

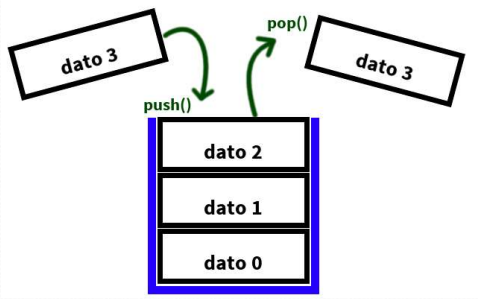
- *LIFO.- Last input, first output (último dato en entrar es el primero en salir)*
- *FILO.- First input, last output (primer dato en entrar es el último en salir)*



6

Operaciones básicas con pilas

- *Push.- Método para insertar un dato en la cima de la pila*
- *Pop.- Método para eliminar el dato ubicado en la cima de la pila*



7

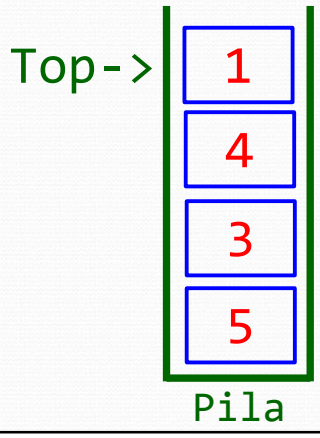
Método Push

El método Push() recibe como parámetro el dato que se desea insertar en la pila

8

Método Pop

Pop()



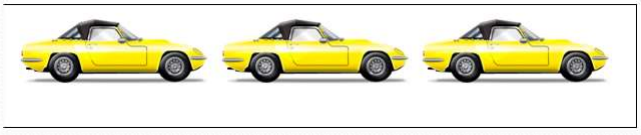
El método Pop() no recibe parámetros; simplemente elimina el dato ubicado en el Top de la pila

9

Ejemplos de pilas en la vida cotidiana



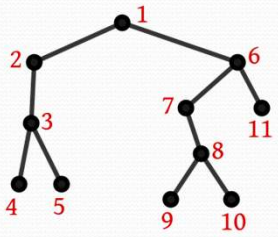


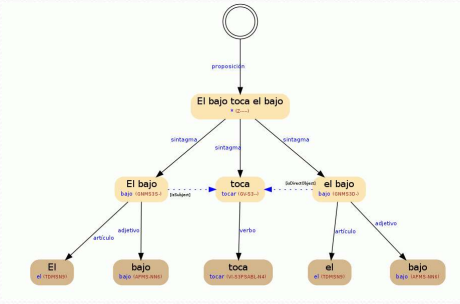



10

Algoritmos que usan pilas

Búsqueda en profundidad





Algoritmo Torres de Hanói (Complejidad $\Theta(2^n)$)

Entrada: Tres pilas de números *origen*, *auxiliar*, *destino*, con la pila *origen* ordenada

Salida: La pila *destino*

1. **si** *origen* $==\{1\}$ **entonces**
    1. **mover** el disco 1 de pila origen a la pila destino (insertarlo arriba de la pila destino)
    2. **terminar**
2. **si no**
    1. *hanoi*($\{1,...,n-1\}$,*origen*,*destino*, *auxiliar*) //mover todas las fichas menos la más grande (*n*) a la varilla auxiliar
3. **mover** disco *n* a *destino* //mover la ficha grande hasta la varilla final
4. *hanoi* (*auxiliar*, *origen*, *destino*) //mover todas las fichas restantes, 1...*n*–1, encima de la ficha grande (*n*)
5. **terminar**

Tipos de pilas

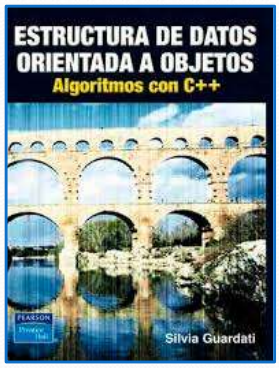
Pilas

- *Estáticas.- Basadas en arreglos*
- *Dinámicas.- Basadas en listas enlazadas*

12

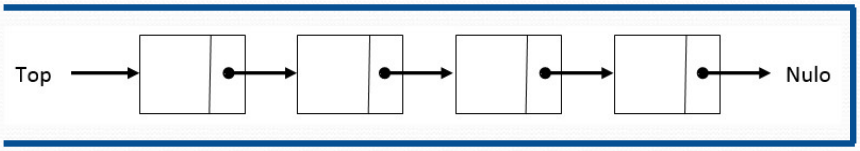
Tipos de pilas

- *Tradicionalmente se imparten las clases de pilas de manera estática (basadas en arreglos)*
  - *El libro "Estructura de datos orientada a objetos. Algoritmos con C++" de Silvia Guardati menciona que las pilas y colas pueden representarse de manera dinámica a través de listas.*



¿Cómo analizaremos las pilas?

- *En estos apuntes se analizan las pilas dinámicas (basadas en listas) y orientadas a objetos*

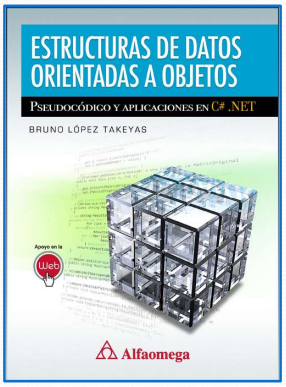


14

Referencias bibliográficas

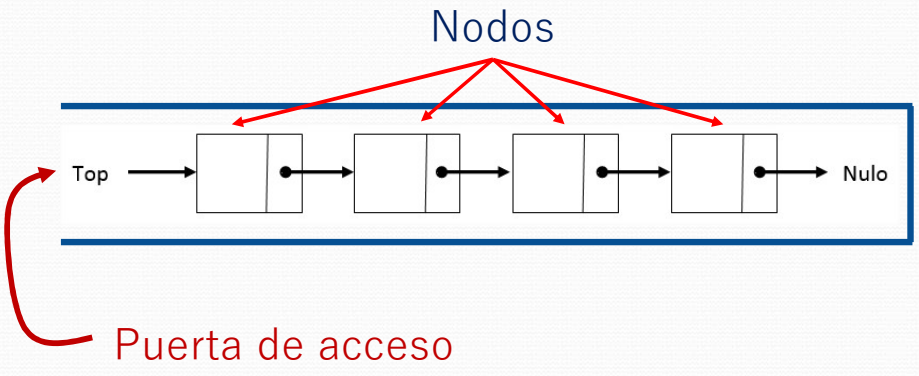
El libro "Estructuras de datos orientadas a objetos. Pseudocódigo y aplicaciones en C# .NET" aborda las pilas de las dos formas:

- *Cap. 4.- Pilas (estáticas)*
- *Cap.6.- Listas enlazadas (sección 6.9 "Implementación de pilas mediante listas simples")*



15

Representación esquemática de una pila dinámica

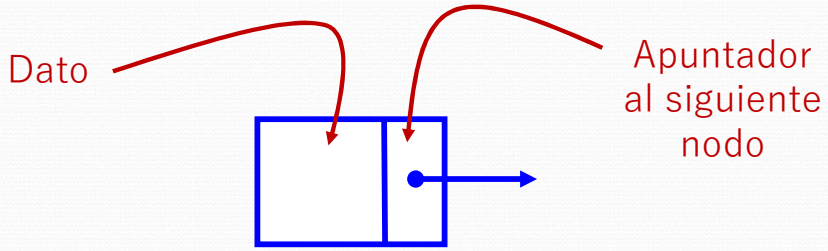


16

Arquitectura de un nodo

Cada nodo tiene 2 secciones:

1. *Dato (puede ser simple o compuesto)*
2. *Apuntador o referencia que enlaza al siguiente nodo en secuencia lógica*



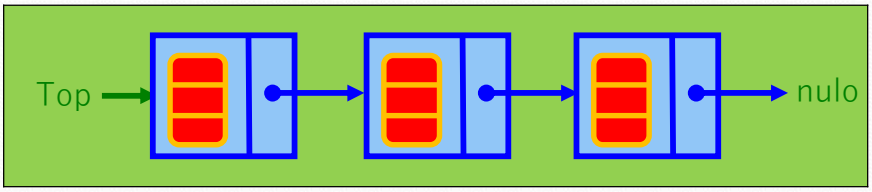
17

Diseño de una pila dinámica orientada a objetos

- Se identifican 3 tipos de objetos en la pila:
  1) *Los objetos con los datos que se desean almacenar en la pila*
  2) *Los objetos de los nodos*
  3) *El objeto de la pila*

18

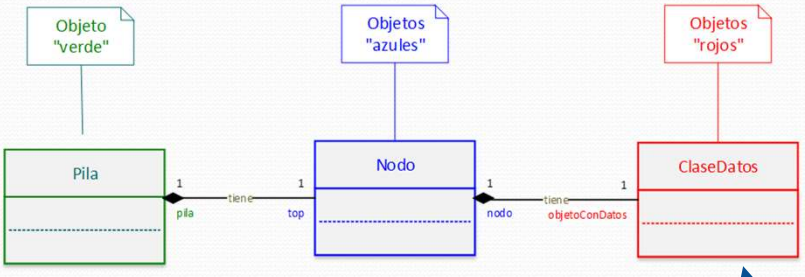
Esquema de los objetos



La **pila (objeto verde)** dentro tiene una secuencia lógica de **nodos (objetos azules)** enlazados por apuntadores y cada uno de ellos a su vez tiene dentro un **objeto con los datos** que se desean almacenar y ordenar **(objetos rojos)**

19

Diseño genérico y didáctico de la pila dinámica orientada a objetos



- *Esta clase puede ser...*
- `Empleado`
- `Escuela`
- `Médico`
- *etc. (según lo que se desee almacenar en la pila)*

20

Diseño orientado a objetos de una pila dinámica

- Se diseña una pila con objetos creados por medio de varios tipos de clases:
  - *Clase "roja".- Sirve para crear objetos con los datos que se desean almacenar en la pila*
  - *Clase "azul".- Clase para crear los nodos*
  - *Clase "verde".- Clase para crear el objeto de la pila dinámica*

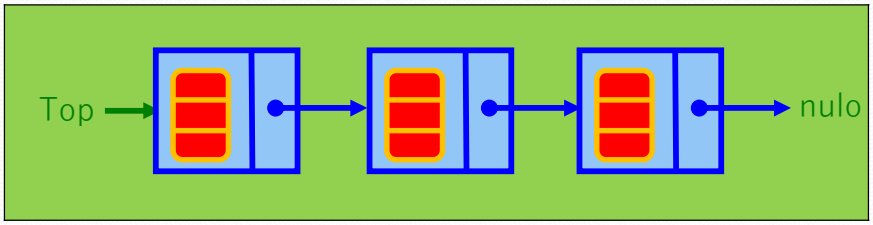
21

Notación de colores

- Los estudiantes deben poner especial atención a los colores usados en las figuras explicativas del resto del curso.
- Los objetos estarán identificados por colores
  - *Objetos "rojos".- Contienen los datos que se desean almacenar en la pila dinámica*
  - *Objetos "azules".- Representan los nodos de la pila*
  - *Objeto "verde".- Es la pila dinámica*

22

Diseño de clases



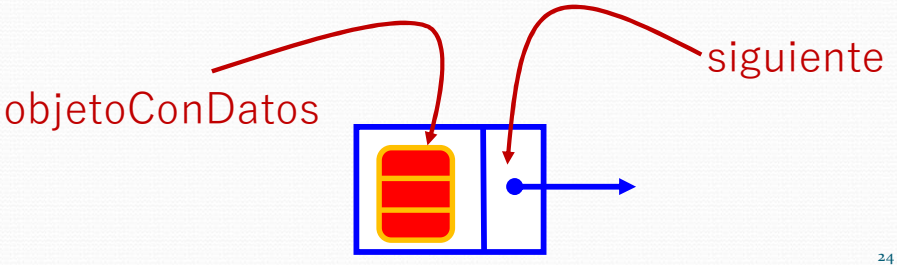
- **Clase "verde"**.- *Define la **composición** entre la pila y el nodo Top. También tiene los métodos y propiedades para administrar la pila dinámica.*
- **Clase "azul"**.- *Define los componentes de los nodos.*
- **Clase "roja"**.- *Definiciones de los datos que se desean almacenar en la pila dinámica.*

23

Declaración del nodo

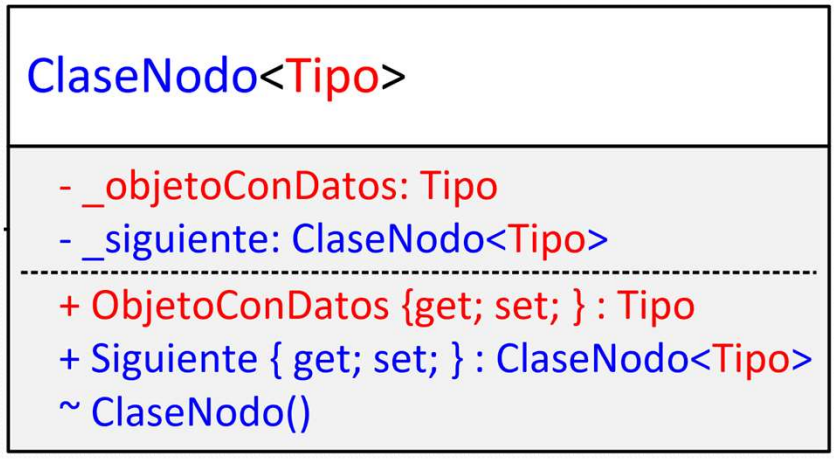
Cada nodo es un obtiene 2 atributos (con sus propiedades):

1. ***objetoConDatos***.- *El nodo recibirá un objeto con los datos que se desean almacenar en la pila*
2. ***siguiente***.- *Apuntador que enlaza al siguiente nodo lógico en la pila*



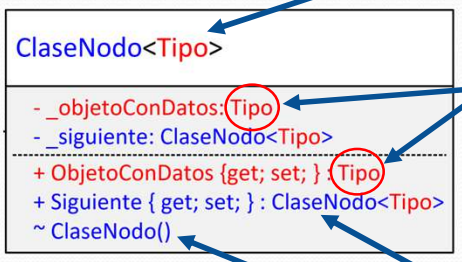
24

Diseño de la clase de los nodos



25

Diseño de la `ClaseNodo`

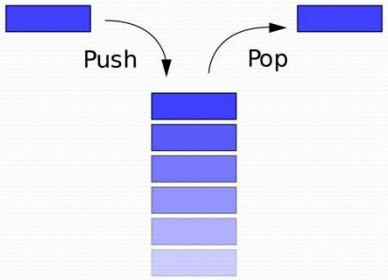


- *Clase parametrizada para recibir cualquier tipo de objeto "rojo"*
- *El parámetro `<Tipo>` define el tipo de objeto "rojo" que estará dentro de cada nodo "azul"*
- *El apuntador `Siguiente` NO almacena otro nodo "azul" sino apunta hacia otro nodo de su mismo tipo*
- *Al eliminar un nodo "azul", su destructor elimina el objeto "rojo" que contiene*

26

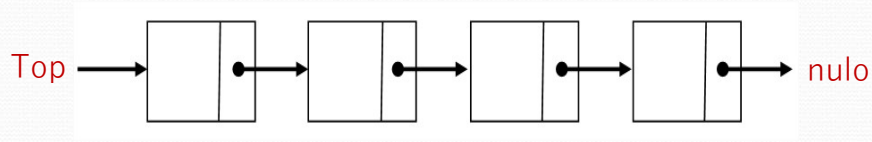
Operaciones en una pila dinámica orientada a objetos

- *Creación de la pila*
- *Push(dato)*
- *Pop( )*
- *Pop(dato)*
- *Recorrido*
- *Búsqueda*
- *Eliminar todos los nodos (vaciar)*



27

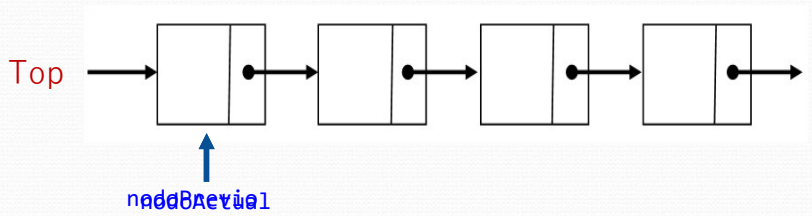
Recorrido de los nodos en una pila dinámica



- *Se verifica que la pila no esté vacía*
- *El recorrido empieza en el* `Top`
- *Se avanza al próximo nodo a través del apuntador* `Siguiente`
- *En algunos casos es necesario guardar en una variable el nodo previo al cambiar al siguiente nodo*
- *El recorrido termina al llegar al nodo que apunta a nulo (indicador del fondo de la pila)*

28

Ejemplo de recorrido en una pila dinámica

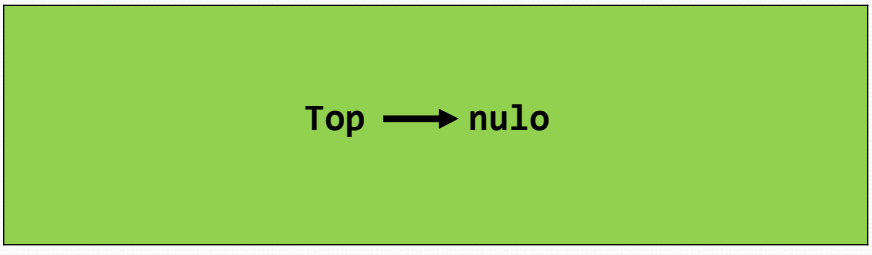


Termina el recorrido
(llega al fondo de la pila)

29

Creación de una pila dinámica

Cuando se crea una pila dinámica el Top apunta a nulo



30

Situaciones críticas

- *Son las situaciones que se pueden presentar al realizar operaciones con estructuras de datos*
- *El programador debe prever para diseñar algoritmos eficientes*

31

Método Push

- Situaciones críticas:
  - *Alta a pila vacía*
  - *Alta al principio.- Se inserta el dato al inicio de la pila*
  - ***No** permitir datos duplicados*

Se debe recorrer la pila completa antes de insertar un nodo nuevo para validar que no exista duplicado

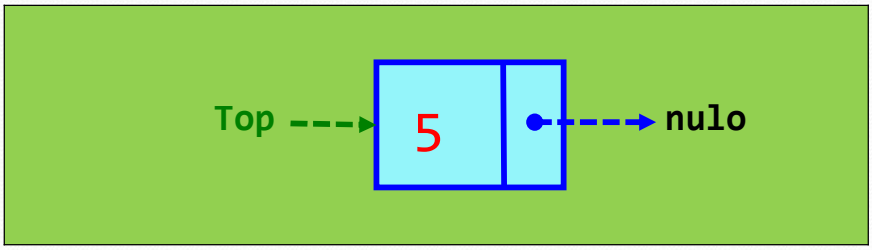
Recuerde:
La pila almacena los datos desordenados

32

Push en una pila dinámica vacía

Cuando se detecta la pila dinámica vacía:

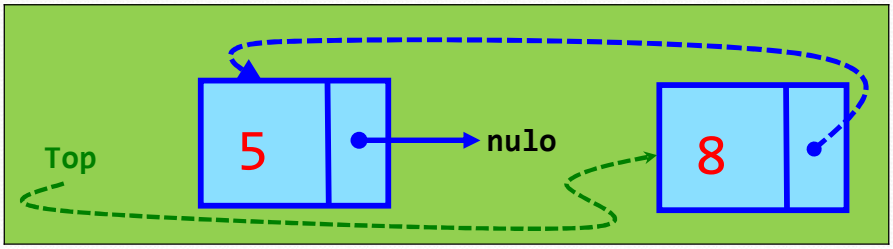
1) *Crear el nodo nuevo "azul"*
2) *Insertar el dato "rojo"*
3) *El nodo nuevo apunta a nulo*
4) *El Top apunta al nodo nuevo*



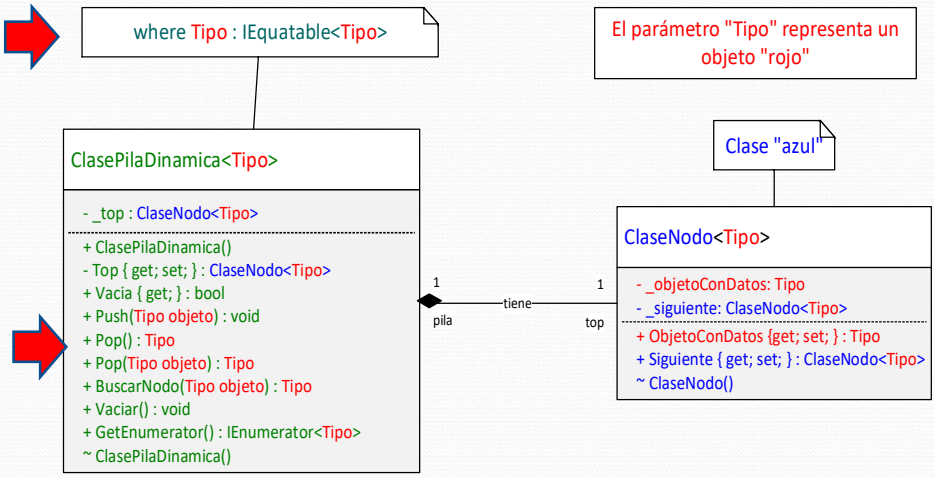
Push al principio en una pila dinámica

Cuando se ejecuta el método Push:

1) *Crear el nodo nuevo "azul"*
2) *Insertar el dato "rojo"*
3) *El nodo nuevo apunta al Top*
4) *El Top apunta al nodo nuevo*



Diseño de la clase de la pila dinámica



35

Composición Pila-Nodo

- *¿Por qué es una composición **1..1** si la pila tiene muchos nodos dentro de ella?*
  - *Porque la cardinalidad de una composición se define por la cantidad de atributos de tipo "parte" contenidos en la clase del "todo" (regla 1 de la composición)*

36

ClasePilaDinamica<Tipo>

- *Clase parametrizada para prepararla para que pueda recibir cualquier `Tipo` de objeto "rojo"*
- *Tiene una restricción de tipos para "obligar" a que la clase "roja" implemente `IEquatable`*
  - *Se requiere el método `Equals()` para buscar un objeto "rojo" almacenado en la pila*
- *¿Por qué no tiene restricción de tipos para `IComparable`?*

37

Componentes de la clase

- `_top`.- *Atributo privado que apunta al primer nodo de la pila*
- `ClasePilaDinamica()`.- *Es el constructor que inicializa vacía la pila.*
- `Vacia`.- *Propiedad pública booleana de solo lectura para detectar si la pila está vacía (devuelve `true` si la pila está vacía).*

38

Componentes de la clase *(cont.)*

- `Top`.- *Propiedad privada que apunta al primer nodo de la pila*
- `Push(Tipo objeto):void` .- *Método público que recibe como parámetro el objeto "rojo" que se desea almacenar en la pila.*
- `Pop():Tipo` .- *Método público que **NO** recibe parámetro y elimina el nodo ubicado en la puerta de la pila. **Devuelve** el objeto "rojo" eliminado.*
- `Pop(Tipo objeto):Tipo` .- *Método público que recibe como parámetro el objeto "rojo" intermedio o final que se desea borrar de la pila. **Devuelve** el objeto "rojo" eliminado.*

39

Componentes de la clase *(cont.)*

- `BuscarNodo(Tipo objeto):Tipo` .- *Método público que recibe como parámetro el objeto "rojo" que se desea consultar en la pila. Devuelve el objeto "rojo" localizado.*
- `Vaciar():void` .- *Método público que recorre la pila para eliminar todos los nodos "azules" con sus respectivos objetos "rojos"*

40

Componentes de la clase *(cont.)*

- `GetEnumerator():IEnumerator<Tipo>`.- *Iterador que recorre cada nodo "azul" de la pila para consultar su objeto "rojo"*

- `~ClasePilaDinamica()`.- *Destructor que invoca al método `Vaciar()` para eliminar todos los `nodos` de la pila.*

41

Tarea 2.01.- DF Pilas (constructor, vacía e iterador)

- Subir a MS Teams los archivos *.vsdx con diagramas de flujo de:
  - *Constructor de la pila dinámica*
  - *Propiedad para detectar si la pila dinámica está vacía*
  - *Iterador GetEnumerator()*

42

Diseño de la clase "roja"

- *Requisitos: Debe contener al menos un dato de los siguientes tipos:*
  - *Int*
  - *Double*
  - *String*
  - *Char*
  - *DateTime*
  - *Bool*
  - *String con la ruta del archivo que contiene una fotografía del objeto*

Se recomienda consultar las filminas
"El lenguaje C# y diseño de formas"
Para usar el PictureBox

43

Diseño de la clase "roja" (*cont.*)

IMPORTANTE

Considere un dato dentro de su clase "roja" que sea único e irrepetible que sirva para identificar a un objeto y que sea el criterio de comparación entre objetos

(Una clave para identificar y comparar objetos "rojos")

44

Tarea 2.02

- *Resolver la Tarea 2.02.- Diseño de la clase "roja" de la pila dinámica en MS Teams*
- *Subir el archivo *.vsdx con el diagrama de la clase "roja" elaborado en Microsoft Visio*
- *Incluir las interfaces correspondientes*

45

Tarea 2.03.- DF Pilas dinámicas (Push)

- Subir a MS Teams el archivo *.vsdx con diagrama de flujo de:
  - *Método Push() para agregar un objeto "rojo" a la pila dinámica*

46

Diseño de la forma de la aplicación visual

- *Requisitos: Debe contener al menos uno de estos controles visuales:*
  - *Textbox*
  - *Button*
  - *ComboBox*
  - *DateTimePicker*
  - *CheckBox*
  - *PictureBox*
  - *DataGridView*
  - *RadioButton*

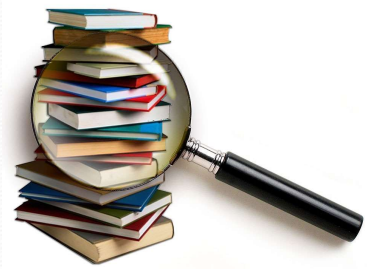
Elija el control adecuado para cada dato capturado

Se recomienda consultar las filminas ***"Uso de los controles visuales"***

47

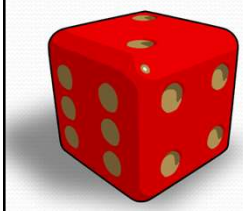
Investigación

- *Agregue un botón a su aplicación para crear* `10` *nodos con datos generados de manera aleatoria*
- *Muestre los datos generados en el dataGridView*



48

LECTURA



Para generar datos aleatorios, se recomienda la lectura de los apuntes

¿Cómo generar datos aleatorios en C#?

Incluye generar:

- *Nombres aleatorios*
- *Sexo*
- *Fecha*
- *Datos booleanos*
- *etc.*

https://nlaredo.tecnm.mx/takeyas/Apuntes/Fundamentos%20de%20Programacion/Apuntes/07.-Aleatorios.pdf

49

Tarea 2.04.- Diseño de la forma de pilas dinámicas

- Diseñar la forma en C#:
- *Capturar los datos usando los controles visuales adecuados*
- *Agregar un dataGridView de solo lectura para visualizar los datos de los objetos "rojos"*
- *Implementar el método Push() para agregar objetos "rojos" a la pila dinámica y visualizarlos en el dataGridView*
- *Subir a MS Teams un archivo comprimido con la aplicación completa (P. ejem. LopezTakeyasBruno.ZIP)*

50

Tarea 2.05.- DF Pilas dinámicas (Buscar)

- Subir a MS Teams el archivo *.vsdx con diagrama de flujo de:
  - *Método para buscar un objeto "rojo" en la pila dinámica (debe devolver el objeto encontrado)*

51

Eliminación de datos en una pila dinámica orientada a objetos

- *Se sobrecarga el método* `Pop()`
- `Pop()`.- *No recibe parámetros porque simplemente elimina el nodo apuntado por* `Top`
- `Pop(Tipo objeto)`.- *Recibe como parámetro el objeto "rojo" intermedio o al final que se desea eliminar de la pila*
  - *Durante el recorrido es necesario "guardar" el nodo previo*
  - *No se puede interrumpir la búsqueda por anticipado ya que la pila almacena datos desordenados*

52

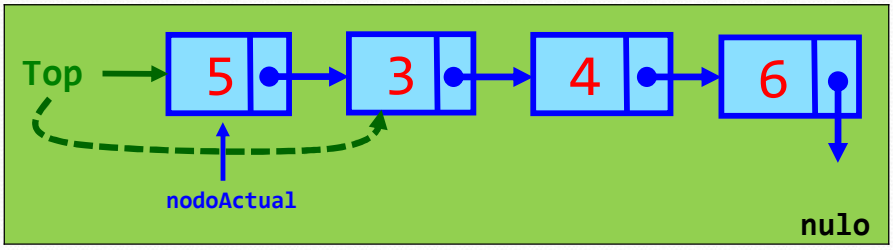
Situaciones críticas del método Pop()

- *Si la pila está vacía.- Se dispara una excepción*
- *Baja al principio.- Se elimina el dato apuntado por Top*

Método Pop( )

Cuando se elimina el nodo ubicado en la puerta de la pila:

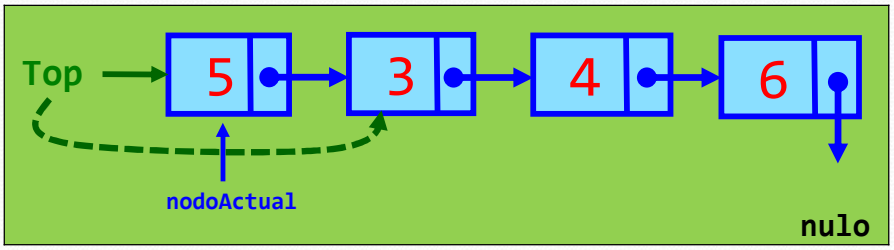
1) *El Top apunta al nodo que apuntaba el primer nodo de la pila*
2) *Se elimina el nodoActual*



Método Pop(Tipo objeto)

Cuando se elimina el nodo ubicado en la puerta de la pila:

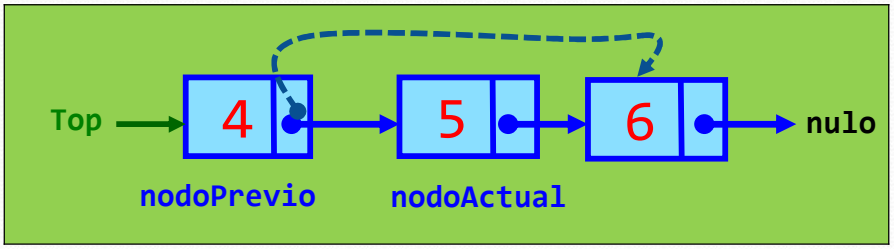
1) *El Top apunta al nodo que apuntaba el primer nodo de la pila*
2) *Se elimina el nodoActual*



Pop(Tipo objeto).- Baja intermedia en una pila

Cuando se elimina un nodo intermedio:

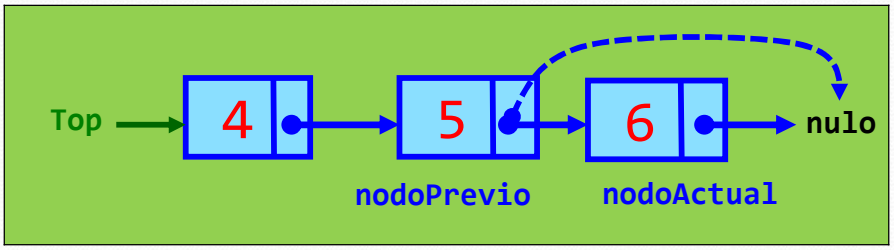
1) *El nodoPrevio apunta al nodo que apuntaba el nodoActual*
2) *Se elimina el nodoActual*



Pop(Tipo objeto).- Baja al final en una pila

Cuando se elimina el nodo ubicado en el fondo de la pila:

1) *El nodoPrevio apunta al nodo que apuntaba el nodoActual (ahora apunta a nulo)*
2) *Se elimina el nodoActual*



Eliminación de objetos

- *La forma natural de borrar un objeto es asignarle el valor null*

```
// Creación del objeto "azul" del nodoActual
ClaseNodo<Tipo> nodoActual = new ClaseNodo<Tipo>();
   .
   .
nodoActual = null; // Eliminación del nodoActual
```

- *Sin embargo, no todos los datos aceptan el valor null, entonces… ¿cómo se eliminarían?*

58

Destructor de la clase "azul"

- *La `ClaseNodo<Tipo>` es parametrizada y está preparada para recibir un objeto "rojo" de cualquier tipo.*
- *El destructor de la clase "azul" elimina el objeto con datos "rojo" que contiene.*
- *Utiliza `default(Tipo)` para eliminar el objeto "rojo" porque desconoce si el ObjetoConDatos acepta el valor null.*

```
~ClaseNodo() // Destructor de la clase "azul"
{
     // Elimina el ObjetoConDatos "rojo"
     ObjetoConDatos = default(Tipo);
}
```

Tarea 2.06.- DF Pilas dinámicas (Pop)

- Subir a MS Teams el archivo *.vsdx con diagrama de flujo de:
  - Método `Pop( )`
    - *Agregar un botón "Pop" para este método*
  - Método `Pop(Tipo objeto)`
    - *Seleccionar un renglón del dataGridView y después oprimir el botón "Eliminar"*
    - *Al seleccionar un renglón del dataGridView se deben mostrar los datos en los controles adecuados*
  - En ambos casos:
    - *Confirmar la operación*
    - *Subir los diagramas en archivos separados*
    - ***Devolver el objeto "rojo" eliminado***

60

Vaciar la pila

- Situaciones críticas:
  - *Verificar si la pila ya está vacía*
  - *Recorrer los nodos "azules" de la pila*
  - *Durante el recorrido, guardar el* `nodoPrevio`
  - *Eliminar el* `nodoPrevio`
  - *Al terminar, el* `Top` *debe apuntar a nulo*

61

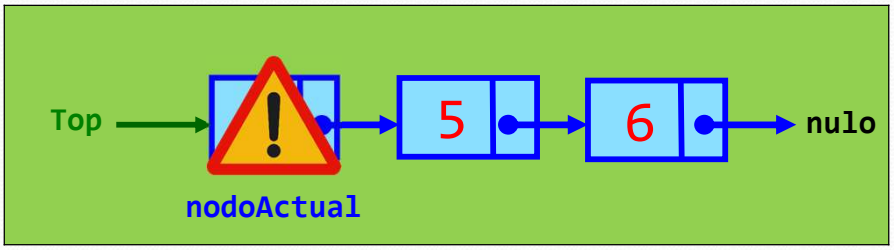
Precaución al vaciar la pila

- **NOTA IMPORTANTE:**
  - *Durante el recorrido de los nodos "azules" de la pila,* ***NO*** *se debe eliminar el* `nodoActual`
  - *Si se eliminara el* `nodoActual` *se pierde el apuntador* `Siguiente`
  - *Se perdería la secuencia lógica de los nodos "azules"*

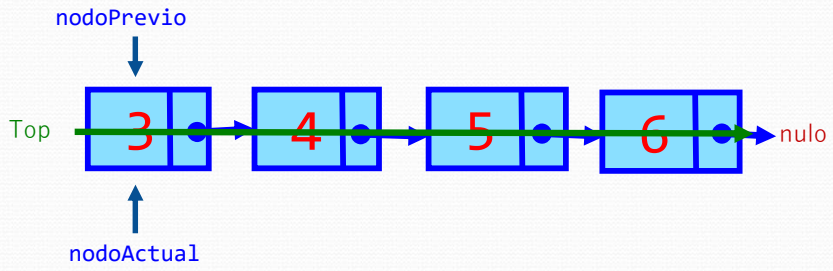
62

Precaución al vaciar la pila

- NOTA IMPORTANTE:
  - *Si se eliminara el `nodoActual` . . .*
  - *¿Cómo verificaría cuál es el siguiente nodo?*
  - *Por lo tanto **NO** debe eliminarse el `nodoActual` sino el `nodoPrevio`*



Ejemplo: Vaciar la pila



Termina el recorrido

64

Tarea 2.07.- DF Pilas dinámicas (Vaciar y Destructor)

- Subir a MS Teams los archivos *.vsdx con diagramas de flujo de:
  - *Método para vaciar la pila dinámica*
  - *Destructor de la pila dinámica*

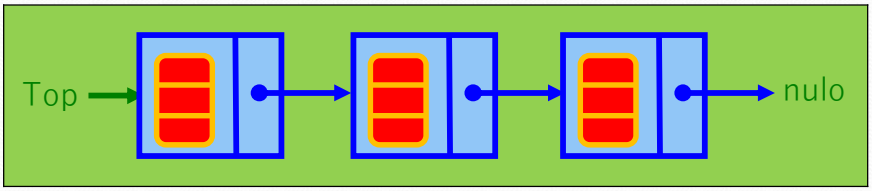
65

Tarea 2.08.- Aplicación completa de pilas dinámicas

- *Agregar un botón "Vaciar"*
- *Solicitar al usuario que confirme las operaciones (insertar, eliminar, vaciar, etc.). Preguntarle si está seguro que desea realizar la operación solicitada*
- *Mostrar los mensajes adecuados*
- *Subir a MS Teams un archivo comprimido con la aplicación completa (P. ejem. LopezTakeyasBruno.ZIP)*

66

Nivel de abstracción



- *No se debe perder de vista que los objetos "rojos" son privados, por lo tanto sus componentes son inacesibles para el objeto "verde"*

67

Nivel de abstracción (*cont.*)

- *El objeto "verde" recibe un objeto "rojo" y lo compara para almacenarlo ....*

  ¡¡¡ SIN SABER LO QUE TIENE DENTRO !!!

- *¿Cómo es posible que el objeto "verde" compare y almacene objetos "rojos" sin tener acceso a sus componentes?*

68

"Pensar en objetos …"

- *El objeto "verde" **NO** compara los objetos "rojos" (ellos mismos se comparan entre sí).*
- *El objeto "verde" **NO** requiere acceso a los componentes de los objetos "rojos" para manipularlos.*
- *Recibe cualquier tipo de objetos "rojos" …*

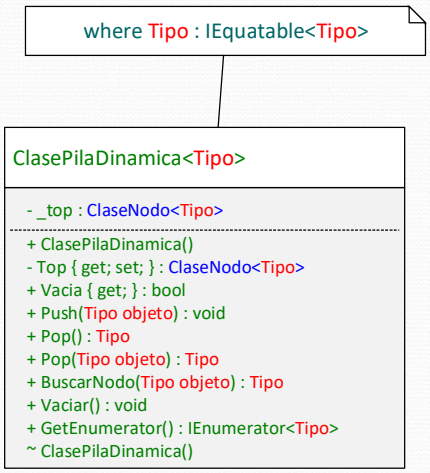
¡¡¡ SIN MODIFICAR NI UNA LÍNEA DE SU CÓDIGO!!!

- ***¿Cómo lo logra? …***
  - *Clases parametrizadas*
  - *Uso de interfaces*
  - *Composición*
  - *Comportamiento polimórfico*
  - *Restricción de tipos*

69

Diseño de una clase polimórfica

- La clase de la pila tiene estas características:
  - *Almacena objetos "rojos" desordenados*
  - *No permite duplicados*



70

Diseño de una clase polimórfica *(cont.)*

- *¿Qué cambios se harían para que la misma clase "**verde**" almacene los datos ordenados?*
- *¿Qué cambios se harían para que la misma clase "**verde**" permita objetos "rojos" duplicados?*

71

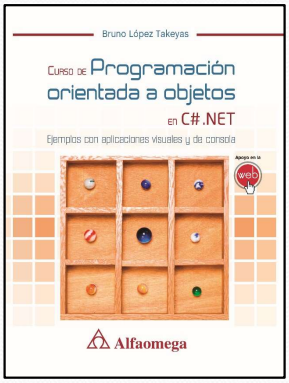
Diseño de una clase polimórfica *(cont.)*

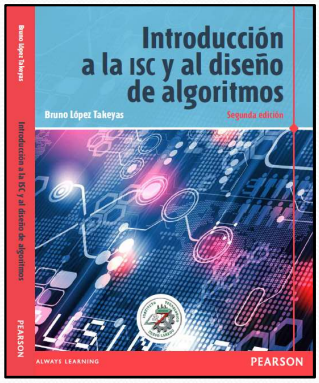
- *¿Qué cambios se harían para que la misma clase "**verde**" muestre cualquiera de los siguientes comportamientos:*
  - *Objetos "rojos" ordenados sin duplicados*
  - *Objetos "rojos" ordenados con duplicados*
  - *Objetos "rojos" desordenados sin duplicados*
  - *Objetos "rojos" desordenados con duplicados?*

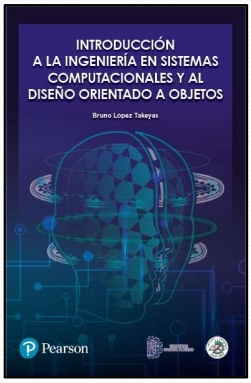
72

Otros libros del autor

https://nlaredo.tecnm.mx/takeyas/Libro







bruno.lt@nlaredo.tecnm.mx

Bruno López Takeyas