Lunatia Project

DESIGN REPORT

Pattarapol Tantechasa, 103883220

Object Oriented Programming

COS20007

Table of Contents

Introduction

Lunatia Project is a console-based game that extends from a Swin-Adventure case study. Several features had been added to enhances and provide more gameplay experience to the user. List of features can be found below.

1. Gathering System
2. Crafting System
3. Recipes object
4. Integration of story
5. Reading data from JSON and text file

Design Patterns

There are three design patterns this project extends from the original Swin-Adventure: Factory, Singleton and Facade pattern. These three-design pattern provide solution to feature this project needed, it helps me to design my project to be more maintainable, well structure and easy to use.

FACTORY PATTERN

The Factory Pattern handled everything about creating object, this including creation of Item, Recipe, Gatherable Object, Location and Path. With these factory class for different type of object, I'm able to add method to create multiple objects at once or even create objects from an external data file like JSON and text file. This helps the creation process of the program to be more organize and improves readability.

SINGLETON PATTERN

The Singleton Pattern is used to handle every recipe object in one place, which is a recipe book object. It also used to make sure there could be only one recipe book exist in the game. Recipe book object handle everything related to Recipe object, including getting a list of recipes and locating a recipe. Without this object player wouldn't be able to do these things.

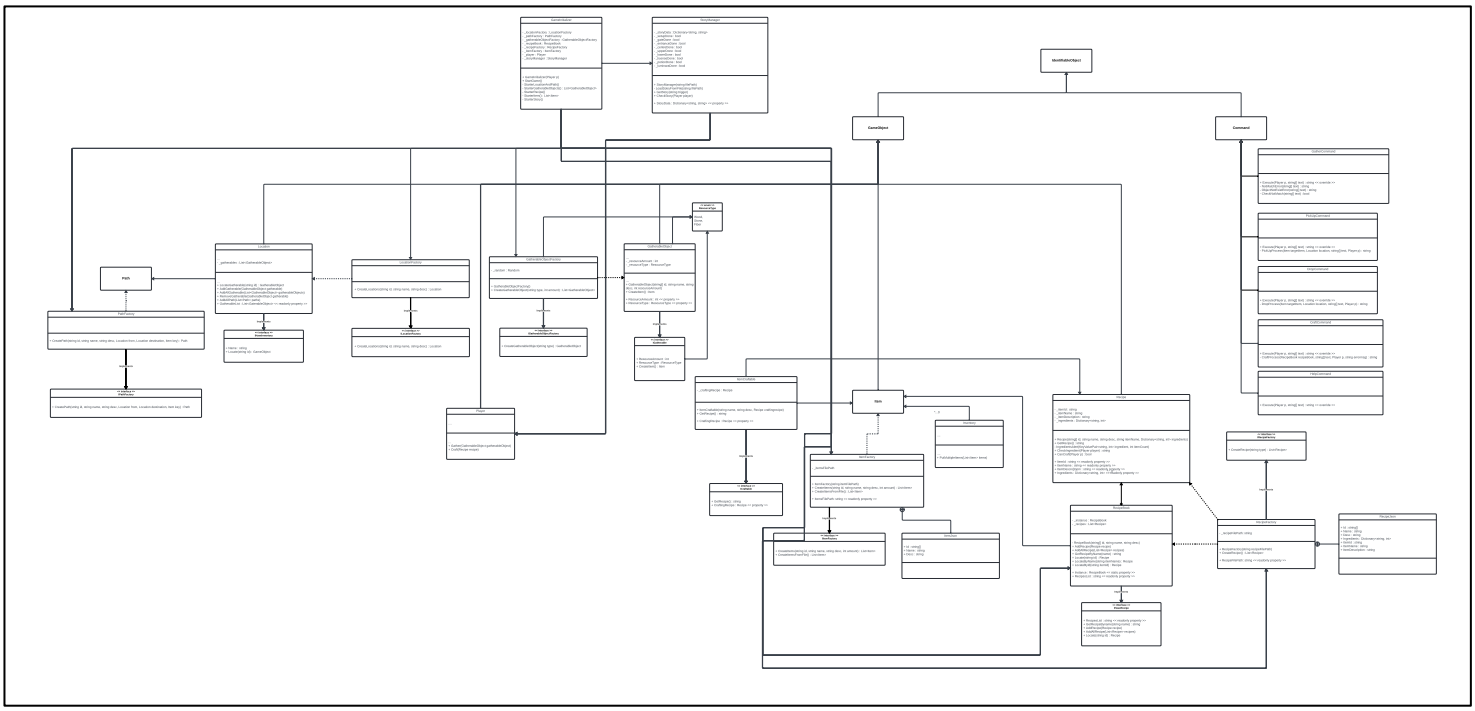
FACADE PATTERN

The Facade Pattern had been used for mainly setting up environment for the game. I call methods from many classes including factory classes to make sure the game had been correctly setup as intended. It also calls methods from Story Manager class to output story to player at the start of the game.

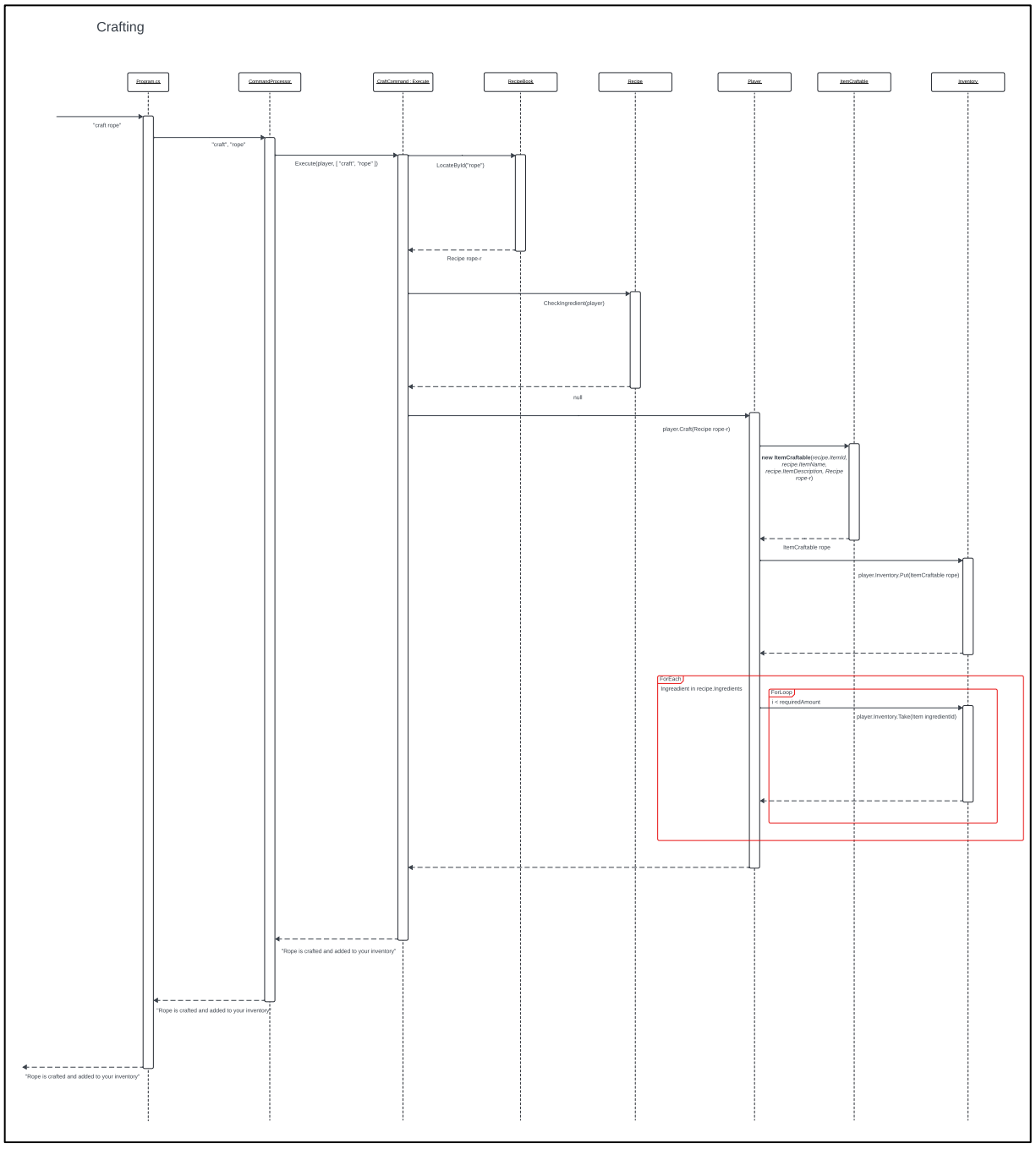
Output Screenshot

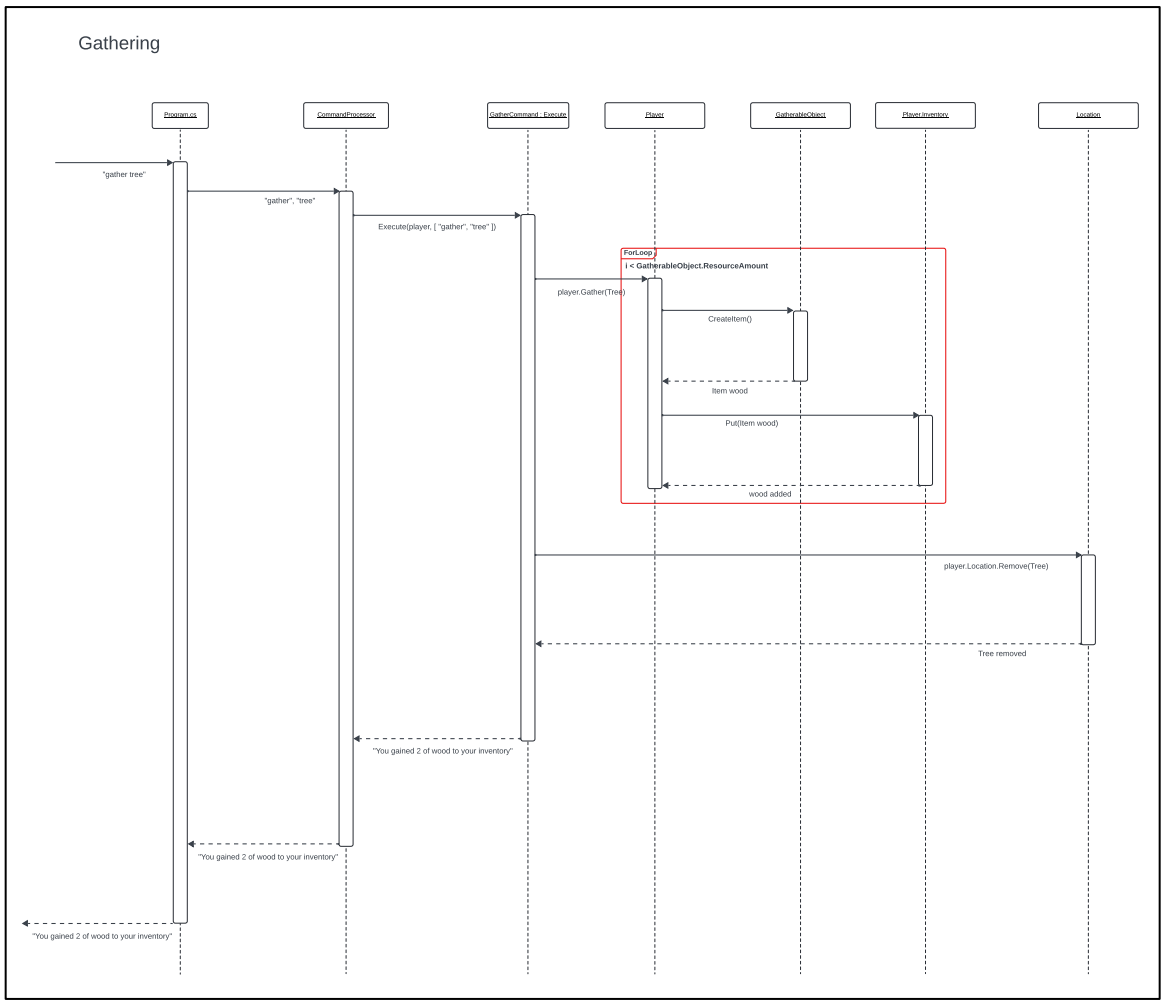
```
_________________________
Welcome to Adventurer!
_________________________

Enter player's name:
Pattarapol Tantechasa
Enter player's description:


You awaken, disoriented, in a place unknown.

Around you, trees sway, rocks sit, and grass rustles.

This forest feels unfamiliar... what will you do now?

Enter a command:
look at here
Lunatia Forest, a myterious forest located in Lunatia region. (lunatiaforest)
Location contains:
        No Item
Exist Paths:
        1. Lunatia City Front Gate, a path from Lunatia Forest to a Lunatia City Front Gate (east)
        2. Lunatia Forest Upper, a path from Lunatia Forest to Lunatia Forest Upper (uphill)
        3. Lunatia Forest Lower, a path from Lunatia Forest to Lunatia Forest Lower (downhill)
List of Gatherables:
        Plenty of Tree, A normal tree that gives you wood. (tree)
        Plenty of Rock, A normal rock that gives you stone. (rock)
        Plenty of Grass, A normal grass that gives you fiber. (grass)


Enter a command:
look at me
You are Pattarapol Tantechasa
You are carrying
        1 x Recipe Book (recipebook)


Enter a command:
look at recipebook
A List of All Recipe:
        1. Rope Recipe, a recipe for crafting a rope (rope-r)
        2. Strong Rope Recipe, a recipe for crafting a strong rope (strongrope-r)
        3. Ladder Recipe, a recipe for crafting a ladder (ladder-r)
        4. Clear Potion Recipe, a recipe for crafting a clear potion (clearpotion-r)


Enter a command:
look at uphill
Lunatia Forest Upper Path, a path from Lunatia Forest to Lunatia Forest Upper that requires Ladder to tr
avel through.

Enter a command:
look at east
Lunatia City Front Gate Path, a path from Lunatia Forest to a Lunatia City Front Gate.

Enter a command:
move east
You have moved to : Lunatia City Front Gate

A huge gate looms before a vast city.

Yet, no signs of life stir behind walls.

Enter a command:
look around
Lunatia City Front Gate, a Front Gate to enter Lunatia City. (lunatiacitygate)
Location contains:
        No Item
Exist Paths:
        1. Lunatia Forest, a path from Lunatia City Front Gate to Lunatia Forest (west)
        2. Lunatia City Entrance, a path from Lunatia City to Lunatia City Entrance (east)
List of Gatherables:


Enter a command:
look at east
Lunatia City Entrance Path, a path from Lunatia City to Lunatia City Entrance that requires Lunatia City
 Entrance License to travel through.

Enter a command:
move east
You can't travel through this path yet. What item could help you get through this path?

Enter a command:
move west
You have moved to : Lunatia Forest

Enter a command:
gather tree
You gained 16 of Wood to your inventory

Enter a command:
mine rock
You gained 22 of Stone to your inventory

Enter a command:
cut grass
You gained 23 of Fiber to your inventory

Enter a command:
craft rope
Rope is crafted and added to your inventory
```

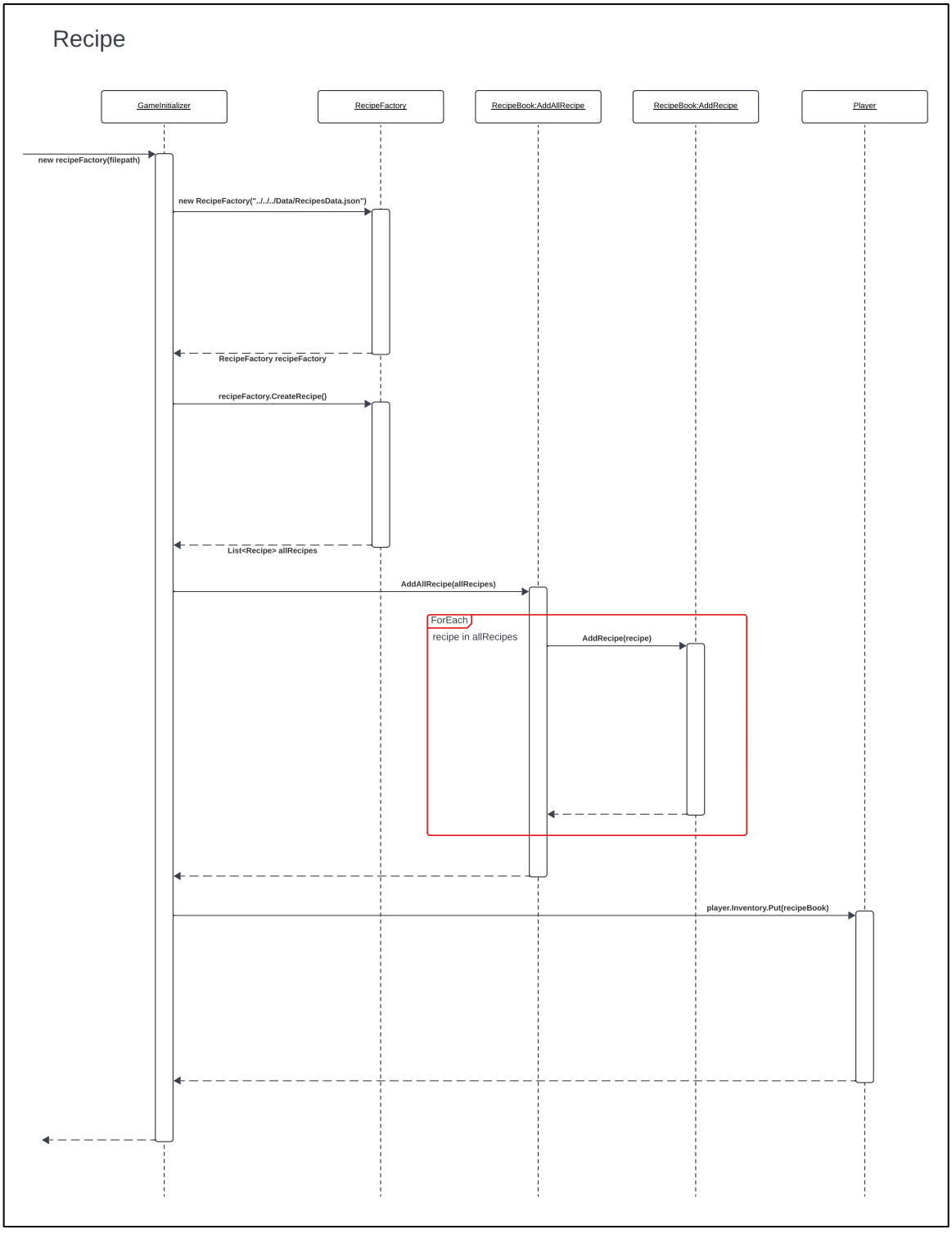
UML Class Diagram

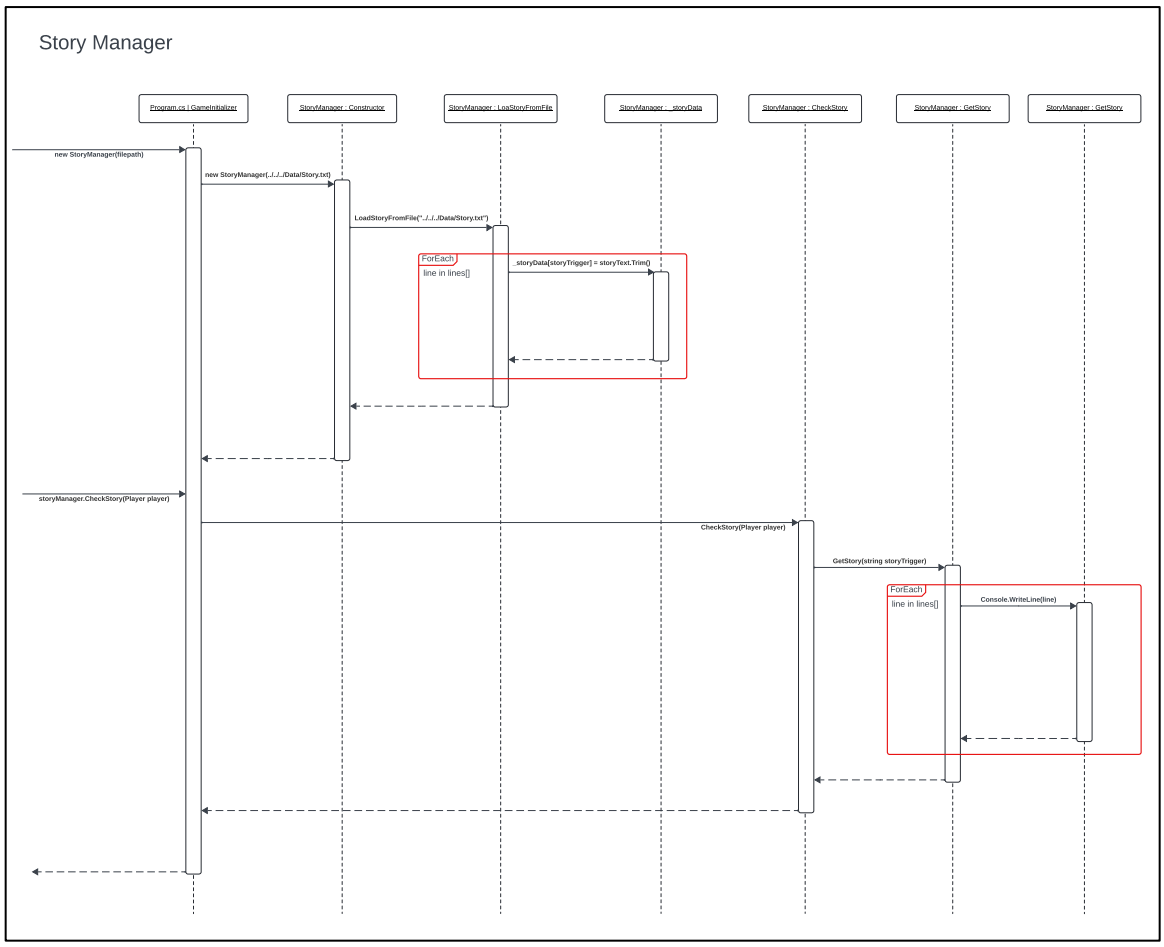


Sequence Diagrams




Gathering
Program.cs
CommandProcessor
GatherCommand : Execute
Player
GatherableObject
Player.Inventory
Location
"gather tree"
"gather", "tree"
Execute(player, ["gather", "tree"])
ForLoop
i < GatherableObject.ResourceAmount
player.Gather(Tree)
CreateItem()
Item wood
Put(Item wood)
wood added
player.Location.Remove(Tree)
Tree removed
"You gained 2 of wood to your inventory"
"You gained 2 of wood to your inventory"
"You gained 2 of wood to your inventory"


Recipe
GameInitializer
RecipeFactory
RecipeBook:AddAllRecipe
RecipeBook:AddRecipe
Player
new recipeFactory(filepath)
new RecipeFactory("../../../Data/RecipesData.json")
RecipeFactory recipeFactory
recipeFactory.CreateRecipe()
List<Recipe> allRecipes
AddAllRecipe(allRecipes)
ForEach
recipe in allRecipes
AddRecipe(recipe)
player.Inventory.Put(recipeBook)



Roles and Responsibilities

Roles and Responsibilities	Type	Notes
RecipeFactory, ItemFactory, PathFactory, GatherableObjectFactory, LocationFactory	Class	This all are handling everything related to object creation based on its type. Some can create object by reading data from JSON file, while other making it easier to create multiple objects in one line. It helps simplify a creation of objects in the game.
IRecipeFactory, IItemFactory, IPathFactory, IGatherableObjectFactory, ILocationFactory	Interface	This set a rules or contracts for what factory class of each object need to have.

RecipeBook	Class	This class will handle all things related to recipe object apart from creation process. All recipe created will be store in this object and only one RecipeBook object can exist in the game. Player will need this to able to craft thing because they need to be able to get a recipe for item that they want to craft.
GameInitializer	Class	This class responsible for setting up environment in the game. Call creation method from factory classes of each object. This also call method to get story from StoryManger class, to provides a setup story for player at the beginning of the game.
StoryManager	Class	This class responsible for everything related to storytelling, it will load data from external text file and store it as dictionary variable. We could later get those stories with story trigger name and gain output the value of that story to the player.
GatherCommand	Class	Handles every command related to gathering, allows player to input a gather command, output any input error if exist and perform gathering process.
CraftCommand	Class	Handles every command related to crafting, allows player to input crafting command. This allows player to input craft command by item's name, id or a recipe id of that item.
PickUpCommand	Class	Allow player to input command to pick up item exist in their current location to their inventory.
DropCommand	Class	Allow player to input command to drop item from their inventory to their current location.
HelpCommand	Class	Allow player to input command to get their help with features in the game, this will provides player

		information of what they can do and how to guides of playing the game.
ResourceType	Enum	This provides an all types of resource existed in the game, if later we want to change any value of these resources. We could change only in this file which improve code maintainability.
ICraftable	Interface	Sets rules for item that are craft able, that they need to be able to get their own recipe unlike normal item object.
IHaveRecipe	Interface	Sets rules for RecipeBook that it needs to be able to have list of recipe and methods to add single or multiple recipes and locate recipe.
IGatherable	Interface	Sets rules for GatherableObject that it needs to be able to create resource item based on its type by amount that they hold.
GatherableObject	Class	A class for gatherable object, this stores everything that it needs to know for what it needed to be able to do.

Integrated New Knowledge

LANGUAGE-INTEGRATED QUERY (LINQ)

LINQ enable us to query data (e.g. filter, sort, group etc.) in a similar way we would write SQL queries but integrated within C#. The project used LINQ to rework a how it output list of objects for class like Inventory and Location. It enables the project to group objects by name or short description and get count of that group, then output count of an object follow by its name and description. This eliminated a repeated multiple line of same object in the output, replaced with one line but with count in front of it.

System.Linq

https://learn.microsoft.com/en-us/dotnet/api/system.linq?view=net-7.0

GroupBy Method

https://learn.microsoft.com/en-us/dotnet/api/system.linq.enumerable.groupby?view=net-7.0#system-linq-enumerable-groupby-4(system-collections-generic-ienumerable((-0))-system-func((-0-1))-system-func((-0-2))-system-func((-1-system-collections-generic-ienumerable((-2))-3)))

Select Method

https://learn.microsoft.com/en-us/dotnet/api/system.linq.enumerable.select?view=net-7.0

WORKING WITH JSON FILE

"System.Text.Json" is a built-in .NET library that enable us to work with JSON data file. It provides functionality for serialization (convert object to JSON format) and deserialization (convert JSON data into objects). This project will deserialize JSON data format of Recipes and Items, used those to create all recipes and items at once in the setup process of the game. It will first read as a JSON file as a string, then deserialize that string into a JSON object created private to that class. Lastly, map information in that JSON object in the Recipe or Item object in the creation process.

System.Text.Json

https://learn.microsoft.com/en-us/dotnet/api/system.text.json?view=net-7.0

JSON Deserialization

https://learn.microsoft.com/en-us/dotnet/api/system.text.json.jsonserializer.deserialize?view=net-7.0#system-text-json-jsonserializer-deserialize(system-text-json-jsondocument-system-text-json-serialization-metadata-jsontypeinfo)

WORKING WITH TEXT FILE

Story Manager class will read story data from text file, and store that in a dictionary of string and string. Story trigger will store as a key, and the actual story data will be store as a value. Once the method *CheckStory()* has been called, we could get story based on the story trigger name given.