COS214 Project

Generated by Doxygen 1.8.15

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

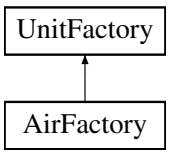
Chapter 3

Class Documentation

3.1 AirFactory Class Reference

`#include <AirFactory.h>`

Inheritance diagram for AirFactory:



Public Member Functions

- AirFactory (double budget, int level, std::string type="Air")

  Constructor for AirFactory class used to instantiate an AirFactory object.
- ArmyComponent * createVehicle ()

  Calls constructor of AirVehicle, using level to determine powerRating.
- ArmyComponent * createSoldier ()

  Calls constructor of AirUnit, using level to determine powerRating.

Additional Inherited Members

3.1.1 Detailed Description

The AirFactory class is a derived class derived from the UnitFactory class (See the definition of the UnitFactory class)

The AirFactory will be used to create Air Units for the War. The AirFactory has methods "createSoldier()" and "createVehicle()" which will create Soldier objects and Vehicle objects respectively.

Note

This class is ONLY used to create AirUnit objects (Soldiers or Vehicles)

3.1.2 Constructor & Destructor Documentation

3.1.2.1 AirFactory()

```
AirFactory::AirFactory (
            double budget,
            int level,
            std::string type = "Air" )
```

Constructor for AirFactory class used to instantiate an AirFactory object.

Author

Reuben Jooste (u21457060)

Parameters

budget	Starting budget of AirFactory class
level	Starting level of AirFactory class
type	Type will be "Air" since this function creates Air army components

Warning

The "budget" must be a positive value. The "level" must be greater than zero.

3.1.3 Member Function Documentation

3.1.3.1 createSoldier()

```
ArmyComponent * AirFactory::createSoldier ( )  [virtual]
```

Calls constructor of AirUnit, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a AirUnit)

Note

This function may return NULL if the budget has run out.

Implements UnitFactory.

3.1.3.2 createVehicle()

```
ArmyComponent * AirFactory::createVehicle ( )  [virtual]
```

Calls constructor of AirVehicle, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a AirVehicle)

Note

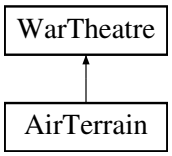
This function may return NULL if the budget has run out.

Implements UnitFactory.

3.2 AirTerrain Class Reference

```
#include <AirTerrain.h>
```

Inheritance diagram for AirTerrain:



Public Member Functions

- AirTerrain ()

  This is the default constructor of the class.

3.2.1 Detailed Description

(See the definition of the WarTheatre class)

The AirTerrain class is a derived class derived from the WarTheatre class The AirTerrain will be used to create an Air Terrain Theatre where the war can will take place.

Note

This class is used to expand the War by adding a new Theatre to it.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 AirTerrain()

```
AirTerrain::AirTerrain ( )
```

This is the default constructor of the class.

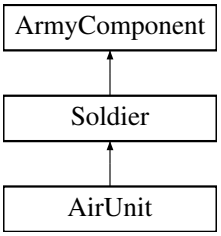
Author

Jonelle Coertze (u21446271)

3.3 AirUnit Class Reference

```
#include <AirUnit.h>
```

Inheritance diagram for AirUnit:



Public Member Functions

- AirUnit (int powerRating)

  Constructs AirUnit object, calling constructor of parent Soldier.
- int calculateAirOffense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's AirOffence statistic.
- int calculateAirDefense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's AirDefence statistic.
- int calculateSeaOffense ()

  Calculates the SeaOffense statistic of the unit.
- int calculateSeaDefense ()

  Calculates the SeaDefence statistic of the unit.
- int calculateLandOffense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's LandOffence statistic.
- int calculateLandDefense ()

  Calculates the LandDefence statistic of the unit.

Additional Inherited Members

3.3.1 Detailed Description

(See the definition of the Soldier class)

The AirUnit class is a derived class derived from the Soldier class. The AirUnit will be used to create Air unit such as a Soldier to fight on the battlefield i.e. an air WarTheatre.

Note

This class is used to do the calculations for a Soldier and to instantiate a Soldier object.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 AirUnit()

```
AirUnit::AirUnit (
            int powerRating )
```

Constructs AirUnit object, calling constructor of parent Soldier.

Author

Luke Lawson (u21433811)

Parameters

powerRating	The powerRating of the particular unit as per factory's cost (higher cost -> higher power)

Warning

The powerRating must be greater than zero.

3.3.3 Member Function Documentation

3.3.3.1 calculateAirDefense()

```
int AirUnit::calculateAirDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's AirDefence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of unit

Note

The returned value is a random integer value.

Implements Soldier.

3.3.3.2 calculateAirOffense()

```
int AirUnit::calculateAirOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's AirOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of unit

Note

The returned value is a random integer value.

Implements Soldier.

3.3.3.3 calculateLandDefense()

```
int AirUnit::calculateLandDefense ( )  [virtual]
```

Calculates the LandDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is an AirUnit and not a LandUnit.

Implements Soldier.

3.3.3.4 calculateLandOffense()

```
int AirUnit::calculateLandOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's LandOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of unit

Note

The returned value is a random integer value.

Warning

It returns 0 since this is an AirUnit and not a LandUnit.

Implements Soldier.

3.3.3.5 calculateSeaDefense()

```
int AirUnit::calculateSeaDefense ( )  [virtual]
```

Calculates the SeaDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is an AirUnit and not a SeaUnit.

Implements Soldier.

3.3.3.6 calculateSeaOffense()

```
int AirUnit::calculateSeaOffense ( )  [virtual]
```

Calculates the SeaOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

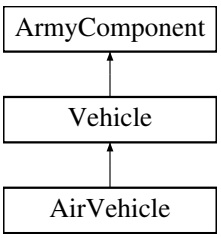
It returns 0 since this is an AirUnit and not a SeaUnit.

Implements Soldier.

3.4 AirVehicle Class Reference

```
#include <AirVehicle.h>
```

Inheritance diagram for AirVehicle:



Public Member Functions

- AirVehicle (int powerRating)

  Constructs AirVehicle object, using powerRating to randomly generate attributes from Normal Dist. (higher power -> better attributes)
- int calculateAirOffense ()

  Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's AirOffence statistic.
- int calculateAirDefense ()

  Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's AirDefence statistic.
- int calculateSeaOffense ()

  Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's SeaOffence statistic.
- int calculateSeaDefense ()

  Calculates the SeaDefence statistic of the vehicle.
- int calculateLandOffense ()

  Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's LandOffence statistic.
- int calculateLandDefense ()

  Calculates the LandDefence statistic of the vehicle.

Additional Inherited Members

3.4.1 Detailed Description

(See the definition of the Vehicle class)

The AirVehicle class is a derived class derived from the Vehicle class. The AirVehicle will be used to create an individual unit (vehicle) which will fight alongside soldier units in the war.

Note

This class is used to do the calculations for a Vehicle and to instantiate a Vehicle object.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 AirVehicle()

```
AirVehicle::AirVehicle (
            int powerRating )
```

Constructs AirVehicle object, using powerRating to randomly generate attributes from Normal Dist. (higher power -> better attributes)

Author

Luke Lawson (u21433811)

Parameters

powerRating	The powerRating of the particular vehicle as per factory's cost (higher cost -> higher power)

Warning

The powerRating cannot be a negative number.

3.4.3 Member Function Documentation

3.4.3.1 calculateAirDefense()

```
int AirVehicle::calculateAirDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's AirDefence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing AirDefence statistic of vehicle

Implements Vehicle.

3.4.3.2 calculateAirOffense()

```
int AirVehicle::calculateAirOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's AirOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing AirOffense statistic of vehicle

Implements Vehicle.

3.4.3.3 calculateLandDefense()

```
int AirVehicle::calculateLandDefense ( )  [virtual]
```

Calculates the LandDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is an AirVehicle and not a LandVehicle.

Implements Vehicle.

3.4.3.4 calculateLandOffense()

```
int AirVehicle::calculateLandOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's Land↩ Offence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of vehicle

Warning

It returns 0 since this is an AirVehicle and not a LandVehicle.

Implements Vehicle.

3.4.3.5 calculateSeaDefense()

```
int AirVehicle::calculateSeaDefense ( )  [virtual]
```

Calculates the SeaDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is an AirVehicle and not a SeaVehicle.

Implements Vehicle.

3.4.3.6 calculateSeaOffense()

```
int AirVehicle::calculateSeaOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's SeaOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaOffense statistic of vehicle

Note

The returned value is a random integer value.

Warning

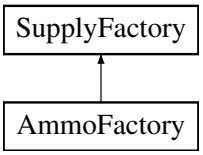
It returns 0 since this is an AirVehicle and not a SeaVehicle.

Implements Vehicle.

3.5 AmmoFactory Class Reference

```
#include <AmmoFactory.h>
```

Inheritance diagram for AmmoFactory:



Public Member Functions

- AmmoFactory (int budget, std::string type)

  Class constructor for the AmmoFactory to initialize the budget.
- Supply * makeSupply (int quantity)

  Creates ammo supplies by creating a new AmmoSupply product.

Additional Inherited Members

3.5.1 Detailed Description

The AmmoFactory class is a derived class derived from the SupplyFactory class (See the definition of the SupplyFactory class)

The AmmoFactory will be used to create Ammo Supplies for the Country's Armies. The AmmoFactory has a method "makeSupply()" which will create the AmmoSuppply object.

Note

This class is ONLY used to create AmmoSupply objects

3.5.2 Constructor & Destructor Documentation

3.5.2.1 AmmoFactory()

```
AmmoFactory::AmmoFactory (
            int budget,
            std::string type )
```

Class constructor for the AmmoFactory to initialize the budget.

Author

Arno Jooste (u21457451)

Parameters

budget	The amount that can be spent to make ammo supplies.
type	The type of the factory.

Warning

"type" has to be "Ammo" and "budget" may not be less than or equal to zero.

3.5.3 Member Function Documentation

3.5.3.1 makeSupply()

```
Supply * AmmoFactory::makeSupply (
            int quantity )  [virtual]
```

Creates ammo supplies by creating a new AmmoSupply product.

Author

Arno Jooste (u21457451)

Parameters

quantity	The quantity of ammo supplies to be produced by the ammo factory.

Returns

Pointer to newly created AmmoSupply product.

Warning

"quantity" need to be positive.

Note

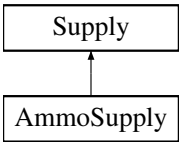
This function may return NULL if the budget has run out.

Implements SupplyFactory.

3.6 AmmoSupply Class Reference

`#include <AmmoSupply.h>`

Inheritance diagram for AmmoSupply:



Public Member Functions

- AmmoSupply (int factoryLevel, int quantity)

  Constructor for AmmoSupply class to specify the factory level and quantity that will be produced.
- int getAmmoBonus ()

  Getter for the ammo bonus member variable.
- void setAmmoBonus (int bonus)

  Setter for the ammo bonus member variable.

Additional Inherited Members

3.6.1 Detailed Description

This class is derived from the Supply class. This is the actual product created by the AmmoFactory.

- Armies will use ammo supply objects to fight the war.
- If an army's ammo supply runs out they are in great disadvantage.

Note

The ammo supply has a variety of small and large ammo i.e. small bullets and heavy bullets.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 AmmoSupply()

```
AmmoSupply::AmmoSupply (
            int factoryLevel,
            int quantity )
```

Constructor for AmmoSupply class to specify the factory level and quantity that will be produced.

Author

Arno Jooste (21457451)

Parameters

factoryLevel	Specifies the currrent factory level in order to set the multiplier of the bonus.
quantity	Specifies the quentity of ammo supplies to be produced. This amount will be used to calculate the ammoBonus.

Warning

The factoryLevel must be a value greater than the integer value 0.
The quantity must also be a value greater than zero.

3.6.3 Member Function Documentation

3.6.3.1 getAmmoBonus()

```
int AmmoSupply::getAmmoBonus ( )
```

Getter for the ammo bonus member variable.

Author

Arno Jooste (u21457451)

Returns

ammo bonus of type int.

3.6.3.2 setAmmoBonus()

```
void AmmoSupply::setAmmoBonus (
            int bonus )
```

Setter for the ammo bonus member variable.

Author

Arno Jooste (u21457451)

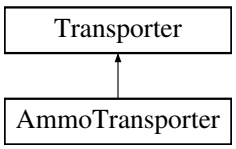
Parameters

bonus	Specfies to which value the ammo bonus will be set.

3.7 AmmoTransporter Class Reference

```
#include <AmmoTransporter.h>
```

Inheritance diagram for AmmoTransporter:



Public Member Functions

- AmmoTransporter ()

  Constructor for the AmmoTransporter class used to instantiate the object.
- virtual ~AmmoTransporter ()

  Destructor for the AmmoTransporter class used to deallocate the dynamic memory used by the member variable corresponderList.
- virtual void notify (Corresponder *corresponder)

  Notify all Corresponder objects in the corresponderList variable.

Additional Inherited Members

3.7.1 Detailed Description

This class is derived from the Trnasporter class.

- We use this class to showcase the transport of ammo supplies which a Country sends to its Army.
- This class notifies all armies of a country that the country has sent ammo supplies to it.

Warning

If this transport line is destroyed by an enemy country, the country will not be able to send any ammo supplies. Thus being in a great disadvantage.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 AmmoTransporter()

```
AmmoTransporter::AmmoTransporter ( )
```

Constructor for the AmmoTransporter class used to instantiate the object.

Author

Reuben Jooste (u21457060)

3.7.2.2 ~AmmoTransporter()

```
AmmoTransporter::~AmmoTransporter ( )  [virtual]
```

Destructor for the AmmoTransporter class used to deallocate the dynamic memory used by the member variable corresponderList.

Author

Reuben Jooste (u21457060)

3.7.3 Member Function Documentation

3.7.3.1 notify()

```
void AmmoTransporter::notify (
            Corresponder * corresponder )  [virtual]
```

Notify all Corresponder objects in the corresponderList variable.

Author

Reuben Jooste (u21457060)

Parameters

corresponder	pointer to the Corresponder in which a changed has happened.

Note

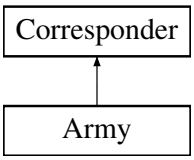
If the parameter is NULL the function would simply throw an exception and terminate.

Implements Transporter.

3.8 Army Class Reference

`#include <Army.h>`

Inheritance diagram for Army:



Public Member Functions

- Army (std::vector< ArmyComponent * > *battalions, std::vector< ArmyComponent * > *individuals, std::vector< Supply * > *supplies, std::string type)

  Constructor to initialise an army object with ArmyComponents, Supplies and a type.
- void applyStrategyBonus ()

  Function to apply bonus to the army's BattleStatistics.
- void recuperate ()

  Function to use medical supplies to replenish army's morale/hp.
- void addNewAmmoSupplies (AmmoSupply *ammo)

  Function to add to Army's available ammo supplies.
- void addNewMedicalSupplies (MedicalSupply *)

  Function to add to Army's available medical supplies.
- void changeStrategy (std::string newStrat)

  Function to change an Army's current strategy.
- void setBattleField (WarTheatre *theatre)

  Function move this Army into a War Theatre.
- void attackTransport (Country *country)

  Function for an Army to attack a Country's.
- std::string getType ()

  Function to change an Army's current strategy.
- BattleStatistics * getBattleStatistics ()

  a function to get the BattleStatistics of a army to alter the statistics
- void setName (std::string Name)

  a function to set the name of the army
- std::string getName ()

  a function to get the name of the army

Additional Inherited Members

3.8.1 Detailed Description

The Army class is a big component in the War since we use this class to keep track of the army's supplies (AmmoSupply & MedicalSupply).

- We also use this class to fight against other Countries.
- We are also able to attack the enemy's Transport which will disable the enemy's transport lines. Thus they won't be able to send supplies to their armies, giving our Army a slight advantage.

Note

This class has vectors for the ammo and medical supplies to separate the two supplies such that we can keep track of the size for both supplies.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Army()

```
Army::Army (
            std::vector< ArmyComponent * > * battalions,
            std::vector< ArmyComponent * > * individuals,
            std::vector< Supply * > * supplies,
            std::string type )
```

Constructor to initialise an army object with ArmyComponents, Supplies and a type.

Author

Luke Lawson (u21433811)

Parameters

battalions	pointer to vector of ArmyComponent pointers representing battalions
individuals	pointer to vector of ArmyComponent pointers representing individuals (vehicles or units)
supplies	pointer to vector of Supply pointers for Ammo and Meds
type	string type of the army. Land, Air or Sea

Warning

To instantiate the Army object, the battalions, individuals and suppplies need to be already defined.

Note

The type is used to specify which army we will instantiate i.e. in which theatre the army will be fighting (Land, Air or Sea).

3.8.3 Member Function Documentation

3.8.3.1 addNewAmmoSupplies()

```
void Army::addNewAmmoSupplies (
            AmmoSupply * ammo )
```

Function to add to Army's available ammo supplies.

Author

Luke Lawson (u21433811)

Parameters

ammo	pointer to AmmoSupply object to be added

3.8.3.2 addNewMedicalSupplies()

```
void Army::addNewMedicalSupplies (
            MedicalSupply * meds )
```

Function to add to Army's available medical supplies.

Author

Luke Lawson (u21433811)

Parameters

ammo	pointer to MedicalSupply object to be added

3.8.3.3 applyStrategyBonus()

```
void Army::applyStrategyBonus ( )
```

Function to apply bonus to the army's BattleStatistics.

Author

Luke Lawson (u21433811)

3.8.3.4 attackTransport()

```
void Army::attackTransport (
            Country * country )
```

Function for an Army to attack a Country's.

Author

Luke Lawson (u21433811)

Parameters

country	pointer to Country whose Transport lines are to be attacked

3.8.3.5 changeStrategy()

```
void Army::changeStrategy (
            std::string newStrat )
```

Function to change an Army's current strategy.

Author

Luke Lawson (u21433811)

Parameters

newStart	string represening the new strategy to adopt

3.8.3.6 getBattleStatistics()

```
BattleStatistics * Army::getBattleStatistics ( )
```

a function to get the BattleStatistics of a army to alter the statistics

Author

Jonelle Coertze (u21446271)

Returns

pointer to a BattleStatistics object

3.8.3.7 getName()

```
std::string Army::getName ( )
```

a function to get the name of the army

Author

Jonelle Coertze (u21446271)

Returns

a string giving the army's name

3.8.3.8 getType()

```
std::string Army::getType ( )
```

Function to change an Army's current strategy.

Author

Luke Lawson (u21433811)

Returns

string representing the army's type (Air, Land or Sea)

3.8.3.9 recuperate()

```
void Army::recuperate ( )
```

Function to use medical supplies to replenish army's morale/hp.

Author

Luke Lawson (u21433811)

Note

This function is used to heal our army by using the medical supplies.

3.8.3.10 setBattleField()

```
void Army::setBattleField (
            WarTheatre * theatre )
```

Function move this Army into a War Theatre.

Author

Luke Lawson (u21433811)

Parameters

theatre	pointer to War Theatre to be added to

Warning

The specified WarTheatre needs to be of the same type as the Army object. We cannot let Land armies fight in an Air theatre.

3.8.3.11 setName()

```
void Army::setName (
            std::string Name )
```

a function to set the name of the army

Author

Jonelle Coertze (u21446271)

Parameters

Name	string to set the name of the Army

3.9 ArmyBuilder Class Reference

`#include <ArmyBuilder.h>`

Public Member Functions

- ArmyBuilder (std::string type, std::vector< UnitFactory * > *unitFactories, std::vector< SupplyFactory * > *supplyFactories)

  Class constructor used to instantiate the object and initialize the type member variable.
- std::vector< ArmyComponent * > * createIndividuals ()

  Function to create individual army components (soldiers or vehicles)
- std::vector< ArmyComponent * > * buildBattalions ()

  Function to create battalions which consist out of other battalions, soldiers or vehicles.
- std::vector< Supply * > * determineSupplies ()

  Function tp create supplies for the army.
- Army * putArmyTogether ()

  This function is used to merge the different parts (objects) of an army into one Army object.
- Army * getArmy ()

  Function to receive the newly constructed Army object.

- std::vector< ArmyComponent * > * getIndividuals ()

  This function is used to return the vector of individuals which was create by the createIndividuals() method.
- std::vector< ArmyComponent * > * getBattalions ()

  This function is used to return the vector of battalions which was create by the buildBattalions() method.
- std::vector< Supply * > * getSupplies ()

  This function is used to return the vector of supplies which was create by the determineSupplies() method.
- void setIndividuals (std::vector< ArmyComponent * > *individuals)

  This function will set the member variable individuals in order to keep track of the individuals created.
- void setBattalions (std::vector< ArmyComponent * > *battalions)

  This function will set the member variable battalions in order to keep track of the battalions created.
- void setSupplies (std::vector< Supply * > *supplies)

  This function will set the member variable supplies in order to keep track of the supplies created.

3.9.1 Detailed Description

The ArmyBuilder class is a big component in the War since we use this class build each Country's army.

- This class creates the different parts of the army such as the Battalions, Individuals (Soldiers or Vehicles) and the army's supplies (AmmoSupplies or MedicalSupplies).
- The builder makes use of the country's factories to build/train the different supplies/units.
- If the country's factory do not have the budget to create even just a single unit/supply then the builder will not be able to build the army.

Warning

Without this class the Country will not be able to raise an army.

Note

This class has vectors for the ammo and medical supplies to separate the two supplies such that we can keep track of the size for both supplies.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 ArmyBuilder()

```
ArmyBuilder::ArmyBuilder (
            std::string type,
            std::vector< UnitFactory * > * unitFactories,
            std::vector< SupplyFactory * > * supplyFactories )
```

Class constructor used to instantiate the object and initialize the type member variable.

Author

Reuben Jooste (u21457060)

Parameters

type	Specifies which type of army builder this class will construct
unitFactories	UnitFactories to choose from for creating the army
supplyFactories	SupplyFactories to choose from for creating the supplies

Warning

To instantiate the ArmyBuilder object, unit factories and supply factories vectors need be already defined.

Note

The type is used to specify which armies this builder will instantiate i.e. Land, Air or Sea.

3.9.3 Member Function Documentation

3.9.3.1 buildBattalions()

```
std::vector< ArmyComponent * > * ArmyBuilder::buildBattalions ( )
```

Function to create battalions which consist out of other battalions, soldiers or vehicles.

Author

Reuben Jooste (u21457060)

Returns

Pointer to a list of pointers to battalions

Note

This function may return NULL if all the unit factories did not have the budget to create a single soldier or vehicle.
This function uses the "createIndividuals()" to build the battalions.

3.9.3.2 createIndividuals()

```
std::vector< ArmyComponent * > * ArmyBuilder::createIndividuals ( )
```

Function to create individual army components (soldiers or vehicles)

Author

Reuben Jooste (u21457060)

Returns

Pointer to a list used for storing pointers to ArmyComponents

Note

This function may return NULL if all the unit factories did not have the budget to create a single soldier or vehicle.

3.9.3.3 determineSupplies()

```
std::vector< Supply * > * ArmyBuilder::determineSupplies ( )
```

Function tp create supplies for the army.

Author

Reuben Jooste (u21457060)

Returns

Pointer to a list of pointers of Suppply objects (AmmoSupply or MedicalSupply)

Note

This function may return NULL if all the suppply factories did not have the budget to create a single ammo or medical supply.

3.9.3.4 getArmy()

```
Army * ArmyBuilder::getArmy ( )
```

Function to receive the newly constructed Army object.

Author

Reuben Jooste (u21457060)

Returns

Member variable of constructed Army

3.9.3.5 getBattalions()

```
std::vector< ArmyComponent * > * ArmyBuilder::getBattalions ( )
```

This function is used to return the vector of battalions which was create by the buildBattalions() method.

Author

Reuben Jooste (u21457060)

Returns

Return vector of battalion ArmyComponents

Warning

If this function returns NULL then the constructor initialised the member variable as NULL. This happens if a builder's constructor is called with the second argument as NULL.

3.9.3.6 getIndividuals()

```
std::vector< ArmyComponent * > * ArmyBuilder::getIndividuals ( )
```

This function is used to return the vector of individuals which was create by the createIndividuals() method.

Author

Reuben Jooste (u21457060)

Returns

Return vector of individual ArmyComponents

Warning

If this function returns NULL then the constructor initialised the member variable as NULL. This happens if a builder's constructor is called with the second argument as NULL.

3.9.3.7 getSupplies()

```
std::vector< Supply * > * ArmyBuilder::getSupplies ( )
```

This function is used to return the vector of supplies which was create by the determineSupplies() method.

Author

Reuben Jooste (u21457060)

Returns

Return vector of supplies

Warning

If this function returns NULL then the constructor initialised the member variable as NULL. This happens if a builder's constructor is called with the third argument as NULL.

3.9.3.8 putArmyTogether()

```
Army * ArmyBuilder::putArmyTogether ( )
```

This function is used to merge the different parts (objects) of an army into one Army object.

Author

Reuben Jooste (u21457060)

Returns

Completed Army object

Note

This function returns a Army object by call the Army class's constructor

3.9.3.9 setBattalions()

```
void ArmyBuilder::setBattalions (
            std::vector< ArmyComponent * > * battalions )
```

This function will set the member variable battalions in order to keep track of the battalions created.

Author

Reuben Jooste (u21457060)

Parameters

battalions	The parameter is used to set our member variable by making a deep copy of it.

Warning

If the parameter is NULL the member variable will NOT be set.
This function overwrites the current member variable because this function should only be called inside the buildBattalions() method.

3.9.3.10 setIndividuals()

```
void ArmyBuilder::setIndividuals (
            std::vector< ArmyComponent * > * individuals )
```

This function will set the member variable individuals in order to keep track of the individuals created.

Author

Reuben Jooste (u21457060)

Parameters

individuals	The parameter is used to set our member variable by making a deep copy of it.

Warning

If the parameter is NULL the member variable will NOT be set.
This function overwrites the current member variable because this function should only be called inside the buildBattalions() method and the createIndividuals() method.

3.9.3.11 setSupplies()

```
void ArmyBuilder::setSupplies (
            std::vector< Supply * > * supplies )
```

This function will set the member variable supplies in order to keep track of the supplies created.

Author

Reuben Jooste (u21457060)

Parameters

supplies	The parameter is used to set our member variable by making a deep copy of it.

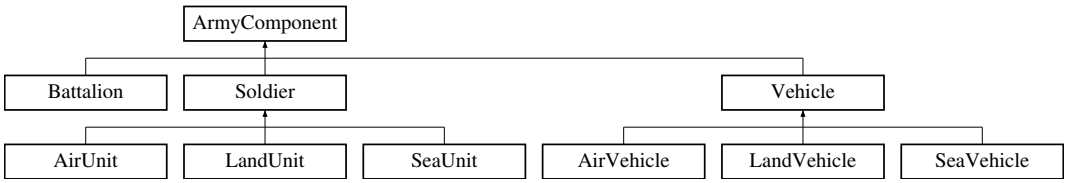
Warning

If the parameter is NULL the member variable will NOT be set.
This function overwrites the current member variable because this function should only be called inside the determineSupplies() method.

3.10 ArmyComponent Class Reference

`#include <ArmyComponent.h>`

Inheritance diagram for ArmyComponent:



Public Member Functions

- virtual int calculateAirOffense ()=0

  Determines AirOffence statistic of the ArmyComponent. Implemented in derived classes.
- virtual int calculateAirDefense ()=0

  Determines AirDefence statistic of the ArmyComponent. Implemented in derived classes.
- virtual int calculateSeaOffense ()=0

  Determines SeaOffence statistic of the ArmyComponent. Implemented in derived classes.
- virtual int calculateSeaDefense ()=0

  Determines SeaDefence statistic of the ArmyComponent. Implemented in derived classes.
- virtual int calculateLandOffense ()=0

  Determines LandOffence statistic of the ArmyComponent. Implemented in derived classes.
- virtual int calculateLandDefense ()=0

  Determines LandDefence statistic of the ArmyComponent. Implemented in derived classes.
- virtual void addMember (ArmyComponent *newMember)=0

  Interface function for adding objects to composite objects (Battalion)

3.10.1 Detailed Description

This class is just an abstract class for the following classes:

- Soldier
- Vehicle
- Battalion

We use this class to instantiate different components of an army such that those components can be used to help fight the on going war.

3.10.2 Member Function Documentation

3.10.2.1 addMember()

```
virtual void ArmyComponent::addMember (
            ArmyComponent * newMember )  [pure virtual]
```

Interface function for adding objects to composite objects (Battalion)

Author

Luke Lawson (u21433811)

Parameters

newMember	pointer to the ArmyComponent to be added to the Battalion (Composite)

Implemented in Vehicle, Soldier, and Battalion.

3.10.2.2 calculateAirDefense()

```
virtual int ArmyComponent::calculateAirDefense ( )  [pure virtual]
```

Determines AirDefence statistic of the ArmyComponent. Implemented in derived classes.

Author

Luke Lawson (u21433811)

Returns

int representing value of the AirDefence statistic of the ArmyComponent

Implemented in LandVehicle, SeaVehicle, LandUnit, SeaUnit, AirUnit, AirVehicle, Vehicle, Soldier, and Battalion.

3.10.2.3 calculateAirOffense()

```
virtual int ArmyComponent::calculateAirOffense ( )  [pure virtual]
```

Determines AirOffence statistic of the ArmyComponent. Implemented in derived classes.

Author

Luke Lawson (u21433811)

Returns

int representing value of the AirOffence statistic of the ArmyComponent

Implemented in LandVehicle, SeaVehicle, AirVehicle, LandUnit, AirUnit, SeaUnit, Vehicle, Soldier, and Battalion.

3.10.2.4 calculateLandDefense()

```
virtual int ArmyComponent::calculateLandDefense ( )  [pure virtual]
```

Determines LandDefence statistic of the ArmyComponent. Implemented in derived classes.

Author

Luke Lawson (u21433811)

Returns

int representing value of the LandDefence statistic of the ArmyComponent

Implemented in LandVehicle, SeaVehicle, AirUnit, LandUnit, AirVehicle, SeaUnit, Vehicle, Soldier, and Battalion.

3.10.2.5 calculateLandOffense()

```
virtual int ArmyComponent::calculateLandOffense ( )  [pure virtual]
```

Determines LandOffence statistic of the ArmyComponent. Implemented in derived classes.

Author

Luke Lawson (u21433811)

Returns

int representing value of the LandOffence statistic of the ArmyComponent

Implemented in LandVehicle, LandUnit, SeaVehicle, AirUnit, AirVehicle, SeaUnit, Vehicle, Soldier, and Battalion.

3.10.2.6 calculateSeaDefense()

```
virtual int ArmyComponent::calculateSeaDefense ( )  [pure virtual]
```

Determines SeaDefence statistic of the ArmyComponent. Implemented in derived classes.

Author

Luke Lawson (u21433811)

Returns

int representing value of the SeaDefence statistic of the ArmyComponent

Implemented in LandVehicle, LandUnit, SeaVehicle, AirUnit, AirVehicle, SeaUnit, Vehicle, Soldier, and Battalion.

3.10.2.7 calculateSeaOffense()

```
virtual int ArmyComponent::calculateSeaOffense ( )  [pure virtual]
```

Determines SeaOffence statistic of the ArmyComponent. Implemented in derived classes.

Author

Luke Lawson (u21433811)

Returns

int representing value of the SeaOffence statistic of the ArmyComponent

Implemented in LandVehicle, LandUnit, SeaVehicle, AirUnit, AirVehicle, SeaUnit, Vehicle, Soldier, and Battalion.

3.11 ArmyDirector Class Reference

```
#include <ArmyDirector.h>
```

Public Member Functions

- ArmyDirector (ArmyBuilder *builder)
  Constructor for the ArmyDirector class to instantiate the object and set the member variable.
- void constructArmy ()
  This function is used to construct an army which will be used by the Country to fight the war.

3.11.1 Detailed Description

The ArmyDirector class is a big component in the War since we use this class to command the ArmyBuilder to build the Country's army.

- Each country must have a director otherwise the country will not be able to build any armies.
- The director receives the command "raise army" from the millitary commander and then only instructs the ArmyBuilder to build the army.

  Warning

  If the member variable (builder) is NULL we cannot build the army.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 ArmyDirector()

```
ArmyDirector::ArmyDirector (
            ArmyBuilder * builder )
```

Constructor for the ArmyDirector class to instantiate the object and set the member variable.

Author

Reuben Jooste (u21457060)

Parameters

builder	pointer to an existing ArmyBuilder object used to set this class' member variable

Warning

The parameter MUST NOT BE NULL otherwise the it is impossible to build an army for a Country.

3.11.3 Member Function Documentation

3.11.3.1 constructArmy()

```
void ArmyDirector::constructArmy ( )
```

This function is used to construct an army which will be used by the Country to fight the war.

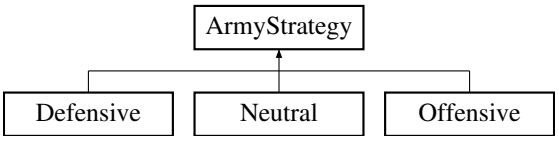
Author

Reuben Jooste (u21457060)

3.12 ArmyStrategy Class Reference

```
#include <ArmyStrategy.h>
```

Inheritance diagram for ArmyStrategy:



Public Member Functions

- virtual void applyStrategyBonus (BattleStatistics, Battalion ∗)

  Applies desired bonuses to BattleStatistics.

3.12.1 Detailed Description

This class is only used to specify the strategy an Army is using.

3.12.2 Member Function Documentation

3.12.2.1 applyStrategyBonus()

```
void ArmyStrategy::applyStrategyBonus (
            BattleStatistics in,
            Battalion * inBattalion )  [virtual]
```

Applies desired bonuses to BattleStatistics.

Author

Thomas Blendulf (u21446131)

Parameters

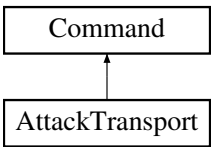
BattleStatistics	passes in the BattleStatistics to be edited.
Battalion	passes in the Battalion to calculate base statistics to be editted.

Reimplemented in Defensive, Neutral, and Offensive.

3.13 AttackTransport Class Reference

```
#include <AttackTransport.h>
```

Inheritance diagram for AttackTransport:



Public Member Functions

- void setTransport (Country *)

  sets the Transporter to be attacked by the army.
- void execute ()

  executes the attack on the Transporter.

Public Attributes

- Country * transport

Additional Inherited Members

3.13.1 Member Function Documentation

3.13.1.1 execute()

```
void AttackTransport::execute ( )  [virtual]
```

executes the attack on the Transporter.

Author

Thomas Blendulf(u21446131)

Implements Command.

3.13.1.2 setTransport()

```
void AttackTransport::setTransport (
            Country * in )
```

sets the Transporter to be attacked by the army.

Author

Thomas Blendulf(u21446131)

Parameters

Transporter	containing transporter target to be updated to.

3.13.2 Member Data Documentation

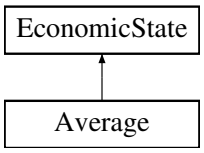
3.13.2.1 transport

```
Country* AttackTransport::transport
```

3.14 Average Class Reference

`#include <Average.h>`

Inheritance diagram for Average:



Public Member Functions

- int decideMyTurn (Country *country)

  randomly decide what a country can do during their turn

3.14.1 Member Function Documentation

3.14.1.1 decideMyTurn()

```
int Average::decideMyTurn (
            Country * country )  [virtual]
```

randomly decide what a country can do during their turn

Author

Jonelle Coertze (u21446271)

Parameters

country	pointer to an existing Country object to have access to the country's army and alliences

Returns

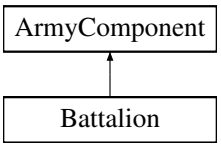
an int corresponding with the decision

Implements EconomicState.

3.15 Battalion Class Reference

`#include <Battalion.h>`

Inheritance diagram for Battalion:



Public Member Functions

- Battalion ()
- int calculateAirOffense ()

  Traverses members to get the sum of the AirOffence statistics. This is the statistic value for the Battalio.
- int calculateAirDefense ()

  Traverses members to get the sum of the AirDefence statistics. This is the statistic value for the Battalion.
- int calculateSeaOffense ()

  Traverses members to get the sum of the SeaOffence statistics. This is the statistic value for the Battalion.
- int calculateSeaDefense ()

  Traverses members to get the sum of the SeaDefence statistics. This is the statistic value for the Battalion.
- int calculateLandOffense ()

  Traverses members to get the sum of the LandOffence statistics. This is the statistic value for the Battalion.
- int calculateLandDefense ()

  Traverses members to get the sum of the LandDefence statistics. This is the statistic value for the Battalion.
- void addMember (ArmyComponent ∗newMember)

  Adds ArmyComponent to this Composite object.

3.15.1 Constructor & Destructor Documentation

3.15.1.1 Battalion()

```
Battalion::Battalion ( )
```

3.15.2 Member Function Documentation

3.15.2.1 addMember()

```
void Battalion::addMember (
            ArmyComponent * newMember )  [virtual]
```

Adds ArmyComponent to this Composite object.

Author

Luke Lawson (u21433811)

Parameters

newMember	new ArmyComponent pointer to add to members vector

Implements ArmyComponent.

3.15.2.2 calculateAirDefense()

```
int Battalion::calculateAirDefense ( )  [virtual]
```

Traverses members to get the sum of the AirDefence statistics. This is the statistic value for the Battalion.

Author

Luke Lawson (u21433811)

Returns

int value for AirDefence statistic of Battalion

Implements ArmyComponent.

3.15.2.3 calculateAirOffense()

```
int Battalion::calculateAirOffense ( )  [virtual]
```

Traverses members to get the sum of the AirOffence statistics. This is the statistic value for the Battalio.

Author

Luke Lawson (u21433811)

Returns

int value for AirOffence statistic of Battalion

Implements ArmyComponent.

3.15.2.4 calculateLandDefense()

```
int Battalion::calculateLandDefense ( )  [virtual]
```

Traverses members to get the sum of the LandDefence statistics. This is the statistic value for the Battalion.

Author

Luke Lawson (u21433811)

Returns

int value for LandDefence statistic of Battalion

Implements ArmyComponent.

3.15.2.5 calculateLandOffense()

```
int Battalion::calculateLandOffense ( )  [virtual]
```

Traverses members to get the sum of the LandOffence statistics. This is the statistic value for the Battalion.

Author

Luke Lawson (u21433811)

Returns

int value for LandOffence statistic of Battalion

Implements ArmyComponent.

3.15.2.6 calculateSeaDefense()

```
int Battalion::calculateSeaDefense ( )  [virtual]
```

Traverses members to get the sum of the SeaDefence statistics. This is the statistic value for the Battalion.

Author

Luke Lawson (u21433811)

Returns

int value for SeaDefence statistic of Battalion

Implements ArmyComponent.

3.15.2.7 calculateSeaOffense()

```
int Battalion::calculateSeaOffense ( )  [virtual]
```

Traverses members to get the sum of the SeaOffence statistics. This is the statistic value for the Battalion.

Author

Luke Lawson (u21433811)

Returns

int value for SeaOffence statistic of Battalion

Implements ArmyComponent.

3.16 BattleStatistics Class Reference

```
#include <BattleStatistics.h>
```

Public Member Functions

- int getAirAttack ()
- int getAirDefence ()
- int getLandAttack ()
- int getLandDefence ()
- int getSeaAttack ()
- int getSeaDefence ()
- int getMorale ()
- int getAvailableAmmo ()
- int getMedical ()
- void setAirAttack (int in)
- void setAirDefence (int in)
- void setLandAttack (int in)
- void setLandDefence (int in)
- void setSeaAttack (int in)
- void setSeaDefence (int in)
- void setMorale (int in)
- void setAvailableAmmo (int in)
- void setMedical (int in)

Friends

- class Defensive
- class Neutral
- class Offensive

3.16.1 Member Function Documentation

3.16.1.1 getAirAttack()

```
int BattleStatistics::getAirAttack ( )
```

3.16.1.2 getAirDefence()

```
int BattleStatistics::getAirDefence ( )
```

3.16.1.3 getAvailableAmmo()

```
int BattleStatistics::getAvailableAmmo ( )
```

3.16.1.4 getLandAttack()

```
int BattleStatistics::getLandAttack ( )
```

3.16.1.5 getLandDefence()

```
int BattleStatistics::getLandDefence ( )
```

3.16.1.6 getMedical()

```
int BattleStatistics::getMedical ( )
```

3.16.1.7 getMorale()

```
int BattleStatistics::getMorale ( )
```

3.16.1.8 getSeaAttack()

```
int BattleStatistics::getSeaAttack ( )
```

3.16.1.9 getSeaDefence()

```
int BattleStatistics::getSeaDefence ( )
```

3.16.1.10 setAirAttack()

```
void BattleStatistics::setAirAttack (
            int in )
```

3.16.1.11 setAirDefence()

```
void BattleStatistics::setAirDefence (
            int in )
```

3.16.1.12 setAvailableAmmo()

```
void BattleStatistics::setAvailableAmmo (
            int in )
```

3.16.1.13 setLandAttack()

```
void BattleStatistics::setLandAttack (
            int in )
```

3.16.1.14 setLandDefence()

```
void BattleStatistics::setLandDefence (
            int in )
```

3.16.1.15 setMedical()

```
void BattleStatistics::setMedical (
            int in )
```

3.16.1.16 setMorale()

```
void BattleStatistics::setMorale (
            int in )
```

3.16.1.17 setSeaAttack()

```
void BattleStatistics::setSeaAttack (
            int in )
```

3.16.1.18 setSeaDefence()

```
void BattleStatistics::setSeaDefence (
            int in )
```

3.16.2 Friends And Related Function Documentation

3.16.2.1 Defensive

```
friend class Defensive  [friend]
```

3.16.2.2 Neutral

```
friend class Neutral  [friend]
```

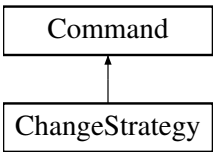
3.16.2.3 Offensive

```
friend class Offensive  [friend]
```

3.17 ChangeStrategy Class Reference

`#include <ChangeStrategy.h>`

Inheritance diagram for ChangeStrategy:



Public Member Functions

- ChangeStrategy ()
- void setStrategy (std::string)

  sets the strategy to be executed by the commmand pattern.
- void execute ()

  calls setStrategy in the stored Army.

Public Attributes

- std::string newStrategy

Additional Inherited Members

3.17.1 Constructor & Destructor Documentation

3.17.1.1 ChangeStrategy()

`ChangeStrategy::ChangeStrategy ( )`

3.17.2 Member Function Documentation

3.17.2.1 execute()

`void ChangeStrategy::execute ( )  [virtual]`

calls setStrategy in the stored Army.

Author

Thomas Blendulf(u21446131)

Implements Command.

3.17.2.2 setStrategy()

```
void ChangeStrategy::setStrategy (
            std::string in )
```

sets the strategy to be executed by the commmand pattern.

Author

Thomas Blendulf(u21446131)

Parameters

string	containing state to be updated to.

3.17.3 Member Data Documentation

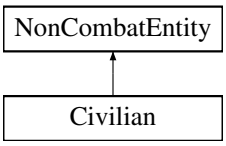
3.17.3.1 newStrategy

```
std::string ChangeStrategy::newStrategy
```

3.18 Civilian Class Reference

```
#include <Civilian.h>
```

Inheritance diagram for Civilian:



Public Member Functions

- Civilian ()
  A default constructor that sets the designation to a citizen.
- Civilian (std::string Designation)
  a parameterized constructor to craete a new Civilian with a specific designation
- NonCombatEntity * clone ()
  clone the current Civilian
- void setDesignation (std::string Designation)
  set the attribute designation, which can be a refugee or a citizen
- std::string getDesignation ()
  get the attribute designation, which can be a refugee or a citizen

3.18.1 Constructor & Destructor Documentation

3.18.1.1 Civilian() [1/2]

```
Civilian::Civilian ( )
```

A default constructor that sets the designation to a citizen.

Author

Jonelle Coertze (u21446271)

3.18.1.2 Civilian() [2/2]

```
Civilian::Civilian (
            std::string Designation )
```

a parameterized constructor to craete a new Civilian with a specific designation

Author

Jonelle Coertze (u21446271)

Parameters

Designation	string to indicate if a civilian is a refugee or a citizen

3.18.2 Member Function Documentation

3.18.2.1 clone()

```
NonCombatEntity * Civilian::clone ( )  [virtual]
```

clone the current Civilian

Author

Jonelle Coertze (u21446271)

Returns

a pointer to the cloned/new NonCombatEntity : Civilian

Implements NonCombatEntity.

3.18.2.2 getDesignation()

```
std::string Civilian::getDesignation ( )
```

get the attribute designation, which can be a refugee or a citizen

Author

Jonelle Coertze (u21446271)

Returns

a string to indicate if a civilian is a refugee or a citizen

3.18.2.3 setDesignation()

```
void Civilian::setDesignation (
            std::string Designation )
```

set the attribute designation, which can be a refugee or a citizen

Author

Jonelle Coertze (u21446271)

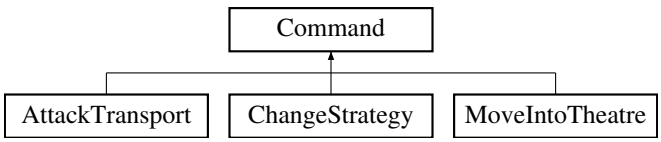
Parameters

Designation	string to indicate if a civilian is a refugee or a citizen

3.19 Command Class Reference

```
#include <Command.h>
```

Inheritance diagram for Command:



Public Member Functions

- Command ()

- void setArmy (Army ∗)
  sets the army to be executed on.
- Army ∗ getArmy ()
  returns the currently stored army.
- virtual void execute ()=0

Protected Attributes

- Army ∗ army

3.19.1 Constructor & Destructor Documentation

3.19.1.1 Command()

```
Command::Command ( )
```

3.19.2 Member Function Documentation

3.19.2.1 execute()

```
virtual void Command::execute ( )  [pure virtual]
```

Implemented in ChangeStrategy, AttackTransport, and MoveIntoTheatre.

3.19.2.2 getArmy()

```
Army * Command::getArmy ( )
```

returns the currently stored army.

Author

Thomas Blendulf(u21446131)

Returns

Army∗.

3.19.2.3 setArmy()

```
void Command::setArmy (
            Army * in )
```

sets the army to be executed on.

Author

Thomas Blendulf(u21446131)

Parameters

Army	containing army to be updated to.

3.19.3 Member Data Documentation

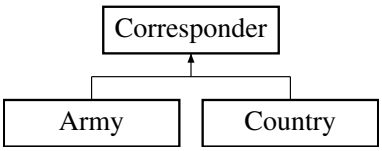
3.19.3.1 army

```
Army* Command::army  [protected]
```

3.20 Corresponder Class Reference

```
#include <Corresponder.h>
```

Inheritance diagram for Corresponder:



Public Member Functions

- void regToTransport (Transporter *ammoTransportLine, Transporter *medTransportLine)

  This function is used to register the Country to the ammo and medical transport lines.

Protected Attributes

- Transporter * medicalTransportLine
- Transporter * ammoTransportLine

3.20.1 Detailed Description

This class is used to register a Country to its transport lines when the country object is instantiated.

Warning

The transport lines are the only way a Country can transport supplies (AmmoSupply & MedicalSupply) to its armies.

3.20.2 Member Function Documentation

3.20.2.1 regToTransport()

```
void Corresponder::regToTransport (
            Transporter * ammoTransportLine,
            Transporter * medTransportLine )
```

This function is used to register the Country to the ammo and medical transport lines.

Parameters

ammoTransportLine	The AmmoTransporter for the country. We use this transport line to send ammo supplies to the country's army.
medTransportLine	The MedicalTransporter for the country. We use this transport line to send medical supplies to the country's army.

Note

It is possible to pass in NULL values for both argumenys but if done so then the Country would be in a great DISADVANTAGE to the other enemy countries.

3.20.3 Member Data Documentation

3.20.3.1 ammoTransportLine

```
Transporter* Corresponder::ammoTransportLine  [protected]
```

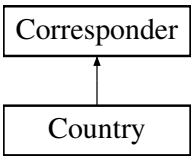
3.20.3.2 medicalTransportLine

```
Transporter* Corresponder::medicalTransportLine  [protected]
```

3.21 Country Class Reference

```
#include <Country.h>
```

Inheritance diagram for Country:



Public Member Functions

- Country (std::string ecoState, std::string name)

  Constructor to initialise a Country based on its starting EconomicState.
- ~Country ()

  Destructor to deallocate any dynamic memory involved.
- std::string getName ()

  Getter for the Country name.
- bool isSurrendered ()

  Gets whether Country has surrender from the war.
- void earnGDP (double gdpEarned)

  Function to increase Country GDP and manage change of economic state.
- void spendGDP (double gdpSpent)

  Function to decrease Country GDP and manage change of economic state.
- void takeTurn (War *currWar)

  Function to decide and enact the Country's play for a turn.
- void formAlliance ()

  Function to add random Country from neutral to this Country's alliance.
- void raiseArmy ()

  Function to call appropriate creational structures to create an army and add it to Country's armies.
- void upgradeUnitFactory ()

  Function to upgrade a Country's Unit Factory such to produce better military units.
- void upgradeSupplyFactory ()

  Function to upgrade a Country's Supply Factory such to produce better/greater quantity of medical supplies and ammo.
- void enterArmyIntoTheatre (War *war)

  Function to use MilitaryCommander to send an Army into a WarTheatre.
- void changeArmyStrategy ()

  Function to use MilitaryCommander to change the Army's strategy.
- void attackTransport ()

  Function to use MilitaryCommander to instruct an army to attack another Country's transport.
- void surrender ()

  Function to cause Country to surrender and withdraw from the War and alliance.
- void destroyTransport ()

  Function to set this Country's Transport to NULL.
- void sendSupplies (AmmoSupply *ammo, MedicalSupply *meds)

  Function to send/distribute supplies to a Country's armies.
- AmmoSupply * getNewAmmoSupply ()

  Function to get the newly created supply such that we know which supply to send to the transport line.
- MedicalSupply * getNewMedicalSupply ()

  Function to get the newly created supply such that we know which supply to send to the transport line.
- void setNewAmmoSupplies (AmmoSupply *newAmmoSupply)

  Function to set the member variable to store the newly created ammo supply.
- void setNewMedicalSupplies (MedicalSupply *newMedicalSupply)

  Function to set the member variable to store the newly created medical supply.
- Army * getArmy ()

  Function to return the army variable of this Country class.

Static Public Attributes

- static std::vector< Country * > alliance1
- static std::vector< Country * > alliance2
- static std::vector< Country * > neutral

Additional Inherited Members

3.21.1 Detailed Description

The Country class is a big component in the War since we use this class to fight the on going war. Countries will be able to form an alliance with other countries and fight together againts other countries.

- For this War there will only be three alliance groups: group1, group2 and group3. Group 1 will fight against Countries in Group 2 and the Countries in Group 3 will "watch" the on going war and may decide (at a later stage) which alliance group they want to join.
- Each Country will also have different factories for Supplies and to create Soldiers/Vehicles. But creating the units and supplies will cost them money therefore each Country will need to keep track of their budget (gdp) in order to determine if they can afford to create more supplies/units.
- If a Country starts to realise that it is losing the war then it has the option of surrendering to the enemy Country.
- The Country also has the option to upgrade its different factories but this would only be possible if the Country is in a Rich EconomicState.
- Countries will have military commanders which will issue different commands for the Country for example raising an army to prepare for the war.
- Finally the Country will be able to create suppplies and signal the military commander to send the supplies to the Country's transport lines such that it can be transported to the armies of the Country.

Note

This class has vectors for the ammo and medical supplies to separate the two supplies such that we can keep track of the size for both supplies.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 Country()

```
Country::Country (
            std::string ecoState,
            std::string name )
```

Constructor to initialise a Country based on its starting EconomicState.

Author

Luke Lawson (u21433811)

Parameters

ecoState	String of value Rich, Average or Poor
name	the name of the Country

Note

The ecoState can only be one of the specified three (Rich, Average, Poor)

3.21.2.2 ~Country()

```
Country::~Country ( )
```

Destructor to deallocate any dynamic memory involved.

Author

Luke Lawson (u21433811)

3.21.3 Member Function Documentation

3.21.3.1 attackTransport()

```
void Country::attackTransport ( )
```

Function to use MilitaryCommander to instruct an army to attack another Country's transport.

Author

Luke Lawson (u21433811)

3.21.3.2 changeArmyStrategy()

```
void Country::changeArmyStrategy ( )
```

Function to use MilitaryCommander to change the Army's strategy.

Author

Luke Lawson (u21433811)

3.21.3.3 destroyTransport()

```
void Country::destroyTransport ( )
```

Function to set this Country's Transport to NULL.

Author

Luke Lawson (u21433811)

3.21.3.4 earnGDP()

```
void Country::earnGDP (
            double gdpEarned )
```

Function to increase Country GDP and manage change of economic state.

Author

Luke Lawson (u21433811)

Parameters

gdpEarned	double which indicates the amount to increase GDP by

Warning

The parameter must be a positive double value. If we want to decrease the GDP we must use the function "spendGDP()".

3.21.3.5 enterArmyIntoTheatre()

```
void Country::enterArmyIntoTheatre (
            War * war )
```

Function to use MilitaryCommander to send an Army into a WarTheatre.

Author

Luke Lawson (u21433811)

Parameters

war	pointyer to the War the country is currently engaged in

3.21.3.6 formAlliance()

```
void Country::formAlliance ( )
```

Function to add random Country from neutral to this Country's alliance.

Author

Luke Lawson (u21433811)

3.21.3.7 getArmy()

```
Army * Country::getArmy ( )
```

Function to return the army variable of this Country class.

Author

Reuben Jooste (u21457060)

Returns

Returns the army of the Country as a pointer

3.21.3.8 getName()

```
std::string Country::getName ( )
```

Getter for the Country name.

Author

Luke Lawson (u21433811)

Returns

string name of the Country

3.21.3.9 getNewAmmoSupply()

```
AmmoSupply * Country::getNewAmmoSupply ( )
```

Function to get the newly created supply such that we know which supply to send to the transport line.

Author

Reuben Jooste (u21457060)

Returns

The newly created ammo supply

3.21.3.10 getNewMedicalSupply()

```
MedicalSupply * Country::getNewMedicalSupply ( )
```

Function to get the newly created supply such that we know which supply to send to the transport line.

Author

Reuben Jooste (u21457060)

Returns

The newly created medical supply

3.21.3.11 isSurrendered()

```
bool Country::isSurrendered ( )
```

Gets whether Country has surrender from the war.

Author

Luke Lawson (u21433811)

Returns

boolean value of hasSurrendered

3.21.3.12 raiseArmy()

```
void Country::raiseArmy ( )
```

Function to call appropriate creational structures to create an army and add it to Country's armies.

Author

Luke Lawson (u21433811)

3.21.3.13 sendSupplies()

```
void Country::sendSupplies (
            AmmoSupply * ammo,
            MedicalSupply * meds )
```

Function to send/distribute supplies to a Country's armies.

Author

Luke Lawson (u21433811)

Parameters

ammo	AmmoSupplies to be transported
meds	MedicalSupplies to be transported

3.21.3.14 setNewAmmoSupplies()

```
void Country::setNewAmmoSupplies (
            AmmoSupply * newAmmoSupply )
```

Function to set the member variable to store the newly created ammo supply.

Author

Reuben Jooste (u21457060)

Parameters

newAmmoSupply	The new ammo supply

3.21.3.15 setNewMedicalSupplies()

```
void Country::setNewMedicalSupplies (
            MedicalSupply * newMedicalSupply )
```

Function to set the member variable to store the newly created medical supply.

Author

Reuben Jooste (u21457060)

Parameters

newAmmoSupply	The new medical supply

3.21.3.16 spendGDP()

```
void Country::spendGDP (
            double gdpSpent )
```

Function to decrease Country GDP and manage change of economic state.

Author

Luke Lawson (u21433811)

Parameters

gdpSpent	double which indicates the amount to decrease GDP by

Warning

The parameter must be a negative double value. If we want to increase the GDP we must use the function "earnGDP()".

3.21.3.17 surrender()

```
void Country::surrender ( )
```

Function to cause Country to surrender and withdraw from the War and alliance.

Author

Luke Lawson (u21433811)

3.21.3.18 takeTurn()

```
void Country::takeTurn (
            War * currWar )
```

Function to decide and enact the Country's play for a turn.

Author

Luke Lawson (u21433811)

Parameters

currWar	pointer to the War the Country is currently engaged in

3.21.3.19 upgradeSupplyFactory()

```
void Country::upgradeSupplyFactory ( )
```

Function to upgrade a Country's Supply Factory such to produce better/greater quantity of medical supplies and ammo.

Author

Luke Lawson (u21433811)

Warning

This function can only be used when the Country is in a Rich EconomicState

3.21.3.20 upgradeUnitFactory()

```
void Country::upgradeUnitFactory ( )
```

Function to upgrade a Country's Unit Factory such to produce better military units.

Author

Luke Lawson (u21433811)

Warning

This function can only be used when the Country is in a Rich EconomicState

3.21.4 Member Data Documentation

3.21.4.1 alliance1

```
std::vector< Country * > Country::alliance1  [static]
```

3.21.4.2 alliance2

```
std::vector< Country * > Country::alliance2  [static]
```

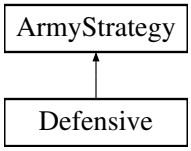
3.21.4.3 neutral

```
std::vector< Country * > Country::neutral  [static]
```

3.22 Defensive Class Reference

```
#include <Defensive.h>
```

Inheritance diagram for Defensive:



Public Member Functions

- void applyStrategyBonus (BattleStatistics, Battalion *)
  Applies desired Defensive bonuses to BattleStatistics.

3.22.1 Member Function Documentation

3.22.1.1 applyStrategyBonus()

```
void Defensive::applyStrategyBonus (
            BattleStatistics in,
            Battalion * inArmy )  [virtual]
```

Applies desired Defensive bonuses to BattleStatistics.

Author

Thomas Blendulf (u21446131)

Parameters

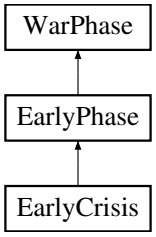
BattleStatistics	passes in the BattleStatistics to be edited.
Battalion	passes in the Battalion to calculate base statistics to be editted.

Reimplemented from ArmyStrategy.

3.23 EarlyCrisis Class Reference

`#include <EarlyCrisis.h>`

Inheritance diagram for EarlyCrisis:



Public Member Functions

- EarlyCrisis ()

  Sets next to null and peaceChance.
- void outputChange ()

  This function just shows when there was a change to the current WarPhase.

Additional Inherited Members

3.23.1 Constructor & Destructor Documentation

3.23.1.1 EarlyCrisis()

`EarlyCrisis::EarlyCrisis ( )`

Sets next to null and peaceChance.

Author

Thomas Blendulf (u21446131)

3.23.2 Member Function Documentation

3.23.2.1 outputChange()

```
void EarlyCrisis::outputChange ( )  [virtual]
```

This function just shows when there was a change to the current WarPhase.

Author

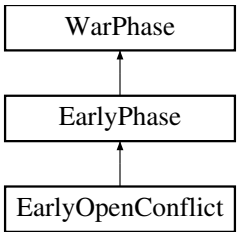
Thomas Blendulf (u21446131)

Reimplemented from EarlyPhase.

3.24 EarlyOpenConflict Class Reference

```
#include <EarlyOpenConflict.h>
```

Inheritance diagram for EarlyOpenConflict:



Public Member Functions

- EarlyOpenConflict ()

  Sets next to EarlyCrisis and peaceChance.
- void outputChange ()

  This function just shows when there was a change to the current WarPhase.

Additional Inherited Members

3.24.1 Constructor & Destructor Documentation

3.24.1.1 EarlyOpenConflict()

```
EarlyOpenConflict::EarlyOpenConflict ( )
```

Sets next to EarlyCrisis and peaceChance.

Author

Thomas Blendulf (u21446131)

3.24.2 Member Function Documentation

3.24.2.1 outputChange()

```
void EarlyOpenConflict::outputChange ( )  [virtual]
```

This function just shows when there was a change to the current WarPhase.

Author

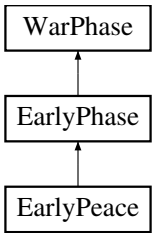
Thomas Blendulf (u21446131)

Reimplemented from EarlyPhase.

3.25 EarlyPeace Class Reference

```
#include <EarlyPeace.h>
```

Inheritance diagram for EarlyPeace:



Public Member Functions

- EarlyPeace ()

  Sets next to EarlyOpenConflict and peaceChance.

Additional Inherited Members

3.25.1 Constructor & Destructor Documentation

3.25.1.1 EarlyPeace()

```
EarlyPeace::EarlyPeace ( )
```

Sets next to EarlyOpenConflict and peaceChance.

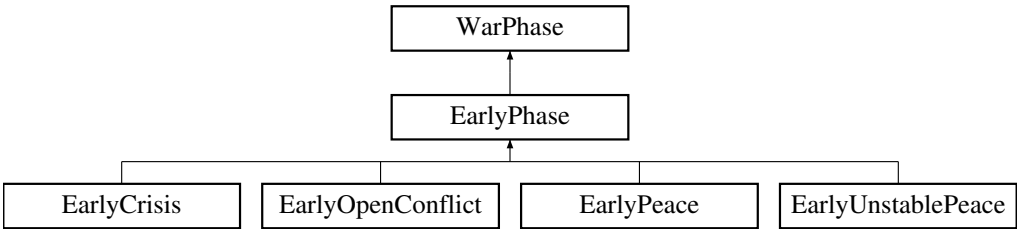
Author

Thomas Blendulf (u21446131)

3.26 EarlyPhase Class Reference

```
#include <EarlyPhase.h>
```

Inheritance diagram for EarlyPhase:



Public Member Functions

- void handleChange (War *)

  Sets Wars phase to next early phase or MidPhase.
- virtual void outputChange ()

  This function will be implemented by the derived classes. (EarlyCrisis, EarlyOpenConflict & EarlyPeace)

Public Attributes

- EarlyPhase * next

3.26.1 Member Function Documentation

3.26.1.1 handleChange()

```
void EarlyPhase::handleChange (
            War * inWar )  [virtual]
```

Sets Wars phase to next early phase or MidPhase.

Author

Thomas Blendulf (u21446131)

Parameters

*War**	passes in the war which must have its phase changed.

Implements WarPhase.

3.26.1.2 outputChange()

```
virtual void EarlyPhase::outputChange ( )  [inline], [virtual]
```

This function will be implemented by the derived classes. (EarlyCrisis, EarlyOpenConflict & EarlyPeace)

Author

Thomas Blendulf (u21446131)

Reimplemented in EarlyCrisis, EarlyOpenConflict, and EarlyUnstablePeace.

3.26.2 Member Data Documentation

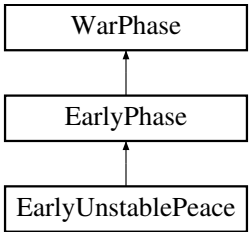
3.26.2.1 next

```
EarlyPhase* EarlyPhase::next
```

3.27 EarlyUnstablePeace Class Reference

```
#include <EarlyUnstablePeace.h>
```

Inheritance diagram for EarlyUnstablePeace:



Public Member Functions

- EarlyUnstablePeace ()

  Sets next to EarlyOpenConflict and peaceChance.
- void outputChange ()

  This function just shows when there was a change to the current WarPhase.

Additional Inherited Members

3.27.1 Constructor & Destructor Documentation

3.27.1.1 EarlyUnstablePeace()

```
EarlyUnstablePeace::EarlyUnstablePeace ( )
```

Sets next to EarlyOpenConflict and peaceChance.

Author

Thomas Blendulf (u21446131)

3.27.2 Member Function Documentation

3.27.2.1 outputChange()

```
void EarlyUnstablePeace::outputChange ( )  [virtual]
```

This function just shows when there was a change to the current WarPhase.

Author

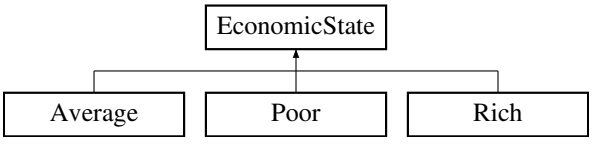
Thomas Blendulf (u21446131)

Reimplemented from EarlyPhase.

3.28 EconomicState Class Reference

```
#include <EconomicState.h>
```

Inheritance diagram for EconomicState:



Public Member Functions

- virtual int decideMyTurn (Country *country)=0

  randomly decide what a country can do during their turn

3.28.1 Member Function Documentation

3.28.1.1 decideMyTurn()

```
virtual int EconomicState::decideMyTurn (
            Country * country )  [pure virtual]
```

randomly decide what a country can do during their turn

Author

Jonelle Coertze (u21446271)

Parameters

country	pointer to an existing Country object to have access to the country's army and alliences

Returns

an int corresponding with the decision

Implemented in Average, Poor, and Rich.

3.29 EconommicState Class Reference

```
#include <EconomicState.h>
```

3.29.1 Detailed Description

This class will be used to specify in which economic state the country is throughout the duration of the War.

There are three states in which a country can find itself:

- Rich: Being in a rich state gives the country the benefit of upgrading its UnitFactory and SupplyFactory. This would increase the budget of the factories therefore making it possible to create more units and supplies giving them a greater advantage to winnning the War.
- Average: Being in an average state has the same advantages of being in a rich state except the Country should be careful not to spend too much gdp since they are only in an average state.
- Poor: Being in a poor state has the disadvantage of not being able to upgrade factories. Therefore if the factories run out of budget then the country will not be able to create armies and send supplies to the armies. In this state the Country has the option to surrender to the enemy.

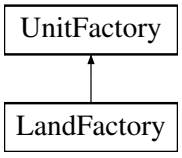
Note

The economic state of a country can change rappidly throughout the duration of the War.

3.30 LandFactory Class Reference

`#include <LandFactory.h>`

Inheritance diagram for LandFactory:



Public Member Functions

- LandFactory (double budget, int level, std::string type="Land")

  Constructor for LandFactory class used to instantiate an LandFactory object.
- ArmyComponent * createVehicle ()

  Calls constructor of LandVehicle, using level to determine powerRating.
- ArmyComponent * createSoldier ()

  Calls constructor of LandUnit, using level to determine powerRating.

Additional Inherited Members

3.30.1 Detailed Description

The LandFactory class is a derived class derived from the UnitFactory class (See the definition of the UnitFactory class)

The LandFactory will be used to create Land Units for the War. The LandFactory has methods "createSoldier()" and "createVehicle()" which will create Soldier objects and Vehicle objects respectively.

Note

This class is ONLY used to create LandUnit objects (Soldiers or Vehicles)

3.30.2 Constructor & Destructor Documentation

3.30.2.1 LandFactory()

```
LandFactory::LandFactory (
            double budget,
            int level,
            std::string type = "Land" )
```

Constructor for LandFactory class used to instantiate an LandFactory object.

Author

Reuben Jooste (u21457060)

Parameters

budget	Starting budget of LandFactory class
level	Starting level of LandFactory class
type	Type will be "Land" since this function creates Land army components

Warning

The "budget" must be a positive value. The "level" must be greater than zero.

3.30.3 Member Function Documentation

3.30.3.1 createSoldier()

```
ArmyComponent * LandFactory::createSoldier ( )  [virtual]
```

Calls constructor of LandUnit, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a LandUnit)

Note

This function may return NULL if the budget has run out.

Implements UnitFactory.

3.30.3.2 createVehicle()

```
ArmyComponent * LandFactory::createVehicle ( )  [virtual]
```

Calls constructor of LandVehicle, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a LandVehicle)

Note

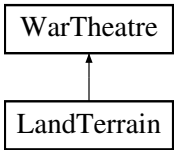
This function may return NULL if the budget has run out.

Implements UnitFactory.

3.31 LandTerrain Class Reference

`#include <LandTerrain.h>`

Inheritance diagram for LandTerrain:



Public Member Functions

- LandTerrain ()

  This is the default constructor of the class.

3.31.1 Detailed Description

(See the definition of the WarTheatre class)

The LandTerrain class is a derived class derived from the WarTheatre class The LandTerrain will be used to create a Land Terrain Theatre where the war can will take place.

Note

This class is used to expand the War by adding a new Theatre to it.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 LandTerrain()

`LandTerrain::LandTerrain ( )`

This is the default constructor of the class.

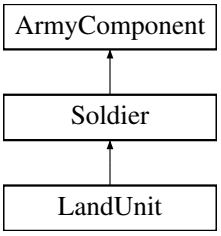
Author

Jonelle Coertze (u21446271)

3.32 LandUnit Class Reference

`#include <LandUnit.h>`

Inheritance diagram for LandUnit:



Public Member Functions

- LandUnit (int powerRating)

  Constructs LandUnit object, calling constructor of parent Soldier.
- int calculateAirOffense ()

  Calculates the AirOffense statistic of the unit.
- int calculateAirDefense ()

  Calculates the AirDefence statistic of the unit.
- int calculateSeaOffense ()

  Calculates the SeaOffense statistic of the unit.
- int calculateSeaDefense ()

  Calculates the SeaDefence statistic of the unit.
- int calculateLandOffense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's LandOffence statistic.
- int calculateLandDefense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's LandDefence statistic.

Additional Inherited Members

3.32.1 Detailed Description

(See the definition of the Soldier class)

The LandUnit class is a derived class derived from the Soldier class. The LandUnit will be used to create Land unit such as a Soldier to fight on the battlefield i.e. an Land WarTheatre.

Note

This class is used to do the calculations for a Soldier and to instantiate a Soldier object.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 LandUnit()

```
LandUnit::LandUnit (
            int powerRating )
```

Constructs LandUnit object, calling constructor of parent Soldier.

Author

Luke Lawson (u21433811)

Parameters

powerRating	The powerRating of the particular unit as per factory's cost (higher cost -> higher power)

Warning

The powerRating must be greater than zero.

3.32.3 Member Function Documentation

3.32.3.1 calculateAirDefense()

```
int LandUnit::calculateAirDefense ( )  [virtual]
```

Calculates the AirDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a LandUnit and not an AirUnit.

Implements Soldier.

3.32.3.2 calculateAirOffense()

```
int LandUnit::calculateAirOffense ( )  [virtual]
```

Calculates the AirOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a LandUnit and not an AirUnit.

Implements Soldier.

3.32.3.3 calculateLandDefense()

```
int LandUnit::calculateLandDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's Land↩ Defence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandDefence statistic of unit

Note

The returned value is a random integer value.

Implements Soldier.

3.32.3.4 calculateLandOffense()

```
int LandUnit::calculateLandOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's LandOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of unit

Note

The returned value is a random integer value.

Implements Soldier.

3.32.3.5 calculateSeaDefense()

```
int LandUnit::calculateSeaDefense ( )  [virtual]
```

Calculates the SeaDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a LandUnit and not a SeaUnit.

Implements Soldier.

3.32.3.6 calculateSeaOffense()

```
int LandUnit::calculateSeaOffense ( )  [virtual]
```

Calculates the SeaOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

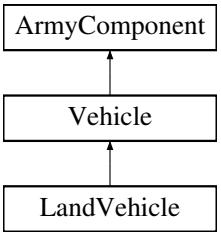
It returns 0 since this is a LandUnit and not a SeaUnit.

Implements Soldier.

3.33 LandVehicle Class Reference

```
#include <LandVehicle.h>
```

Inheritance diagram for LandVehicle:



Public Member Functions

- LandVehicle (int powerRating)

  Constructs LandVehicle object, using powerRating to randomly generate attributes from Normal Dist. (higher power -> better attributes)
- int calculateAirOffense ()

  Calculates the AirOffense statistic of the vehicle.
- int calculateAirDefense ()

  Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's AirDefence statistic.
- int calculateSeaOffense ()

  Calculates the SeaOffense statistic of the vehicle.
- int calculateSeaDefense ()

  Calculates the SeaDefence statistic of the vehicle.
- int calculateLandOffense ()

  Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's LandOffence statistic.
- int calculateLandDefense ()

  Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's LandDefence statistic.

Additional Inherited Members

3.33.1 Detailed Description

(See the definition of the Vehicle class)

The LandVehicle class is a derived class derived from the Vehicle class. The LandVehicle will be used to create an individual unit (vehicle) which will fight alongside soldier units in the war.

Note

This class is used to do the calculations for a Vehicle and to instantiate a Vehicle object.

3.33.2 Constructor & Destructor Documentation

3.33.2.1 LandVehicle()

```
LandVehicle::LandVehicle (
            int powerRating )
```

Constructs LandVehicle object, using powerRating to randomly generate attributes from Normal Dist. (higher power -> better attributes)

Author

Luke Lawson (u21433811)

Parameters

powerRating	The powerRating of the particular vehicle as per factory's cost (higher cost -> higher power)

Warning

The powerRating cannot be a negative number.

3.33.3 Member Function Documentation

3.33.3.1 calculateAirDefense()

```
int LandVehicle::calculateAirDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's AirDefence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing AirDefence statistic of vehicle

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a LandVehicle and not an AirVehicle.

Implements Vehicle.

3.33.3.2 calculateAirOffense()

```
int LandVehicle::calculateAirOffense ( )  [virtual]
```

Calculates the AirOffense statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a LandVehicle and not an AirVehicle.

Implements Vehicle.

3.33.3.3 calculateLandDefense()

```
int LandVehicle::calculateLandDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's Land↩ Defence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandDefence statistic of vehicle

Note

The returned value is a random integer value.

Implements Vehicle.

3.33.3.4 calculateLandOffense()

```
int LandVehicle::calculateLandOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's Land↩ Offence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of vehicle

Note

The returned value is a random integer value.

Implements Vehicle.

3.33.3.5 calculateSeaDefense()

```
int LandVehicle::calculateSeaDefense ( )  [virtual]
```

Calculates the SeaDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a LandVehicle and not a SeaVehicle.

Implements Vehicle.

3.33.3.6 calculateSeaOffense()

```
int LandVehicle::calculateSeaOffense ( )  [virtual]
```

Calculates the SeaOffense statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

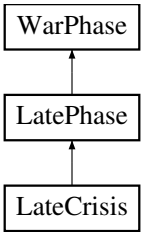
It returns 0 since this is a LandVehicle and not a SeaVehicle.

Implements Vehicle.

3.34 LateCrisis Class Reference

`#include <LateCrisis.h>`

Inheritance diagram for LateCrisis:



Public Member Functions

- LateCrisis ()

  Sets next to LateOpenConflict.
- void outputChange ()

  This function just shows when there was a change to the current WarPhase.

Additional Inherited Members

3.34.1 Constructor & Destructor Documentation

3.34.1.1 LateCrisis()

`LateCrisis::LateCrisis ( )`

Sets next to LateOpenConflict.

Author

Thomas Blendulf (u21446131)

3.34.2 Member Function Documentation

3.34.2.1 outputChange()

`void LateCrisis::outputChange ( )  [virtual]`

This function just shows when there was a change to the current WarPhase.

Author

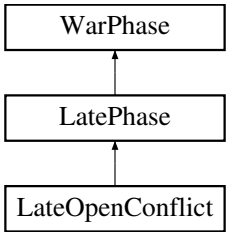
Thomas Blendulf (u21446131)

Reimplemented from LatePhase.

3.35 LateOpenConflict Class Reference

`#include <LateOpenConflict.h>`

Inheritance diagram for LateOpenConflict:



Public Member Functions

- LateOpenConflict ()
  Sets next to LateUnstablePeace.
- void outputChange ()
  This function just shows when there was a change to the current WarPhase.

Additional Inherited Members

3.35.1 Constructor & Destructor Documentation

3.35.1.1 LateOpenConflict()

`LateOpenConflict::LateOpenConflict ( )`

Sets next to LateUnstablePeace.

Author

Thomas Blendulf (u21446131)

3.35.2 Member Function Documentation

3.35.2.1 outputChange()

`void LateOpenConflict::outputChange ( )  [virtual]`

This function just shows when there was a change to the current WarPhase.

Author

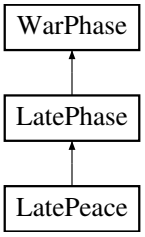
Thomas Blendulf (u21446131)

Reimplemented from LatePhase.

3.36 LatePeace Class Reference

`#include <LatePeace.h>`

Inheritance diagram for LatePeace:



Public Member Functions

- LatePeace ()

  Sets next to EarlyOpenConflict and peaceChance.

Additional Inherited Members

3.36.1 Constructor & Destructor Documentation

3.36.1.1 LatePeace()

`LatePeace::LatePeace ( )`

Sets next to EarlyOpenConflict and peaceChance.

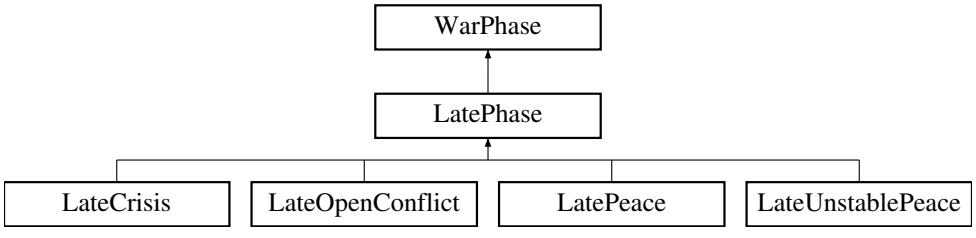
Author

Thomas Blendulf (u21446131)

3.37 LatePhase Class Reference

`#include <LatePhase.h>`

Inheritance diagram for LatePhase:



Public Member Functions

- void handleChange (War *)

  Sets Wars phase to next Late phase.
- virtual void outputChange ()

  This function will be implemented by the derived classes. (LateCrisis, LateOpenConflict, LatePeace & LateUnstablePeace)

Public Attributes

- LatePhase * next

3.37.1 Member Function Documentation

3.37.1.1 handleChange()

```
void LatePhase::handleChange (
            War * inWar )  [virtual]
```

Sets Wars phase to next Late phase.

Author

Thomas Blendulf (u21446131)

Parameters

*War**	passes in the war which must have its phase changed.

Implements WarPhase.

3.37.1.2 outputChange()

```
virtual void LatePhase::outputChange ( )  [inline], [virtual]
```

This function will be implemented by the derived classes. (LateCrisis, LateOpenConflict, LatePeace & LateUnstablePeace)

Author

Thomas Blendulf (u21446131)

Reimplemented in LateCrisis, LateOpenConflict, and LateUnstablePeace.

3.37.2 Member Data Documentation

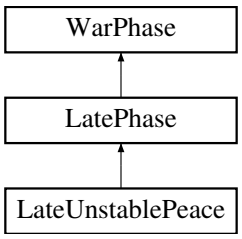
3.37.2.1 next

```
LatePhase* LatePhase::next
```

3.38 LateUnstablePeace Class Reference

```
#include <LateUnstablePeace.h>
```

Inheritance diagram for LateUnstablePeace:



Public Member Functions

- LateUnstablePeace ()

  Sets next to null.
- void outputChange ()

  This function just shows when there was a change to the current WarPhase.

Additional Inherited Members

3.38.1 Constructor & Destructor Documentation

3.38.1.1 LateUnstablePeace()

```
LateUnstablePeace::LateUnstablePeace ( )
```

Sets next to null.

Author

Thomas Blendulf (u21446131)

3.38.2 Member Function Documentation

3.38.2.1 outputChange()

```
void LateUnstablePeace::outputChange ( )  [virtual]
```

This function just shows when there was a change to the current WarPhase.

Author

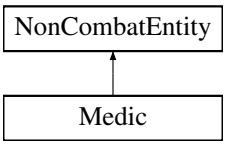
Thomas Blendulf (u21446131)

Reimplemented from LatePhase.

3.39 Medic Class Reference

```
#include <Medic.h>
```

Inheritance diagram for Medic:



Public Member Functions

- Medic ()

  A default constructor that generates the amount of healing of a Medic.
- Medic (int Healing)

  a parameterized constructor to be used in the clone function
- NonCombatEntity * clone ()

  clone the current Medic
- int getHealing ()

  get a random number between 1 and 10 that can be seen as the healing value

3.39.1 Constructor & Destructor Documentation

3.39.1.1 Medic() [1/2]

```
Medic::Medic ( )
```

A default constructor that generates the amount of healing of a Medic.

Author

Jonelle Coertze (u21446271)

3.39.1.2 Medic() [2/2]

```
Medic::Medic (
            int Healing )
```

a parameterized constructor to be used in the clone function

Author

Jonelle Coertze (u21446271)

Parameters

Healing	an integer to set the healing of the Medic

3.39.2 Member Function Documentation

3.39.2.1 clone()

```
NonCombatEntity * Medic::clone ( )  [virtual]
```

clone the current Medic

Author

Jonelle Coertze (u21446271)

Returns

a pointer to the cloned/new NonCombatEntity : Medic

Implements NonCombatEntity.

3.39.2.2 getHealing()

```
int Medic::getHealing ( )
```

get a random number between 1 and 10 that can be seen as the healing value

Author

Jonelle Coertze (u21446271)

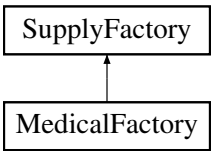
Returns

a integer to indicate the amount of healing of a Medic

3.40 MedicalFactory Class Reference

```
#include <MedicalFactory.h>
```

Inheritance diagram for MedicalFactory:



Public Member Functions

- MedicalFactory (int budget, std::string type)

  Class constructor for MedicalFactory.
- Supply * makeSupply (int quantity)

  Creates medical supplies by creating a new MedicalSupply product.

Additional Inherited Members

3.40.1 Detailed Description

The MedicalFactory class is a derived class derived from the SupplyFactory class (See the definition of the SupplyFactory class)

The MedicalFactory will be used to create Medical Supplies for the Country's Armies. The MedicalFactory has a method "makeSupply()" which will create the MedicalSuppply object.

Note

This class is ONLY used to create MedicalSupply objects

3.40.2 Constructor & Destructor Documentation

3.40.2.1 MedicalFactory()

```
MedicalFactory::MedicalFactory (
            int budget,
            std::string type )
```

Class constructor for MedicalFactory.

Author

Arno Jooste (u21457451)

Parameters

budget	The amount that can be spent to make medical supplies.
type	The type of the factory.

Warning

"type" has to be "Medical" and "budget" may not be less than or equal to zero.

3.40.3 Member Function Documentation

3.40.3.1 makeSupply()

```
Supply * MedicalFactory::makeSupply (
            int quantity )  [virtual]
```

Creates medical supplies by creating a new MedicalSupply product.

Author

Arno Jooste (u21457451)

Parameters

quantity	The quantity of medical supplies to be produced by the medical factory.

Returns

Pointer to newly created MedicalSupply product.

Warning

"quantity" need to be positive.

Note

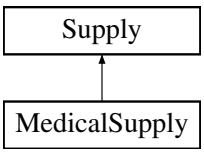
This function may return NULL if the budget has run out.

Implements SupplyFactory.

3.41 MedicalSupply Class Reference

`#include <MedicalSupply.h>`

Inheritance diagram for MedicalSupply:



Public Member Functions

- MedicalSupply (int factoryLevel, int quantity)

  Constructor for MedicalSupply class to specify the factory level and quantity that will be produced.
- int getMedicalBonus ()

  Getter for the medical bonus member variable.
- void setMedicalBonus (int bonus)

  Setter for the medical bonus member variable.

Additional Inherited Members

3.41.1 Detailed Description

This class is derived from the Supply class. This is the actual product created by the MedicalFactory.

- Armies will use medical supply objects to recuperate during the war.
- If an army's medical supply runs out they are in great disadvantage.

Note

The medical supply has a variety of small and large medical kits.

3.41.2 Constructor & Destructor Documentation

3.41.2.1 MedicalSupply()

```
MedicalSupply::MedicalSupply (
            int factoryLevel,
            int quantity )
```

Constructor for MedicalSupply class to specify the factory level and quantity that will be produced.

Author

Arno Jooste (21457451)

Parameters

factoryLevel	Specifies the currrent factory level in order to set the multiplier of the bonus.
quantity	Specifies the quentity of medical supplies to be produced. This amount will be used to calculate the medicalBonus

Warning

The factoryLevel must be a value greater than the integer value 0.
The quantity must also be a value greater than zero.

3.41.3 Member Function Documentation

3.41.3.1 getMedicalBonus()

```
int MedicalSupply::getMedicalBonus ( )
```

Getter for the medical bonus member variable.

Author

Arno Jooste (u21457451)

Returns

medical bonus of type int.

3.41.3.2 setMedicalBonus()

```
void MedicalSupply::setMedicalBonus (
            int bonus )
```

Setter for the medical bonus member variable.

Author

Arno Jooste (u21457451)

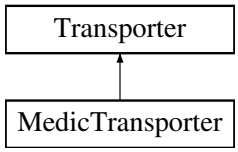
Parameters

bonus	Specfies to which value the medical bonus will be set.

3.42 MedicTransporter Class Reference

`#include <MedicTransporter.h>`

Inheritance diagram for MedicTransporter:



Public Member Functions

- MedicTransporter ()

  Constructor for the MedicTransport class used to instantiate the object.
- virtual ~MedicTransporter ()

  Destructor for the MedicTransport class used to deallocate the dynamic memory used by the member variable corresponderList.
- virtual void notify (Corresponder *corresponder)

  Notify all Corresponder objects in the corresponderList variable.

Additional Inherited Members

3.42.1 Detailed Description

This class is derived from the Trnasporter class.

- We use this class to showcase the transport of medical supplies which a Country sends to its Army.
- This class notifies all armies of a country that the country has sent medical supplies to it.

Warning

If this transport line is destroyed by an enemy country, the country will not be able to send any medical supplies. Thus being in a great disadvantage.

3.42.2 Constructor & Destructor Documentation

3.42.2.1 MedicTransporter()

```
MedicTransporter::MedicTransporter ( )
```

Constructor for the MedicTransport class used to instantiate the object.

Author

Reuben Jooste (u21457060)

3.42.2.2 ~MedicTransporter()

```
MedicTransporter::~MedicTransporter ( )  [virtual]
```

Destructor for the MedicTransport class used to deallocate the dynamic memory used by the member variable corresponderList.

Author

Reuben Jooste (u21457060)

3.42.3 Member Function Documentation

3.42.3.1 notify()

```
void MedicTransporter::notify (
            Corresponder * corresponder )  [virtual]
```

Notify all Corresponder objects in the corresponderList variable.

Author

Reuben Jooste (u21457060)

Parameters

corresponder	pointer to the Corresponder in which a changed has happened.

Note

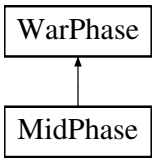
If the parameter is NULL the function would simply throw an exception and terminate.

Implements Transporter.

3.43 MidPhase Class Reference

`#include <MidPhase.h>`

Inheritance diagram for MidPhase:



Public Member Functions

- MidPhase ()
- void handleChange (War ∗)

  Sets Wars phase to LateCrisis.

Additional Inherited Members

3.43.1 Constructor & Destructor Documentation

3.43.1.1 MidPhase()

```
MidPhase::MidPhase ( )
```

3.43.2 Member Function Documentation

3.43.2.1 handleChange()

```
void MidPhase::handleChange (
            War * inWar )  [virtual]
```

Sets Wars phase to LateCrisis.

Author

Thomas Blendulf (u21446131)

Parameters

War∗	passes in the war which must have its phase changed.

Implements WarPhase.

3.44 MilitaryCommander Class Reference

`#include <MilitaryCommander.h>`

Public Member Functions

- MilitaryCommander ()
- void changeStrategy ()

  executes the changeStrategy Command.
- void setStrategy (Army ∗army, std::string newStrategy)

  sets variables of the ChangeStrategy Command.
- void enterTheatre ()

  executes the enterTheatre Command.
- void setTheatreTarget (Army ∗army, WarTheatre ∗theatreTarget)

  sets variables of the enterTheatre Command.
- void attackTransport ()

  executes the enterTheatre Command.
- void setTransportTarget (Country ∗transportTarget, Army ∗army)

  sets variables of the attackTransport Command.

3.44.1 Detailed Description

This is one of the big components in the War. Each Country will make use of the military commander to issue different commands throughout the duration of the war. The different commands which a commander can issue includes the following:

- change strategy: This command is issued when a Country either has the greater advantage in the war or when it realises that the enemy has the greater advantage.
- set strategy: This command will be issued when a country has decided to use a different strategy in the war.
- enter theatre: Issuing this command will allow the country to start participating in the war.
- set theatre target: This command is issued to order the country's army to attack/fight in a specific WarTheatre.
- attack transport: This command orders the country's army to attack an enemy's transport lines with the hope of destroying it to gain the upper hand in the war.
- set transport target: This command allows the country's army to attack a specific Country's transport lines.

The military commander is definitely a crucial component for a country because without a commander the country will not be able to instruct its army to attack/surrender.

Note

A country can only have one military commander.

3.44.2 Constructor & Destructor Documentation

3.44.2.1 MilitaryCommander()

```
MilitaryCommander::MilitaryCommander ( )
```

3.44.3 Member Function Documentation

3.44.3.1 attackTransport()

```
void MilitaryCommander::attackTransport ( )
```

executes the enterTheatre Command.

Author

Thomas Blendulf(u21446131)

3.44.3.2 changeStrategy()

```
void MilitaryCommander::changeStrategy ( )
```

executes the changeStrategy Command.

Author

Thomas Blendulf(u21446131)

3.44.3.3 enterTheatre()

```
void MilitaryCommander::enterTheatre ( )
```

executes the enterTheatre Command.

Author

Thomas Blendulf(u21446131)

3.44.3.4 setStrategy()

```
void MilitaryCommander::setStrategy (
            Army * army,
            std::string newStrategy )
```

sets variables of the ChangeStrategy Command.

Author

Thomas Blendulf(u21446131)

Parameters

Army∗	the army to be set in the ChangeStrategy Command.
String	the string storing the state of the Command.

Warning

The second argument must be a string value to specify a specific ArmyStrategy

3.44.3.5 setTheatreTarget()

```
void MilitaryCommander::setTheatreTarget (
            Army * army,
            WarTheatre * theatreTarget )
```

sets variables of the enterTheatre Command.

Author

Thomas Blendulf(u21446131)

Parameters

Army∗	the army to be set in the enterTheatre Command.
WarTheatre∗	the war theatre the army is to fight in.

Warning

The second argument must be an existing WarTheatre in the current War

3.44.3.6 setTransportTarget()

```
void MilitaryCommander::setTransportTarget (
            Country * transportTarget,
            Army * army )
```

sets variables of the attackTransport Command.

Author

Thomas Blendulf(u21446131)

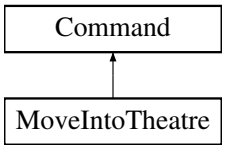
Parameters

Army∗	the army to be set in the attackTransport Command.
Transporter∗	the transport to be attacked.

3.45 MoveIntoTheatre Class Reference

`#include <MoveIntoTheatre.h>`

Inheritance diagram for MoveIntoTheatre:



Public Member Functions

- void setTheatre (WarTheatre ∗)
  sets the war theatre to be executed by the commmand pattern.
- void execute ()
  sets the stored armies war theatre to fight in.

Public Attributes

- WarTheatre ∗ theatre

Additional Inherited Members

3.45.1 Member Function Documentation

3.45.1.1 execute()

`void MoveIntoTheatre::execute ( )  [virtual]`

sets the stored armies war theatre to fight in.

Author

Thomas Blendulf(u21446131)

Implements Command.

3.45.1.2 setTheatre()

```
void MoveIntoTheatre::setTheatre (
            WarTheatre * in )
```

sets the war theatre to be executed by the commmand pattern.

Author

Thomas Blendulf(u21446131)

Parameters

WarTheatre	containing theatre to be updated to.

3.45.2 Member Data Documentation

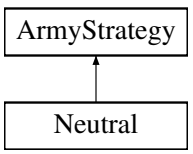
3.45.2.1 theatre

```
WarTheatre* MoveIntoTheatre::theatre
```

3.46 Neutral Class Reference

```
#include <Neutral.h>
```

Inheritance diagram for Neutral:



Public Member Functions

- void applyStrategyBonus (BattleStatistics, Battalion *)

  Applies desired Neutral bonuses to BattleStatistics.

3.46.1 Member Function Documentation

3.46.1.1 applyStrategyBonus()

```
void Neutral::applyStrategyBonus (
            BattleStatistics in,
            Battalion * inArmy )  [virtual]
```

Applies desired Neutral bonuses to BattleStatistics.

Author

Thomas Blendulf (u21446131)

Parameters

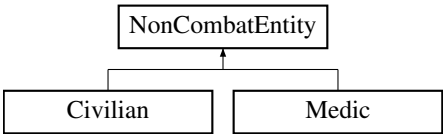
BattleStatistics	passes in the BattleStatistics to be edited.
Battalion	passes in the Battalion to calculate base statistics to be editted.

Reimplemented from ArmyStrategy.

3.47 NonCombatEntity Class Reference

`#include <NonCombatEntity.h>`

Inheritance diagram for NonCombatEntity:



Public Member Functions

- virtual NonCombatEntity * clone ()=0
  clone the current NonCombatEntity

3.47.1 Member Function Documentation

3.47.1.1 clone()

```
virtual NonCombatEntity* NonCombatEntity::clone ( )  [pure virtual]
```

clone the current NonCombatEntity

Author

Jonelle Coertze (u21446271)

Returns

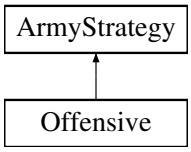
a pointer to the cloned/new NonCombatEntity

Implemented in Civilian, and Medic.

3.48 Offensive Class Reference

`#include <Offensive.h>`

Inheritance diagram for Offensive:



Public Member Functions

- void applyStrategyBonus (BattleStatistics, Battalion *)

  Applies desired Offensive bonuses to BattleStatistics.

3.48.1 Member Function Documentation

3.48.1.1 applyStrategyBonus()

```
void Offensive::applyStrategyBonus (
            BattleStatistics in,
            Battalion * inArmy )  [virtual]
```

Applies desired Offensive bonuses to BattleStatistics.

Author

Thomas Blendulf (u21446131)

Parameters

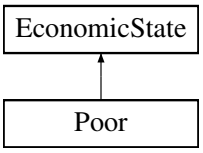
BattleStatistics	passes in the BattleStatistics to be edited.
Battalion	passes in the Battalion to calculate base statistics to be editted.

Reimplemented from ArmyStrategy.

3.49 Poor Class Reference

`#include <Poor.h>`

Inheritance diagram for Poor:



Public Member Functions

- int decideMyTurn (Country *country)

  randomly decide what a country can do during their turn

3.49.1 Member Function Documentation

3.49.1.1 decideMyTurn()

```
int Poor::decideMyTurn (
            Country * country )  [virtual]
```

randomly decide what a country can do during their turn

Author

Jonelle Coertze (u21446271)

Parameters

country	pointer to an existing Country object to have access to the country's army and alliences

Returns

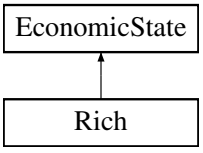
an int corresponding with the decision

Implements EconomicState.

3.50 Rich Class Reference

```
#include <Rich.h>
```

Inheritance diagram for Rich:



Public Member Functions

- int decideMyTurn (Country *country)
  randomly decide what a country can do during their turn

3.50.1 Member Function Documentation

3.50.1.1 decideMyTurn()

```
int Rich::decideMyTurn (
            Country * country )  [virtual]
```

randomly decide what a country can do during their turn

Author

Jonelle Coertze (u21446271)

Parameters

country	pointer to an existing Country object to have access to the country's army and alliences

Returns

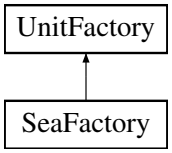
an int corresponding with the decision

Implements EconomicState.

3.51 SeaFactory Class Reference

`#include <SeaFactory.h>`

Inheritance diagram for SeaFactory:



Public Member Functions

- SeaFactory (double budget, int level, std::string type="Sea")
  Constructor for SeaFactory class used to instantiate an SeaFactory object.
- ArmyComponent * createVehicle ()
  Calls constructor of SeaVehicle, using level to determine powerRating.
- ArmyComponent * createSoldier ()
  Calls constructor of SeaUnit, using level to determine powerRating.

Additional Inherited Members

3.51.1 Detailed Description

The SeaFactory class is a derived class derived from the UnitFactory class (See the definition of the UnitFactory class)

The SeaFactory will be used to create Sea Units for the War. The SeaFactory has methods "createSoldier()" and "createVehicle()" which will create Soldier objects and Vehicle objects respectively.

Note

This class is ONLY used to create SeaUnit objects (Soldiers or Vehicles)

3.51.2 Constructor & Destructor Documentation

3.51.2.1 SeaFactory()

```
SeaFactory::SeaFactory (
            double budget,
            int level,
            std::string type = "Sea" )
```

Constructor for SeaFactory class used to instantiate an SeaFactory object.

Author

Reuben Jooste (u21457060)

Parameters

budget	Starting budget of SeaFactory class
level	Starting level of SeaFactory class
type	Type will be "Sea" since this function creates Sea army components

Warning

The "budget" must be a positive value. The "level" must be greater than zero.

3.51.3 Member Function Documentation

3.51.3.1 createSoldier()

```
ArmyComponent * SeaFactory::createSoldier ( )  [virtual]
```

Calls constructor of SeaUnit, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a SeaUnit)

Note

This function may return NULL if the budget has run out.

Implements UnitFactory.

3.51.3.2 createVehicle()

```
ArmyComponent * SeaFactory::createVehicle ( )  [virtual]
```

Calls constructor of SeaVehicle, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a SeaVehicle)

Note

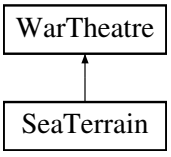
This function may return NULL if the budget has run out.

Implements UnitFactory.

3.52 SeaTerrain Class Reference

```
#include <SeaTerrain.h>
```

Inheritance diagram for SeaTerrain:



Public Member Functions

- SeaTerrain ()

  This is the default constructor of the class.

3.52.1 Detailed Description

(See the definition of the WarTheatre class)

The SeaTerrain class is a derived class derived from the WarTheatre class The SeaTerrain will be used to create a Sea Terrain Theatre where the war can will take place.

Note

This class is used to expand the War by adding a new Theatre to it.

3.52.2 Constructor & Destructor Documentation

3.52.2.1 SeaTerrain()

```
SeaTerrain::SeaTerrain ( )
```

This is the default constructor of the class.

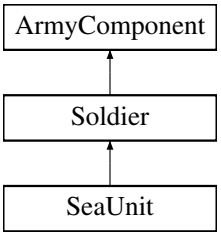
Author

Jonelle Coertze (u21446271)

3.53 SeaUnit Class Reference

```
#include <SeaUnit.h>
```

Inheritance diagram for SeaUnit:



Public Member Functions

- SeaUnit (int powerRating)

  Constructs LandUnit object, calling constructor of parent Soldier.
- int calculateAirOffense ()

  Calculates the AirOffense statistic of the unit.
- int calculateAirDefense ()

  Calculates the AirDefence statistic of the unit.
- int calculateSeaOffense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's SeaOffence statistic.
- int calculateSeaDefense ()

  Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's SeaDefence statistic.
- int calculateLandOffense ()

  Calculates the LandOffence statistic of the unit.
- int calculateLandDefense ()

  Calculates the LandDefence statistic of the unit.

Additional Inherited Members

3.53.1 Detailed Description

(See the definition of the Soldier class)

The SeaUnit class is a derived class derived from the Soldier class. The SeaUnit will be used to create Sea unit such as a Soldier to fight on the battlefield i.e. an Sea WarTheatre.

Note

This class is used to do the calculations for a Soldier and to instantiate a Soldier object.

3.53.2 Constructor & Destructor Documentation

3.53.2.1 SeaUnit()

```
SeaUnit::SeaUnit (
            int powerRating )
```

Constructs LandUnit object, calling constructor of parent Soldier.

Author

Luke Lawson (u21433811)

Parameters

powerRating	The powerRating of the particular unit as per factory's cost (higher cost -> higher power)

3.53.3 Member Function Documentation

3.53.3.1 calculateAirDefense()

```
int SeaUnit::calculateAirDefense ( )  [virtual]
```

Calculates the AirDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaUnit and not an AirUnit.

Implements Soldier.

3.53.3.2 calculateAirOffense()

```
int SeaUnit::calculateAirOffense ( )  [virtual]
```

Calculates the AirOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaUnit and not an AirUnit.

Implements Soldier.

3.53.3.3 calculateLandDefense()

```
int SeaUnit::calculateLandDefense ( )  [virtual]
```

Calculates the LandDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaUnit and not a LandUnit.

Implements Soldier.

3.53.3.4 calculateLandOffense()

```
int SeaUnit::calculateLandOffense ( )  [virtual]
```

Calculates the LandOffence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaUnit and not a LandUnit.

Implements Soldier.

3.53.3.5 calculateSeaDefense()

```
int SeaUnit::calculateSeaDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's SeaDefence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaDefence statistic of unit

Note

The returned value is a random integer value.

Implements Soldier.

3.53.3.6 calculateSeaOffense()

```
int SeaUnit::calculateSeaOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by trainingLevel) to randomly generate the unit's SeaOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaOffence statistic of unit

Note

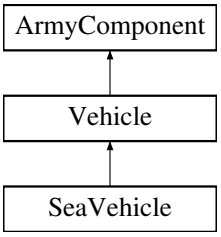
The returned value is a random integer value.

Implements Soldier.

3.54 SeaVehicle Class Reference

`#include <SeaVehicle.h>`

Inheritance diagram for SeaVehicle:



Public Member Functions

- SeaVehicle (int powerRating)

  Constructs SeaVehicle object, using powerRating to randomly generate attributes from Normal Dist. (higher power -> better attributes)
- int calculateAirOffense ()

  Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's AirOffence statistic.
- int calculateAirDefense ()

  Calculates the AirDefence statistic of the vehicle.
- int calculateSeaOffense ()

  Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's SeaOffence statistic.
- int calculateSeaDefense ()

  Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's SeaDefence statistic.
- int calculateLandOffense ()

  Calculates the LandOffence statistic of the vehicle.
- int calculateLandDefense ()

  Calculates the LandDefence statistic of the vehicle.

Additional Inherited Members

3.54.1 Detailed Description

(See the definition of the Vehicle class)

The SeaVehicle class is a derived class derived from the Vehicle class. The SeaVehicle will be used to create an individual unit (vehicle) which will fight alongside soldier units in the war.

Note

This class is used to do the calculations for a Vehicle and to instantiate a Vehicle object.

3.54.2 Constructor & Destructor Documentation

3.54.2.1 SeaVehicle()

```
SeaVehicle::SeaVehicle (
            int powerRating )
```

Constructs SeaVehicle object, using powerRating to randomly generate attributes from Normal Dist. (higher power -> better attributes)

Author

Luke Lawson (u21433811)

Parameters

powerRating	The powerRating of the particular vehicle as per factory's cost (higher cost -> higher power)

Warning

The powerRating must be a value greater than zero.

3.54.3 Member Function Documentation

3.54.3.1 calculateAirDefense()

```
int SeaVehicle::calculateAirDefense ( )  [virtual]
```

Calculates the AirDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaVehicle and not an AirVehicle.

Implements Vehicle.

3.54.3.2 calculateAirOffense()

```
int SeaVehicle::calculateAirOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's AirOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing AirOffense statistic of vehicle

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaVehicle and not an AirVehicle.

Implements Vehicle.

3.54.3.3 calculateLandDefense()

```
int SeaVehicle::calculateLandDefense ( )  [virtual]
```

Calculates the LandDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaVehicle and not a LandVehicle.

Implements Vehicle.

3.54.3.4 calculateLandOffense()

```
int SeaVehicle::calculateLandOffense ( )  [virtual]
```

Calculates the LandOffence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Note

The returned value is a random integer value.

Warning

It returns 0 since this is a SeaVehicle and not a LandVehicle.

Implements Vehicle.

3.54.3.5 calculateSeaDefense()

```
int SeaVehicle::calculateSeaDefense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by armourRating) to randomly generate the unit's Sea↩ Defence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaDefence statistic of vehicle

Note

The returned value is a random integer value.

Implements Vehicle.

3.54.3.6 calculateSeaOffense()

```
int SeaVehicle::calculateSeaOffense ( )  [virtual]
```

Use Normal Distribution (with mean and stddev scaled by weaponClass) to randomly generate the unit's SeaOffence statistic.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaOffense statistic of vehicle

Note

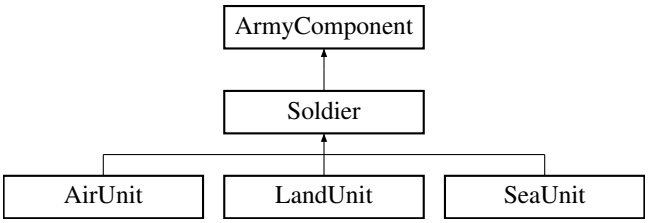
The returned value is a random integer value.

Implements Vehicle.

3.55 Soldier Class Reference

```
#include <Soldier.h>
```

Inheritance diagram for Soldier:



Public Member Functions

- Soldier (int powerRating)

  Construct Solider using powerLevel to determine trainingLevel.
- virtual int calculateAirOffense ()=0

  Calculates the AirOffense statistic of the unit. Implemented in child classes.
- virtual int calculateAirDefense ()=0

  Calculates the AirDefence statistic of the unit. Implemented in child classes.
- virtual int calculateSeaOffense ()=0

  Calculates the SeaOffense statistic of the unit. Implemented in child classes.
- virtual int calculateSeaDefense ()=0

  Calculates the SeaDefence statistic of the unit. Implemented in child classes.
- virtual int calculateLandOffense ()=0

  Calculates the LandOffence statistic of the unit. Implemented in child classes.
- virtual int calculateLandDefense ()=0

  Calculates the LandDefence statistic of the unit. Implemented in child classes.
- void addMember (ArmyComponent *newMember)

  Composite method to create composite ArmyComponents. Stubbed here.

Protected Attributes

- int trainingLevel

3.55.1 Constructor & Destructor Documentation

3.55.1.1 Soldier()

```
Soldier::Soldier (
            int powerRating )
```

Construct Solider using powerLevel to determine trainingLevel.

Author

Luke Lawson (u21433811)

Parameters

powerRating	powerRating of the Soldier (powerRating = trainingLevel)

3.55.2 Member Function Documentation

3.55.2.1 addMember()

```
void Soldier::addMember (
            ArmyComponent * newMember )  [virtual]
```

Composite method to create composite ArmyComponents. Stubbed here.

Author

Luke Lawson (u21433811)

Parameters

newMember	pointer to ArmyComponent to add to composite object

Implements ArmyComponent.

3.55.2.2 calculateAirDefense()

```
virtual int Soldier::calculateAirDefense ( )  [pure virtual]
```

Calculates the AirDefence statistic of the unit. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing AirDefence statistic of unit

Implements ArmyComponent.

Implemented in LandUnit, SeaUnit, and AirUnit.

3.55.2.3 calculateAirOffense()

```
virtual int Soldier::calculateAirOffense ( )  [pure virtual]
```

Calculates the AirOffense statistic of the unit. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing AirOffense statistic of unit

Implements ArmyComponent.

Implemented in LandUnit, AirUnit, and SeaUnit.

3.55.2.4 calculateLandDefense()

```
virtual int Soldier::calculateLandDefense ( )  [pure virtual]
```

Calculates the LandDefence statistic of the unit. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing LandDefence statistic of unit

Implements ArmyComponent.

Implemented in AirUnit, LandUnit, and SeaUnit.

3.55.2.5 calculateLandOffense()

```
virtual int Soldier::calculateLandOffense ( )  [pure virtual]
```

Calculates the LandOffence statistic of the unit. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of unit

Implements ArmyComponent.

Implemented in LandUnit, AirUnit, and SeaUnit.

3.55.2.6 calculateSeaDefense()

```
virtual int Soldier::calculateSeaDefense ( )  [pure virtual]
```

Calculates the SeaDefence statistic of the unit. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaDefence statistic of unit

Implements ArmyComponent.

Implemented in LandUnit, AirUnit, and SeaUnit.

3.55.2.7 calculateSeaOffense()

```
virtual int Soldier::calculateSeaOffense ( )  [pure virtual]
```

Calculates the SeaOffense statistic of the unit. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaOffense statistic of unit

Implements ArmyComponent.

Implemented in LandUnit, AirUnit, and SeaUnit.

3.55.3 Member Data Documentation

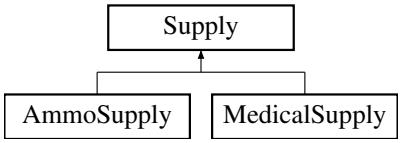
3.55.3.1 trainingLevel

```
int Soldier::trainingLevel  [protected]
```

3.56 Supply Class Reference

```
#include <Supply.h>
```

Inheritance diagram for Supply:



Public Member Functions

- Supply (int quantity)

  Class constructor for the Supply class that wil initialize the quantity member variable.
- virtual ~Supply ()

  Virtual Class destructor to reset member variable.

Protected Attributes

- int quantity

3.56.1 Constructor & Destructor Documentation

3.56.1.1 Supply()

```
Supply::Supply (
            int quantity )  [inline]
```

Class constructor for the Supply class that wil initialize the quantity member variable.

Author

Arno Jooste (u21457451)

Parameters

quantity	The amount that is produced by the factory.

Warning

The quantity must be a value greate than zero.

3.56.1.2 ~Supply()

```
virtual Supply::~Supply ( )  [inline], [virtual]
```

Virtual Class destructor to reset member variable.

Author

Arno Jooste (u21457451)

3.56.2 Member Data Documentation

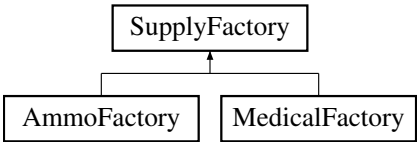
3.56.2.1 quantity

```
int Supply::quantity  [protected]
```

3.57 SupplyFactory Class Reference

```
#include <SupplyFactory.h>
```

Inheritance diagram for SupplyFactory:



Public Member Functions

- SupplyFactory (int budget, std::string type)

  Class constructor for the SuplyFactory which will initialize the budget of the factory as well as set the level to 1.
- virtual ~SupplyFactory ()

  Class destructor to reset the member variables.
- virtual Supply * makeSupply (int quantity)=0

  Factory method to let AmmoFactory and MedicalFactory create the AmmoSupply and MedicalSupply products, respectively.
- void upgrade ()

  Upgrades the factory, which will increase the budget capacity and level.
- void setBudget (int newBudget)

  Sets a new budget for the factory (it will mostly be used by the upgrade() method to increase/set a new budget).
- int getBudget ()

  Getter for the current budget of the factory in order to get access to the private member variable.
- int getLevel ()

  Getter for the current level of the factory in order to get access to the private member variable.
- int getTotalSpent ()

  Getter for the total amount spent so far by the factory. This will be used to test if the factory can produce more supplies based on the budget capacity.
- std::string getType ()

  Getter for the type of factory ,either an Ammo or a medical factory.

Protected Attributes

- double totalSpent

3.57.1 Detailed Description

This class is a necessity if a Country wants to create ammo supplies or medical supplies and send the supplies to the Country's army. There are two types of factories:

- AmmoFactory
- MedicalFactory

3.57.2 Constructor & Destructor Documentation

3.57.2.1 SupplyFactory()

```
SupplyFactory::SupplyFactory (
            int budget,
            std::string type )
```

Class constructor for the SuplyFactory which will initialize the budget of the factory as well as set the level to 1.

Author

Arno Jooste (u21457451)

Parameters

budget	The amount that can be spent to make supplies.
type	The type of supply factory that will be created.

Warning

The budget must be a interger value greater than zero.

3.57.2.2 ~SupplyFactory()

```
SupplyFactory::~SupplyFactory ( )  [virtual]
```

Class destructor to reset the member variables.

Author

Arno Jooste (u21457451)

3.57.3 Member Function Documentation

3.57.3.1 getBudget()

```
int SupplyFactory::getBudget ( )
```

Getter for the current budget of the factory in order to get access to the private member variable.

Author

Arno Jooste (u21457451)

Returns

current budget of type int.

3.57.3.2 getLevel()

```
int SupplyFactory::getLevel ( )
```

Getter for the current level of the factory in order to get access to the private member variable.

Author

Arno Jooste (u21457451)

Returns

current factory level of type int.

3.57.3.3 getTotalSpent()

```
int SupplyFactory::getTotalSpent ( )
```

Getter for the total amount spent so far by the factory. This will be used to test if the factory can produce more supplies based on the budget capacity.

Author

Arno Jooste (u21457451)

Returns

current amount spent of type int.

Note

This helps us to keep track of how much we already spent on supplies.

3.57.3.4 getType()

```
std::string SupplyFactory::getType ( )
```

Getter for the type of factory ,either an Ammo or a medical factory.

Author

Reuben Jooste (u21457060)

Returns

Type of factory

3.57.3.5 makeSupply()

```
virtual Supply* SupplyFactory::makeSupply (
            int quantity )  [pure virtual]
```

Factory method to let AmmoFactory and MedicalFactory create the AmmoSupply and MedicalSupply products, respectively.

Author

Arno Jooste (u21457451)

Returns

pointer to newly created Supply product (it will be either a MedicalSupply or AmmoSupply).

Implemented in MedicalFactory, and AmmoFactory.

3.57.3.6 setBudget()

```
void SupplyFactory::setBudget (
            int newBudget )
```

Sets a new budget for the factory (it will mostly be used by the upgrade() method to increase/set a new budget).

Author

Arno Jooste (u21457451)

Parameters

newBudget	

Warning

The input must be a value greater than zero.

Note

This function only increases the current budget by the given parameter.

3.57.3.7 upgrade()

```
void SupplyFactory::upgrade ( )
```

Upgrades the factory, which will increase the budget capacity and level.

Author

Arno Jooste (u21457451)

3.57.4 Member Data Documentation

3.57.4.1 totalSpent

```
double SupplyFactory::totalSpent  [protected]
```

3.58 This Class Reference

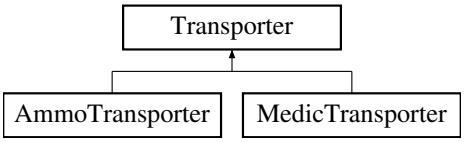
3.58.1 Detailed Description

used to specify in which phase the on going War is.

3.59 Transporter Class Reference

```
#include <Transporter.h>
```

Inheritance diagram for Transporter:



Public Member Functions

- virtual void notify (Corresponder *corresponder)=0

  Notify all Corresponder objects of the changes made by the parameter object. Implemented in derived classes.
- void registerCorresponder (Corresponder *corresponder)

  Register a Corresponder object by adding the passed in object to the list of corresponders.

Protected Attributes

- std::list< Corresponder * > corresponderList

3.59.1 Detailed Description

This class is used to create transport lines for a Country.

There are two types of transport lines:

- AmmoTransporter : the ammo transport line which is used by the Country to send suppplies to its armies.
- MedicTransporter : the medical transport line which is used by the Country to send medical supplies to its armies

Warning

If a country does not have a ammo transport line it cannot send any ammo supplies to its army.
If a country does not have a medical transport line it cannot send any medcial supplies to its army meaning the army will not be able to recuperate during the War.

3.59.2 Member Function Documentation

3.59.2.1 notify()

```
virtual void Transporter::notify (
            Corresponder * corresponder )  [pure virtual]
```

Notify all Corresponder objects of the changes made by the parameter object. Implemented in derived classes.

Author

Reuben Jooste (u21457060)

Parameters

corresponder	pointer to the Corresponder in which a changed has happened.

Implemented in AmmoTransporter, and MedicTransporter.

3.59.2.2 registerCorresponder()

```
void Transporter::registerCorresponder (
            Corresponder * corresponder )
```

Register a Corresponder object by adding the passed in object to the list of corresponders.

Author

Reuben Jooste (u21457060)

Parameters

corresponder	pointer to a Corresponder object which needs to be added to the list of corresponders.

Note

This function allows the Transport lines to know which Country is currently sending supplies to the armies.

3.59.3 Member Data Documentation

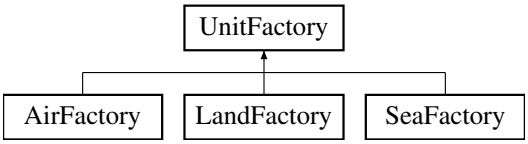
3.59.3.1 corresponderList

```
std::list<Corresponder*> Transporter::corresponderList  [protected]
```

3.60 UnitFactory Class Reference

```
#include <UnitFactory.h>
```

Inheritance diagram for UnitFactory:



Public Member Functions

- UnitFactory (double budget, int level, std::string type)

  Constructor of the UnitFactory class used to instantiate a UnitFactory object.
- virtual ArmyComponent * createVehicle ()=0

  Calls constructor of appropriate Vehicle (Air, Land or Sea), using level to determine powerRating. Implemented in child class.
- virtual ArmyComponent * createSoldier ()=0

  Calls constructor of appropriate Soldier (Air, Land or Sea), using level to determine powerRating. Implemented in child class.
- std::string getType ()

  Getter to get the type of unit factory being used to create products.
- double getTotalSpent ()

  Getter to get the total amount spent so far to determine if we can afford another product.
- double getBudget ()

  Getter to get the Factory's budget.
- void setNewBudget (double newBudget)

  Function to set the new budget of the factory after we upgraded the factory.
- void upgrade ()

  Upgrades the factory, which will increase the budget capacity and level.

Protected Member Functions

- int determineActualLevel ()

  Function to transform Factory's level to a valid value between 1-10. Purpose is to prevent potential bugs from other functions affecting accurate ArmyComponent creation.

Protected Attributes

- double cost
- int level
- double totalSpent

3.60.1 Detailed Description

This class is a necessity if a Country wants to create different units for its army. There are three types of factories based on which type of WarTheatre the country is fighting in:

- AirFactory: This factory will create an AirUnit which includes:
  - Soldiers and Vehicles
- LandFactory: This factory will create an LandUnit which includes:
  - Soldiers and Vehicles
- SeaFactory: This factory will create an SeaUnit which includes:
  - Soldiers and Vehicles

Note

The country will only create units for the specific type of war theatre it is in. In other words air units will only be created for an air-type war theatre etc.

3.60.2 Constructor & Destructor Documentation

3.60.2.1 UnitFactory()

```
UnitFactory::UnitFactory (
            double budget,
            int level,
            std::string type )
```

Constructor of the UnitFactory class used to instantiate a UnitFactory object.

Author

Reuben Jooste (u21457060)

Parameters

budget	The starting budget of the factory
level	The starting level of the factory (all factories start at level one)
type	The type of factory

Warning

The budget must be a interger value greater than zero.
The level must also be an integer value greate than zero.

3.60.3 Member Function Documentation

3.60.3.1 createSoldier()

```
virtual ArmyComponent* UnitFactory::createSoldier ( )  [pure virtual]
```

Calls constructor of appropriate Soldier (Air, Land or Sea), using level to determine powerRating. Implemented in child class.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a Land/Sea/AirUnit)

Implemented in LandFactory, SeaFactory, and AirFactory.

3.60.3.2 createVehicle()

```
virtual ArmyComponent* UnitFactory::createVehicle ( )  [pure virtual]
```

Calls constructor of appropriate Vehicle (Air, Land or Sea), using level to determine powerRating. Implemented in child class.

Author

Luke Lawson (u21433811)

Returns

pointer to newly created ArmyComponent (which will be a Land/Sea/AirVehicle)

Implemented in LandFactory, SeaFactory, and AirFactory.

3.60.3.3 determineActualLevel()

```
int UnitFactory::determineActualLevel ( )  [protected]
```

Function to transform Factory's level to a valid value between 1-10. Purpose is to prevent potential bugs from other functions affecting accurate ArmyComponent creation.

Author

Luke Lawson (u21433811)

Returns

int within range 1-10

3.60.3.4 getBudget()

```
double UnitFactory::getBudget ( )
```

Getter to get the Factory's budget.

Author

Reuben Jooste (u21457060)

Returns

The maximum amount we can spent on creating products

3.60.3.5 getTotalSpent()

```
double UnitFactory::getTotalSpent ( )
```

Getter to get the total amount spent so far to determine if we can afford another product.

Author

Reuben Jooste (u21457060)

Returns

The amount spent so far on creating products

3.60.3.6 getType()

```
std::string UnitFactory::getType ( )
```

Getter to get the type of unit factory being used to create products.

Author

Reuben Jooste (u21457060)

Returns

The type of factory (Air, Sea or Land)

3.60.3.7 setNewBudget()

```
void UnitFactory::setNewBudget (
            double newBudget )
```

Function to set the new budget of the factory after we upgraded the factory.

Author

Reuben Jooste (u21457060)

Parameters

newBudget	The new budget of the factory

Warning

The passed in value must be a value greater than zero.

Note

This function only increases the current budget by the passed in value.

3.60.3.8 upgrade()

```
void UnitFactory::upgrade ( )
```

Upgrades the factory, which will increase the budget capacity and level.

Author

Arno Jooste (u21457451)

3.60.4 Member Data Documentation

3.60.4.1 cost

```
double UnitFactory::cost  [protected]
```

3.60.4.2 level

```
int UnitFactory::level  [protected]
```

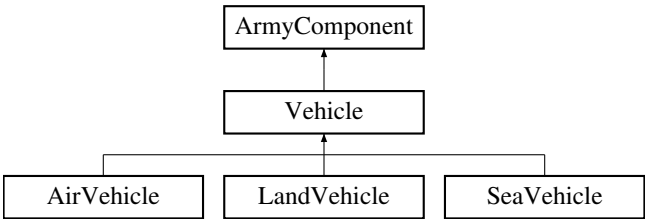
3.60.4.3 totalSpent

```
double UnitFactory::totalSpent  [protected]
```

3.61 Vehicle Class Reference

`#include <Vehicle.h>`

Inheritance diagram for Vehicle:



Public Member Functions

- Vehicle (int powerRating)

  Construct Vehicle using powerRating to determine attribute values.
- virtual int calculateAirOffense ()=0

  Calculates the AirOffense statistic of the vehicle. Implemented in child classes.
- virtual int calculateAirDefense ()=0

  Calculates the AirDefence statistic of the vehicle. Implemented in child classes.
- virtual int calculateSeaOffense ()=0

  Calculates the SeaOffense statistic of the vehicle. Implemented in child classes.
- virtual int calculateSeaDefense ()=0

  Calculates the SeaDefence statistic of the vehicle. Implemented in child classes.
- virtual int calculateLandOffense ()=0

  Calculates the LandOffence statistic of the vehicle. Implemented in child classes.
- virtual int calculateLandDefense ()=0

  Calculates the LandDefence statistic of the vehicle. Implemented in child classes.
- void addMember (ArmyComponent *newMember)

  Composite method to create composite ArmyComponents. Stubbed here.

Protected Attributes

- int armourRating
- int weaponClass

3.61.1 Constructor & Destructor Documentation

3.61.1.1 Vehicle()

```
Vehicle::Vehicle (
            int powerRating )
```

Construct Vehicle using powerRating to determine attribute values.

Author

Luke Lawson (u21433811)

Parameters

powerRating	int used to determine armourRating and weaponClass of Vehicle

3.61.2 Member Function Documentation

3.61.2.1 addMember()

```
void Vehicle::addMember (
            ArmyComponent * newMember )  [virtual]
```

Composite method to create composite ArmyComponents. Stubbed here.

Author

Luke Lawson (u21433811)

Parameters

newMember	pointer to ArmyComponent to add to composite object

Implements ArmyComponent.

3.61.2.2 calculateAirDefense()

```
virtual int Vehicle::calculateAirDefense ( )  [pure virtual]
```

Calculates the AirDefence statistic of the vehicle. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing AirDefence statistic of vehicle

Implements ArmyComponent.

Implemented in LandVehicle, SeaVehicle, and AirVehicle.

3.61.2.3 calculateAirOffense()

```
virtual int Vehicle::calculateAirOffense ( )  [pure virtual]
```

Calculates the AirOffense statistic of the vehicle. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing AirOffense statistic of vehicle

Implements ArmyComponent.

Implemented in LandVehicle, SeaVehicle, and AirVehicle.

3.61.2.4 calculateLandDefense()

```
virtual int Vehicle::calculateLandDefense ( )  [pure virtual]
```

Calculates the LandDefence statistic of the vehicle. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing LandDefence statistic of vehicle

Implements ArmyComponent.

Implemented in LandVehicle, SeaVehicle, and AirVehicle.

3.61.2.5 calculateLandOffense()

```
virtual int Vehicle::calculateLandOffense ( )  [pure virtual]
```

Calculates the LandOffence statistic of the vehicle. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing LandOffence statistic of vehicle

Implements ArmyComponent.

Implemented in LandVehicle, SeaVehicle, and AirVehicle.

3.61.2.6 calculateSeaDefense()

```
virtual int Vehicle::calculateSeaDefense ( )  [pure virtual]
```

Calculates the SeaDefence statistic of the vehicle. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaDefence statistic of vehicle

Implements ArmyComponent.

Implemented in LandVehicle, SeaVehicle, and AirVehicle.

3.61.2.7 calculateSeaOffense()

```
virtual int Vehicle::calculateSeaOffense ( )  [pure virtual]
```

Calculates the SeaOffense statistic of the vehicle. Implemented in child classes.

Author

Luke Lawson (u21433811)

Returns

int value representing SeaOffense statistic of vehicle

Implements ArmyComponent.

Implemented in LandVehicle, SeaVehicle, and AirVehicle.

3.61.3 Member Data Documentation

3.61.3.1 armourRating

```
int Vehicle::armourRating  [protected]
```

3.61.3.2 weaponClass

```
int Vehicle::weaponClass  [protected]
```

3.62 War Class Reference

```
#include <War.h>
```

Public Member Functions

- War ()

  Constructor to initialise a War object and set initialise its WarPhase and War theatres.
- void setWarPhase (WarPhase *)
- void setUpCountries ()

  When called will prompt the user to enter the countries for Alliance 1, Alliance 2 and Neutral Countries.
- WarTheatre * getLandTheatre ()

  Getter for the Land Theatre of the War.
- WarTheatre * getAirTheatre ()

  Getter for the Air Theatre of the War.
- WarTheatre * getSeaTheatre ()

  Getter for the Sea Theatre of the War.
- void changePhase ()

  Method Changes the next corresponding phase of the War Phases.
- void startWarSim ()

  Method to be run if the user desires a uninterruptable war simulation.
- void startWarGame ()

  Method to be run if the user desires a interruptable war simulation, whereby an alliance1's countries decisions are decided by the user.
- void stopWar ()

  Ends the war.

3.62.1 Detailed Description

This class represent the actual war gameplay.

- The war will consist of three different war theatres namely: AirTheatre, LandTheatre and SeaTheatre.
- Throughout the gameplay it is possible to change from one WarPhase to another. The different war phases include:
  - EarlyPhase: This phases is sort of the introduction phase of a war where the tention between two countries is only starting to develop. There are three early phases of the war:
    - EarlyCrisis
    - EarlyOpenConflict
    - EarlyPeace
    - EarlyUnstablePeace
  - MidPhase: In this phase the actual war between two countries starts. We enter this phase once two countries have declared War against one another.
  - LatePhase: Once we enter the late phase the war starts coming to an end. In order end the war one country either needs to surrender against the enemy country or the country needs to be defeated by the enemy country. There are four late phases of the war:
    - LateCrisis
    - LateOpenConflict
    - LatePeace
    - LateUnstablePeace

    Note

    The flow of the war phases will be from early phase -> mid phase -> late phase

3.62.2 Constructor & Destructor Documentation

3.62.2.1 War()

```
War::War ( )
```

Constructor to initialise a War object and set initialise its WarPhase and War theatres.

Author

Thomas Blendulf (u21446131)

3.62.3 Member Function Documentation

3.62.3.1 changePhase()

```
void War::changePhase ( )
```

Method Changes the next corresponding phase of the War Phases.

Author

Thomas Blendulf (u21446131)

3.62.3.2 getAirTheatre()

```
WarTheatre * War::getAirTheatre ( )
```

Getter for the Air Theatre of the War.

Returns

Air WarTheatre*

Author

Thomas Blendulf (u21446131)

3.62.3.3 getLandTheatre()

```
WarTheatre * War::getLandTheatre ( )
```

Getter for the Land Theatre of the War.

Returns

Land WarTheatre*

Author

Thomas Blendulf (u21446131)

3.62.3.4 getSeaTheatre()

```
WarTheatre * War::getSeaTheatre ( )
```

Getter for the Sea Theatre of the War.

Returns

Sea WarTheatre*

Author

Thomas Blendulf (u21446131)

3.62.3.5 setUpCountries()

```
void War::setUpCountries ( )
```

When called will prompt the user to enter the countries for Alliance 1, Alliance 2 and Neutral Countries.

Author

Thomas Blendulf (u21446131)

3.62.3.6 setWarPhase()

```
void War::setWarPhase (
            WarPhase * in )
```

3.62.3.7 startWarGame()

```
void War::startWarGame ( )
```

Method to be run if the user desires a interruptable war simulation, whereby an alliance1's countries decisions are decided by the user.

Author

Thomas Blendulf (u21446131)

3.62.3.8 startWarSim()

```
void War::startWarSim ( )
```

Method to be run if the user desires a uninterruptable war simulation.

Author

Thomas Blendulf (u21446131)

3.62.3.9 stopWar()

```
void War::stopWar ( )
```

Ends the war.

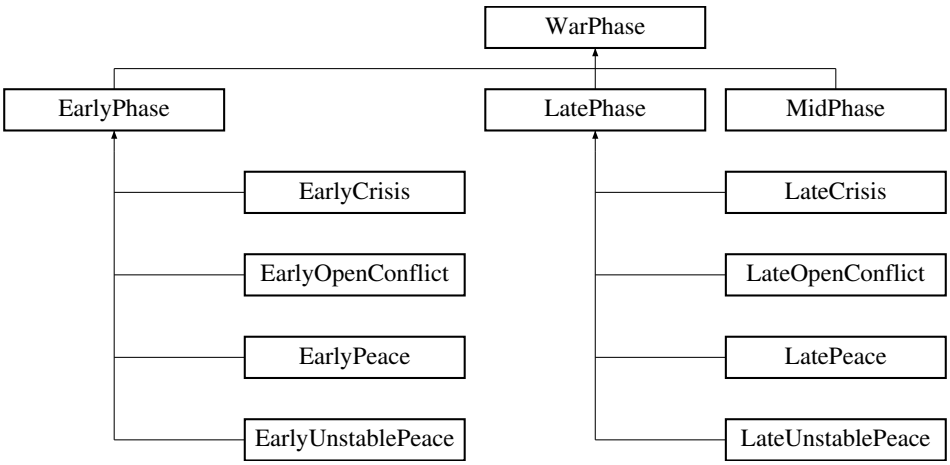
Author

Thomas Blendulf (u21446131)

3.63 WarPhase Class Reference

```
#include <WarPhase.h>
```

Inheritance diagram for WarPhase:



Public Member Functions

- virtual void handleChange (War *war)=0

  This function will be implemented by the derived classes (EarlyPhase, MidPhase & LatePhase)

Public Attributes

- double peaceChance

3.63.1 Member Function Documentation

3.63.1.1 handleChange()

```
virtual void WarPhase::handleChange (
            War * war )  [pure virtual]
```

This function will be implemented by the derived classes (EarlyPhase, MidPhase & LatePhase)

Author

Thomas Blendulf (u21446131)

Parameters

war	The on going war which needs to change its phase

Implemented in EarlyPhase, LatePhase, and MidPhase.

3.63.2 Member Data Documentation

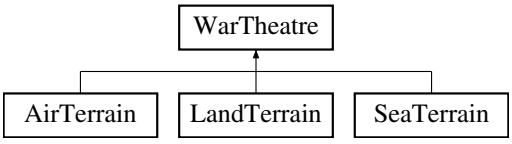
3.63.2.1 peaceChance

```
double WarPhase::peaceChance
```

3.64 WarTheatre Class Reference

```
#include <WarTheatre.h>
```

Inheritance diagram for WarTheatre:



Public Member Functions

- WarTheatre (std::string Type, std::string Name)
  a paramaterized constructor to set the type, name and the medics and civilian vectors
- ~WarTheatre ()
  a destructor to delete the medics and civilian vectors
- void applyTerrainBonus ()
  the template method used to call the appropriate terrain's adjustDefence and adjustAttack methods
- void conflict ()
- void addArmy (Army *newArmy)
  a function to add an army to the war theatre
- void replenishNonCombatEntities ()
  a function to add a fixed number of civilains and medics each time
- std::string getType ()
  a get method to return the type of the war theatre
- std::string getName ()
  a get method to return the name of the war theatre
- Army * getArmies ()
  a get method to return the armies currently present in the war theatre
- int getContentionState ()
  a get method to return the contention state to specify what is happening in the current war theatre.

3.64.1 Detailed Description

This class is used to create different locations where the War will take place.

- A Country can enter its armies to different war theatres. The different war theatres include:
  - AirTerrain: This is where air units will be fighting against each other.
  - LandTerrain: This is where land units will be fighting against each other.
  - SeaTerrain: This is where sea units will be fighting against each other.
- A theatre will also have civilians and medics. This is to make is more realistic to an actual war.
- It will also consist of armies from two alliances. The two alliances will be fighting against each other.

Note

The War will only consists of the three terrains (one of each type of terrain).

3.64.2 Constructor & Destructor Documentation

3.64.2.1 WarTheatre()

```
WarTheatre::WarTheatre (
            std::string Type,
            std::string Name )
```

a paramaterized constructor to set the type, name and the medics and civilian vectors

Author

Jonelle Coertze (u21446271)

Parameters

Type	string used to indicate if the war theatre is a land, sea or air terrain
Name	string used to give the war theatre a name

Warning

The type can only be Sea, Air or Land. Not any other type.

3.64.2.2 ~WarTheatre()

```
WarTheatre::~WarTheatre ( )
```

a destructor to delete the medics and civilian vectors

Author

Jonelle Coertze (u21446271)

3.64.3 Member Function Documentation

3.64.3.1 addArmy()

```
void WarTheatre::addArmy (
            Army * newArmy )
```

a function to add an army to the war theatre

Author

Jonelle Coertze (u21446271)

Parameters

newArmy	pointer to an army object that wants to enter the war theatre

3.64.3.2 applyTerrainBonus()

```
void WarTheatre::applyTerrainBonus ( )
```

the template method used to call the appropriate terrain's adjustDefence and adjustAttack methods

Author

Jonelle Coertze (u21446271)

3.64.3.3 conflict()

```
void WarTheatre::conflict ( )
```

3.64.3.4 getArmies()

```
Army * WarTheatre::getArmies ( )
```

a get method to return the armies currently present in the war theatre

Author

Jonelle Coertze (u21446271)

Returns

pointer to the army array containing the armies currently present in the war theatre

3.64.3.5 getContentionState()

```
int WarTheatre::getContentionState ( )
```

a get method to return the contention state to specify what is happening in the current war theatre.

Author

Jonelle Coertze (u21446271)

Returns

integer value (a value between 0 and 3, both included)

3.64.3.6 getName()

```
std::string WarTheatre::getName ( )
```

a get method to return the name of the war theatre

Author

Jonelle Coertze (u21446271)

Returns

string used to indicate the name of the war theatre

3.64.3.7 getType()

```
std::string WarTheatre::getType ( )
```

a get method to return the type of the war theatre

Author

Jonelle Coertze (u21446271)

Returns

string used to indicate the type(air/land/sea) of the war theatre

3.64.3.8 replenishNonCombatEntities()

```
void WarTheatre::replenishNonCombatEntities ( )
```

a function to add a fixed number of civilains and medics each time

Author

Jonelle Coertze (u21446271)