

EXERCISE 1

In this tutorial, you will practice how to create objects based upon a farmyard scenario. You will learn how to create a blueprint class for each type of animal on the farm and a Tester class which will allow you to create your objects and call / assign data to your variables. You will learn to do this without a constructor.

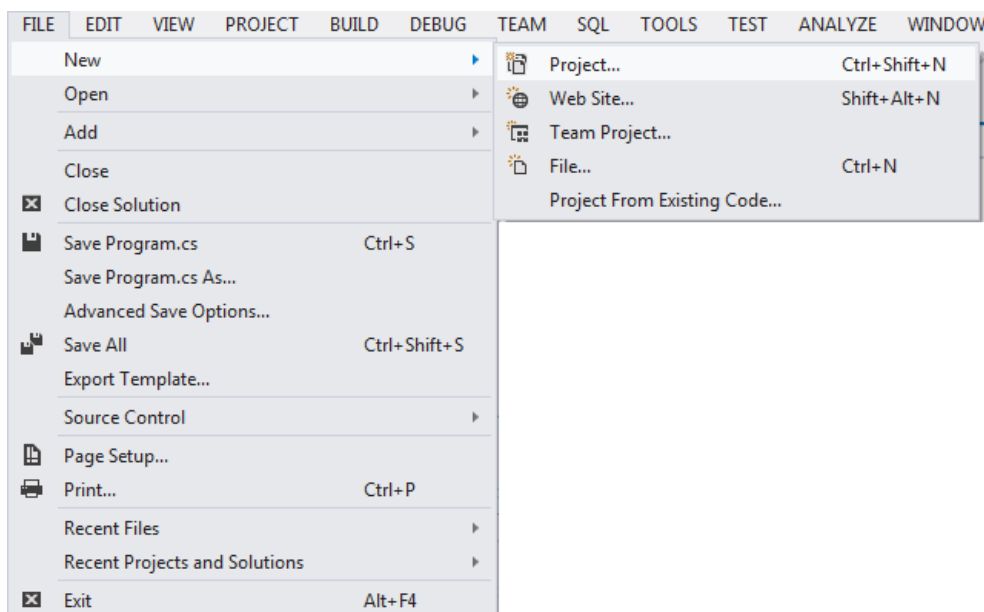
Your farm will have 2 dogs, 2 cats and 2 horses. You will create the following files:

Dog.java	Cat.java	Horse.java	FarmTester.java
variables additional methods	variables additional methods	variables additional methods	Main method Create your objects and pass in data Print out the stored details on each object.

TASK 1: CREATING OBJECTS WITHOUT THE USE OF A CONSTRUCTOR

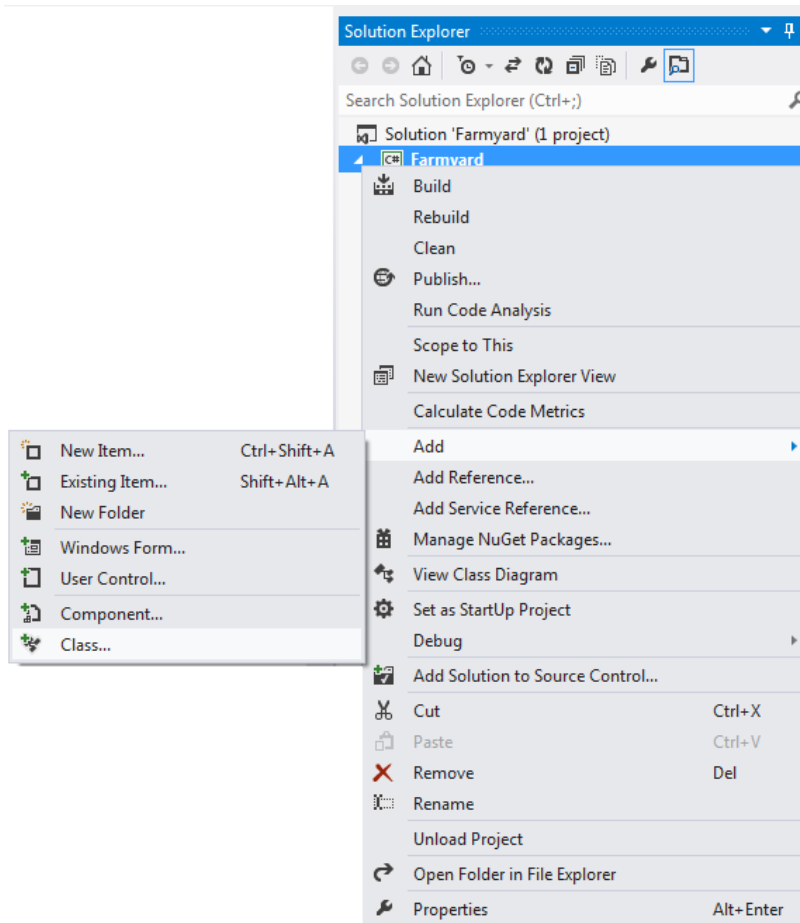
Step 1:

Open Visual Studio. Select `File>New>Project`. Choose `Visual C#` and `C# Console Application`. Name the project `Farmyard` and select finish.

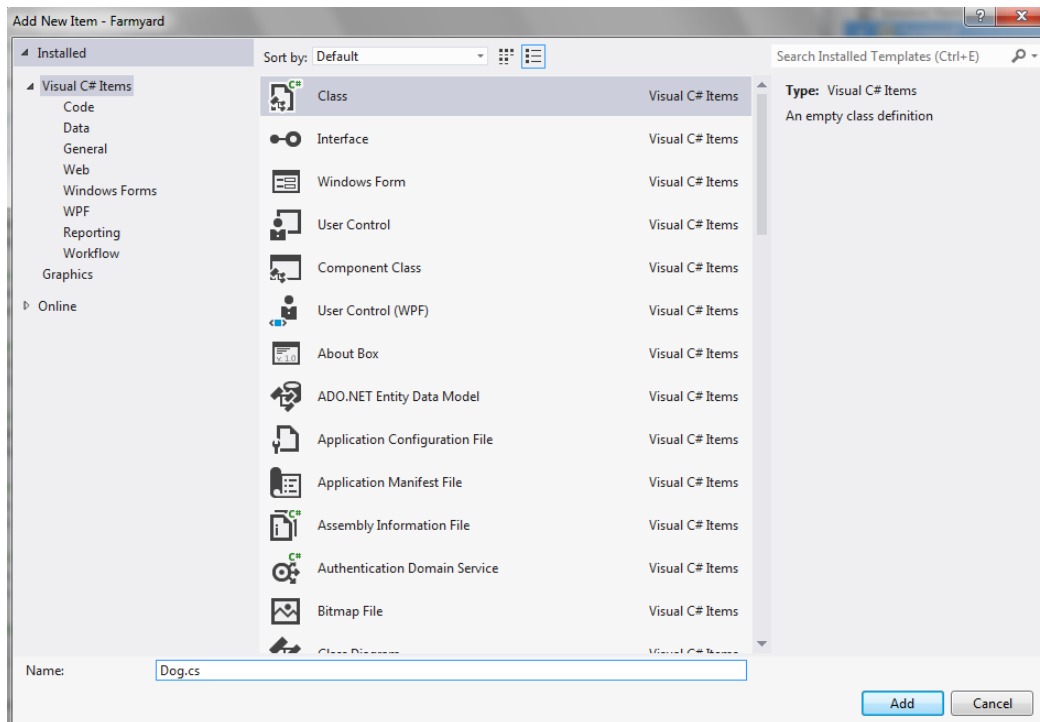


Step 2:

Next, you will add the class files to the Solution, by right clicking on the Solution name in the Solution Explorer and going to Add-> New-> Class



Call this Dog.



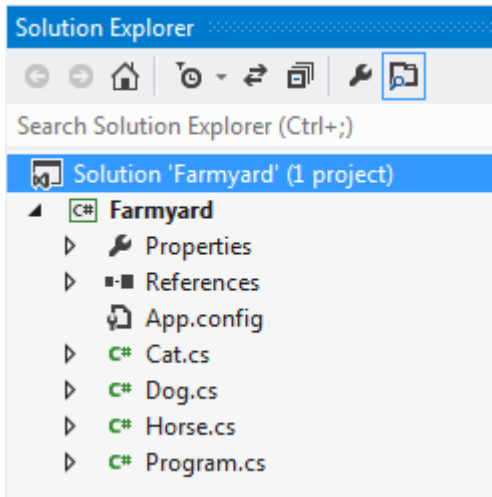
This class will act as a blueprint for and will allow you to create objects of type `Dog`. You do not need to have a main method here as this is a blueprint class and within this class. Within this class, you will have your variables and methods that every dog object will have. Once the class is created, it should look like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Farmyard
{
    public class Dog
    {

    }
}
```

Repeat step to create a blueprint class for your `Horse` and `Cat` objects.



Your `Program` class will be your tester file and will be the one that contains the main method. This will be the file where you will create your objects and also where you will call and pass data to your methods. It will be the file that you will compile and run. It will look like this initially.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Farmyard
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Step 4:

Inside the `Dog`, `Horse` and `Cat` classes you must declare public variables which will be the fields of the class e.g. the dog/cat/horse's name, age and the colour. Initially you will create your dog, horse and cat objects without the use of a constructor by accessing the public variables you create here.

<code>public String catName;</code>	<code>public String dogName;</code>	<code>public String horseName;</code>
<code>public String catColour;</code>	<code>public String dogColour;</code>	<code>public String horseColour;</code>
<code>public int catAge;</code>	<code>public int dogAge;</code>	<code>public int horseAge;</code>

Step 5:

Next, within the dog, horse and cat class you will create a method that will output the sound that the cat/horse/dog will make onto the console.

<pre>public void makeNoise(){ Console.WriteLine("Meooww"); }</pre>	<pre>public void makeNoise(){ Console.WriteLine("Woof"); }</pre>	<pre>public void makeNoise(){ Console.WriteLine("Neigh"); }</pre>
--	--	---

Within this section, you must also create a print method that will allow you to output the contents of the variables for each object that you create onto the screen.

<pre>public void printDetails(){ Console.WriteLine("Cat Details are as follows: \nCat Name: " + catName + "\nCat Colour: " +catColour +"\nCat Age: " + catAge); }</pre>	<pre>public void printDetails(){ Console.WriteLine("Dog Details are as follows: \nDog Name: " + dogName + "\nDog Colour: " +dogColour + "\nDog Age: " + dogAge); }</pre>	<pre>public void printDetails(){ Console.WriteLine("Horse Details are as follows: \nHorse Name: "+horseName + "\nHorse Colour: " +horseColour + "\nHorse Age: " + horseAge); }</pre>
--	--	--

Step 6:

Your class files should now look like this:

Cat.java	<pre>public class Cat { public String catName; public String catColour; public int catAge; public void makeNoise(){ Console.WriteLine("Meow"); } public void printDetails(){ Console.WriteLine("Cat Details are as follows: \nCat Name: " + catName + "\nCat Colour: " +catColour +"\nCat Age: " + catAge); } }</pre>
Horse.java	<pre>public class Horse { public String horseName; public String horseColour; public int horseAge; public void makeNoise(){ Console.WriteLine("Neigh"); } public void printDetails(){ Console.WriteLine("Horse Details are as follows: \nHorse Name: "+horseName + "\nHorse Colour: " +horseColour + "\nHorse Age: " + horseAge); } }</pre>

Dog.java

```
public class Dog {  
  
    public String dogName;  
    public String dogColour;  
    public int dogAge;  
  
    public void makeNoise() {  
        Console.WriteLine("Woof Woof!");  
    }  
  
    public void printDetails() {  
        Console.WriteLine("Dog Details are as follows: \nDog Name: " + dogName  
+ "\nDog Colour: " + dogColour + "\nDog Age: " + dogAge);  
    }  
}
```

Step 7:

You will now need to navigate to your `Program` class which contains the main method. This is the class in which you will create your objects. First create a dog object. We use the class name first followed by the name we supply for the object, call it `dogOne`. Follow this with an equal's sign and then using the new keyword finished by referencing the class again.

```
Dog dogOne = new Dog();
```

Create another dog object and also objects of type horse and cat.

```
Dog dogOne = new Dog();  
Dog dogTwo = new Dog();
```

```
Cat catOne = new Cat();  
Cat catTwo = new Cat();
```

```
Horse horseOne = new Horse();  
Horse horseTwo = new Horse();
```

Step 8:

You can now set the properties of your objects. First, use the name of your object followed by a dot. Next, you must select the relevant property that you want to set e.g. `dogName`. Then use an equals sign followed by the value you want to set it to. Make sure you pay attention to the data type of the property you are setting so that you supply a correct value.

```
dogOne.dogName = "Marley";  
dogOne.dogColour = "Brown";  
dogOne.dogAge = 4;
```

```
dogTwo.dogName = "Rue";  
dogTwo.dogColour = "Black";  
dogTwo.dogAge = 3;
```

```
catOne.catName = "Tillums";  
catOne.catColour = "Black";  
catOne.catAge = 5;
```

```
catTwo.catName = "Scoop";
catTwo.catColour = "Grey";
catTwo.catAge = 7;

horseOne.horseName = "Shergar";
horseOne.horseColour = "Brown";
horseOne.horseAge = 10;

horseTwo.horseName = "Angus";
horseTwo.horseColour = "Black";
horseTwo.horseAge = 6;
```

Step 9:

Now call your `printDetails` method for each object to print out the details. You first use the name of the object followed by a dot and then the method name. There are no arguments for `printDetails` so the brackets at the end of the statement will be empty.

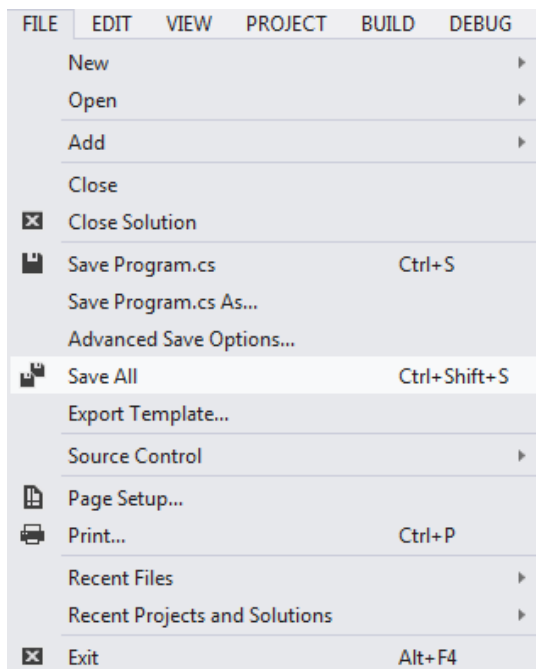
```
dogOne.printDetails();
dogTwo.printDetails();

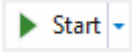
horseOne.printDetails();
horseTwo.printDetails();

catOne.printDetails();
catTwo.printDetails();
```

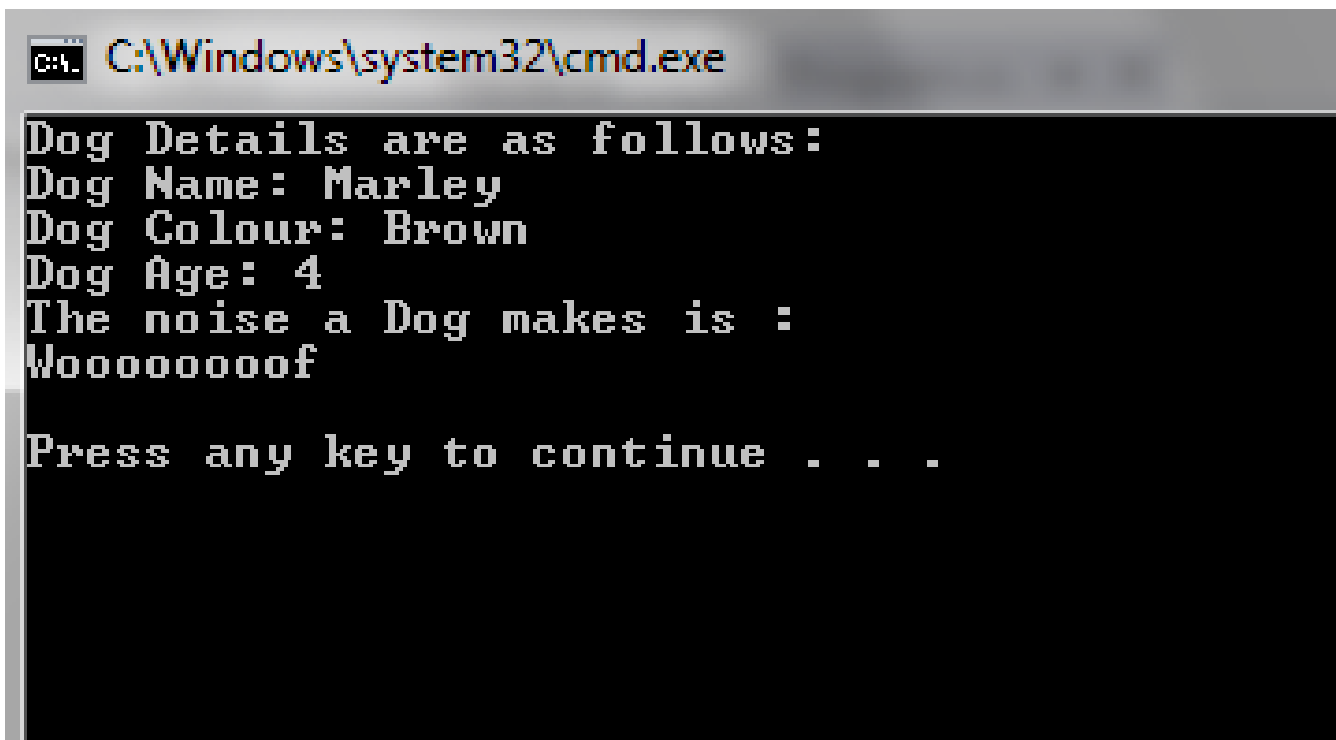
Step 10:

Save your code by selecting `File>Save All` or using the shortcut `Ctrl+Shift+S`.



Now run your project by selecting the run button from the toolbar  or using the shortcut `Ctrl+F5`

Example output is shown below.



```
C:\Windows\system32\cmd.exe
Dog Details are as follows:
Dog Name: Marley
Dog Colour: Brown
Dog Age: 4
The noise a Dog makes is :
Woooooooooof

Press any key to continue . . .
```

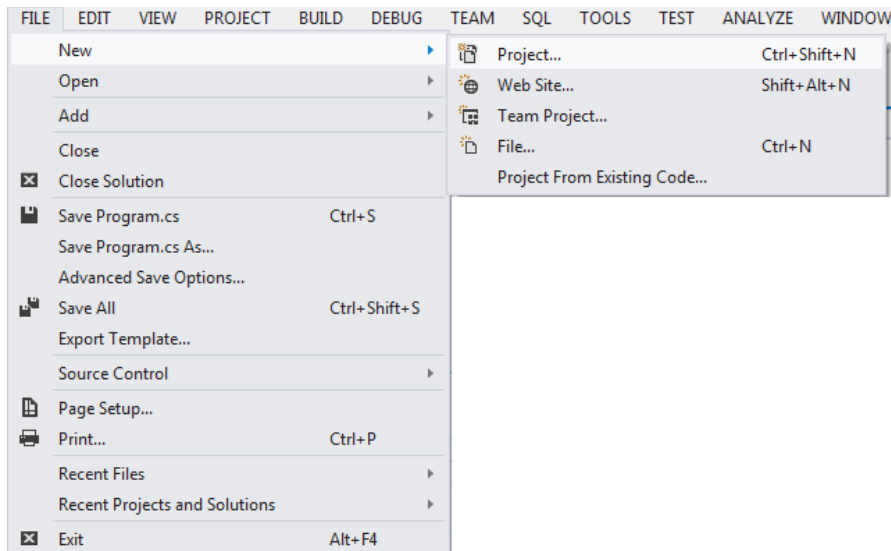

EXERCISE 2 – CREATING OBJECTS (SET METHODS)

Your farm will have 2 dogs, 2 cats and 2 horses. You will create the following files:

Dog.java	Cat.java	Horse.java	FarmTester.java
private variables	private variables	private variables	Main method
getters	getters	getters	Create your objects and pass in data
setters	setters	setters	Print out the stored details on each object.
additional methods	additional methods	additional methods	

Step 1:

Create a new project, C# Console Application. Name it `FarmyardSetMethods` and click Finish.



Create a new class and call it `Dog`. This is what the class will look like when it is initially created.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Farmyard
{
    public class Dog
    {

    }
}
```

Repeat step to create a blueprint class for your `Horse` and `Cat` objects.

Step 2:

Inside the Dog, Horse and Cat classes you must declare private variables which will be the fields of the class e.g. the dog/cat/horse's name, age and the colour. This is what each of your classes will look like after creating the variables.

<pre>public class Dog { private String dogName; private String dogColour; private int dogAge; }</pre>	<pre>public class Cat { private String catName; private String catColour; private int catAge; }</pre>	<pre>public class Horse { private String horseName; private String horseColour; private int horseAge; }</pre>
---	---	---

Step 3:

Now inside each of your classes for the animals you will create set methods which when called will allow you to set the properties of the object you have created. You will need a set method for each individual field of the object you wish to set. The set method accepts a parameter e.g. String newName. The data type will be the same as the field that you are changing. We set the field to equal the value of the parameter thus passing in values from another class and setting the properties of our object.

<pre>public void setName (String newName) { dogName = newName; } public void setColour (String newColour) { dogColour = newColour; } public void setAge (int newAge) { dogAge = newAge; }</pre>	<pre>public void setName (String newName) { catName = newName; } public void setColour (String newColour) { catColour = newColour; } public void setAge (int newAge) { catAge = newAge; }</pre>	<pre>public void setName (String newName) { horseName = newName; } public void setColour (String newColour) { horseColour = newColour; } public void setAge (int newAge) { horseAge = newAge; }</pre>
---	---	---

Step 4:

Underneath your set methods you will need to create get methods which when called will return the relevant values of your objects properties. We will use the get methods in collaboration with the printDetails which will be created later.

<pre>public String getName(){ return dogName; } public String getColour(){ return dogColour; } public int getAge(){ return dogAge; }</pre>	<pre>public String getName(){ return catName; } public String getColour(){ return catColour; } public int getAge(){ return catAge; }</pre>	<pre>public String getName(){ return horseName; } public String getColour(){ return horseColour; } public int getAge(){ return horseAge; }</pre>
--	--	--

Step 5:

Next, within the dog, horse and cat class you will create a method that will output the sound that the cat/horse/dog will make onto the console.

```
public void makeNoise(){
    Console.WriteLine("Woof");
}
```

```
public void makeNoise(){
    Console.WriteLine("Meooww");
}
```

```
public void makeNoise(){
    Console.WriteLine("Neigh");
}
```

Step 6:

Create a printDetails method in each class using the get methods created above. Here is the code.

```
public void printDetails(){
    Console.WriteLine("Dog
    Details are as follows:
    \nDog Name: " + getName()
    + "\nDog Colour: "
    +getColour() + "\nDog Age:
    " + getAge());
}
```

```
public void printDetails(){
    Console.WriteLine("Horse
    Details are as follows:
    \nHorse Name: "+getName()
    + "\nHorse Colour: "
    +getColour() + "\nHorse Age:
    " + getAge());
}
```

```
public void printDetails(){
    Console.WriteLine(
    "Cat Details are as
    follows: \nCat Name: " +
    getName()
    + "\nCat Colour: "
    +getColour() + "\nCat Age: "
    + getAge());
}
```

Step 7:

Navigate to your tester class which contains your main method. Here you will again create two objects of each animal class but this time use the set methods that you created to set the properties of the objects.

```
Dog dogOne = new Dog();
Dog dogTwo = new Dog();
```

```
Cat catOne = new Cat();
Cat catTwo = new Cat();
```

```
Horse horseOne = new Horse();
Horse horseTwo = new Horse();
```

Reference the object's name followed by a dot to access the set method. Then inside a set of brackets supply the value that you want to be set to the property.

```
dogOne.setName("Fido");
dogOne.setColour("Brown");
dogOne.setAge(5);
```

```
dogTwo.setName("Rue");
dogTwo.setColour("Black");
dogTwo.setAge(2);
```

```
catOne.setName("Molly");
catOne.setColour("Brown");
catOne.setAge(6);
```

```
catTwo.setName("Jasper");  
catTwo.setColour("White");  
catTwo.setAge(1);
```

```
horseOne.setName("Angus");  
horseOne.setColour("Tan");  
horseOne.setAge(3);
```

```
horseTwo.setName("Angie");  
horseTwo.setColour("Grey");  
horseTwo.setAge(1);
```

You can call these set methods again and again to change the properties of your objects e.g. we wanted to change the age of our dogOne to 6 we would simply call the `setAge` for that object and supply 6 as the parameter. Try changing some of the values to get experience using these methods.

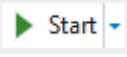
```
dogOne.setAge(6);
```

Step 8:

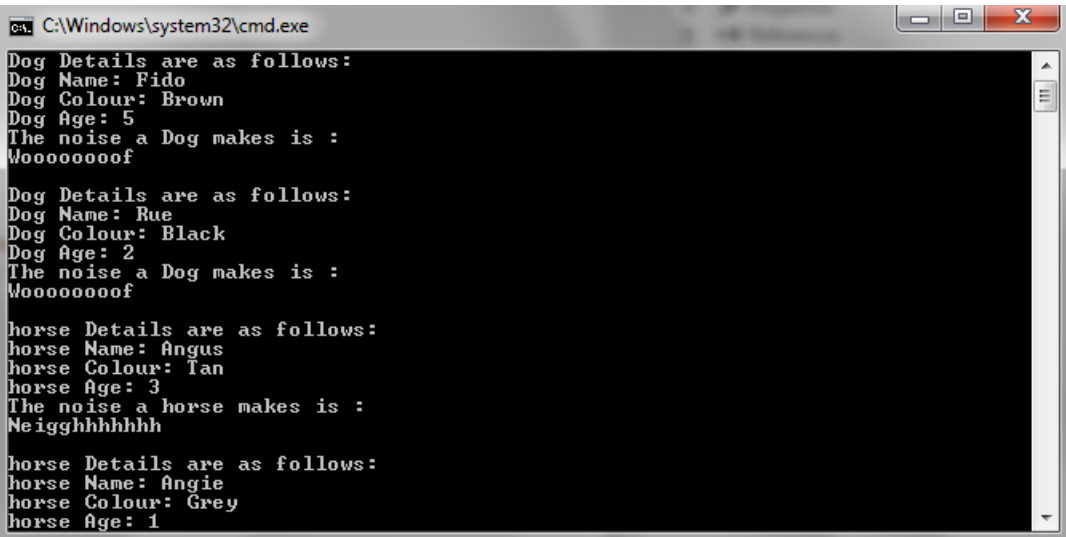
Call your `printDetails` method by using the object's name followed by a '.' and then the name of the method.

```
dogOne.printDetails();  
dogTwo.printDetails();  
  
horseOne.printDetails();  
horseTwo.printDetails();  
  
catOne.printDetails();  
catTwo.printDetails();
```

Step 9:

Save your work by selecting `File>Save All` or using the shortcut `Ctrl+Shift+S`. Then run your project using the Run button on the toolbar  or using the shortcut `Ctrl+F5`.

Example output is shown below.



```
C:\Windows\system32\cmd.exe  
Dog Details are as follows:  
Dog Name: Fido  
Dog Colour: Brown  
Dog Age: 5  
The noise a Dog makes is :  
Wooooooooof  
  
Dog Details are as follows:  
Dog Name: Rue  
Dog Colour: Black  
Dog Age: 2  
The noise a Dog makes is :  
Wooooooooof  
  
horse Details are as follows:  
horse Name: Angus  
horse Colour: Tan  
horse Age: 3  
The noise a horse makes is :  
Neigghhhhhh  
  
horse Details are as follows:  
horse Name: Angie  
horse Colour: Grey  
horse Age: 1
```

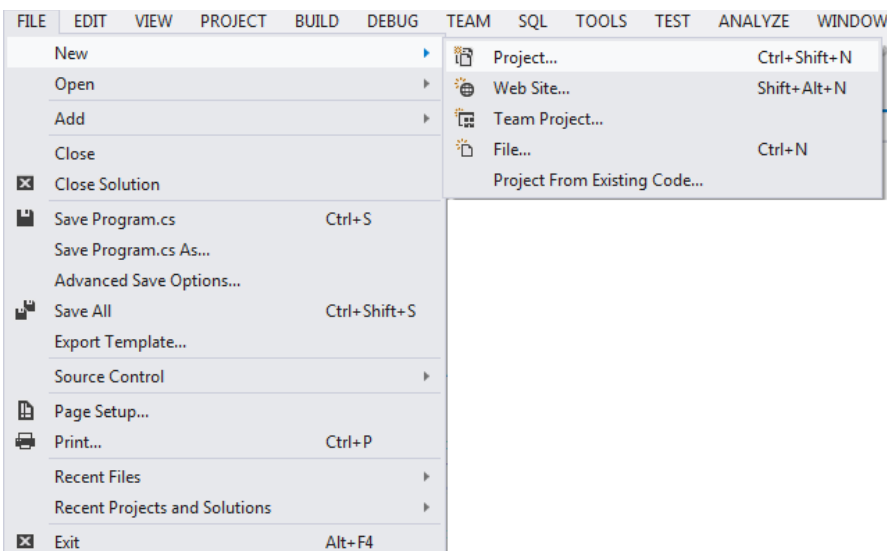
EXERCISE 3 – USING A CONSTRUCTOR

Your farm will have 2 dogs, 2 cats and 2 horses. You will create the following files:

Dog.java	Cat.java	Horse.java	FarmTester.java
private variables	private variables	private variables	Main method
constructor	constructor	constructor	Create your objects and pass in data
getters	getters	getters	Print out the stored details on each object.
setters	setters	setters	
additional methods	additional methods	additional methods	

Step 1:

Create a new project, C# Console Application. Name it `FarmyardConstructor` and click Finish.



Create a new class and call it `Dog`. This is what the class will look like when it is initially created.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Farmyard
{
    public class Dog
    {

    }
}
```

Repeat step to create a blueprint class for your `Horse` and `Cat` objects.

Step 2:

Inside the Dog, Horse and Cat classes you must declare private variables which will be the fields of the class e.g. the dog/cat/horse's name, age and the colour. This is what each of your classes will look like after creating the variables.

<pre>public class Dog { private String dogName; private String dogColour; private int dogAge; }</pre>	<pre>public class Cat { private String catName; private String catColour; private int catAge; }</pre>	<pre>public class Horse { private String horseName; private String horseColour; private int horseAge; }</pre>
---	---	---

Step 3:

You will now create the constructor which will be used when creating your objects. Constructor declarations look similar to method declarations except they use the name of the class and have no return type. Use the public keyword followed by the name of the class e.g. Dog.

<pre>public Dog(String DogName, String DogColour, int DogAge) { dogName = DogName; dogColour = DogColour; dogAge = DogAge; }</pre>	<pre>public Cat(String CatName, String CatColour, int CatAge) { catName = CatName; catColour = CatColour; catAge = CatAge; }</pre>	<pre>public Horse(String HorseName, String HorseColour, int HorseAge) { horseName = HorseName; horseColour = HorseColour; horseAge = HorseAge; }</pre>
--	--	--

The variables inside brackets e.g. DogName, are going to be used to pass in values to our properties so for example you set the String dogName to equal the value of whatever is passed in from DogName when the constructor is invoked in the main method.

Step 4:

Now inside each of your classes for the animals you will create set methods which when called will allow you to set the properties of the object you have created. You will need a set method for each individual field of the object you wish to set. The set method accepts a parameter e.g. String newName. The data type will be the same as the field that you are changing. We set the field to equal the value of the parameter thus passing in values from another class and setting the properties of our object.

<pre>public void setName(String newName) { dogName = newName; } public void setColour(String newColour) { dogColour = newColour; } public void setAge(int newAge) { dogAge = newAge; }</pre>	<pre>public void setName(String newName) { catName = newName; } public void setColour(String newColour) { catColour = newColour; } public void setAge(int newAge) { catAge = newAge; }</pre>	<pre>public void setName(String newName) { horseName = newName; } public void setColour(String newColour) { horseColour = newColour; } public void setAge(int newAge) { horseAge = newAge; }</pre>
--	--	--

Step 5:

Underneath your set methods you will need to create get methods which when called will return the relevant values of your objects properties. We will use the get methods in collaboration with the `printDetails` which will be created later.

<pre>public String getName(){ return dogName; } public String getColour(){ return dogColour; } public int getAge(){ return dogAge; }</pre>	<pre>public String getName(){ return catName; } public String getColour(){ return catColour; } public int getAge(){ return catAge; }</pre>	<pre>public String getName(){ return horseName; } public String getColour(){ return horseColour; } public int getAge(){ return horseAge; }</pre>
--	--	--

Step 5:

Next, within the dog, horse and cat class you will create a method that will output the sound that the cat/horse/dog will make onto the console.

<pre>public void makeNoise(){ Console.WriteLine("Woof"); }</pre>	<pre>public void makeNoise(){ Console.WriteLine("Meoww"); }</pre>	<pre>public void makeNoise(){ Console.WriteLine("Neigh"); }</pre>
--	---	---

Step 6:

Create a `printDetails` method in each class using the get methods created above. Here is the code.

<pre>public void printDetails(){ Console.WriteLine("Dog Details are as follows: \nDog Name: " + getName() + "\nDog Colour: " +getColour() + "\nDog Age: " + getAge()); }</pre>	<pre>public void printDetails(){ Console.WriteLine("Horse Details are as follows: \nHorse Name: "+getName() + "\nHorse Colour: " +getColour() + "\nHorse Age: " + getAge()); }</pre>	<pre>public void printDetails(){ Console.WriteLine("Cat Details are as follows: \nCat Name: " + getName() + "\nCat Colour: " +getColour() +"\nCat Age: " + getAge()); }</pre>
--	--	--

Step 7:

Navigate to your tester class which contains the main method. Here you will invoke your constructor to create your objects. First use the class name of the object you are creating e.g. `Dog` followed by a name of your choice in this example `dogOne`. Following this will be an equals sign and the new keyword and then the class name again. Inside brackets you will need to supply the parameters which match the properties that the object possesses e.g. name, colour and age. Create two objects for each of your classes.

The code required is shown below.

```
Dog dogOne = new Dog("Fido", "Black", 8);
Dog dogTwo = new Dog("Max", "White", 6);

Cat catOne = new Cat("Minx", "Ginger", 2);
Cat catTwo = new Cat("Scoop", "Black", 4);

Horse horseOne = new Horse("Shergar", "Brown", 12);
Horse horseTwo = new Horse("Ben", "Black", 4);
```

Step 8:

Call your `printDetails` method by using the object's name followed by a `.` and then the name of the method.

```
dogOne.printDetails();
dogTwo.printDetails();

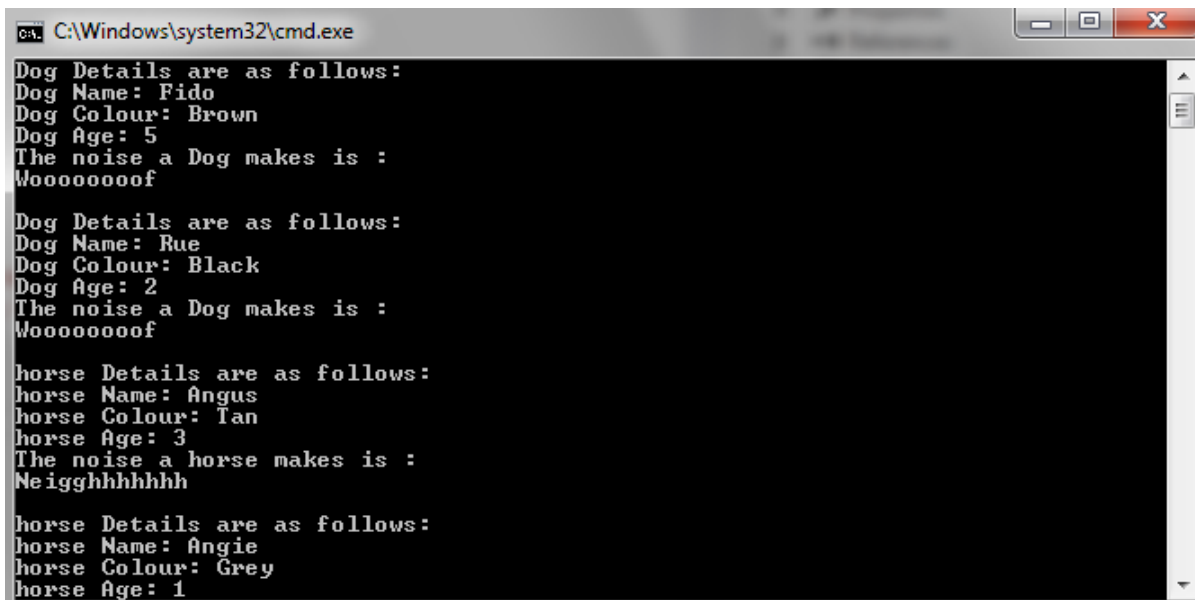
horseOne.printDetails();
horseTwo.printDetails();

catOne.printDetails();
catTwo.printDetails();
```

Step 9:

Save your work by selecting `File>Save All` or alternatively, use the shortcut `Ctrl+Shift+S`. Once you have done this, run your project using the `Run` button on the toolbar or by using the shortcut `Ctrl+F5`.

Example output is shown below:



```
C:\Windows\system32\cmd.exe
Dog Details are as follows:
Dog Name: Fido
Dog Colour: Brown
Dog Age: 5
The noise a Dog makes is :
Wooooooooof

Dog Details are as follows:
Dog Name: Rue
Dog Colour: Black
Dog Age: 2
The noise a Dog makes is :
Wooooooooof

horse Details are as follows:
horse Name: Angus
horse Colour: Tan
horse Age: 3
The noise a horse makes is :
Neigghhhhhh

horse Details are as follows:
horse Name: Angie
horse Colour: Grey
horse Age: 1
```

EXERCISE 4- DOCTOR'S SURGERY

In this exercise you will create classes to represent doctors, nurses and patients. Use constructors to create objects of these types. Print out the details for your objects at the end.