Project 1 Metrics

Weighted Methods per Class:

I've counted the lines of methods per class by looking at the total number of lines in each class and subtracting empty lines and method signatures. This gives us the following list:

Brewery	38 lines
Container	6 lines
ContainerManager	7 lines
Ingredient	7 lines
Inventory	14 lines
Library	7 lines
List	15 lines
Recipe	13 lines
Average	**13.375 lines**

This is quite a decent weighted methods per class. We have the advantage that what we are trying to do is not too difficult compared to a lot of other programming challenges, but we managed to keep lines per class to a relative minimum.

Depth of Inheritance Tree:

For the Depth of Inheritance Tree metric we totalled zero tallies. This is because we didn't use any inheritance in our code. We decided not to use inheritance to lower the complexity of our code, and didn't see any places that required inheritance. Making it more simple to have better cohesion between our classes.

Number of Children:

Once again, given that we have no inheritance, we have no children inheriting from parents in our code. This is good from a complexity perspective because changes in each class are isolated to that class and do not affect subclasses.

Coupling Between Objects:

The following graph shows for each class which other classes are mentioned at least once.

Class	**What the class references**
Brewery	Inventory, ContainerManager, Library, Ingredient, Recipe, Container, List
Container	None
ContainerManager	List, Container
Ingredient	None
Inventory	List, Ingredient
Library	List, Recipe
List	None
Recipe	List, Ingredient

But this is a little difficult to look at, because we want to see what may be affected when we change a class. An easier way to look at this is with the following inverted chart:

Class	**What references the class**
Brewery	None
Container	Brewery, ContainerManager
ContainerManager	Brewery

Ingredient	Brewery, Inventory, Recipe
Inventory	Brewery
Library	Brewery
List	Brewery, ContainerManager, Inventory, Library, Recipe
Recipe	Brewery, Library

Here, we can easily see which classes we should worry about when we make changes to any other class. Any changes to any class may propagate to the brewery, but changes in the brewery will not affect any other class. As another example, since list is mentioned in many different classes, if we change list, we may have to change code in a lot of different places.

Response for a Class:

For our classes, we made sure to not have too many methods in them. A lot of our instances outside of the class itself are used in the main. For example, Recipes methods are used in the main class(Brewery) to create a new recipe to make a batch. We tried to limit the amount of methods inside a class to make sure the class has only one purpose to serve.

Class	**Methods**	**Instances outside of class**
Container	2	2
ContainerManager	3	2
Recipe	4	12 (most in Brewery/Main)
List	7	18 (most in Brewery/Main)
Library	5	8
Ingredient	4	14 (most in Brewery/Main)

Inventory	6	9
Brewery	2	0

Cohesion Across Methods:

In our methods, the attributes are for the most part uniform. The methods all relate to their classes and behave in a way that is unique to only that class. For example, our methods in Ingredient serve to add/remove ingredients from our stock. It only handles the ingredients and that's it. We tried to make sure a method has one purpose as to not overcomplicate it and to keep it clean. For example, the Library class methods have a purpose to store things in the library or remove them. Each method has one specific purpose.