



**UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA**

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS214 Software Modelling War Simulation Project

by

**Oliver Welsh 21432962
Kaitlyn Sookdhew 21483974
Robert Officer 20431122
Wian Koekemoer 19043512
Megan Hugo 20538422
Latasha Friend 21526053**

**Submitted in partial fulfilment of the requirements for the degree BSc
Computer Science
In the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria**

Lecturer: Dr Linda Marshall

November 2022

Table of contents

Table of contents	2
Task 2: Design	3
2.1 Functional Requirements	3
War Theatres	3
Transportation	3
Entities	3
Phases of War	3
Changes to War Engine	3
Launch Re-enactment	4
2.2 Activity Diagrams	4
2.3 Design Patterns	8
1. Abstract Factory	8
2. Chain of responsibility	8
3. Command	8
4. Composite	8
5. Decorator	8
6. Factory Method	8
7. Memento	8
8. Prototype	8
9. Template Method	8
10. State	8
11. Strategy	8
2.4 Class Structure	9
2.5 Class Diagram	10
2.6 Sequence and Communication Diagrams	11
2.7 State Diagrams	15
2.8 Object Diagrams	16
Task 3: Implementation	18
Task 4: Report	19
4.1 Research Brief	19
4.2 Application of Design Patterns	20
Task 5: Development Practices	30
5.1 Use of GitHub	30
5.2 C++ Documentation	31
5.3 Doxygen	31
5.4 Code Development	31
5.5 Unit Testing	31
Composite Test	31
Memento Test	31
Participants Test	31
Task 6: Demo and Presentation	32
References	33
Plagiarism Declaration	34

Task 2: Design

2.1 Functional Requirements

War Theatres

1. Generate an X by Y (X may equal Y) map which will be used as the battlefield for the war simulator.
2. Have multiple types of cells within the map to differ terrain types.
3. Allocate resources randomly to cells within the map.
4. Assign cells on opposite ends of the map to each opposing force whereby each side can contain multiple types of allies.
5. Display the X by Y map in the terminal.

Transportation

1. Have the ability for the military to move from one cell to another.
2. Simulate fuel costs when the military moves.
3. Simulate terrain traversal challenges through the increased use of fuel over specific cells.
4. Allow for the military to be created in order to protect captured cells when the captured cell's military traverses forward.

Entities

1. The map must contain 2 groups which will go to war against one another.
2. Each group will contain multiple countries.
3. Each country will have its own set of troops
4. Each troop will have their own stats based on its troop type and country of origin.
5. There will be 2 overall troop types, Infantry and Tanks.
6. Troops will be able to perform their own unique attacks based on their type.
7. Troops will be able to form complex squads with multiple different troops.
8. Squads will be able to retreat.
9. Squads must be able to switch between defensive and aggressive.
10. Squads must be able to perform a frontline attack or mining attack.

Phases of War

1. The simulation must have multiple states. The start, traversal, battle and end.
 - a. At the start of the battle, militaries will be allocated to cells giving each country its initial starting land.
 - b. For traversal of the battle, each side will alternate moves advancing one cell at a time. This will result in the collection of resources, traversal costs and the capturing of cells.
 - c. Once squads have captured land and traversed multiple cells forward they will eventually clash with enemy units beginning battle on that row. Each row's units will continuously battle one another moving the border of the land which they own.
 - d. The war will end either when both sides have run out of resources or one group of countries does not occupy any cells.

Changes to War Engine

1. Countries will improve the strength of their units through each loop iteration and the amount will be influenced by how many cells each country owns.

Launch Re-enactment

1. We need to design a terminal graphical user interface which users can use to start the simulation, switch between design mode and real mode and view the simulation as well as the final statistics of the battle.
2. Design mode should allow the user to perform the following.
 - a. Save the current battle
 - b. Pause the current battle
 - c. Edit the current battle
 - d. Continue the current battle
3. Real mode will perform all sequences of events automatically displaying the simulation to the user.
4. Once the battle is over the final statistics, such as the winner of the battle and what percentage of land is owned by each country, will be displayed.

2.2 Activity Diagrams

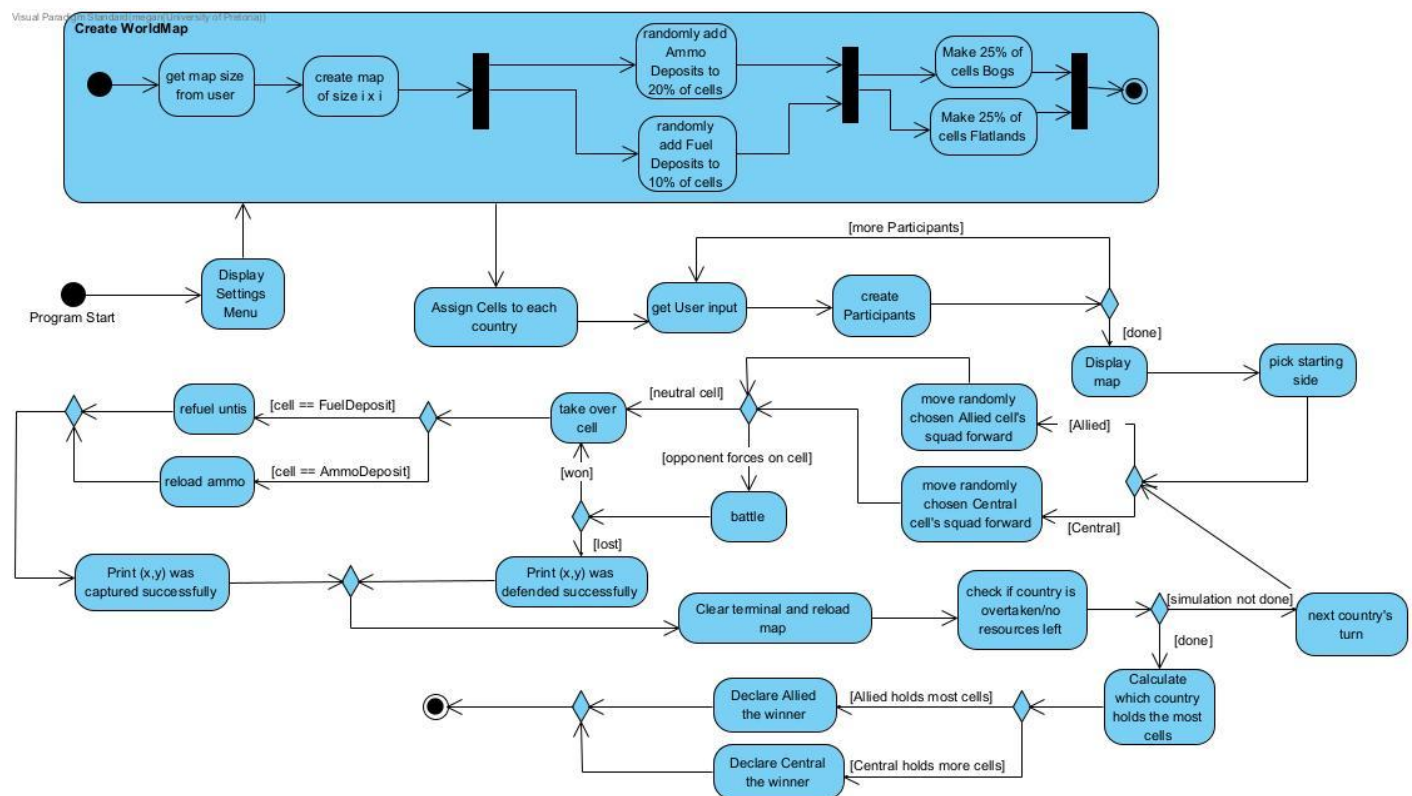


Fig 2.2.1 Overall Activity Diagram (Real Mode)

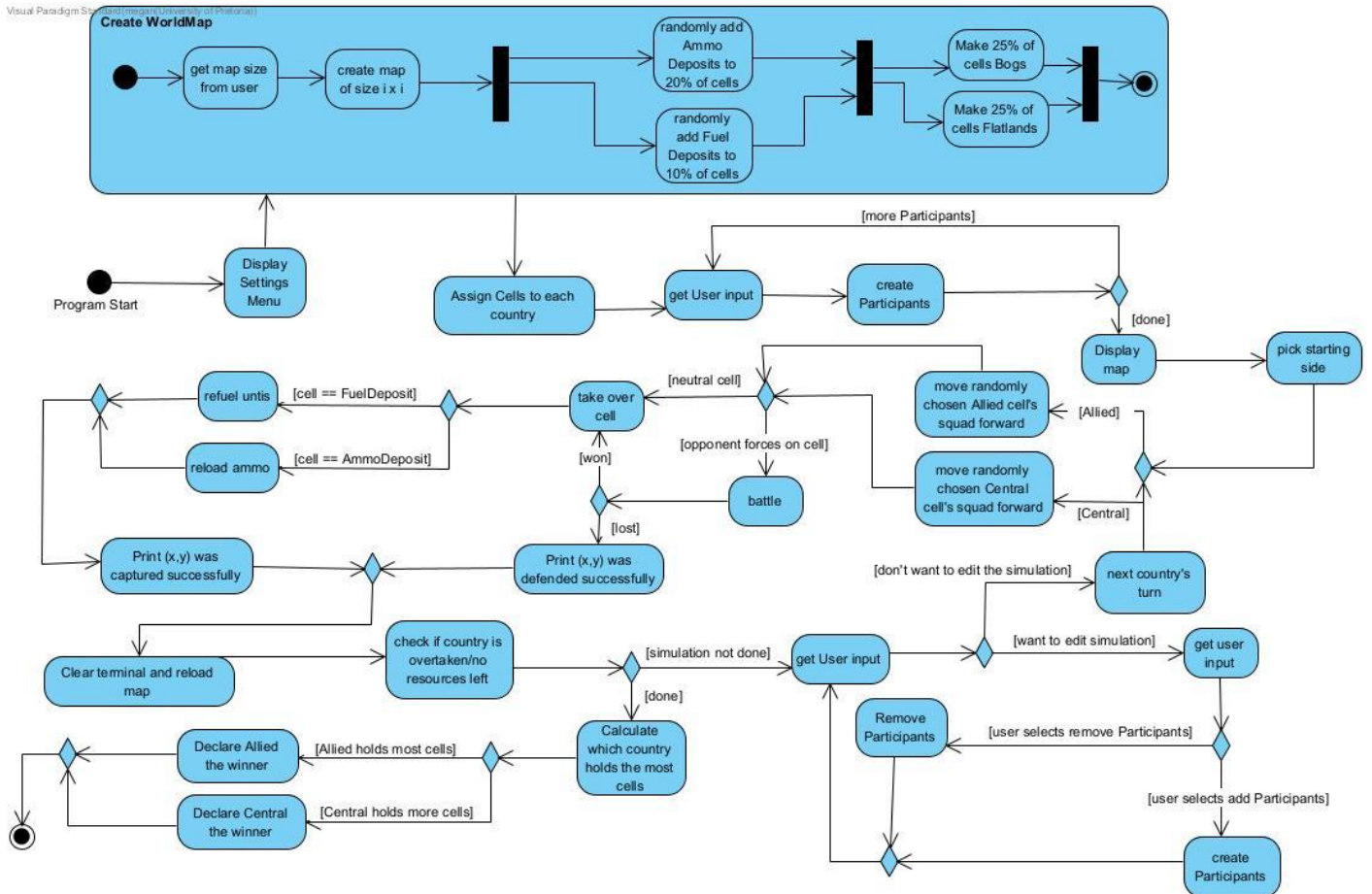


Fig 2.2.2 Overall Activity Diagram (Design Mode)

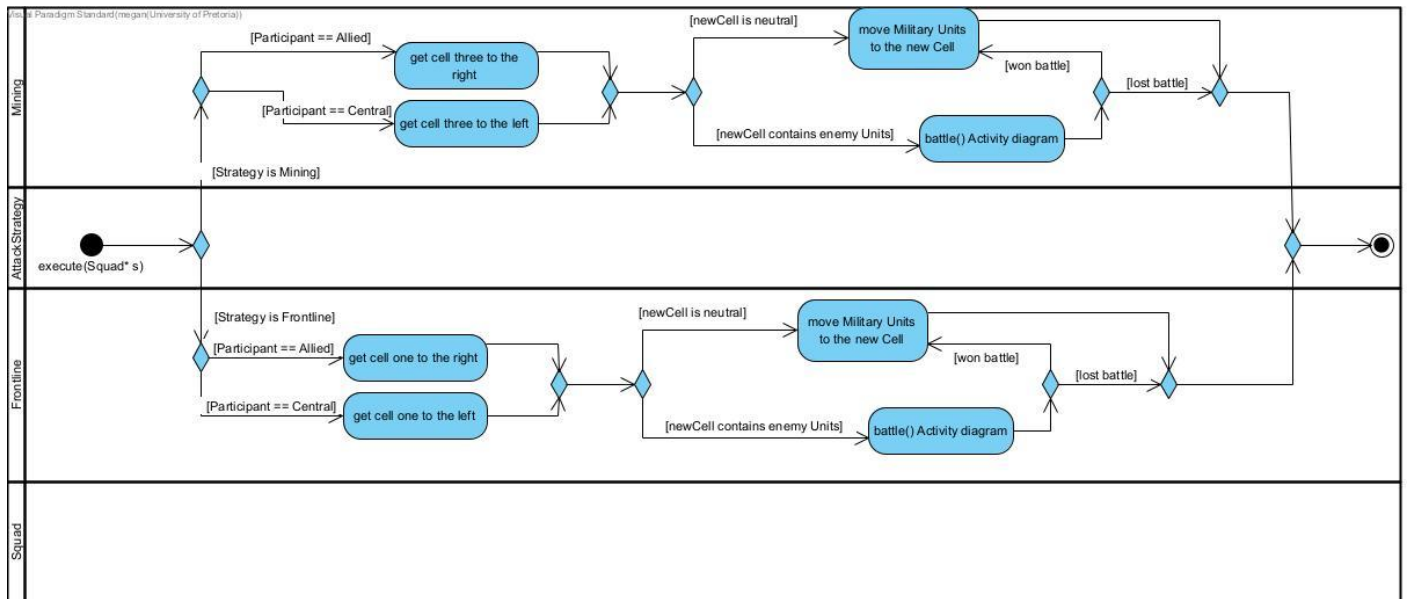


Fig 2.2.3 Strategy Activity Diagram

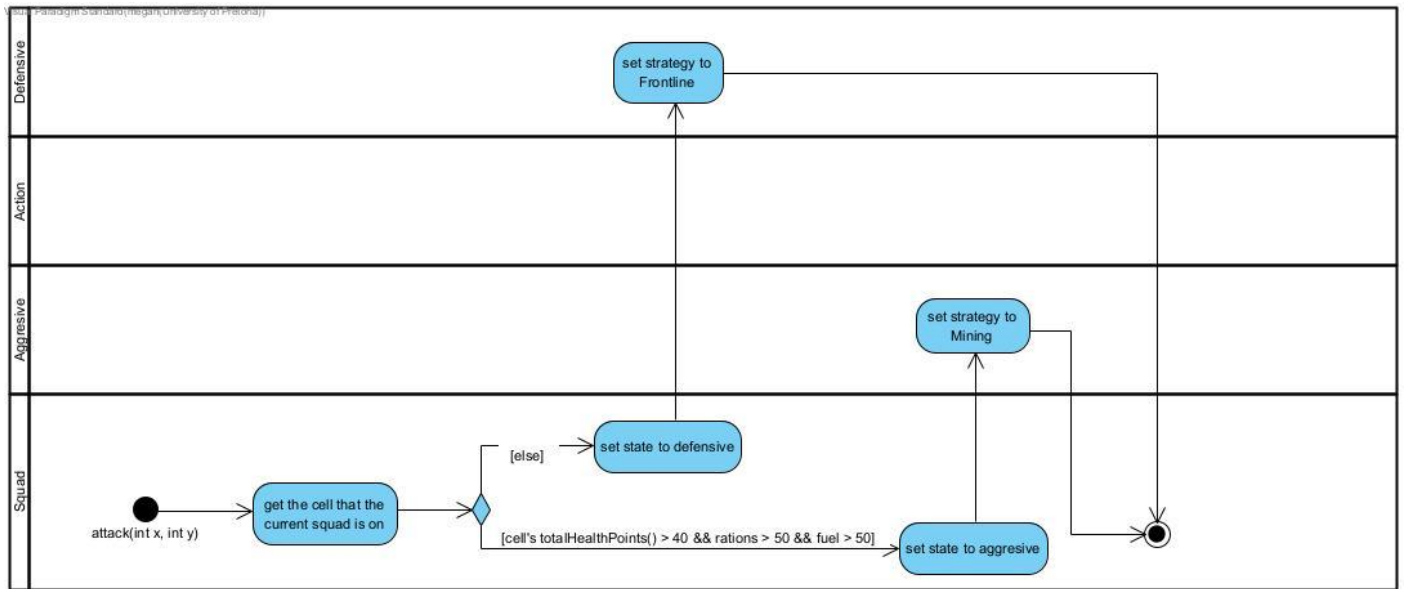


Fig 2.2.4 State Activity Diagram

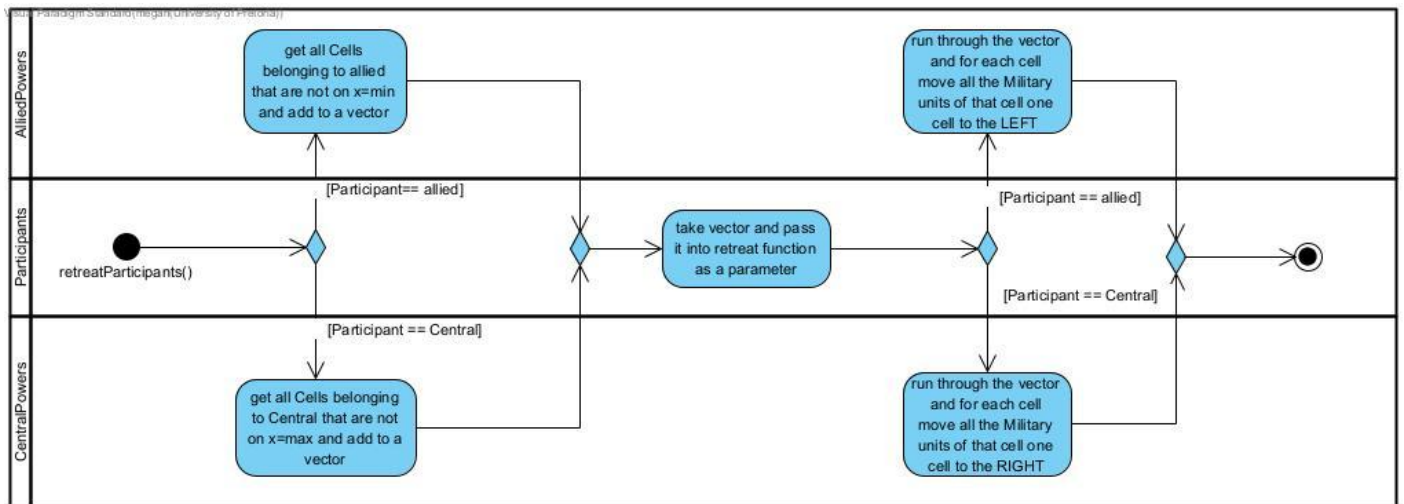


Fig 2.2.5 Template Method Activity Diagram

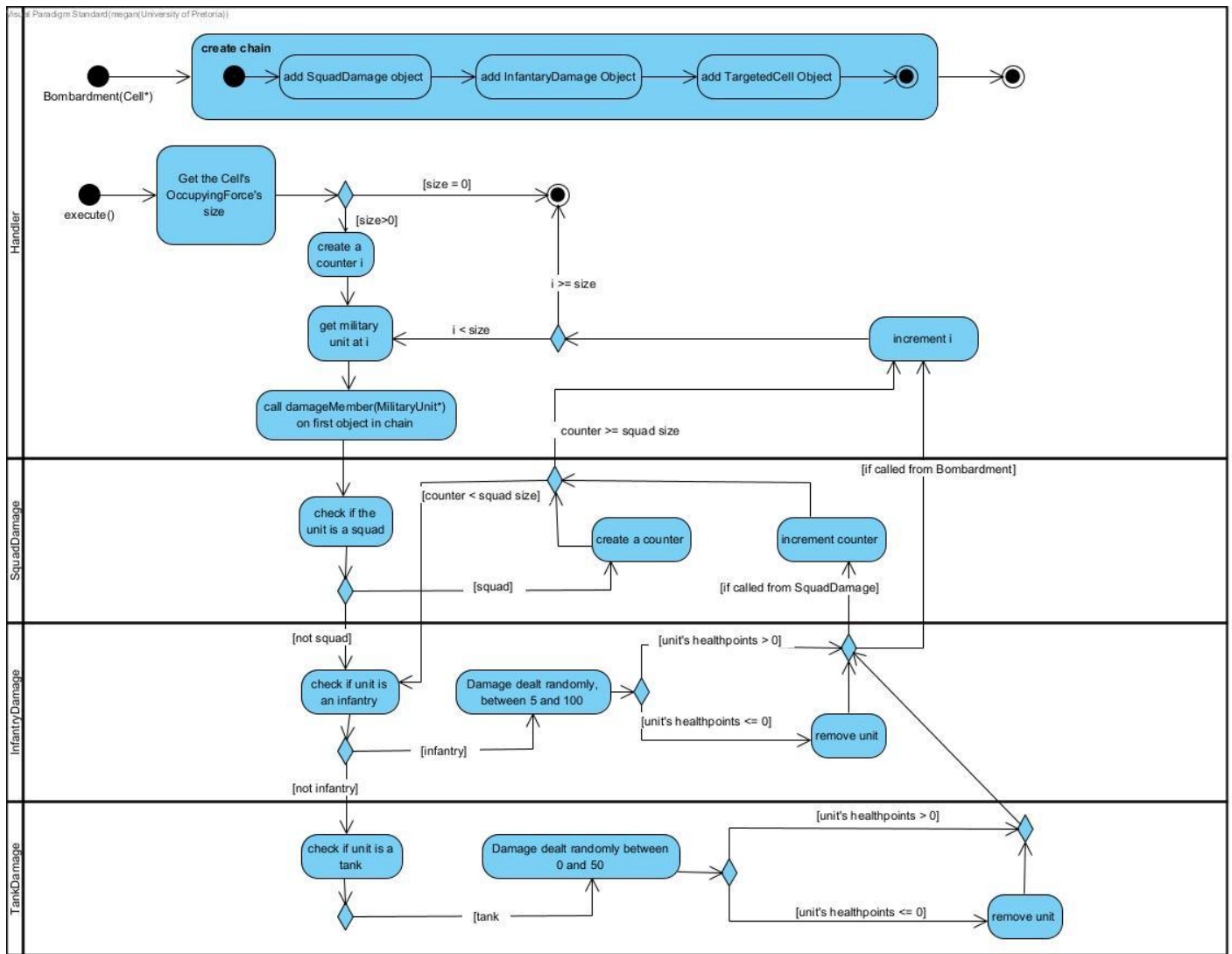


Fig 2.2.6 Chain of Responsibility Activity Diagram

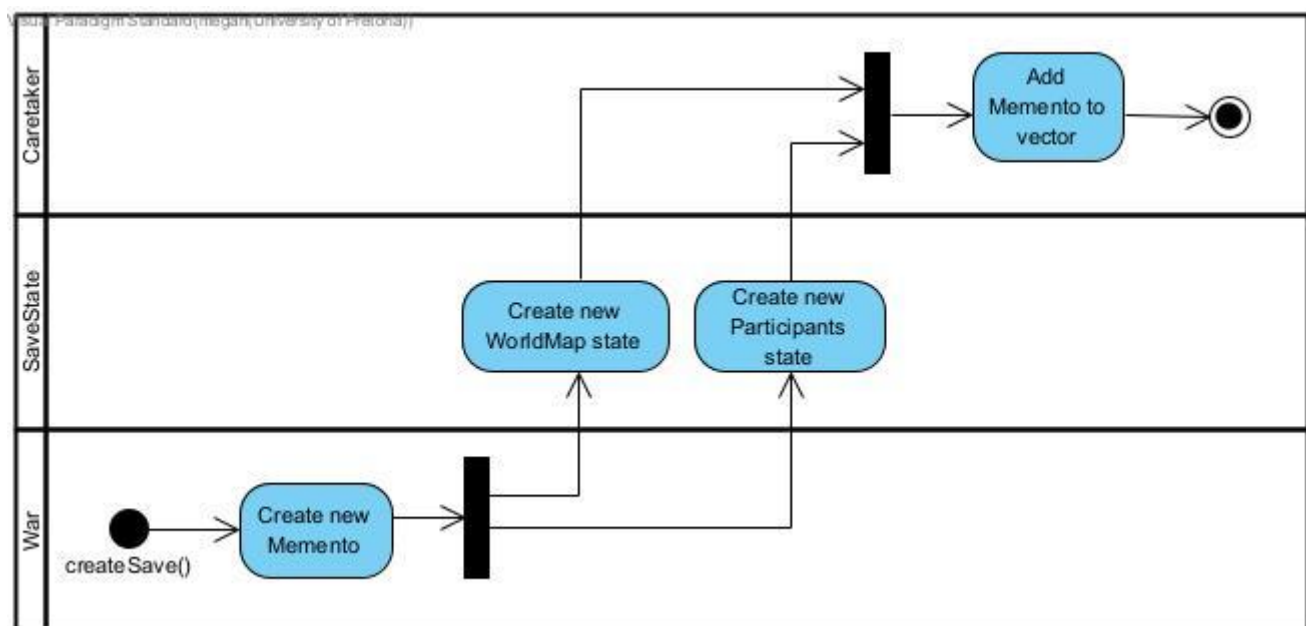


Fig 2.2.7 Memento Activity Diagram

2.3 Design Patterns

1. Abstract Factory

Solves the problem of producing entities by allowing us to produce multiple types of entities that belong to different countries.

2. Chain of responsibility

Solves the problem of allowing each entity to perform a different attack.

3. Command

Allows entities to receive the attack command.

4. Composite

Provides the ability to build complex squads of multiple troops.

5. Decorator

Allows for resources to be allocated to cells.

6. Factory Method

Provides the ability to have multiple resources on a single map.

7. Memento

Helps implement design mode by allowing for the state of the battle to be saved and to be manipulated at any time.

8. Prototype

Allows for multiple of the same unit to be created.

9. Template Method

Provides the ability for both powers to retreat.

10. State

Solves the functional requirement of squads being able to switch between the aggressive and the defensive state.

11. Strategy

Provides squads with the ability to perform either a frontline attack or a mining attack.

2.4 Class Structure

1. Abstract Factory

Participants:

- AbstractFactory: Factory
- ConcreteFactory: AlliedFactory, CentralFactory
- AbstractProduct: Infantry, Tank
- ConcreteProduct: AlliedInfantry, CentralInfantry, AlliedTank, CentralTank

2. Chain of Responsibility

Participants:

- Handler: Bombardment
- ConcreteHandler: InfantryDamage, TankDamage, SquadDamage

3. Command

Participants:

- Command: Order
- ConcreteCommand: Bombardment
- Invoker: MilitaryUnit
- Receiver: Cell

4. Composite

Participants:

- Component: MilitaryUnit
- Composite: Squad
- Leaf: TeamMembers

5. Decorator

Participants:

- Component: Cell
- ConcreteComponent: Bog, Flatlands
- Decorator: CellFeatures
- ConcreteDecorator: AmmoDeposit, FuelDeposit

6. Factory Method

Participants:

- Creator: FeatureFactory
- ConcreteCreator: AmmoDepoFactory, FuelDepoFactory
- Product: CellFeatures
- ConcreteProduct: AmmoDeposit, FuelDeposit

7. Memento

Participants:

- Caretaker: Caretaker
- Memento: SaveState
- Originator: War

8. Prototype

Participants:

- Prototype: MilitaryUnit, TeamMembers, Infantry, Tank
- ConcretePrototype: Squad, AlliedInfantry, AlliedTank, CentralInfantry, CentralTank

9. Template Method

Participants:

- AbstractClass: Participants
- ConcreteClass: CentralPowers, AlliedPowers

10. State

Participants:

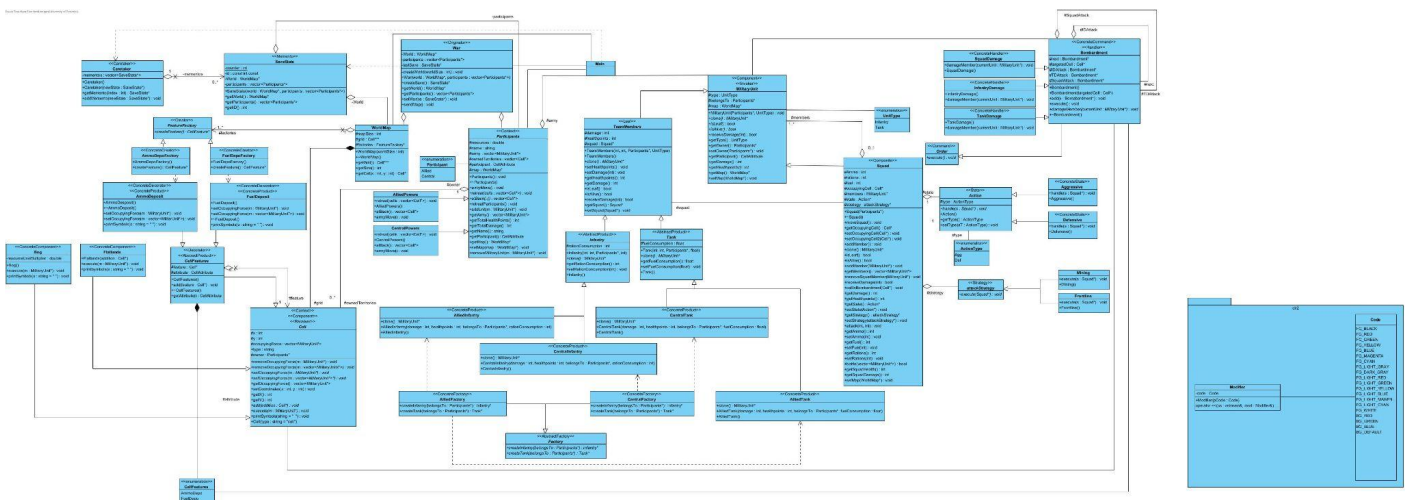
- Context: Squad
- State: Action
- ConcreteState: Aggressive, Defensive

11. Strategy

Participants:

- Context: Squad
- Strategy: AttackStrategy
- ConcreteStrategy: Mining, Frontline

2.5 Class Diagram



Class Diagram can be accessed [here](#).

2.6 Sequence and Communication Diagrams

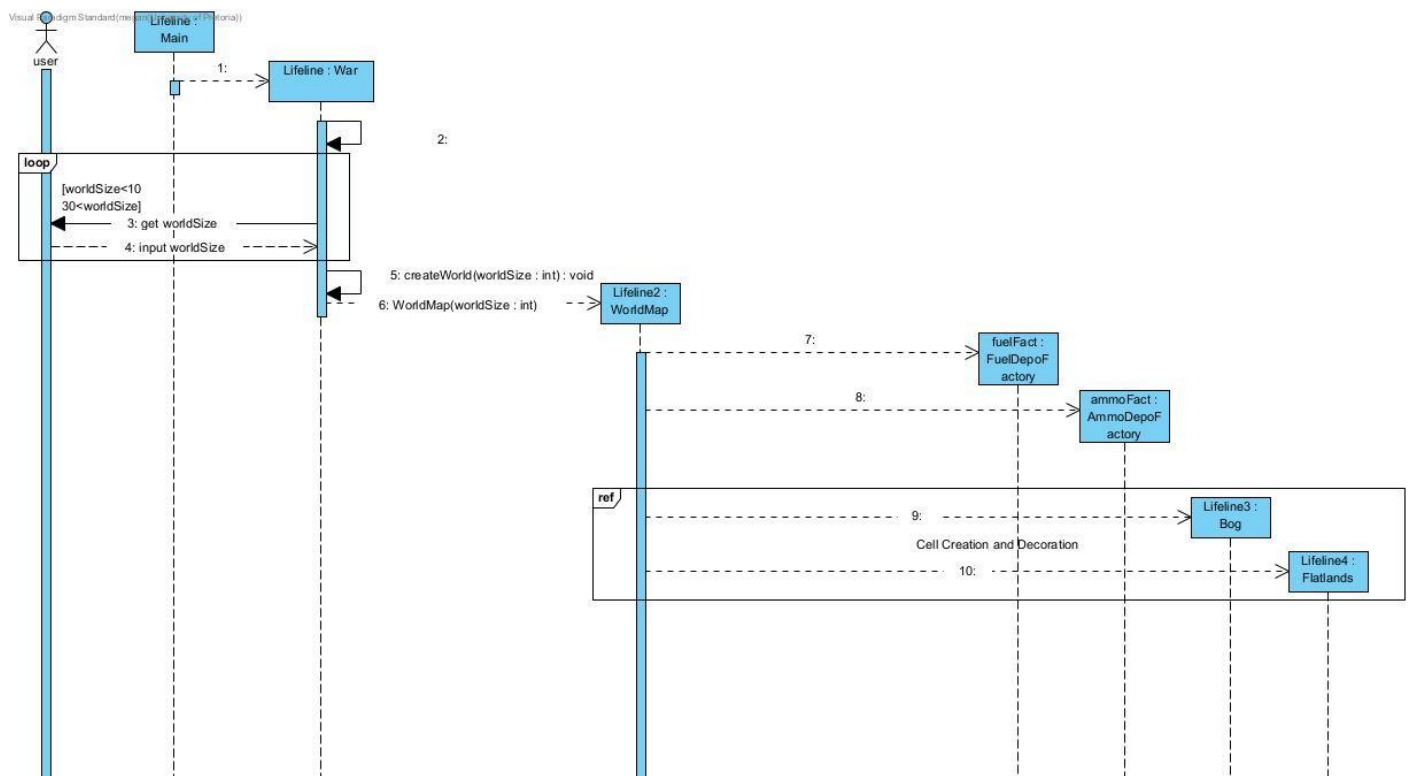


Fig 2.6.1 War Initialization Sequence Diagram

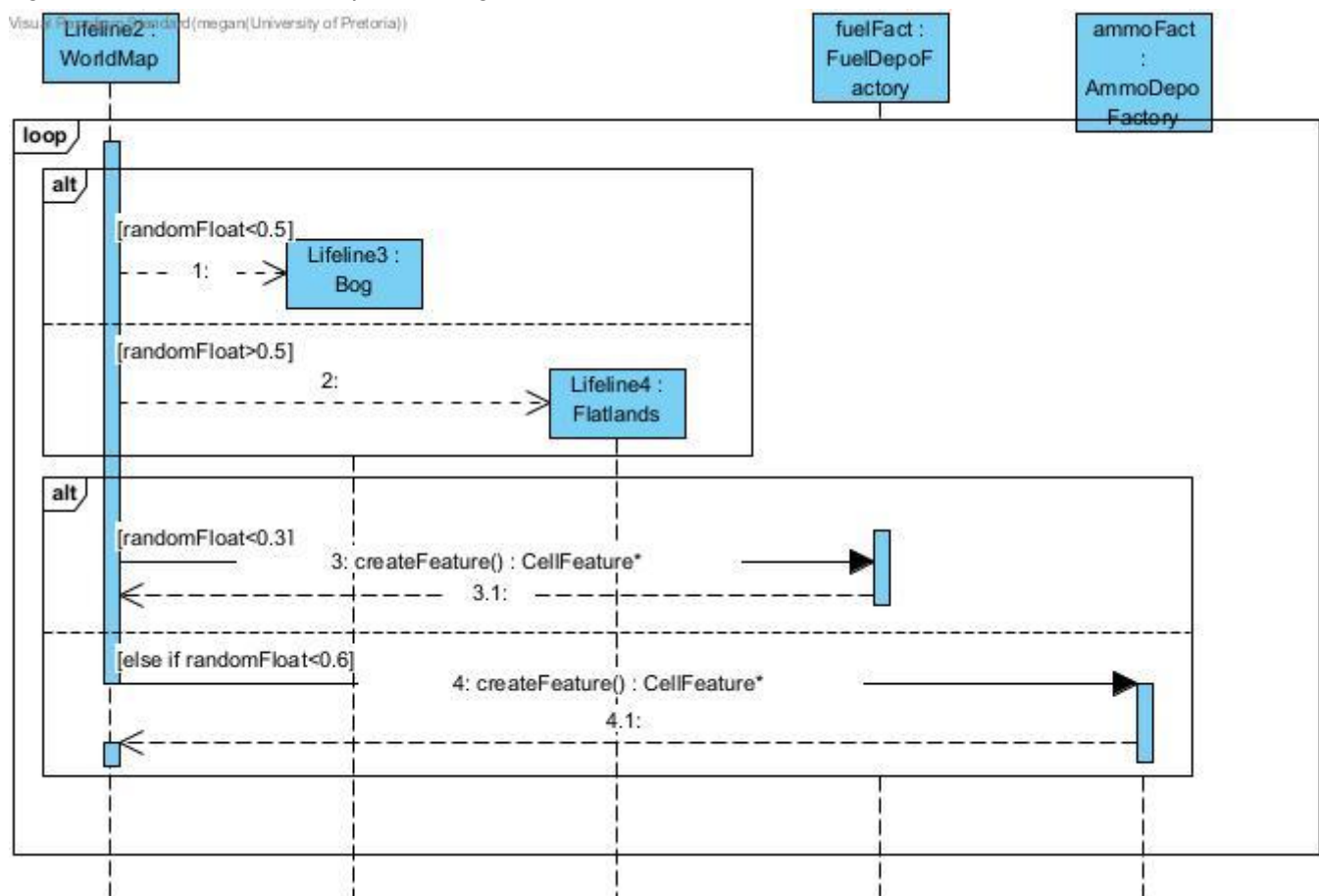


Fig 2.6.2 Cell Creation and Decoration Sequence Diagram

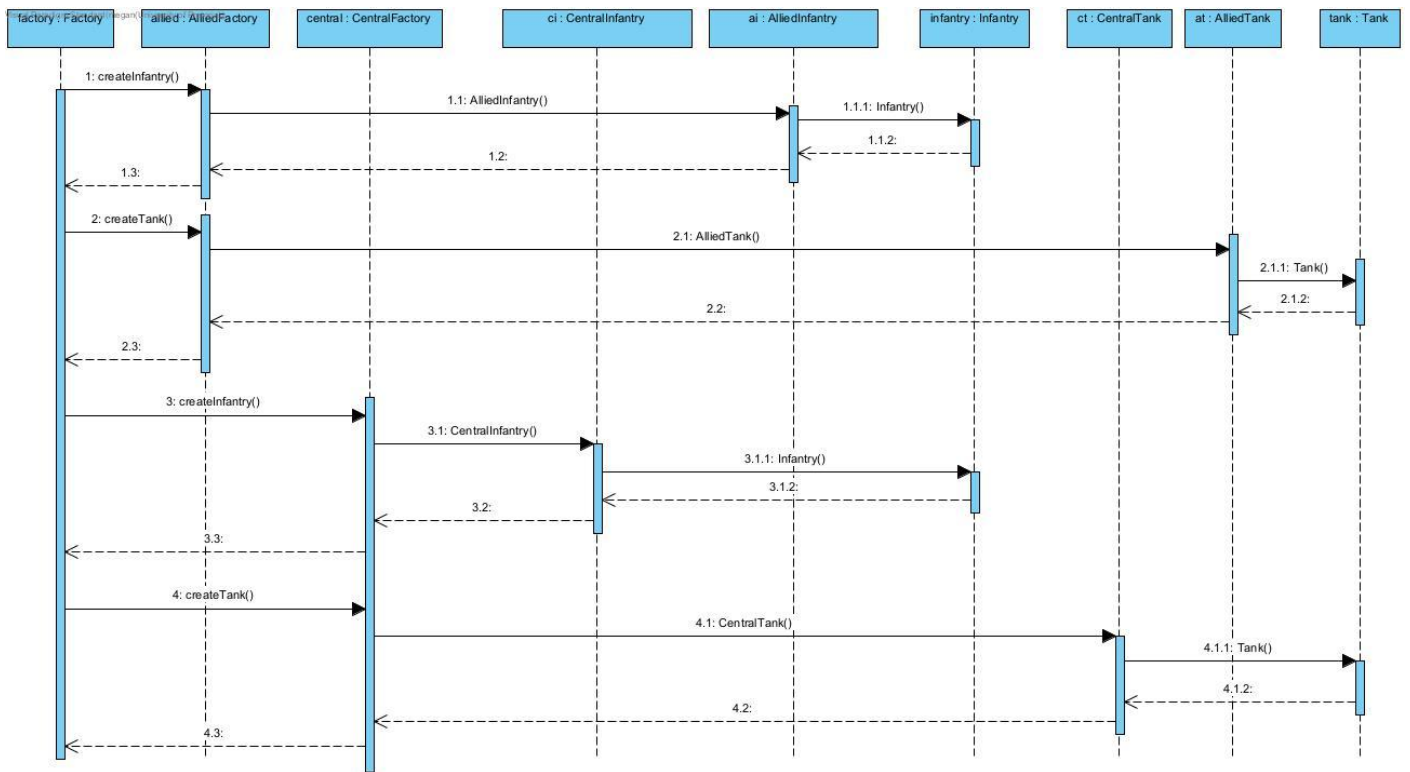


Fig 2.6.3 AbstractFactory Sequence Diagram

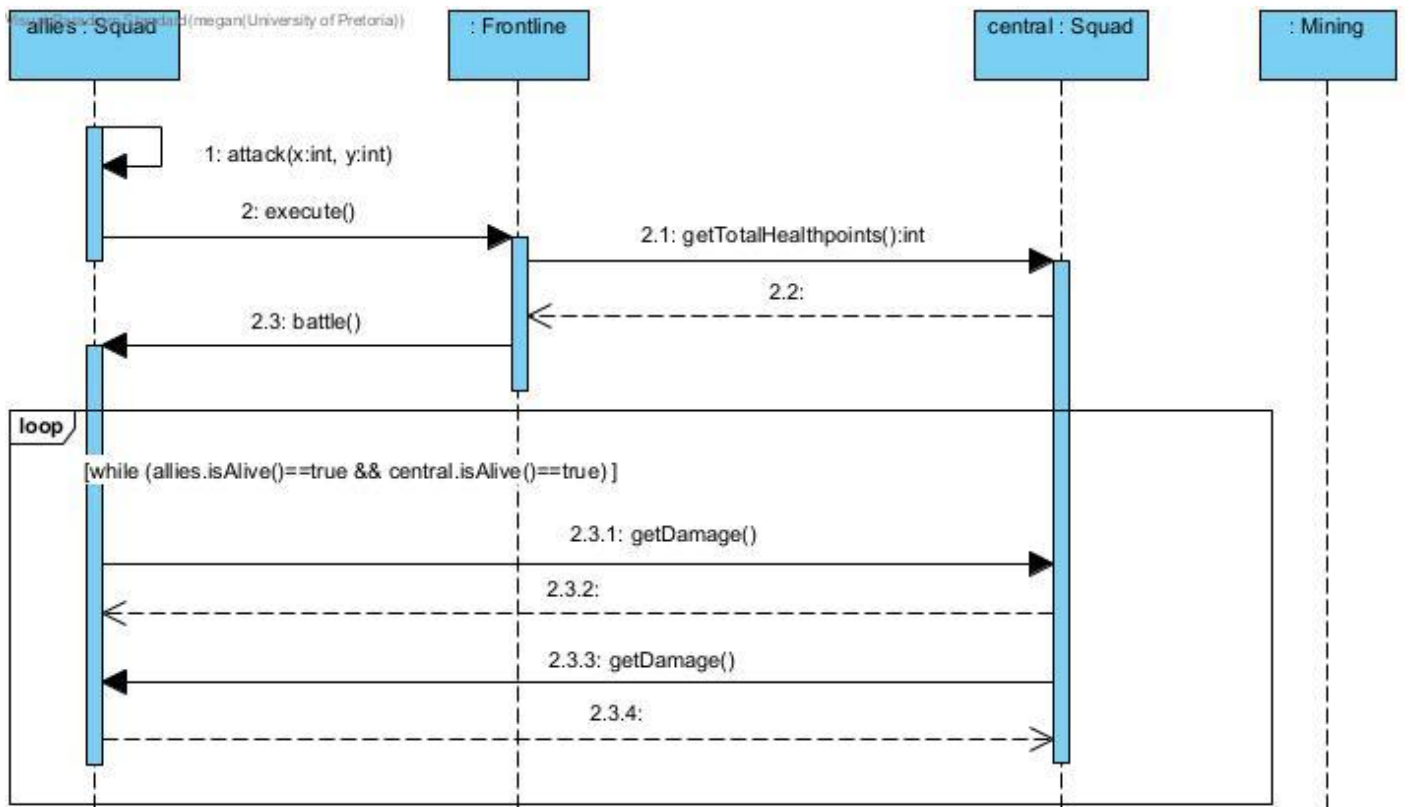


Fig 2.6.4 Strategy Sequence Diagram

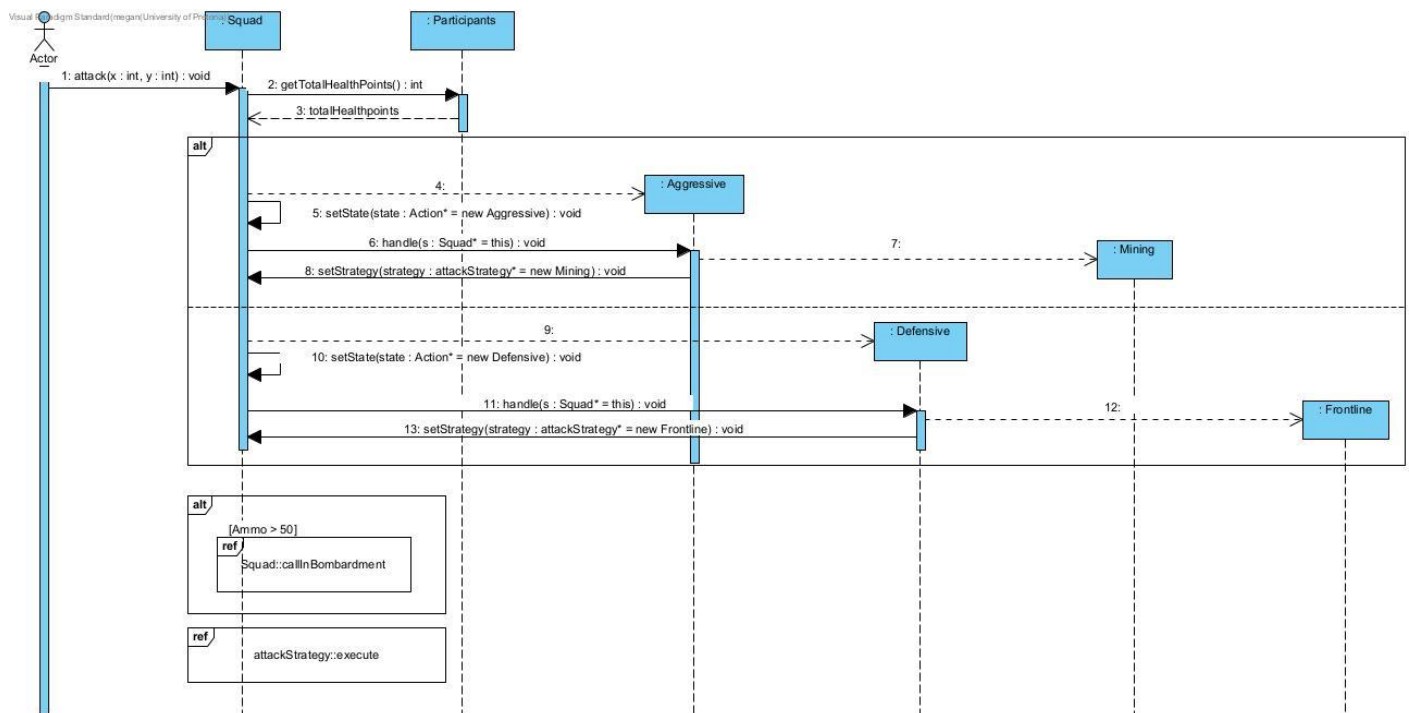


Fig 2.6.5 Squad::attack() method Sequence Diagram

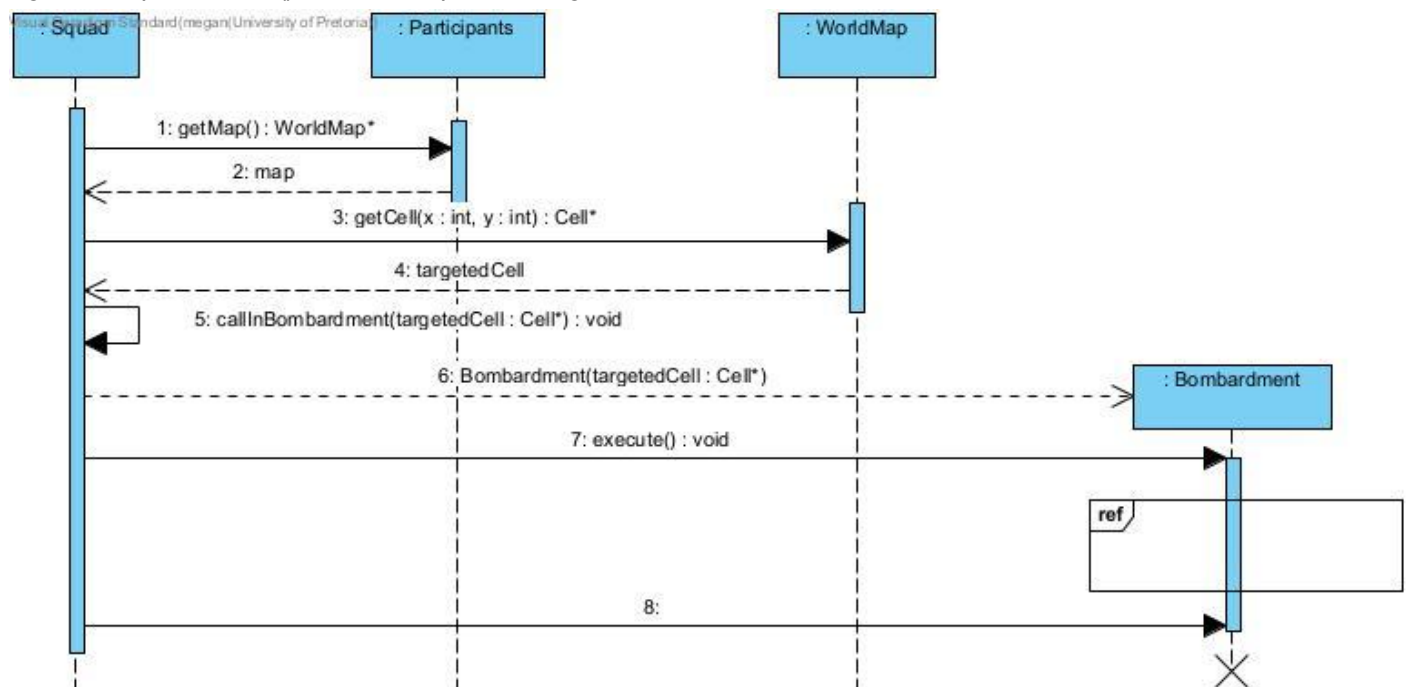


Fig 2.6.6 Squad::Bombardment() Sequence Diagram

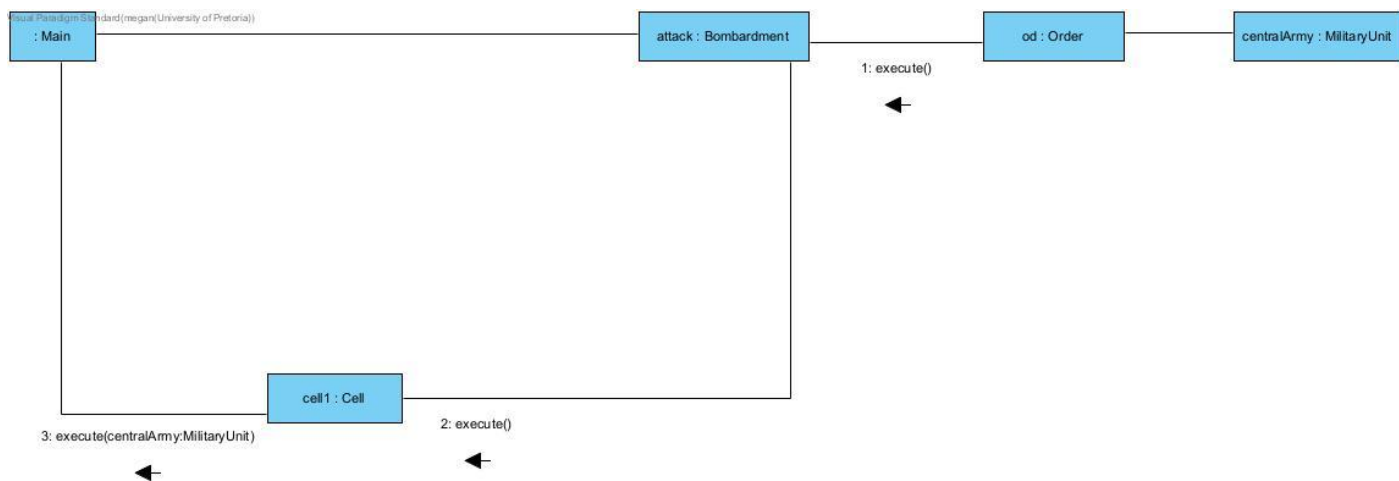


Fig 2.6.7 Command Communication Diagram

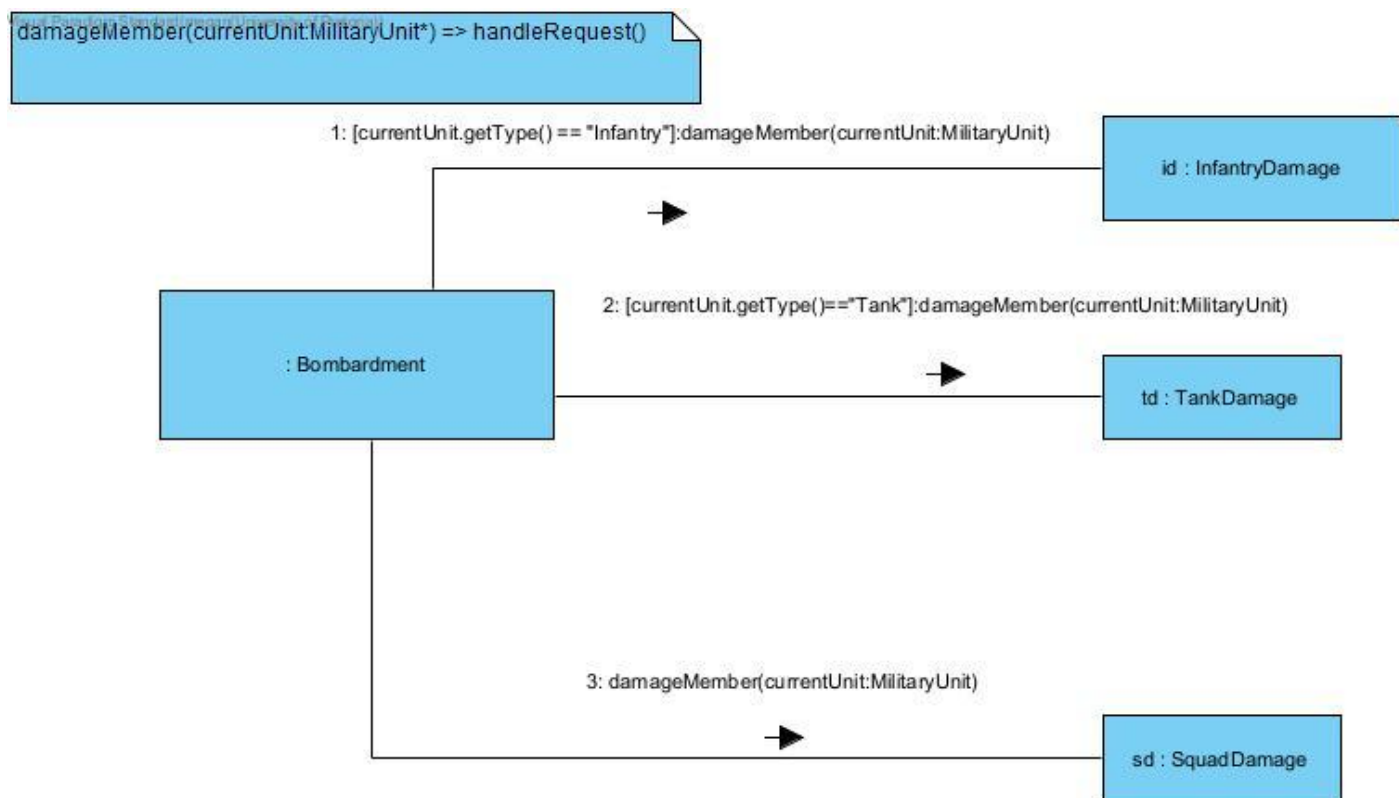


Fig 2.6.8 Chain of Responsibility Communication Diagram

2.7 State Diagrams

Visual Paradigm Standard (megan@University of Pretoria)

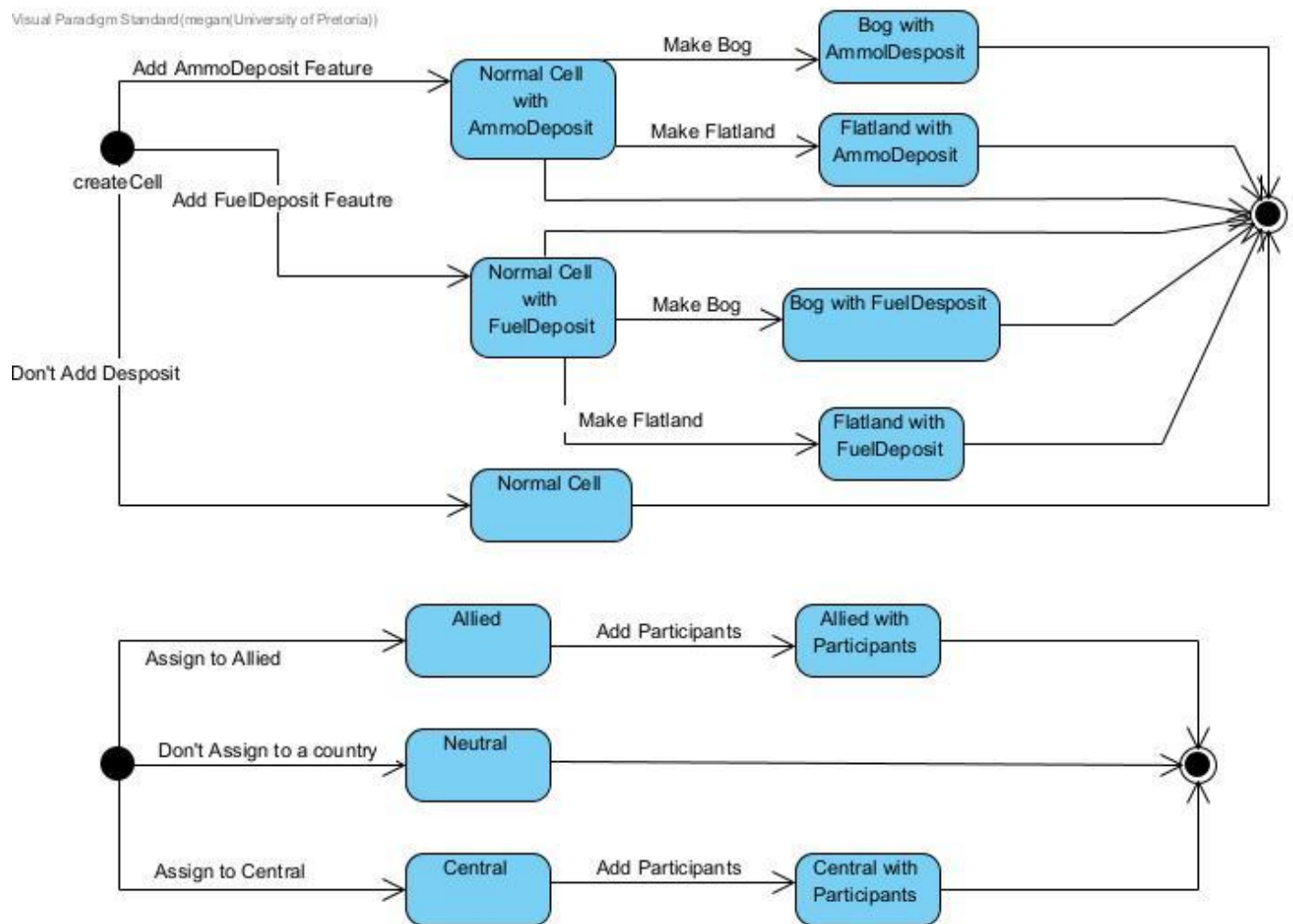


Fig 2.7.1 Cell State Diagram (Both)

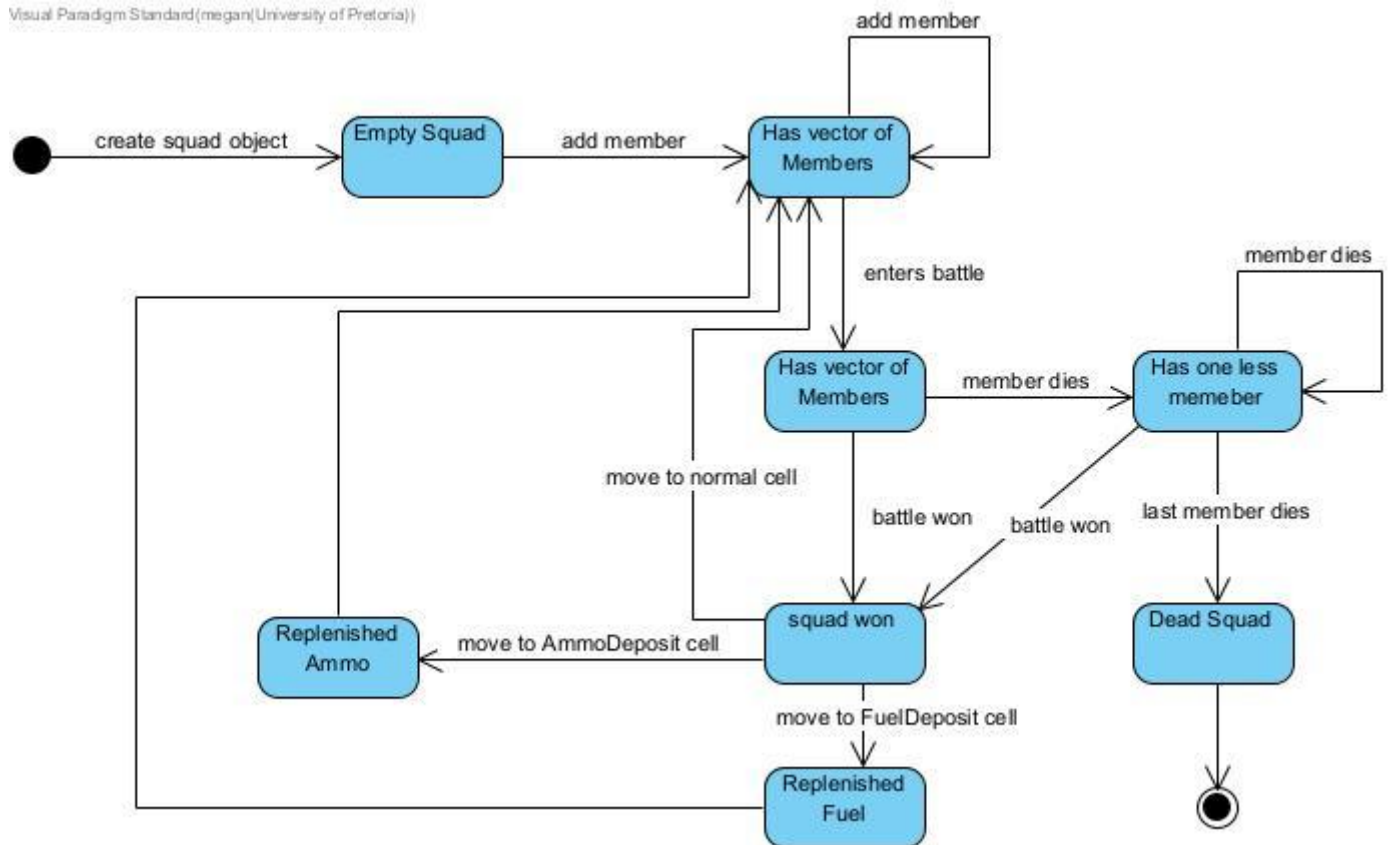


Fig 2.7.2 Squad State Diagram

2.8 Object Diagrams

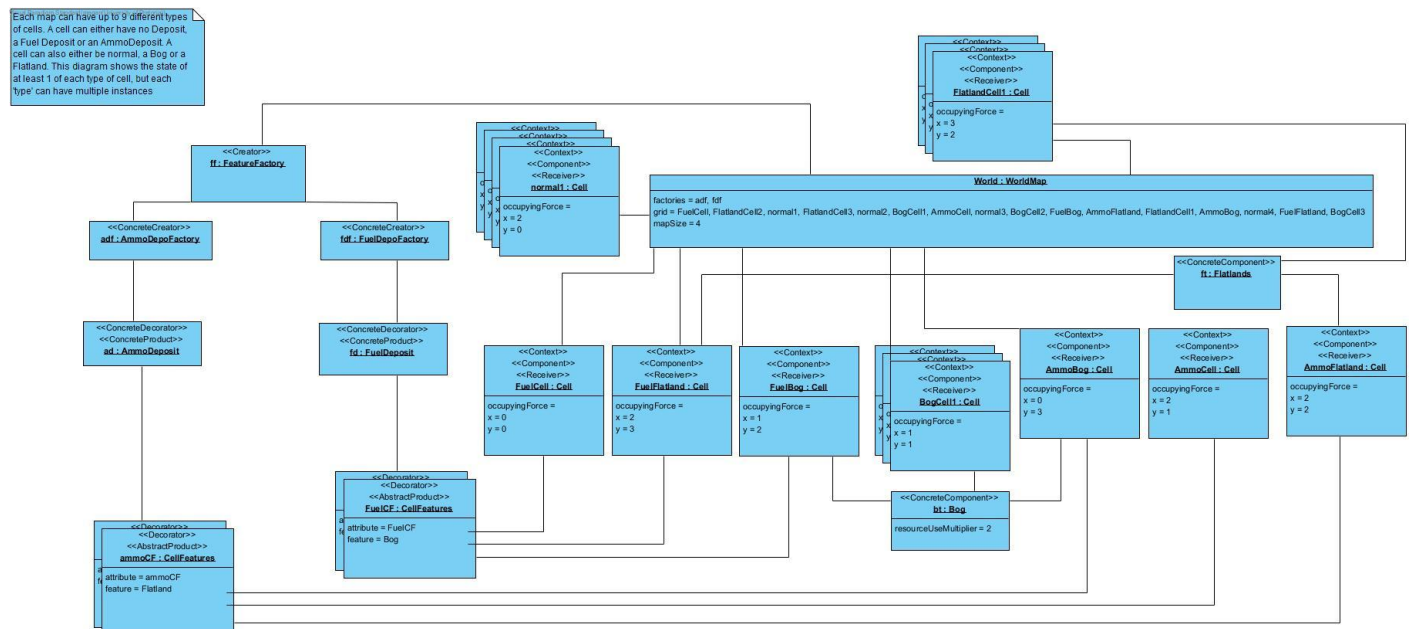


Fig 2.8.1 Object Diagram after the creation of the map

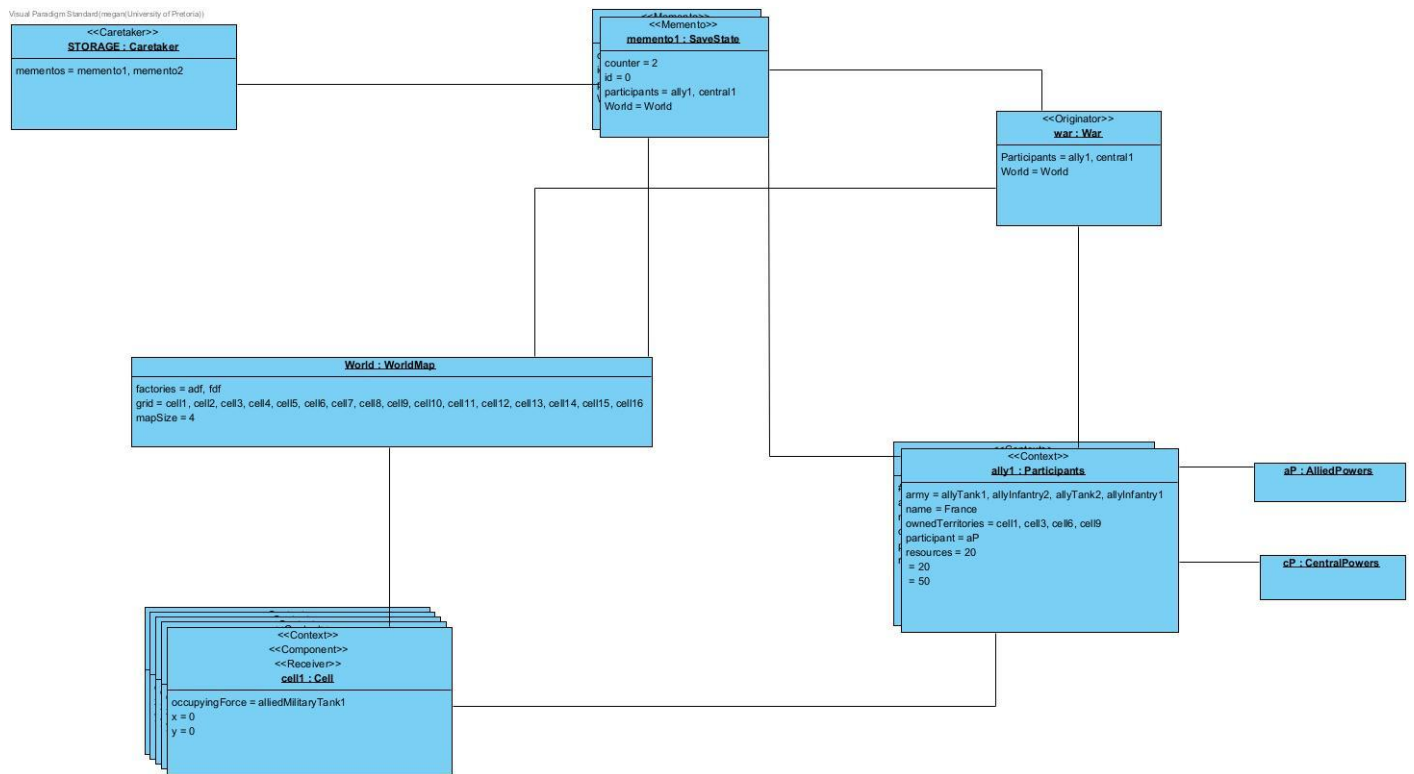


Fig 2.8.2 Object Diagram of creating a Memento

[Link to all diagrams.](#)

Task 3: Implementation

```
Allied health: 16650
Central health: 17250
Allied Powers lost the war
[ Cel 0 Cel 0 Cel 0 BogA 0 Cel 1 Cel F0 Cel 1 Cel 0 FlaA 0 Cel 0 ]
[ Cel 0 Bog F0 Cel 1 Cel 1 CelA 0 Cel 1 CelA 0 Cel 0 BogA 0 Cel 1 ]
[ Bog F1 Cel 0 Cel 0 Cel 1 Cel 0 Cel 0 Cel 0 Cel 0 Cel 1 Cel 0 ]
[ Cel 1 Cel 0 Cel 0 Cel 0 Cel F1 Cel 0 Cel 0 Cel 0 Cel 0 Cel 0 ]
[ Cel 0 Cel 0 Cel 0 Cel 0 Cel 2 Bog F0 CelA 0 Cel 1 Cel 0 FlaA 0 ]
[ Cel 0 Cel 0 CelA 1 Cel 1 FlaA 0 Cel 0 Cel 0 Cel 0 Cel 1 Cel 1 ]
[ Cel 1 BogA 0 Cel 0 Cel 1 Cel 0 Cel 0 Cel 0 Cel 0 Cel 1 Cel 1 ]
[ Cel 1 Cel 0 FlaA 0 Fla F1 Cel 0 Cel 0 CelA 0 Cel 0 Cel 0 Cel 0 ]
[ Cel 0 Fla F0 Bog F0 Cel 0 Cel F2 Cel 0 CelA 0 Cel 0 Cel 0 Cel 0 ]
[ Cel 0 Cel 0 Cel 1 Cel 1 CelA 0 Cel 1 Cel 0 Cel 0 Cel 0 Cel 1 ]
```

Task 4: Report

4.1 Research Brief

World War 1, also known as “The Great War”, began toward the end of July 1914 and concluded in November 1918. Europe, the Middle East, Africa and parts of Asia were enveloped in fighting. The Allied Powers, consisting of Great Britain, France, Italy, Russia, Romania, Canada, Japan and the US, fought against the Central Powers which included Germany, Austria-Hungary, Bulgaria and the Ottoman Empire. The sky became a new theatre of war as the invention of the aeroplane had just been brought into the world and with the combination of trench warfare and chemical weapons, the world bore witness to one of the most brutal and deadly wars in human history.

Trench warfare is a form of land warfare that is deeply rooted in WWI. This consists of both sides digging trenches which sheltered soldiers from both *incoming fire*¹ and artillery strikes. Advancing forces were at a significant disadvantage as crossing over to the opposing sides’ trenches, accurately named “No Mans Land”, provided little cover for soldiers against the defending side’s fire and artillery. On some occasions, trenches were built using sandbags filled with clay. This technique was used when groundwater constantly flooded the dug trenches. In order to overcome the advantages enemies had within their trenches, different strategies were used. One of these strategies was mining. Tunnels were dug underneath enemy trenches and explosives were detonated, destroying enemy trench lines. On some occasions troops could be moved undetected past enemy lines, allowing for surprise attacks to commence.

Armoured vehicles, like the tank, were prototyped first in 1915. Although throughout the war tanks were improved, their initial deployment was underwhelming as they were enormously heavy, overheated and barely rolled at two miles per hour. This meant that getting stuck in trenches was a regular occurrence as manoeuvring this behemoth was beyond difficult. They were also highly fuel-thirsty, using 800 litres every 30 km on average.

Resupplying troops was a significant issue as supply lines were often cut off when parts of an army were completely surrounded by enemy combatants. Resupplying often was carried out by railway but the closer the tracks got to the frontline, the more these supply routes were prone to be damaged by artillery. Smaller supply dumps were formed to prevent this. Horse carts were then used to transport these supplies from the smaller dumps to the frontlines to parts of the army that required them. The further away from these supply dumps that troops were, the more difficult it was to transport food, water and ammunition to them as resupplies needed to cross war-ridden terrain in order to reach the troops.

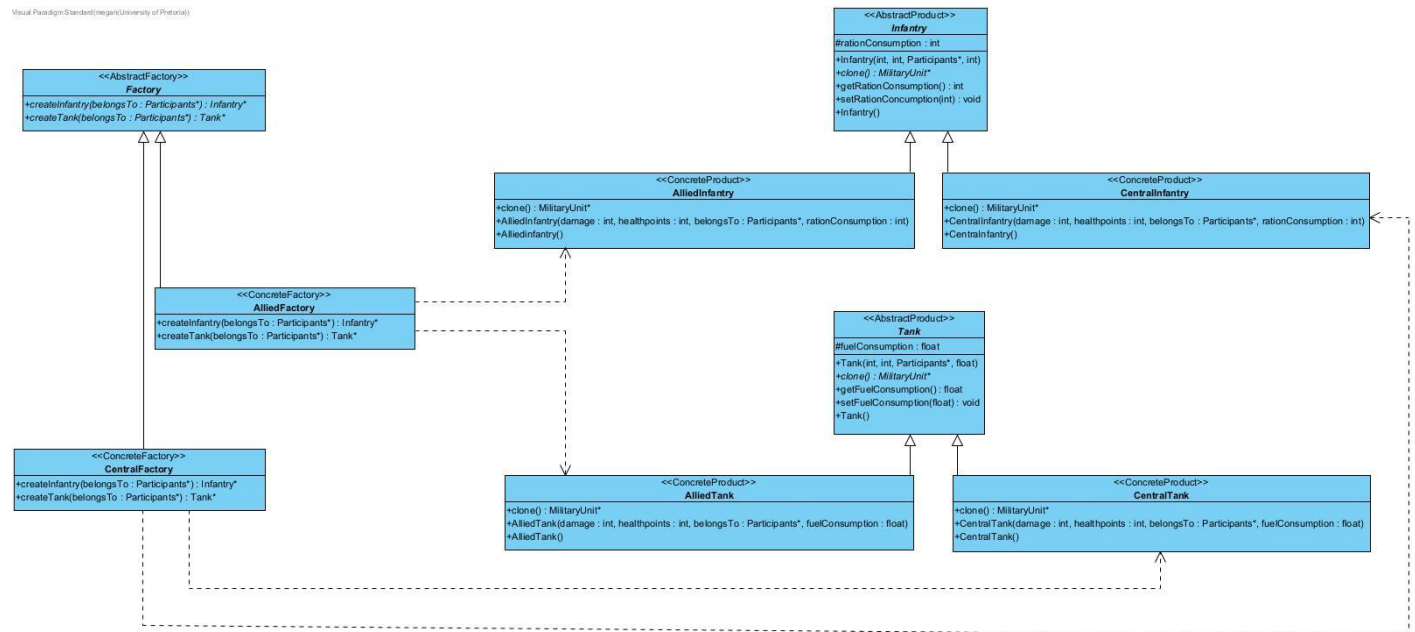
¹ When the opposing side is firing in the direction of oneself

4.2 Application of Design Patterns

The system that has been designed is a complicated construction of 11 design patterns that are used to efficiently and effectively implement our war simulation. Each design pattern not only plays a unique role in the system by addressing specific functional requirements but also works together to form a larger interface, which itself will also address specific functional requirements. Below is a list of our design patterns as well as a minimal definition of what that design pattern's intent in our program is.

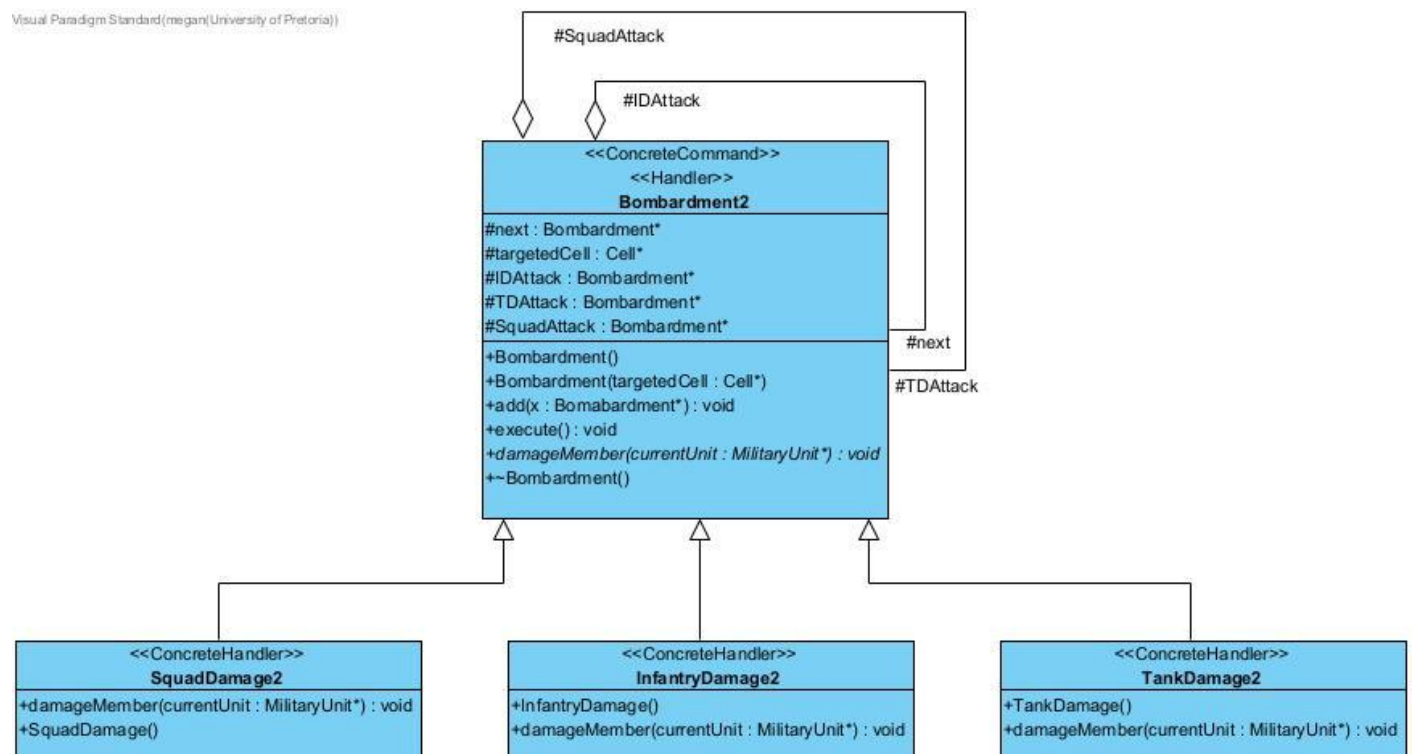
1. Abstract Factory

Intent: This is an interface for producing families of related TeamMember objects without specifying the concrete classes.



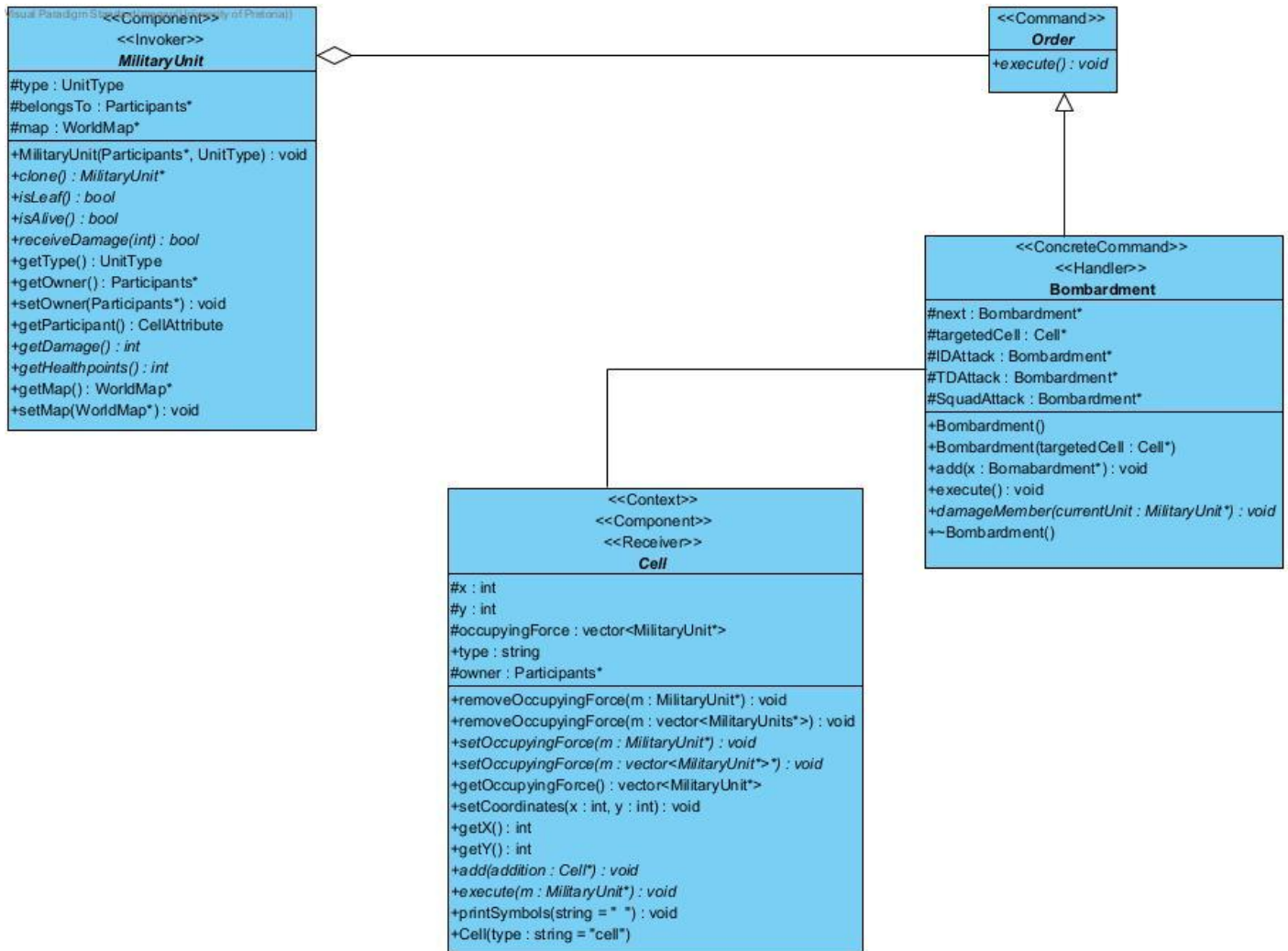
2. Chain of responsibility

Intent: Lets you pass bombardment requests along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it to the next handler in the chain.



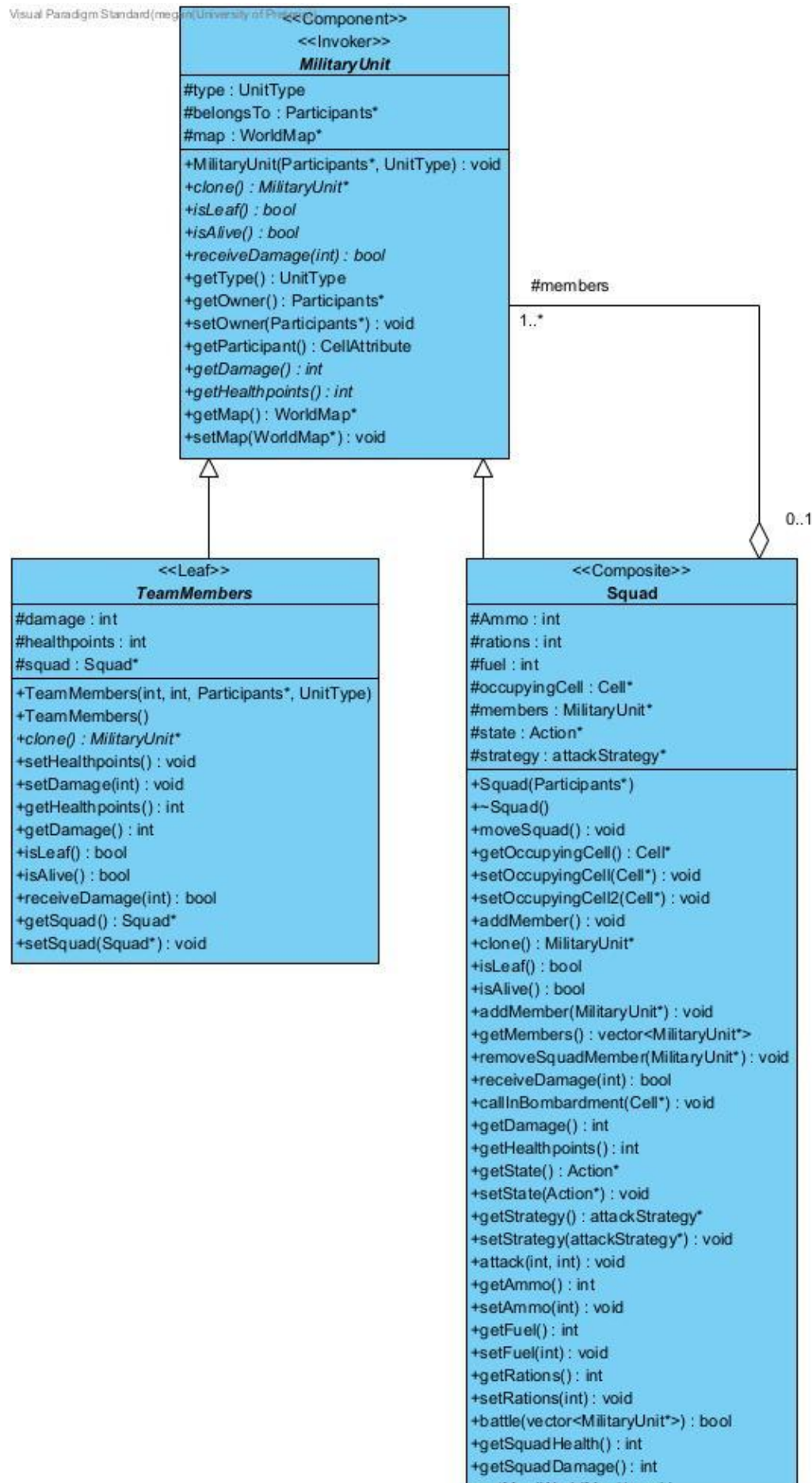
3. Command

Intent: Encapsulates a request into an object that contains all information about the attack request.



4. Composite

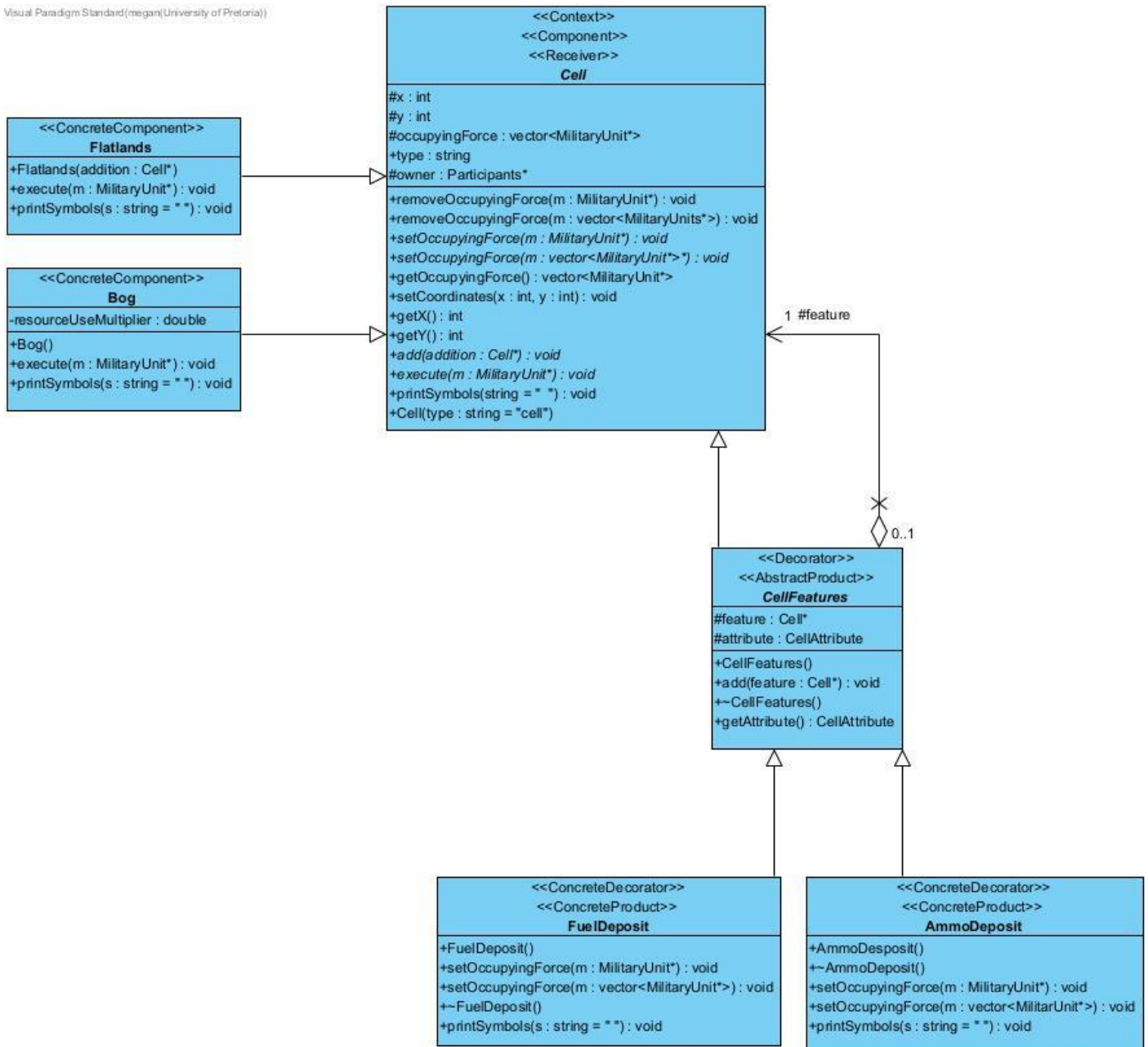
Intent: Allows the creation of unique Squads through a set of hierarchies which form a tree structure.



5. Decorator

Intent: Allows for the attachment of different deposits to cells.

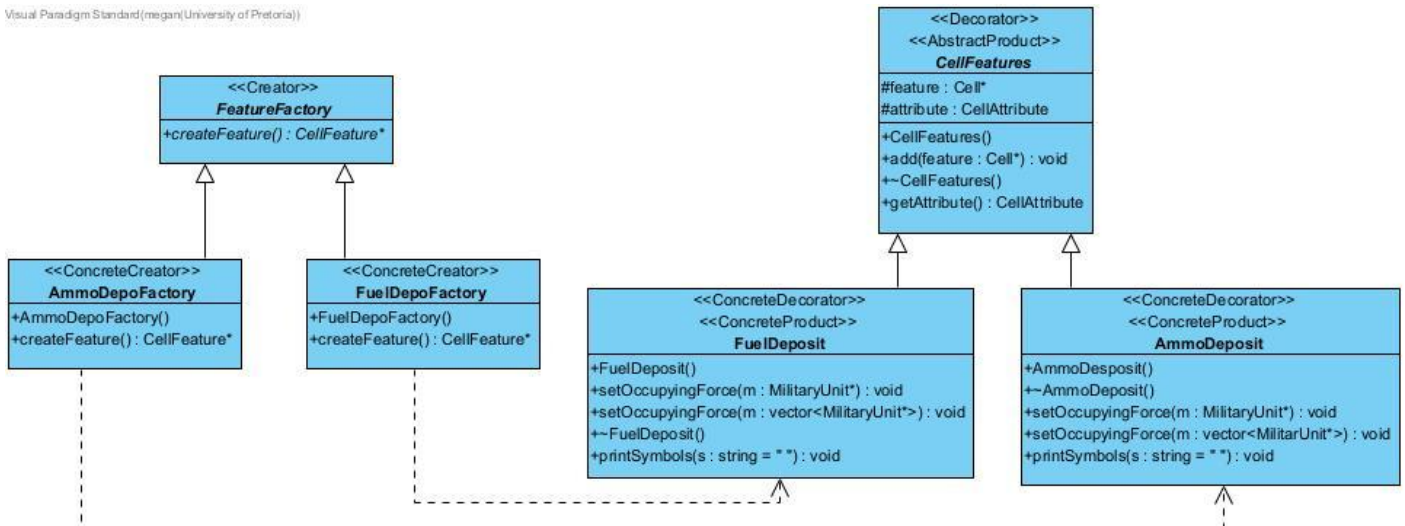
Visual Paradigm Standard (megan@University of Pretoria)



6. Factory Method

Intent: This is an interface for creating resource objects in a superclass, but allows subclasses to select the type of resources that will be created.

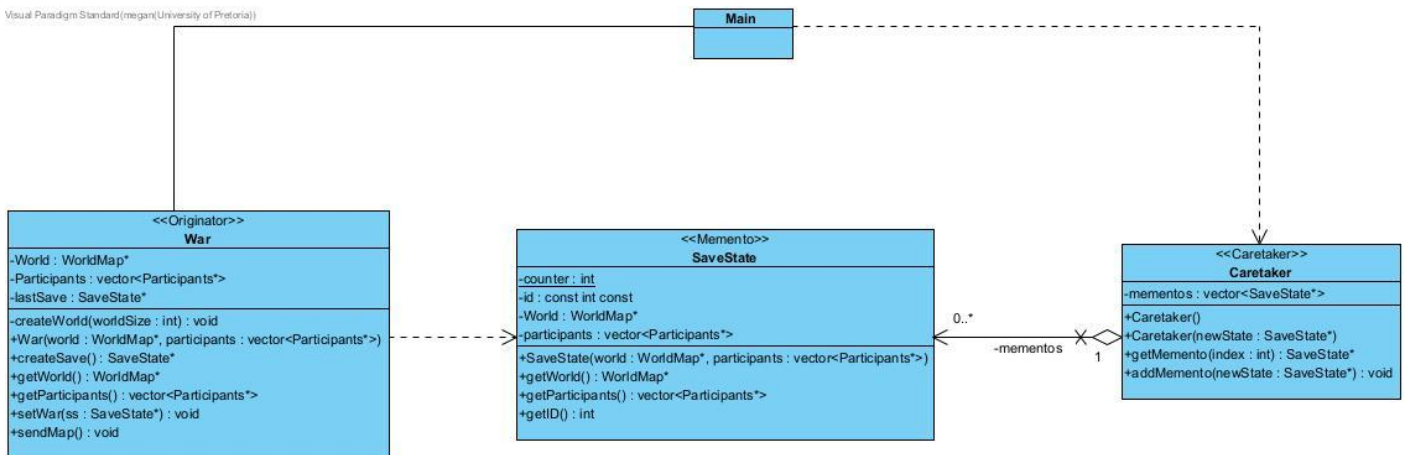
Visual Paradigm Standard (megan/University of Pretoria)



7. Memento

Intent: Capture and externalise the War's internal state so that the current War can be saved and restored to this state later when the war resumes.

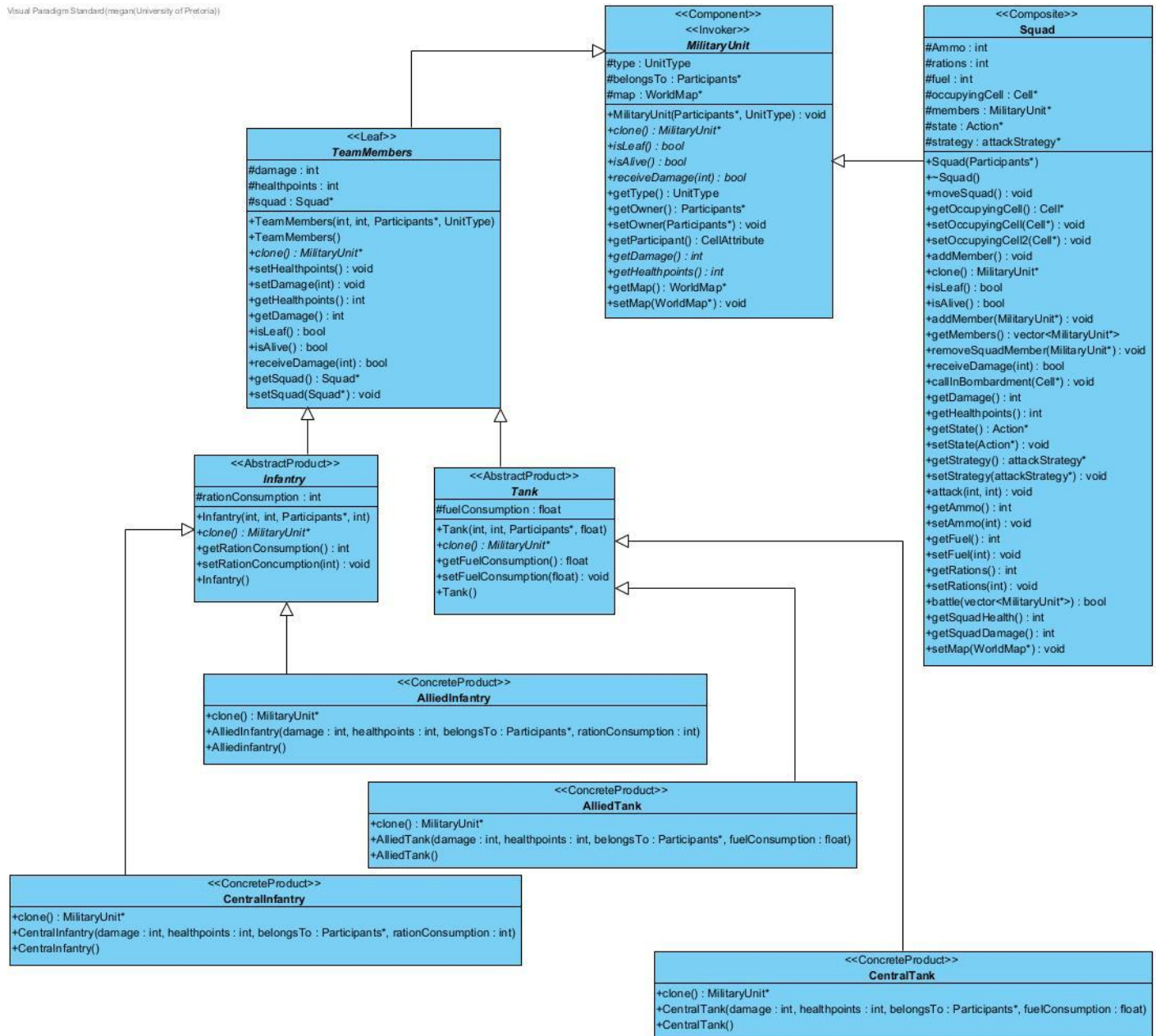
Visual Paradigm Standard (megan/University of Pretoria)



8. Prototype

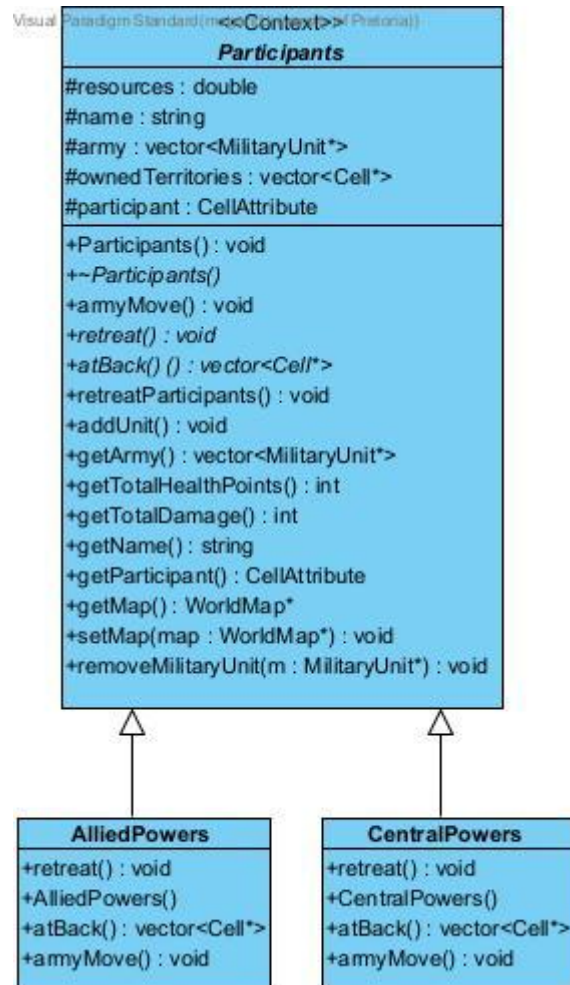
Intent: Specify the kinds of MilitaryUnit objects to create using a prototypical instance, and create new objects by cloning this prototype. Make copies of existing Military unit objects.

Visual Paradigm Standard (megan@University of Pretoria)



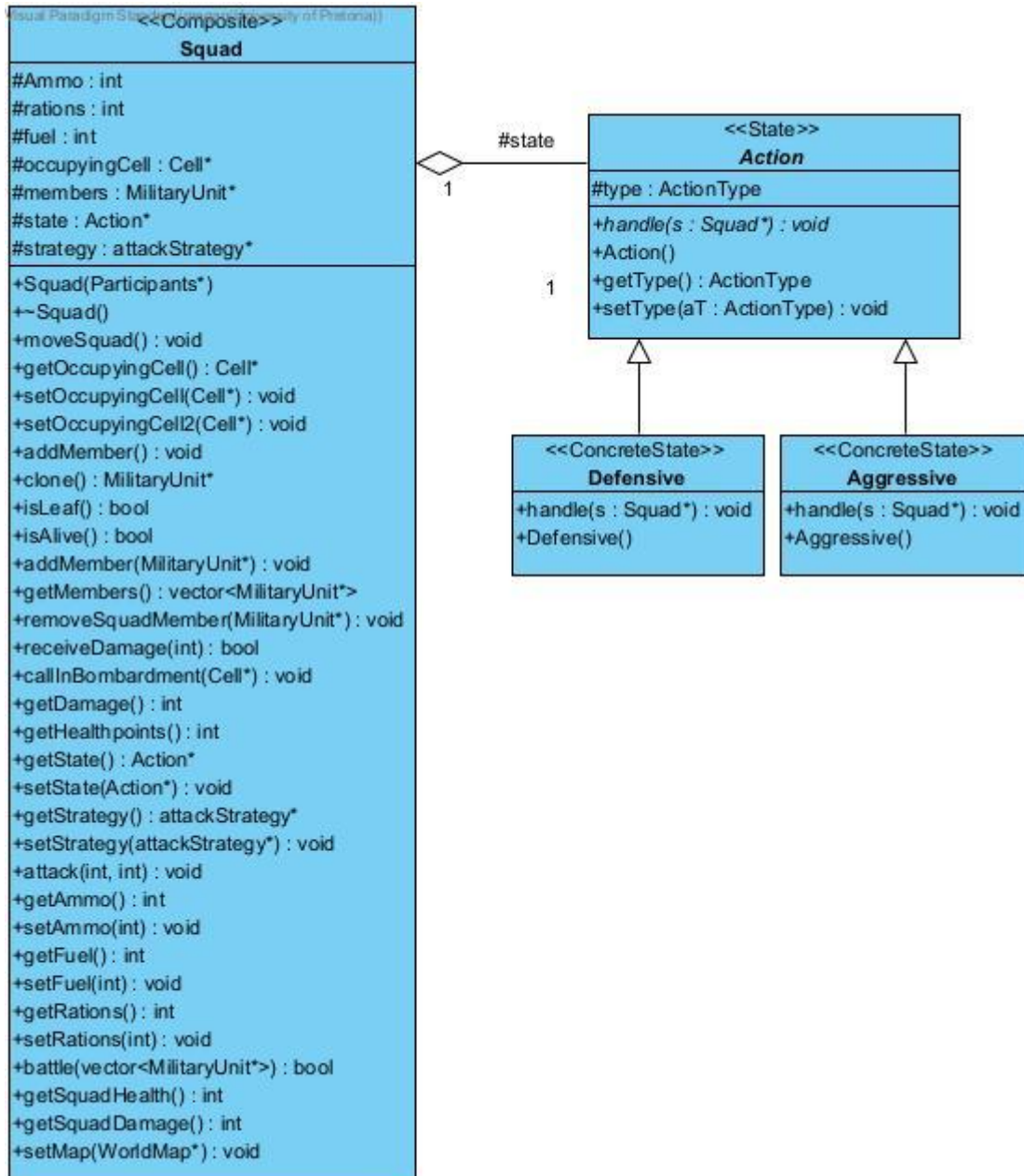
9. Template Method

Intent: Define the skeleton of the retreat algorithm in participants, deferring some steps to the subclasses CentralPowers and AlliedPowers.



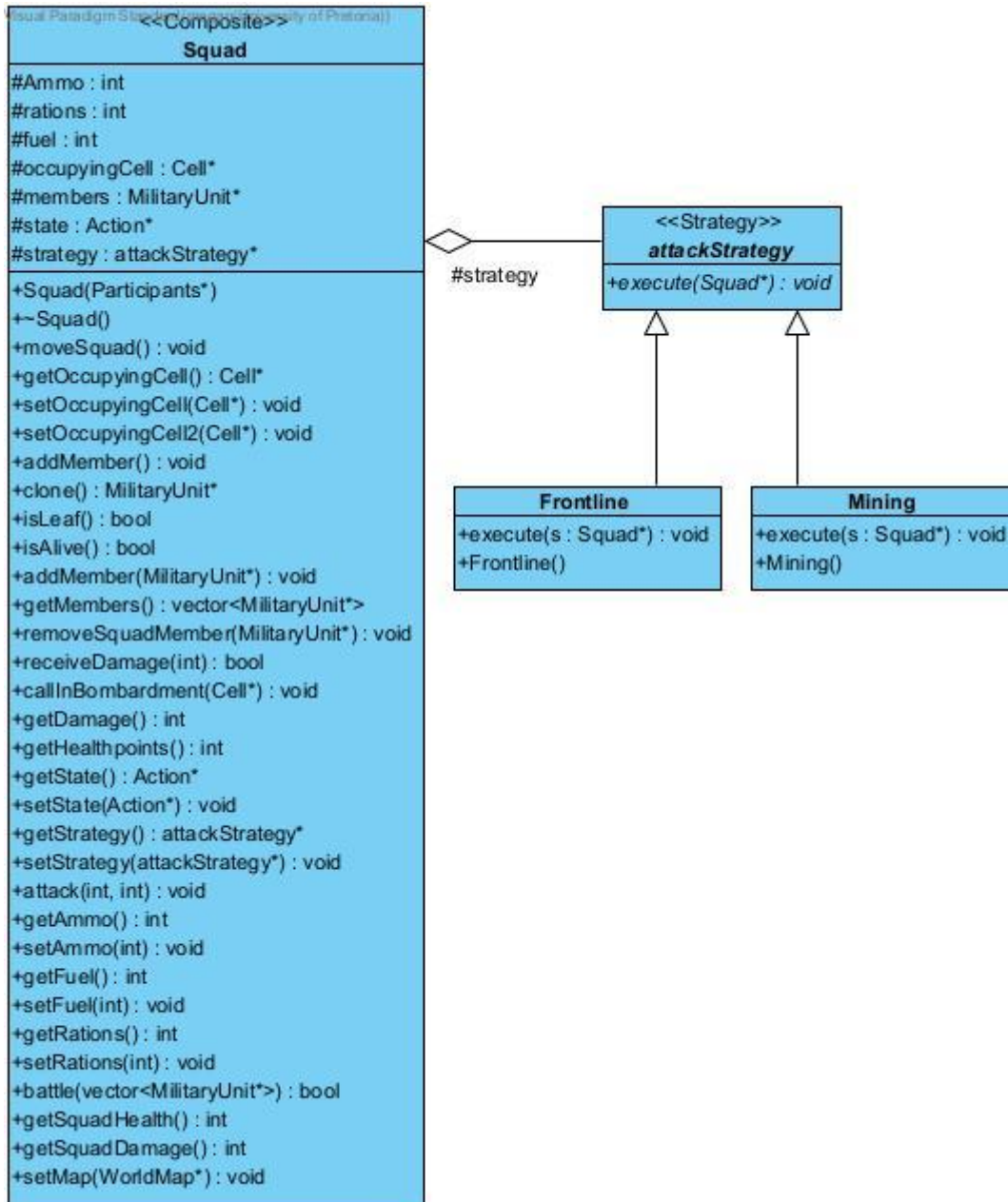
10. State

Intent: Allow participants to alter their behaviour when their internal state changes from aggressive and defensive.



11. Strategy

Intent: Define a family of algorithms and place them in different classes so that the algorithms can vary independently from the client.



Our application's interface is built on the above 11 patterns. These, however, are controlled by the main - which begins our simulation and allows it to run. It starts by generating the war theatre, which is the 2-dimensional map in which the simulation will take place. The factory method and decorator pattern are our key patterns in creating the map. The factory method allows our 2D map to allocate resources to cells. It does this allocation by producing decorator objects which are then attached to the individual cells in the 2D map [Fig 2.6.2].

Furthermore, the map needs to be populated with the troops and squads which will make up the armies for the war. To achieve this we used the composite, prototype and abstract factory patterns. While the resources are being allocated to the map the composite pattern is used to create complex squads of multiple military units which will then be placed on the map. These units are produced by the abstract factory [Fig 2.6.3] as it allows us to have different types of units for both participants in the war. However, for the creation of duplicate troops that will share the same qualities we have implemented a prototype method which is used to make clones of any given troop.

Once all units have been generated, each squad on the map will have a state, either aggressive or defensive, which will be implemented using the state pattern. Using this state the strategy will be which the squad will use to attack will be assigned [Fig 2.2.4]. The different squads attack using either a mining or a frontline (see task 4.1 above) attack which we have implemented using a strategy design pattern [Fig 2.2.3 and Fig 2.6.4].

Furthermore, a participant can attack their opponent using a bombardment [Fig 2.6.6]. This is implemented using the chain of responsibility [Fig 2.2.6 and Fig 2.6.8] as each handler decides either to process the request or to pass it to the next handler in the chain. The command pattern [Fig 2.6.7] is used to encapsulate the information of this attack.

When one of the participants needs to retreat, the different opponents need to move in opposite directions but the skeleton of the algorithm remains the same. The template pattern [Fig 2.2.5] is used to defer the direction of the retreat to the subclasses CentralPowers and AlliedPowers.

The memento design pattern [Fig 2.2.7] is used in the design mode of the simulation to save the current state of the war and restore this state when the war is resumed.

Task 5: Development Practices

5.1 Use of GitHub



Github Username | Team member

Rob-Off | Robert Officer

Slaaiblaar | Wian Koekemoer

Kaityss | Kaitlyn Sookdhew

TashUni | Latasha Friend

Meggie-H | Megan Hugo

ogb-Welsh | Oliver Welsh

5.2 C++ Documentation

Comments can be found in the code itself.

5.3 Doxygen

[Link to Doxygen pdf](#)

5.4 Code Development

Project was developed using contract first design, where we designed and modelled our design patterns and classes before coding them

5.5 Unit Testing

Three groups of tests were run. One for the Composite pattern, Memento pattern and Participants class, which forms part of the Template pattern.

Composite Test

Test 1 - Test that constructors function correctly.

Test 2 - Tests the initial state of the Squad component.

Test 3 - Checks if the map that is given to each MilitaryUnit.

Test 4 - Tests if deletions of MilitaryUnits remove them from the vector they are in.

Test 5 - Tests setting of OccupyingCell as well as setting the corresponding cells occupying force

Test 6 - Tests the receiveDamage function to check if damage has been received

Test 7 - Tests if the copying done is a deep copy of a shallow one and if values copied are the same across

Test 8 - Runs a small simulation of a battle to test if the outcoming values and outputs are correct


Memento Test

Test 1 - Tests to see if restoring SaveStates returns the correct SaveState.

Participants Test

Test 1 - Tests to see if deleting a squad removes it from its corresponding army automatically.

Task 6: Demo and Presentation

- Link to video: [Video Demonstration](#)
- Link to google doc:  **War Simulation Project [The Cookie Club]**
- Link to GitHub repository: [COS214_ War Project \[The Cookie Club\]](#)

References

Brooks, Ernest. n.d. "Trench warfare." Wikipedia. Accessed November 1, 2022.

https://en.wikipedia.org/wiki/Trench_warfare#Geography.

History.com Editors. 2009. "First Tank Produced." First Tank Produced - HISTORY. Accessed November 1,

2022. <https://www.history.com/this-day-in-history/first-tank-produced>.

Resch, Margit. n.d. "World War I." Wikipedia. Accessed November 1, 2022.

https://en.wikipedia.org/wiki/World_War_I.

Tanks Encyclopedia. n.d. "Little Willie, Lincoln Machine number 1 (1915)." Tank Encyclopedia. Accessed

November 2, 2022. https://www.tanks-encyclopedia.com/ww1/gb/little_willie.php.

Refactoring Guru (no date) *Design patterns, Refactoring.Guru*. Accessed:

November 7, 2022. <https://refactoring.guru/design-patterns>

Marshal, L. and Pieterse, V. (2021) *Tackling Design Patterns*. Department of Computer Science, University

of Pretoria. Accessed: November 7, 2022. <https://www.cs.up.ac.za/cs/lmarshall/TDP/TDP.html>

Plagiarism Declaration

I hereby certify that this report is my own work, except where duly acknowledged and that no plagiarism has been committed in the writing of the report.

Signed: Oliver Welsh of student number 21432962 on this day 07 November 2022.



Signed: Kaitlyn Sookdhew of student number 21483974 on this day 07 November 2022.



Signed: Robert Officer of student number 20431122 on this day 07 November 2022.



Signed: Wian Koekemoer of student number 19043512 on this day 07 November 2022.



Signed: Megan Hugo of student number 20538422 on this day 07 November 2022.



Signed: Latasha Friend of student number 21526053 on this day 07 November 2022.

