Trabajo Practico n°2:

: *Curso de verano 2024 - Algoritmos y Programación II :*

Universidad de Buenos Aires Facultad de Ingenieria

Nombre: Natanael Daniel Brizuela.

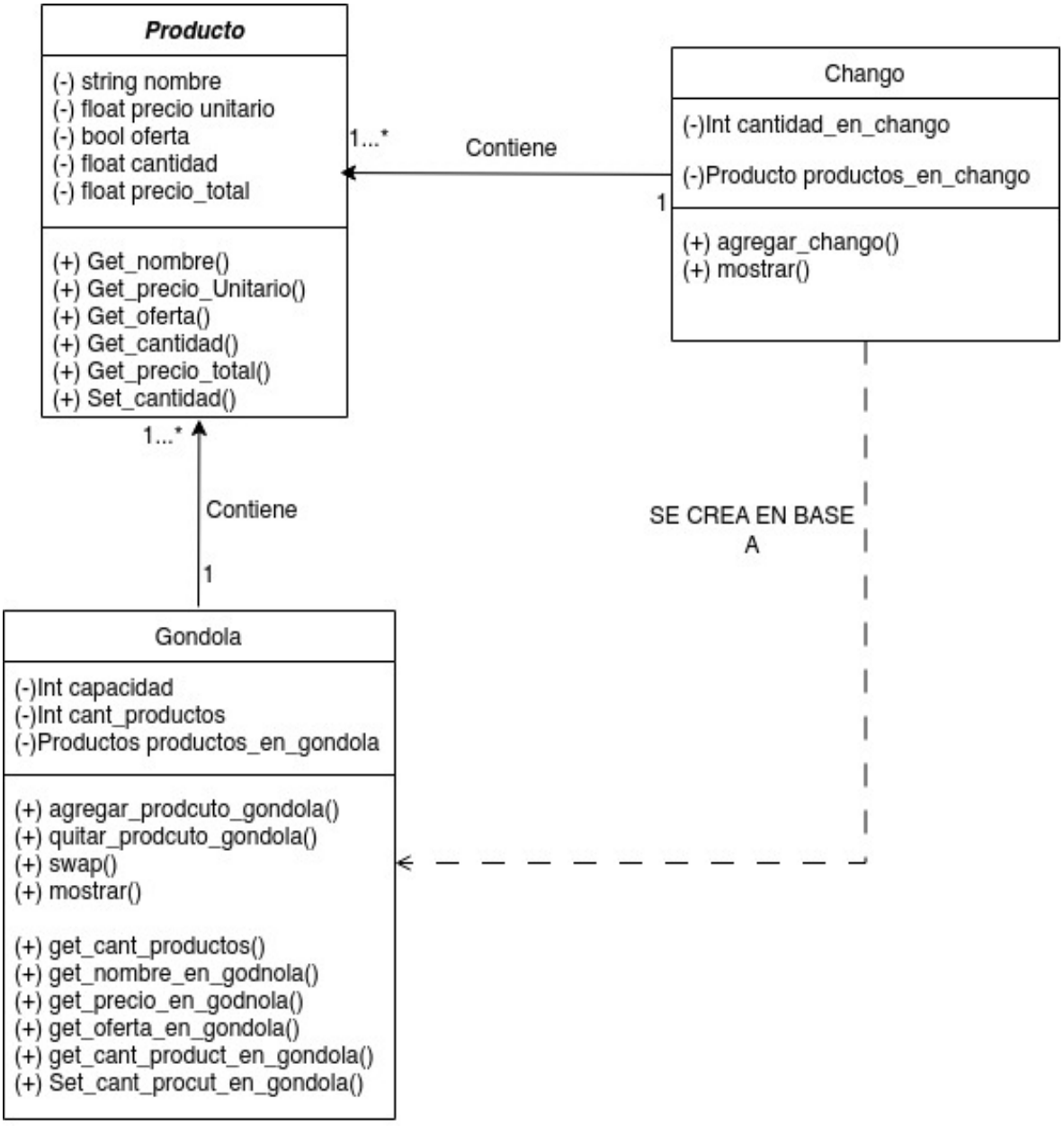
Padron: 110267

DOCUMENTACION

o Diagrama de clases UML

o Descripción de cada TDA, indicando las pre y poscondiciones de cada
- Una de las operaciones.

Diagrama **UML**



:**Descripciones:**

Proposito: La clase Producto representa un producto individual con nombre, precio, oferta ,cantidad de elementos y precio total.

Atributos:

nombre: Nombre del producto (string).

precio_unit: Precio unitario del producto (float).

oferta: Indica si el producto tiene una oferta (bool). TDA: Producto

cantidad: Cantidad de unidades del producto (int).

precioTotal: Precio total del producto si tiene oferta lo aplica (precio * cantidad) - (precio * cantidad * DESCUENTO)(float).

Metodos:

Producto(string nombre, float precio, bool oferta, int cantidad): Constructor con parámetros que inicializa un nuevo objeto Producto. Pre: —. Pos: Inicializa el objeto Producto con nombre, precio, oferta y cantidad.

Producto(const Producto &Prod): Constructor de copia que inicializa una copia de un objeto Producto. Pre: -. Pos: Inicializa una copia de un objeto

Get_nombre(): Devuelve el nombre del producto. Pre: El objeto debe estar correctamente instanciado. Pos: El objeto devuelve el nombre del producto en forma de string.

Get_precios(): Devuelve el precio unitario del producto. Pre: El objeto debe estar correctamente instanciado. Pos: El objeto devuelve el precio unitario del producto en forma de float.

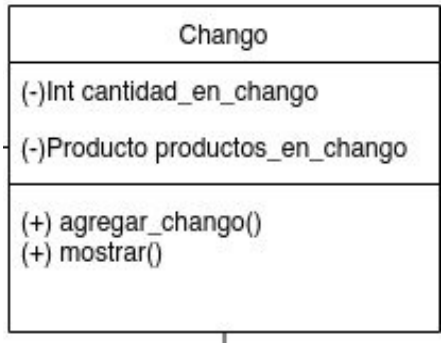
Get_oferta(): Devuelve si el producto tiene una oferta. Pre: El objeto debe estar correctamente instanciado. Pos: El objeto devuelve si el producto tiene una oferta en forma de booleano.

Get_cantidad(): Devuelve la cantidad de unidades del producto. Pre: El objeto debe estar correctamente instanciado. Pos: El objeto devuelve la cantidad de unidades del producto en forma de float.

Get_precio_total(): Devuelve el precio total del producto, aplicando la oferta si está disponible. Pre: El objeto debe estar correctamente instanciado. Pos: El objeto devuelve el precio total del producto en forma de float.

Set_cantidad(int nuev_cantidad): Modifica la cantidad de unidades del producto. Pre: El objeto debe estar correctamente instanciado. Pos: Se define una nueva cantidad de unidades del Producto.

Propósito: La clase Chango representa un carrito de compra que puede contener una colección de productos. Esta clase se crea a partir de la gondola.



TDA: Chango

Atributos:

cantidade_en_chango: Cantidad de productos en el carrito (int).

productos_chango: Vector dinámico que almacena los productos en el carrito (Producto**).

Metodos:

Chango(int cantida_en_chango): Constructor con parámetros que inicializa un nuevo objeto Chango. Pre: El parámetro cantidade_en_chango debe ser mayor o igual a cero. Pos: Crea el objeto Chango con una cantidad de elementos y un vector dinámico nulo.

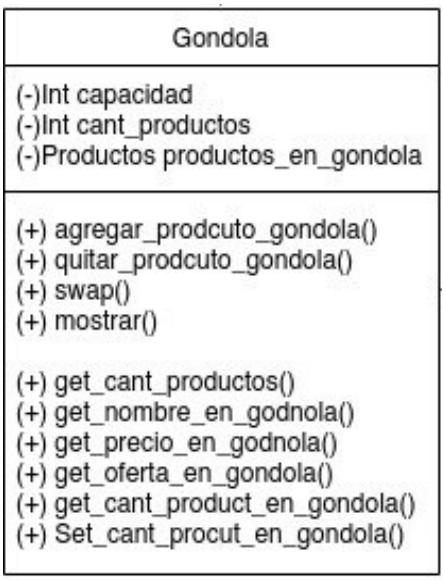
agrear_chango(string nombre, float precio, bool oferta, int cantidad): Agrega un nuevo producto al chango. Pre: Los parámetros deben ser válidos. Pos: Agrega un nuevo producto al vector dinamico productos_chango.

mostrar(): Muestra por pantalla una tabla con información de los productos en el chango. Pre: La clase Chango debe haber sido correctamente inicializada. Pos: Se muestra por pantalla una tabla con la siguiente información de los productos en el chango:

+ Nombre del producto (string).
+ Precio unitario (float).
+ Si está en oferta (S/N) (bool).
+ Cantidad (int).
+ Precio total del producto (float).

~Chango(): Destructor que elimina la memoria dinámica solicitada.

Propósito: La clase Gondola representa una góndola en un supermercado que puede almacenar una colección de productos.



Gondola

Atributos:

Capacidad: Capacidad máxima de productos en la góndola (int).

Cant_productos: Cantidad actual de productos en la góndola (int).

Productos_en_gondola: Vector dinámico que almacena los productos en la góndola (Producto**).

Metodos:

Gondola(int capacidad): Constructor con parámetros que inicializa un nuevo objeto Gondola. Pre: El prarmetro capacidad debe ser un entero positivo. Pos: Crea el objeto Gondola con cero productos y un vector dinámico con una capacidad maxima.

Gondola(const Gondola &P): Constructor de copia que crea una copia de un objeto Gondola.

~Gondola(): Destructor que elimina la memoria dinámica solicitada.

agregar_producto_gondola(const string nombre, float precio, bool oferta, int cantidad): Agrega un nuevo producto a la góndola.

Pre:

- ‘nombre’ debe contener una cadena de caracteres válida que represente el nombre del producto.

- ‘precio’ debe ser un valor numérico positivo que represente el precio del producto.

- ‘oferta’ debe ser un valor booleano que indica si el producto tiene una oferta.

- ‘cantidad’ debe ser un valor entero positivo que represente la cantidad del producto a agregar.

- La góndola debe tener espacio disponible para agregar el nuevo producto.

Pos:

- Se agrega un nuevo producto a la góndola. El nuevo producto se crea con los valores especificados en las variables nombre, precio, oferta y cantidad. Si la góndola estaba llena, se duplica su capacidad.

- Si se crea un nuevo vector el atributo productos_en_gondola de góndola se actualiza para apuntar al nuevo vector de productos.

- La cantidad de productos en la góndola se incrementa en 1.

quitar_producto_gondola(int indice): Elimina el producto en el índice especificado. Pre: El índice debe ser menor que la cantidad de productos en la góndola.

Pos:

- Se elimina el producto en el índice especificado.

- La cantidad de productos en la góndola se decrementa en 1.

- Si la cantidad de productos en la góndola es menor a la mitad de la capacidad, la capacidad se reduce a la mitad.

swap(int indice): Intercambia el producto en el índice especificado con el último producto en la góndola. Pre: El índice debe ser menor que la cantidad de productos en la góndola. Pos: Si el índice está dentro del rango válido, se intercambian el producto en el índice especificado con el último producto en la góndola.

mostrar(): Muestra una lista de todos los productos en la góndola.

get_cant_productos(): Devuelve la cantidad de productos en la góndola.

get_nombre_en_gondola(int indice): Devuelve el nombre del producto ubicado en la posición indice.

get_precio_en_godnola(int indice): Devuelve el precio del producto ubicado en la posición indice.

get_oferta_en_gondola(int indice): Devuelve si el producto en la posicion ‘indice’ tiene una oferta disponible.

get_cant_product_en_godnola(int indice): Devuelve la cantidad de elementos que tiene el producto ubicado en la posicion ‘indice’ de la gondola.

Set_cant_product_en_gondola(int nuevCantidad, int indice): Modifica la cantidad de elementos que tiene el prodcuto ubicado en la posicion ‘indice’ de la gondola.