Exercises in OOP programming

Please complete a minimum of 8 'chips'

Exercise 1

Write a class Student with the following properties: StudentID, Name, Address, Tutor, YearGroup. Write accessor methods for each of these properties (these are methods that allow you to retrieve or to set these properties). Remember that the properties should be private and the methods should be public. Declare an instance of a New Student and test that these methods work. The solution could be form-based or console-based.

```
Enter student ID: 1235
Enter student's name: Jenny Hayes
Enter student's address: 13 Harris Drive, Coventry
Enter student's tutor group: SES
Enter student's year group: 12

This is the information about the student.

ID: 1235
Name: Jenny Hayes
Address: 13 Harris Drive, Coventry
Tutor Group: SES
Year Group: 12
```

Exercise 2

Extend your main program code in Exercise 1 to enter and display the data for a class of 5 students. To input the data for more than one student use an array of type Student. However, VB will not allow you to use the keyword NEW in an array declaration – rather tedious of it! Each time you record a new student you will then need to set each element of the array as a new instance of the class, e.g. 'Student(i) = New Student'. You should still be able to do this in a loop.

Exercise 3

Write a class Employee with the following properties: EmployeeID, DateOfBirth, and JobTitle. Write methods to set the employee ID, date of birth and job title of that employee. Write a method to return the employee details and test these for an employee 'FirstEmployee'.

```
Enter employee ID: AB45
Enter employee's date of birth: 13/03/1978
Enter employee's job title: Mechanic

This is the information about the employee.

ID: AB45
Date Of Birth: 13/03/1978
Job Title: Mechanic
```

Exercise 4

Extend your code in Exercise 3 to include two sub-classes: Hourly paid Employee and Salaried Employee. Add properties for the hourly pay or annual pay for these two sub-classes and methods to set the hourly rate or annual pay for each employee. Also add separate methods to display the weekly pay for each sub class, and ensure that your sub-classes inherit

the methods from the base class. Add two new employees of each type to your program and test that you can enter and display both the employee and salary details.

Exercise 5

Modify your code in Exercise 4 so that the method which returns the employee details is 'overrideable'. Then, for each sub-class, replace your methods which return the salary details with a single method which 'overrides' that in the base class and returns both the employee and the salary details.

Exercise 6

A specialist tyre company fits tyres. They require a tyre-fitting system to record each fitting as it takes place. The system should have two classes initially: Tyre and Fitting. The data to be stored about the tyre are: the tyre type, the price of the tyre and the number in stock. The data to be stored about Fitting are: the car registration, the number of tyres fitted and the fitting date.

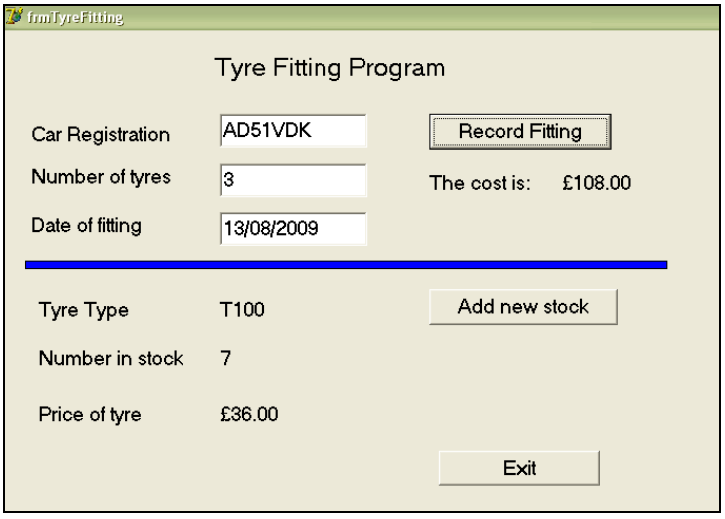
Initially, imagine that there is only one tyre type and that all cars have the same tyre type.

Tyre Type	Number in stock	Price of tyre
T100	10	£36.00

Create a from based OOP system that can input and process the data for a handful of sales. Do not worry about data storage. Array declaration needs to be handled as in Exercise 2. The 'system' should be able to:

- record fitting details: date of fitting, number of tyres and car registration
- calculate the cost of the fitting – use a function in the tyre class and pass a parameter for the number of tyres fitted
- reduce the number in stock by the number of tyres fitted – as above
- add stock: increment the number of tyres for a tyre type by 10.

The following is an example screen.



Extension

Extend your system to work with a number of different tyre types with different prices. Add the facility for all aspects of your system to work with a range of different tyres and prices. Do not worry about storing data.