SUTD 2021 50.003 Problem Set 1

James Raphael Tiovalen

Q1 & Q2 – Safe Collision-Free Railway Network API

Note that this system implementation prioritizes safety as part of the specified requirement, instead of efficiency. The `List` interface specified here is the `java.util.List` interface. Sample code is included with this document (`Railway.java`).

`class GlobalNetworkManager`	`// Globally manages the railwayNetwork through a specified set of rules.`
`RailwayNetwork railwayNetwork` `List<Train> trains`	 **`association 0..*`**

`void update()`
 	
`void move(Train t)`
 	
`void wait(Train t)`
	`// Update the global time and all of the Train positions.`
	`// Move the specified Train t by its moveSpeed. If the train is at the end of a Track and the next Junction ahead in its trainRoute is empty, set the Track.isOccupied to false, then move the train to that next Junction.`
	`// This is called when the next Track or Junction is occupied.`

`class Train`	

`int trainID`
`int trainType`
`Engine engine`
`RailwayObject currentObject`
`double moveSpeed`
`double position`
 	
`Route trainRoute`
	
	`// 0 - narrow, 1 - meter, 2 - broad.`
	`association 1`
	`association 1`
	
	`// This indicates the Train's position on a specific Track (0.0f-1.0f).`

`void changeEngine(int type)`	`// This is called when the Train object is at a Junction, has trainType 0 (narrow gauge train) and has an upcoming waypoint on its trainRoute. It creates a new Engine and assigns it to this specific Train.`

`class Engine`	
`int engineID`	

`class Route`	
`List<Junction> waypoints`	`// List of target Junctions.`
`void push()` `void pop()`	

`class RailwayNetwork`	`// Similar to a graph network object.`

`List<Track> tracks`
`List<Junction> junctions`
	`// Similar to graph edges.`
	`// Similar to graph vertices/nodes.`
	`// Both:` **`association 0..*`**

`void addTrack(int trackID)`
`void removeTrack(int trackID)`
`void addJunction(int junctionID)`
`void removeJunction(int junctionID)`	

`abstract class RailwayObject`	
`int id` `boolean isOccupied`	

`class Track extends RailwayObject`	

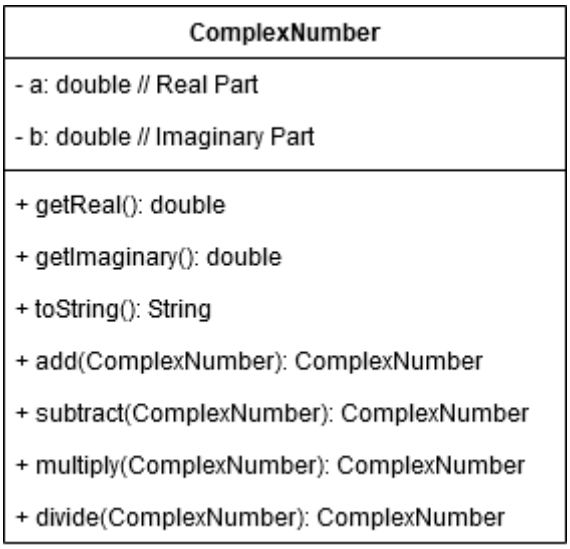
`int trackType`
`double trackLength`
`Junction startJunction`
`Junction endJunction`
	`// 0 - narrow, 1 - meter, 2 - broad.`
	
	association 1
	association 1

`class Junction extends RailwayObject`	
`List<Track> incidentTracks`	**association 0..***
`List<Track> listConnectedTracks(int type)`	`// This is called to list the incident Tracks of a specific trackType that this Junction handles (since each Junction is further divided into individual establishments that exclusively handle a specific track).`

Q3 – Two Complex Numbers Calculator

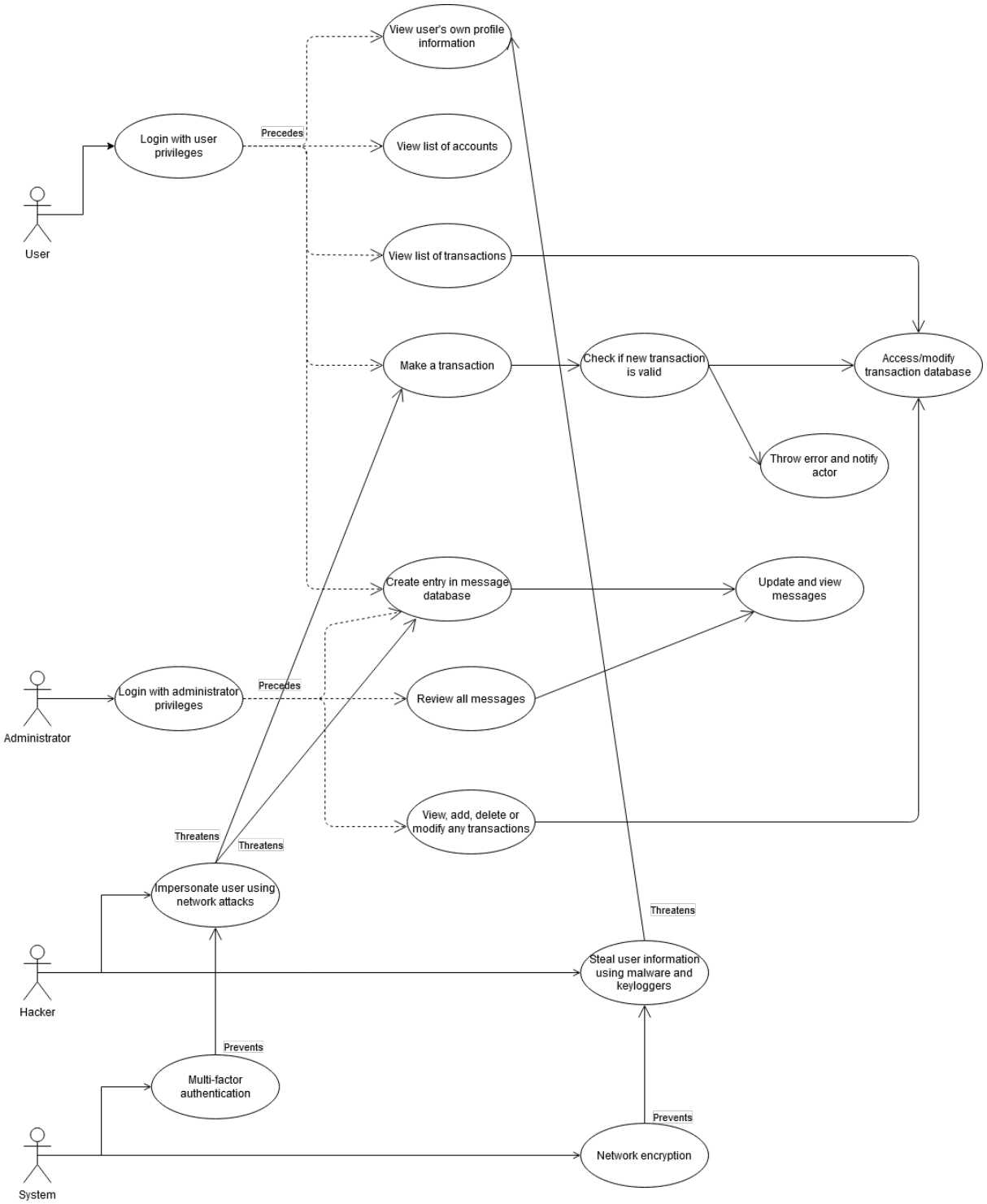
Code is included with this document (`ComplexNumberCalculator.java`).

Class Diagram:



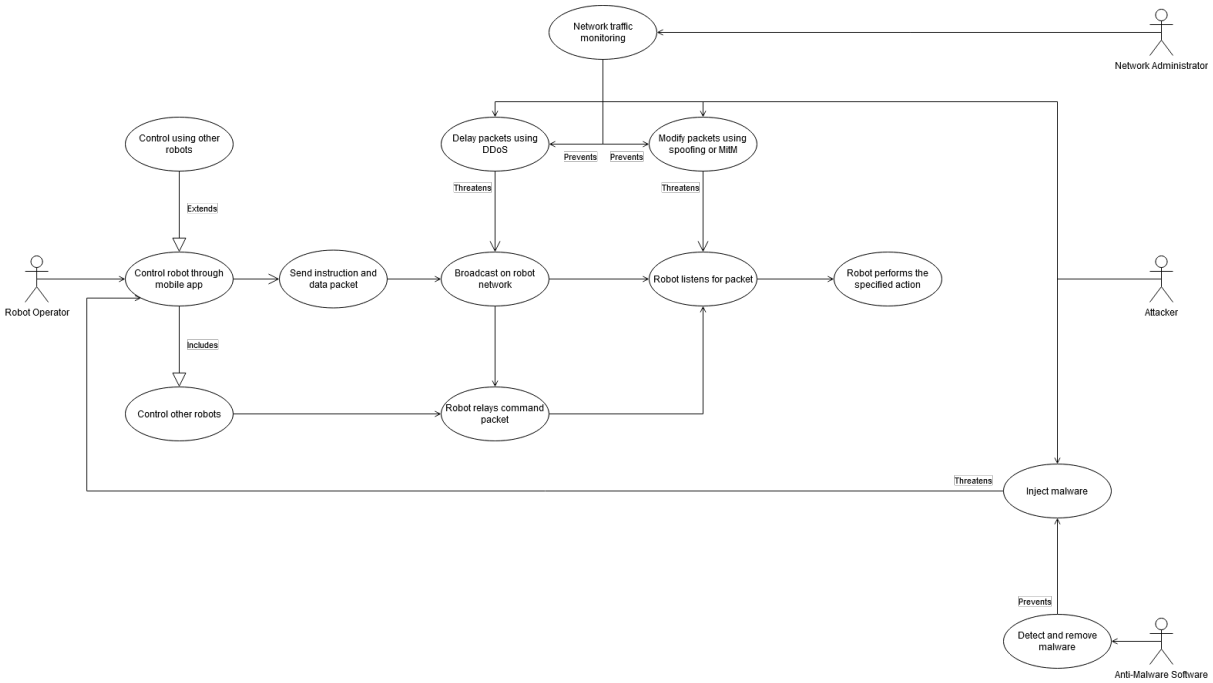
Q4 & Q5 – Augmented KBO Case Study Use Case Diagram

Use Case Diagram:



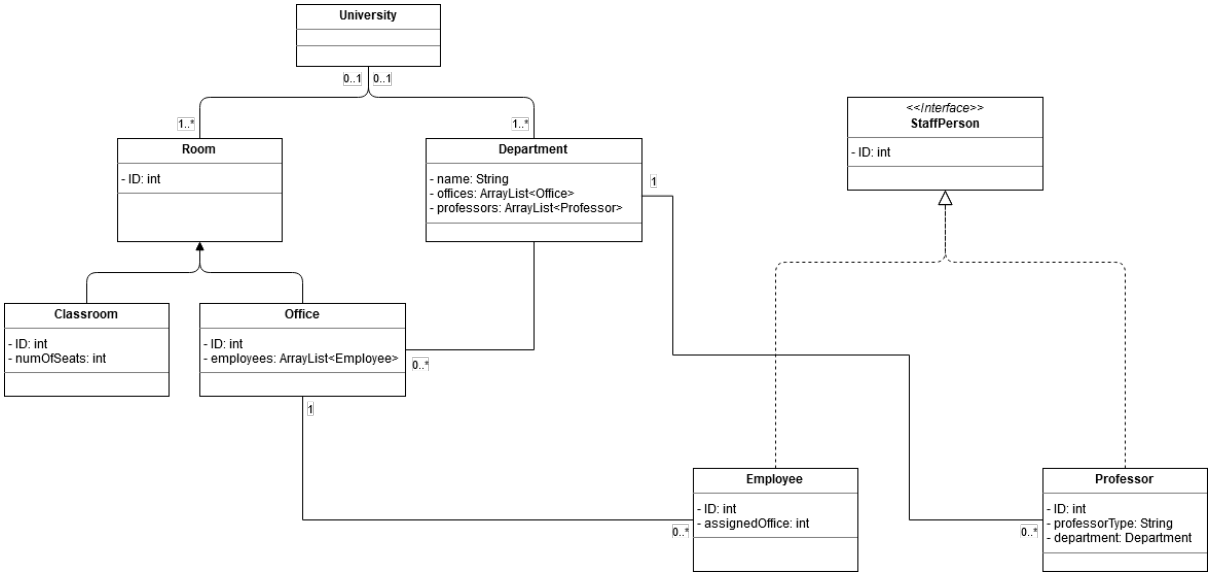
Q6 – Robot Swarm Network Security Use Case Diagram

Use Case Diagram:



Q7 – University HR & Logistics Class Diagram

Class Diagram:



Q8 – Hardware Update Wizard State Machine Diagram

State Machine Diagram: