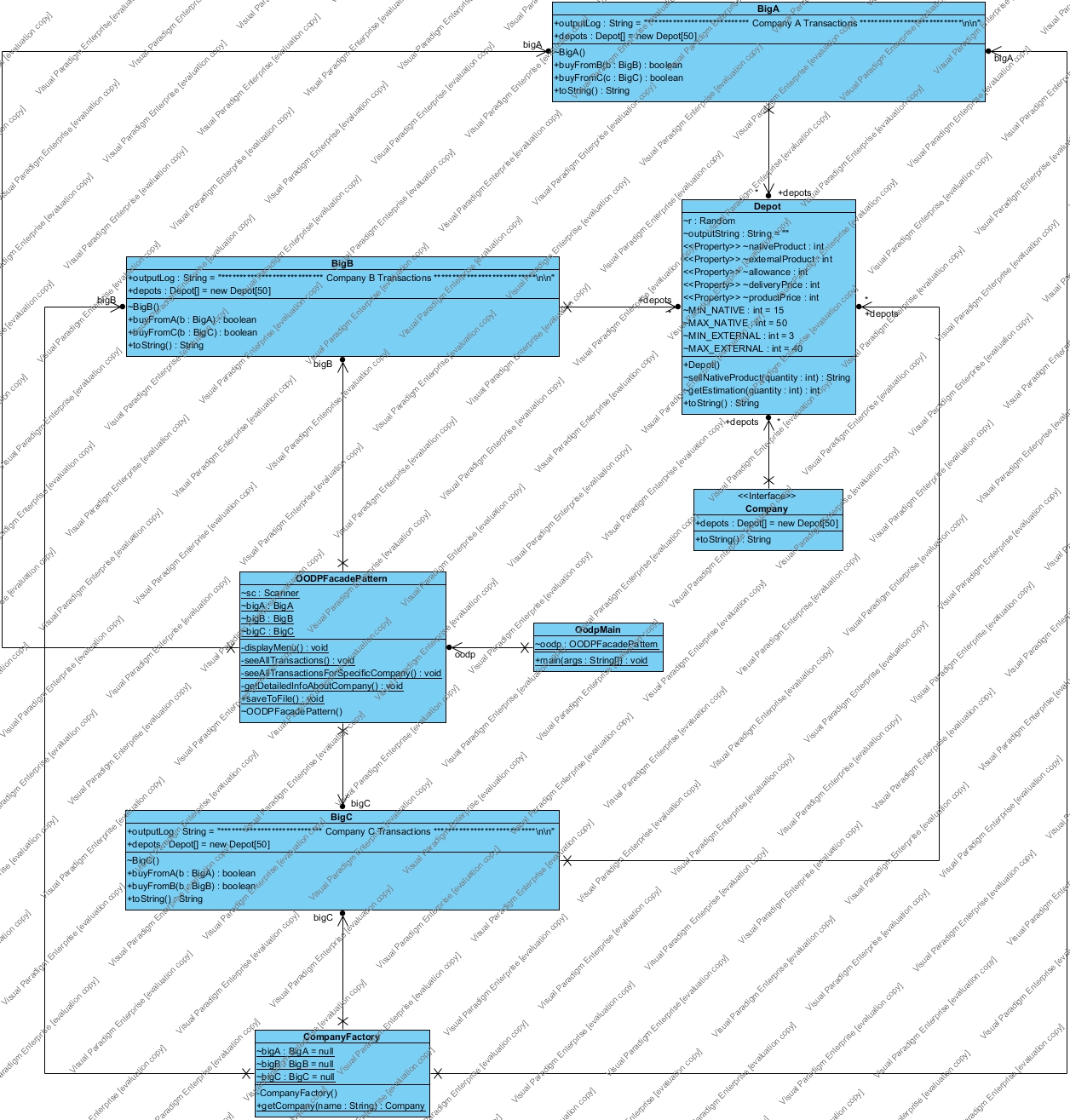
**Class Diagram :**



The design patterns used :

1. Singleton
2. Factory Design
3. Façade

**Singleton:**

In software development, it’s a design pattern which is used to prevent any class to be instantiated more than once. In scenarios, where in whole project or system, you only need one instance of any specific class to operate with, so rather than creating a new one every time it needs to be used, we implement singleton pattern within the class to return any previously created instance, or create a new one if already not instantiated.

In our project, we used this pattern in the CompanyFactory class which is responsible for creating objects of several companies and help reusing those objects throughout the simulation.

**Factory Design:**

This is a type of creational design patterns because it provides one of the efficient and effective way to create an object. It’s one of the most used In Java too. In factory pattern, we instantiate any useful classes without exposing the underline creational logic to the user. In our project, we implemented a Company interface with basic attributes and created concrete classes regarding different companies implementing that interface. Where ever these companies’ references are to be used, CompanyFactory is used as in between to provide those references without exposing actual creation logic of these companies’ object creation.

**Façade Pattern:**

It’s one of the object oriented design patterns used to hide the complexities of any system, and make use of an interface to provide access to the user It’s one of the structural design patterns but in our case, it’s making use of OOPFacadePattern class to handle all the complex operations which operate with factory pattern accessibilities and grant access to OodpMain to operate all those functionalities keeping it simple and clean.