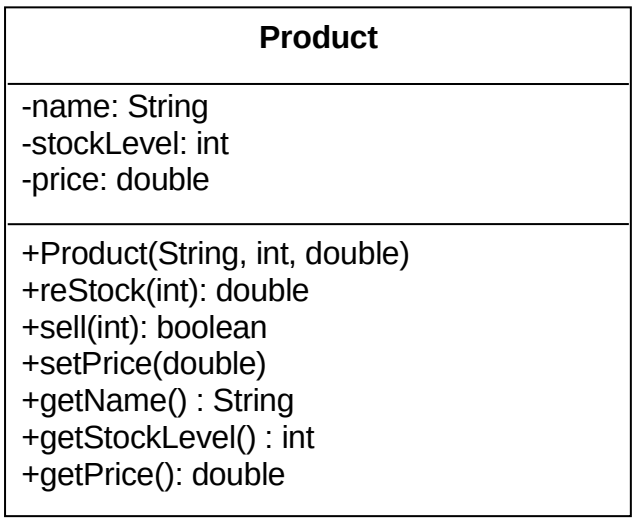
SELF-DIRECTED TASK for CN4001 students

ASSESSED TASK: **18 marks**

Consider the following UML design for a **Product** class:



The **attributes** of this class allow details of a given product (name, stock level and price) to be stored. The **method**s of this class should behave as follows:

+Product(String, int, double)	The *constructor* receives three parameters and uses them to initialise the name of the product, the number of items initially stocked and the price per item respectively.
+buyStock(int): double	Receives the number of new items of stock bought, updates the total value of stock and returns the value of stock bought.
+sell(int) : boolean	Receives the number of items sold, reduces the stock level accordingly (first checking to see if enough items are in stock) and returns true if the sale was successful and false otherwise.
+setPrice(double)	Receives and sets a new price for the product.
+getName() : String **+getStockLevel() : int** **+getPrice(): double**	These methods return the values of the respective attributes.

The Task

a) Implement the **Product** class.

(3 marks)

b) Develop a **ShopApp** program that holds **5 Product objects in an array**, representing products sold in a shop. The **ShopApp** program should provide a allow users to **display** all products, **buy** stock, **sell** stock, **re-price** an item of stock and display the **total value** of all stock in the shop.

Your student number should be displayed at the top of the interface and extra credit will be given for developing a **menu driven interface,** for some **input validation** and the use of **methods**.

(7 marks)

*Note, you can tackle a simpler task to display, buy, sell and re-price a **single Product** object (rather than 5 Product objects in an array) – but this can only earn a maximum of **3 marks**.*

c) Once the **ShopApp** program is complete, fill in the program review table on the following page.

(8 marks)

. THIS IS AN INDIVIDUAL TASK

You will be asked to upload **all the files making up your application** and your **program review table** to Moodle by **Thursday 16th December 2021 (**no later than 4.00pm**).**

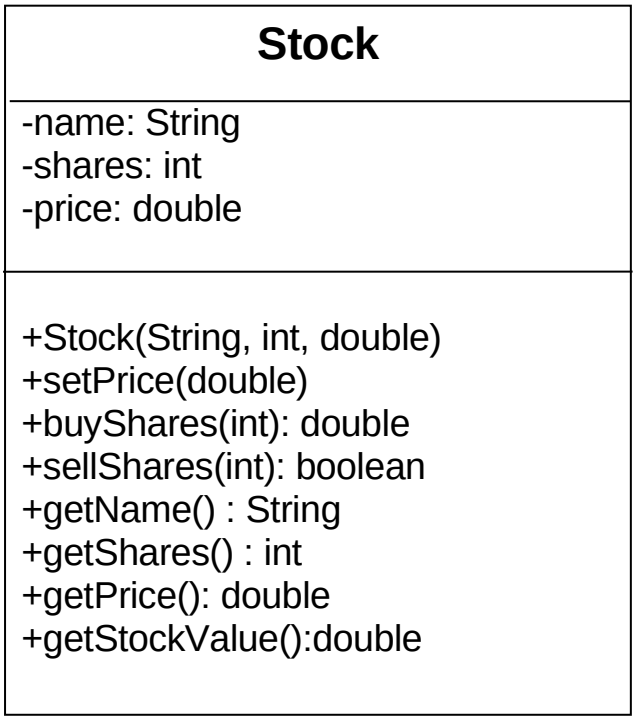
Student ID:		
Programming Concept (1 mark each = 8 marks)	**Example code in ShopApp.java**	**Comment**
variables		
input/output		
choices		
loops		
input validation		
methods		
arrays		
classes/objects		

SELF-DIRECTED TASK for CD4001 students (Apprentices)

ASSESSED TASK: **18 marks**

Consider an application to keep track of a customer's investment in a particular items of stock (for example the stock may be shares in Apple PLC).

The following UML design for a **Stock** class has been developed:



The **attributes** of this class allow details of a given item of stock (name, number of shares and price per share) to be stored. The **method**s of this class should behave as follows:

+Stock(String, int, double)	The *constructor* receives three parameters and uses them to initialise the name of the stock, the number of shares and the price per share respectively.
+setPrice(double)	Receives and sets a new price per share of the given stock.
+buyShares(int): double	Receives the number of new shares to buy in the given stock, updates the total number of shares and returns the total price of the shares bought.

+sellShares(int) : boolean	Receives the number of new shares to sell in the given stock and (if there are sufficient shares to sell) updates the total number of shares and returns true, otherwise returns false.
+getName() : String **+getShares() : int** **+getPrice(): double**	These methods return the values of the respective attributes.
+getStockValue( ) : double	Returns the current value of the shares held in the given stock

The Task

a) Implement the **Stock** class.

(3 marks)

b) Develop a **PortfolioApp** program that holds shares in **5 Stock objects in an array**. The **PortfolioApp** program allow users to **display** all stock holdings, **buy** shares in a given item of stock, **sell** shares in a given item of stock, **re-price** the share price of an item of stock and display the **total value** of all stock held in the portfolio.

Your student number should be displayed at the top of the interface and extra credit will be given for developing a **menu driven interface,** for some **input validation** and the use of **methods**.

(7 marks)

*Note, you can tackle a simpler task to display, buy, sell and re-price a **single Stock** object (rather than 5 Stock objects in an array) – but this can only earn a maximum of **3 marks**.*

c) Once the **PortfolioApp** program is complete, fill in the program review table on the following page.

(8 marks)

THIS IS AN INDIVIDUAL TASK

You will be asked to upload **all the files making up your application** and your **program review table** to Moodle by **Thursday 16th December 2021 (**no later than 4.00pm**).**

Student ID:		
Programming Concept (1 mark each = 8 marks)	**Example code in PortfolioApp.java**	**Comment**
variables		
input/output		
choices		
loops		
input validation		
methods		
arrays		
classes/objects		