Eyes of the Dragon Tutorials
Part 10
Character Classes

I'm writing these tutorials for the MonoGame 3.8 framework using Visual Studio 2019. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the Eyes of the Dragon page of my web blog. I will be making each version of the project available on GitHub here. It will be included on the page that links to the tutorials.

I had started a new tutorial that I was adding in editors for the game. I realized when I was about half done that I needed to revamp a few things. They are minor things but changing them now means that they won't have to be changed down the road. After changing a few things I will then work on the editors.

First thing I think is it will be better to be able to read in character classes at run time rather than have static character classes. The ability is already there, I just need to make a few quick adjustments. First, right click the **Fighter**, **Thief**, **Priest**, and **Wizard** classes in the **CharacterClasses** folder of the **RpgLibrary** project and select **Delete**. You are not going to need them. I am going to add a manager class to the **RpgLibrary** to manage the entity types. Right click the **RpgLibrary**, select **Add** and then **Class**. Name this class **EntityDataManager**. This is the code for the **EntityDataManager** class.

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

namespace RpgLibrary.CharacterClasses
{
    public class EntityDataManager
    {
        #region Field Region

        readonly Dictionary<string, EntityData> entityData =
            new Dictionary<string, EntityData>();

        #endregion

        #region Property Region
        public Dictionary<string, EntityData> EntityData
        {
            get { return entityData; }
        }

        #endregion

        #region Constructor Region
        #endregion

        #region Method Region
        #endregion
    }
}
```

A fairly straight forward class. There is one field, **entityData**, that is marked readonly to hold all of the **EntityData** objects in the game. This modifier is like the **const** modifier. The difference is you can assign it a value in the class declaration or in the constructor of the class. This doesn't mean that the field can't be modified though. It just means that there can't be an assignment to the field. You can access the field using methods and properties, like adding an item to the list using the **Add** method.

The **entityData** field is a **Dictionary<string, EntityData>**. I used a dictionary as **EntityData** instances should have unique names, as they are names for character classes. There is a property to expose the field.

I'm going to make an update to the **EntityData** class. I'm going to remove the static methods that I added and add in an override of the **ToString** method. I'm also going to add in a **Clone** method to clone an **EntityData** class. The way I'm going to do the editors you are not going to need the static methods.

I'm going to do it in such a way that content will be written to an XML file. You can then read in this XML file using the **Content Pipeline** and reflection. Change the **EntityData** class to the following.

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

namespace RpgLibrary.CharacterClasses
{
    public class EntityData
    {
        #region Field Region

        public string EntityName;
        public int Strength;
        public int Dexterity;
        public int Cunning;
        public int Willpower;
        public int Magic;
        public int Constitution;
        public string HealthFormula;
        public string StaminaFormula;
        public string MagicFormula;

        #endregion

        #region Constructor Region

        private EntityData()
        {
        }

        public EntityData(
            string entityName,
            int strength,
            int dexterity,
            int cunning,
            int willpower,
            int magic,
            int constitution,
            string health,
            string stamina,
            string mana)
        {
            EntityName = entityName;
            Strength = strength;
            Dexterity = dexterity;
            Cunning = cunning;
            Willpower = willpower;
            Cunning = cunning;
            Willpower = willpower;
            Magic = magic;
            Constitution = constitution;
            HealthFormula = health;
            StaminaFormula = stamina;
            MagicFormula = mana;
        }

        #endregion

        #region Method Region

        public override string ToString()
        {
            string toString = "Name = " + EntityName + ", ";

            toString += "Strength = " + Strength.ToString() + ", ";
            toString += "Dexterity = " + Dexterity.ToString() + ", ";
            toString += "Cunning = " + Cunning.ToString() + ", ";
            toString += "Willpower = " + Willpower.ToString() + ", ";
            toString += "Magic = " + Magic.ToString() + ", ";
            toString += "Constitution = " + Constitution.ToString() + ", ";
            toString += "Health Formula = " + HealthFormula + ", ";
            toString += "Stamina Formula = " + StaminaFormula + ", ";
            toString += "Magic Formula = " + MagicFormula;

            return toString;
        }

        public object Clone()
        {
            EntityData data = new EntityData
            {
                EntityName = EntityName,
                Strength = Strength,
                Dexterity = Dexterity,
                Cunning = Cunning,
                Willpower = Willpower,
                Magic = Magic,
                Constitution = Constitution,
                HealthFormula = HealthFormula,
                StaminaFormula = StaminaFormula,
                MagicFormula = MagicFormula
            };

            return data;
        }

        #endregion
```

```
    }
}
```

The **ToString** and an equals sign, another space and then the field. For all fields except the last I also add a comma and a space. I didn't implement the **ICloneable** interface this time. You will also want to change your **Animation** class to not use the **ICloneable** interface. In the .NET Compact version, which the Xbox 360 uses, you will get a compile time error that **ICloneable** is not available due to its protection level. You will still want to have a **Clone** method though, just don't use the interface to do it. We are probably fine but I want to try and keep things as cross platform as possible.

The **Clone** method uses the private constructor to create a new instance of **EntityData**. It then sets all of the fields of the instance using the current object.

Before I get to the editors I want to update the **Entity** class. Instead of it being an abstract class I want it to be a sealed class. A sealed class is a class that can't be inherited from. I also want to add in an enumeration for different types of entities. Even though there is a lot of code I don't think there is anything that truly needs to be explained. Things that were protected were made private. A few fields were added and properties to expose them. The public constructor now takes a string for the name, an **EntityData** that defines the type of entity, an **EntityGender** for the gender of the entity, and an **EntityType** for the type of entity. When I talk about types of entities there will be four distinct groups. There are characters, like the player's characters, NPCs for the player to interact with, monsters and creatures. I could have lumped all enemies the player will face into one category, monster, but I though there may be instances where it will be useful to have animals, like giant spiders or wolves. This is the updated **Entity** class.

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

namespace RpgLibrary.CharacterClasses
{
    public enum EntityGender { Male, Female, NonBinary, Unknown }
    public enum EntityType { Character, NPC, Monster, Creature }

    public sealed class Entity
    {
        #region Vital Field and Property Region

        private string entityName;
        private string entityClass;
        private EntityType entityType;
        private EntityGender gender;

        public string EntityName
        {
            get { return entityName; }
            private set { entityName = value; }
        }

        public string EntityClass
        {
            get { return entityClass; }
            private set { entityClass = value; }
        }

        public EntityType EntityType
        {
            get { return entityType; }
            private set { entityType = value; }
        }

        public EntityGender Gender
        {
            get { return gender; }
            private set { gender = value; }
        }

        #endregion

        #region Basic Attribute and Property Region

        private int strength;
        private int dexterity;
        private int cunning;
        private int willpower;
        private int magic;
        private int constitution;
        private int strengthModifier;
        private int dexterityModifier;
        private int cunningModifier;
        private int willpowerModifier;
        private int magicModifier;
        private int constitutionModifier;

        public int Strength
        {
            get { return strength + strengthModifier; }
            private set { strength = value; }
        }

        public int Dexterity
        {
            get { return dexterity + dexterityModifier; }
            private set { dexterity = value; }
        }

        public int Cunning
        {
            get { return cunning + cunningModifier; }
            private set { cunning = value; }
        }

        public int Willpower
        {
            get { return willpower + willpowerModifier; }
            private set { willpower = value; }
        }

        public int Magic
        {
            get { return magic + magicModifier; }
            private set { magic = value; }
```

```
}

public int Constitution
{
    get { return constitution + constitutionModifier; }
    private set { constitution = value; }
}

#endregion

#region Calculated Attribute Field and Property Region

private AttributePair health;
private AttributePair stamina;
private AttributePair mana;

public AttributePair Health
{
    get { return health; }
}

public AttributePair Stamina
{
    get { return stamina; }
}

public AttributePair Mana
{
    get { return mana; }
}

private int attack;
private int damage;
private int defense;

#endregion

#region Level Field and Property Region

private int level;
private long experience;

public int Level
{
    get { return level; }
    private set { level = value; }
}

public long Experience
{
    get { return experience; }
    private set { experience = value; }
}

#endregion

#region Constructor Region

private Entity()
{
```

```
            Strength = 0;
            Dexterity = 0;
            Cunning = 0;
            Willpower = 0;
            Magic = 0;
            Constitution = 0;
            health = new AttributePair(0);
            stamina = new AttributePair(0);
            mana = new AttributePair(0);
        }

        public Entity(
            string name,
            EntityData entityData,
            EntityGender gender,
            EntityType  entityType)
        {
            EntityName = name;
            EntityClass = entityData.EntityName;
            Gender = gender;
            EntityType = entityType;
            Strength = entityData.Strength;
            Dexterity = entityData.Dexterity;
            Cunning = entityData.Cunning;
            Willpower = entityData.Willpower;
            Magic = entityData.Magic;
            Constitution = entityData.Constitution;
            health = new AttributePair(0);
            stamina = new AttributePair(0);
            mana = new AttributePair(0);
        }

        #endregion
    }
}
```

The next step is to add in the editor. Make sure that all open files are saved for this step. The next step is to add a project for the editors. The editors will do the serializing and deserializing of your content. You can deserialize your objects from the editor so that you don't have to read in your files and parse them manually. Right click the solution in the solution explorer. Select **Add** and then **New Project**. Select a Windows Forms project from the C# list and name this new project **RpgEditor**. Now, right click the **RpgEditor** project and select **Manage NuGet Packages**. Click the **Browse** tab on that screen. Search for **MonoGame.Framework.Content.Pipeline** and install it. Also search for **MonoGame.Framework.DesktopGL** and install it.

Now you need to reference your two libraries. Right click the **RpgEditor** project and select **Add Reference**. From the **Projects** tab select the **RpgLibrary** and **XRpgLibrary** entries. Right click the **Form1** class in the solution explorer and select **REname** to rename it. Rename it to **FormMain** and in the dialog box that pops up choose to rename the code references from **Form1** to **FormMain**. Right click the **RpgEditor** again, select **Add** and then **Class**. Name this new class **XnaSerializer**. The code for the class follows next.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Xml;
using Microsoft.Xna.Framework.Content.Pipeline.Serialization.Intermediate;

namespace RpgEditor
{
    static class XnaSerializer
    {
        public static void Serialize<T>(string filename, T data)
        {
            XmlWriterSettings settings = new XmlWriterSettings
            {
                Indent = true
            };

            using (XmlWriter writer = XmlWriter.Create(filename, settings))
            {
                IntermediateSerializer.Serialize<T>(writer, data, null);
            }
        }

        public static T Deserialize<T>(string filename)
        {
            T data;

            using (FileStream stream = new FileStream(filename, FileMode.Open))
            {
                using (XmlReader reader = XmlReader.Create(stream))
                {
                    data = IntermediateSerializer.Deserialize<T>(reader, null);
                }
            }

            return data;
        }
    }
}
```

That looks like a whole pile of ugly but once it is explained I don't think you will find it that bad. First off I added using statements for the **System.IO** and **System.Xml** name spaces because I use class from both of them. There is also a using statement for an XNA name space. That is the reason I added the reference to the **Content Pipeline**. I needed access to the **IntermediateSerializer** class. With this class you can write out XML content in a way that the **Content Pipeline** can read it in using reflection as I mentioned earlier. This class is a static class. You never need to create an instance of this class. You use the two static methods to serialize and deserialize objects. They are generic methods, like the **Load** method of the **ContentManager** class. You can specify what type you want to use. This will prevent the need to write methods to serialize and deserialize every class you want to write out and read in.

You can just use the generic method and pass in the type you want to use. You will see it in action in the editor.

The first method, **Serialize<T>**, is used to serialize the object into an XML document. It basically writes out the object in XML format. There maybe times when you don't want a field or property written. You can flag it with an attribute that will prevent it from being serialized. An example would be if you had a **Texture2D** field. You don't want to serialize that, just your other data.

The first step in serializing using the **IntermediateSerializer** class is to create an **XmlWriterSettings** object. You use this object to set the **Indent** property to true as you want items indented. Unlike **using** directives for name spaces a **using** statement that you find inside code automatically disposes of any disposable objects with the statement ends. In this case the **XmlWriter** will be disposed when the statement ends. Inside the using statement is why I ended up deciding to create generic methods. The **Serialize** method of the **IntermediateSerializer** class requires a type, **T**, like the **Load** method of the **ContentManager**. I can use the **T** from the generic method in the call to **Serialize**. The parameters that I pass to the **Serialize** method are an XML stream, the **XmlWriter**, the object to be serialized, and a reference path. You can pass in null for this if you don't want to use one.

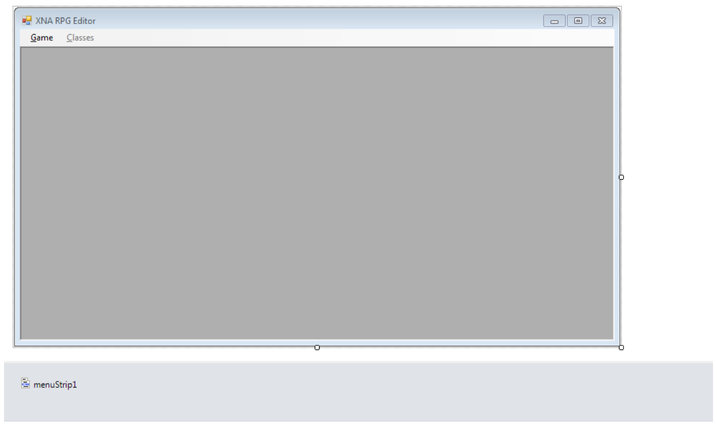
The **Deserialize<T>** method returns an object of type T. It first has a local variable of type T that it can return. To use the **XmlReader** class to read in an XML file you need a stream for input. In a using statement I create a stream to open the file passed in as a parameter. Inside that code block there is another using statement that creates an **XmlReader** to read in the XML file. I then call the **Deserialize** method of the **IntermediateSerializer** class to deserialize the document. I then return the object that was deserialized. It would be a good idea to check for exceptions such as the XML file was of the wrong type or the file didn't exist or the file could not be created. That would work well in the editor though and I will do that there.

I want to add a class to the **RpgLibrary** before moving on to the editor. This class will describe your games. It is used more for the editor than anything. Right click the **RpgLibrary** project, select **Add** and then **Class**. Name this class **RolePlayingGame**. This is the code.

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

namespace RpgLibrary
{
    public class RolePlayingGame
    {
        #region Field Region

        string name;
        string description;

        #endregion

        #region Property Region

        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public string Description
        {
            get { return description; }
            set { description = value; }
        }

        #endregion

        #region Constructor Region

        private RolePlayingGame()
        {
        }

        public RolePlayingGame(string name, string description)
        {
            Name = name;
            Description = description;
        }

        #endregion
    }
}
```

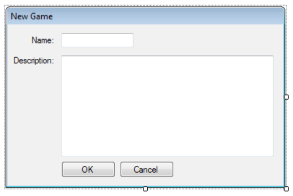
There are two fields in the class. One for the name of the game and the other for a description of the game. There are public properties to expose the fields. The constructor of the class takes a name and description and sets the fields using the properties. This class will be fleshed out more as we go.

Designing forms is never easily done using text. I will show you a picture of what my finished forms look like in Visual C# and what controls were added. Open the designer view of **FormMain** by right clicking it and selecting **View Designer**. This is what my finished form looked like in the designer.
I first made the form a little bigger. I then I dragged on was a **Menu Strip** control. In the first entry of the **Menu Strip** type **&Game**. The & before Game means that if the user presses ALT+G it will open that menu. Under the **Game** add in four other entries: **&New Game**, **&Open Game**, **&Save Game**, -, and **E&xit Game**. Entering the -, minus sign, gives you a separator in your menu. Beside **&Game** you want to add **&Classes**. Set the **Enabled** property for this item to false. You now want to set a few property of **FormMain**. Set the **IsMdiContainer** property to true, the **Text** property to **Eyes of the Dragon Editor**, and the **StartUpPosition** to **CenterScreen**. MDI is a way of having multiple children forms of a parent form. This way you can have multiple forms open like the form for creating character classes and a form for weapons and switch back and forth between them easily.



The next form I want to create is a form for creating a new game. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormNewGame**. This is what my

FormNewGame looked like in the designer.



First, make the form a little bigger to house all of the controls. Drag on a **Label** and set its **Text** property to **Name:**. Drag on on a second **Label** and set its **Text** property to **Description:**. Drag on a **Text Box** and position it beside the **Name: Label**. Set its **Name** property to **txtName**. Drag a second **Text Box** onto the form. Set its **Name** property to **txtDescription** and its **Multiline** property to true. Make it a little bigger to give the user more of an area to work with. Drag two buttons onto the form and position them under **txtDescription**. Set the **Name** of the first to **btnOK**, the **Text** property to **OK** and the **DialogResult** property to **OK**. Set the **Name** of the second to **btnCancel**, **Text** to **Cancel**, and **DialogResult** to **Cancel**. On the form itself set the **Text** property to **New Game**, **Control Box** to false, and **FormBorderStyle** to **FixedDialog**.

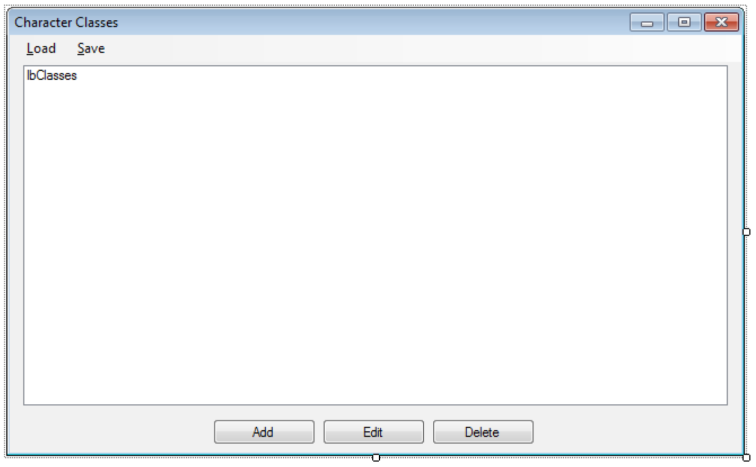
While this form is open might as well code the logic for it. Right click **FormNewGame** in the solution explorer and select **View Code**. Change the code for **FormNewGame** to the following.

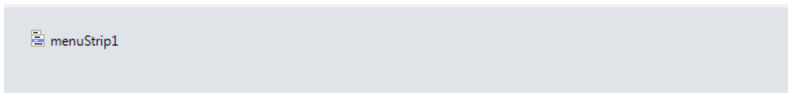
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary;

namespace RpgEditor
{
    public partial class FormNewGame : Form
    {
        #region Field Region

        RolePlayingGame rpg;

        #endregion

        #region Property Region

        public RolePlayingGame RolePlayingGame
        {
            get { return rpg; }
        }

        #endregion

        #region Constructor Region

        public FormNewGame()
        {
            InitializeComponent();
            btnOK.Click += new EventHandler(btnOK_Click);
        }

        #endregion

        #region Event Handler Region

        void btnOK_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(txtName.Text) ||
string.IsNullOrEmpty(txtDescription.Text))
            {
                MessageBox.Show("You must enter a name and a description.", "Error");
                return;
            }

            rpg = new RolePlayingGame(txtName.Text, txtDescription.Text);

            this.Close();
        }

        #endregion
    }
}
```

There is a field of type **RolePlayingGame** to be constructed using the form and a public property to expose it as read only, get only. The constructor wires an event handler for the **Click** event of **btnOK**. The event handler checks to see if **txtName** or **txtDescription** have text in them. If they don't a message box is shown and the method exits. Because I set the **DialogResult** of **btnOK** to OK the form will close when **btnOK** is pressed. If both text boxes had values then I create a new **RolePlayingGame** object with their values and then close the form.

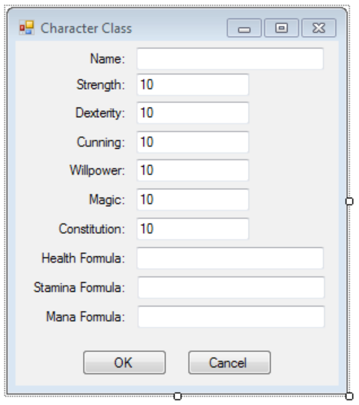
I'm going to add two forms for dealing with entity data, or character classes. The one form holds all of the character classes in the game. The second is use for creating new classes and editing character classes. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormClasses**. What my form looked like in the designer is on the next page. To get started drag a **Menu Strip** onto the form and making the form a little bigger. You can set the **Text** property of the 9413form to **Character Classes**. For the **Menu Strip** you can set the first item to **&Load** and the second item to **&Save**. Drag a **List Box** onto form and size it similarly to mine. You can also drag three **Buttons** onto the form positioning them under the **List Box**.

For the **List Box** set the following properties. **Name** is **lbClasses**, and **Anchor** property to **Top**, **Bottom**, **Left**, **Right**. Setting the **Anchor** property to that will have the list box change its size as you change the size of the form. For the first button set its **Name** property to **btnAdd**, **Anchor** property to **Bottom**, and its **Text** property to **Add**. The second button's **Name**, **Anchor**, and **Text** properties are **btnEdit**, **Bottom**, and **Edit** respectively. Set those same properties for the last to **btnDelete**, **Bottom**, and **Delete**.





The last form I'm going to add is the form for creating a new character class or editing an existing one. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormEntityData**. What my form looked like is next.



Name:	
Strength:	10
Dexterity:	10
Cunning:	10
Willpower:	10
Magic:	10
Constitution:	10
Health Formula:	
Stamina Formula:	
Mana Formula:	

Make the form bigger so there is room for all of the controls. I then dragged 10 **Label** controls onto the form. Set the text properties of the labels to **Name:**, **Strength:**, **Dexterity:**, **Cunning:**, **Willpower:**, **Magic:**, **Constitution:**, **Health Formula:**, **Stamina Formula:**, and **Mana Formula:**. You will then drag a **Text Box** onto the form and position it beside the **Name:** label. Make it a little bigger and set the **Name** property to **tbName**.

I next dragged a **Masked Text Box** onto the form. Set the **Mask** property to 00 which means the text box will only accept 2 digits. Set the **Text** property to 10 as well. Instead of having to set those properties 5 more times you can replicate the control. Select the control and while holding the **Ctrl** key drag the **Masked Text Box** below the other. Repeat the process until there is a masked text box beside the other attributes. Once you have all six set their **Name** properties to match the text of the label that they are across from: **mtbStrength**, **mtbDexterity**, **mtbCunning**, **mtbWillpower**, **mtbMagic**, and **mtbConstitution**.

You will want to drag a **Text Box** beside the remaining labels. Name them according to the label they are beside: **tbHealth**, **tbStamina**, and **tbMana**. The last two controls to drag onto the form are the buttons. Set the **Name** of the one to **btnOK**, and the **Text** property to **OK**. For the second button set the **Name** to **btnCancel**, the text to **Cancel**, and the **DialogResult** to **Cancel**. Click on the title bar of the form to bring up the properties of the form. Set the **AcceptButton** property to **btnOK**, the **CancelButton** property to **btnCancel**, **FormBorderStyle** to **FixedDialog**, and the **Text** property to **Character Class**.

I'm going to add in the logic for the form as there isn't a lot of code to it. Right click **FormEntityData** and select **View Code**. This is the code for that form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.CharacterClasses;

namespace RpgEditor
{
    public partial class FormEntityData : Form
    {
        #region Field Region

        EntityData entityData = null;

        #endregion

        #region Property Region

        public EntityData EntityData
        {
            get { return entityData; }
            set { entityData = value; }
        }

        #endregion

        #region Constructor Region

        public FormEntityData()
        {
            InitializeComponent();
            this.Load += new EventHandler(FormEntityData_Load);
            this.FormClosing += new FormClosingEventHandler(FormEntityData_FormClosing);

            btnOK.Click += new EventHandler(btnOK_Click);
            btnCancel.Click += new EventHandler(btnCancel_Click);
        }
        #endregion

        #region Event Handler Region

        void FormEntityData_Load(object sender, EventArgs e)
        {
            if (entityData != null)
            {
                tbName.Text = entityData.EntityName;
                mtbStrength.Text = entityData.Strength.ToString();
                mtbDexterity.Text = entityData.Dexterity.ToString();
                mtbCunning.Text = entityData.Cunning.ToString();
                mtbWillpower.Text = entityData.Willpower.ToString();
                mtbConstitution.Text = entityData.Constitution.ToString();
                tbHealth.Text = entityData.HealthFormula;
                tbStamina.Text = entityData.StaminaFormula;
                tbMana.Text = entityData.MagicFormula;
            }
        }

        void FormEntityData_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (e.CloseReason == CloseReason.UserClosing)
            {
                e.Cancel = true;
            }
        }

        void btnOK_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(tbName.Text) || string.IsNullOrEmpty(tbHealth.Text) ||
                string.IsNullOrEmpty(tbStamina.Text) || string.IsNullOrEmpty(tbMana.Text))
            {
                MessageBox.Show("Name, Health Formula, Stamina Formula and Mana Formula must
have values.");
                return;
            }

            int str = 0;
            int dex = 0;
            int cun = 0;
            int wil = 0;
            int mag = 0;
            int con = 0;

            if (!int.TryParse(mtbStrength.Text, out str))
            {
                MessageBox.Show("Strength must be numeric.");
                return;
            }

            if (!int.TryParse(mtbDexterity.Text, out dex))
            {
                MessageBox.Show("Dexterity must be numeric.");
                return;
            }

            if (!int.TryParse(mtbCunning.Text, out cun))
            {
                MessageBox.Show("Cunning must be numeric.");
                return;
            }

            if (!int.TryParse(mtbWillpower.Text, out wil))
            {
                MessageBox.Show("Willpower must be numeric.");
                return;
            }

            if (!int.TryParse(mtbMagic.Text, out mag))
            {
                MessageBox.Show("Magic must be numeric.");
                return;
            }

            if (!int.TryParse(mtbConstitution.Text, out con))
            {
                MessageBox.Show("Constitution must be numeric.");
                return;
            }

            entityData = new EntityData(
                tbName.Text,
                str,
                dex,
                cun,
                wil,
                mag,
                con,
                tbHealth.Text,
                tbStamina.Text,
                tbMana.Text);

            this.FormClosing -= FormEntityData_FormClosing;
            this.Close();
        }

        void btnCancel_Click(object sender, EventArgs e)
        {
            entityData = null;

            this.FormClosing -= FormEntityData_FormClosing;
            this.Close();
        }

        #endregion
    }
}
```

I added in a using statement for the **CharacterClasses** name space of the **RpgLibrary**. I have a field **entityData** of type **EntityData** for the entity data being entered. There is also a public property to expose it to other forms.

In the constructor of the form I wire event handlers for the **Load** event of the form and the **Click** events of **btnOK** and **btnCancel**. In the **Load** event of the form I check to see **entityData** is not null. If it is not null I fill out the text boxes and masked text boxes with the appropriate fields from **entityData**. The click event of **btnOK** checks to make sure that the **Text** properties of **tbName**, **tbHealth**,

tbStamina, and **tbMana** have values. If they don't I display a message box and exit the method. There are then six local integer variables for the stats of **EntityData**. Next follows a series of if statements where I use the **TryParse** method of the int class to parse the **Text** property of the masked text boxes for the stats. Even though the mask only allows numeric values they can be empty and if you just use the **Parse** method that will generate an exception and your program will crash. If any of the **TryParse** calls fail I display a message box and exit the method. I then set the **entityData** field to a new instance using the values of the text boxes and local integer variables. Finally I close the form. In the **Click** event handler for **btnCancel** I set the **entityData** field to null. I then close the form.

I'm going to add in some basic logic for the **FormClasses** now. Right click **FormClasses** in the **RpgEditor** project and select **View Code**. Add the following code for **FormClasses**.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.CharacterClasses;

namespace RpgEditor
{
    public partial class FormClasses : Form
    {
        #region Field Region

        EntityDataManager entityDataManager = new EntityDataManager();

        #endregion

        #region Constructor Region
        public FormClasses()
        {
            InitializeComponent();

            loadToolStripMenuItem.Click += new EventHandler(loadToolStripMenuItem_Click);
            saveToolStripMenuItem.Click += new EventHandler(saveToolStripMenuItem_Click);

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Menu Item Event Handler Region

        void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }
        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
            using (FormEntityData frmEntityData = new FormEntityData())
            {
                frmEntityData.ShowDialog();
                if (frmEntityData.EntityData != null)
                {
                    lbClasses.Items.Add(frmEntityData.EntityData.ToString());
                }
            }
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
        }

        #endregion
    }
}
```

There is an **EntityDataManager** field in this class because it will hold the different **EntityData** objects. In the constructor I wire the event handlers for the menu items and the buttons. I added some basic code to the **btnAdd_Click** event handler. There is a using statement that creates a new instance of **FormEntityData** so that when you are done with the form it will be disposed of. Inside I call the **ShowDialog** method of the form to display it as a modal form. That required the form to be closed before you can use the calling form again. I check to see if the **EntityData** property of the form is not null. If it is I add the **EntityData** object to the list box of **EntityData** objects. I will add in more functionality in another tutorial.

I'm also going to add a little logic to the **FromMain**. Right click **FormMain** and select **View Code**. Update the code for **FormMain** to the following.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary;
using RpgLibrary.CharacterClasses;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{
    public partial class FormMain : Form
    {
        #region Field Region

        RolePlayingGame rolePlayingGame;
        FormClasses frmClasses;

        #endregion

        #region Property Region
        #endregion

        #region Constructor Region
        public FormMain()
        {
            InitializeComponent();

            newGameToolStripMenuItem.Click += new EventHandler(newGameToolStripMenuItem_Click);
            openGameToolStripMenuItem.Click += new
EventHandler(openGameToolStripMenuItem_Click);
            saveGameToolStripMenuItem.Click += new
EventHandler(saveGameToolStripMenuItem_Click);
            exitToolStripMenuItem.Click += new EventHandler(exitToolStripMenuItem_Click);
            classesToolStripMenuItem.Click += new EventHandler(classesToolStripMenuItem_Click);
        }

        #endregion

        #region Menu Item Event Handler Region

        void newGameToolStripMenuItem_Click(object sender, EventArgs e)
        {
            using (FormNewGame frmNewGame = new FormNewGame())
            {
                DialogResult result = frmNewGame.ShowDialog();

                if (result == DialogResult.OK && frmNewGame.RolePlayingGame != null)
                {
                    classesToolStripMenuItem.Enabled = true;
                    rolePlayingGame = frmNewGame.RolePlayingGame;
                }
            }
        }

        void openGameToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        void saveGameToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        void classesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (frmClasses == null)
            {
                frmClasses = new FormClasses();
                frmClasses.MdiParent = this;
            }
            frmClasses.Show();
        }

        #endregion

        #region Method Region
        #endregion
    }
}
```

There is a field of type **RolePlayingGame** to hold the **RolePlayingGame** object associated with the editor. There is also a field **FormClasses** for the form that holds all of the **EntityData** objects. The constructor wires the event handlers for all of the menu items.

The handler for the **New Game** menu item has a using statement that creates an instance of **FormNewGame**. Inside the using statement I call the **ShowDialog** method of the form and capture the result in the variable **result**. If **result** is **OK**, meaning the form was closed by clicking **OK**, and the **RolePlayingGame** property of the form is not null I set the **Classes** menu item to **Enabled** and the **rolePlayingGame** field to the **RolePlayingGame** property of **FormNewGame**. In the **Exit** menu item handler I just call **Close** on the form to close the form. Later I will add in code to make sure that data is saved before closing the form.

The handler for the **Classes** menu item checks to see if **frmClasses** is null. If it is then it has not been created yet so I create it. I also set the **MdiParent** property of the form to be the current instance of **FormMain** using **this**. After creating the form if necessary I call the **Show** method rather than **ShowDialog**. This allows you to flip between child forms of the parent form easily.

Part 11 Continued
Game Editors

I'm going to be continuing on with game editors in this tutorial. Since I've moved to a dynamic class system a few updates need to be made to class in the **ItemClasses** of the **RpgLibrary** project. They were using **Type** values for allowable classes. Instead I can use string values. The classes need to be updated to use string instead of **Type**. The code for the **BaseItem**, **Weapon**, **Armor**, and **Shield** classes follows next.

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

namespace RpgLibrary.ItemClasses
{
    public enum Hands { Main, Off, Both }

    public enum ArmorLocation { Body, Head, Hands, Feet }

    public abstract class BaseItem
    {
```

```
#region Field Region

protected List<string> allowableClasses = new List<string>();
string name;
string type;
int price;
float weight;
bool equipped;

#endregion

#region Property Region

public List<string> AllowableClasses
{
    get { return allowableClasses; }
    protected set { allowableClasses = value; }
}

public string Type
{
    get { return type; }
    protected set { type = value; }
}

public string Name
{
    get { return name; }
    protected set { name = value; }
}

public int Price
{
    get { return price; }
    protected set { price = value; }
}

public float Weight
{
    get { return weight; }
    protected set { weight = value; }
}

public bool IsEquiped
{
    get { return equipped; }
    protected set { equipped = value; }
}

#endregion

#region Constructor Region

public BaseItem(
    string name,
    string type,
    int price,
    float weight,
    params string[] allowableClasses)
{
```

```
            foreach (string t in allowableClasses)
                AllowableClasses.Add(t);

            Name = name;
            Type = type;
            Price = price;
            Weight = weight;
            IsEquiped = false;
        }

        #endregion

        #region Abstract Method Region

        public abstract object Clone();

        public virtual bool CanEquip(string characterType)
        {
            return allowableClasses.Contains(characterType);
        }

        public override string ToString()
        {
            string itemString = "";
            itemString += Name + ", ";
            itemString += Type + ", ";
            itemString += Price.ToString() + ", ";
            itemString += Weight.ToString();
            return itemString;
        }

        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RpgLibrary.ItemClasses
{
    public class Armor : BaseItem
    {
        #region Field Region

        ArmorLocation location;
        int defenseValue;
        int defenseModifier;

        #endregion

        #region Property Region
        public ArmorLocation Location
        {
            get { return location; }
            protected set { location = value; }
        }

        public int DefenseValue
```

```
{
    get { return defenseValue; }
    protected set { defenseValue = value; }
}

public int DefenseModifier
{
    get { return defenseModifier; }
    protected set { defenseModifier = value; }
}

#endregion

#region Constructor Region

public Armor(
    string armorName,
    string armorType,
    int price,
    float weight,
    ArmorLocation locaion,
    int defenseValue,
    int defenseModifier,
    params string[] allowableClasses)
    : base(armorName, armorType, price, weight, allowableClasses)
{
    Location = location;
    DefenseValue = defenseValue;
    DefenseModifier = defenseModifier;
}

#endregion

#region Abstract Method Region

public override object Clone()
{
    string[] allowedClasses = new string[allowableClasses.Count];

    for (int i = 0; i < allowableClasses.Count; i++)
        allowedClasses[i] = allowableClasses[i];

    Armor armor = new Armor(
        Name,
        Type,
        Price,
        Weight,
        Location,
        DefenseValue,
        DefenseModifier,
        allowedClasses);

    return armor;
}

public override string ToString()
{
    string armorString = base.ToString() + ", ";

    armorString += Location.ToString() + ", ";
```

```
            armorString += DefenseValue.ToString() + ", ";
            armorString += DefenseModifier.ToString();

            foreach (string t in allowableClasses)
                armorString += ", " + t;

            return armorString;
        }
        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RpgLibrary.ItemClasses
{
    public class Shield : BaseItem
    {
        #region Field Region

        int defenseValue;
        int defenseModifier;

        #endregion

        #region Property Region

        public int DefenseValue
        {
            get { return defenseValue; }
            protected set { defenseValue = value; }
        }

        public int DefenseModifier
        {
            get { return defenseModifier; }
            protected set { defenseModifier = value; }
        }

        #endregion

        #region Constructor Region

        public Shield(
            string shieldName,
            string shieldType,
            int price,
            float weight,
            int defenseValue,
            int defenseModifier,
            params string[] allowableClasses)
            : base(shieldName, shieldType, price, weight, allowableClasses)
        {
            DefenseValue = defenseValue;
            DefenseModifier = defenseModifier;
        }
        #endregion

        #region Abstract Method Region

        public override object Clone()
        {
            string[] allowedClasses = new string[allowableClasses.Count];

            for (int i = 0; i < allowableClasses.Count; i++)
                allowedClasses[i] = allowableClasses[i];

            Shield shield = new Shield(
                Name,
                Type,
                Price,
                Weight,
                DefenseValue,
                DefenseModifier,
                allowedClasses);

            return shield;
        }
        public override string ToString()
        {
            string shieldString = base.ToString() + ", ";

            shieldString += DefenseValue.ToString() + ", ";
            shieldString += DefenseModifier.ToString();

            foreach (string t in allowableClasses)
                shieldString += ", " + t;

            return shieldString;
        }
        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RpgLibrary.ItemClasses
{
    public class Weapon : BaseItem
    {
        #region Field Region

        Hands hands;
        int attackValue;
        int attackModifier;
        int damageValue;
        int damageModifier;

        #endregion

        #region Property Region

        public Hands NumberHands
```

```
{
    get { return hands; }
    protected set { hands = value; }
}

public int AttackValue
{
    get { return attackValue; }
    protected set { attackValue = value; }
}

public int AttackModifier
{
    get { return attackModifier; }
    protected set { attackModifier = value; }
}

public int DamageValue
{
    get { return damageValue; }
    protected set { damageValue = value; }
}

public int DamageModifier
{
    get { return damageModifier; }
    protected set { damageModifier = value; }
}

#endregion

#region Constructor Region

public Weapon(
    string weaponName,
    string weaponType,
    int price,
    float weight,
    Hands hands,
    int attackValue,
    int attackModifier,
    int damageValue,
    int damageModifier,
    params string[] allowableClasses)
    : base(weaponName, weaponType, price, weight, allowableClasses)
{
    NumberHands = hands;
    AttackValue = attackValue;
    AttackModifier = attackModifier;
    DamageValue = damageValue;
    DamageModifier = damageModifier;
}

#endregion

#region Abstract Method Region

public override object Clone()
{
    string[] allowedClasses = new string[allowableClasses.Count];
```

```
            for (int i = 0; i < allowableClasses.Count; i++)
                allowedClasses[i] = allowableClasses[i];

            Weapon weapon = new Weapon(
                Name,
                Type,
                Price,
                Weight,
                NumberHands,
                AttackValue,
                AttackModifier,
                DamageValue,
                DamageModifier,
                allowedClasses);

            return weapon;
        }
        public override string ToString()
        {
            string weaponString = base.ToString() + ", ";

            weaponString += NumberHands.ToString() + ", ";
            weaponString += AttackValue.ToString() + ", ";
            weaponString += AttackModifier.ToString() + ", ";
            weaponString += DamageValue.ToString() + ", ";
            weaponString += DamageModifier.ToString();

            foreach (string t in allowableClasses)
                weaponString += ", " + t;

            return base.ToString();
        }

        #endregion
    }
}
```

The next step is to add in items to the editor. First open the design view for **FormMain** by right clicking it and selecting **View Designer**. Beside the **Classes** menu item add the following entry **&Items**. Set the **Enabled** property to false. Under **&Items** you want to add **&Weapons**, **&Armor**, and **&Shield**.

A lot of forms are going to use the same layout as **FormClasses**. So I'm going to create a master form called **FormDetails**. Any form that has that layout can inherit from **FormDetails** instead of **Form**. Right click the **RpgEditor** project in the solution explorer, select **Add** and then **Windows Form**. Name this form **FormDetails**. Set the **Size** property of **FormDetails** to the size of **FormClasses**. Go back to the design view of **FormClasses**. While holding down the **Ctrl** key click on the **Menu Strip**, **List Box**, and the three **Buttons**.

Now, copy the controls by pressing **Ctrl-C**. Switch to **FormDetails** and press **Ctrl-V** to paste the controls onto the form. Click on the title bar of the form to deselect the other controls. Rename **lbClasses** to **lbDetails**. Set the **Anchor** property of **lbDetails** to **Top**, **Left, Bottom, Right**. Click on the arrow pointing at **FormDetails** to expand the entries under it. Bring up the code for **FormDetails.Designer.cs** by right clicking it and selecting **View Code**. Change these fields to protected rather than private.

```
protected System.Windows.Forms.Button btnDelete;
protected System.Windows.Forms.Button btnEdit;
protected System.Windows.Forms.Button btnAdd;
protected System.Windows.Forms.ListBox lbDetails;
protected System.Windows.Forms.MenuStrip menuStrip1;
protected System.Windows.Forms.ToolStripMenuItem loadToolStripMenuItem;
protected System.Windows.Forms.ToolStripMenuItem saveToolStripMenuItem;
```

There are two more things I want to do before leaving the details form. I want to add a static protected field for an **ItemManager**. I also want to move the **EntityDataManager** from **FormClasses** to **FormDetails**. This way they will be available to any child form that needs to work with items and entity data. As the game progresses there will be more manager classes that will be needed. Change the code of **FormDetails** to the following.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.ItemClasses;
using RpgLibrary.CharacterClasses;

namespace RpgEditor
{
    public partial class FormDetails : Form
    {
        #region Field Region

        protected static ItemManager ItemManager;
        protected static EntityDataManager EntityDataManager;

        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormDetails()
        {
            InitializeComponent();

            if (FormDetails.ItemManager == null)
                ItemManager = new ItemManager();

            if (FormDetails.EntityDataManager == null)
                EntityDataManager = new EntityDataManager();
        }

        #endregion
    }
}
```

The constructor checks to see if the fields are null. If they are null it creates a new instance of the fields. Since the fields are static you can't reference them using this. You need to reference them using the class name.

Go to the **Design View** of **FormClasses** by right clicking it in the solution explorer and selecting **View Designer**. Remove all of the controls that were on the form. Right click **FormClasses** again and select **View Code** to bring up the code for the form. Change the code for **FormClasses** to the following.

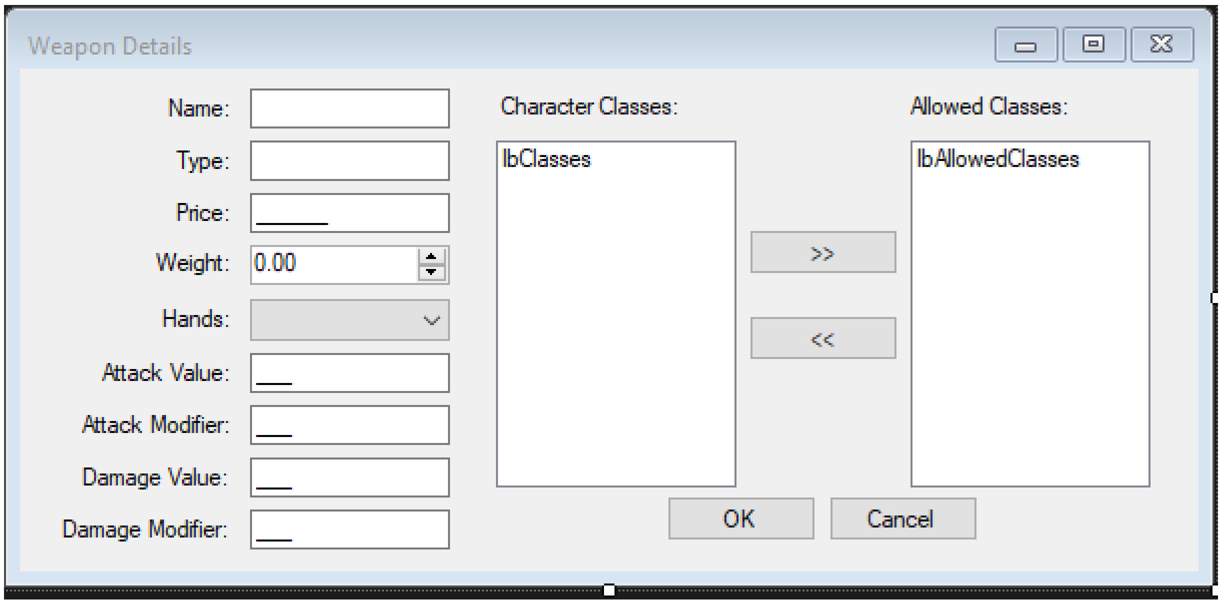
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.CharacterClasses;

namespace RpgEditor
{
    public partial class FormClasses : FormDetails
    {
        #region Field Region
        #endregion

        #region Constructor Region
        public FormClasses()
        {
            InitializeComponent();

            loadToolStripMenuItem.Click += new EventHandler(loadToolStripMenuItem_Click);
            saveToolStripMenuItem.Click += new EventHandler(saveToolStripMenuItem_Click);

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Menu Item Event Handler Region

        void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
            using (FormEntityData frmEntityData = new FormEntityData())
            {
                frmEntityData.ShowDialog();
                if (frmEntityData.EntityData != null)
                {
                    lbDetails.Items.Add(frmEntityData.EntityData.ToString());
                }
            }
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
        }

        #endregion
    }
}
```

The changes are that I now inherit from **FormDetails** instead of **Form**, the **EntityDataManager** field was removed form the class, and in the **Click** event handler of **btnAdd** I use **lbDetails** instead of **lbClasses**. If you go back to the design view of **FormClasses** all of the controls should be there with little blue arrows in the top corner. Set the **Anchor** property of **lbDetails** to **Top**, **Left**, **Bottom**, **Right**.

I'm going to add in the forms for weapons, armor, and shields now. Right click the **RpgEditor** solution, select **Add** and then **Windows Form**. Call this new form **FormWeapon**. Repeat the process and name the new forms **FormShield** and **FormArmor**. Set the **Text** property of **FormWeapon** to **Weapons**. For **FormShield** set the **Text** property to **Shield** and for **FormArmor** set **Text** to **Armor**. Make each of the forms the size of your **FormDetails**. Set the **Anchor** property of **lbDetails** on all of the forms to **Top**, **Left**, **Bottom**, **Right**. I added in code skeletons for each of the forms as well. The code follows next in the following order: **FormWeapon**, **FormShield**, and **FormArmor**.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RpgEditor
{
    public partial class FormWeapon : FormDetails
    {
        #region Field Region
        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormWeapon()
        {
            InitializeComponent();

            loadToolStripMenuItem.Click += new EventHandler(loadToolStripMenuItem_Click);
            saveToolStripMenuItem.Click += new EventHandler(saveToolStripMenuItem_Click);

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Menu Item Event Handler Region

        void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }
        void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
        }

        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RpgEditor
{
    public partial class FormArmor : FormDetails
    {
        #region Field Region
        #endregion

        #region Property Region
        #endregion
        #region Constructor Region

        public FormArmor()
        {
            InitializeComponent();

            loadToolStripMenuItem.Click += new EventHandler(loadToolStripMenuItem_Click);
            saveToolStripMenuItem.Click += new EventHandler(saveToolStripMenuItem_Click);

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Menu Item Event Handler Region

        void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }
        void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
        }

        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RpgEditor
{
    public partial class FormShield : FormDetails
    {
        #region Field Region
        #endregion
        #region Property Region
        #endregion

        #region Constructor Region

        public FormShield()
        {
            InitializeComponent();

            loadToolStripMenuItem.Click += new EventHandler(loadToolStripMenuItem_Click);
            saveToolStripMenuItem.Click += new EventHandler(saveToolStripMenuItem_Click);

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Menu Item Event Handler Region

        void loadToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }
        void saveToolStripMenuItem_Click(object sender, EventArgs e)
        {
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
        }

        #endregion
    }
}
```

I'm going to add some details forms for specific item types. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormWeaponDetails**. My finished form looked like below. Set its **Text** property to **Weapon Details** and its **FormBorderStyle** to **FixedDialog**.



There are a lot of controls on the form, an alternative method follows. First make your form bigger to hold all of the controls. Drag a **Label** on and set its **Text** property to **Name:**. Drag a **Text Box** on to the form and position it beside the **Label**. Set the **Name** property of the **Text Box** to **tbName**. Drag a **Label** onto the form and position it under the **Name: Label**. Set the **Text** property to **Type:**. Drag a **Text Box** to the right of that **Label**. Set its **Name** property to **tbType**. Add another label under the **Type: Label** and set its **Text** property to **Price:**. Position a **Masked Text Box** to the right of that label. Resize it to match the other **Text Boxes**. Set its **Name** property to **mtbPrice** and its **Mask** property to **00000**. Drag another **Label** under the **P**rice: label and set its **Text** property to **Weight:**. Drag a **Numeric Up Down** beside that **Label**. Set its **Name** property to **nudWeight**, the **Decimal Places** to **2**. Resize it so it lines up with the other controls.

Drag a **Label** on and set the **Text** property to **Hands:**. Drag a **Combo Box** beside that and size it so that it matches the other controls. Set its **Name** property to **cboHands** and the **DropDownStyle** to **Drop Down List**. Drag another **Label** on and set its **Text** property to **Attack Value:** and a **Masked Text Box** beside it and set **Name** property to **mtbAttackValue**. Resize the control so that it matches the others and set its **Mask** property to **000**. While holding down the **ctrl** key select that **Label** and **Masked Text Box**. While still holding the **ctrl** key drag the controls down. That will replicated the controls. Set the **Text** property of the label to **Attack Modifier:** and the **Name** property of the **Masked Text Box** to **mtbAttackModifier**. Repeat the process twice more. For the first **Label** set its **Text** property to **Damage Value:** and for the second **Damage Modifier:**. Set the **Name** property of the first **Masked Text Box** to **mtbDamageValue** and the second to **mtbDamageModifier**. Drag a **Label** and position it to the right of **tbName**. Set its **Text** property to **Character Classes:**. Drag a **List Box** and position it under the **Character Classes: Label**. Make it longer so there is enough room for a **Button** below it and set its **Name** property to **lbClasses**. Drag two buttons to the right of that **List Box**. Position them sort of centered vertically. Set the **Text** property of the first **Button** to **>>** and the **Name** property to **btnMoveAllowed**. Set the **Text** property of the second **Button** to **<<** and the **Name** property to **btnRemoveAllowed**. Drag a **List Box** and position it to the right of the two **Buttons**. Set its size and position so that it matches **lbClasses**. Set its **Name** property to **lbAllowedClasses**. Drag two more buttons and position them below the **List Boxes**. Set the **Name** property of the first **Button** to **btnOK** and its **Text** property to **OK**. Set the **Name** property of the second to **btnCancel** and the **Text** property to **Cancel**.

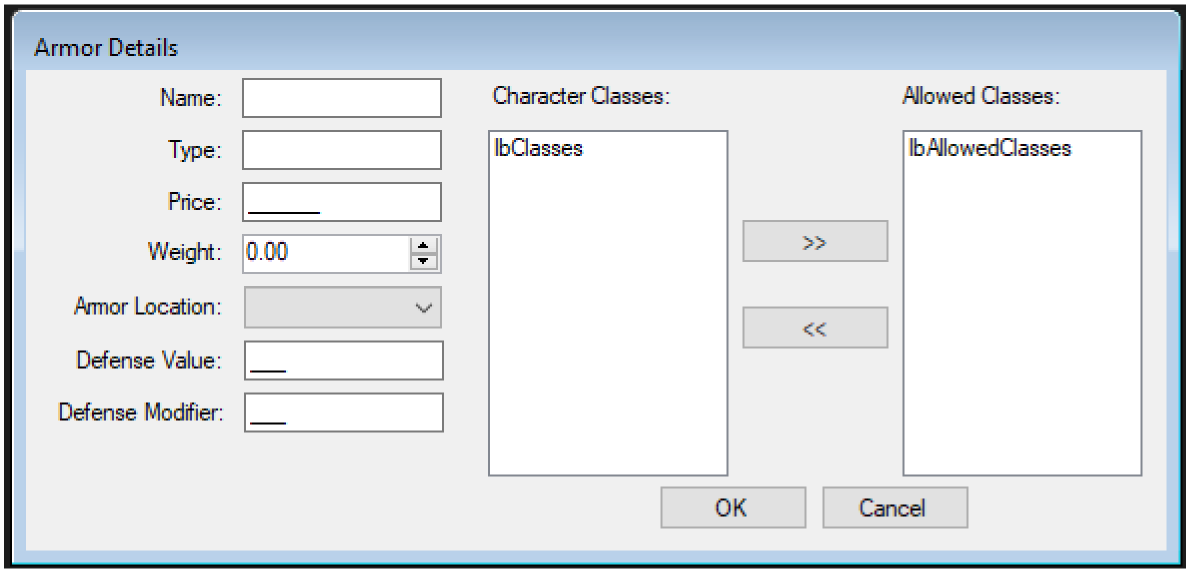
If dragging on the controls was too much work you can alter the **FormWeaponDetail.Designer.cs** file to add the controls. It is just a lot of code.

```
namespace RpgEditor
{
    partial class FormWeaponDetail
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.tbName = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.tbType = new System.Windows.Forms.TextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.mtbPrice = new System.Windows.Forms.MaskedTextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.nudWeight = new System.Windows.Forms.NumericUpDown();
            this.label5 = new System.Windows.Forms.Label();
            this.cboHands = new System.Windows.Forms.ComboBox();
            this.label6 = new System.Windows.Forms.Label();
            this.mtbAttackValue = new System.Windows.Forms.MaskedTextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.mtbAttackModifier = new System.Windows.Forms.MaskedTextBox();
            this.label7 = new System.Windows.Forms.Label();
            this.mtbDamageValue = new System.Windows.Forms.MaskedTextBox();
            this.label8 = new System.Windows.Forms.Label();
            this.mtbDamageModifier = new System.Windows.Forms.MaskedTextBox();
            this.lbClasses = new System.Windows.Forms.ListBox();
            this.label9 = new System.Windows.Forms.Label();
            this.lbAllowedClasses = new System.Windows.Forms.ListBox();
            this.label10 = new System.Windows.Forms.Label();
            this.btnMoveAllowed = new System.Windows.Forms.Button();
            this.btnRemoveAllowed = new System.Windows.Forms.Button();
            this.btnOK = new System.Windows.Forms.Button();
            this.btnCancel = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.nudWeight)).BeginInit();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(71, 13);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(38, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Name:";
            //
            // tbName
            //
            this.tbName.Location = new System.Drawing.Point(115, 10);
            this.tbName.Name = "tbName";
            this.tbName.Size = new System.Drawing.Size(100, 20);
            this.tbName.TabIndex = 1;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(75, 39);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(34, 13);
            this.label2.TabIndex = 2;
            this.label2.Text = "Type:";
            //
            // tbType
            //
            this.tbType.Location = new System.Drawing.Point(115, 36);
            this.tbType.Name = "tbType";
            this.tbType.Size = new System.Drawing.Size(100, 20);
            this.tbType.TabIndex = 3;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(75, 65);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(34, 13);
            this.label3.TabIndex = 4;
            this.label3.Text = "Price:";
            //
            // mtbPrice
            //
            this.mtbPrice.Location = new System.Drawing.Point(115, 62);
            this.mtbPrice.Mask = "000000";
            this.mtbPrice.Name = "mtbPrice";
            this.mtbPrice.Size = new System.Drawing.Size(100, 20);
            this.mtbPrice.TabIndex = 5;
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(65, 90);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(44, 13);
            this.label4.TabIndex = 6;
            this.label4.Text = "Weight:";
            //
            // nudWeight
            //
            this.nudWeight.DecimalPlaces = 2;
            this.nudWeight.Location = new System.Drawing.Point(115, 88);
            this.nudWeight.Name = "nudWeight";
            this.nudWeight.Size = new System.Drawing.Size(100, 20);
            this.nudWeight.TabIndex = 7;
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(68, 118);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(41, 13);
            this.label5.TabIndex = 8;
            this.label5.Text = "Hands:";
            //
            // cboHands
            //
            this.cboHands.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboHands.FormattingEnabled = true;
            this.cboHands.Location = new System.Drawing.Point(115, 115);
            this.cboHands.Name = "cboHands";
            this.cboHands.Size = new System.Drawing.Size(100, 21);
            this.cboHands.TabIndex = 9;
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(38, 145);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(71, 13);
            this.label6.TabIndex = 10;
            this.label6.Text = "Attack Value:";
            //
            // mtbAttackValue
            //
            this.mtbAttackValue.Location = new System.Drawing.Point(115, 142);
            this.mtbAttackValue.Mask = "000";
            this.mtbAttackValue.Name = "mtbAttackValue";
            this.mtbAttackValue.Size = new System.Drawing.Size(100, 20);
            this.mtbAttackValue.TabIndex = 11;
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Location = new System.Drawing.Point(28, 171);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(81, 13);
            this.label11.TabIndex = 10;
            this.label11.Text = "Attack Modifier:";
            //
            // mtbAttackModifier
            //
```

```
this.mtbAttackModifier.Location = new System.Drawing.Point(115, 168);
this.mtbAttackModifier.Mask = "000";
this.mtbAttackModifier.Name = "mtbAttackModifier";
this.mtbAttackModifier.Size = new System.Drawing.Size(100, 20);
this.mtbAttackModifier.TabIndex = 11;
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(28, 197);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(80, 13);
this.label7.TabIndex = 10;
this.label7.Text = "Damage Value:";
//
// mtbDamageValue
//
this.mtbDamageValue.Location = new System.Drawing.Point(115, 194);
this.mtbDamageValue.Mask = "000";
this.mtbDamageValue.Name = "mtbDamageValue";
this.mtbDamageValue.Size = new System.Drawing.Size(100, 20);
this.mtbDamageValue.TabIndex = 11;
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(18, 223);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(90, 13);
this.label8.TabIndex = 10;
this.label8.Text = "Damage Modifier:";
//
// mtbDamageModifier
//
this.mtbDamageModifier.Location = new System.Drawing.Point(115, 220);
this.mtbDamageModifier.Mask = "000";
this.mtbDamageModifier.Name = "mtbDamageModifier";
this.mtbDamageModifier.Size = new System.Drawing.Size(100, 20);
this.mtbDamageModifier.TabIndex = 11;
//
// lbClasses
//
this.lbClasses.FormattingEnabled = true;
this.lbClasses.Location = new System.Drawing.Point(238, 36);
this.lbClasses.Name = "lbClasses";
this.lbClasses.Size = new System.Drawing.Size(120, 173);
this.lbClasses.TabIndex = 12;
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(238, 12);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(95, 13);
this.label9.TabIndex = 13;
this.label9.Text = "Character Classes:";
//
// lbAllowedClasses
//
this.lbAllowedClasses.FormattingEnabled = true;
this.lbAllowedClasses.Location = new System.Drawing.Point(445, 36);
this.lbAllowedClasses.Name = "lbAllowedClasses";
this.lbAllowedClasses.Size = new System.Drawing.Size(120, 173);
this.lbAllowedClasses.TabIndex = 12;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(442, 12);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(86, 13);
this.label10.TabIndex = 13;
this.label10.Text = "Allowed Classes:";
//
// btnMoveAllowed
//
this.btnMoveAllowed.Location = new System.Drawing.Point(364, 80);
this.btnMoveAllowed.Name = "btnMoveAllowed";
this.btnMoveAllowed.Size = new System.Drawing.Size(75, 23);
this.btnMoveAllowed.TabIndex = 14;
this.btnMoveAllowed.Text = ">>";
this.btnMoveAllowed.UseVisualStyleBackColor = true;
//
// btnRemoveAllowed
//
this.btnRemoveAllowed.Location = new System.Drawing.Point(364, 123);
this.btnRemoveAllowed.Name = "btnRemoveAllowed";
this.btnRemoveAllowed.Size = new System.Drawing.Size(75, 23);
this.btnRemoveAllowed.TabIndex = 14;
this.btnRemoveAllowed.Text = "<<";
this.btnRemoveAllowed.UseVisualStyleBackColor = true;
//
// btnOK
//
this.btnOK.Location = new System.Drawing.Point(323, 213);
this.btnOK.Name = "btnOK";
this.btnOK.Size = new System.Drawing.Size(75, 23);
this.btnOK.TabIndex = 15;
this.btnOK.Text = "OK";
this.btnOK.UseVisualStyleBackColor = true;
//
// btnCancel
//
this.btnCancel.Location = new System.Drawing.Point(404, 213);
this.btnCancel.Name = "btnCancel";
this.btnCancel.Size = new System.Drawing.Size(75, 23);
this.btnCancel.TabIndex = 15;
this.btnCancel.Text = "Cancel";
this.btnCancel.UseVisualStyleBackColor = true;
//
// FormWeaponDetail
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(589, 251);
this.Controls.Add(this.btnCancel);
this.Controls.Add(this.btnOK);
this.Controls.Add(this.btnRemoveAllowed);
this.Controls.Add(this.btnMoveAllowed);
this.Controls.Add(this.label10);
```

```
            this.Controls.Add(this.label9);
            this.Controls.Add(this.lbAllowedClasses);
            this.Controls.Add(this.lbClasses);
            this.Controls.Add(this.mtbDamageModifier);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.mtbDamageValue);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.mtbAttackModifier);
            this.Controls.Add(this.label11);
            this.Controls.Add(this.mtbAttackValue);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.cboHands);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.nudWeight);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.mtbPrice);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.tbType);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.tbName);
            this.Controls.Add(this.label1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.Name = "FormWeaponDetail";
            this.Text = "Weapon Details";
            ((System.ComponentModel.ISupportInitialize)(this.nudWeight)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox tbName;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox tbType;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.MaskedTextBox mtbPrice;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.NumericUpDown nudWeight;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.ComboBox cboHands;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.MaskedTextBox mtbAttackValue;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.MaskedTextBox mtbAttackModifier;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.MaskedTextBox mtbDamageValue;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.MaskedTextBox mtbDamageModifier;
        private System.Windows.Forms.ListBox lbClasses;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.ListBox lbAllowedClasses;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Button btnMoveAllowed;
        private System.Windows.Forms.Button btnRemoveAllowed;
        private System.Windows.Forms.Button btnOK;
        private System.Windows.Forms.Button btnCancel;
    }
}
```

The next form to add is a form for the various types of armor in the game. I will again give you the option of designing the form or copying/pasting the **Designer.cs** file. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormArmorDetails**. Set the **Text** property to **Armor Details** and the **FormBorderStyle** to **FixedDialog**. My form looked like below. Instead of adding all of the controls again, first make the form a big enough so that it will easily fit all of the controls. Go back to the design view of **FormWeaponDetail**. Hold down the **ctrl** key and click all of the controls except the **Combo Box** and the controls related attack. Press **ctrl+C** to copy the controls. Go back to **FormArmorDetail** and press **ctrl+V** to paste the controls. Move them until they fit nicely on the form. Drag a **Label** below the **Weight** label and set its **Text** property to **Armor Location**:. Drag a **Combo Box** box beside that **Label**. Set its **Name** property to **cboArmorLocation** and the **DropDownStyle** to **DropDownList**. Drag a **Label** below **Armor Location:** and set its **Text** property to **Defense Value:**. Drag a **Masked Text Box** beside it and set its **Mask** property to **000** and its **Name** property to **mtbDefenseValue**. While holding down the **ctrl** key select the **Label** and **Masked Text Box** you just added. Still holding down the **ctrl** key drag them down to replicate them. Set the **Text** property of the **Label** to **Defense Modifier:**. For the **Masked Text Box** set the **Name** property to **mtbDefenseModifier**.



In case you're more comfortable copy/pasting code the code for the designer follows next.

```
namespace RpgEditor
{
    partial class FormArmorDetails
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
```

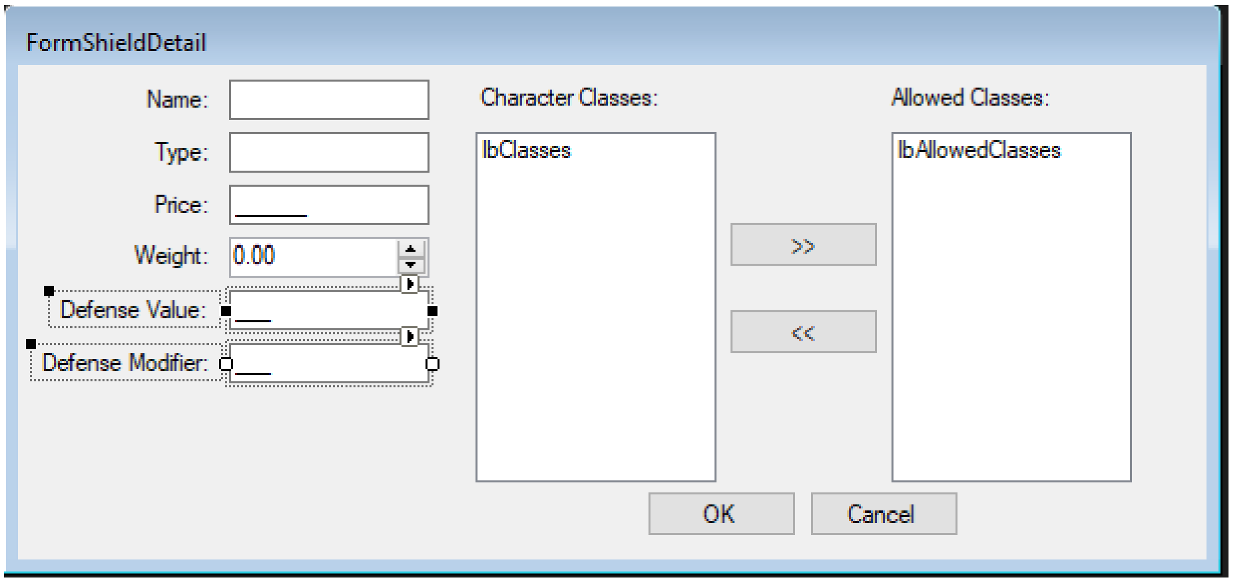
```
protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#region Windows Form Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.btnCancel = new System.Windows.Forms.Button();
    this.btnOK = new System.Windows.Forms.Button();
    this.btnRemoveAllowed = new System.Windows.Forms.Button();
    this.btnMoveAllowed = new System.Windows.Forms.Button();
    this.label10 = new System.Windows.Forms.Label();
    this.label9 = new System.Windows.Forms.Label();
    this.lbAllowedClasses = new System.Windows.Forms.ListBox();
    this.lbClasses = new System.Windows.Forms.ListBox();
    this.nudWeight = new System.Windows.Forms.NumericUpDown();
    this.label4 = new System.Windows.Forms.Label();
    this.mtbPrice = new System.Windows.Forms.MaskedTextBox();
    this.label3 = new System.Windows.Forms.Label();
    this.tbType = new System.Windows.Forms.TextBox();
    this.label2 = new System.Windows.Forms.Label();
    this.tbName = new System.Windows.Forms.TextBox();
    this.label1 = new System.Windows.Forms.Label();
    this.label5 = new System.Windows.Forms.Label();
    this.cboArmorLocation = new System.Windows.Forms.ComboBox();
    this.label6 = new System.Windows.Forms.Label();
    this.mtbDefenseValue = new System.Windows.Forms.MaskedTextBox();
    this.label7 = new System.Windows.Forms.Label();
    this.mtbDefenseModifier = new System.Windows.Forms.MaskedTextBox();
    ((System.ComponentModel.ISupportInitialize)(this.nudWeight)).BeginInit();
    this.SuspendLayout();
    //
    // btnCancel
    //
    this.btnCancel.Location = new System.Drawing.Point(397, 207);
    this.btnCancel.Name = "btnCancel";
    this.btnCancel.Size = new System.Drawing.Size(75, 23);
    this.btnCancel.TabIndex = 30;
    this.btnCancel.Text = "Cancel";
    this.btnCancel.UseVisualStyleBackColor = true;
    //
    // btnOK
    //
    this.btnOK.Location = new System.Drawing.Point(316, 207);
    this.btnOK.Name = "btnOK";
    this.btnOK.Size = new System.Drawing.Size(75, 23);
    this.btnOK.TabIndex = 31;
    this.btnOK.Text = "OK";
    this.btnOK.UseVisualStyleBackColor = true;
    //
```

```
// btnRemoveAllowed
//
this.btnRemoveAllowed.Location = new System.Drawing.Point(357, 117);
this.btnRemoveAllowed.Name = "btnRemoveAllowed";
this.btnRemoveAllowed.Size = new System.Drawing.Size(75, 23);
this.btnRemoveAllowed.TabIndex = 28;
this.btnRemoveAllowed.Text = "<<";
this.btnRemoveAllowed.UseVisualStyleBackColor = true;
//
// btnMoveAllowed
//
this.btnMoveAllowed.Location = new System.Drawing.Point(357, 74);
this.btnMoveAllowed.Name = "btnMoveAllowed";
this.btnMoveAllowed.Size = new System.Drawing.Size(75, 23);
this.btnMoveAllowed.TabIndex = 29;
this.btnMoveAllowed.Text = ">>";
this.btnMoveAllowed.UseVisualStyleBackColor = true;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(435, 6);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(86, 13);
this.label10.TabIndex = 26;
this.label10.Text = "Allowed Classes:";
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(231, 6);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(95, 13);
this.label9.TabIndex = 27;
this.label9.Text = "Character Classes:";
//
// lbAllowedClasses
//
this.lbAllowedClasses.FormattingEnabled = true;
this.lbAllowedClasses.Location = new System.Drawing.Point(438, 30);
this.lbAllowedClasses.Name = "lbAllowedClasses";
this.lbAllowedClasses.Size = new System.Drawing.Size(120, 173);
this.lbAllowedClasses.TabIndex = 24;
//
// lbClasses
//
this.lbClasses.FormattingEnabled = true;
this.lbClasses.Location = new System.Drawing.Point(231, 30);
this.lbClasses.Name = "lbClasses";
this.lbClasses.Size = new System.Drawing.Size(120, 173);
this.lbClasses.TabIndex = 25;
//
// nudWeight
//
this.nudWeight.DecimalPlaces = 2;
this.nudWeight.Location = new System.Drawing.Point(108, 82);
this.nudWeight.Name = "nudWeight";
this.nudWeight.Size = new System.Drawing.Size(100, 20);
this.nudWeight.TabIndex = 23;
//
```

```
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(58, 84);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(44, 13);
this.label4.TabIndex = 22;
this.label4.Text = "Weight:";
//
// mtbPrice
//
this.mtbPrice.Location = new System.Drawing.Point(108, 56);
this.mtbPrice.Mask = "000000";
this.mtbPrice.Name = "mtbPrice";
this.mtbPrice.Size = new System.Drawing.Size(100, 20);
this.mtbPrice.TabIndex = 21;
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(68, 59);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(34, 13);
this.label3.TabIndex = 20;
this.label3.Text = "Price:";
//
// tbType
//
this.tbType.Location = new System.Drawing.Point(108, 30);
this.tbType.Name = "tbType";
this.tbType.Size = new System.Drawing.Size(100, 20);
this.tbType.TabIndex = 19;
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(68, 33);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(34, 13);
this.label2.TabIndex = 18;
this.label2.Text = "Type:";
//
// tbName
//
this.tbName.Location = new System.Drawing.Point(108, 4);
this.tbName.Name = "tbName";
this.tbName.Size = new System.Drawing.Size(100, 20);
this.tbName.TabIndex = 17;
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(64, 7);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(38, 13);
this.label1.TabIndex = 16;
this.label1.Text = "Name:";
//
// label5
//
```

```
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(21, 111);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(81, 13);
            this.label5.TabIndex = 32;
            this.label5.Text = "Armor Location:";
            //
            // cboArmorLocation
            //
            this.cboArmorLocation.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cboArmorLocation.FormattingEnabled = true;
            this.cboArmorLocation.Location = new System.Drawing.Point(109, 108);
            this.cboArmorLocation.Name = "cboArmorLocation";
            this.cboArmorLocation.Size = new System.Drawing.Size(99, 21);
            this.cboArmorLocation.TabIndex = 33;
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(22, 138);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(80, 13);
            this.label6.TabIndex = 34;
            this.label6.Text = "Defense Value:";
            //
            // mtbDefenseValue
            //
            this.mtbDefenseValue.Location = new System.Drawing.Point(109, 135);
            this.mtbDefenseValue.Mask = "000";
            this.mtbDefenseValue.Name = "mtbDefenseValue";
            this.mtbDefenseValue.Size = new System.Drawing.Size(100, 20);
            this.mtbDefenseValue.TabIndex = 35;
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(13, 164);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(90, 13);
            this.label7.TabIndex = 34;
            this.label7.Text = "Defense Modifier:";
            //
            // mtbDefenseModifier
            //
            this.mtbDefenseModifier.Location = new System.Drawing.Point(109, 161);
            this.mtbDefenseModifier.Mask = "000";
            this.mtbDefenseModifier.Name = "mtbDefenseModifier";
            this.mtbDefenseModifier.Size = new System.Drawing.Size(100, 20);
            this.mtbDefenseModifier.TabIndex = 35;
            //
            // FormArmorDetails
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(570, 240);
            this.ControlBox = false;
            this.Controls.Add(this.mtbDefenseModifier);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.mtbDefenseValue);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.cboArmorLocation);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.btnCancel);
            this.Controls.Add(this.btnOK);
            this.Controls.Add(this.btnRemoveAllowed);
            this.Controls.Add(this.btnMoveAllowed);
            this.Controls.Add(this.label10);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.lbAllowedClasses);
            this.Controls.Add(this.lbClasses);
            this.Controls.Add(this.nudWeight);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.mtbPrice);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.tbType);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.tbName);
            this.Controls.Add(this.label1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.Name = "FormArmorDetails";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Armor Details";
            ((System.ComponentModel.ISupportInitialize)(this.nudWeight)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button btnCancel;
        private System.Windows.Forms.Button btnOK;
        private System.Windows.Forms.Button btnRemoveAllowed;
        private System.Windows.Forms.Button btnMoveAllowed;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.ListBox lbAllowedClasses;
        private System.Windows.Forms.ListBox lbClasses;
        private System.Windows.Forms.NumericUpDown nudWeight;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.MaskedTextBox mtbPrice;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox tbType;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox tbName;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.ComboBox cboArmorLocation;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.MaskedTextBox mtbDefenseValue;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.MaskedTextBox mtbDefenseModifier;
    }
}
```

That just leaves the the form for shields. Right click the **RpgEditor**, select **Add** and then **Windows Form**. Name this new form **FormShieldDetail**. Set the **Text** property of the form to **Shield Details** and the **FormBorderStyle** to **FixedDialog**. My finished form is next.



There is no need to add all of the controls to this form as well. Switch to the design view of **FormArmorDetail**. You can either click on all of the controls while holding down the **ctrl** key or you can drag a rectangle around all of the controls. Once you have all of the controls selected press **ctrl+C** to copy them. Go back to the design view of **FormShieldDetail** and press **ctrl+V** to paste the controls. Click on the **Armor Location: Label** and the **cboArmorLocation Combo Box** and delete them. The code from the designer follows next.

```
namespace RpgEditor
{
    partial class FormShieldDetail
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
```

```
private void InitializeComponent()
{
    this.mtbDefenseModifier = new System.Windows.Forms.MaskedTextBox();
    this.label7 = new System.Windows.Forms.Label();
    this.mtbDefenseValue = new System.Windows.Forms.MaskedTextBox();
    this.label6 = new System.Windows.Forms.Label();
    this.btnCancel = new System.Windows.Forms.Button();
    this.btnOK = new System.Windows.Forms.Button();
    this.btnRemoveAllowed = new System.Windows.Forms.Button();
    this.btnMoveAllowed = new System.Windows.Forms.Button();
    this.label10 = new System.Windows.Forms.Label();
    this.label9 = new System.Windows.Forms.Label();
    this.lbAllowedClasses = new System.Windows.Forms.ListBox();
    this.lbClasses = new System.Windows.Forms.ListBox();
    this.nudWeight = new System.Windows.Forms.NumericUpDown();
    this.label4 = new System.Windows.Forms.Label();
    this.mtbPrice = new System.Windows.Forms.MaskedTextBox();
    this.label3 = new System.Windows.Forms.Label();
    this.tbType = new System.Windows.Forms.TextBox();
    this.label2 = new System.Windows.Forms.Label();
    this.tbName = new System.Windows.Forms.TextBox();
    this.label1 = new System.Windows.Forms.Label();
    ((System.ComponentModel.ISupportInitialize)(this.nudWeight)).BeginInit();
    this.SuspendLayout();
    //
    // mtbDefenseModifier
    //
    this.mtbDefenseModifier.Location = new System.Drawing.Point(105, 137);
    this.mtbDefenseModifier.Mask = "000";
    this.mtbDefenseModifier.Name = "mtbDefenseModifier";
    this.mtbDefenseModifier.Size = new System.Drawing.Size(100, 20);
    this.mtbDefenseModifier.TabIndex = 57;
    //
    // label7
    //
    this.label7.AutoSize = true;
    this.label7.Location = new System.Drawing.Point(9, 140);
    this.label7.Name = "label7";
    this.label7.Size = new System.Drawing.Size(90, 13);
    this.label7.TabIndex = 55;
    this.label7.Text = "Defense Modifier:";
    //
    // mtbDefenseValue
    //
    this.mtbDefenseValue.Location = new System.Drawing.Point(105, 111);
    this.mtbDefenseValue.Mask = "000";
    this.mtbDefenseValue.Name = "mtbDefenseValue";
    this.mtbDefenseValue.Size = new System.Drawing.Size(100, 20);
    this.mtbDefenseValue.TabIndex = 56;
    //
    // label6
    //
    this.label6.AutoSize = true;
    this.label6.Location = new System.Drawing.Point(18, 114);
    this.label6.Name = "label6";
    this.label6.Size = new System.Drawing.Size(80, 13);
    this.label6.TabIndex = 54;
    this.label6.Text = "Defense Value:";
    //
    // btnCancel
```

```
//
this.btnCancel.Location = new System.Drawing.Point(394, 210);
this.btnCancel.Name = "btnCancel";
this.btnCancel.Size = new System.Drawing.Size(75, 23);
this.btnCancel.TabIndex = 50;
this.btnCancel.Text = "Cancel";
this.btnCancel.UseVisualStyleBackColor = true;
//
// btnOK
//
this.btnOK.Location = new System.Drawing.Point(313, 210);
this.btnOK.Name = "btnOK";
this.btnOK.Size = new System.Drawing.Size(75, 23);
this.btnOK.TabIndex = 51;
this.btnOK.Text = "OK";
this.btnOK.UseVisualStyleBackColor = true;
//
// btnRemoveAllowed
//
this.btnRemoveAllowed.Location = new System.Drawing.Point(354, 120);
this.btnRemoveAllowed.Name = "btnRemoveAllowed";
this.btnRemoveAllowed.Size = new System.Drawing.Size(75, 23);
this.btnRemoveAllowed.TabIndex = 48;
this.btnRemoveAllowed.Text = "<<";
this.btnRemoveAllowed.UseVisualStyleBackColor = true;
//
// btnMoveAllowed
//
this.btnMoveAllowed.Location = new System.Drawing.Point(354, 77);
this.btnMoveAllowed.Name = "btnMoveAllowed";
this.btnMoveAllowed.Size = new System.Drawing.Size(75, 23);
this.btnMoveAllowed.TabIndex = 49;
this.btnMoveAllowed.Text = ">>";
this.btnMoveAllowed.UseVisualStyleBackColor = true;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(432, 9);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(86, 13);
this.label10.TabIndex = 46;
this.label10.Text = "Allowed Classes:";
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(228, 9);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(95, 13);
this.label9.TabIndex = 47;
this.label9.Text = "Character Classes:";
//
// lbAllowedClasses
//
this.lbAllowedClasses.FormattingEnabled = true;
this.lbAllowedClasses.Location = new System.Drawing.Point(435, 33);
this.lbAllowedClasses.Name = "lbAllowedClasses";
this.lbAllowedClasses.Size = new System.Drawing.Size(120, 173);
this.lbAllowedClasses.TabIndex = 44;
```

```
//
// lbClasses
//
this.lbClasses.FormattingEnabled = true;
this.lbClasses.Location = new System.Drawing.Point(228, 33);
this.lbClasses.Name = "lbClasses";
this.lbClasses.Size = new System.Drawing.Size(120, 173);
this.lbClasses.TabIndex = 45;
//
// nudWeight
//
this.nudWeight.DecimalPlaces = 2;
this.nudWeight.Location = new System.Drawing.Point(105, 85);
this.nudWeight.Name = "nudWeight";
this.nudWeight.Size = new System.Drawing.Size(100, 20);
this.nudWeight.TabIndex = 43;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(55, 87);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(44, 13);
this.label4.TabIndex = 42;
this.label4.Text = "Weight:";
//
// mtbPrice
//
this.mtbPrice.Location = new System.Drawing.Point(105, 59);
this.mtbPrice.Mask = "000000";
this.mtbPrice.Name = "mtbPrice";
this.mtbPrice.Size = new System.Drawing.Size(100, 20);
this.mtbPrice.TabIndex = 41;
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(65, 62);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(34, 13);
this.label3.TabIndex = 40;
this.label3.Text = "Price:";
//
// tbType
//
this.tbType.Location = new System.Drawing.Point(105, 33);
this.tbType.Name = "tbType";
this.tbType.Size = new System.Drawing.Size(100, 20);
this.tbType.TabIndex = 39;
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(65, 36);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(34, 13);
this.label2.TabIndex = 38;
this.label2.Text = "Type:";
//
// tbName
```

```
        //
        this.tbName.Location = new System.Drawing.Point(105, 7);
        this.tbName.Name = "tbName";
        this.tbName.Size = new System.Drawing.Size(100, 20);
        this.tbName.TabIndex = 37;
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(61, 10);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(38, 13);
        this.label1.TabIndex = 36;
        this.label1.Text = "Name:";
        //
        // FormShieldDetail
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(592, 244);
        this.ControlBox = false;
        this.Controls.Add(this.mtbDefenseModifier);
        this.Controls.Add(this.label7);
        this.Controls.Add(this.mtbDefenseValue);
        this.Controls.Add(this.label6);
        this.Controls.Add(this.btnCancel);
        this.Controls.Add(this.btnOK);
        this.Controls.Add(this.btnRemoveAllowed);
        this.Controls.Add(this.btnMoveAllowed);
        this.Controls.Add(this.label10);
        this.Controls.Add(this.label9);
        this.Controls.Add(this.lbAllowedClasses);
        this.Controls.Add(this.lbClasses);
        this.Controls.Add(this.nudWeight);
        this.Controls.Add(this.label4);
        this.Controls.Add(this.mtbPrice);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.tbType);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.tbName);
        this.Controls.Add(this.label1);
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;
        this.Name = "FormShieldDetail";
        this.Text = "FormShieldDetail";
        ((System.ComponentModel.ISupportInitialize)(this.nudWeight)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.MaskedTextBox mtbDefenseModifier;
    private System.Windows.Forms.Label label7;
    private System.Windows.Forms.MaskedTextBox mtbDefenseValue;
    private System.Windows.Forms.Label label6;
    private System.Windows.Forms.Button btnCancel;
    private System.Windows.Forms.Button btnOK;
    private System.Windows.Forms.Button btnRemoveAllowed;
    private System.Windows.Forms.Button btnMoveAllowed;
```

```
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.ListBox lbAllowedClasses;
        private System.Windows.Forms.ListBox lbClasses;
        private System.Windows.Forms.NumericUpDown nudWeight;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.MaskedTextBox mtbPrice;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox tbType;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox tbName;
        private System.Windows.Forms.Label label1;
    }
}
```

This tutorial is getting long so I'm just going to add in some basic logic to the forms. I will start with the code for **FormMain**. Right click **FormMain** in the solution explorer and select **View Code**. Change the code to the following. This is the new code for that form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary;
using RpgLibrary.CharacterClasses;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{
    public partial class FormMain : Form
    {
        #region Field Region

        RolePlayingGame rolePlayingGame;
        FormClasses frmClasses;
        FormArmor frmArmor;
        FormShield frmShield;
        FormWeapon frmWeapon;

        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormMain()
        {
            InitializeComponent();

            newGameToolStripMenuItem.Click += new EventHandler(newGameToolStripMenuItem_Click);
            openGameToolStripMenuItem.Click += new
EventHandler(openGameToolStripMenuItem_Click);
            saveGameToolStripMenuItem.Click += new
EventHandler(saveGameToolStripMenuItem_Click);
```

```
    exitToolStripMenuItem.Click += new EventHandler(exitToolStripMenuItem_Click);
    classesToolStripMenuItem.Click += new EventHandler(classesToolStripMenuItem_Click);
    armorToolStripMenuItem.Click += new EventHandler(armorToolStripMenuItem_Click);
    shieldToolStripMenuItem.Click += new EventHandler(shieldToolStripMenuItem_Click);
    weaponToolStripMenuItem.Click += new EventHandler(weaponToolStripMenuItem_Click);
}

#endregion

#region Menu Item Event Handler Region

void newGameToolStripMenuItem_Click(object sender, EventArgs e)
{
    using (FormNewGame frmNewGame = new FormNewGame())
    {
        DialogResult result = frmNewGame.ShowDialog();

        if (result == DialogResult.OK && frmNewGame.RolePlayingGame != null)
        {
            classesToolStripMenuItem.Enabled = true;
            itemsToolStripMenuItem.Enabled = true;
            rolePlayingGame = frmNewGame.RolePlayingGame;
        }
    }
}

void openGameToolStripMenuItem_Click(object sender, EventArgs e)
{
}

void saveGameToolStripMenuItem_Click(object sender, EventArgs e)
{
}

void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}

void classesToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (frmClasses == null)
    {
        frmClasses = new FormClasses();
        frmClasses.MdiParent = this;
    }
    frmClasses.Show();
}

void armorToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (frmArmor == null)
    {
        frmArmor = new FormArmor();
        frmArmor.MdiParent = this;
    }
    frmArmor.Show();
}

void shieldToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            if (frmShield == null)
            {
                frmShield = new FormShield();
                frmShield.MdiParent = this;
            }

            frmShield.Show();
        }
        void weaponToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (frmWeapon == null)
            {
                frmWeapon = new FormWeapon();
                frmWeapon.MdiParent = this;
            }

            frmWeapon.Show();
        }

        #endregion

        #region Method Region
        #endregion
    }
}
```

There are three new fields. One for each of the forms that list all of the different item types. In the constructor I wire event handlers for the **armorToolStripMenuItem**, **shieldToolStripMenuItem**, and **weaponToolStripMenuItem Click** events. The handler for **newGameToolStripMenuItem**'s **Click** event if there was a **RolePlayingGame** object on the form displayed I set the **Enabled** property of **itemsToolStripMenuItem** to true so it is enabled. In the event handler of the **Click** events of the menu items I check to see if the appropriate form is null. If it is null I create it an instance and set the **MdiParent** property to this, the current instance. Outside of the if statement I call the **Show** method of the form rather than **ShowDialog**.

I will add some basic code to each of the detail forms. What I did was add a field for the type of item and a property to expose it. I also handle the click events of the OK and Cancel buttons. Change the code of **FormArmorDetails**, **FormShieldDetails**, and **FormWeaponDetails** to the following.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{
    public partial class FormArmorDetails : Form
    {
        #region Field Region
        Armor armor = null;

        #endregion

        #region Property Region

        public Armor Armor
        {
            get { return armor; }
            set { armor = value; }
        }

        #endregion

        #region Constructor Region

        public FormArmorDetails()
        {
            InitializeComponent();
            this.Load += new EventHandler(FormArmorDetails_Load);
            btnOK.Click += new EventHandler(btnOK_Click);
            btnCancel.Click += new EventHandler(btnCancel_Click);
        }

        #endregion

        #region Event Handler Region

        void FormArmorDetails_Load(object sender, EventArgs e)
        {
        }

        void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.ItemClasses;

namespace RpgEditor
{
    public partial class FormShieldDetail : Form
    {
        #region Field Region

        Shield shield;

        #endregion

        #region Property Region

        public Shield Shield
        {
            get { return shield; }
            set { shield = value; }
        }

        #endregion

        #region Constructor Region

        public FormShieldDetail()
        {
            InitializeComponent();

            this.Load += new EventHandler(FormShieldDetails_Load);

            btnOK.Click += new EventHandler(btnOK_Click);
            btnCancel.Click += new EventHandler(btnCancel_Click);
        }

        #endregion

        #region Event Handler Region

        void FormShieldDetails_Load(object sender, EventArgs e)
        {
        }

        void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        #endregion
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using RpgLibrary.ItemClasses;
```

```
namespace RpgEditor
{
    public partial class FormWeaponDetail : Form
    {
        #region Field Region

        Weapon weapon = null;

        #endregion

        #region Property Region

        public Weapon Weapon
        {
            get { return weapon; }
            set { weapon = value; }
        }

        #endregion

        #region Constructor Region

        public FormWeaponDetail()
        {
            InitializeComponent();

            this.Load += new EventHandler(FormWeaponDetail_Load);

            btnOK.Click += new EventHandler(btnOK_Click);
            btnCancel.Click += new EventHandler(btnCancel_Click);
        }

        #endregion

        #region Event Handler Region

        void FormWeaponDetail_Load(object sender, EventArgs e)
        {
        }

        void btnOK_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        #endregion
    }
}
```

Part 11 Continued

Congratulations for making it this far. I know it is a lot to digest but I decided to merge the three existing tutorials into one. We are in the home stretch though so stick with it and we will get there.

This is the third part of the tutorial on adding a game editor to the project. This tutorial will be more about coding than designing forms. You will want to make the editor the start up project for the duration of this tutorial. Right click the **RpgEditor** project in the solution explorer and select **Set As StartUp Project**.

I want to make a quick change to the forms that hold all of a specific class in the game. To start remove all code from **FormWeapon**, **FormArmor**, **FormShield**, and **FormClasses** that has to do with menu items. For example the code for **FormWeapon** looks like this now.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RpgEditor
{
    public partial class FormWeapon : FormDetails
    {
        #region Field Region
        #endregion

        #region Property Region
        #endregion

        #region Constructor Region

        public FormWeapon()
        {
            InitializeComponent();

            btnAdd.Click += new EventHandler(btnAdd_Click);
            btnEdit.Click += new EventHandler(btnEdit_Click);
            btnDelete.Click += new EventHandler(btnDelete_Click);
        }

        #endregion

        #region Button Event Handler Region

        void btnAdd_Click(object sender, EventArgs e)
        {
        }

        void btnEdit_Click(object sender, EventArgs e)
        {
        }

        void btnDelete_Click(object sender, EventArgs e)
        {
        }

        #endregion
    }
}
```

Go to the designer view of **FormDetails** by right clicking it and selecting **View Designer**. Right click on the **Menu Item Strip** and select **Delete**. Now click on the **List Box** and move its top border up a little.

You don't want the forms that inherit from **FormDetails** to close when the user tries to close the form. You only want to close them when the parent MDI form is closed. Luckily enough there is an event that you can wire and stop that from happening. Right click **FormDetails** in the solution explorer and select **View Code**. Change the **Constructor** region to the following and add this new region.

```
#region Constructor Region

public FormDetails()
{
    InitializeComponent();

    if (FormDetails.ItemManager == null)
        ItemManager = new ItemManager();

    if (FormDetails.EntityDataManager == null)
        EntityDataManager = new EntityDataManager();

    this.FormClosing += new FormClosingEventHandler(FormDetails_FormClosing);
}

#endregion

#region Event Handler Region

void FormDetails_FormClosing(object sender, FormClosingEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing)
    {
        e.Cancel = true;
        this.Hide();
    }

    if (e.CloseReason == CloseReason.MdiFormClosing)
    {
        e.Cancel = false;
        this.Close();
    }
}

#endregion
```

The **FormClosing** event is an event that can be canceled. The **FormClosingEventArgs** argument has a property, **CloseReason**, that holds the reason why the form is being closed. If the value is **UserClosing**, the user tried to close the form, I set the **Cancel** property to true so the event will be canceled and I call the **Hide** method to hide the form. If the **CloseReason** is **MdiFormClosing** then the main form is closing and you want to close the form. I set the **Cancel** property to false and call the **Close** method of the form. Right click the **FormDetails** form again in the solution explorer and this time select **View Designer**. Set the **MinimizeBox** property to false so the user can't minimize the form.

When a menu item is selected you want to bring that form to the front. Lucky enough there is a method you can call to do just that, **BringToFront**. Right click **FormMain** in the solution explorer and select **View Code** to open the code for that form. Change the event handlers for the **Menu Item Event Handler** region to the following.

```
#region Menu Item Event Handler Region
void newGameToolStripMenuItem_Click(object sender, EventArgs e)
{
    using (FormNewGame frmNewGame = new FormNewGame())
    {
        DialogResult result = frmNewGame.ShowDialog();

        if (result == DialogResult.OK && frmNewGame.RolePlayingGame != null)
        {
            classesToolStripMenuItem.Enabled = true;
            itemsToolStripMenuItem.Enabled = true;
            rolePlayingGame = frmNewGame.RolePlayingGame;
        }
    }
}

void openGameToolStripMenuItem_Click(object sender, EventArgs e)
{
}

void saveGameToolStripMenuItem_Click(object sender, EventArgs e)
{
}

void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}

void classesToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (frmClasses == null)
    {
        frmClasses = new FormClasses();
        frmClasses.MdiParent = this;
    }

    frmClasses.Show();
    frmClasses.BringToFront();
}

void armorToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (frmArmor == null)
    {
        frmArmor = new FormArmor();
        frmArmor.MdiParent = this;
    }

    frmArmor.Show();
    frmArmor.BringToFront();
}

void shieldToolStripMenuItem_Click(object sender, EventArgs e)
{
```

```
            if (frmShield == null)
            {
                frmShield = new FormShield();
                frmShield.MdiParent = this;
            }

            frmShield.Show();
            frmShield.BringToFront();
        }

        void weaponToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (frmWeapon == null)
            {
                frmWeapon = new FormWeapon();
                frmWeapon.MdiParent = this;
            }

            frmWeapon.Show();
            frmWeapon.BringToFront();
        }

        #endregion
```

The new code just calls the **BringToFront** method after the **Show** method so that form will be on top of all other forms. What I'm going to do next is handle creating a new game from the main form. I want to add a couple static fields and get only properties to expose their values. Also add a using statement for the **System.IO** name space.

```
using System.IO;

static string gamePath = "";
static string classPath = "";
static string itemPath = "";

#region Property Region

public static string GamePath
{
    get { return gamePath; }
}

public static string ClassPath
{
    get { return classPath; }
}

public static string ItemPath
{
    get { return itemPath; }
}

#endregion
```

There are a few things I need to do with the **RpgLibrary**. In order to deserialize the objects you need a constructor that takes no parameters. Instead of adding constructors that take no parameters to the item classes I instead created data classes with just public fields that match. For the **RolePlayingGame** class I did just add in a constructor that took no parameters. Add the following constructor to the constructor region of the **RolePlayingGame** class.

```
private RolePlayingGame()
{}
```

Now, right click the **ItemClasses** folder in the **RpgLibrary** project, select **Add** and then **Class**. Name the class **ArmorData**. Repeat the process twice and name the classes **ShieldData** and **WeaponData**. The code for those three classes follows next.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace RpgLibrary.ItemClasses
{
    public class ArmorData
    {
        public string Name;
        public string Type;
        public int Price;
        public float Weight;
        public bool Equipped;
        public ArmorLocation ArmorLocation;
        public int DefenseValue;
        public int DefenseModifier;
        public string[] AllowableClasses;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace RpgLibrary.ItemClasses
{
    public class ShieldData
    {
        public string Name;
        public string Type;
        public int Price;
        public float Weight;
        public bool Equipped;
        public int DefenseValue;
        public int DefenseModifier;
        public string[] AllowableClasses;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace RpgLibrary.ItemClasses
{
    public class WeaponData
    {
        public string Name;
        public string Type;
        public int Price;
        public float Weight;
        public bool Equipped;
        public Hands NumberHands;
        public int AttackValue;
        public int AttackModifier;
        public int DamageValue;
        public int DamageModifier;
        public string[] AllowableClasses;
    }
}
```

I'm going to add in a class to manage all of these item data classes. Right click the **ItemClasses** folder in the **RpgLibrary** project, select **Add** and then **Class**. Name this new class **ItemDataManager**. This is the code for that class.

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

namespace RpgLibrary.ItemClasses
{
    public class ItemDataManager
    {
        #region Field Region

        Dictionary<string, ArmorData> armorData = new Dictionary<string, ArmorData>();
        Dictionary<string, ShieldData> shieldData = new Dictionary<string, ShieldData>();
        Dictionary<string, WeaponData> weaponData = new Dictionary<string, WeaponData>();

        #endregion

        #region Property Region

        public Dictionary<string, ArmorData> ArmorData
        {
            get { return armorData; }
            set { armorData = value; }
        }

        public Dictionary<string, ShieldData> ShieldData
        {
            get { return shieldData; }
            set { shieldData = value; }
        }

        public Dictionary<string, WeaponData> WeaponData
        {
            get { return weaponData; }
            set { weaponData = value; }
        }

        #endregion

        #region Constructor Region
        #endregion
        #region Method Region
        #endregion
    }
}
```

It works the same way as the **EntityDataManager** class works except there are three dictionaries instead of the one. The properties are also get and set for serialization with the **IntermediateSerializer** so the data will be serialized.

I want to change the **ItemManager** field in **FormDetails** to be **ItemDataManager** instead. I also want to expose the **itemManager** and **entityDataManager** field to other forms. Change the **Field**, **Property**, and **Constructor** regions in **FormDetails** to the following.

```
        #region Field Region

        protected static ItemDataManager itemManager;
        protected static EntityDataManager entityDataManager;

        #endregion

        #region Property Region

        public static ItemDataManager ItemManager
        {
            get { return itemManager; }
            private set { itemManager = value; }
        }

        public static EntityDataManager EntityDataManager
        {
            get { return entityDataManager; }
            private set { entityDataManager = value; }
        }

        #endregion

        #region Constructor Region

        public FormDetails()
        {
            InitializeComponent();

            if (FormDetails.ItemManager == null)
                ItemManager = new ItemDataManager();

            if (FormDetails.EntityDataManager == null)
                EntityDataManager = new EntityDataManager();
            this.FormClosing += new FormClosingEventHandler(FormDetails_FormClosing);
        }

        #endregion
```

I want to add a method to **FormClasses**, **FormArmor**, **FormShield**, and **FormWeapon**. This method will fill the list box of the form with the appropriate data. The code for the method of each form follows next.

FormClasses

```
        #region Method Region

        public void FillListBox()
        {
            lbDetails.Items.Clear();

            foreach (string s in FormDetails.EntityDataManager.EntityData.Keys)
                lbDetails.Items.Add(FormDetails.EntityDataManager.EntityData[s]);
        }

        #endregion
```

FormArmor

```
public void FillListBox()
{
    lbDetails.Items.Clear();

    foreach (string s in FormDetails.ItemManager.ArmorData.Keys)
        lbDetails.Items.Add(FormDetails.ItemManager.ArmorData[s]);
}
```

FormShield

```
public void FillListBox()
{
    lbDetails.Items.Clear();

    foreach (string s in FormDetails.ItemManager.ShieldData.Keys)
        lbDetails.Items.Add(FormDetails.ItemManager.ShieldData[s]);
}
```

FormWeapon

```
public void FillListBox()
{
    lbDetails.Items.Clear();

    foreach (string s in FormDetails.ItemManager.WeaponData.Keys)
        lbDetails.Items.Add(FormDetails.ItemManager.WeaponData[s]);
}
```

In the **Click** event handler of the **New Game** menu item in **FormMain** is where I will handle creating a new game. Change the **newGameMenuToolStripItem_Click** method to the following.

```
void newGameToolStripMenuItem_Click(object sender, EventArgs e)
{
    using (FormNewGame frmNewGame = new FormNewGame())
    {
        DialogResult result = frmNewGame.ShowDialog();

        if (result == DialogResult.OK && frmNewGame.RolePlayingGame != null)
        {
            FolderBrowserDialog folderDialog = new FolderBrowserDialog();

            folderDialog.Description = "Select folder to create game in.";
            folderDialog.SelectedPath = Application.StartupPath;
            DialogResult folderResult = folderDialog.ShowDialog();

            if (folderResult == DialogResult.OK)
            {
                try
                {
                    gamePath = Path.Combine(folderDialog.SelectedPath, "Game");
                    classPath = Path.Combine(gamePath, "Classes");
                    itemPath = Path.Combine(gamePath, "Items");

                    if (Directory.Exists(gamePath))
                        throw new Exception("Selected directory already exists.");

                    Directory.CreateDirectory(gamePath);
                    Directory.CreateDirectory(classPath);
                    Directory.CreateDirectory(itemPath + @"\Armor");
                    Directory.CreateDirectory(itemPath + @"\Shield");
                    Directory.CreateDirectory(itemPath + @"\Weapon");

                    rolePlayingGame = frmNewGame.RolePlayingGame;

                    XnaSerializer.Serialize<RolePlayingGame>(gamePath + @"\Game.xml",
                    rolePlayingGame);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.ToString());
                    return;
                }

                classesToolStripMenuItem.Enabled = true;
                itemsToolStripMenuItem.Enabled = true;
            }
        }
    }
}
```

The new code is in the if statement that checks to see if the result of the dialog was **DialogResult.OK** and that the **RolePlayingGame** property of the form is not null. If those were both true I create a **FolderBrowserDialog** object. I set the **SelectedPath** property to be the path where the editor started from. I also set the **Description** property to let the user know what folder they are browsing for. I capture the result of showing that dialog. If the result was **DialogResult.OK** there is a **try-catch** block where I attempt to make directories to save the game in. I use the **Combine** method of the **Path** class to create the paths to save to. The path for the game, **gamePath**, is the selected path from the dialog and **Game**. The **classPath** is the **gamePath** and **Classes**. The **itemPath** is the **gamePath** and **Items**. I then check to see if **gamePath** exists. If it does I throw an exception saying that a game already exists in that directory. I only want to have one game per directory. I then call the **CreateDirectory** method of the **Directory** class to create the directories. Each sub-item type will be stored in a directory of its own under the **itemPath** directory. The **rolePlayingGame** field is set to the **RolePlayingGame** property of the new game dialog. I then call the **Serialize<T>** method to serialize the **rolePlayingGame** field. As you can see I specify **RolePlayingGame** for **T**. For the file name I use the **gamePath** and **\Game.xml**. If there was an exception I catch it and display it in a message box and exit the method. If the game was created successfully I set the **Enabled** property of the class and item menu items.

There is a bit of a dependency here. Items require that you have the classes that are allowed to use the item. So in order to code the item forms you need to code the forms dealing with character classes first.

Right click **FormClasses** and select **View Code**. I'm going to update the event handler for the click event of **btnAdd**. Change the code for **btnAdd_Click** to the following. Add the **AddEntity** method to the **Method Region**.

```
private void AddEntity(EntityData entityData)
{
    if (FormDetails.EntityDataManager.EntityData.ContainsKey(entityData.EntityName))
    {
        DialogResult result = MessageBox.Show(
            entityData.EntityName + " already exists. Do you want to overwrite it?",
            "Existing Character Class",
            MessageBoxButtons.YesNo);

        if (result == DialogResult.No)
            return;

        FormDetails.EntityDataManager.EntityData[entityData.EntityName] = entityData;

        FillListBox();

        return;
    }

    lbDetails.Items.Add(entityData.ToString());

    FormDetails.EntityDataManager.EntityData.Add(
        entityData.EntityName,
        entityData);
}
```

The event handler creates a new form inside of a using statement so it will be disposed of when the code exits the block of code. I call the **ShowDialog** method of the form to display it. If the **EntityData** property of the form is not null I call the **AddEntity** method passing in the **EntityData** property of the form.

The **AddEntity** data method will add the **EntityData** adds the new **EntityData** object to **lbDetails** and the the **EntityDataManager** on **FormDetails**. If there exists an **EntityData** object with the same name as the new one and you try and add it an exception will be thrown. So, there is an if statement that checks to see if the key is in the dictionary. If it is I display a message box stating that there is already an entry and if the user wants to overwrite it. If they select No I exit the method. If they want to overwrite it I assign the new **EntityData** object to that key value. I call the method **FillListBox** to refill the list box with the **EntityData** objects. I then exit the method. If there wasn't an existing **EntityData** object I add it to the list box and add it to the **EntityDataManager**.

I'm going to code saving games next. I believe that you should be able to write out data before trying to read it in. The first step is to add code to the **saveGameMenuToolStripItem_Click** event handler in your **FormMain**. Change the code for that method to the following.

```
void saveGameToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (rolePlayingGame != null)
    {
        try
        {
            XnaSerializer.Serialize<RolePlayingGame>(gamePath + @"\Game.xml", rolePlayingGame);

            FormDetails.WriteEntityData();
            FormDetails.WriteItemData();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString(), "Error saving game.");
        }
    }
}
```

I check to see if **rolePlayingGame** is not null. If it is null then there is nothing to save. If it is not then there is a try-catch block that tries to save the game. I first use the **Serialize<T>** method to write out the **rolePlayingGame** field. I then call static methods that I added to **FormDetails** for writing out the values in the **EntityDataManager** and the **ItemManager**. If an exception was thrown I display a message box with the error.

The **WriteEntityData** and **WriteItemData** methods write out the data for the appropriate type. Add the following methods to the **Method Region** of **FormDetails**.

```
public static void WriteEntityData()
{
    foreach (string s in EntityDataManager.EntityData.Keys)
    {
        XnaSerializer.Serialize<EntityData>(
            FormMain.ClassPath + @"\" + s + ".xml",
            EntityDataManager.EntityData[s]);
    }
}

public static void WriteItemData()
{
    foreach (string s in ItemManager.ArmorData.Keys)
    {
        XnaSerializer.Serialize<ArmorData>(
            FormMain.ItemPath + @"\Armor\" + s + ".xml",
            ItemManager.ArmorData[s]);
    }

    foreach (string s in ItemManager.ShieldData.Keys)
    {
        XnaSerializer.Serialize<ShieldData>(
            FormMain.ItemPath + @"\Shield\" + s + ".xml",
            ItemManager.ShieldData[s]);
    }

    foreach (string s in ItemManager.WeaponData.Keys)
    {
        XnaSerializer.Serialize<WeaponData>(
            FormMain.ItemPath + @"\Weapon\" + s + ".xml",
            ItemManager.WeaponData[s]);
    }
}
```

The first method, **WriteEntityData**, loops through all of the keys in the **EntityData** dictionary in the **EntityDataManager** class. Inside the loop I call the **Serialize<T>** method of the **XnaSerializer** class. For the file name I use the static **ClassPath** property of **FormMain**, add a \ to place it in that directory, add the name of the **EntityData**, and an **xml** extension. The **WriteItemData** method works basically the same way. The difference is where they are written. Armor is written to the **Armor** sub-directory of the **Items** folder, shields to **Shield**, and weapons to **Weapon**. As more classes are added to the game you just add in methods to write them out.

Opening a game will require a little more work than writing it out. The way I decided to handle reading in a game is to display a **FolderBrowserDialog** to allow the user to browse to the folder that holds their game. From there I try and open the game. Change the **openGameToolMenuStripItem_Click** method to the following and add the methods **OpenGame** and **PrepareForms**.

```
void openGameToolStripMenuItem_Click(object sender, EventArgs e)
{
    FolderBrowserDialog folderDialog = new FolderBrowserDialog();

    folderDialog.Description = "Select Game folder";
    folderDialog.SelectedPath = Application.StartupPath;

    bool tryAgain = false;

    do
    {
        DialogResult folderResult = folderDialog.ShowDialog();
        DialogResult msgBoxResult;

        if (folderResult == DialogResult.OK)
        {
            if (File.Exists(folderDialog.SelectedPath + @"\Game\Game.xml"))
            {
                try
                {
                    OpenGame(folderDialog.SelectedPath);
                    tryAgain = false;
                }
                catch (Exception ex)
                {
                    msgBoxResult = MessageBox.Show(
                        ex.ToString(),
                        "Error opening game.",
                        MessageBoxButtons.RetryCancel);

                    if (msgBoxResult == DialogResult.Cancel)
                        tryAgain = false;
                    else if (msgBoxResult == DialogResult.Retry)
                        tryAgain = true;
                }
            }
            else
            {
                msgBoxResult = MessageBox.Show(
                    "Game not found, try again?",
                    "Game does not exist",
                    MessageBoxButtons.RetryCancel);

                if (msgBoxResult == DialogResult.Cancel)
                    tryAgain = false;
                else if (msgBoxResult == DialogResult.Retry)
                    tryAgain = true;
            }
        }
    } while (tryAgain);
}
private void OpenGame(string path)
{
    gamePath = Path.Combine(path, "Game");
    classPath = Path.Combine(gamePath, "Classes");
    itemPath = Path.Combine(gamePath, "Items");

    rolePlayingGame = XnaSerializer.Deserialize<RolePlayingGame>(
        gamePath + @"\Game.xml");

    FormDetails.ReadEntityData();
    FormDetails.ReadItemData();

    PrepareForms();
}
private void PrepareForms()
{
    if (frmClasses == null)
    {
        frmClasses = new FormClasses();
        frmClasses.MdiParent = this;
    }

    frmClasses.FillListBox();

    if (frmArmor == null)
    {
        frmArmor = new FormArmor();
        frmArmor.MdiParent = this;
    }

    frmArmor.FillListBox();

    if (frmShield == null)
    {
        frmShield = new FormShield();
        frmShield.MdiParent = this;
    }

    frmShield.FillListBox();

    if (frmWeapon == null)
    {
        frmWeapon = new FormWeapon();
        frmWeapon.MdiParent = this;
    }

    frmWeapon.FillListBox();
    classesToolStripMenuItem.Enabled = true;
    itemsToolStripMenuItem.Enabled = true;
}
```

The **openGameToolStripMenuItem** method first creates a **FolderBrowserDialog** object and sets the **Description** property to **Select Game folder**. I also set the **SelectedPath** to the folder the application started in using the **StartUpPath** property of the **Application** class. There is a bool variable that will determine if the user would like to try again if there is an error opening the game. It is set to false initially.

I did it in a do-while loop instead of a while loop as you know the loop needs to go through at least once. There are two **DialogResult** variables inside of the loop. **folderResult** holds the result of the **ShowDialog** method of the **FolderBrowserDialog** object. **msgResult** holds the result of any message boxes displayed. There is an if statement to check if **folderResult** is **DialogResult.OK**. Inside that if statement there is an if statement that checks to see if **Game.xml** exists in the **Game** folder of the selected folder. Inside that if statement there is a try-catch block where I try to actually open the game.

In the try part I call the **OpenGame** method and set **tryAgain** to false because opening the game worked. In the catch part I capture the result of a message box. I set the buttons for the message box so they are **Retry** and **Cancel**. If the result is **Cancel** I set **tryAgain** to false as the user doesn't want to try again. If it is **Retry** I set **tryAgain** to true.

In the else part of the if statement that checked if a game exits I capture the result of a message box similar to the previous one. I then do the same action. If **Cancel** is select **tryAgain** is set to false and true if **Retry** was selected.

The **OpenGame** method is where I actually try and read in the data for the game. I first create the paths to read data to like I did when I created a new game. I then deserialize the **Game.xml** file into the **rolePlayingGame** field. I call the **ReadEntityData** and **ReadItemData** methods of **FormDetails** to read in the entity data and item data. I then call **PrepareForms** to prepare the forms.

In the **PrepareForms** method I check if each of the forms is null. If it is null I create a new instance and set the **MdiParent** property to this, the current form. I then call the **FillListBox** method on the form to fill the list box with the data for that form.

You need to add two static methods to **FormDetails**, **ReadEntityData** and **ReadItemData**, to read in the data. You also need to add a using statement for the **System.IO** name space.

```
public static void ReadEntityData()
{
    entityDataManager = new EntityDataManager();

    string[] fileNames = Directory.GetFiles(FormMain.ClassPath, "*.xml");

    foreach (string s in fileNames)
    {
        EntityData entityData = XnaSerializer.Deserialize<EntityData>(s);
        entityDataManager.EntityData.Add(entityData.EntityName, entityData);
    }
}
public static void ReadItemData()
{
    itemManager = new ItemDataManager();

    string[] fileNames = Directory.GetFiles(
        Path.Combine(FormMain.ItemPath, "Armor"),
        "*.xml");

    foreach (string s in fileNames)
    {
        ArmorData armorData = XnaSerializer.Deserialize<ArmorData>(s);
        itemManager.ArmorData.Add(armorData.Name, armorData);
    }

    fileNames = Directory.GetFiles(
        Path.Combine(FormMain.ItemPath, "Shield"),
        "*.xml");

    foreach (string s in fileNames)
    {
        ShieldData shieldData = XnaSerializer.Deserialize<ShieldData>(s);
        itemManager.ShieldData.Add(shieldData.Name, shieldData);
    }

    fileNames = Directory.GetFiles(
        Path.Combine(FormMain.ItemPath, "Weapon"),
        "*.xml");

    foreach (string s in fileNames)
    {
        WeaponData weaponData = XnaSerializer.Deserialize<WeaponData>(s);
        itemManager.WeaponData.Add(weaponData.Name, weaponData);
    }
}
```

To read in data you need a file name of the file you want to read in. I have each type of data in a folder of its own. I use the **GetFiles** method of the **Directory** class to get all of the files in that directory that returns an array of strings with the file names. In a foreach loop I iterate over all of the items in the array of file names. Inside of the foreach loop there is a variable of the type of data I want to read in. I use the **Deserialize<T>** method of the **XnaSerializer** class to deserialize the file into the variable. I then add the object to the manager class.

I think that this is more than enough for this tutorial. The editors are coming along nicely but I think you deserve a break from them. In the next tutorial I will move back to the game instead of the editors. I encourage you to visit my blog, https://cynthiamcmahon.ca/blog/, for the latest news on my tutorials and other goodness.

Good luck in your game programming adventures!
Cynthia