Clase Alumno

- getCuenta(): string
- getNombre(): string
- setCuenta(string): void
- setNombre(string): void

Definición: Clase del objeto Alumno utilizado para almacenar nombre y cuenta.

- getCuenta(): string

Retorna un string con el número de cuenta del Objeto tipo Alumno.

- getNombre(): string

Retorna un string con el nombre del Alumno.

- setCuenta(string): void

Con esta función se envía un string con el número de cuenta de un Alumno para ser almacenado. No retorna nada.

- setNombre(string): void

La función pedirá un string con el nombre del alumno al que se le asignará dicho nombre.

Clase Nodo

- setSiguiente(Nodo*): void
- setAnterior(Nodo*): void
- setItem(Object*): void
- getItem(): Object*
- getSiguiente(): Nodo*
- getAnterior(): Nodo*

Definición: Clase del objeto tipo Nodo. Almacena el nodo siguiente, el item y el nodo anterior de un nodo, es útil para las listas enlazadas.

- setSiguiente(Nodo*): void

La función pide un objeto de tipo Nodo para almacenarlo como el nodo precedente del nodo con el que se está trabajando. No retorna nada.

- setAnterior(Nodo*): void

La función pide un objeto de tipo Nodo para almacenarlo como el nodo antecedente del nodo con el que se está trabajando. No retorna nada.

- setItem(Object*): void

Esta función tomara el Objecto enviado para ingresarlo como el item del nodo. No retorna nada.

- getItem(): Object*

Retorna el item de tipo Object almacenado en el Nodo.

-getSiguiente(): Nodo*

Retorna el objecto de tipo Nodo del nodo siguiente apuntado por el nodo con el que se trabaja.

-getAnterior(): Nodo*

Retorna el nodo anterior, que es un objeto tipo Nodo, del nodo con el que se trabaja.

Clase Símbolo

- getSimbolo(): char
- setSimbolo(char): void

Definición: Clase del objeto tipo Símbolo. Almacena un char, el cual se puede actualizar y consultar.

- setSimbolo(char): void

Almacena el char enviado en el objeto Símbolo. No retorna nada

- getSimbolo(): char

Retorna el char almacenado en el objeto Símbolo.

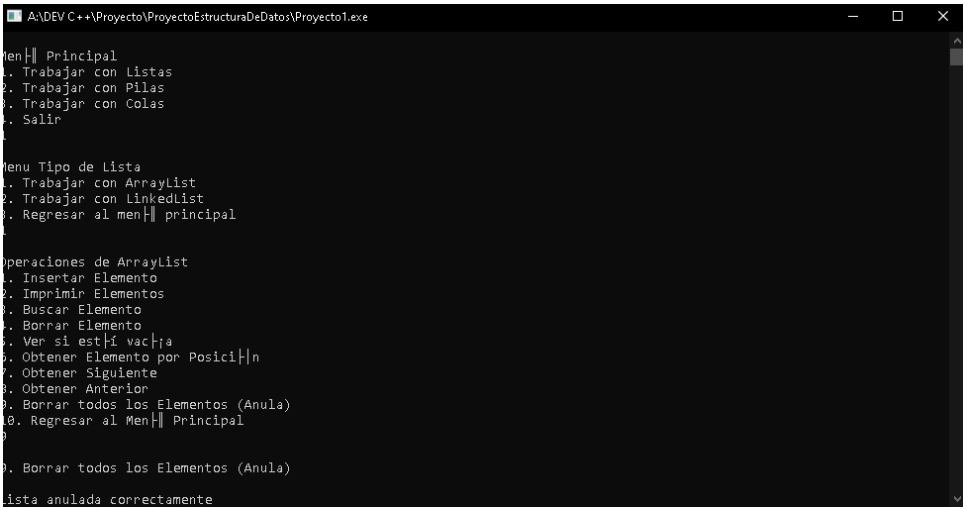
Clase TDALista

- anula(): void
- inserta(Object*, int): bool
- siguiente(int): Object*
- anterior(int): Object*
- append(Object*): void
- imprimir_lista(): void
- suprime(int): bool
- recupera(int): Object*
- localiza(Object*): int
- primero(): Object*
- vacia(): bool

Definición. Clase correspondiente al Tipo de Dato Abstracto lista. Es capaz de almacenar un tipo de dato “Alumno”.

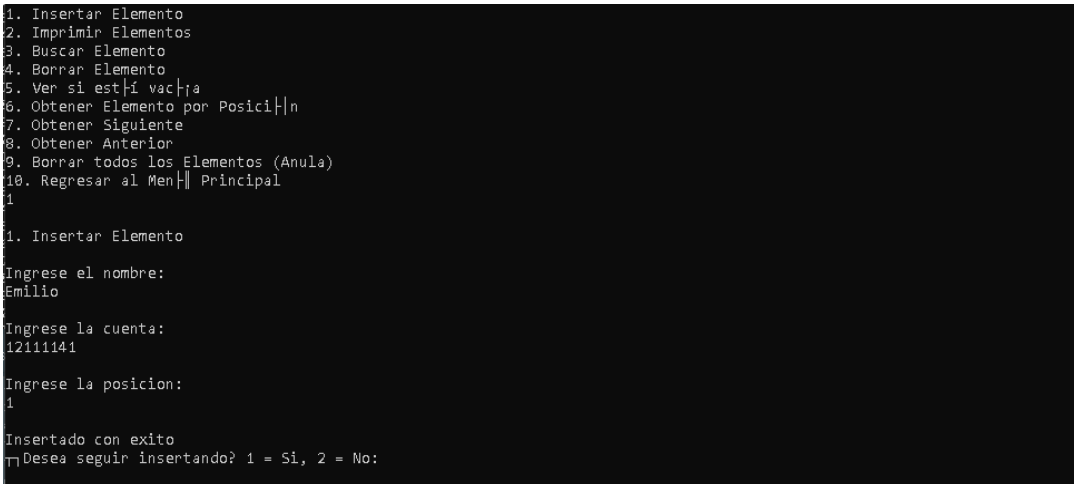
- anula(): void

Elimina todos los elementos presentes en la lista. En caso de que la lista ya esté vacía, imprime un mensaje. No retorna nada.



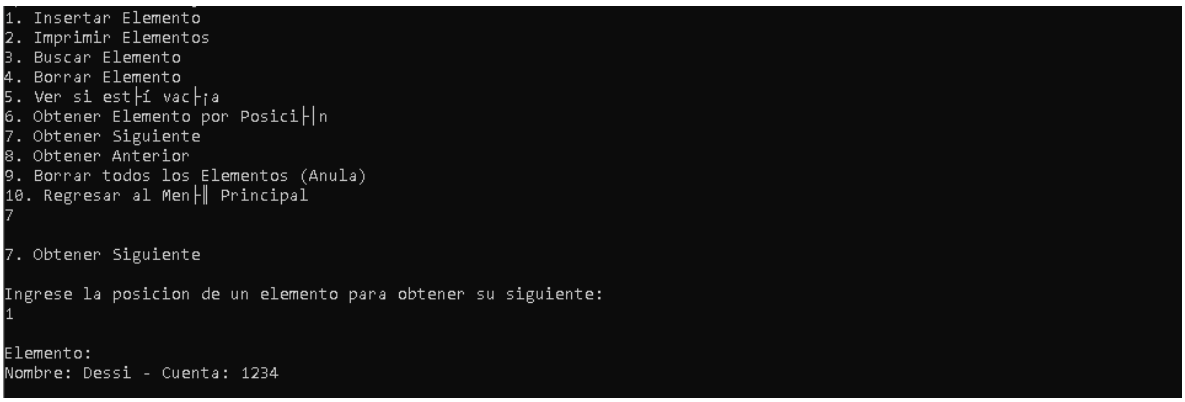
- inserta(Object*, int): bool

Solicita al usuario el elemento a insertar y la posición a insertarlo. Luego deberá indicarle al usuario si la operación se realizó con éxito, y preguntar al usuario si desea introducir otro elemento, de ser así continuar introduciendo elementos nuevos, hasta que el usuario decida salir, en cuyo caso deberá regresar al Menú Operaciones de Listas. Retorna un booleano.



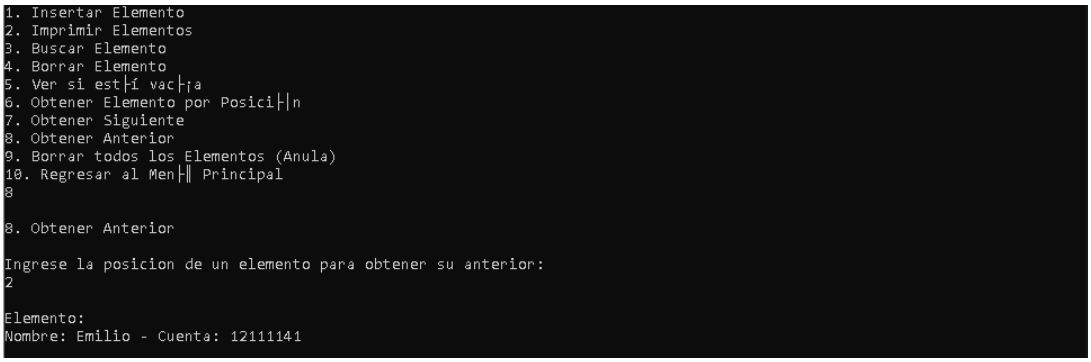
- siguiente(int): Object*

Pedirá un entero que representa una posición en la Lista y mostrará el elemento que está en la posición siguiente. Si no hay elemento en la posición siguiente mostrará un mensaje indicando que el espacio siguiente está vacío. Retorna un apuntador al objeto a utilizar.



- anterior(int): Object*

Pedirá un entero que representa una posición en la Lista y mostrará el elemento que está en la posición anterior. Si no hay elemento en la posición anterior mostrará un mensaje indicando que el espacio anterior está vacío. Retorna un apuntador al objeto a utilizar.



- append(Object*): void

Inserta un objeto de tipo Alumno, en la primera posición vacía de la lista. No retorna nada.

- imprimir_lista(): void

Despliega de manera estética los elementos que estén en la lista. No retorna nada

- suprime(int): bool

Le solicitara al usuario una posición en la lista, y elimina el elemento que esté en esa posición, le avisa al usuario si la eliminación fue exitosa o no. Retorna un booleano.

- recupera(int): Object*

Solicita al usuario un entero, que representa una posición en la lista, desplegará el elemento que se encuentre en esa posición (si la posición es válida). Si no, muestra un mensaje indicando que la posición no es válida. Retorna un apuntador al objeto a utilizar.

- localiza(Object*): int

Le solicita al usuario que introduzca un número de cuenta, y debe de buscar ese dato en los elementos de la lista, desplegando el primer Alumno encontrado en pantalla, junto con su posición. De no encontrar el dato en ningún elemento de la lista, entonces despliega un mensaje indicándolo. Retorna un entero.

- primero(): Object*

Retorna el primer elemento almacenado en la lista.

- vacia(): bool

Muestra un mensaje indicando que la lista está vacía o no está vacía.

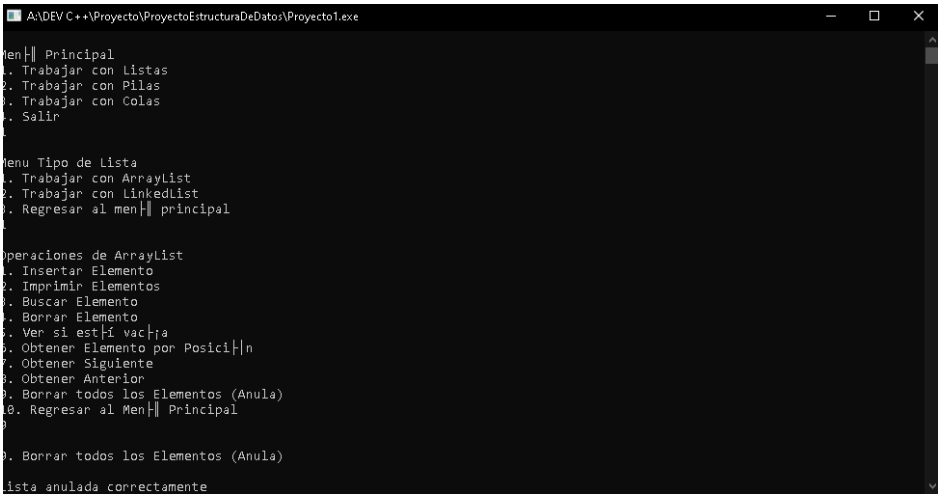
Clase LinkedList

- anula(): void
- inserta(Object*, int): bool
- siguiente(int): Object*
- anterior(int): Object*
- append(Object*): void
- imprimir_lista(): void
- suprime(int): bool
- recupera(int): Object*
- localiza(Object*): int
- primero(): Object*
- vacia(): bool

Definición. Implementación de TDALista, haciendo uso de listas enlazadas.

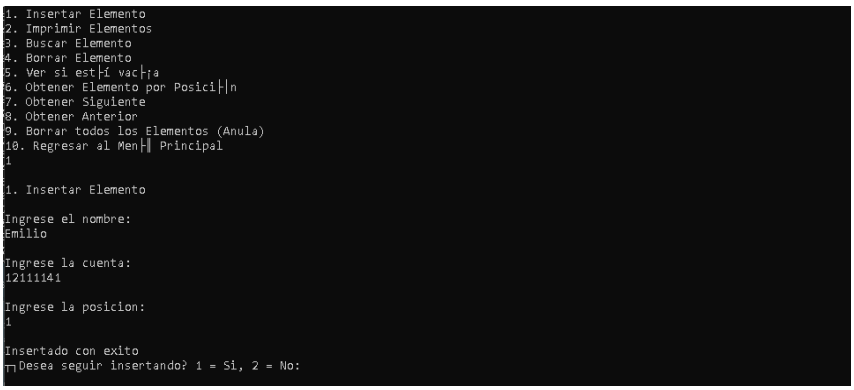
- anula(): void

Elimina todos los elementos presentes en la lista. En caso de que la lista ya esté vacía, imprime un mensaje. No retorna nada.



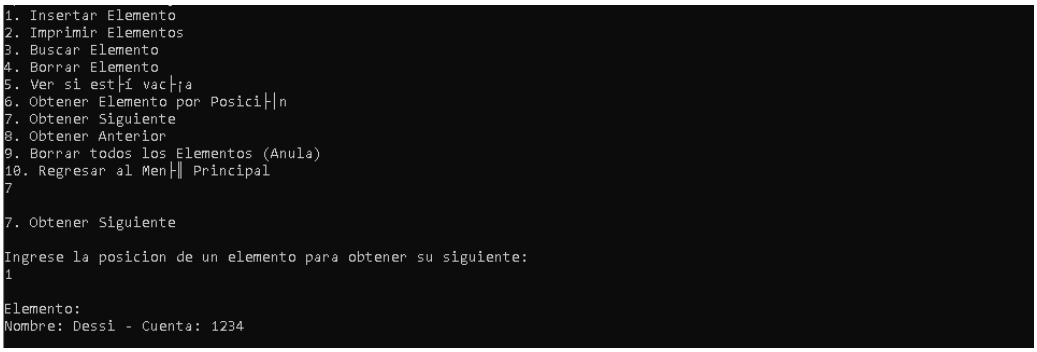
- inserta(Object*, int): bool

Solicita al usuario el elemento a insertar y la posición a insertarlo. Luego deberá indicarle al usuario si la operación se realizó con éxito, y preguntar al usuario si desea introducir otro elemento, de ser así continuar introduciendo elementos nuevos, hasta que el usuario decida salir, en cuyo caso deberá regresar al Menú Operaciones de Listas. Retorna un booleano.



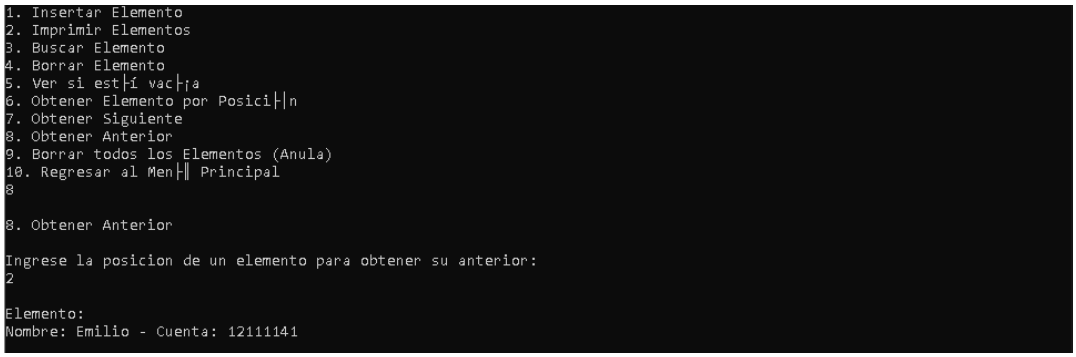
- siguiente(int): Object*

Pedirá un entero que representa una posición en la Lista y mostrará el elemento que está en la posición siguiente. Si no hay elemento en la posición siguiente mostrará un mensaje indicando que el espacio siguiente está vacío. Retorna un apuntador al objeto a utilizar.



- anterior(int): Object*

Pedirá un entero que representa una posición en la Lista y mostrará el elemento que está en la posición anterior. Si no hay elemento en la posición anterior mostrará un mensaje indicando que el espacio anterior está vacío. Retorna un apuntador al objeto a utilizar.



- append(Object*): void

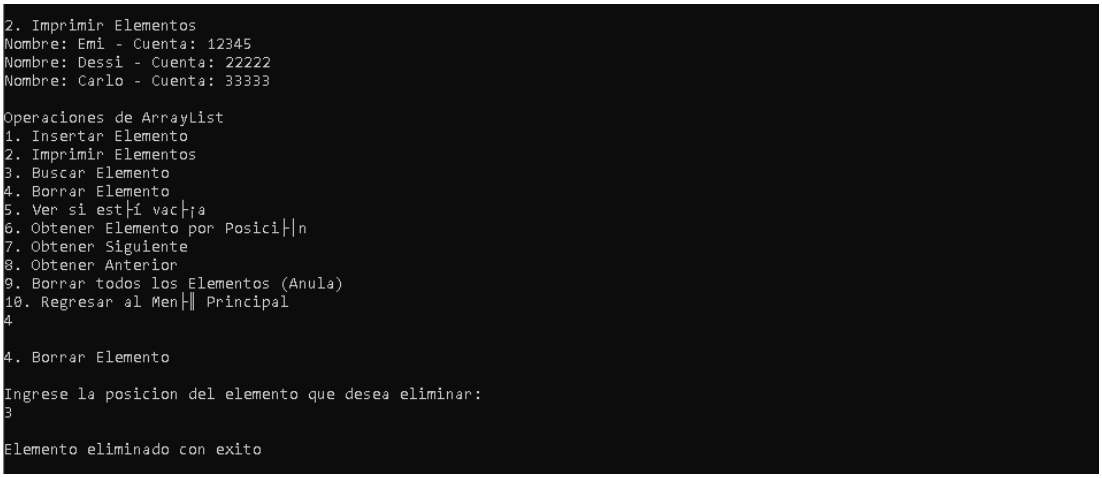
Inserta un objeto de tipo Alumno, en la primera posición vacía de la lista. No retorna nada.

- imprimir_lista(): void

Despliega de manera estética los elementos que estén en la lista. No retorna nada

- suprime(int): bool

Le solicitara al usuario una posición en la lista, y elimina el elemento que esté en esa posición, le avisa al usuario si la eliminación fue exitosa o no. Retorna un booleano.



- recupera(int): Object*

Solicita al usuario un entero, que representa una posición en la lista, desplegará el elemento que se encuentre en esa posición (si la posición es válida). Si no, muestra un mensaje indicando que la posición no es válida. Retorna un apuntador al objeto a utilizar.

- localiza(Object*): int

Le solicita al usuario que introduzca un número de cuenta, y debe de buscar ese dato en los elementos de la lista, desplegando el primer Alumno encontrado en pantalla, junto con su posición. De no encontrar el dato en ningún elemento de la lista, entonces despliega un mensaje indicándolo. Retorna un entero.

- primero(): Object*

Retorna el primer elemento almacenado en la lista.

- vacia(): bool

Muestra un mensaje indicando que la lista está vacía o no está vacía.

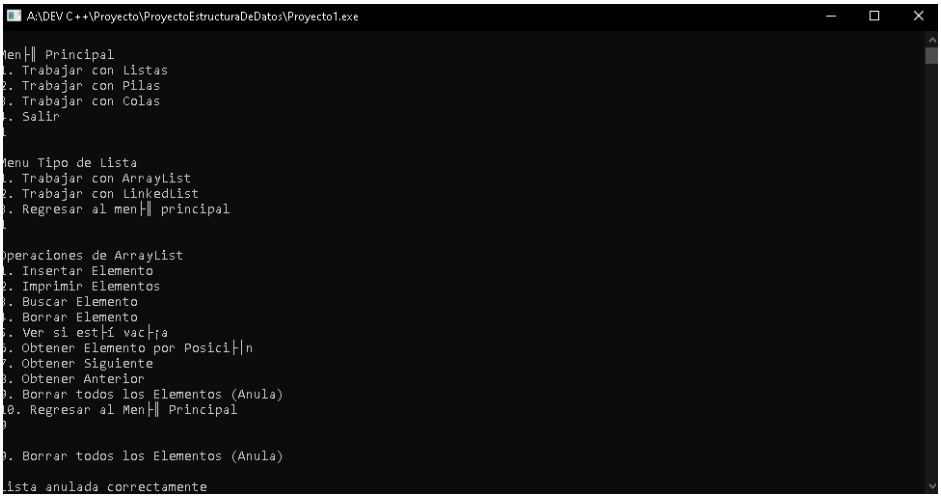
ClaseArrayList

- anula(): void
- inserta(Object*, int): bool
- siguiente(int): Object*
- anterior(int): Object*
- append(Object*): void
- imprimir_lista(): void
- suprime(int): bool
- recupera(int): Object*
- localiza(Object*): int
- primero(): Object*
- vacia(): bool

Definición. Implementación de TDALista, haciendo uso de arreglos.

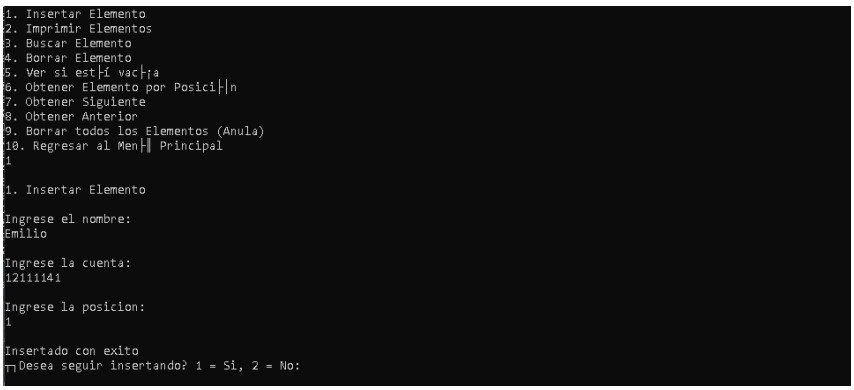
- anula(): void

Elimina todos los elementos presentes en la lista. En caso de que la lista ya esté vacía, imprime un mensaje. No retorna nada.



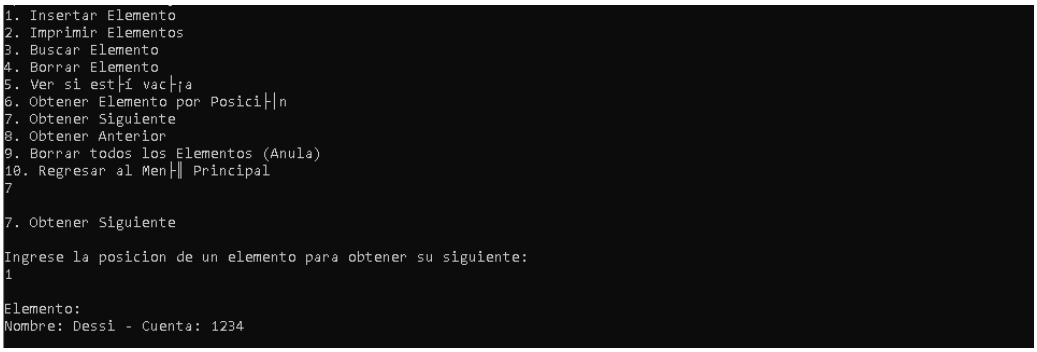
- inserta(Object*, int): bool

Solicita al usuario el elemento a insertar y la posición a insertarlo. Luego deberá indicarle al usuario si la operación se realizó con éxito, y preguntar al usuario si desea introducir otro elemento, de ser así continuar introduciendo elementos nuevos, hasta que el usuario decida salir, en cuyo caso deberá regresar al Menú Operaciones de Listas. Retorna un booleano.



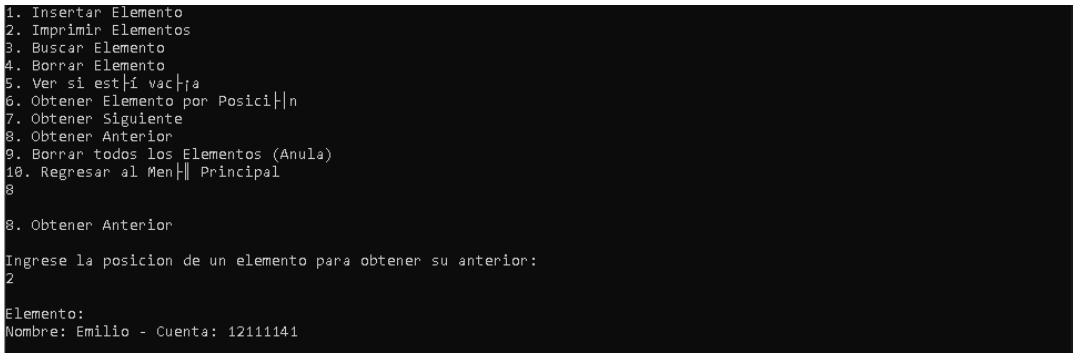
- siguiente(int): Object*

Pedirá un entero que representa una posición en la Lista y mostrará el elemento que está en la posición siguiente. Si no hay elemento en la posición siguiente mostrará un mensaje indicando que el espacio siguiente está vacío. Retorna un apuntador al objeto a utilizar.



- anterior(int): Object*

Pedirá un entero que representa una posición en la Lista y mostrará el elemento que está en la posición anterior. Si no hay elemento en la posición anterior mostrará un mensaje indicando que el espacio anterior está vacío. Retorna un apuntador al objeto a utilizar.



- append(Object*): void

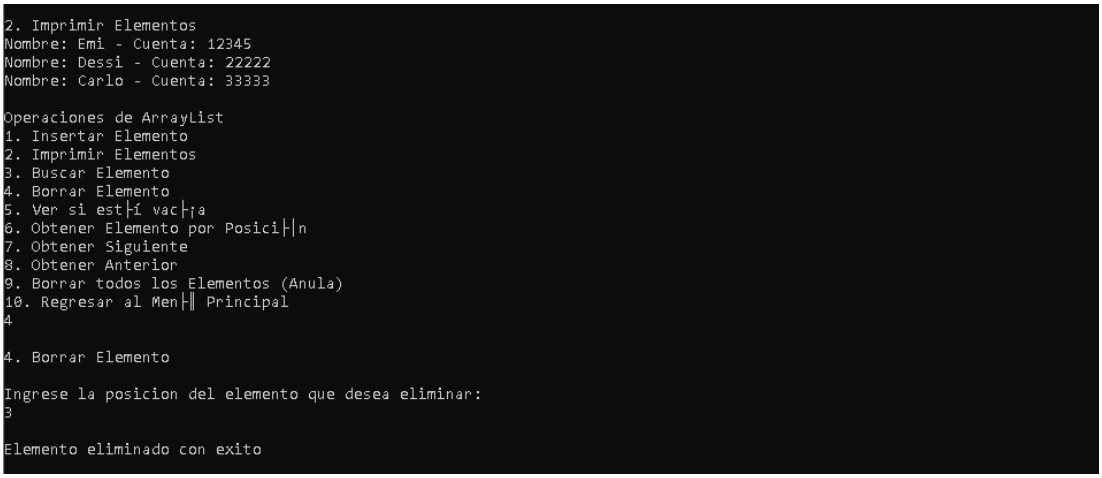
Inserta un objeto de tipo Alumno, en la primera posición vacía de la lista. No retorna nada.

- imprimir_lista(): void

Despliega de manera estética los elementos que estén en la lista. No retorna nada

- suprime(int): bool

Le solicitara al usuario una posición en la lista, y elimina el elemento que esté en esa posición, le avisa al usuario si la eliminación fue exitosa o no. Retorna un booleano.



- recupera(int): Object*

Solicita al usuario un entero, que representa una posición en la lista, desplegará el elemento que se encuentre en esa posición (si la posición es válida). Si no, muestra un mensaje indicando que la posición no es válida. Retorna un apuntador al objeto a utilizar.

- localiza(Object*): int

Le solicita al usuario que introduzca un número de cuenta, y debe de buscar ese dato en los elementos de la lista, desplegando el primer Alumno encontrado en pantalla, junto con su posición. De no encontrar el dato en ningún elemento de la lista, entonces despliega un mensaje indicándolo. Retorna un entero.

- primero(): Object*

Retorna el primer elemento almacenado en la lista.

- vacia(): bool

Muestra un mensaje indicando que la lista está vacía o no está vacía.

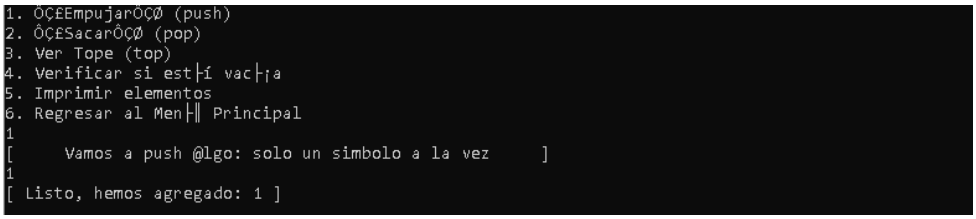
Clase TDAPila

- Push(Object*): void
- Pop(): Object*
- Top(): Object*
- vacia(): bool
- printStack(): void
- recorrerPila(int): void

Definición. Clase correspondiente al Tipo de Dato Abstracto Pila. Es capaz de almacenar un tipo de dato “Símbolo”.

- Push(Object*): void

Solicita un Símbolo al usuario, para que se “empuje” (insertar en el tope) en la pila. No retorna nada.



- Pop(): Object*

Saca un elemento de la Pila y muestra el resultado (Si es exitoso, mostrará el elemento, si no lo es, indica que no hay elementos en la pila). Retorna un apuntador al objeto a utilizar.

- Top(): Object*

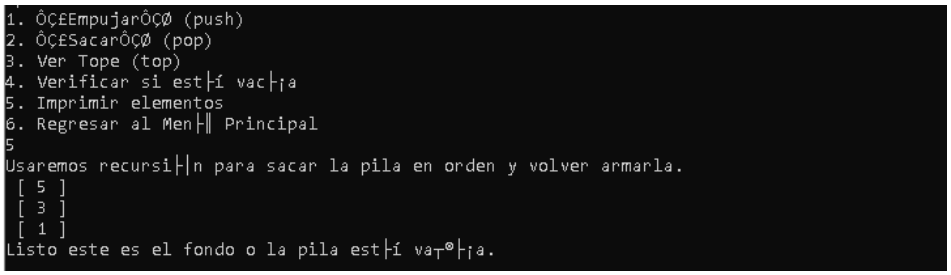
Desplegará el elemento en el tope de la pila. Retorna un apuntador al objeto a utilizar.

- vacia(): bool

Desplegará un mensaje: “Está Vacía”, si está vacía, o, “No Está Vacía”, en el caso de que no esté vacía. Retorna un booleano.

- printStack(): void

Imprimirá los elementos de la pila en el orden en el que deberían salir de ella. No retorna nada.



- recorrerPila(int): void

Recorre la pila de manera interna. No retorna nada.

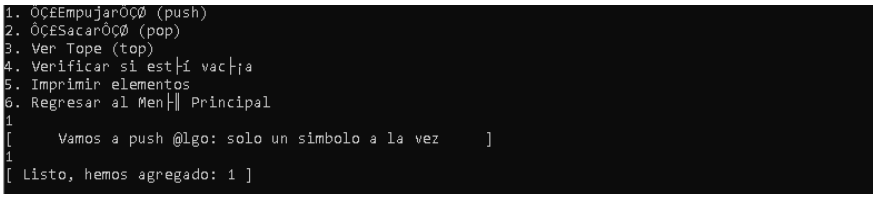
Clase LinkedStack

- Push(Object*): void
- Pop(): Object*
- Top(): Object*
- vacia(): bool
- printStack(): void
- recorrerPila(int): void

Definición. Implementación de TDAPila haciendo uso de listas enlazadas. Es capaz de almacenar un tipo de dato “Símbolo”.

- Push(Object*): void

Solicita un Símbolo al usuario, para que se “empuje” (insertar en el tope) en la pila. No retorna nada.



- Pop(): Object*

Saca un elemento de la Pila y muestra el resultado (Si es exitoso, mostrará el elemento, si no lo es, indica que no hay elementos en la pila). Retorna un apuntador al objeto a utilizar.

- Top(): Object*

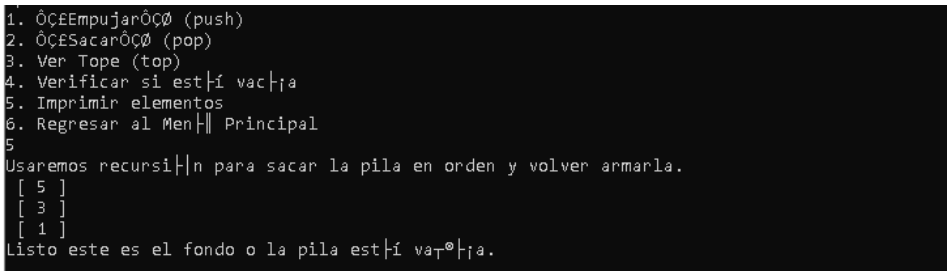
Desplegará el elemento en el tope de la pila. Retorna un apuntador al objeto a utilizar.

- vacia(): bool

Desplegará un mensaje: “Está Vacía”, si está vacía, o, “No Está Vacía”, en el caso de que no esté vacía. Retorna un booleano.

- printStack(): void

Imprimirá los elementos de la pila en el orden en el que deberían salir de ella. No retorna nada.



- recorrerPila(int): void

Recorre la pila de manera interna. No retorna nada.

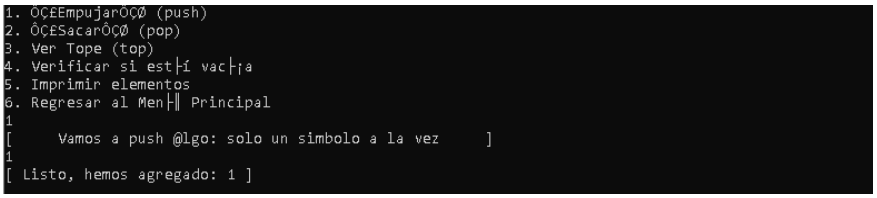
Clase ArrayStack

- Push(Object*): void
- Pop(): Object*
- Top(): Object*
- vacia(): bool
- printStack(): void
- recorrerPila(int): void

Definición. Implementación de TDAPila haciendo uso de arreglos. Es capaz de almacenar un tipo de dato “Símbolo”.

- Push(Object*): void

Solicita un Símbolo al usuario, para que se “empuje” (insertar en el tope) en la pila. No retorna nada.



- Pop(): Object*

Saca un elemento de la Pila y muestra el resultado (Si es exitoso, mostrará el elemento, si no lo es, indica que no hay elementos en la pila). Retorna un apuntador al objeto a utilizar.

- Top(): Object*

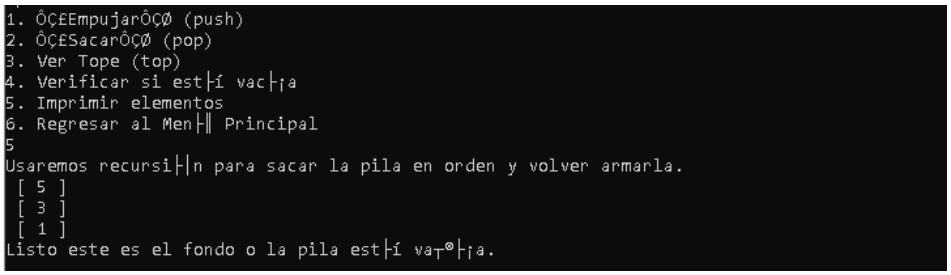
Desplegará el elemento en el tope de la pila. Retorna un apuntador al objeto a utilizar.

- vacia(): bool

Desplegará un mensaje: “Está Vacía”, si está vacía, o, “No Está Vacía”, en el caso de que no esté vacía. Retorna un booleano.

- printStack(): void

Imprimirá los elementos de la pila en el orden en el que deberían salir de ella. No retorna nada.



- recorrerPila(int): void

Recorre la pila de manera interna. No retorna nada.

Clase TDACola

- frente(): Object*
- imprime_cola(): void
- saca_de_cola(): Object*
- pone_en_cola(Object*): void
- vacia(): bool

Definición: Clase de Tipo de Dato Abstracto Cola. Almacena el tipo de dato Alumno.

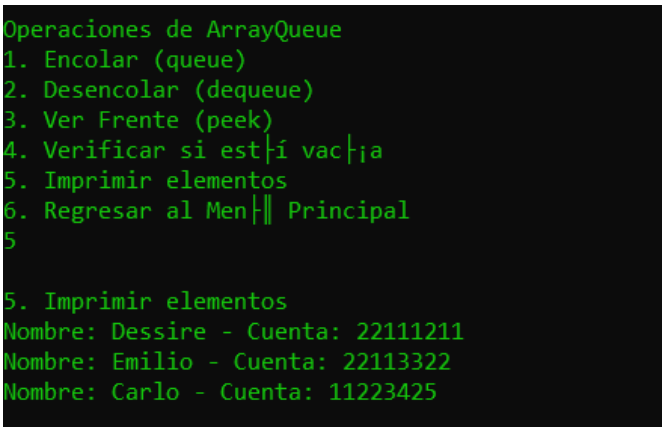
- frente(): Object*

Retorna el objeto de tipo Alumno que encabeza a la cola, es decir, el primero en ser agregado y el primero en ser sacado.

```
Menu Tipo de Cola
1. Trabajar con ArrayQueue
2. Trabajar con LinkedQueue
3. Regresar al menu principal
1

Operaciones de ArrayQueue
1. Encolar (queue)
2. Desencolar (dequeue)
3. Ver Frente (peek)
4. Verificar si est├í vac├¡a
5. Imprimir elementos
6. Regresar al Men├╣ Principal
1

1. Encolar (queue)

1. Insertar Elemento

Ingrese el nombre:
Dessire

Ingrese la cuenta:
22111211

Puesto en cola realizado con exito

Operaciones de ArrayQueue
1. Encolar (queue)
2. Desencolar (dequeue)
3. Ver Frente (peek)
4. Verificar si est├í vac├¡a
5. Imprimir elementos
6. Regresar al Men├╣ Principal
3

3. Ver Frente (peek)
Nombre: Dessire - Cuenta: 22111211
```

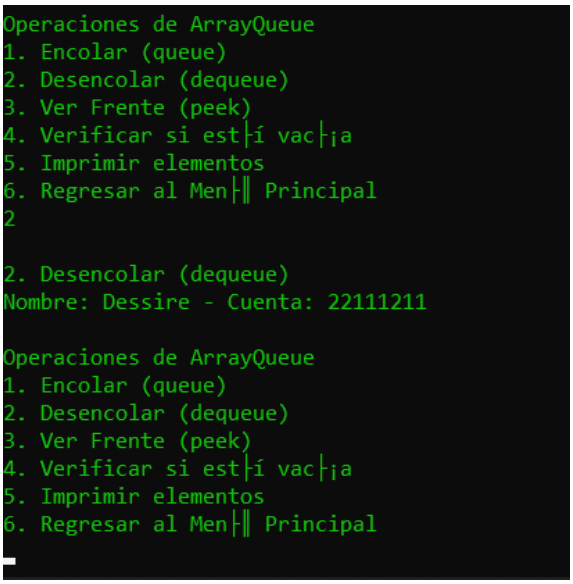
- imprime_cola(): void

Imprime la cola en el orden en el que saldrán los objetos



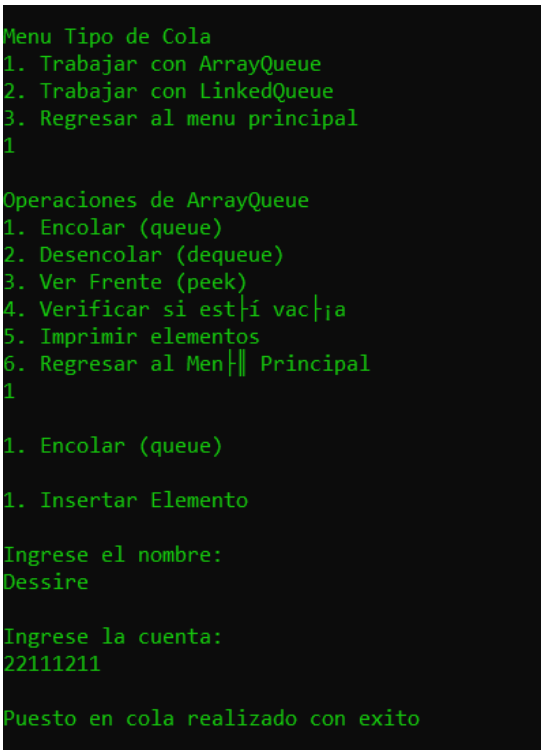
- saca_de_cola(): Object*

Sacará de cola al objeto que se encuentre en frente. Retorna dicho objeto.



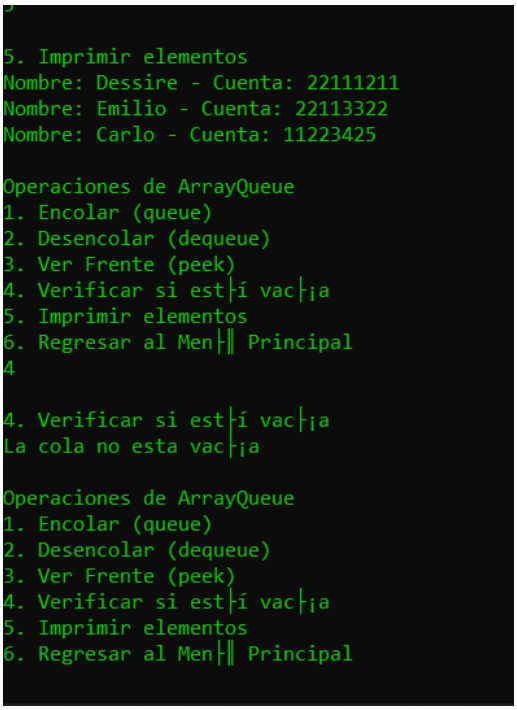
- pone_en_cola(Object*): void

Se le enviará un objeto de tipo alumno para agregar a la cola. No retorna nada.



- vacia(): bool

Retorna verdadero si la cola está vacía, y falso si esta tiene al menos un elemento.



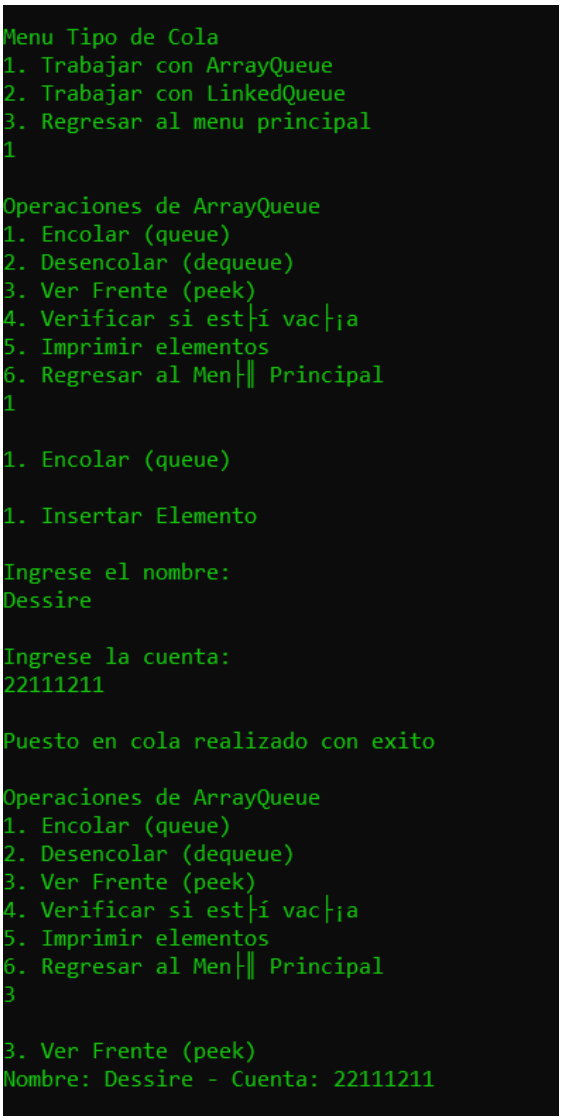
Clase Arrayqueue

- frente(): Object*
- imprime_cola(): void
- saca_de_cola(): Object*
- pone_en_cola(Object*): void
- vacia(): bool

Definición: Clase de Tipo de Dato Abstracto Cola. Almacena el tipo de dato Alumno.

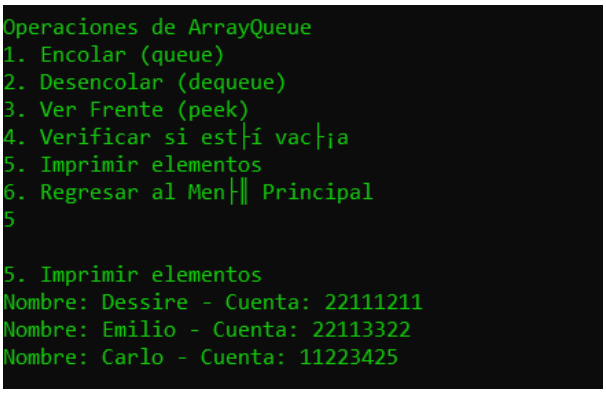
- frente(): Object*

Retorna el objeto de tipo Alumno que encabeza a la cola, es decir, el primero en ser agregado y el primero en ser sacado.



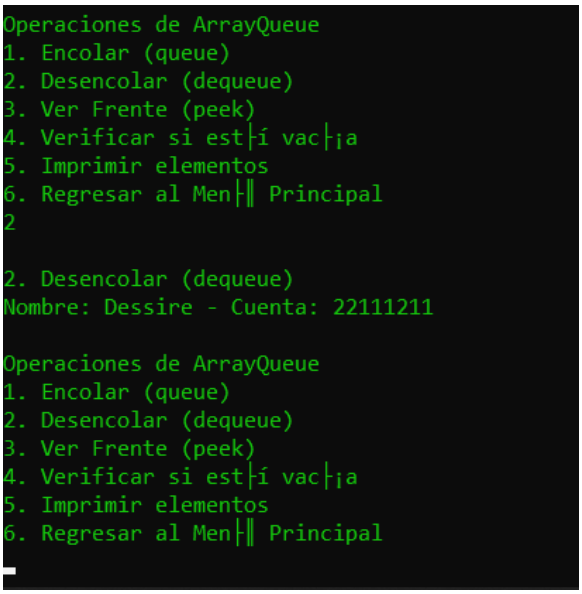
- imprime_cola(): void

Imprime la cola en el orden en el que saldrán los objetos



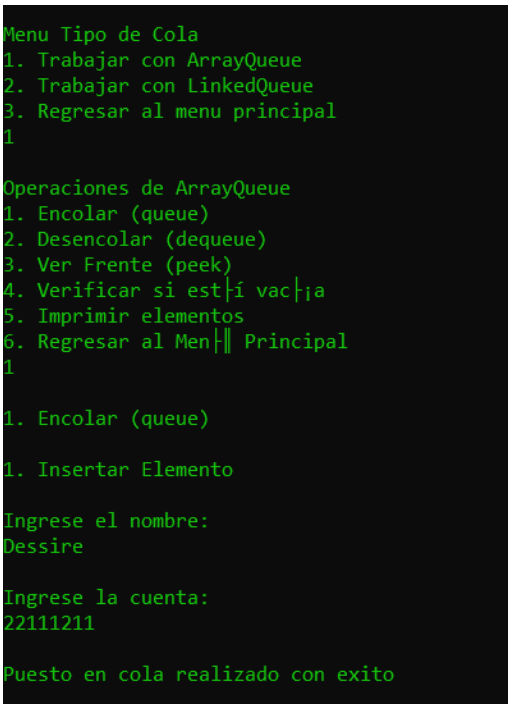
- saca_de_cola(): Object*

Sacará de cola al objeto que se encuentre en frente. Retorna dicho objeto.



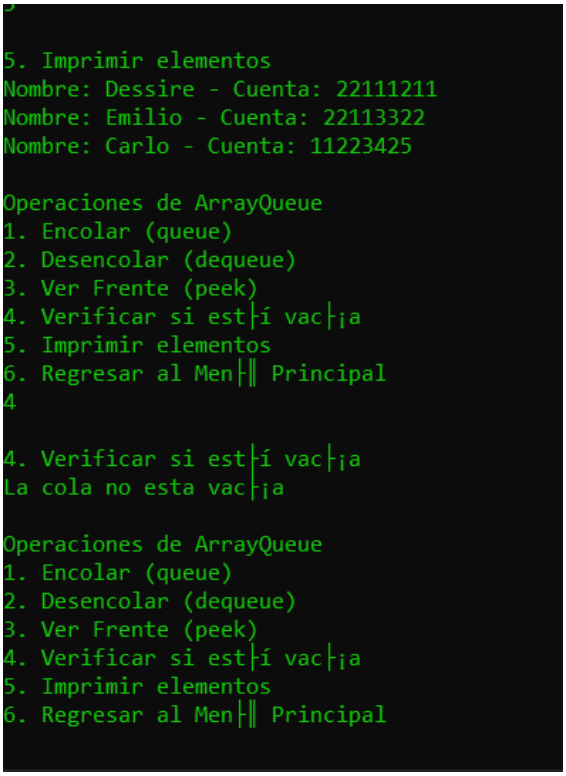
- pone_en_cola(Object*): void

Se le enviará un objeto de tipo alumno para agregar a la cola. No retorna nada.



- vacia(): bool

Retorna verdadero si la cola está vacía, y falso si esta tiene al menos un elemento.



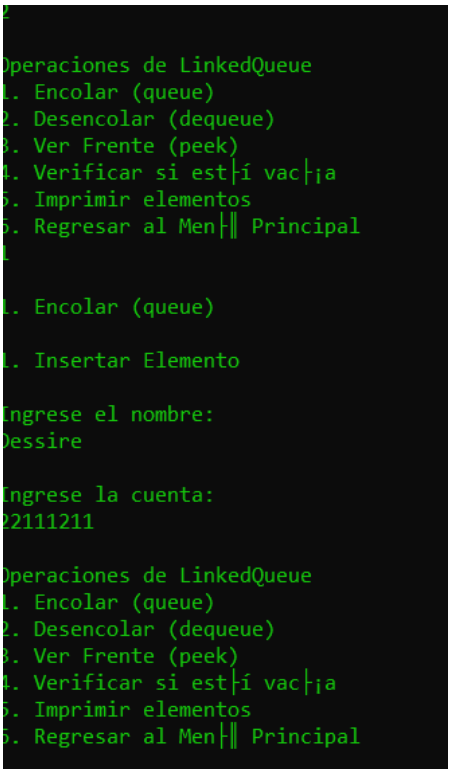
Clase Linkedqueue

- frente(): Object*
- imprime_cola(): void
- saca_de_cola(): Object*
- pone_en_cola(Object*): void
- vacia(): bool

Definición: Clase de Tipo de Dato Abstracto Cola. Almacena el tipo de dato Alumno.

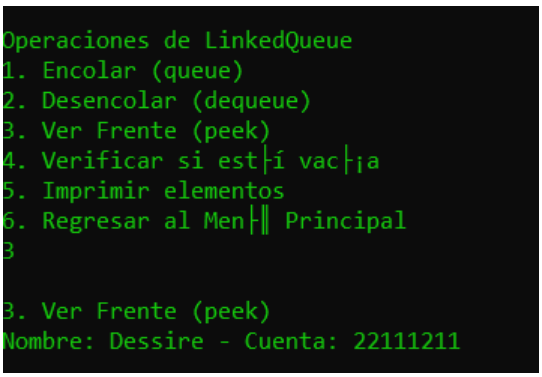
-pone_en_cola(Object*): void

Se le enviará un objeto de tipo alumno para agregar a la cola. No retorna nada.



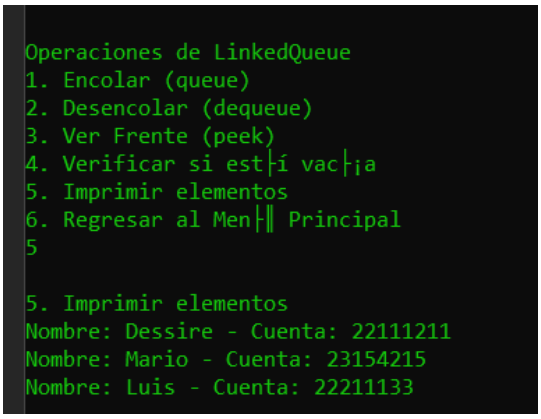
- frente(): Object*

Retorna el objeto de tipo Alumno que encabeza a la cola, es decir, el primero en ser agregado y el primero en ser sacado.



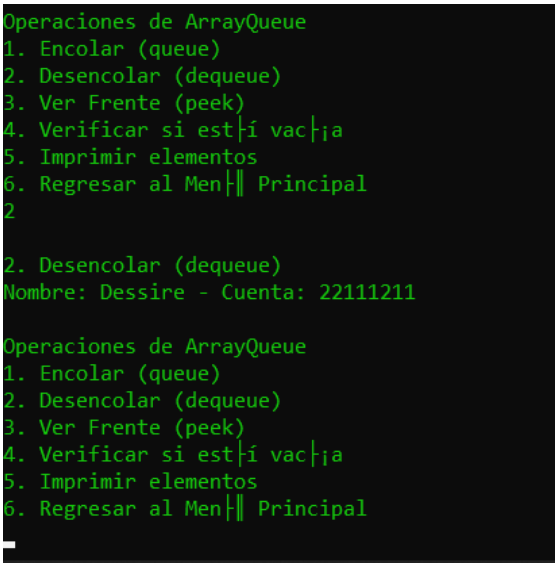
- imprime_cola(): void

Imprime la cola en el orden en el que saldrán los objetos



-saca_de_cola(): Object*

Sacará de cola al objeto que se encuentre en frente. Retorna dicho objeto.



-vacia(): bool

Retorna verdadero si la cola está vacía, y falso si esta tiene al menos un elemento.