XNA 4.0 RPG Tutorials

Part 11b

Game Editors

I'm writing these tutorials for the new XNA 4.0 framework. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the XNA 4.0 RPG tutorials page of my web site. I will be making my version of the project available for download at the end of each tutorial. It will be included on the page that links to the tutorials.

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 { One, Two }

    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 characterstring)
        {
            return allowableClasses.Contains(characterstring);
        }

        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 s in allowableClasses)
                armorString += ", " + s;

            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)
        {
```

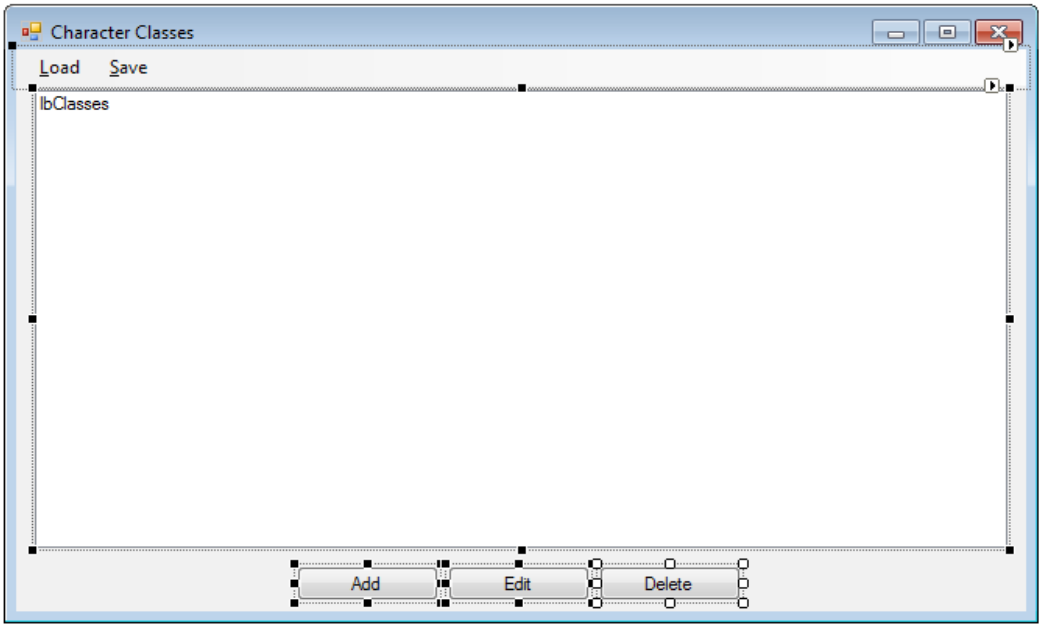
```
            DefenseValue = defenseValue;
            DefenseModifier = defenseModifier;
        }

        #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 s in allowableClasses)
                shieldString += ", " + s;

            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 s in allowableClasses)
                weaponString += ", " + s;

            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**. The form should look like this.



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**. 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
    }
}
```

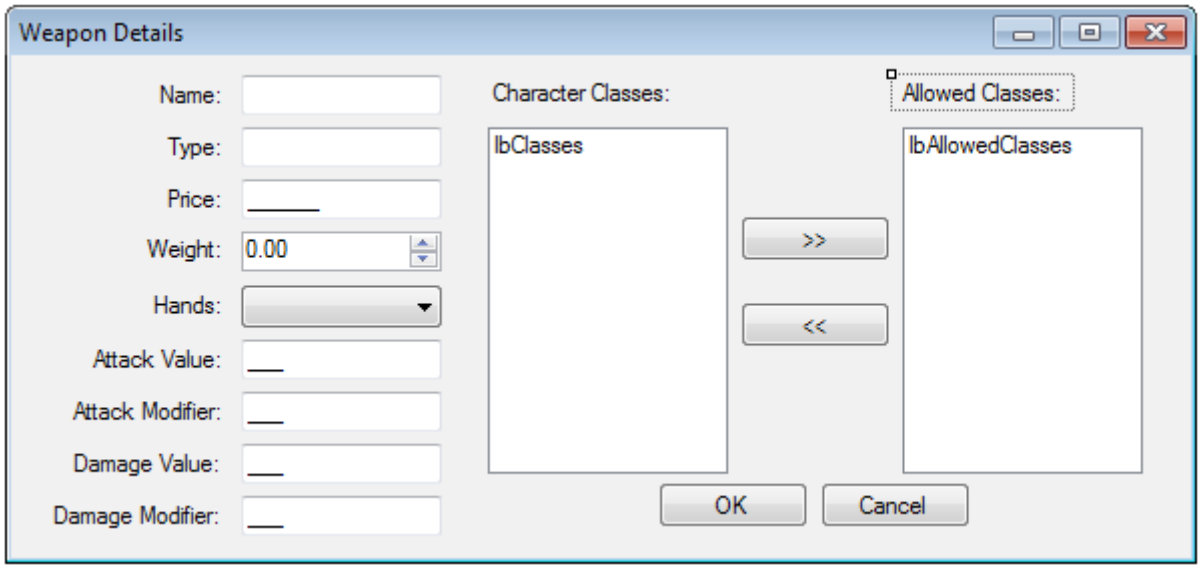
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RpgEditor
{
    public partial class 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
    }
}
```

```
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
    }
}
```

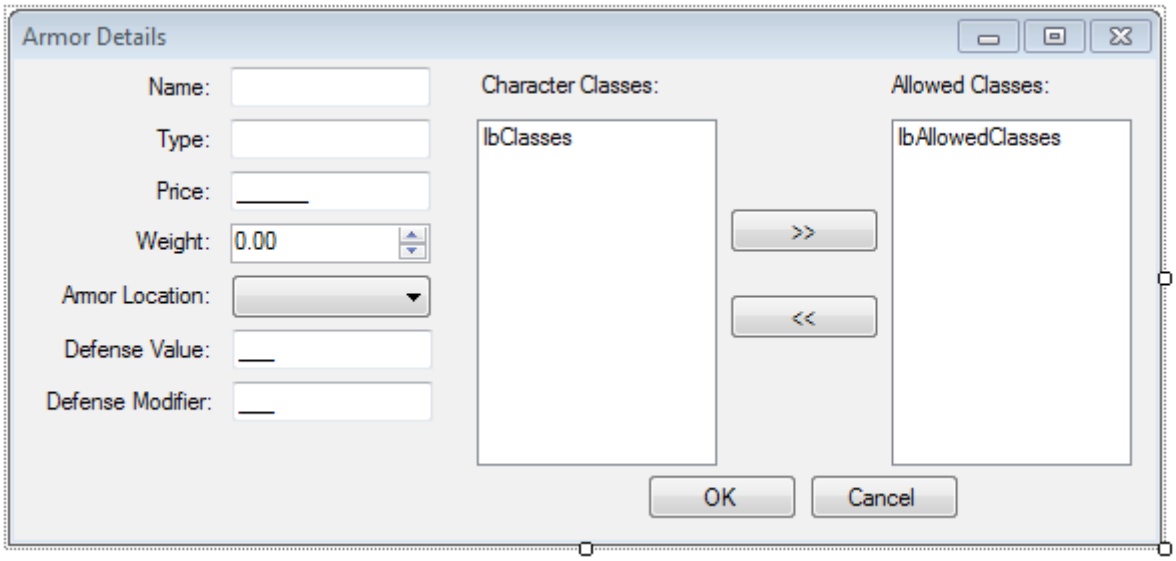
I'm going to add some details forms for specific item types. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormWeaponDetails**. My finished form looked like below. Set its **Text** property to **Weapon Details** and its **FormBorderStyle** to **FixedDialog**.



There are a lot of controls on the form. 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 **Price:** 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 its **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**.

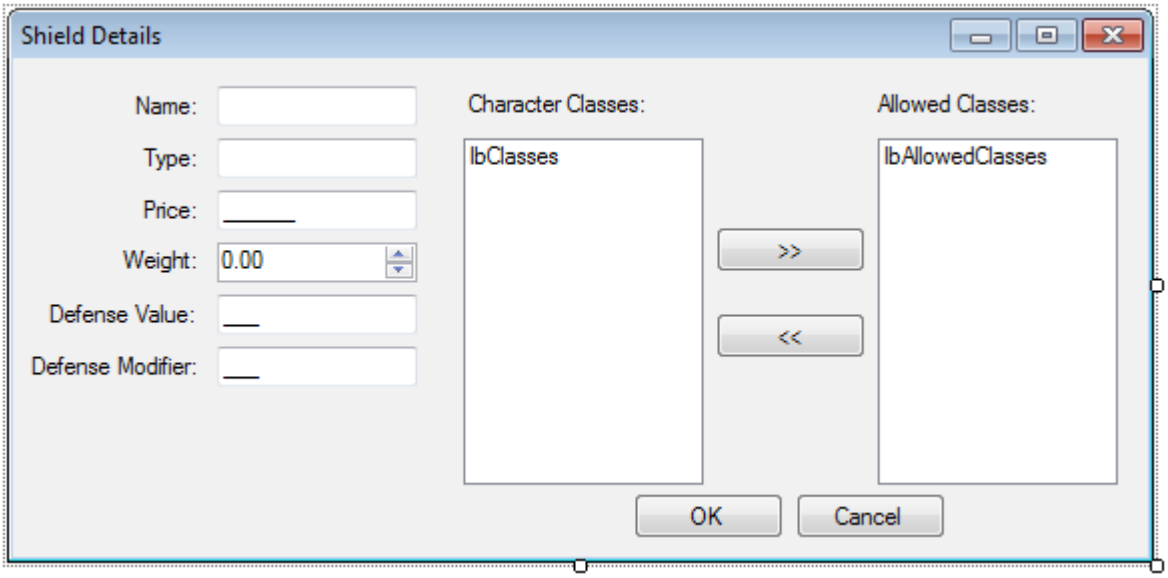
The next form to add is a form for the various types of armor in the game. Right click the **RpgEditor** project, select **Add** and then **Windows Form**. Name this new form **FormArmorDetails**. Set the **Text** property to **Armor Details** and the **FormBorderStyle** to **FixedDialog**. My form looked 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**.

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.

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 FormShieldDetails : Form
    {
        #region Field Region

        Shield shield;

        #endregion

        #region Property Region
```

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

        #endregion

        #region Constructor Region

        public FormShieldDetails()
        {
            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
    }
}
```

I'm going to end this tutorial here as it is rather on the long side. I want to try and keep them to a reasonable length so that you don't have too much to digest at once. I encourage you to visit the news page of my site, XNA Game Programming Adventures, for the latest news on my tutorials.

Good luck in your game programming adventures!

Jamie McMahon