In this exercise you will be using inheritance to promote code re-use. We will be starting with a base class and add more capability to the child class. The Server is the parent of Supervisor who is in turn the parent of StoreManager.

Remember in an inheritance hierarchy the parent class is normally with the most members but is the least capable class and the most derived class probably implements the least number of members but is the most capable class.

# You must follow the specifications exactly (To demo to instructor at the end of this class)

Because the required code statements for each type is very few, you may choose to implement all the classes in a single code file.

Server  
Class

Supervisor  
→ Server

Manager  
→ Supervisor

There are only three classes participating in this application as shown in the diagram above. Each of the classes is fully described below.

# The Server class

This class comprises of five members. This is parent of all the other classes in this system.

|  |
| --- |
| **Server**  **Class** |
| **Fields** |
| **# duties : List<string>**  **# wages : double** |
| **Properties** |
| **+ «C# properties setter is absent» Name: string** |
| **Methods** |
| **+ «constructor» Server(name : string)**  **+ ToString() : string** |

#### Description of class members

##### Fields:

duties – This protected field is a list of strings representing the duties of this server objects.

wages – This protected field is a double representing the wage of this server object.

##### Properties:

**Name** – this string property represents the name of this object. The getter is public and the setter is absent.

##### Constructor:

**public Server(string name)** – This constructor takes a string parameter and assigns it to the appropriate property. It also initialized the wages field with a double and the duties field with about four to five strings (duties).

##### Methods:

**public override string ToString()** – This is a public method overrides the corresponding method in the object class to return the Name and all the duties of this server object. [Use the **string.Join(", ", duties)** to convert the list to a string.

# The Supervisor class

This class inherits all the members of the Server class except of course the constructor. It only implements its own constructor.

|  |
| --- |
| **Supervisor**  **Class**  **Server** |
| **Fields** |
|  |
| **Properties** |
|  |
| **Methods** |
| **+ «constructor» Supervisor( name : string )** |

#### Description of class members

##### Fields:

The ones inherited from its parent.

##### Properties:

The ones inherited from its parent.

##### Constructor:

**public Supervisor(string name)** – This constructor takes one parameter, it invokes the base constructor with the supplied argument. Remember that the statement to call the base constructor must be placed before the body of the method.   
It also adds at least another duty to the parent list of duties.

##### Methods:

The ones inherited from its parent . Nothing to implement.

# The Manager class

This class inherits from the Supervisor class and it only declares a constructor.

|  |
| --- |
| **Manager**  **Class**  **Supervisor** |
| **Fields** |
|  |
| **Properties** |
|  |
| **Methods** |
| **+ «constructor» Manager(name : string)** |

#### Description of class members

##### Fields:

The ones inherited from its parent.

##### Properties:

The ones inherited from its parent.

##### Constructor:

**public Manager(string name)** – This constructor takes one parameter. It invokes the base constructor with it.  
It also adds at least another duty to the duties of its parent.

##### Methods:

The ones inherited from its parent.

### Test Harness

Write the code to create one object from each class and display the resulting object.

//Create a list of Severs with one of each object types

List<Server> workers = new List<Server>()

{

new Server("Jatin"),

new Supervisor("Yash"),

new Manager("Lovely");

};

//Use foreach loop to display each object

foreach(var worker in workers)

Console.WriteLine(worker);