50.003 Elements of Software Construction
Problem Set 1
Joel Huang 1002530

Exercise 1.2 & 1.3 – Railway Network API

Broadly, a NetworkManager sends global update signals to update the positions of Train objects according to their pre-set movement speed. This implementation prioritizes only safety and not efficiency, meaning that a lot of Trains will be waiting for Junctions and Tracks to be cleared.

`class NetworkManager`	`// Manages the network through a set of rules`
`RailNetwork railNetwork`	
`List<Train> trains`	**`association 0..*`**
`void Update()`	`// updates the global time and all train positions`
`void move(Train t)`	`// moves Train t by its moveSpeed, increasing its position. if at the end of a Track and the Junction ahead is empty, set Track.isOccupied to false, then move the train to a Junction`
`void wait(Train t)`	`// called when the Junction or Track ahead is occupied`

`class Train`	
`int trainID`	
`int trainType`	`// 0 – broad, 1 – meter, 2 - narrow`
`Engine engine`	**`association 1`**
`RailObject currentlyOn`	**`association 1`**
`double moveSpeed`	
`double position`	`// Train object’s position on the Track (0f-1.0f)`
`Route trainRoute`	
`void changeEngine(int type)`	`// called when the Train object is at a Junction, has trainType 2 and has an upcoming waypoint on its trainRoute. Creates a new Engine and assigns it to this Train.`

`class Engine`	
`int engineID`	

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

`class RailNetwork`	`// Loosely analagous to a Graph Object`
`List<Track> tracks`	`// Loosely analagous to Graph Edges`
`List<Junction> junctions`	`// Loosely analagous to Graph Nodes`
	`both associations 0..*`
`void addTrack(int id)`	
`void removeTrack(int id)`	
`void addJunction(int id)`	
`void removeJunction(int id)`	

`abstract class RailObject`	
`int id`	
`boolean isOccupied`	

`class Track extends RailObject`	
`int trackType`	
`double trackLength`	`// 0 – broad, 1 – meter, 2 – narrow`
`Junction startJunction`	**`association 1`**
`Junction endJunction`	**`association 1`**

`class Junction extends RailObject`	
`List<Track> adjacencyList`	**`association 0..*`**

Exercise 2 – Complex Number Calculator

Usage:

```
num1 = ComplexNumber(3,0.5f);
num2 = ComplexNumber(2,0.5f);
print(Calculator.add(num1, num2));

>> 5.0 + 1.0i
```

abstract class ComplexNumber	

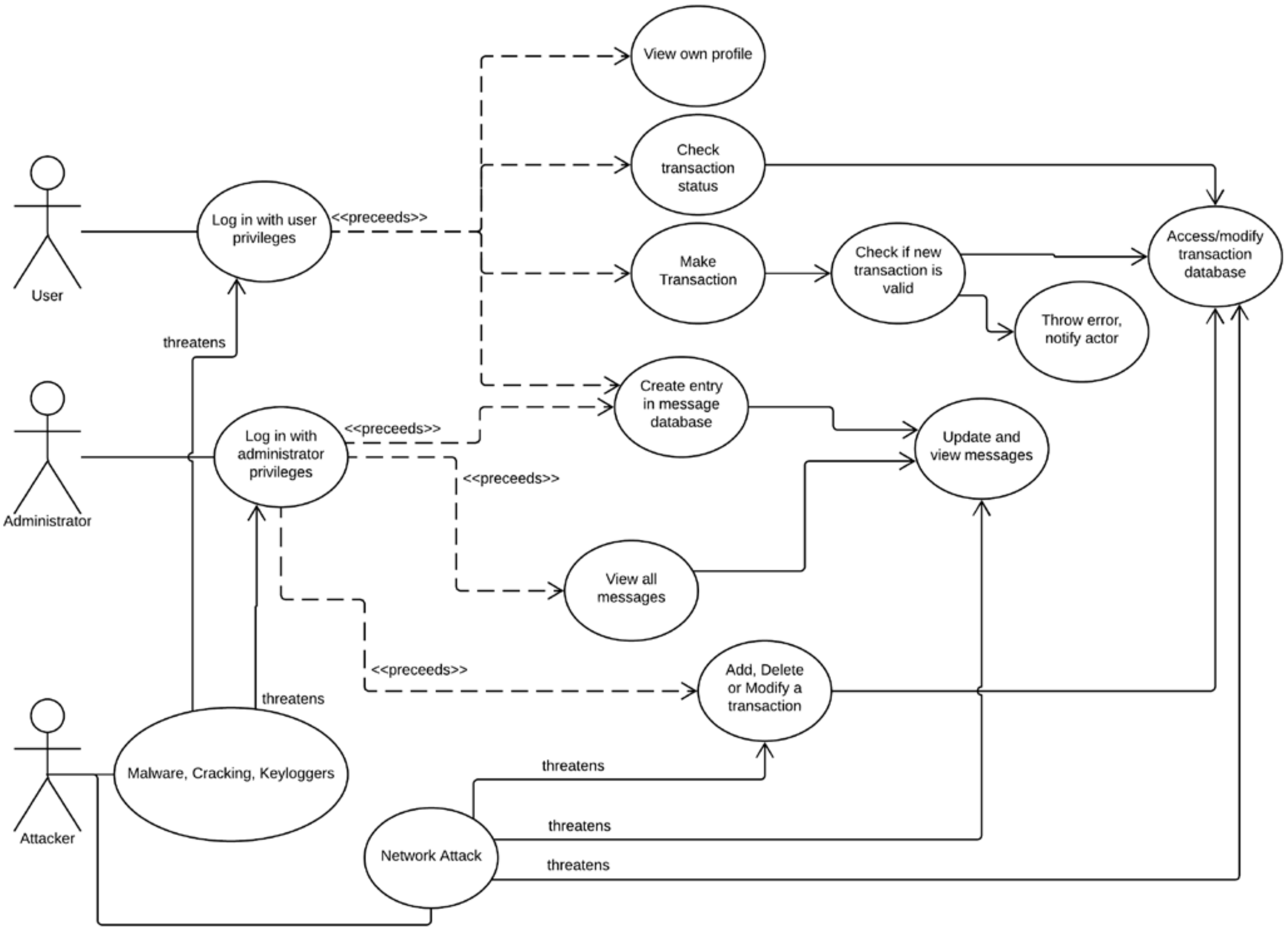
class ComplexNumberRect extends ComplexNumber	
double r	// real part
double i	// imaginary part
ComplexNumberRect(double r, double i)	
String toString()	// Format the print output as r if i = 0, i if r = 0, or r + i otherwise.

class ComplexNumberPolar extends ComplexNumber	
double r	// radius
double theta	// angle
ComplexNumberPolar(double r, double theta)	
String toString()	// Format the print output as r if theta = 0, theta if r = 0, or r + theta otherwise.

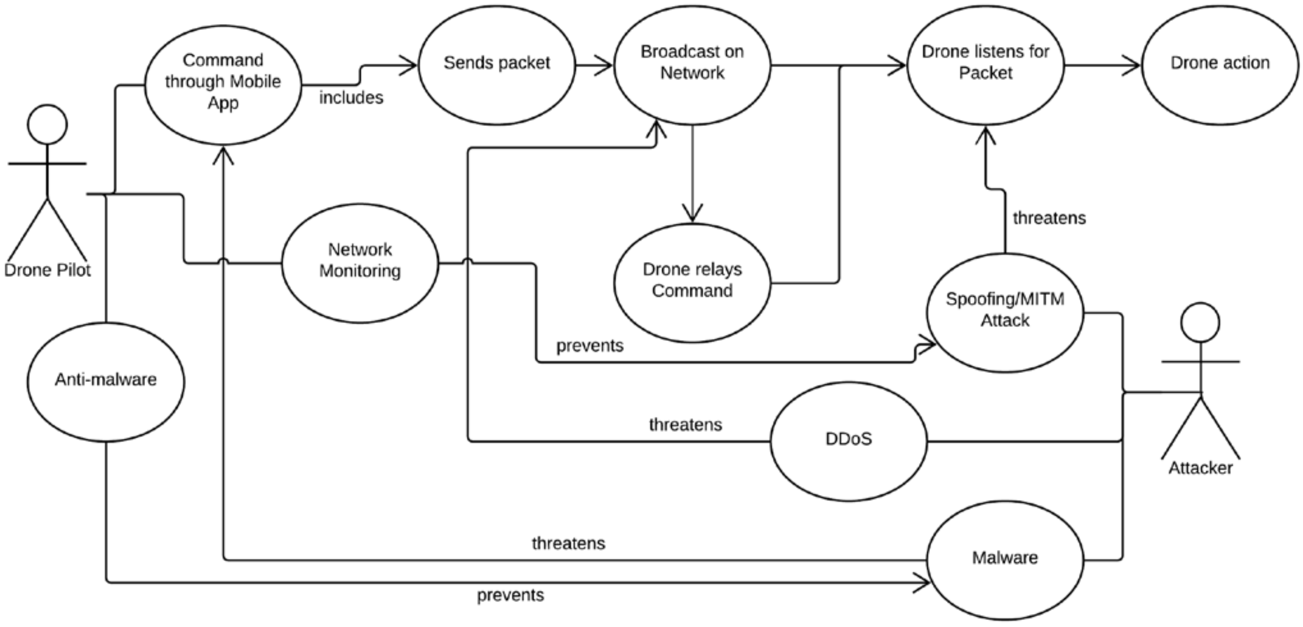
class Calculator	

ComplexNumber add(ComplexNumber num1, ComplexNumber num2)
ComplexNumber subtract(ComplexNumber num1, ComplexNumber num2)
ComplexNumber multiply(ComplexNumber num1, ComplexNumber num2)
ComplexNumber divide(ComplexNumber num1, ComplexNumber num2)
	// addition and subtraction work by applying the operators to the real parts and imaginary parts of num1 and num2 respectively.
	// multiplication follows the formula (x + yi)(u + vi) = (xu + yv) + (xv + yu)i
	// division involves calculating the conjugate and multiplying by it

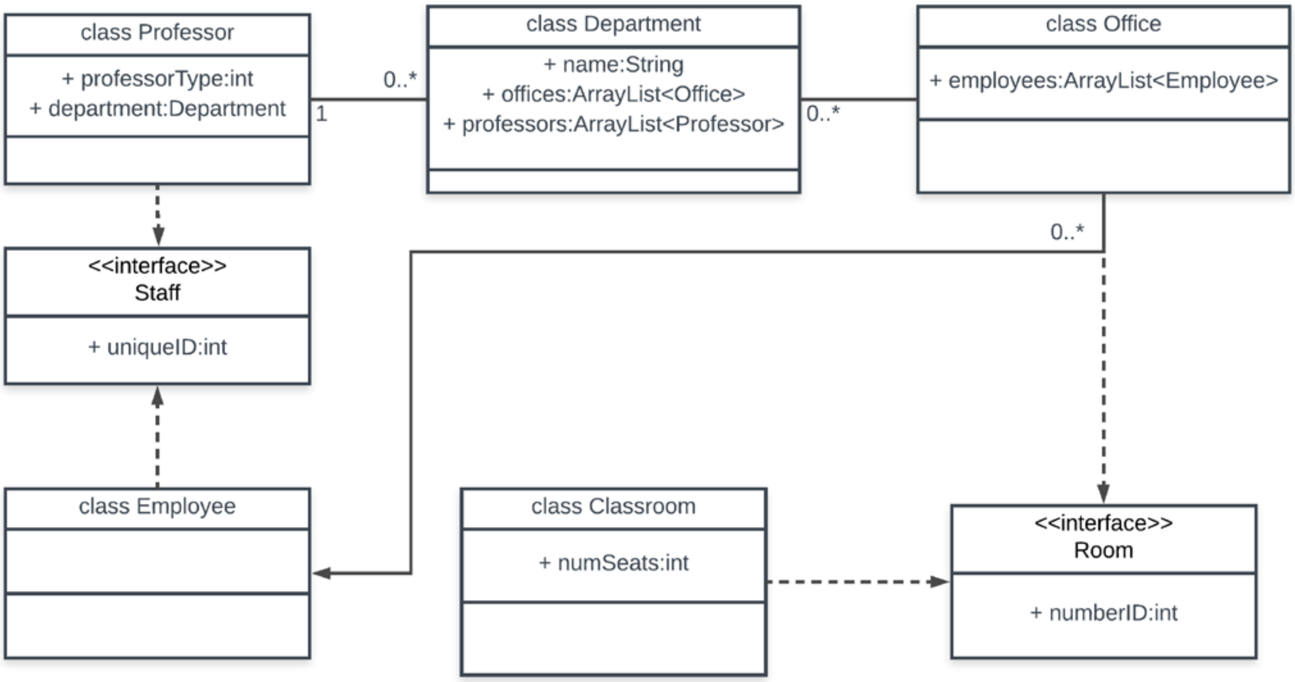
Exercise 4 & 5 – Augmented KBO Use Case Diagram


View own profile
Check transaction status
User
Log in with user privileges
<<preceeds>>
Make Transaction
Check if new transaction is valid
Access/modify transaction database
Throw error, notify actor
threatens
Create entry in message database
<<preceeds>>
Log in with administrator privileges
Update and view messages
Administrator
<<preceeds>>
View all messages
<<preceeds>>
Add, Delete or Modify a transaction
threatens
Malware, Cracking, Keyloggers
Attacker
threatens
threatens
Network Attack
threatens

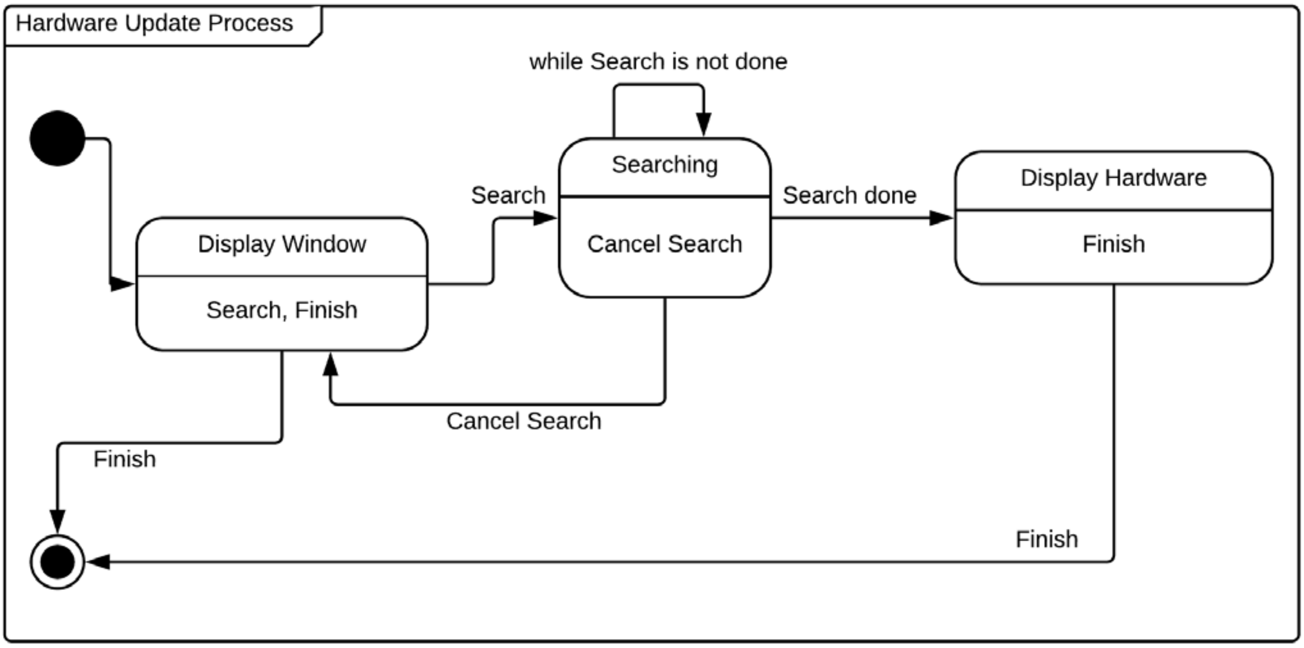
Exercise 6 – Drone Network Security Use-Misuse Case Diagram



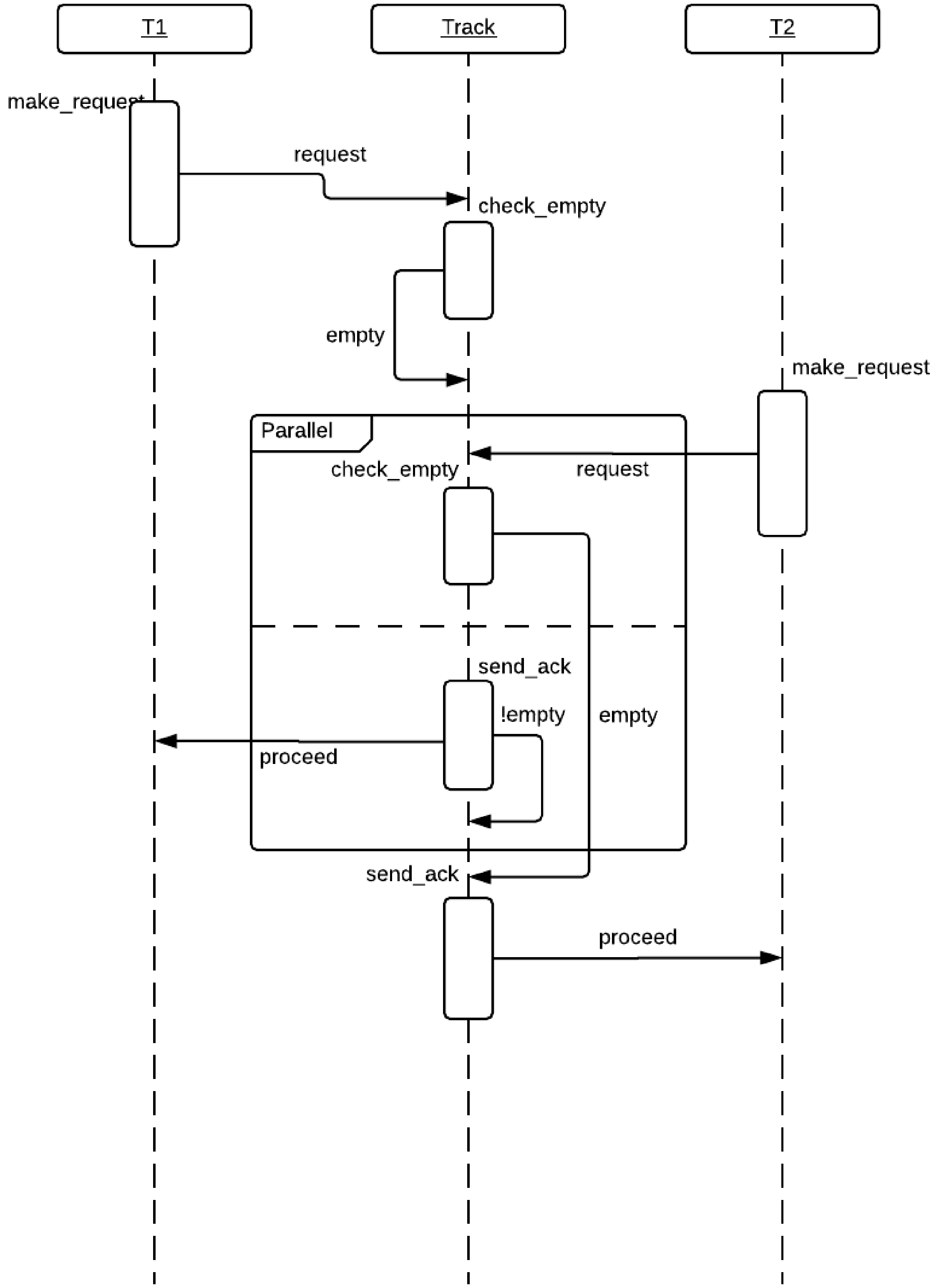
Exercise 7 – University HR/Logistics Class Diagram



Exercise 8 – Hardware Update Wizard State Machine



Exercise 9 – Train Change Track Request UML Sequence Diagram


T1
Track
T2
make_request
request
check_empty
empty
make_request
Parallel
request
check_empty
send_ack
!empty
empty
proceed
send_ack
proceed