**5% of your final mark**

**Question 1)**

the formula for calculating your maximum heart rate in beats per minute is 220 minus your age in years. Your target heart rate is a range that is 50-85% of your maximum heart rate.Create a class called Rates.

The class attributes should include the person's first name,last name, year of birth and the current year. Your class should have a constructor that receives this data as parameters. For each attribute provide

a property with set and get accessors. the class also should include a property that calculates and returns the person's age (in years), a property that calculates and returns the person's maximum heart rate and properties that calculate and return the person's minimum and maximum target heart rates.

Write an app that prompts for the person's information, instantiates an object of class Rates and displays the information from that object, including the person's first name, last name, and year of birth. then calculates and displays the person's age in years, maximum heart rate and target-heart-rate range.

**Question 2)**

you will design a "starter" Profile class for a person. The class attributes should include the person's first name, last name, gender, date of birth (consisting of separate attributes for the month, day and year of birth), height (in inches) and weight (in pounds). Your class should have a constructor that receives this data. For each attribute provide a property with set and get accessors. the class also should include methods that calculate and return the user's age in years, maximum heart rate and target-heart-rate range (see previous question ) , and body mass index. Write an app that prompts for the person's

information, instantiates an object of class Profile for that person and displays the information from that object - including the person's first name, last name, gender , date of birth, height and weight. then calculates and displays the person's age in years, BMI (body mass index) , maximum heart rate and target-heart-rate range.

**Question 3)**

create a class called Auto. give the Auto class a constructor that defines properties for model and cylinder.

create get methods for each of the properties. create methods that start the engine, accelerate the car

and break the car. have the methods return a string that shows the method was called.

create classes for 3 cars of your choice. Override the start, accelerate, and break methods.

create an output that will show the car's name, model, and action called.

**Question 4)**

Create an inheritance hierarchy that a bank might use to represent customer's bank accounts. All customers at this bank can deposit(credit) money into their accounts and withdraw (debit) money from their accounts. More specific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts on the other hand, charge a fee per transaction.

Create base class Account and derived classes SavingsAccount and CheckingAccount that inherit from class Account.

Base class Account should include one private instance variable of type decimal to represent the account

balance. The class should provide a constructor that receives an initial balance and uses it to initialize

the instance variable with a public property. The property should validate the initial balance to ensure

that it's greater than or equal to 0.0 .

The class should provide two public methods. Method Credit should add an amount to the current balance. Method Debit should withdraw money from the Account and ensure that the debit amount does not exceed the Account's balance. If it does, the balance should be left unchanged, and the method should display the message "Debit amount exceeded account balance."

The class should also provide a get accessor in property Balance that returns the current balance.

Derived class SavingsAccount should inherit the functionality of an Account, but also include a

decimal instance variable indicating the interest rate (percentage) assigned to the Account.

SavingsAccount's constructor should receive the initial balance, as well as an initial value for the

interest rate. SavingsAccount should provide public method CalculateInterest that returns a decimal

indicating the amount of interest earned by an account. Method CalculateInterest should determine

this amount by multiplying the interest rate by the account balance. (note: SavingsAccount should

inherit methods Credit and Debit without redefining them).

Derived class CheckingAccount should inherit from base class Account and include a decimal instance

variable that represents the fee charged per transaction. CheckingAccount's constructor should receive

the initial balance, as well as a parameter indicating a fee amount. Class CheckingAccount should

redefine methods Credit and Debit so that they subtract the fee from the account balance whenever either transaction is performed successfully. CheckingAccount's versions of these methods should invoke the base-class Account version to perform the updates to an account balance. CheckingAccount's Debit method should charge a fee only if money is actually withdrawn.(note: the debit amount does not exceed the account balance). (hint: Define Account's Debit method so that it returns a bool indicating whether

money was withdrawn. then use the return value to determine whether a fee should be charged.)

After defining the classes in this hierarchy, write an app that creates objects of each class

and tests their methods. Add interest to the SavingsAccount object by first invoking its

CalculateInterest method, then passing the returned interest amount to the object's Credit method.