

EXERCISE 1

The tasks in the first part of this practical are step by step instructions as to how an object is created in C#. This is followed by a number of tasks for you to complete by applying the knowledge that you have previously gained.

TASK 1: Creating a mobile phone program

You go into a mobile phone shop to buy a mobile phone – what options do you have?

- Pay as you go or pay monthly
- Type of phone

Let's say you decide to go pay as you go how is the phone then unique to you?

- You are allocated a phone number

How is the account activated?

- The account balance is set to zero

What functions can be carried out from an account perspective?

- Top up account with credit
- Make a call – reduces credit
- Send a text – reduces credit
- Get balance

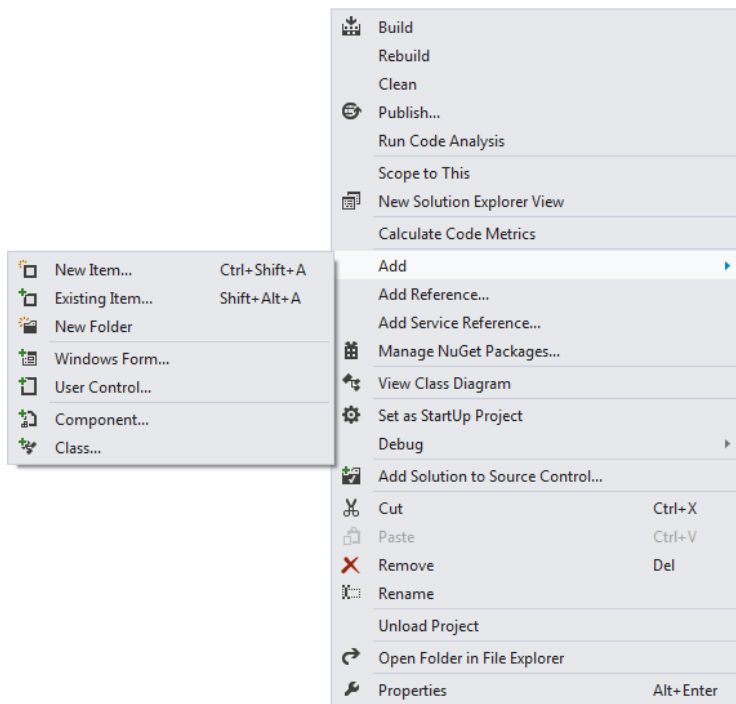
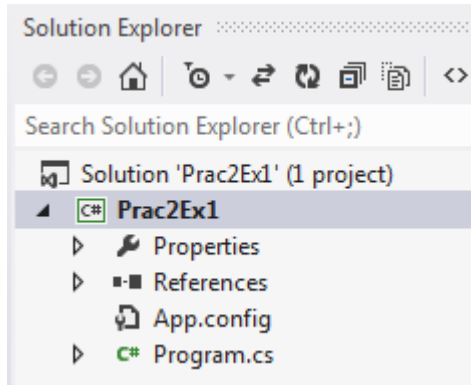
There are other functions that you could apply but let's start to code this part of the solution.

Step 1:

Run Visual Studio. It will open to the start page from which you can create a new project. Select `File>New>Project` and choose a `Visual C# Console Application`. Name this project `Prac2Ex1`.

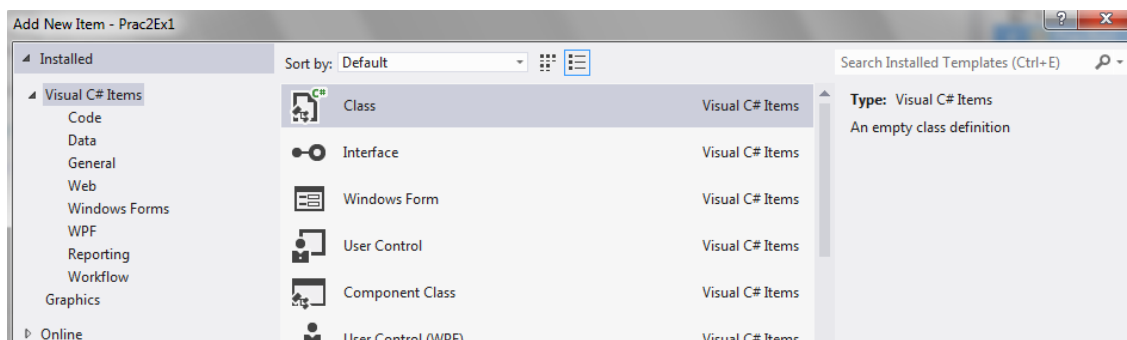


Step 2: From the Solution Explorer on the right hand side of the screen select the project file, shown here as `Prac2Ex1`. Right click on it which will open a new menu. Select `Add>Class`.



Step 3:

A new window will open which will allow you to name the new class that you are creating. Name the class mobile and click Add.



Step 4:

Type the following code:

```
class Mobile
{
    private String accType, device, number;
    private double balance;
    public Mobile(String myAccType, String deviceType, String mobileNo)
    {
        accType = myAccType;
        device = deviceType;
        number = mobileNo;
        balance = 0;
    }
}
```

This function accepts the account type (either PayAsYouGo or Monthly), the name of the chosen device (e.g. iPhone 5S) and a mobile number as a string. It then sets each of the values into variables which can be called later. Let's have a look at that now.

Step 5:

We want to print out account information, but we may not want to do this all at the same time. For example there may be instances where we only want the balance displayed or just the mobile phone number. So we will set each of these functions up separately.

Take a new line after the closing curly bracket of the Mobile public function (but before the closing curly bracket of the Mobile public class definition) and type the following:

```
public String getNumber()
{
    return number;
}

public String getAccType()
{
    return accType;
}

public String getDevice()
{
    return device;
}

public double getBalance()
{
    return balance;
}
```

Step 6:

Save your work (CTRL+Shift+S)

Step 7:


This class sets up the values a “mobile” will have and has begun to identify some of the functions that will be possible on it but we would be best to test it. To do this we will navigate back to the Program class which contains our main method.

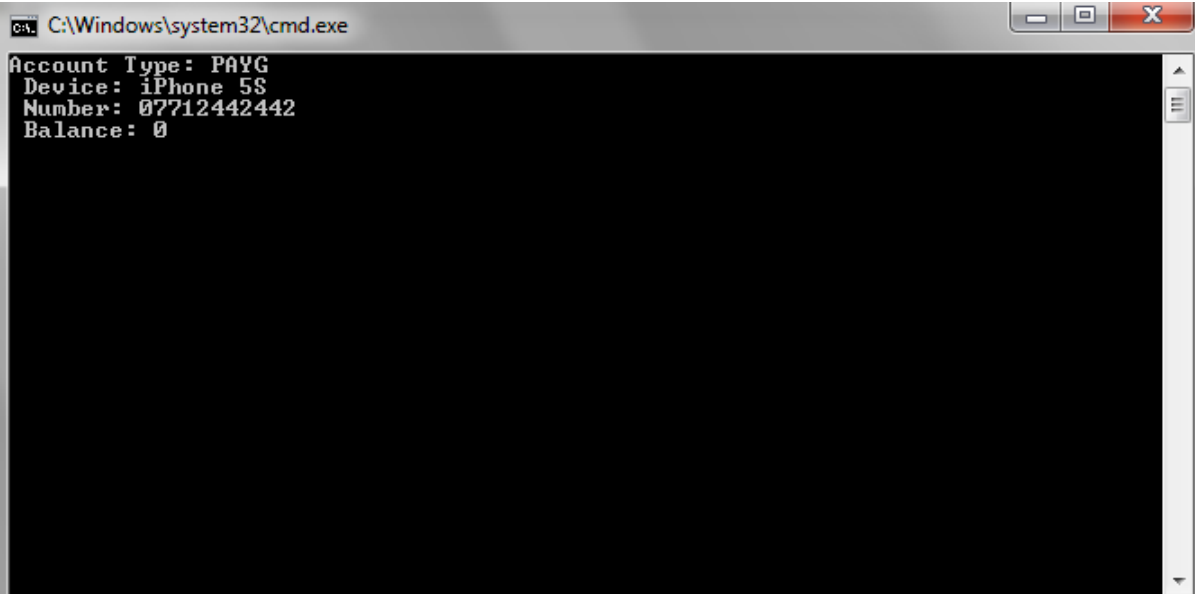
Next, using the Solution Explorer, double click on `Program.cs` this will open it. Type the following code inside the brackets of the main method:

```
static void Main(string[] args)
{
    Mobile jimMobile = new Mobile("PAYG", "iPhone 5S", "07712442442");
    Console.WriteLine("Account Type: " + jimMobile.getAccType() + "\n Device: " + jimMobile.getDevice() + "\n Number: "
        + jimMobile.getNumber() + "\n Balance: " + jimMobile.getBalance());

    Console.Read();
}
```

Step 8:

Save your work (Ctrl+Shift+S) . Run the program by clicking the start button on the toolbar  or use the shortcut Ctrl+F5. A console window should open. Example output is shown below:



```
C:\Windows\system32\cmd.exe
Account Type: PAYG
Device: iPhone 5S
Number: 07712442442
Balance: 0
```

Step 9:

Now let's go back to the Mobile class and add some more functions, that we noted at the start of this practical, such as:

- Top up account with credit
- Make a call – reduces credit
- Send a text – reduces credit

Click on the Mobile.cs (both Mobile.cs and Program.cs should remain open and you can toggle between the two).

Before the last closing curly bracket of the class add the following code:

```
public void addCredit(double amount)
{
    balance += amount;
}
public void makeCall(int minutes)
{
    balance -= minutes * CALL_COST;
}
public void sendText(int numOfText)
{
    balance -= numOfText * TEXT_COST;
}
```

Step 10:

You will notice that we use two variables here CALL_COST and TEXT_COST these will need to be declared and initialised at the top of the class with the other variables.

These variables will not be changed during the operation of the program so we can set them to be constants using the const keyword shown below.

Set CALL_COST to equal 0.245 and TEXT_COST to equal 0.078.

Here is the entire code for `Mobile.cs`, check that you have the same code:

```
class Mobile
{
    private String accType, device, number;
    private double balance;
    private const double CALL_COST = 0.245;
    private const double TEXT_COST = 0.078;

    public Mobile(String myAccType, String deviceType, String mobileNo)
    {
        accType = myAccType;
        device = deviceType;
        number = mobileNo;
        balance = 0;
    }

    public String getNumber()
    {
        return number;
    }
    public String getAccType()
    {
        return accType;
    }
    public String getDevice()
    {
        return device;
    }
    public double getBalance()
    {
        return balance;
    }
    public void addCredit(double amount)
    {
        balance += amount;
    }
    public void makeCall(int minutes)
    {
        balance -= minutes * CALL_COST;
    }
    public void sendText(int numOfText)
    {
        balance -= numOfText * TEXT_COST;
    }
}
```

Step 11:

Navigate to the `Program.cs` tab and add this code after the last `Console.WriteLine` statement to test the functions we just implemented.

```
jimMobile.addCredit(10.00);

Console.WriteLine("Credit successful \n\n Mobile Number: " + jimMobile.getNumber() + "\n Balance: "
    + jimMobile.getBalance());

jimMobile.makeCall(15);

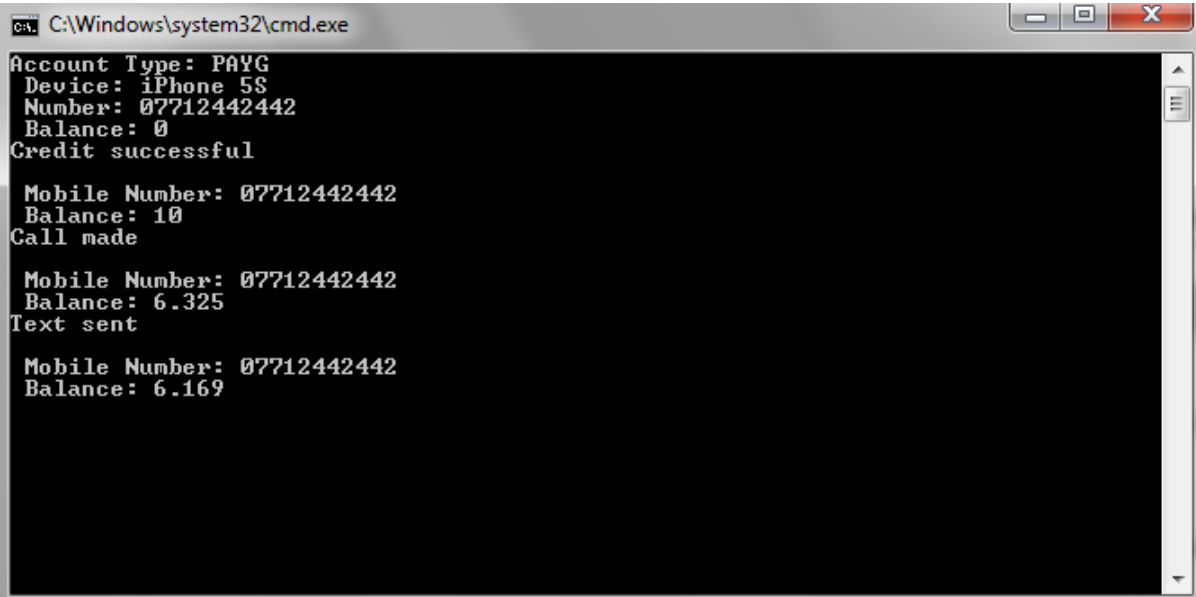
Console.WriteLine("Call made \n\n Mobile Number: " + jimMobile.getNumber() + "\n Balance: "
    + jimMobile.getBalance());

jimMobile.sendText(2);

Console.WriteLine("Text sent \n\n Mobile Number: " + jimMobile.getNumber() + "\n Balance: "
    + jimMobile.getBalance());

Console.Read();
```

Step 12: Save your work (Ctrl+Shift+S) and run the program. Example output is shown below.



```
C:\Windows\system32\cmd.exe
Account Type: PAYG
Device: iPhone 5S
Number: 07712442442
Balance: 0
Credit successful

Mobile Number: 07712442442
Balance: 10
Call made

Mobile Number: 07712442442
Balance: 6.325
Text sent

Mobile Number: 07712442442
Balance: 6.169
```

TASK 2: EXTEND THE MOBILE CLASS

In this part of the practical you will add more functionality to the mobile class and also introduce a few more tests to ensure that what you are coding in the class actually executes as you think it should.

Step 1

In the Program class add lines of code to create another mobile user. Here is a hint:

```
Mobile <nameOfVariable> = new Mobile ("Account Type", "Device Type",  
"Mobile Number");
```

Also you will only be editing the Program class; the Mobile class does not need to be amended for this part of the practical.

Step 2

Add credit of your choice to this new mobile user and then make a call (add the minutes of your choice) and finally send one or more text messages. Remember to output the change to the balance after each call to Mobile class to ensure that the output is what you expect.

Step 3

Consider one more method that the mobile phone class could benefit from and program it in the Mobile class. Then in the Program class test this function to ensure it is working appropriately.

TASK 3: CREATE A SQUARE CLASS

Step 1:

Create a new project (C# Console Application) and call it Prac2Task3.

Step 2:

Create a new class in your project and name it Square. Open the new class and type the code shown below.

```
class Square  
{  
  
    private int sideLength;  
    private int area;  
  
    public Square(int initialLength) {  
        sideLength = initialLength;  
        area = sideLength * sideLength;  
    }  
  
    public int getArea() {  
        return area;  
    }  
  
    public void grow() {  
        sideLength = 2 * sideLength;  
    }  
}
```


Step 3:

Open the Program class which contains the main method and type this code between the set of curly brackets.

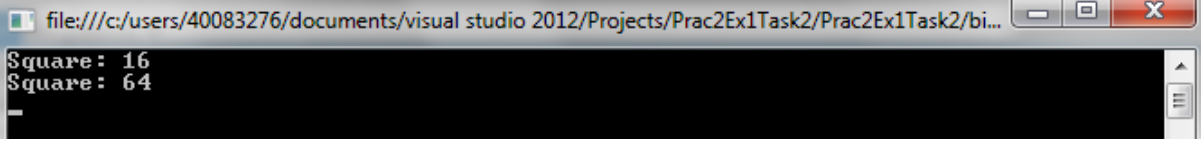
```
static void Main(string[] args)
{
    Square mySquare = new Square(4);
    Console.WriteLine("Square: " + mySquare.getArea());
    mySquare.grow();
    Console.WriteLine("Square: " + mySquare.getArea());
    Console.Read();
}
```

Step 4:

Execute the program and examine the results.

- The aim of the program is to work out the area of a square and then increase (or grow) the area of that square.
- Try to determine why the above program does not run as you would expect.

Shown below are the results we wish to display.



```
file:///c:/users/40083276/documents/visual studio 2012/Projects/Prac2Ex1Task2/Prac2Ex1Task2/bi...
Square: 16
Square: 64
```

TASK 4: PROGRAM TO SWAP THE VALUES OF TWO VARIABLES**Step 1:**

Create a new project and type the following code into the main method in Program.cs.

```
static void Main(string[] args)
{
    // declare and initialise variables
    int first = 3;
    int second = 4;

    // write out initial values
    Console.WriteLine("First variable: " + first + " Second variable: " + second);

    //swap values
    first = second;
    second = first;

    // Write out new values
    Console.WriteLine("First variable: " + first + " Second variable: " + second);

    Console.Read();
}
```

The expectation for this program is that the variable first would contain the value of the variable second i.e. 4 and that variable second would contain the value of the variable first i.e. 3.

If you run the program you will see that this is not the case. Now write a version of the program that does work as expected.

TASK 5: A CIRCLE CLASS

What are the syntax errors in the following program?

How many can you spot before typing this into Visual Studio?

```
public class Circle {  
    public static void main(String[] args) {  
        double radius = 4;  
        double area; // to hold area  
        Double circumf; // to hold circumference  
  
        // do the calculations  
        Console.WriteLine( "Area: " + area + \nCircumference: "  
+ circumf);  
        circum = 4 * Math.PI * radius;  
  
        // Class Math provides methods for common mathematical  
        functions e.g. PI  
        area = Math.PI * raduis * radius * radius  
        //output answers  
    }  
}
```

When you work out what the syntax errors are type the code into a class in Visual Studio and run it. Then fix all of the errors to determine if it provides you with the correct answers.

TASK 6: SWAP THE VALUES OF THREE VARIABLES

Write a program that stores three numbers into variables num1, num2 and num3. It should then swap them around so that num1 holds the value originally in num3, num2 holds the value originally in num1 and num3 holds the value originally in num2.

It should print the contents of the variables before and after swapping them, to demonstrate that it works.

EXERCISE 2

TASK 7: AN EMPLOYEE CLASS

Implement a class called Employee. This class should have a name (of type string) and a salary (of type double). You will need a constructor with two arguments:

- `public Employee (String employeeName, double currentSalary)`

You will also need the following methods:

- `getName` – this should return the name of the employee
- `getSalary` – this should return the salary of the employee
- `raiseSalary` – this should raise the employee's salary by a certain percentage

To test this class you will need to use the `Program.cs` class which contains the main method. This class should test all of the methods.

TASK 8: A STUDENT CLASS

Implement a class Student with the following properties.

- A student has a name
- A score for an examination

You should program the following methods:

- `getName` – this should return the student's name
- `addExam` – this should add an exam mark to the student
- `getTotalMark` – this should return the total marks gained by the student
- `getAverageMark` – this should calculate the average mark gained by the student (for this you will need the total number of exams taken)

Use the `Program` class to test all methods in the `Student` class.

Amend the `Student` class to remove a mark from the student's results. This will assume that you have already added a mark to the student (in later weeks we will look to checking this first).

TASK 9: CONVERTING CELSIUS TO FAHRENHEIT

Write a program to compute the Fahrenheit equivalent of the temperature 20° Celsius, and write the answer to the console output window in the form of:

The Fahrenheit equivalent of 20 degrees Celsius is 52.00

Note that the conversion may be done using the formula:

$F = (9/5) * C + 32$: where F is the required Fahrenheit temperature, and C is the Celsius temperature.

EXERCISE 3

TASK10: IMPLEMENTING A CAR CLASS

Implement a class `Car` with the following properties which you will set in the constructor.

- fuel efficiency which is measured in miles per gallon or mpg
- fuel in the tank (in litres) with an initial level of 0

You will develop a program to add fuel to the car, drive a distance in miles and estimate the cost of the journey.

You should program the following methods

- `drive` – this should simulate driving the car for a certain distance in miles. (HINT: fuel used = distance driven divided by mpg). The gallons used for the journey have to be converted into the litres equivalent.
- `convertToLitres` – a method that will take a parameter of gallons and convert that value to litres. This can be done by multiplying the value by 4.546
- `getFuel` – this should return the amount of fuel in the tank
- `addFuel` – this should add fuel to the tank
- `fuelCost` – the cost of fuel in pennies either added to the tank or used for driving the distance passed to the drive method.
- For the purpose of this program the cost per litre of petrol is approximately 135.9 pence.
- `getCost` – a method that returns the cost of fuel in pounds and pence.

For the purposes of this class you can assume that the drive method is never called with a distance that consumes more than the available fuel.

Write appropriate code in the Program class to test all of the methods that you have implemented.