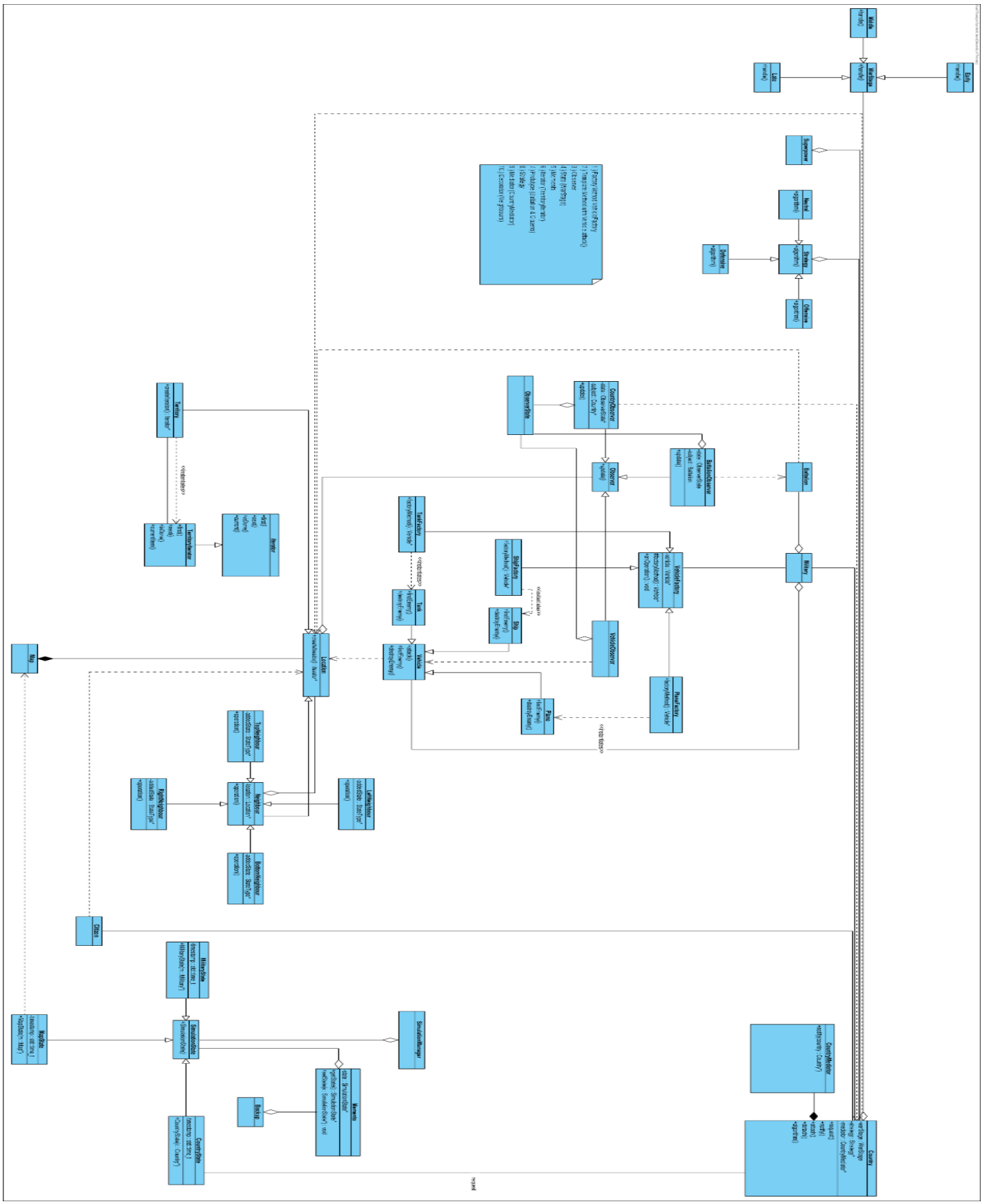
COS-214 PROJECT JJJJM

Contents Page

Task 1: Practical Assignment 5 **2**
Task 2: Design **3**
2.1 Functional requirements **3**
2.2 Activity Diagram **4**
2.3 Choice of design patterns and processes **5**
2.4 Design of classes **6**
2.5 Class Diagram **8**
2.6 Sequence and Communication Diagrams **9**
2.7 State Diagram **11**
2.8 Object Diagram **12**
Task 3: Implementation **14**
3.1 Main program: **14**
Task 4: Report **15**
4.1 Research Brief **15**
Introduction **15**
War Theatres **15**
Transportation **15**
Entities **15**
Phases of War **16**
Changes in Strategies and Tactics Over Time **16**
4.2 Design Report **17**
War Theatres **17**
Entities **17**
Transportation **18**
Phases of War **18**
Changes in Strategies and Tactics Over Time **19**
Miscellaneous Design Patterns **19**
Overview **21**
Task 5: Development Practices **22**
5.1 Github repository **22**
5.3 Doxygen documentation **22**
5.5 Automated tests **22**
Extra **22**
Google Doc 23
References **24**

Task 1: Practical Assignment 5



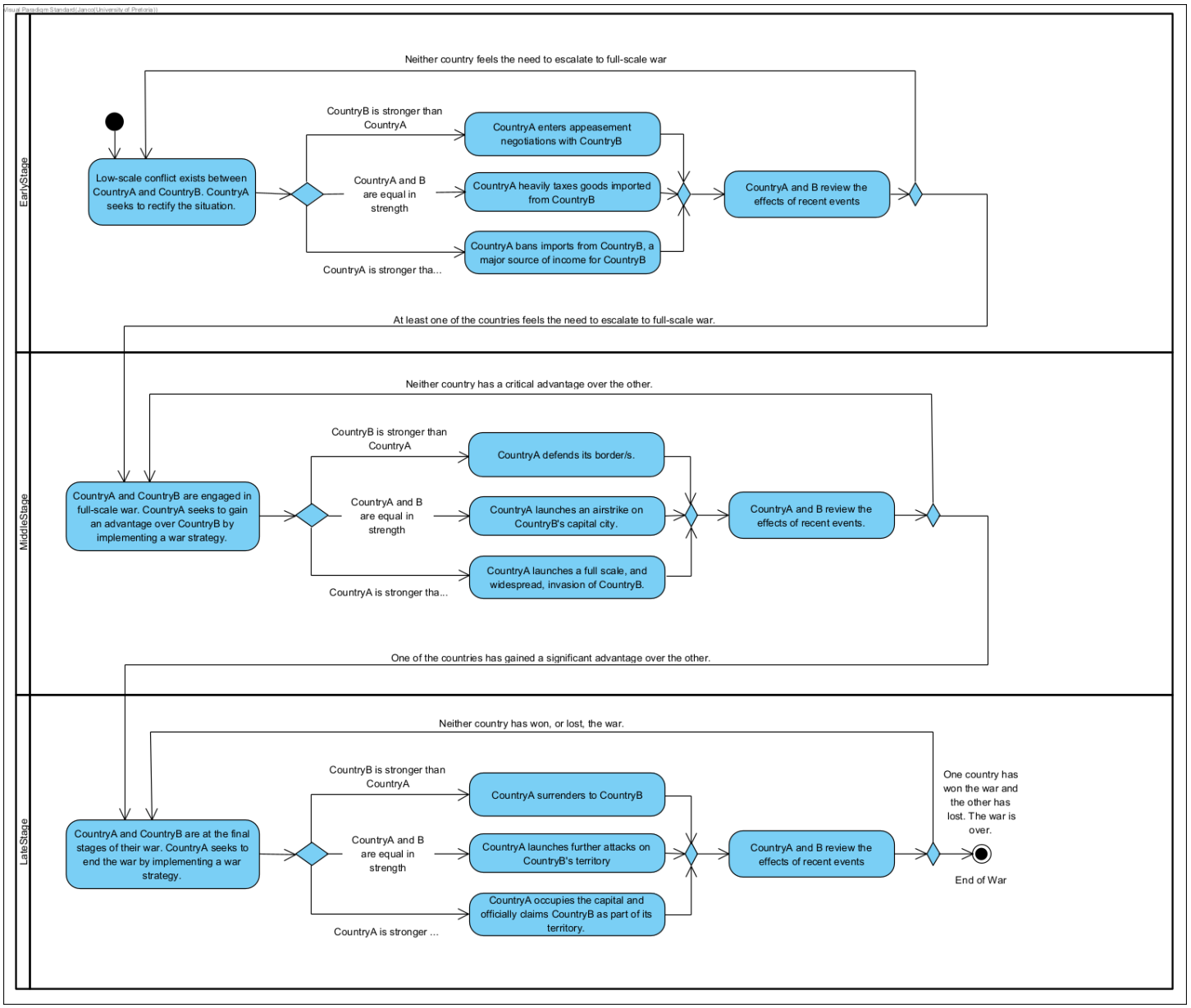
Task 2: Design

2.1 Functional requirements

The functional requirements are as follows:

- The system will prompt the user to start the simulation in design mode or not with console input and output.
- The system will initialise the simulation with default values.
- During the run of the simulation, the system will call the takeTurn() function on each of the current countries.
- After each turn, the system displays a summary of all actions taken and changed states during that turn.
- In design mode, the system will give the user the opportunity to change the current state of the simulation before the next turn starts.
- The simulation continues running until only one superpower remains within the simulation or the user chooses to quit.
- When the simulation is started, a map consisting of a grid of linked territories is created together with all the countries grouped by superpower.
- Each country's state and military is initialised to default values.
- During the simulation each country's actions taken during their turn are dictated by the strategy they employ which may change during the simulation based on their state.
- The war stage changes during the simulation and influences the strategies used by the countries.
- When a country performs an action that influences another country, the state of another country might change.

2.2 Activity Diagram



2.3 Choice of design patterns and processes

We have decided to use the following patterns to address the functional requirements of the system:

- The State pattern is used to model the changing war stages throughout the simulation.
- The Memento pattern is used to capture and store the states of the system so that changes made to the simulation can be reverted.
- The Strategy pattern is used to model the different actions that a country can take during their turn.
- The Decorator pattern is used to decorate territories with pointers to their neighbours to construct a linked grid of territories.
- The Iterator pattern is used to iterate over the linked grid of territories.
- The Observer pattern is used to notify locations of changes to countries.
- The Factory Method is used to construct vehicle objects in a uniform manner.
- The Template Method is used in the Factory Method to alter which vehicle gets constructed depending on the factory object which calls the manufactureVehicle() function.
- The Prototype pattern is used to easily create new battalions from existing objects.
- The Singleton pattern is used to ensure that all countries access the same war stage object.

2.4 Design of classes

SimulationManager: This is the class that manages the simulation as a whole and facilitates the turns of all the countries as well as the map and mementos.
SimulationState: This is the concrete class that will be stored in mementos and is sufficient to restore the state of the entire simulation.
Memento: This class stores a SimulationState object and can set and restore the state of the simulation.
Backup: This is the caretaker of the Memento and stores Memento objects.

StageContext: This class acts as the context in the State design pattern and functions as a Singleton.
StageContextState: The class to store the state of the current stage of the war.
WarStage: This class is the abstract state participant which models the current stage of the war.
EarlyStage: The first concrete state WarStage.
MiddleStage: The second concrete state WarStage.
LateStage: The last concrete state WarStage.

Location: This is the abstract component class of the decorator pattern and acts as a space on the grid of spaces.
Neighbour: This is the abstract decorator class that stores a pointer to another location that will act as its neighbour. It inherits from Location.
LeftNeighbour: A concrete decorator storing a pointer to a location to the left of the current location.
RightNeighbour: A concrete decorator storing a pointer to a location to the right of the current location.
TopNeighbour: A concrete decorator storing a pointer to a location above the current location.
BottomNeighbour: A concrete decorator storing a pointer to a location below the current location.
Neighbour: An abstract class providing the interface for all other neighbours.
Territory: A concrete location to be decorated with neighbours.
Iterator: The abstract iterator interface.
LocationIterator: The concrete iterator used to traverse the linked grid of Locations.
Map: The class that will contain the entrypoint to the grid of locations and manage operations performed thereon.
MapState: The class that will be used to store the state of the map in the SimulationState.
LocationObserver: The class that updates a location if a country gets destroyed by another country.

Superpower: This is the class that models the alliances between different countries by storing them in different groups.
SuperpowerState: The class to hold the state of the superpower at any given point.
Country: This is the main class modelling the countries fighting in the war which owns a military.
CountryState: This class encapsulates the state of the country and is stored in the SimulationState to restore the state of the system.

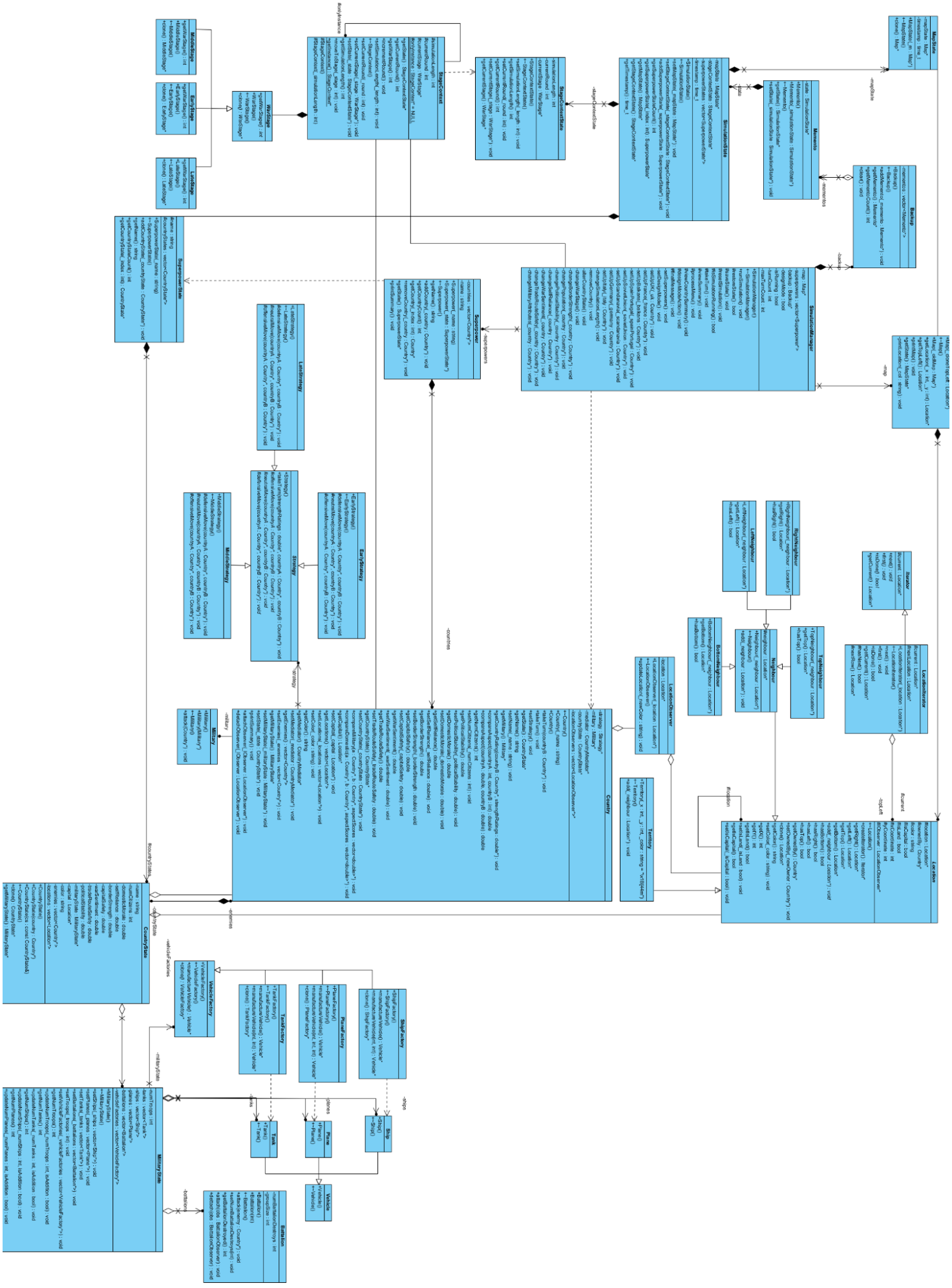
Strategy: The abstract strategy class providing an interface for the countries to take their turns.
MiddleStrategy: The concrete strategy class for countries acting neutrally.
EarlyStrategy: The concrete strategy class for countries acting defensively.
LateStrategy: The concrete strategy class for countries acting offensively.

Military: This is the class that manages the assets of a military used to fight in the war.
MilitaryState: The state of a military that will be stored in the SimulationState to restore the state of the system.
Battalion: The class representing a group of soldiers fighting in the military.

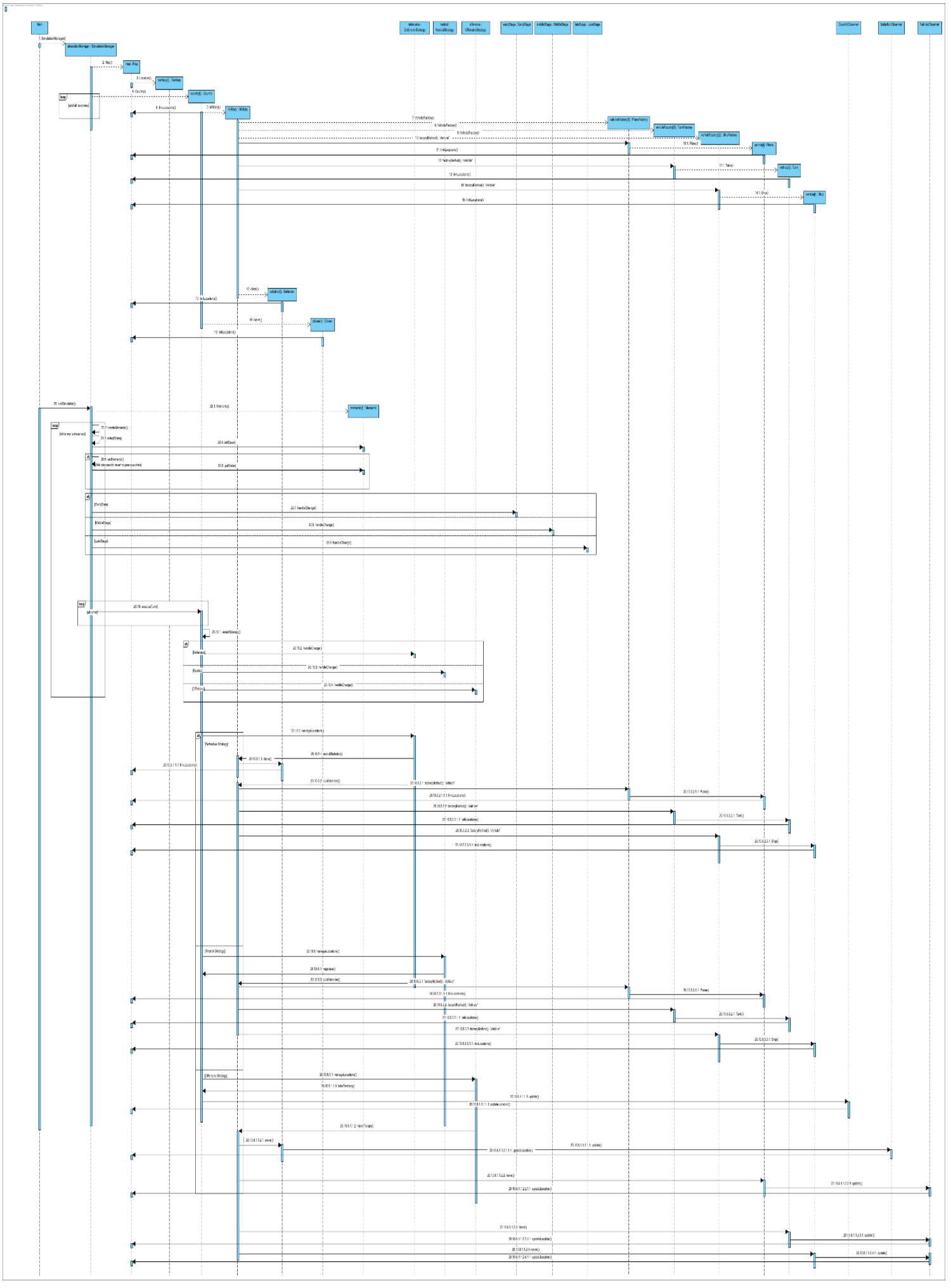
Vehicle: This abstract class models vehicles owned by the military to be used to fight in the war.
VehicleFactory: This abstract class provides the interface for creating vehicles.

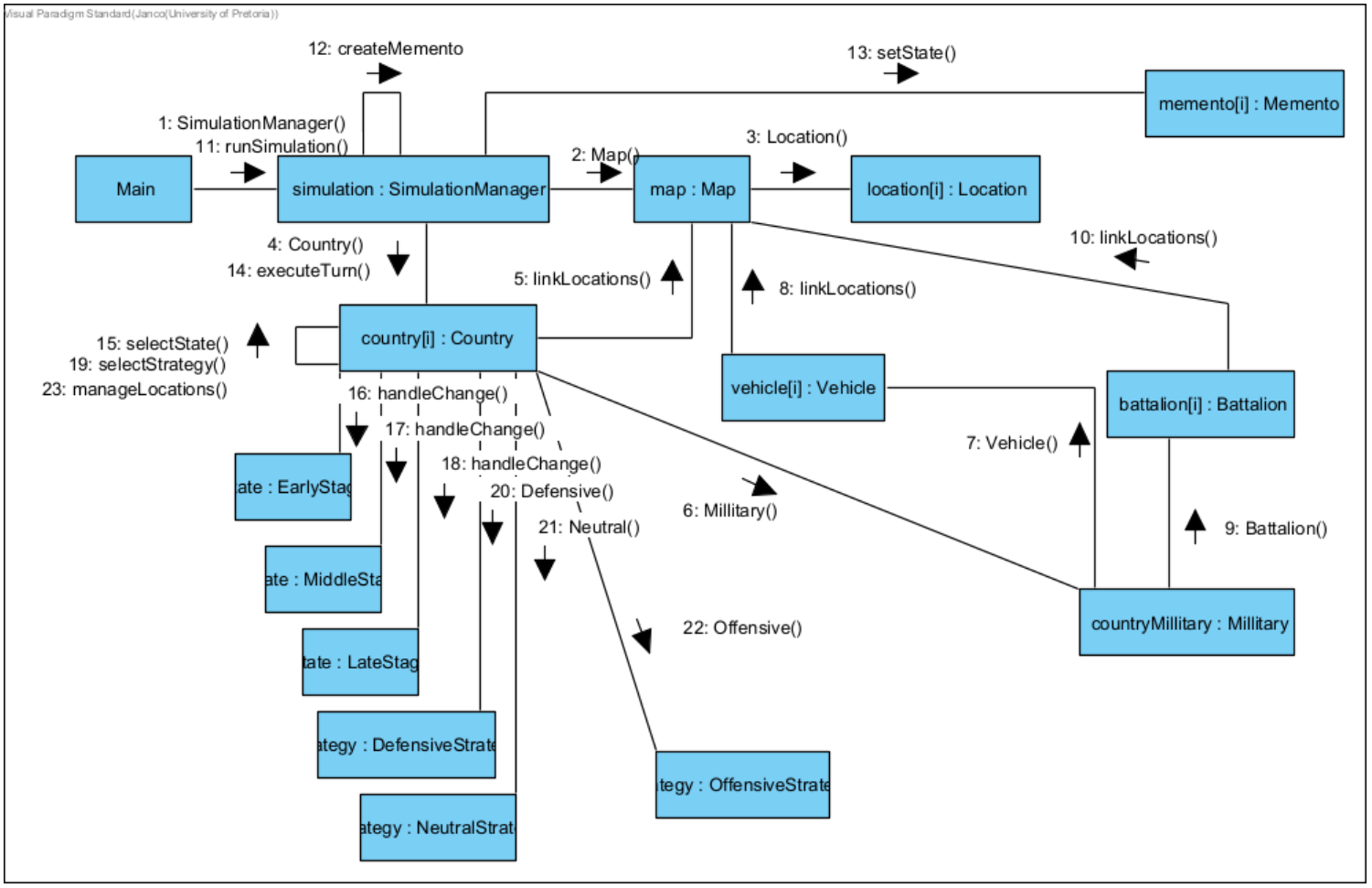
Tank: A concrete land vehicle.
TankFactory: The concrete factory used to create tank objects.
Plane: A concrete air vehicle.
PlaneFactory: The concrete factory used to create plane objects.
Ship: A concrete naval vehicle.
ShipFactory: The concrete factory used to create ship objects.

2.5 Class Diagram

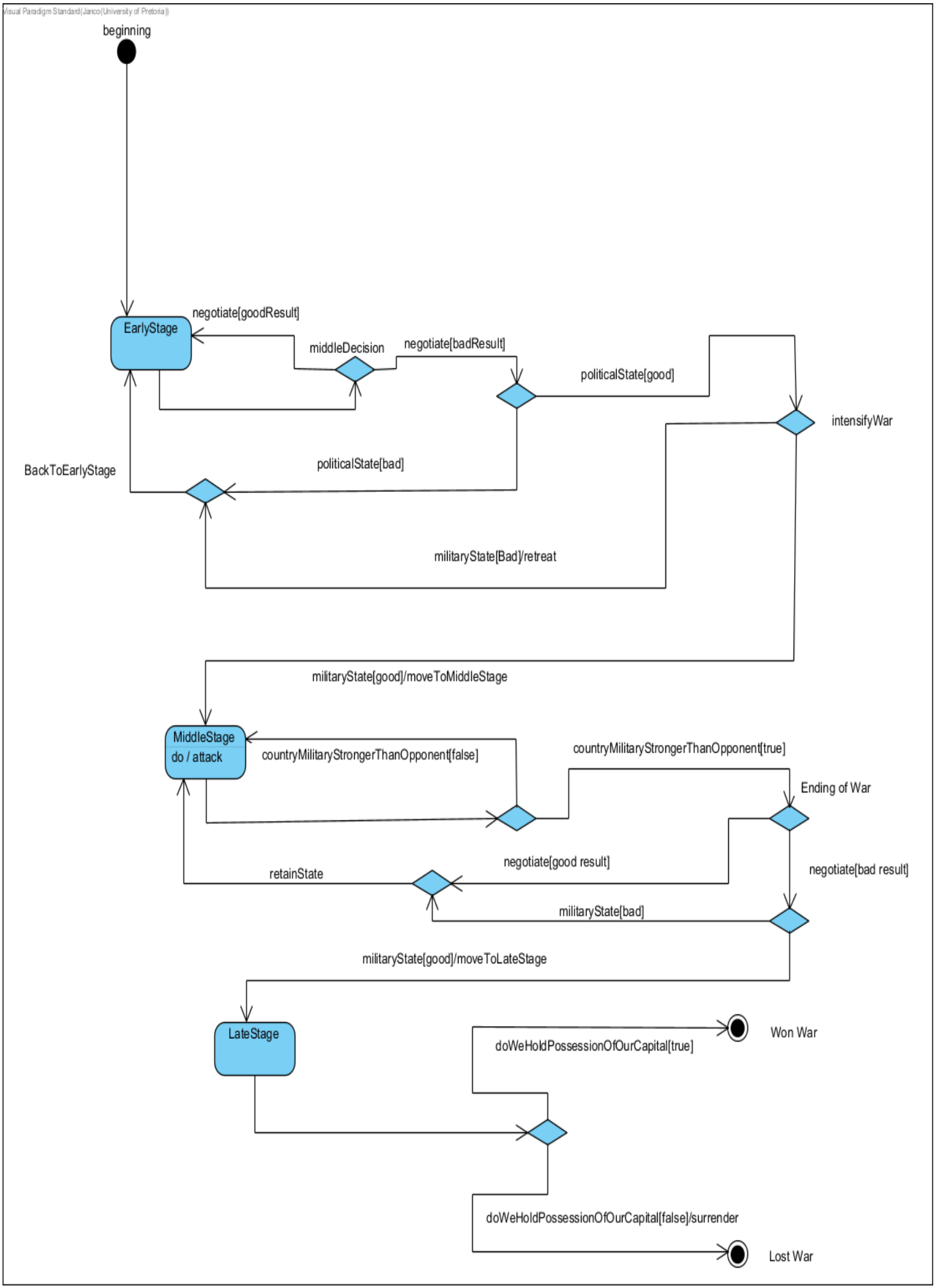


2.6 Sequence and Communication Diagrams

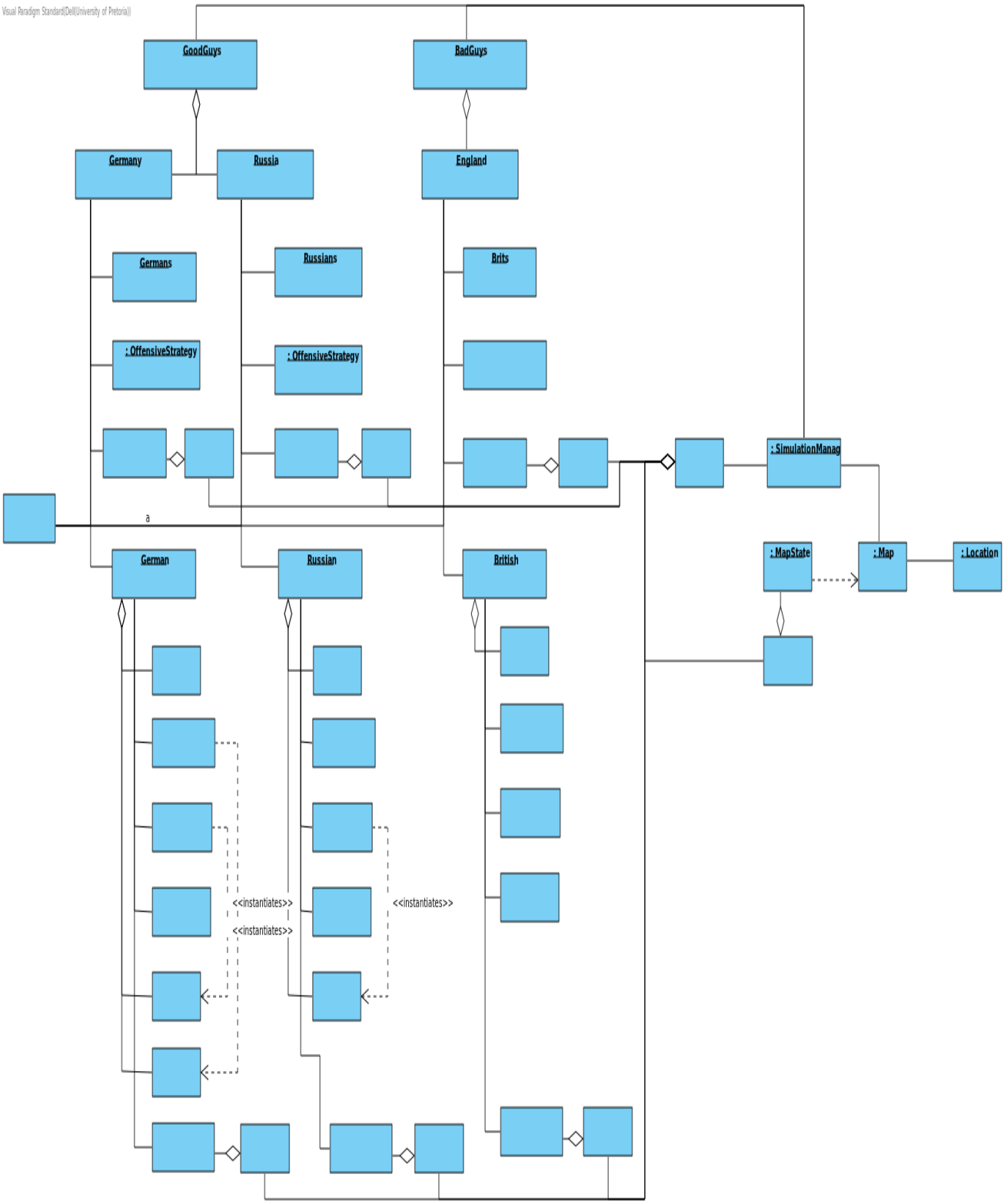



Visual Paradigm Standard(Janco(University of Pretoria))
12: createMemento
13: setState()
memento[i] : Memento
1: SimulationManager()
11: runSimulation()
Main
simulation : SimulationManager
2: Map()
map : Map
3: Location()
location[i] : Location
4: Country()
14: executeTurn()
5: linkLocations()
8: linkLocations()
10: linkLocations()
15: selectState()
19: selectStrategy()
23: manageLocations()
country[i] : Country
vehicle[i] : Vehicle
battalion[i] : Battalion
16: handleChange()
17: handleChange()
18: handleChange()
20: Defensive()
21: Neutral()
6: Millitary()
7: Vehicle()
9: Battalion()
ate : EarlyStag
ate : MiddleSta
tate : LateStag
tegy : DefensiveStrate
tegy : NeutralStrat
22: Offensive()
tegy : OffensiveStrate
countryMillitary : Millitary

2.7 State Diagram



2.8 Object Diagram


Visual Paradigm Standard(Dell(University of Pretoria))
GoodGuys
BadGuys
Germany
Russia
England
Germans
Russians
Brits
: OffensiveStrategy
: OffensiveStrategy
: SimulationManag
a
German
Russian
British
: MapState
: Map
: Location
<<instantiates>>
<<instantiates>>
<<instantiates>>

Task 3: Implementation

3.1 Main program:

https://github.com/DeadpanSplash82/COS-214-Project-JJJJM/tree/main/src

Task 4: Report

4.1 Research Brief

Introduction

For this assignment we chose to base our simulation on the largest scale war in human history, World War II. This was a war fought from 1939 to 1945 between the Allies (Great Britain, the United States and the Soviet Union), and the Axis (Germany, Italy and Japan) (United States Holocaust Memorial Museum, Washington, DC, 2022).

War Theatres

A war theatre is defined as the geographic location in which a war takes place. During World War II, there were two primary theatres, the Pacific Theatre, which encompassed large parts of the Pacific Ocean, and the European Theatre, which spanned from Spain to Western Russia (National Geographic Society et al., 2022). The European Theatre will be the subject of our simulation.

Transportation

During the war, troops were transported through a variety of methods. From Great Britain, they were sent on ships to reach mainland Europe. When on land, troops were then sent in vehicles to reach the front lines. Once there, troops could engage in conflict in many ways, including tanks, planes and on foot. It is also worth noting that troops were often airdropped into the front lines (Pacific Rent-A-Car, n.d.).

Entities

The main entities of World War II that we chose to represent are soldiers, tanks, ships, planes, countries, superpowers, locations of battle and capitals. Many more entities existed; however, this would overcomplicate the simulation. It is obviously impossible to perfectly match the real world with a simulation; thus, it is a good idea to limit the amount of entities we model to ensure accuracy for those entities that we do model.

Phases of War

World War II is said to have five main phases:

- The first phase was the Phoney War (September 1939–April 1940), where the war was declared after Germany invaded Poland. This phase was characterised by the inaction of the countries at war. Each country was biding their time waiting for the other to make the first move (BBC, n.d.).
- The second phase was Blitzkrieg (April 1940–June 1940), which saw Germany take vast amounts of land from Norway, Holland, Belgium and France. During this phase, the war escalated greatly with some countries still remaining passive while others were very offensive (BBC, n.d.).
- The third phase, aptly named, Britain and the empire stands alone (July 1940–June 1941), was the point at which Germany's victory seemed most likely. Great Britain was fighting with few allies, thus, very much on the back foot during this phase and forced to take up the defensive (BBC, n.d.).
- The fourth phase, known as The tide turns (1941–1943), saw more countries get involved in the war due to Germany invading The Soviet Union as well as Japan bombing Pearl Harbor. The increase in countries fighting against Germany caused them to begin losing lands on many fronts and can be seen as the turning point of the war (BBC, n.d.).
- The fifth and final phase, simply named Victory (1943–1945), involved the final push by the Allies to take back Europe. The Allies pushed back into Europe on D-Day and eventually overtook Berlin, thus ending fighting in Europe. Later, the atomic bombs were dropped on Hiroshima and Nagasaki, ending the war as a whole (BBC, n.d.).

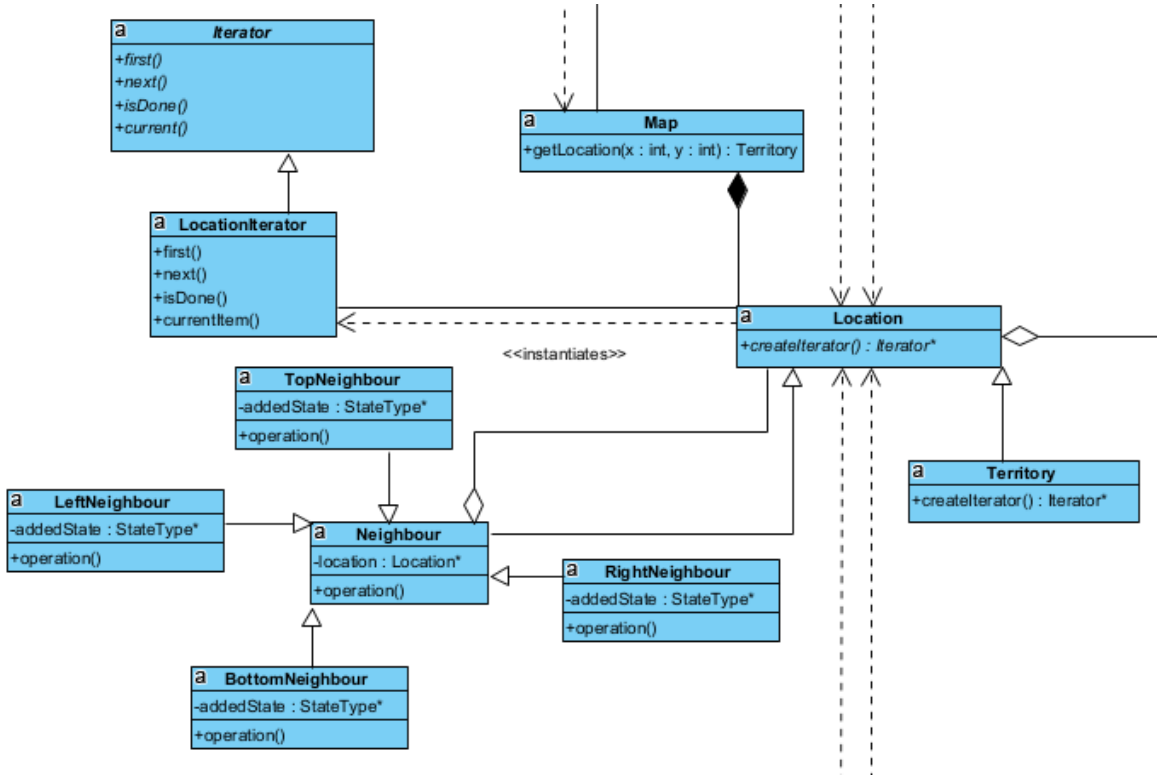
Changes in Strategies and Tactics Over Time

As the war progressed and each country's state changed, different strategies were implemented. Early on, most countries were very passive, still hoping that peace could be achieved without much bloodshed; however, as the Blitzkrieg phase occurred, countries flipped to a more defensive strategy where they only hoped to hold back the seemingly unstoppable German army. Finally, as more countries got involved against Germany, they could adopt an offensive strategy to take back the lands in Europe. There were many other factors which affected the strategy used as well. This included political stability, domestic morale, border strength, capital safety, war sentiment and trade route safety.

4.2 Design Report

War Theatres

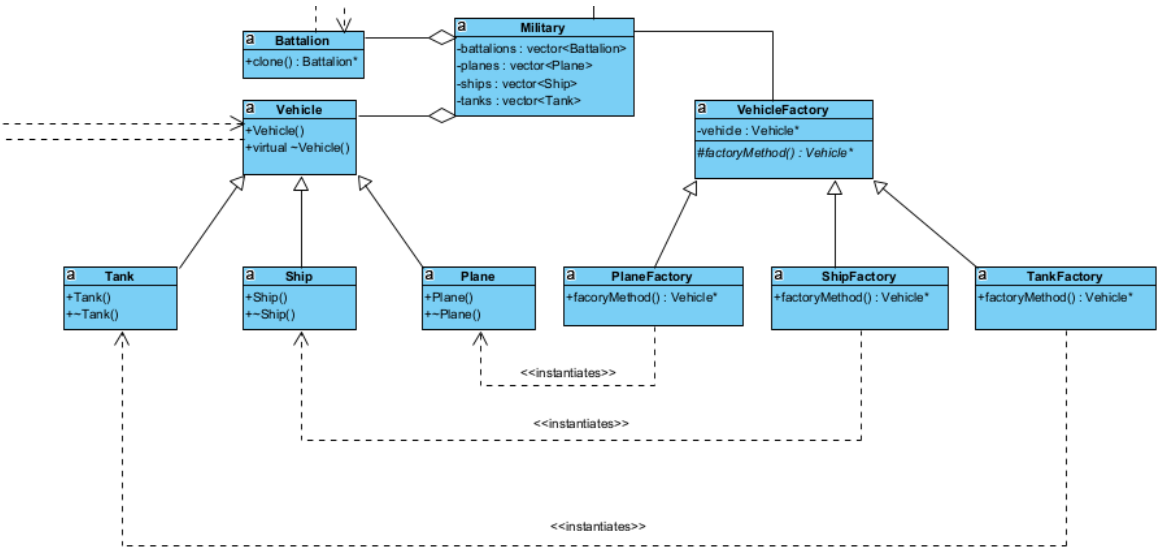
War theatres were seen as a highly essential component in our project design. To implement this component, we had to design a data structure that would allow various entities to traverse the map. The map adds the functionality that every entity has a location or set of locations. The map is implemented as a square graph where each node points to its adjacent node (i.e., right, left, top or bottom). An issue that we faced was that nodes in corners and edges would not have connections to certain adjacent nodes. This was solved using the Decorator Design Pattern. By adding the right, left, top or bottom attributes dynamically, we were able to create unique location nodes for each node of the map. We then used the Iterator Design Pattern to sequentially visit each node in the map. It works by moving as far right as it can until it reaches the edge, then it moves down and to the far left of the next row and continues moving right, in much the same way as we read text. The iterator is used to print out the map to console and also to delete each node of the map.



Abridged class diagram showing the Decorator as well as Iterator Design Pattern.

Entities

To create such a large amount of entities, we used various creational design patterns . We used The Factory Method Design Pattern for creating each vehicle. To do this, we created the PlaneFactory, the TankFactory and the ShipFactory classes. This simplifies the creation process of these multiple vehicles. We also used the Template Method Design Pattern in the Factory Method Design Pattern, since the Factory Method uses the Template Method. We then decided on an object to represent troops. It would be far too taxing on the system to create a new troop object for all 15 million deployed troops; thus, we grouped many troops together in the Battalion class. Since the construction of each battalion would be identical, we chose to use the Prototype Design Pattern. These entities get encapsulated by the Military class, which will use the VehicleFactory and Battalion classes to create new entities for a country to use in battle.



Abridged class diagram showing the Factory Method as well as the Prototype Design Pattern.

Countries are probably the most vital entity in the simulation. They make use of all the other entities and design patterns to carry out the war. These include the locations that a country owns and its military. Countries are also grouped into superpowers, i.e., a group of countries that band together to fight the opposing superpower. In our simulation, there will be two superpowers, the Axis and the Allies.

Transportation

Transportation is simply modelled using the map class and the various entities listed above. For example, in order to transport troops from a country's capital to the front lines they will have to traverse the map, location by location. The same goes for all components of the military.

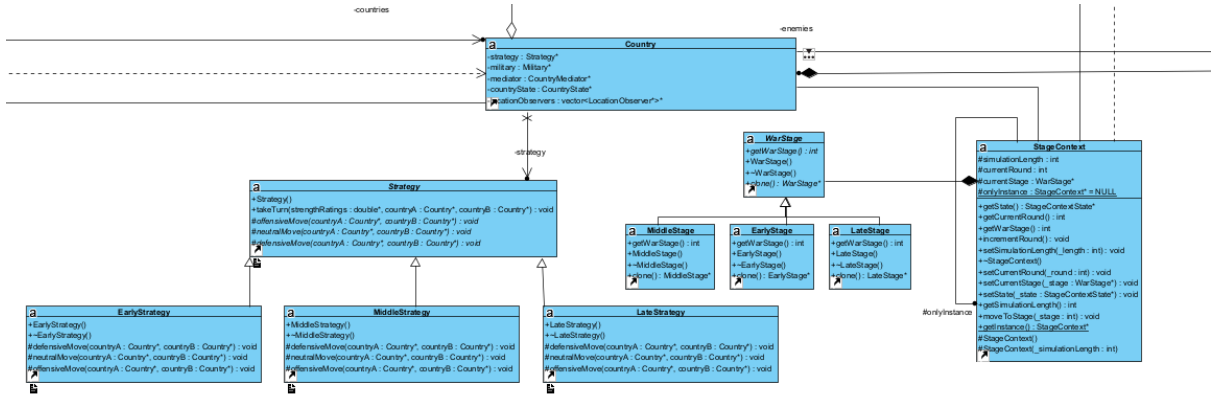
Phases of War

We decided to simplify the five phases of World War II into three more general phases, named early, middle and late. The early phase involves countries being generally less aggressive and opting to mitigate the fighting that takes place. The middle phase is where the majority of the conflict occurs and will cause countries to be far more aggressive, ramping up their militaristic development. The late phase involves either victory or defeat of the countries involved. Here, checks will be made to see which country won the war. The condition for losing is if a country lost its capital city during the war. This was modelled using the State Design Pattern. This allows us to alter a country’s behaviour depending on which phase of war is currently occuring. It’s worth noting that the Singleton Design Pattern was used to ensure that each country only holds a single state object at a time.

See 2.2, Activity Diagram, and 2.7, State Diagram, for a UML representation of the above description.

Changes in Strategies and Tactics Over Time

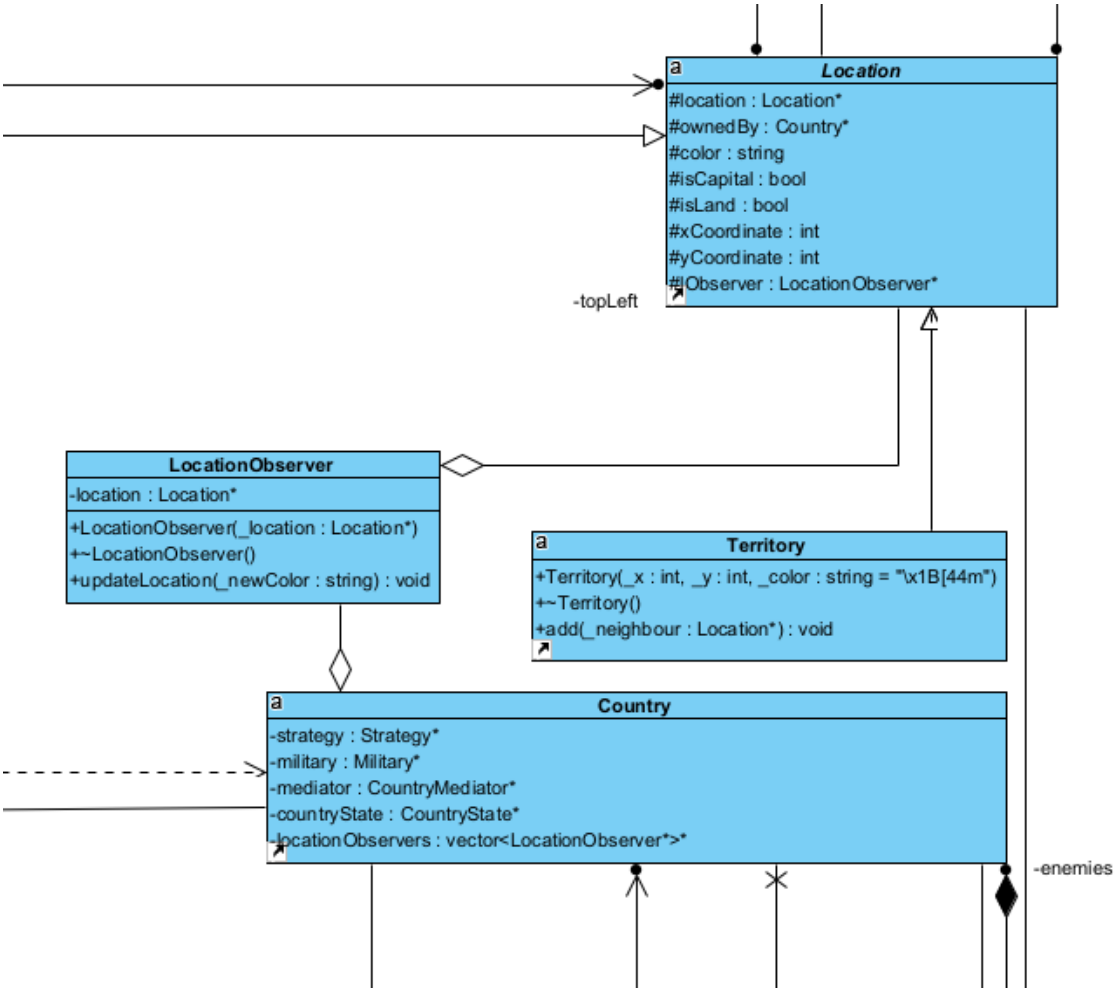
The change in state mentioned in the previous section will inform a country on what strategy to use. We chose three main strategies. An early strategy, which can be seen as defensive, a middle strategy, which can be seen as balanced, and a late strategy, which can be seen as offensive. This was done using the Strategy Design Pattern. Using different algorithms implemented in each of the early, middle and late strategy classes, we can alter a country's behaviour as the war and the countries change.



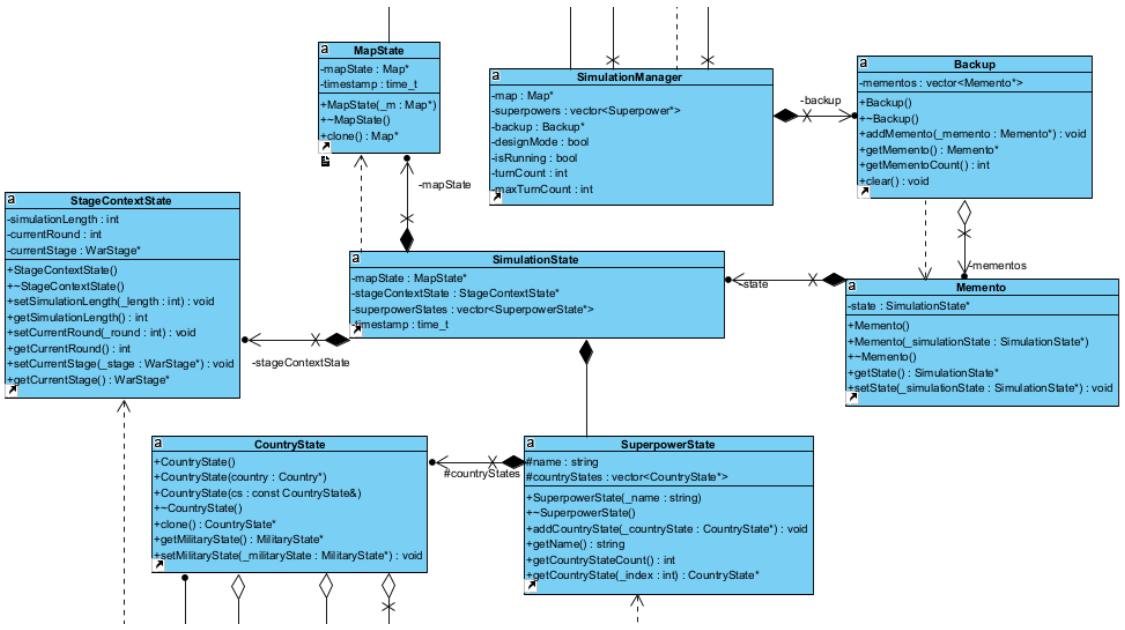
Abridged class diagram showing the State as well as the Strategy Design Pattern.

Miscellaneous Design Patterns

To give further functionality to the simulation, we added the Memento Design Pattern. This will give the user the ability to return to previous turns of the simulation. The overall state of the simulation is fairly large; thus, to encapsulate the entire state, we created the SimulationState class. This holds states to all other classes and is the way in which the memento stores and reverts to previous states. Furthermore the Observer Design Pattern is used to notify a location to change its state whenever the country which holds the location is destroyed.



Abridged class diagram showing the Observer Design Pattern.



Abridged class diagram showing the Memento Design Pattern as well as how state is stored.

Overview

The general idea of our simulation is that a class called the SimulationManager will control the running of the entire simulation. In most of the design patterns, the SimulationManager will function as the client. It will instantiate the starting map and all of the countries with their starting states, including their military and their owned locations. It also controls when the simulation ends with a turn counter and/or various win conditions for each superpower. The interface with which the user will interact is also controlled by the simulation manager using console input and output. At the start of each turn, it will create a new memento allowing the user to revert to previous simulation states. During each turn, a country will be given the opportunity to perform various actions, such as attacking another country, producing more militaristic resources and causing those resources to traverse the map. As the turns go on and the state changes, the war will progress and eventually a winner will be decided, at which point the simulation will be complete.

Task 5: Development Practices

5.1 Github repository:

https://github.com/DeadpanSplash82/COS-214-Project-JJJJM

5.3 Doxygen documentation :

https://github.com/DeadpanSplash82/COS-214-Project-JJJJM/blob/main/Documentation/Documentation.pdf

5.5 Automated tests:

https://github.com/DeadpanSplash82/COS-214-Project-JJJJM/tree/main/test

Extra

Google Doc:

JJJJM PROJECT

References

BBC. (n.d.). *The five phases - World War Two - 3rd level History Revision*. BBC. Retrieved October 31, 2022, from https://www.bbc.co.uk/bitesize/guides/z3vq96f/revision/3

National Geographic Society, MacPhee, M., & Modafferi, M. (2022, May 20). *World War II in Europe*. National Geographic Society. Retrieved October 21, 2022, from https://education.nationalgeographic.org/resource/world-war-ii-europe

Pacific Rent-A-Car. (n.d.). *Car, Land and Air Vehicles During the Second World War*. Pacific Rent-A-Car. Retrieved October 30, 2022, from https://www.pacific-rentals.com/useful-information/car-war.asp

United States Holocaust Memorial Museum, Washington, DC. (2022, April 22). *Axis Alliance in World War II* | *Holocaust Encyclopedia*. Holocaust Encyclopedia. Retrieved October 21, 2022, from https://encyclopedia.ushmm.org/content/en/article/axis-powers-in-world-war-ii