COS214_Project

version 1.0

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:

class_air_factory-eps-converted-to.pdf

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 Constructor & Destructor Documentation

3.1.1.1 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

3.1.2 Member Function Documentation

3.1.2.1 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)

Implements UnitFactory.

3.1.2.2 createVehicle()

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

Calls constructor of AirVehicle, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

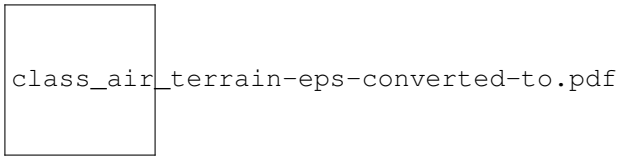
pointer to newly created ArmyComponent (which will be a AirVehicle)

Implements UnitFactory.

3.2 AirTerrain Class Reference

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

Inheritance diagram for AirTerrain:

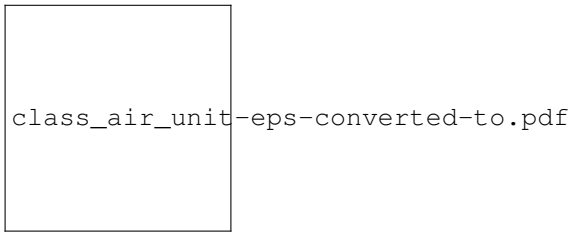


Additional Inherited Members

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 Constructor & Destructor Documentation

3.3.1.1 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)

3.3.2 Member Function Documentation

3.3.2.1 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

Implements Soldier.

3.3.2.2 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

Implements Soldier.

3.3.2.3 calculateLandDefense()

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

Calculates the LandDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.3.2.4 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

Implements Soldier.

3.3.2.5 calculateSeaDefense()

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

Calculates the SeaDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.3.2.6 calculateSeaOffense()

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

Calculates the SeaOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

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 Constructor & Destructor Documentation

3.4.1.1 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)

3.4.2 Member Function Documentation

3.4.2.1 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.

3.4.2.2 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.

3.4.2.3 calculateLandDefense()

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

Calculates the LandDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.4.2.4 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

Implements Vehicle.

3.4.2.5 calculateSeaDefense()

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

Calculates the SeaDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.4.2.6 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

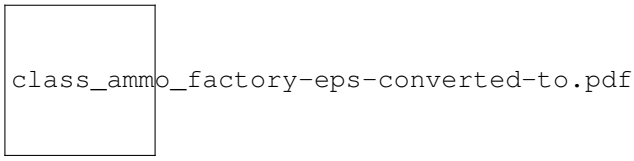
int value representing SeaOffense statistic of vehicle

Implements Vehicle.

3.5 AmmoFactory Class Reference

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

Inheritance diagram for AmmoFactory:



Public Member Functions

- AmmoFactory (int budget)

  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 Constructor & Destructor Documentation

3.5.1.1 AmmoFactory()

```
AmmoFactory::AmmoFactory (
            int budget )
```

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.

3.5.2 Member Function Documentation

3.5.2.1 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

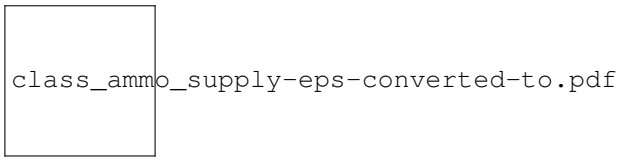
Pointer to newly created AmmoSupply product.

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 Constructor & Destructor Documentation

3.6.1.1 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.

3.6.2 Member Function Documentation

3.6.2.1 getAmmoBonus()

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

Getter for the ammo bonus member variable.

Author

Arno Jooste (u21457451)

Returns

ammo bonus of type int.

3.6.2.2 setAmmoBonus()

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

Setter for the ammo bonus member variable.

Author

Arno Jooste (u21457451)

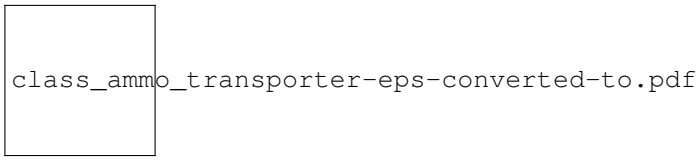
Parameters

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 Constructor & Destructor Documentation

3.7.1.1 AmmoTransporter()

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

Constructor for the AmmoTransporter class used to instantiate the object.

Author

Reuben Jooste (u21457060)

3.7.1.2 ~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)

3.7.2 Member Function Documentation

3.7.2.1 notify()

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

Notify all Corresponder objects in the corresponderList variable.

Author

Reuben Jooste (u21457060)

Parameters

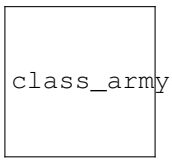
corresponder	pointer to the Corresponder in which a changed has happened.

Implements Transporter.

3.8 Army Class Reference

```
#include <Army.h>
```

Inheritance diagram for Army:



Public Member Functions

- void applyStrategyBonus ()
- void recuperate ()
- void addNewAmmoSupplies (AmmoSupply ∗)
- void addNewMedicalSupplies (MedicalSupply ∗)
- void changeStrategy (std::string)
- void setBattleField (WarTheatre ∗)
- void attackTransport (Country ∗)
- std::string getType ()

Additional Inherited Members

3.8.1 Member Function Documentation

3.8.1.1 addNewAmmoSupplies()

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

3.8.1.2 addNewMedicalSupplies()

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

3.8.1.3 applyStrategyBonus()

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

3.8.1.4 attackTransport()

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

3.8.1.5 changeStrategy()

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

3.8.1.6 getType()

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

3.8.1.7 recuperate()

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

3.8.1.8 setBattleField()

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

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 Constructor & Destructor Documentation

3.9.1.1 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

3.9.2 Member Function Documentation

3.9.2.1 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

3.9.2.2 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

3.9.2.3 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)

3.9.2.4 getArmy()

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

Function to receive the newly constructed Army object.

Author

Reuben Jooste (u21457060)

Returns

Member variable of constructed Army

3.9.2.5 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

3.9.2.6 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

3.9.2.7 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

3.9.2.8 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 Amry object

3.9.2.9 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.

3.9.2.10 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.

3.9.2.11 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.

3.10 ArmyComponent Class Reference

```
#include <ArmyComponent.h>
```

Inheritance diagram for ArmyComponent:

class_army_component-eps-converted-to.pdf

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 Member Function Documentation

3.10.1.1 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.

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

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

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

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

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

3.10.1.7 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 Vehicle, Soldier, AirUnit, AirVehicle, LandUnit, LandVehicle, SeaUnit, SeaVehicle, 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 Constructor & Destructor Documentation

3.11.1.1 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

3.11.2 Member Function Documentation

3.11.2.1 constructArmy()

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

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

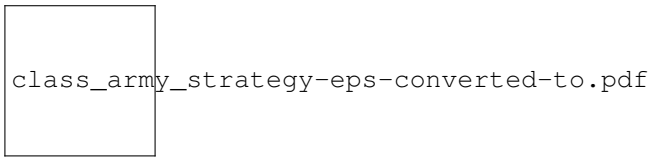
Author

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 Member Function Documentation

3.12.1.1 applyStrategyBonus()

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

Applies desired bonuses to BattleStatistics.

Author

Thomas Blendulf (u21446131)

Parameters

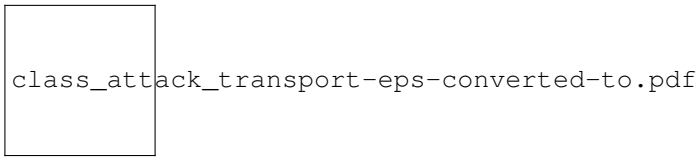
BattleStatistics	passes in the BattleStatistics to be edited.
Battalion	passes in the Battalion to calculate base statistics to be editted.

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 ()

3.14.1 Member Function Documentation

3.14.1.1 decideMyTurn()

```
int Average::decideMyTurn ( ) [virtual]
```

Implements EconomicState.

3.15 Battalion Class Reference

```
#include <Battalion.h>
```

Inheritance diagram for Battalion:

Public Member Functions

- 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 Member Function Documentation

3.15.1.1 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.

3.15.1.2 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.

3.15.1.3 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.

3.15.1.4 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.

3.15.1.5 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.

3.15.1.6 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.

3.15.1.7 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>
```

Friends

- class Defensive
- class Neutral
- class Offensive

3.16.1 Friends And Related Function Documentation

3.16.1.1 Defensive

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

3.16.1.2 Neutral

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

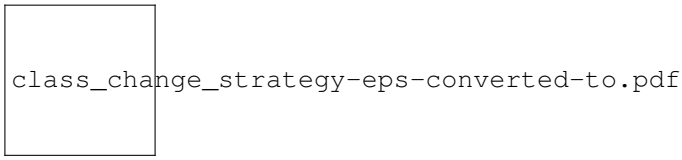
3.16.1.3 Offensive

```
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

- NonCombatEntity * clone ()

3.18.1 Member Function Documentation

3.18.1.1 clone()

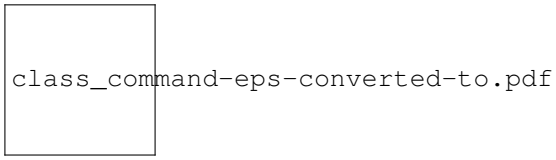
```
NonCombatEntity * Civilian::clone ( ) [virtual]
```

Implements NonCombatEntity.

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, MoveIntoTheatre, and AttackTransport.

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:

class_corresponder-eps-converted-to.pdf

Public Member Functions

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

Protected Attributes

- Transporter * medicalTransportLine
- Transporter * ammoTransportLine

3.20.1 Member Function Documentation

3.20.1.1 regToTransport()

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

3.20.2 Member Data Documentation

3.20.2.1 ammoTransportLine

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

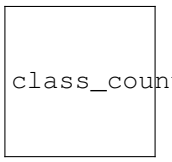
3.20.2.2 medicalTransportLine

```
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 Constructor & Destructor Documentation

3.21.1.1 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

3.21.1.2 ~Country()

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

Destructor to deallocate any dynamic memory involved.

Author

Luke Lawson (u21433811)

3.21.2 Member Function Documentation

3.21.2.1 attackTransport()

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

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

Author

Luke Lawson (u21433811)

3.21.2.2 changeArmyStrategy()

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

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

Author

Luke Lawson (u21433811)

3.21.2.3 destroyTransport()

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

Function to set this Country's Transport to NULL.

Author

Luke Lawson (u21433811)

3.21.2.4 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

3.21.2.5 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

3.21.2.6 formAlliance()

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

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

Author

Luke Lawson (u21433811)

3.21.2.7 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

3.21.2.8 getName()

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

Getter for the Country name.

Author

Luke Lawson (u21433811)

Returns

string name of the Country

3.21.2.9 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

3.21.2.10 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

3.21.2.11 isSurrendered()

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

Gets whether Country has surrender from the war.

Author

Luke Lawson (u21433811)

Returns

boolean value of hasSurrendered

3.21.2.12 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)

3.21.2.13 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

3.21.2.14 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

3.21.2.15 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

3.21.2.16 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

3.21.2.17 surrender()

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

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

Author

Luke Lawson (u21433811)

3.21.2.18 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

3.21.2.19 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)

3.21.2.20 upgradeUnitFactory()

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

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

Author

Luke Lawson (u21433811)

3.21.3 Member Data Documentation

3.21.3.1 alliance1

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

3.21.3.2 alliance2

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

3.21.3.3 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:

Additional Inherited Members

3.24 EarlyOpenConflict Class Reference

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

Inheritance diagram for EarlyOpenConflict:

class_early_open_conflict-eps-converted-to.pdf

Additional Inherited Members

3.25 EarlyPhase Class Reference

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

Inheritance diagram for EarlyPhase:

class_early_phase-eps-converted-to.pdf

Public Member Functions

- void handleChange ()

Public Attributes

- EarlyPhase * next

Additional Inherited Members

3.25.1 Member Function Documentation

3.25.1.1 handleChange()

```
void EarlyPhase::handleChange ( )
```

3.25.2 Member Data Documentation

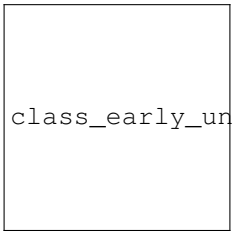
3.25.2.1 next

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

3.26 EarlyUnstablePeace Class Reference

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

Inheritance diagram for EarlyUnstablePeace:

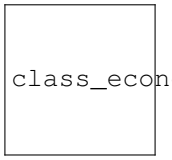


Additional Inherited Members

3.27 EconomicState Class Reference

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

Inheritance diagram for EconomicState:



Public Member Functions

- virtual int decideMyTurn ()=0

3.27.1 Member Function Documentation

3.27.1.1 decideMyTurn()

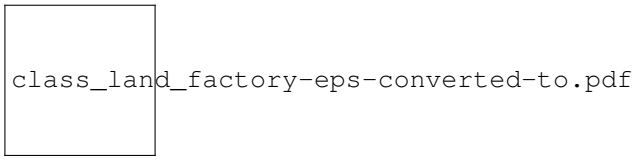
```
virtual int EconomicState::decideMyTurn ( )  [pure virtual]
```

Implemented in Average, Poor, and Rich.

3.28 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

3.28.1 Constructor & Destructor Documentation

3.28.1.1 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

3.28.2 Member Function Documentation

3.28.2.1 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)

Implements UnitFactory.

3.28.2.2 createVehicle()

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

Calls constructor of LandVehicle, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

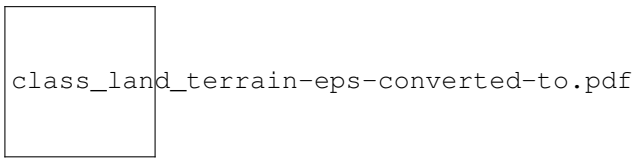
pointer to newly created ArmyComponent (which will be a LandVehicle)

Implements UnitFactory.

3.29 LandTerrain Class Reference

```
#include <LandTerrain.h>
```

Inheritance diagram for LandTerrain:

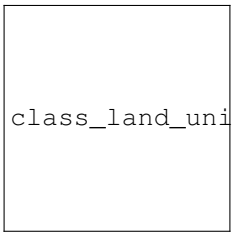


Additional Inherited Members

3.30 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

3.30.1 Constructor & Destructor Documentation

3.30.1.1 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)

3.30.2 Member Function Documentation

3.30.2.1 calculateAirDefense()

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

Calculates the AirDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.30.2.2 calculateAirOffense()

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

Calculates the AirOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.30.2.3 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

Implements Soldier.

3.30.2.4 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

Implements Soldier.

3.30.2.5 calculateSeaDefense()

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

Calculates the SeaDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.30.2.6 calculateSeaOffense()

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

Calculates the SeaOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.31 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

3.31.1 Constructor & Destructor Documentation

3.31.1.1 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)

3.31.2 Member Function Documentation

3.31.2.1 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

Implements Vehicle.

3.31.2.2 calculateAirOffense()

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

Calculates the AirOffense statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.31.2.3 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

Implements Vehicle.

3.31.2.4 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

Implements Vehicle.

3.31.2.5 calculateSeaDefense()

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

Calculates the SeaDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.31.2.6 calculateSeaOffense()

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

Calculates the SeaOffense statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.32 LateCrisis Class Reference

```
#include <LateCrisis.h>
```

Inheritance diagram for LateCrisis:

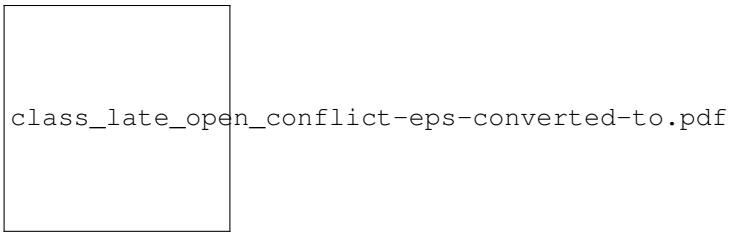
class_late_crisis-eps-converted-to.pdf

Additional Inherited Members

3.33 LateOpenConflict Class Reference

```
#include <LateOpenConflict.h>
```

Inheritance diagram for LateOpenConflict:

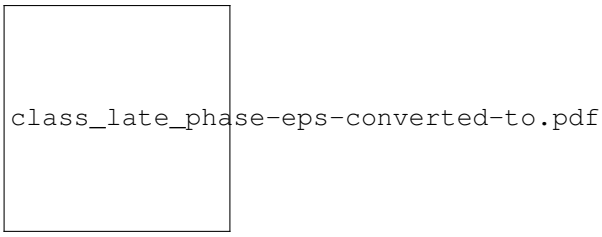


Additional Inherited Members

3.34 LatePhase Class Reference

```
#include <LatePhase.h>
```

Inheritance diagram for LatePhase:



Public Member Functions

- void handleChange ()

Public Attributes

- LatePhase * next

Additional Inherited Members

3.34.1 Member Function Documentation

3.34.1.1 handleChange()

```
void LatePhase::handleChange ( )
```

3.34.2 Member Data Documentation

3.34.2.1 next

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

3.35 LateUnstablePeace Class Reference

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

Inheritance diagram for LateUnstablePeace:

class_late_unstable_peace-eps-converted-to.pdf

Additional Inherited Members

3.36 Medic Class Reference

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

Inheritance diagram for Medic:

class_medic-eps-converted-to.pdf

Public Member Functions

- NonCombatEntity * clone ()

3.36.1 Member Function Documentation

3.36.1.1 clone()

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

Implements NonCombatEntity.

3.37 MedicalFactory Class Reference

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

Inheritance diagram for MedicalFactory:

class_medical_factory-eps-converted-to.pdf

Public Member Functions

- MedicalFactory (int budget)

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

  Creates medical supplies by creating a new MedicalSupply product.

Additional Inherited Members

3.37.1 Constructor & Destructor Documentation

3.37.1.1 MedicalFactory()

```
MedicalFactory::MedicalFactory (
            int budget )
```

Class constructor for MedicalFactory.

Author

Arno Jooste (u21457451)

Parameters

budget	The amount that can be spent to make medical supplies.

3.37.2 Member Function Documentation

3.37.2.1 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

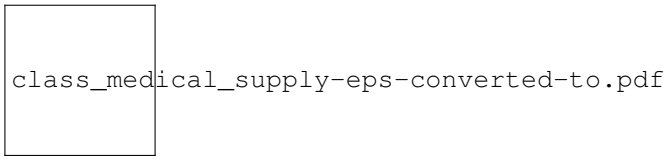
Pointer to newly created MedicalSupply product.

Implements SupplyFactory.

3.38 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

3.38.1 Constructor & Destructor Documentation

3.38.1.1 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

3.38.2 Member Function Documentation

3.38.2.1 getMedicalBonus()

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

Getter for the medical bonus member variable.

Author

Arno Jooste (u21457451)

Returns

medical bonus of type int.

3.38.2.2 setMedicalBonus()

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

Setter for the medical bonus member variable.

Author

Arno Jooste (u21457451)

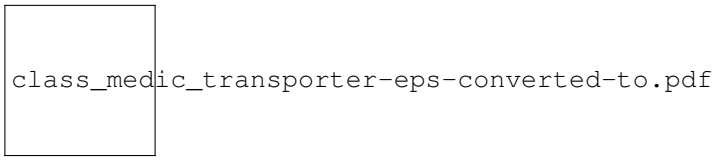
Parameters

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

3.39 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

3.39.1 Constructor & Destructor Documentation

3.39.1.1 MedicTransporter()

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

Constructor for the MedicTransport class used to instantiate the object.

Author

Reuben Jooste (u21457060)

3.39.1.2 ~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)

3.39.2 Member Function Documentation

3.39.2.1 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.

Implements Transporter.

3.40 MidPhase Class Reference

```
#include <MidPhase.h>
```

Inheritance diagram for MidPhase:

Public Member Functions

- void handleChange ()

Additional Inherited Members

3.40.1 Member Function Documentation

3.40.1.1 handleChange()

```
void MidPhase::handleChange ( )
```

3.41 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.

3.41.1 Constructor & Destructor Documentation

3.41.1.1 MilitaryCommander()

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

3.41.2 Member Function Documentation

3.41.2.1 attackTransport()

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

executes the enterTheatre Command.

Author

Thomas Blendulf(u21446131)

3.41.2.2 changeStrategy()

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

executes the changeStrategy Command.

Author

Thomas Blendulf(u21446131)

3.41.2.3 enterTheatre()

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

executes the enterTheatre Command.

Author

Thomas Blendulf(u21446131)

3.41.2.4 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.

3.41.2.5 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.

3.41.2.6 setTransportTarget()

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

sets variables of the attackTransport Command.

Author

Thomas Blendulf(u21446131)

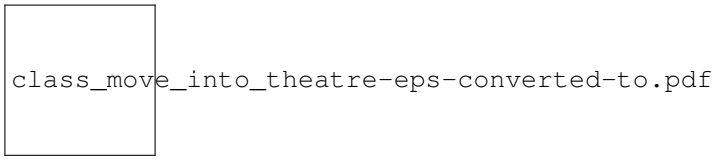
Parameters

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

3.42 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

3.42.1 Member Function Documentation

3.42.1.1 execute()

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

sets the stored armies war theatre to fight in.

Author

Thomas Blendulf(u21446131)

Implements Command.

3.42.1.2 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.

3.42.2 Member Data Documentation

3.42.2.1 theatre

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

3.43 Neutral Class Reference

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

Inheritance diagram for Neutral:

class_neutral-eps-converted-to.pdf

Public Member Functions

- void applyStrategyBonus (BattleStatistics, Battalion)

  Applies desired Neutral bonuses to BattleStatistics.

3.43.1 Member Function Documentation

3.43.1.1 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.

3.44 NonCombatEntity Class Reference

```
#include <NonCombatEntity.h>
```

Inheritance diagram for NonCombatEntity:

class_non_combat_entity-eps-converted-to.pdf

Public Member Functions

- virtual NonCombatEntity * clone ()=0

3.44.1 Member Function Documentation

3.44.1.1 clone()

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

Implemented in Civilian, and Medic.

3.45 Offensive Class Reference

```
#include <Offensive.h>
```

Inheritance diagram for Offensive:

class_offensive-eps-converted-to.pdf

Public Member Functions

- void applyStrategyBonus (BattleStatistics, Battalion)

  Applies desired Offensive bonuses to BattleStatistics.

3.45.1 Member Function Documentation

3.45.1.1 applyStrategyBonus()

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

Applies desired Offensive bonuses to BattleStatistics.

Author

Thomas Blendulf (u21446131)

Parameters

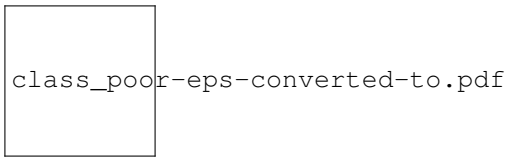
BattleStatistics	passes in the BattleStatistics to be edited.
Battalion	passes in the Battalion to calculate base statistics to be editted.

Reimplemented from ArmyStrategy.

3.46 Poor Class Reference

```
#include <Poor.h>
```

Inheritance diagram for Poor:



Public Member Functions

- int decideMyTurn ()

3.46.1 Member Function Documentation

3.46.1.1 decideMyTurn()

```
int Poor::decideMyTurn ( )  [virtual]
```

Implements EconomicState.

3.47 Rich Class Reference

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

Inheritance diagram for Rich:

class_rich-eps-converted-to.pdf

Public Member Functions

- int decideMyTurn ()

3.47.1 Member Function Documentation

3.47.1.1 decideMyTurn()

```
int Rich::decideMyTurn ( )  [virtual]
```

Implements EconomicState.

3.48 SeaFactory Class Reference

```
#include <SeaFactory.h>
```

Inheritance diagram for SeaFactory:

class_sea_factory-eps-converted-to.pdf

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

3.48.1 Constructor & Destructor Documentation

3.48.1.1 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

3.48.2 Member Function Documentation

3.48.2.1 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)

Implements UnitFactory.

3.48.2.2 createVehicle()

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

Calls constructor of SeaVehicle, using level to determine powerRating.

Author

Luke Lawson (u21433811)

Returns

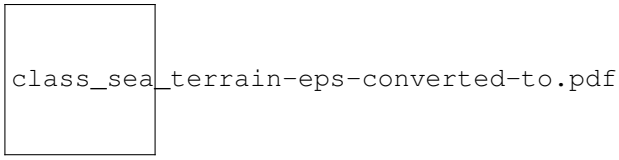
pointer to newly created ArmyComponent (which will be a SeaVehicle)

Implements UnitFactory.

3.49 SeaTerrain Class Reference

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

Inheritance diagram for SeaTerrain:

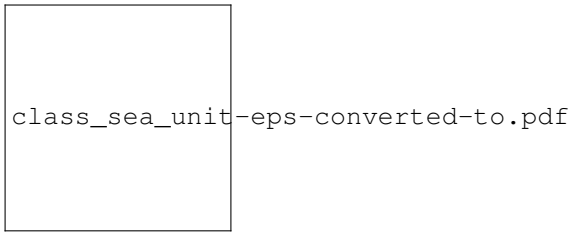


Additional Inherited Members

3.50 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

3.50.1 Constructor & Destructor Documentation

3.50.1.1 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)

3.50.2 Member Function Documentation

3.50.2.1 calculateAirDefense()

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

Calculates the AirDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.50.2.2 calculateAirOffense()

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

Calculates the AirOffense statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.50.2.3 calculateLandDefense()

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

Calculates the LandDefence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.50.2.4 calculateLandOffense()

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

Calculates the LandOffence statistic of the unit.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Soldier.

3.50.2.5 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

Implements Soldier.

3.50.2.6 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

Implements Soldier.

3.51 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

3.51.1 Constructor & Destructor Documentation

3.51.1.1 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)

3.51.2 Member Function Documentation

3.51.2.1 calculateAirDefense()

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

Calculates the AirDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.51.2.2 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

Implements Vehicle.

3.51.2.3 calculateLandDefense()

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

Calculates the LandDefence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.51.2.4 calculateLandOffense()

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

Calculates the LandOffence statistic of the vehicle.

Author

Luke Lawson (u21433811)

Returns

0 (no capability)

Implements Vehicle.

3.51.2.5 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

Implements Vehicle.

3.51.2.6 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

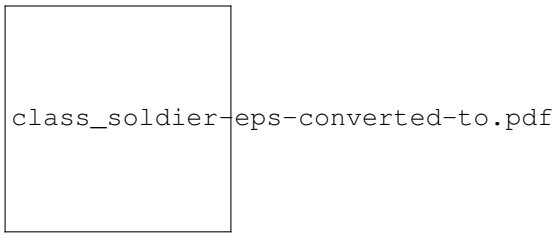
int value representing SeaOffense statistic of vehicle

Implements Vehicle.

3.52 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.
- double getSoldierCost ()

  Function to get the cost of creating a soldier object.

Protected Attributes

- int trainingLevel
- double soldierCost

3.52.1 Constructor & Destructor Documentation

3.52.1.1 Soldier()

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

Construct Solider using powerLevel to determine trainingLevel.

Author

Luke Lawson (u21433811)

Parameters

powerRating	powerRating of the Soldier (powerRating = trainingLevel)

3.52.2 Member Function Documentation

3.52.2.1 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.

3.52.2.2 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 AirUnit, LandUnit, and SeaUnit.

3.52.2.3 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 AirUnit, LandUnit, and SeaUnit.

3.52.2.4 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.

3.52.2.5 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 AirUnit, LandUnit, and SeaUnit.

3.52.2.6 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 AirUnit, LandUnit, and SeaUnit.

3.52.2.7 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 AirUnit, LandUnit, and SeaUnit.

3.52.2.8 getSoldierCost()

```
double Soldier::getSoldierCost ( )
```

Function to get the cost of creating a soldier object.

Author

Reuben Jooste (u21457060)

Returns

The member variable, soldierCost

3.52.3 Member Data Documentation

3.52.3.1 soldierCost

```
double Soldier::soldierCost [protected]
```

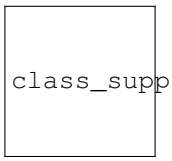
3.52.3.2 trainingLevel

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

3.53 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

3.53.1 Constructor & Destructor Documentation

3.53.1.1 Supply()

```
Supply::Supply (
            int quantity )
```

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.

3.53.1.2 ~Supply()

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

Virtual Class destructor to reset member variable.

Author

Arno Jooste (u21457451)

3.53.2 Member Data Documentation

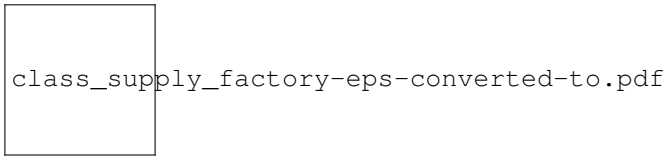
3.53.2.1 quantity

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

3.54 SupplyFactory Class Reference

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

Inheritance diagram for SupplyFactory:



Public Member Functions

- SupplyFactory (int budget)

  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

3.54.1 Constructor & Destructor Documentation

3.54.1.1 SupplyFactory()

```
SupplyFactory::SupplyFactory (
            int budget )
```

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.

3.54.1.2 ~SupplyFactory()

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

Class destructor to reset the member variables.

Author

Arno Jooste (u21457451)

3.54.2 Member Function Documentation

3.54.2.1 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.

3.54.2.2 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.

3.54.2.3 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.

3.54.2.4 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

3.54.2.5 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.

3.54.2.6 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	

3.54.2.7 upgrade()

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

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

Author

Arno Jooste (u21457451)

3.54.3 Member Data Documentation

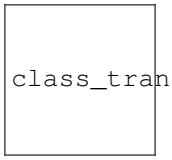
3.54.3.1 totalSpent

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

3.55 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

3.55.1 Member Function Documentation

3.55.1.1 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.

3.55.1.2 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.

3.55.2 Member Data Documentation

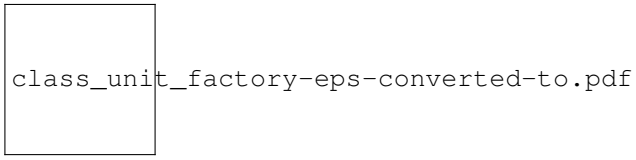
3.55.2.1 corresponderList

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

3.56 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 increaseTotalSpent (double cost)

  Function to increase the total spent after we created a product.
- void setNewBudget (double newBudget)

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

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

3.56.1 Constructor & Destructor Documentation

3.56.1.1 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

3.56.2 Member Function Documentation

3.56.2.1 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 AirFactory, LandFactory, and SeaFactory.

3.56.2.2 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 AirFactory, LandFactory, and SeaFactory.

3.56.2.3 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

3.56.2.4 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

3.56.2.5 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

3.56.2.6 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)

3.56.2.7 increaseTotalSpent()

```
void UnitFactory::increaseTotalSpent (
            double cost )
```

Function to increase the total spent after we created a product.

Author

Reuben Jooste (u21457060)

Parameters

cost	Parameter to increase the current total amount spent by

3.56.2.8 setNewBudget()

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

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

Author

Reuben Jooste (u21457060)

Parameters

newBudget	The new budget of the factory

3.56.2.9 upgrade()

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

3.56.3 Member Data Documentation

3.56.3.1 cost

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

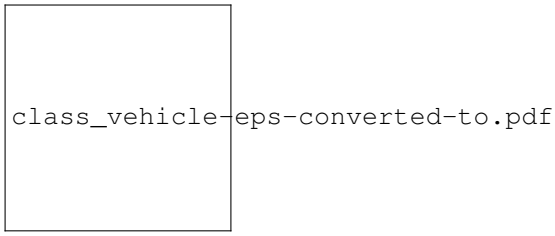
3.56.3.2 level

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

3.57 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.
- double getVehicleCost ()

  Function to get teh cost of creating one Vehicle object.

Protected Attributes

- int armourRating
- int weaponClass
- double vehicleCost

3.57.1 Constructor & Destructor Documentation

3.57.1.1 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

3.57.2 Member Function Documentation

3.57.2.1 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.

3.57.2.2 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 AirVehicle, LandVehicle, and SeaVehicle.

3.57.2.3 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 AirVehicle, LandVehicle, and SeaVehicle.

3.57.2.4 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 AirVehicle, LandVehicle, and SeaVehicle.

3.57.2.5 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 AirVehicle, LandVehicle, and SeaVehicle.

3.57.2.6 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 AirVehicle, LandVehicle, and SeaVehicle.

3.57.2.7 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 AirVehicle, LandVehicle, and SeaVehicle.

3.57.2.8 getVehicleCost()

```
double Vehicle::getVehicleCost ( )
```

Function to get teh cost of creating one Vehicle object.

Author

Reuben Jooste (u21457060)

Returns

The cost of one Vehicle object

3.57.3 Member Data Documentation

3.57.3.1 armourRating

```
int Vehicle::armourRating [protected]
```

3.57.3.2 vehicleCost

```
double Vehicle::vehicleCost [protected]
```

3.57.3.3 weaponClass

```
int Vehicle::weaponClass [protected]
```

3.58 War Class Reference

```
#include <War.h>
```

Public Member Functions

- void setupTheatres ()
- WarTheatre * getLandTheatre ()
- WarTheatre * getAirTheatre ()
- WarTheatre * getSeaTheatre ()
- void changePhase ()
- void startWarSim ()
- void startWarGame ()
- void stopWar ()

3.58.1 Member Function Documentation

3.58.1.1 changePhase()

```
void War::changePhase ( )
```

3.58.1.2 getAirTheatre()

```
WarTheatre* War::getAirTheatre ( )
```

3.58.1.3 getLandTheatre()

```
WarTheatre* War::getLandTheatre ( )
```

3.58.1.4 getSeaTheatre()

```
WarTheatre* War::getSeaTheatre ( )
```

3.58.1.5 setupTheatres()

```
void War::setupTheatres ( )
```

3.58.1.6 startWarGame()

```
void War::startWarGame ( )
```

3.58.1.7 startWarSim()

```
void War::startWarSim ( )
```

3.58.1.8 stopWar()

```
void War::stopWar ( )
```

3.59 WarPhase Class Reference

```
#include <WarPhase.h>
```

Inheritance diagram for WarPhase:

class_war_phase-eps-converted-to.pdf

Public Member Functions

- void handleChange ()

Protected Attributes

- double peaceChance

3.59.1 Member Function Documentation

3.59.1.1 handleChange()

```
void WarPhase::handleChange ( )
```

3.59.2 Member Data Documentation

3.59.2.1 peaceChance

```
double WarPhase::peaceChance  [protected]
```

3.60 WarTheatre Class Reference

```
#include <WarTheatre.h>
```

Inheritance diagram for WarTheatre:

class_war_theatre-eps-converted-to.pdf

Public Member Functions

- void applyTerrainBonus ()
- void conflict ()
- void addArmy (Army *newArmy)
- void replenishNonCombatEntities ()
- std::string getType ()
- std::string getName ()

3.60.1 Member Function Documentation

3.60.1.1 addArmy()

```
void WarTheatre::addArmy (
            Army * newArmy )
```

3.60.1.2 applyTerrainBonus()

```
void WarTheatre::applyTerrainBonus ( )
```

3.60.1.3 conflict()

```
void WarTheatre::conflict ( )
```

3.60.1.4 getName()

```
std::string WarTheatre::getName ( )
```

3.60.1.5 getType()

```
std::string WarTheatre::getType ( )
```

3.60.1.6 replenishNonCombatEntities()

```
void WarTheatre::replenishNonCombatEntities ( )
```