Trabajo Práctico II

Clases:

Producto

Representa a un tipo de producto con atributos como nombre, cantidad, precio unitario y descuento.

También permite la creación de productos que solo contengan nombre y cantidad.

Además de sus métodos getters y setters (para modificar o devolver sus atributos privados), posee un método con el que informa el precio total.

Métodos:

Producto()	Constructor por defecto, inicializa un objeto con sus atributos por defecto
Producto(`string` nombre, `double` precio_unitario, `int` descuento, `int` cantidad)	Sobrecarga del constructor, se usa para la creación de objetos de la Góndola
Producto(`string` nombre, `int` cantidad)	Sobrecarga del constructor, con este constructor se crean los objetos de la clase Chango
`double` precio_total()	Devuelve el producto entre la cantidad y el precio unitario

Góndola

Una representación de una góndola de supermercado.

Tiene asociada una colección de Productos e implementa métodos para modificar su tamaño (duplicando o dividiendo en dos su capacidad) o para agregar y eliminar los Productos que contiene.

Su capacidad de Productos empieza en 5, pero al ser esta una colección dinámica puede modificarse en tiempo de ejecución. Carga sus productos en base al archivo “productos.txt”.

Métodos:

Gondola()	Constructor de la Góndola. Inicializa una Góndola con capacidad para 5 elementos.
~Gondola()	Destructor. Elimina la memoria alojada en el Heap del vector productos.
`void` agregar_producto (`Producto` &p)	Recibe un producto y lo agrega en la última posición del vector productos. Al excederse por sobre el tope, duplica la capacidad del vector productos.
`void` aumentar_tamanio()	Duplica la capacidad del vector productos. En detalle, crea un nuevo vector de productos con más capacidad y copia los valores originales al nuevo vector. Por último, libera la memoria del vector viejo y actualiza la referencia al nuevo vector de productos.
`void` eliminar_producto()	Elimina los productos de los cuales no haya más stock. Si la nueva cantidad de elementos es menor a la mitad del tope, el vector productos se achica a la mitad. Sigue la misma lógica que aumentar_tamanio() para crear un nuevo vector, liberar memoria y actualizar la referencia de productos.
`void` actualizar_archivo()	Sobreescribe el archivo “productos.txt” con el nuevo stock que se encuentra en la Góndola.

Chango

Representa un chango de compra. Posee una colección de Productos que toma de la Góndola en base a una lista en el archivo “compra.txt”

Puede aumentar su tamaño en una unidad al agregar un Producto. También puede listar su contenido e informar el precio total.

Métodos:

Chango()	Constructor. Inicializa un nuevo Chango vacío con capacidad para un elemento.
~Chango()	Destructor. Libera la memoria eliminando el vector productos.
`void` agregar_productos(`Producto` p, `Gondola` g)	Agrega un Producto al Chango aumentando su tamaño en uno. Al mismo tiempo modifica el stock de la Góndola Si la cantidad disponible en la Góndola es menor que la pedida, se agregan al Chango las unidades disponibles e imprime un mensaje informándolo. En caso de no haber stock, también imprime por pantalla un mensaje que lo indica.
`void` aumentar_tamanio()	Aumenta el tamaño del Chango en uno

Otros:

El formato de los archivos es de la siguiente forma:

productos.txt

producto	precio	descuento (0/1)	stock
arroz	1450	0	25
leche	700	1	52
…	…	…	…

compra.txt

producto	cantidad
arroz	10
leche	20
…	…

El programa puede compilarse con `make`, lo cual produce el archivo de salida `super`

Se omitieron en el diagrama UML los constructores (default), destructores, getters y setters, así como las pre y post condiciones de los métodos getter y setter con el fin de mejorar la legibilidad.