***1. Define a class that holds information about a mobile phone device: model, manufacturer, price, owner, battery characteristics (model, hours idle and hours talk) and display characteristics (size and number of colors). Define 3 separate classes (class GSM holding instances of the classes Battery and Display).***

***ОБЯСНЕНИЕ:*** Създаваме си нялкоко полета (fields), като е си е направо задължително те да са **private**, а не public! Във Visial Studio има страхотни бързи шорткъти, за по-бързо писане на полета:  
– За създаването на поле с пропъртитата към него напишете **prop**последвано от 2 табулации. Това ще изведе:

*public int MyProperty { get; set; }*

- За създаването на поле с пълния синтаксис за пропърти към него (което ще ви е от полза, ако искате да направите валидация) напишете **propfull**последвано от 2 табулации. Това ще изведе:

*private int myVar;*

*public int MyProperty*

*{*

*get { return myVar; }*

*set { myVar = value; }*

*}*

***2. Define several constructors for the defined classes that take different sets of arguments (the full information for the class or part of it). Assume that model and manufacturer are mandatory (the others are optional). All unknown data fill with null.***

***ОБЯСНЕНИЕ:*** Тук трябва да се упражнят конструкторите, като основното е да се запознаем с [DRY (Don`t repeat yourself)](http://en.wikipedia.org/wiki/Don%27t_repeat_yourself) принципа, според който трябва да се избягва повторението на един и същи код. Поради тази причина използвайки **„this“** надграждаме конструктора public GSM, без да пействаме всеки път един и същи код в тялото му т.е. всеки нов конструктор наследява кода от предишните (преизползваме предишните конструктори).

***3. Add an enumeration BatteryType (Li-Ion, NiMH, NiCd, …) and use it as a new field for the batteries.***

***ОБЯСНЕНИЕ:*** Добавяме обикновена енумерация за батериите („public **enum** BatteryType“), в която да се съдържа някакъв списък от батерии.

***4. Add a method in the GSM class for displaying all information about it. Try to override ToString().***

***ОБЯСНЕНИЕ:*** Всеки обект си има .ToString метод. За да ни връща обаче исканото в условието инфо за даден обект (в случая някакъв телефон примерно), трябва този метод да го overridе-нем и изрищно да му кажем, какво искаме точно да ни връща като информация.

***5. Use properties to encapsulate the data fields inside the GSM, Battery and Display classes. Ensure all fields hold correct data at any given time.***

***ОБЯСНЕНИЕ:*** Капсулацията на данните се получава, чрез get и set на пропъртитата т.е. там може да се направи валидацията на данните, които се подават към полетата (който задължително трябва да са privаtе). Могат да се направят всевъзможни валидации на данните, в зависимост от нуждите. Между другото, според изискванията в задача 2 към настоящия проект, null в конкретния ни случай се води като коректна стойност за някой от полетата ни.

***6. Add a static field and a property IPhone4S in the GSM class to hold the information about iPhone 4S.***

***ОБЯСНЕНИЕ:*** Идеята тук е да се направи статично поле, в което да се съхранява информация за iPhone4S, която може да се ползва всеки път, когато създаваме обект от тип GSM.IPhone4S (подобно на Int32.MaxValue, което също е static field в структурата Int32).

***7. Write a class GSMTest to test the GSM class:***

***- Create an array of few instances of the GSM class.***

***-Display the information about the GSMs in the array.***

***- Display the information about the static property IPhone4S.***

***ОБЯСНЕНИЕ:*** Трябва да направим масиви от различни стойности, с които да тестваме конструктора GSM. Подаваме му стойностите чрез един for цикъл и гледаме дали валидацията, която написахме по-горе ще сработи.

***8. Create a class Call to hold a call performed through a GSM. It should contain date, time, dialed phone number and duration (in seconds).***

***ОБЯСНЕНИЕ:*** Cъздаваме си нов клас Call, в който подобно на класа GSM си съдаваме privаte полета и конструктор Call.

***9. Add a property CallHistory in the GSM class to hold a list of the performed calls. Try to use the system class List<Call>.***

***ОБЯСНЕНИЕ:*** Cлагаме ново поле (private List<Call> callHistory) и ново пропърти (public List<Call> CallHistory {}) в класа GSM.

***10. Add methods in the GSM class for adding and deleting calls from the calls history. Add a method to clear the call history.***

***ОБЯСНЕНИЕ:*** От нас се иска да добавим три метода, който да добавят (AddCall), трият (DeleteCall) и изчистват call хронологията (ClearCallHistory).

***11. Add a method that calculates the total price of the calls in the call history. Assume the price per minute is fixed and is provided as a parameter.***

***ОБЯСНЕНИЕ:*** Comming soon.

***12. Write a class GSMCallHistoryTest to test the call history functionality of the GSM class.***

***– Create an instance of the GSM class.  
– Add few calls.  
– Display the information about the calls.  
– Assuming that the price per minute is 0.37 calculate and print the total price of the calls in the history.  
– Remove the longest call from the history   
and calculate the total price again.  
– Finally clear the call history and print it.***

***ОБЯСНЕНИЕ:*** Comming soon.

**Кодът към всяка една от задачите можете да видите**[**тук**](https://github.com/ipobornik/TelerikSoftwareAcademy/tree/master/OOP%20CSharp%20Telerik%20course%202014%20Homeworks/OOP%20Defining%20Classes%20Part%20One/DefiningClassesPart1)**.**