**CMP1127M – Programming & Data Structures**
**Data Structures Workshop 1 – Building Complex Objects**

**Outline:**

During the last semester's Lectures we looked at using different primitive types, and some abstract types such as the String and Arrays.

In this week's workshop we are going to look into a little more programming with C# to get us ready for the data structures work starting in the next couple of weeks. We are going to develop a recordSet class.

**Task 1: Create a new Project**

We are going to create a class called Person that can store the details of a person (such as the contact on your phone). Create a new console project. Save the project to your home directory or Pen Drive.

Inside the class Project add the following code (or just **copy** and **paste**):

```
public class Person
{
    //Add variables here

    public Person()
    {
        //Default Constructor
    }

    //Add methods here
}
```

You can see from the code above that there is something new we've not used before '`public Person()`' this is what we call a constructor. This is used by the computer to build the actual object of the class (i.e. the house from the blueprint) with default settings and variables, we'll look at this in more detail later.

You will see that we have no data in this class and we have no functions to run.

**Task 2: Adding variables and functions**

In order to make a complex recordSet or indeed create a class to represent a Person, we need to have some data to store about that person. In your class you need to create some storage variables that will represent the data of a Person. We need to store, name (first and last), address, DOB, Phone number, and others of your choosing. Add the following code your class:

```
    //Add variables here
    public String firstName;
    public String lastName;
    public String DOB;
    public String address;
    public String telNumber;
```

You can add the following code inside the Main so you can inspect the output:

```
static void Main(string[] args)
{
   Person New_Person = new Person();

   Console.WriteLine(New_Person.firstName + " ");
   Console.WriteLine(New_Person.lastName + " ");

   Console.WriteLine("Press Enter to terminate...");
   Console.Read();

}
```

Now when you compile this code and create a new instance on the printout you will see we have several variables with *null* value.

These are all currently `null` so we need to create some methods that allow us to modify these variables. You can add code as shown:

```
    //Add methods here
    public void setFirstName(String aFirstName)
    {
        firstName = aFirstName;
    }
```

Now when you compile and create an instance, you can set say the *firstname* by adding the line `New_Person.setFirstName("George");` in the Main. If you execute this, you will see the variable has changed. Create new methods for all the variables you need to set for your `Person()`.

**STOP: Once you have done this, have a demonstrator look at your work before you proceed.**

**Task 3: Changing the Data Types**

Now you should be able to set the values off all the variables for your record or Person(). Please continue with the following questions:

Q1. Set the Telephone number of your Person() to "FRED". What happens?
Q2. Set the DOB of Person() to "FLINTSTONE". What happens?
Q3. Why does this happen? Discuss with a demonstrator.

What you now need to do is modify your program so that the data the variables are representing are more appropriate to the Data that you are imputing. For example, a telephone number only has numbers, would an Integer be more appropriate here? Try this.

What happens? You get an error…

Is there something we can do about this? Is there a more appropriate data type? Discuss with a demonstrator if you need to. With a different data type, do you notice any other issues?

**Task 4: Creating the RecordSet**

We now have our class `Person`, we can make as many of these `Person` objects as we wish setting different details for each one. Go ahead and make 5 or so of them (`Person1` , `Person2` , etc.).

Now you need to make a new class that is going to represent your `RecordSet`, call this class `Contacts`.

Add the following code after the `Person` class:

```
public class Contacts
{
    // Add variables here
    public Person[] myContacts = new Person[5];
    public int count = 0;

    public Contacts()
    {
        // Default constructor
    }

    // Add methods here
}
```

We now have a RecordSet, look closer at this line of code:

```
public Person[] myContacts = new Person[5];
```

This gives us a variable called `myContacts` with a data type of our class `Person`, we say this is a `RecordSet` because we say the variable can hold 5 `Persons` using the `[5]` syntax.

We now have to create a method that allows us to Add Persons to our set. Add the following code in the usual place:

```
    public void addContact(Person aPerson)
    {
        myContacts[count] = aPerson;
        count++;
    }
```

You should now be able to create several objects of `Person()` and add some details, then create one object of `Contacts()` in the `Main`.

```
Person New_Person1 = new Person();
Person New_Person2 = new Person();
Contacts New_Contacts = new Contacts();
…
…
…
New_Contacts.addContact(New_Person1);
…
…
```

By either printing out or by using the debugger you can see the content of the `New_Contacts`.

**Extra Tasks:** Once you have completed the above tasks, you can attempt the ones below.

**T1:** Modify your program so that the variables for your `Person()` will only accept the correct data. i.e. `firstName` and `lastName` will only accept letters and `telNumber` will only accept numbers but must include the '0' at the front.

**T2:** What happens if you add more than 5 `Person()` details to your `Contacts()`? Modify the code so that when the contact list is full, you cannot add any more.

**T3:** Add a new method to `Contacts()` so that you can delete a `Person()` from the list.