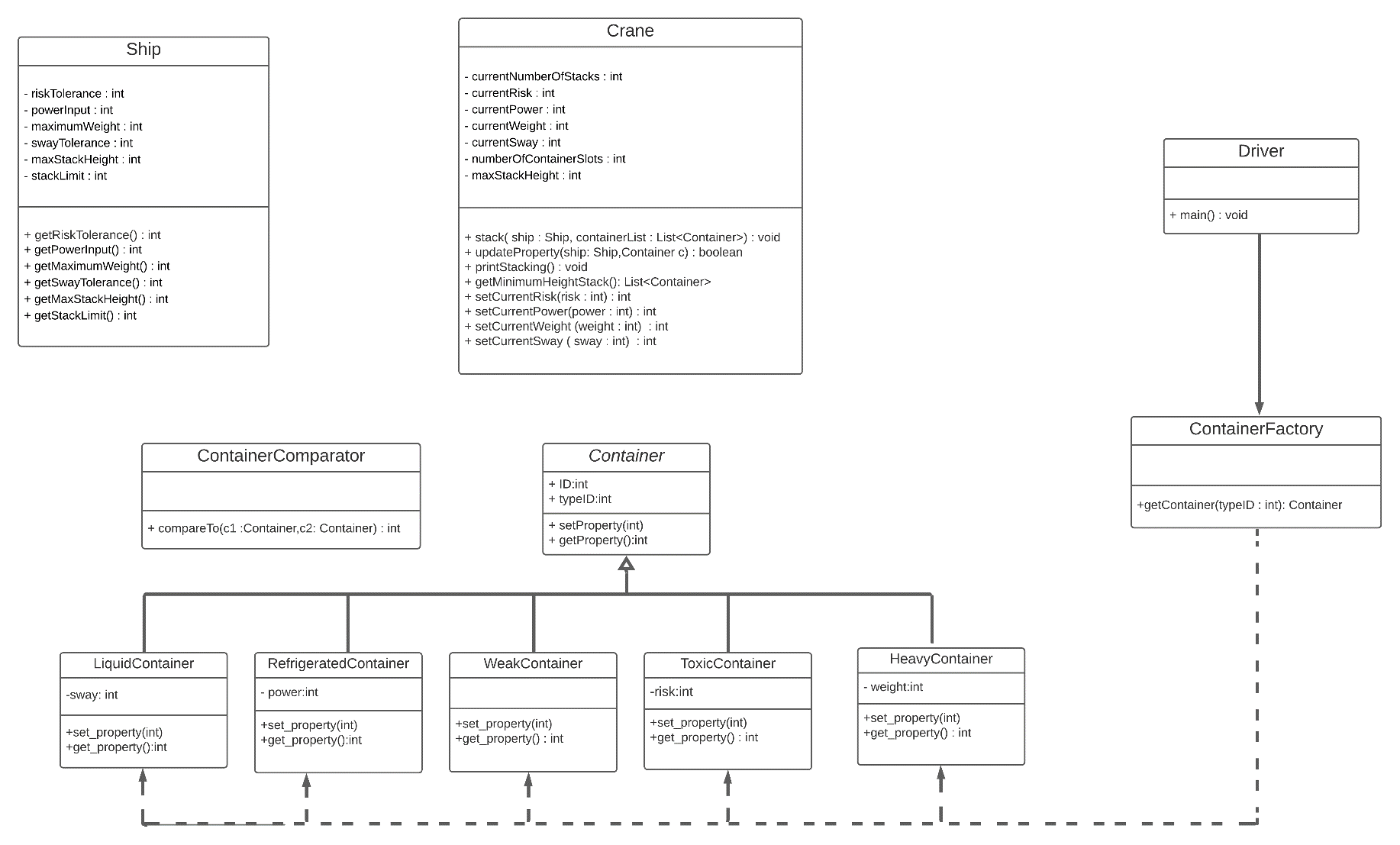
Project Number: **10**

Project Title: **Ships and Cranes**

Member 1: **Shivangi Goel (2018B4A30853P)**

Member 2: **Sanskruti Lohia (2018B1A30659P)**

**Updated UML Class Diagram**



**Brief Description of Various Classes and Algorithm**

* The Container class is implemented using **Factory Design Pattern**.
* Driver class uses an object of ContainerFactory which returns an object of specific type of Container depending on the typeID.
* Container is an abstract class which has different types of concrete class extending it which represent different type of containers.
* The Ship class contains maximum limits of all the properties and the Crane class keeps a track of current status of these properties.
* The stack function in the Crane takes in an object of Ship class and a List of Containers and does prints an output on the following grounds:
* For each Container, we get the stack of containing minimum number of containers.
* The container is added to the stack only if it does not violate any constraint of the ship or the height and weight constraints of the stack.

**Walkthrough of Inputs and Outputs**

**Input Format**

The first line of the file is the number of test cases that are being tested in the project. The layout of each test case is as explained below.

The next six lines of the input correspond to various ship properties like risk tolerance, power input, maximum weight, sway tolerance, maximum stack height, stack limit, respectively. The next line of the input consists of an integer N, number of containers to be stacked by the crane. N lines follow after this, each containing three integers (say a, b, c) which describe individual containers. ‘a’ denotes the serial number (or the ID) of the container, ‘b’ denotes the type (or the type ID), and ‘c’ denotes the exclusive property of the incoming container.

**Output Format**

* Case 1: When a valid stacking is not possible

The code will simply display the statement ‘No Viable Stacking Possible’.

* Case 2: When a valid stacking is possible

There will be N stacks and N lines will be displayed corresponding to each stack with each line having the serial ID of the containers in that particular stack. Consider that the stack limit is 3, the code will display one possible stacking as follows:

The viable stacking is:

Stack 0: id1 id2 id3

Stack 1: id4 id5

Stack 2: id6

**Limitations of the Project**

1. The type of containers considered in this project have only one exclusive property. The containers having mutual properties have been ignored for simplicity. For example, there could be a container which is both heavy and liquid, and thus has both sway and heavy weight.
2. The stack of the ship is assumed to be linear that is the ship is consider to be one dimensional. The ship can be considered as a multidimensional collection of linear stacks.
3. The outlined code gives only a single possible viable stacking of the containers but doesn’t print all the feasible stackings.
4. The stacking algorithm used to solve this method is not the most optimized technique.