# Supermarket simulator

You should create a *supermarket* project. In the *main* method you will write some code, according to the tasks below.

## Part 1 – a product

1. Create a class Product.
2. Every product in your supermarket has a name, a brand name, a price in kroner and a weight in grams. Consider which access modifiers you need for each attribute of your Product class.
3. It should be possible to set all 4 values when creating a new Product object, so 4 parameters should be passed to the constructor (in class Product).
4. In the class Product you should write a method with the following signature:  
    *public string Describe()*  
   that returns a textual description of the product, with all the information in the object (name, brand, price and weight). Consider using this method if you decide to override *ToString* for this class.
5. In the *main* method create 1 instance of Product; call the product “can of beans”, brand should be “International food”, price should be 15.0 and it should weight 500 grams.
6. In the *main* method: print a description of the product you created, using the *Describe()* method.
7. In the class Product write another method with the following signature:  
    *public double PricePerKilo()*  
   that calculates and returns the price/kilogram of the product. In the case of the beans for example, they weight 500 grams, so 0.5 kilograms and their price is 15 kr, so their **price per kilo** is:  
   so your method should return 30 for the beans.
8. In the *main* method: also print the price per kilogram of your product (the beans), using the method *PricePerKilo()* .

## Part 2 – many products

Clone your project and continue with the new cloned one.

In your supermarket there should be many more products, so in the *main* method create a list of products (use a List for that).

1. In the *main* method: create the list, that should be initially empty. Then you will add 5 products of your liking (for example a tomato, apple juice, a package of meat, a t-shirt, etc…). When you create each product, pass all 4 parameters to it, so that each product has a meaningful name, brand, price and also weight.
2. In the *main* method: use a *foreach* loop to print a description of all your 5 products, using the list of products that you just created and the method *Describe()* , or *ToString()* if you have implemented it.
3. Now we would like to know the **total weight** of all the products of the supermarket. But we don’t have access to the weight for the Product objects, because attributes are usually **private**. So you should write a method (in class Product) that returns the weight of a particular Product object. Its signature should be:  
    *public int GetWeight()*and it simply returns the weight of *this* object.
4. In the *main* method: use a **for** loop (not a foreach, just a regular for loop) to print the weight of all products in your list.
5. In the *main* method: use a loop to compute the total value (the total price) of all the products in your list (AKA in your supermarket). [**hint:** your probably need to write a new method in Product class, similar to *GetWeight()* , that returns the price of a product].
6. After you solved point 5) above, also print the **average price** of a product in your supermarket.

## Part 3 – many types of products

Products can be further divided into a few categories, for example:

* packaged food
* fruit & vegetable
* non-food

Change your code according to the following points:

1) Draw a UML diagram for all the classes in the application. Use YUML if you like.   
Then derive 3 classes from your Product class, to represent the 3 categories of products above.  
PackagedFood and FruitAndVegetable products have an expiration date, but NonFood products do not have it. Suggestion: use a DateTime object to represent the expiration date (<https://www.tutorialsteacher.com/csharp/csharp-datetime> )

2) Change the Product class into an abstract class, and modify the main so that it keeps working: for example, look at how you created the products in Part 2, and use the 3 new derived classes instead of Product for every product. You might have to fix the constructors of the new classes, to make it work. The constructor for PackageFood for example, should have 5 parameters, since there is now also an expiration date.

2) Change/override the *Describe* method in the 3 new classes. You should describe a product using its expiration date, if there is one.

3) At the end of your *main*: add some code to print only the products in the list, that are non-food.

4) PackagedFood and FruitAndVegetable products should have a method that checks if the **current date** is close to the **expiration date**:

*public bool IsCloseToExpiration()*

If the distance between the 2 dates is less than 3 days, the method should return true.

Suggestion: use **DateTime.Today** in your method (<https://learn.microsoft.com/en-us/dotnet/api/system.datetime.today?view=net-8.0> ), and look into the *TotalDays* property.

5) Using the method at point 4), change the Describe method for the PackagedFood and FruitAndVegetable classes: if the product is close to expiration, write “ON SALE” before its description, in the Describe method. Remember that overrides are possible only on virtual methods.

More about DateTime and its operations: <https://www.tutorialsteacher.com/articles/get-difference-between-two-dates-in-csharp>