Craft Project Documentation

Andrada Iorgulescu and March 27, 2022

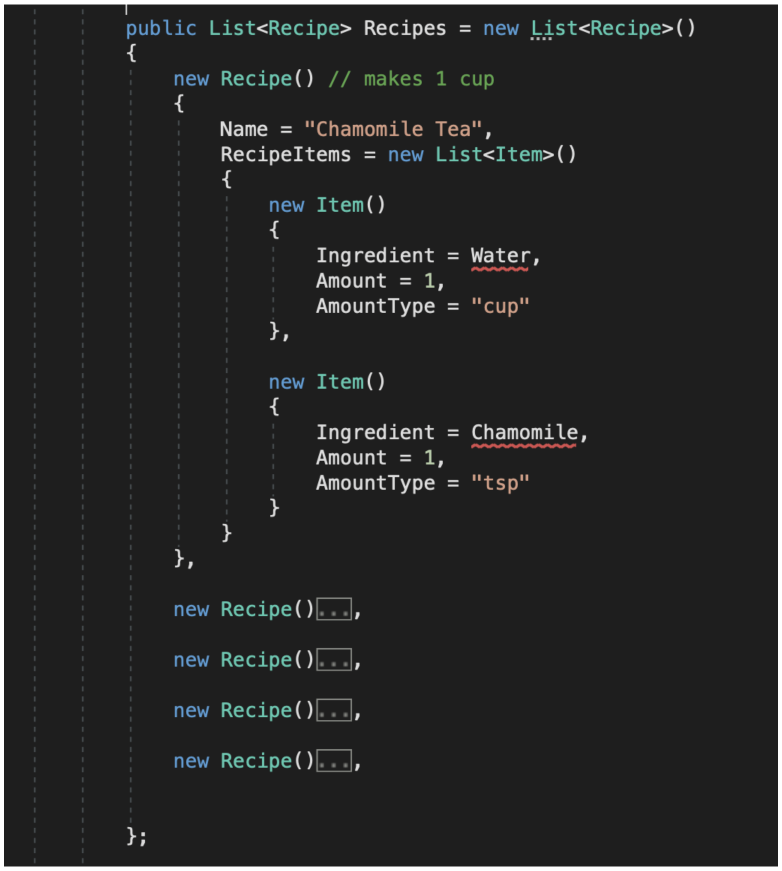
Contents

Object-Oriented Programming Concepts

Encapsulation

Definition: Encapsulation is the practice of grouping properties, methods, or other members together as a single unit. It is very effective at hiding the data and behaviors of an object that are not necessary to the user.

Brief code excerpt(s) from your project:



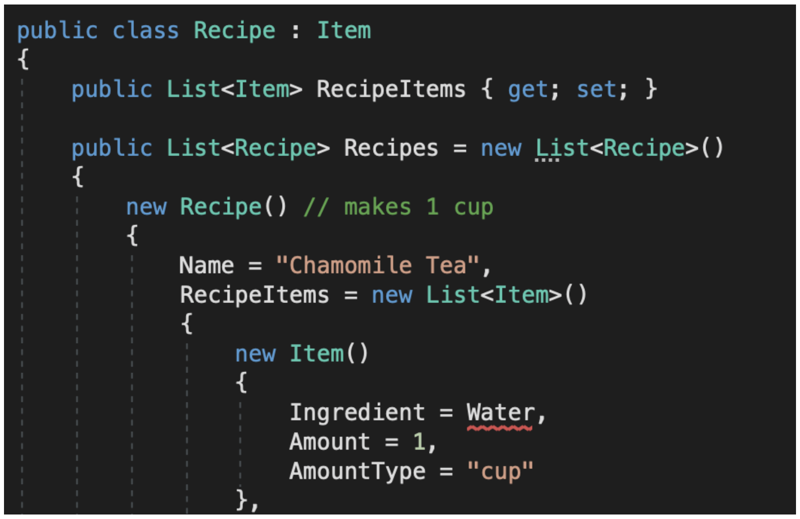
```
public List<Recipe> Recipes = new List<Recipe>()
{
    new Recipe() // makes 1 cup
    {
        Name = "Chamomile Tea",
        RecipeItems = new List<Item>()
        {
            new Item()
            {
                Ingredient = Water,
                Amount = 1,
                AmountType = "cup"
            },

            new Item()
            {
                Ingredient = Chamomile,
                Amount = 1,
                AmountType = "tsp"
            }
        }
    },

    new Recipe()...,

    new Recipe()...,

    new Recipe()...,

    new Recipe()...,


};
```

Explain usage in your project: I created a Recipes List in my Recipe class to list the requirements for each Crafted Item. Nested within the list are five “new Recipe()” members. Nested within each “new Recipe()” is the name of the Crafted Item, the required ingredients, required amount, and required amount type.

Inheritance ("is a")

Definition: Using a new class to reuse, extend, and modify the behavior defined in another(s). The class that is being inherited from is called the *base class*. The class that inherits is called a *derived class*.

Brief code excerpt(s) from your project:



```
public class Recipe : Item
{
    public List<Item> RecipeItems { get; set; }

    public List<Recipe> Recipes = new List<Recipe>()
    {
        new Recipe() // makes 1 cup
        {
            Name = "Chamomile Tea",
            RecipeItems = new List<Item>()
            {
                new Item()
                {
                    Ingredient = Water,
                    Amount = 1,
                    AmountType = "cup"
                },
```

Explain usage in your project: My Recipe class inherits from my Item class. I have made a list for Recipes that states which Inventory items (RecipeItems), and which amounts of those items, are required to create each Crafted item. Inheriting from the Item class allows me to give the items in the Recipes list a name, ingredient, amount, and amount type.

Polymorphism

Definition: The process of redefining methods in derived classes. It is most often used in late binding, where an object's response to the other members in its method is determined based on the object's type at run time.

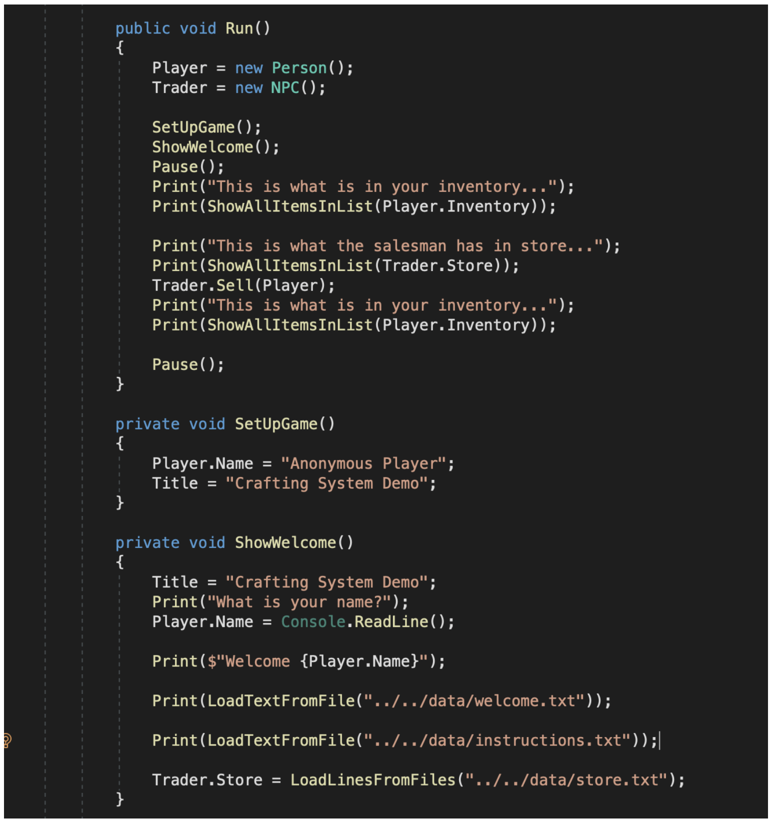
Brief code excerpt(s) from your project:

Explain usage in your project:

Separation of Concern

Definition: The idea that pieces of software should be separated based on the kind of tasks it is created to do.

Brief code excerpt(s) from your project:



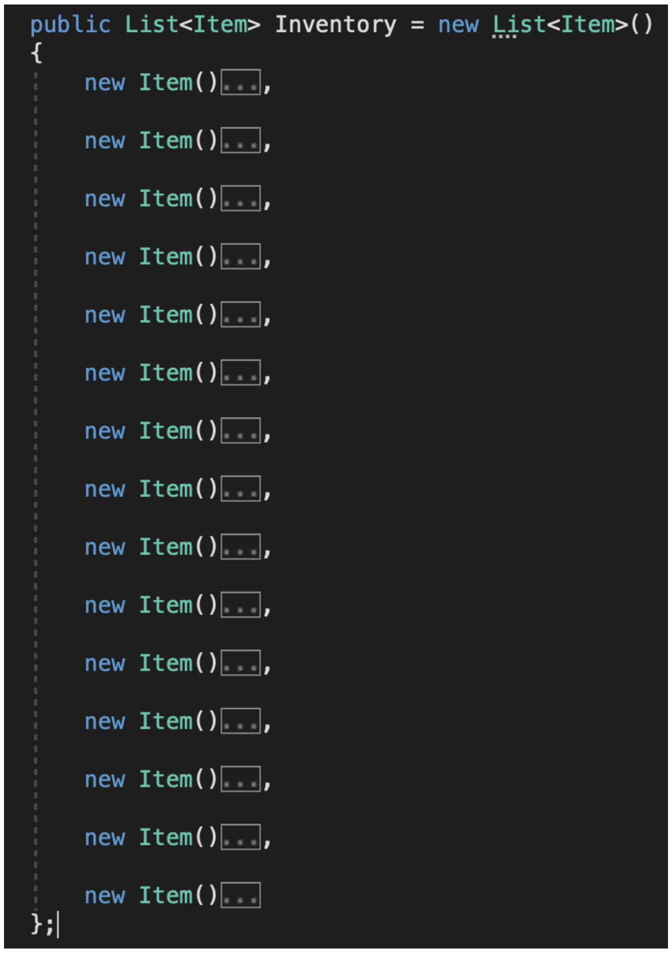
```
public void Run()
{
    Player = new Person();
    Trader = new NPC();

    SetUpGame();
    ShowWelcome();
    Pause();
    Print("This is what is in your inventory...");
    Print(ShowAllItemsInList(Player.Inventory));

    Print("This is what the salesman has in store...");
    Print(ShowAllItemsInList(Trader.Store));
    Trader.Sell(Player);
    Print("This is what is in your inventory...");
    Print(ShowAllItemsInList(Player.Inventory));

    Pause();
}

private void SetUpGame()
{
    Player.Name = "Anonymous Player";
    Title = "Crafting System Demo";
}

private void ShowWelcome()
{
    Title = "Crafting System Demo";
    Print("What is your name?");
    Player.Name = Console.ReadLine();

    Print($"Welcome {Player.Name}");

    Print(LoadTextFromFile("../../data/welcome.txt"));

    Print(LoadTextFromFile("../../data/instructions.txt"));

    Trader.Store = LoadLinesFromFiles("../../data/store.txt");
}
```

Explain usage in your project: In the game class, I have separated the code into 3 separate methods: Run(), SetUpGame(), and ShowWelcome(). SetUpGame() gives the player a placeholder name and a title to the game window. ShowWelcome() asks the player for their name, saves it, and displays it back to them. It also loads the welcome text and instructions text from external files. It also populates the Trader's store inventory from an external file. Run() creates an instance of the player and the trader NPC, and runs the other 2 methods. It also displays the player and trader's inventories.

C# Programming Skills

Collection (e.g., a list, array, dictionary, etc.)

Brief code excerpt(s) from your project:

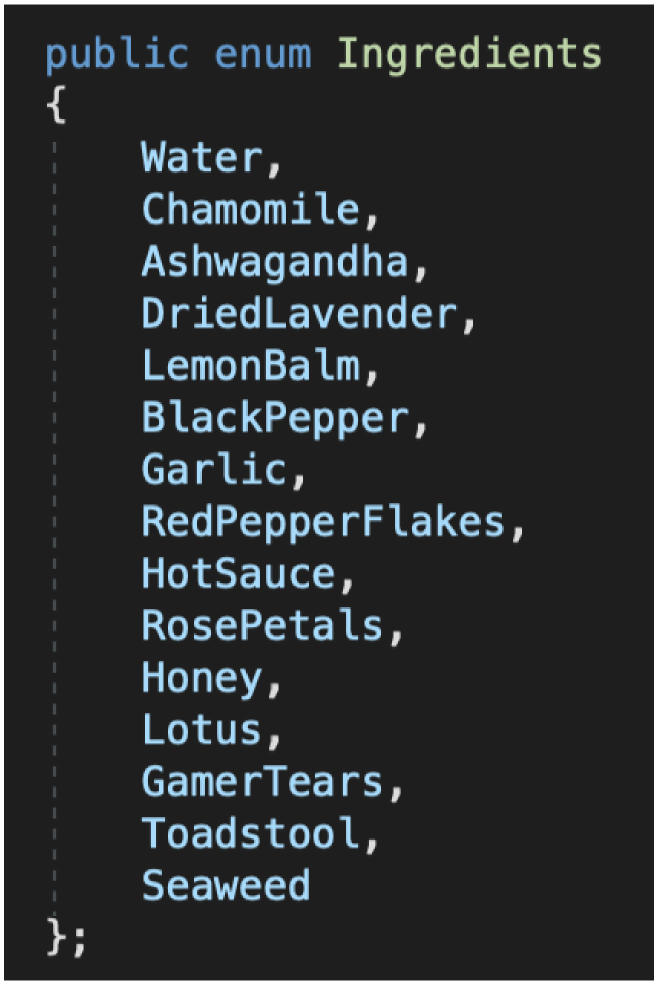


```
public List<Item> Inventory = new List<Item>()
{
    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...,

    new Item()...
};
```

Explain usage in your project: This is an Inventory List I have created in my Person class. It is being used to hold all of the items that the player will begin the game with. These items are the bare minimum needed to make 1 of each kind of potion.

Enum

Brief code excerpt(s) from your project:

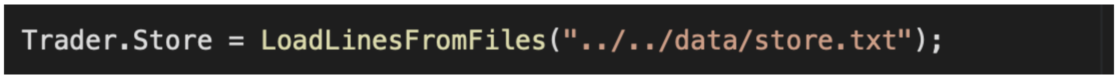


```
public enum Ingredients
{
    Water,
    Chamomile,
    Ashwagandha,
    DriedLavender,
    LemonBalm,
    BlackPepper,
    Garlic,
    RedPepperFlakes,
    HotSauce,
    RosePetals,
    Honey,
    Lotus,
    GamerTears,
    Toadstool,
    Seaweed
};
```

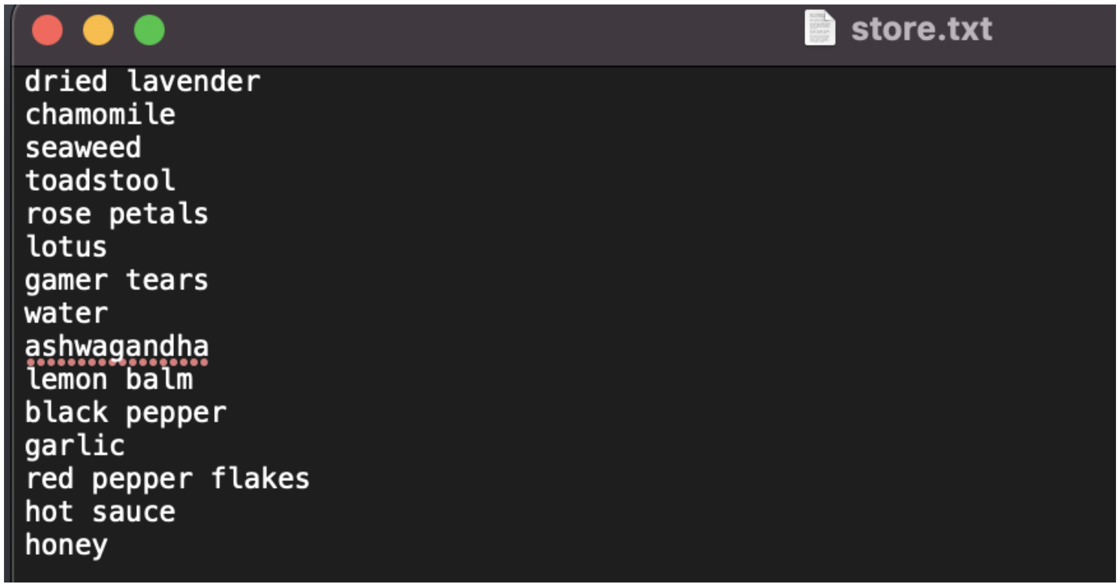
Explain usage in your project: This is the Ingredients enum I created in my Item class. It is used to hold every kind of ingredient that is available in the game. This enum is expanded upon in the Recipe class to explain which ingredients are used in which potion.

External data (read in)

Brief code excerpt(s) from your project:



```
Trader.Store = LoadLinesFromFiles("../../data/store.txt");
```



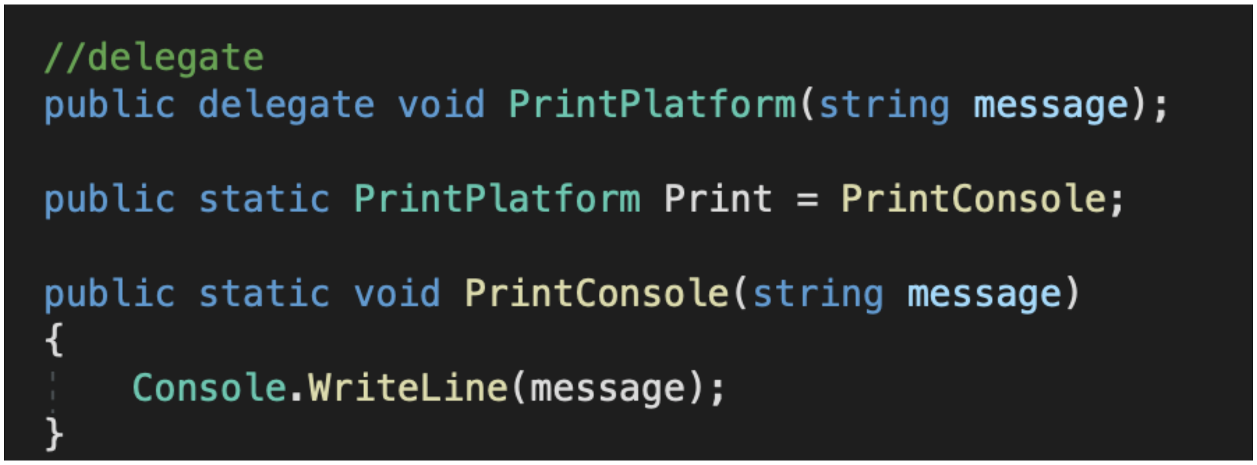
store.txt

```
dried lavender
chamomile
seaweed
toadstool
rose petals
lotus
gamer tears
water
ashwagandha
lemon balm
black pepper
garlic
red pepper flakes
hot sauce
honey
```

Explain usage in your project: The items that are available in the NPC Trader's store are being read in from an external file called "store". This text file is held within the data folder. The code that reads in the items is held within the Game class, because that is where the Trader is instantiated.

Delegate(s)

Brief code excerpt(s) from your project:

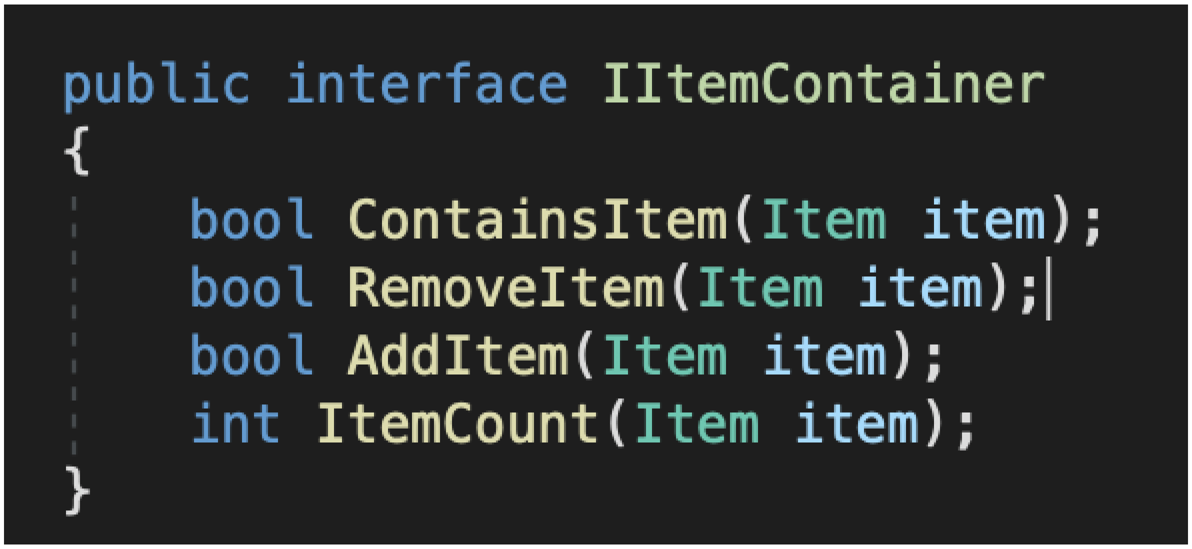


```
//delegate
public delegate void PrintPlatform(string message);

public static PrintPlatform Print = PrintConsole;

public static void PrintConsole(string message)
{
    Console.WriteLine(message);
}
```

Explain usage in your project:

Interface(s)

Brief code excerpt(s) from your project:



```
public interface IItemContainer
{
    bool ContainsItem(Item item);
    bool RemoveItem(Item item);
    bool AddItem(Item item);
    int ItemCount(Item item);
}
```

Explain usage in your project: This interface is used to create methods that are then used in the Person class. These methods were made to be used during the crafting process: to check the player's inventory for the required items in the required amounts, then to remove them and add the crafted item in their place. The RemoveItem method could have also been used to remove purchased items from the NPC's store, and the AddItem method could have been used to add purchased items to the customer's inventory.

One of these: LINQ or XML

Brief code excerpt(s) from your project:

Explain usage in your project:

Required Functionality

Supplier

Brief code excerpt(s) from your project:

Explain usage in your project:

Customer

Brief code excerpt(s) from your project:

Explain usage in your project:

Profit Margin

Brief code excerpt(s) from your project:

Explain usage in your project:

Probability

Brief code excerpt(s) from your project:

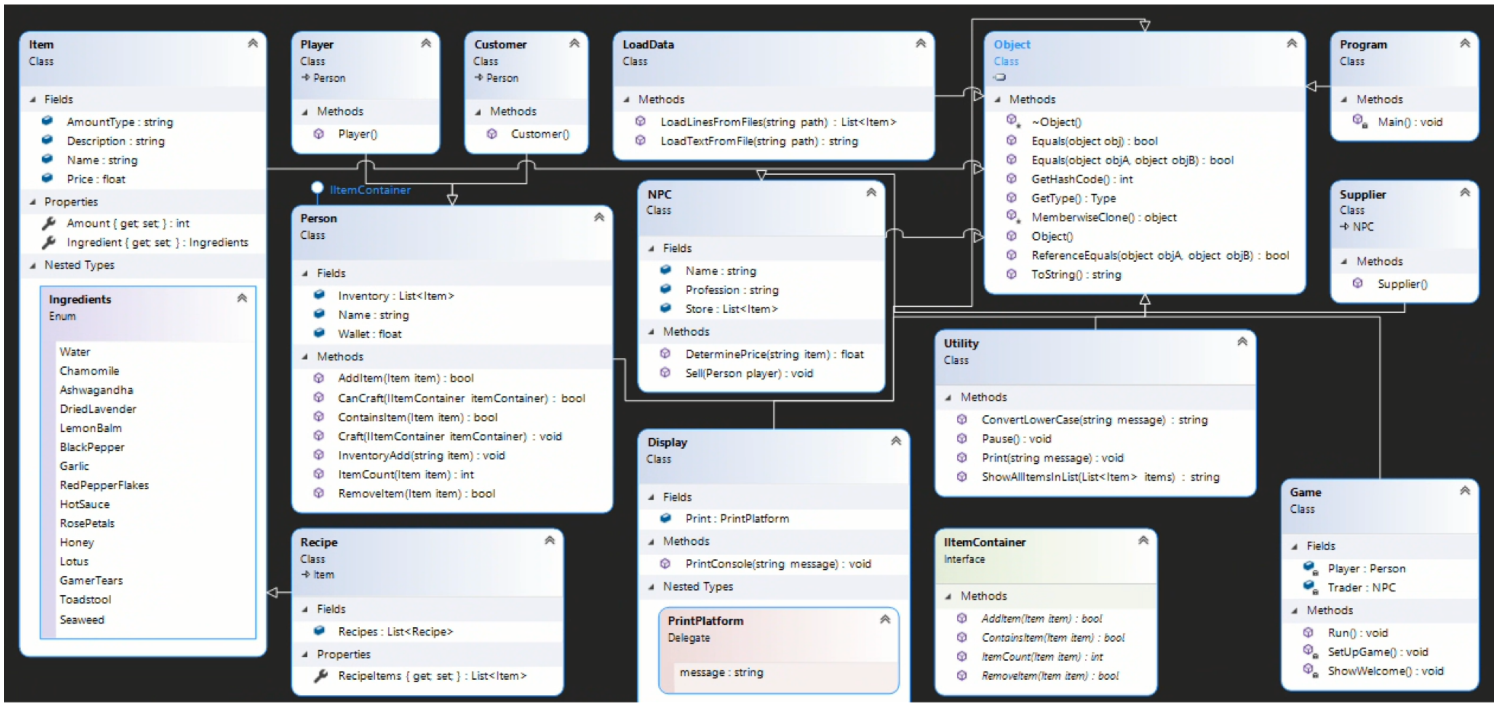
Explain usage in your project:

Generalized Craft Algorithm

Brief code excerpt(s) from your project:

Explain usage in your project:

UML Diagrams



UML Diagram Explanation

Describe your structure and explain your design decisions (i.e., the rationale for your decisions). Include information about inheritance, relationships between classes, and the types of structures that you chose.

Playtesting

Test Session (March 26, 2022)

1) *Alexandra Dimitrovici*

Most Successful Project Aspects

- *The player can add things to their inventory by shopping at the store.*
- *The money in the player's wallet gets removed when you purchase something from the store.*
- *The player gets asked what they want to do next after completing the purchase.*
- *The funds in the player's wallet are clearly displayed before and after shopping at the store.*
- *The descriptions, prices, and names of the items in the store are displayed.*

Biggest Project Issues

- *The player cannot sell items to a customer, and therefore cannot increase their funds.*
- *The player cannot craft items.*
- *The items purchased from the store are not removed from the store's inventory afterward.*

How did you use what you learned to improve your application?

- I split up my methods and classes so that I wouldn't have too much code in one place.
- I created new methods to check the inventory for the required items to craft the recipe, remove the required items from the player's inventory, and add the new crafted items into it.
- I created a customer class and gave it a wallet with enough funds to purchase crafted items from the player.

Credits

- Recipe List in Recipe Class, Inventory List in Person Class, and Ingredients enum in Item class made with information from https://stackoverflow.com/questions/23208087/c-sharp-crafting-logic
- IItemContainer class, ItemCount + AddItem + RemoveItem + ContainsItem + Craft + CanCraft Methods in Person class created with information from https://www.youtube.com/watch?v=gZsJ_rG5hdo

Research

- *https://stackoverflow.com/questions/23208087/c-sharp-crafting-logic*
- *https://www.youtube.com/watch?v=gZsJ_rG5hdo*
- *https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance*
- *https://www.techopedia.com/definition/3787/encapsulation-c*
- *https://www.youtube.com/watch?v=hypDgKReP0c*
- *https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/polymorphism*
- *https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles*
-

Instructor Provided Research

Bolognia, Jean L. "Complementary and Alternative Medicine." Science Direct. https://www.sciencedirect.com/topics/medicine-and-dentistry/chamomile (accessed February 3, 2022).

Buckle, Jane. "Basic Plant Taxonomy, Chemistry, Extraction, Biosynthesis, and Analysis". Science Direct. https://www.sciencedirect.com/science/article/pii/B9780443072369500096 (accessed February 3, 2022).

En.wikipedia.org. "Potion." Wikipedia. https://en.wikipedia.org/wiki/Potion (accessed February 3, 2022).

Healthline.com. "9 Proven Health Benefits of Ashwagandha." Healthline. https://www.healthline.com/nutrition/ashwagandha (accessed February 3, 2022).

Juarascio, Adrienne, Norma G.Cuellar, and Nalaka S.Gooneratne. "Alternative Therapeutics for Sleep Disorders." Science Direct. https://www.sciencedirect.com/science/article/pii/B978143771703710009X (accessed February 3, 2022).

WebMD.com. "Ashwagandha - Uses, Side Effects, and More." WebMD. https://www.webmd.com/vitamins/ai/ingredientmono-953/ashwagandha (accessed February 3, 2022).

WebMD.com. "Alternative Treatments for Insomnia." WebMD. https://www.webmd.com/sleep-disorders/alternative-treatments-for-insomnia (accessed February 3, 2022).