**Analysis:**

I have used **Factory design** pattern in my solution for Q2 and Q3. Factory pattern is a creational design pattern which is one of the best ways of creating an object in Java.

The primary benefit of Factory design pattern is it aids in creation of objects without exposing the creation logic to the user and also helps to refer to newly created object using a common interface.

For my solution, using the Factory design pattern hides the implementation of creation of various specialized Nodes such as the Statement, Sequence, Number Expression, Infix Expression etc. Just by using the factory methods the different specialized nodes can be created. This technique hence encapsulates the complex creation process of specialized nodes and provides a simple interface to help create them. Also it abstracts the way specific constructors for objects are created and the various constraints handled are hidden. It also helps to overcome the problem of creating a **new** object for each specialized node which was tedious in Q 1 and eases the process of creation of objects of different specialized nodes by creating just a single factory object.

The factory pattern also helped to easily create test cases by simply referencing the factory class methods using a factory object. The collection of report for the number of objects instantiated for each class was also made simple by declaring one common report () method in factory class rather than creating a report () method in all specialized nodes. The factory design pattern helps to achieve loose coupling.

For Q 4, I have made use of **Singleton design** pattern. In this design pattern, it consists of a single class which is used for creating objects and it is made sure that only single object for that class is created. It provides a way to access its only object directly without need to instantiate the object of the class.