GangOfSix Documentation

1.0

Generated by Doxygen 1.9.5

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

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AirCraft Class Reference

```
#include <AirCraft.h>
```

Public Member Functions

- AirCraft (int d, int h)
- AirCraft ∗ clone ()

  Returns a clone of the current object and assigns it to a new object.
- void doNotting ()

4.1.1 Constructor & Destructor Documentation

4.1.1.1 AirCraft()

```
AirCraft::AirCraft (
             int d,
             int h )
```

4.1.2 Member Function Documentation

4.1.2.1 clone()

```
AirCraft * AirCraft::clone ( )
```

Returns a clone of the current object and assigns it to a new object.

Return values

An	Aircraft object is returned.

4.1.2.2 doNotting()

```
void AirCraft::doNotting ( )  [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- AirCraft.h

4.2 AirSpace Class Reference

```
#include <AirSpace.h>
```

The documentation for this class was generated from the following file:

- AirSpace.h

4.3 Alliance Class Reference

```
#include <Alliance.h>
```

Public Member Functions

- Alliance ()
- void setAlliance (vector< Country ∗ > alliance)

  This function sets the alliance vector to the one provided in the parameter.
- vector< Country ∗ > getAlliance ()

  The function returns an alliance vector array.
- void addAlly (Country ∗ally)

  The function is used to add a Country to an Alliance vector.
- void removeAlly (Country ∗ally)

  The function removes a Country from an Alliance vector.
- int getHp ()

  A function to get the current allianceHp.

Additional Inherited Members

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Alliance()

```
Alliance::Alliance ( )
```

4.3.2 Member Function Documentation

4.3.2.1 addAlly()

```
void Alliance::addAlly (
             Country * ally )
```

The function is used to add a Country to an Alliance vector.

Parameters

in	*ally*	A Country that is being added to an Alliance vector

4.3.2.2 getAlliance()

```
vector< Country * > Alliance::getAlliance ( )
```

The function returns an alliance vector array.

Return values

An	alliance vector is returned.

4.3.2.3 getHp()

```
int Alliance::getHp ( )  [virtual]
```

A function to get the current allianceHp.

Return values

a	value stored in allianceHp member variable

Implements Country.

4.3.2.4 removeAlly()

```
void Alliance::removeAlly (
            Country * ally )
```

The function removes a Country from an Alliance vector.

Parameters

ally	The country to be removed from an Alliance vector

4.3.2.5 setAlliance()

```
void Alliance::setAlliance (
            vector< Country * > alliance )
```

This function sets the alliance vector to the one provided in the parameter.

Parameters

in	*alliance*	A vector that will be stored in the alliance member variable.

The documentation for this class was generated from the following file:

- Alliance.h

4.4 Attack Class Reference

```
#include <Attack.h>
```

Public Member Functions

- void handleChange (Country *C)

  The function checks if an attack can be made based on the hp of the opposing country.

Additional Inherited Members

4.4.1 Member Function Documentation

4.4.1.1 handleChange()

```
void Attack::handleChange (
            Country * C )  [virtual]
```

The function checks if an attack can be made based on the hp of the opposing country.

Parameters

in	*C*	The opposing Country an attack can be conducted on.

Reimplemented from BattleState.

The documentation for this class was generated from the following file:

- Attack.h

4.5 AttackStrategy Class Reference

```
#include <AttackStrategy.h>
```

Public Member Functions

- virtual void LaunchAttack (Country ∗C)=0

    The function is used to launch an attack on the Country provided in the parameter.

4.5.1 Member Function Documentation

4.5.1.1 LaunchAttack()

```
virtual void AttackStrategy::LaunchAttack (
            Country * C )  [pure virtual]
```

The function is used to launch an attack on the Country provided in the parameter.

Parameters

in	*C*	The Country to attack.

Implemented in DetonateExplosives, FireMissile, and Shoot.

The documentation for this class was generated from the following file:

- AttackStrategy.h

4.6 BattleState Class Reference

```
#include <BattleState.h>
```

Public Member Functions

- BattleState ()
- void Add (BattleState ∗Succ)

  The function adds Succ to the successor member variable.
- virtual void handleChange (Country ∗C)

Public Attributes

- BattleState ∗ successor

4.6.1 Constructor & Destructor Documentation

4.6.1.1 BattleState()

```
BattleState::BattleState ( )
```

4.6.2 Member Function Documentation

4.6.2.1 Add()

```
void BattleState::Add (
            BattleState ∗ Succ )
```

The function adds Succ to the successor member variable.

Parameters

in	*Succ*	a BattleState pointer added to successor

4.6.2.2 handleChange()

```
virtual void BattleState::handleChange (
            Country * C )  [virtual]
```

Reimplemented in Attack, Defend, RequestAlliance, and Surrender.

4.6.3 Member Data Documentation

4.6.3.1 successor

```
BattleState* BattleState::successor
```

The documentation for this class was generated from the following file:

- BattleState.h

4.7 Bomb Class Reference

```
#include <Bomb.h>
```

Public Member Functions

- Bomb (int d, int h)
- Bomb * clone ()

  The function creates a copy of the current object.
- void doNotting ()

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Bomb()

```
Bomb::Bomb (
            int d,
            int h )
```

4.7.2 Member Function Documentation

4.7.2.1 clone()

```
Bomb * Bomb::clone ( )
```

The function creates a copy of the current object.

4.7.2.2 doNotting()

```
void Bomb::doNotting ( )  [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- Bomb.h

4.8 Context Class Reference

```
#include <Context.h>
```

Public Member Functions

- Context (AttackStrategy ∗s)
- ∼Context ()
- void SetState (AttackStrategy ∗s)

  The functions assigns s to the state member variable.
- void implement (Country ∗C)

  A function calling the LaunchAttack() member of state to launch an attack on a Country.

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Context()

```
Context::Context (
            AttackStrategy * s )
```

4.8.1.2 ∼Context()

```
Context::∼Context ( )
```

4.8.2 Member Function Documentation

4.8.2.1 implement()

```
void Context::implement (
             Country ∗ C )
```

A function calling the LaunchAttack() member of state to launch an attack on a Country.

Parameters

in	*C*	A country an attack is being launched on

4.8.2.2 SetState()

```
void Context::SetState (
             AttackStrategy ∗ s )
```

The functions assigns s to the state member variable.

Parameters

in	*s*	A variable assigned to state member variable

The documentation for this class was generated from the following file:

- Context.h

4.9 Country Class Reference

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

Public Member Functions

- Country (std::string, bool)
- Country (BattleState ∗state)

- Country (Country ∗C)
- Country ()
- BattleState ∗ getBattleState ()

  returns the current state of a country in the battle.
- Country ∗ pickOpposingCountry (vector< Country ∗ > c)

  This function is used to pick up a opposing Country from the vector provided.
- virtual void withdraw ()
- void setBattleState (BattleState ∗b)

  The function is used to set the Country's BattleState.
- void selectWarTheatre ()

  Function used to select the area where a battle should take place at.
- void createWarParticipants ()

  A function used to create personeel and weapons that will be used in the war.
- void attackOpposingCountry (Country ∗c)

  A function used to initiate an attack on an opposing Country.
- virtual int getHp ()=0
- CountryBackup ∗ createBackup ()

  A function used to store a country's member variable values at a certain point of the war.
- void reinstateCountry (CountryBackup ∗cb)

  a function to take the country back to a certain point during the war.
- virtual WarParticipantIterator ∗ createWarParticipantIterator ()

  The function creates an iterator to traverse the vector of war participants.
- virtual void addWarParticipant (WarParticipant ∗wp)

  A function used to add a war participant element to the war participant vector.
- virtual std::vector< CountryObserver ∗ > getCountryObservers ()

  The function is used to access all the observers observing the country.
- virtual std::string getName ()

  A function used to get the name of the country.
- WarTheatre ∗ getWarTheatre ()

  The function is used to get the area in which a battle will be taking place.
- virtual bool add (CountryObserver ∗c)

  A function used to add an observer to the vector of observes in a country.
- virtual bool remove (CountryObserver ∗c)

  A function used to stop an observer from observing the country.
- virtual void notify ()
- void InflictDamage (int dmg)
- Country ∗ getOpposingC ()
- void setHp (int HP)
- vector< WarParticipant ∗ > getArtillery ()
- void addArtillery (WarParticipant ∗W)
- virtual ∼Country ()

Protected Attributes

- int hp =1000

4.9.1 Constructor & Destructor Documentation

4.9.1.1 Country() [1/4]

```
Country::Country (
             std::string ,
             bool  )
```

4.9.1.2 Country() [2/4]

```
Country::Country (
             BattleState * state )
```

4.9.1.3 Country() [3/4]

```
Country::Country (
             Country * C )
```

4.9.1.4 Country() [4/4]

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

4.9.1.5 ∼Country()

```
virtual Country::∼Country ( )  [virtual]
```

4.9.2 Member Function Documentation

4.9.2.1 add()

```
virtual bool Country::add (
             CountryObserver * c )  [virtual]
```

A function used to add an observer to the vector of observes in a country.

Parameters

in	*c*	an observer to be added to the observer vector.

Return values

returns	TRUE if the observer was added successfully and FALSE if the obsever was not added.

Reimplemented in IndividualCountry.

4.9.2.2 addArtillery()

```
void Country::addArtillery (
            WarParticipant * W )
```

4.9.2.3 addWarParticipant()

```
virtual void Country::addWarParticipant (
            WarParticipant * wp )  [virtual]
```

A function used to add a war participant element to the war participant vector.

Parameters

in	*wp*	A war participant pointer added to the WarParicipant vector.

4.9.2.4 attackOpposingCountry()

```
void Country::attackOpposingCountry (
            Country * c )
```

A function used to initiate an attack on an opposing Country.

Parameters

in	*c*	A country an attack will be directed to.

4.9.2.5 createBackup()

```
CountryBackup * Country::createBackup ( )
```

A function used to store a country's member variable values at a certain point of the war.

4.9.2.6 createWarParticipantIterator()

```
virtual WarParticipantIterator * Country::createWarParticipantIterator ( )  [virtual]
```

The function creates an iterator to traverse the vector of war participants.

Return values

returns	an iterator pointer that is used to iterate the list.

4.9.2.7 createWarParticipants()

```
void Country::createWarParticipants ( )
```

A function used to create personeel and weapons that will be used in the war.

4.9.2.8 getArtillery()

```
vector< WarParticipant * > Country::getArtillery ( )
```

4.9.2.9 getBattleState()

```
BattleState * Country::getBattleState ( )
```

returns the current state of a country in the battle.

Return values

A	BattleState pointer.

4.9.2.10 getCountryObservers()

```
virtual std::vector< CountryObserver * > Country::getCountryObservers ( )  [virtual]
```

The function is used to access all the observers observing the country.

Return values

a	vector that contains the observers observing the country.

4.9.2.11 getHp()

```
virtual int Country::getHp ( )  [pure virtual]
```

Implemented in Alliance.

4.9.2.12 getName()

```
virtual std::string Country::getName ( )  [virtual]
```

A function used to get the name of the country.

Return values

returns	a string containing the name of the country.

Reimplemented in IndividualCountry.

4.9.2.13 getOpposingC()

```
Country * Country::getOpposingC ( )
```

4.9.2.14 getWarTheatre()

```
WarTheatre * Country::getWarTheatre ( )
```

The function is used to get the area in which a battle will be taking place.

Return values

A	pointer to an object that indicates where the war will be taking place.

4.9.2.15 InflictDamage()

```
void Country::InflictDamage (
             int dmg )
```

4.9.2.16 notify()

```
virtual void Country::notify ( )  [virtual]
```

Reimplemented in IndividualCountry.

4.9.2.17 pickOpposingCountry()

```
Country * Country::pickOpposingCountry (
             vector< Country * > c )
```

This function is used to pick up a opposing Country from the vector provided.

Parameters

in	*c*	A Country vector used to select an opponent.

4.9.2.18 reinstateCountry()

```
void Country::reinstateCountry (
             CountryBackup * cb )
```

a function to take the country back to a certain point during the war.

Parameters

in	*cb*	A backup of the country's member variable values at a certain point in the war.

4.9.2.19 remove()

```
virtual bool Country::remove (
             CountryObserver * c )  [virtual]
```

A function used to stop an observer from observing the country.

Parameters

in	*c*	an observer to be removed from the country's vector of observers.

Return values

returns	TRUE if the observer was removed successfully and FALSE otherwise.

Reimplemented in IndividualCountry.

4.9.2.20 selectWarTheatre()

```
void Country::selectWarTheatre ( )
```

Function used to select the area where a battle should take place at.

4.9.2.21 setBattleState()

```
void Country::setBattleState (
             BattleState * b )
```

The function is used to set the Country's BattleState.

Parameters

in	*b*	The state of the battle being assigned to state member variable.

4.9.2.22 setHp()

```
void Country::setHp (
             int HP )
```

4.9.2.23 withdraw()

```
virtual void Country::withdraw ( )  [inline], [virtual]
```

4.9.3 Member Data Documentation

4.9.3.1 hp

```
int Country::hp =1000  [protected]
```

The documentation for this class was generated from the following file:

- Country.h

4.10 CountryBackup Class Reference

```
#include <CountryBackup.h>
```

Public Member Functions

- CountryBackup (int, BattleState ∗, WarTheatre ∗, vector< WarParticipant ∗ >, vector< CountryObserver ∗ >, Country ∗, bool)
- BattleState ∗ getBattleState ()

  A function used to get the state of the battle.
- WarTheatre ∗ getWarTheatre ()
- WarParticipant ∗ getWarParticipants ()
- CountryObserver ∗ getCountryObservers ()
- virtual ∼CountryBackup ()

Friends

- class Country

4.10.1 Constructor & Destructor Documentation

4.10.1.1 CountryBackup()

```
CountryBackup::CountryBackup (
             int ,
             BattleState ∗ ,
             WarTheatre ∗ ,
             vector< WarParticipant ∗ > ,
             vector< CountryObserver ∗ > ,
             Country ∗ ,
             bool  )
```

4.10.1.2 ∼CountryBackup()

```
virtual CountryBackup::∼CountryBackup ( )  [virtual]
```

4.10.2 Member Function Documentation

4.10.2.1 getBattleState()

```
BattleState ∗ CountryBackup::getBattleState ( )
```

A function used to get the state of the battle.

Return values

returns	a pointer to a BattleState object.

4.10.2.2 getCountryObservers()

```
CountryObserver ∗ CountryBackup::getCountryObservers ( )
```

4.10.2.3 getWarParticipants()

```
WarParticipant ∗ CountryBackup::getWarParticipants ( )
```

4.10.2.4 getWarTheatre()

```
WarTheatre ∗ CountryBackup::getWarTheatre ( )
```

4.10.3 Friends And Related Function Documentation

4.10.3.1 Country

```
friend class Country  [friend]
```

The documentation for this class was generated from the following file:

- CountryBackup.h

4.11 CountryMemory Class Reference

```
#include <CountryMemory.h>
```

Public Member Functions

- CountryMemory ()
- CountryBackup ∗ retrieveBackup ()

  Used to get a previously stored country state.
- void storeBackup (CountryBackup ∗s)

  used to store a country's backup.
- ∼CountryMemory ()

4.11.1 Constructor & Destructor Documentation

4.11.1.1 CountryMemory()

```
CountryMemory::CountryMemory ( )
```

4.11.1.2 ∼CountryMemory()

```
CountryMemory::∼CountryMemory ( )
```

4.11.2 Member Function Documentation

4.11.2.1 retrieveBackup()

```
CountryBackup ∗ CountryMemory::retrieveBackup ( )
```

Used to get a previously stored country state.

Return values

A	pointer to a CountryBackup object.

4.11.2.2 storeBackup()

```
void CountryMemory::storeBackup (
             CountryBackup ∗ s )
```

used to store a country's backup.

Parameters

in	*s*	A variable assigned to backup member variable.

The documentation for this class was generated from the following file:

- CountryMemory.h

4.12 CountryObserver Class Reference

```
#include <CountryObserver.h>
```

Public Member Functions

- virtual void update ()=0

  A function used to update all observers observing the country.

4.12.1 Member Function Documentation

4.12.1.1 update()

```
virtual void CountryObserver::update ( )  [pure virtual]
```

A function used to update all observers observing the country.

Implemented in Medics, and ObservingAllies.

The documentation for this class was generated from the following file:

- CountryObserver.h

4.13 CountryObserverIterator Class Reference

```
#include <CountryObserverIterator.h>
```

Public Member Functions

- CountryObserverIterator (Country ∗)
- void first ()
- void next ()
- bool isLastEl ()
- CountryObserver ∗ currentEl ()

Friends

- class Country

4.13.1 Constructor & Destructor Documentation

4.13.1.1 CountryObserverIterator()

```
CountryObserverIterator::CountryObserverIterator (
             Country * )
```

4.13.2 Member Function Documentation

4.13.2.1 currentEl()

```
CountryObserver * CountryObserverIterator::currentEl ( )
```

4.13.2.2 first()

```
void CountryObserverIterator::first ( )  [virtual]
```

Implements IteratorTool.

4.13.2.3 isLastEl()

```
bool CountryObserverIterator::isLastEl ( )  [virtual]
```

Implements IteratorTool.

4.13.2.4 next()

```
void CountryObserverIterator::next ( )  [virtual]
```

Implements IteratorTool.

4.13.3 Friends And Related Function Documentation

4.13.3.1 Country

```
friend class Country  [friend]
```

The documentation for this class was generated from the following file:

- CountryObserverIterator.h

4.14 Defend Class Reference

```
#include <Defend.h>
```

Public Member Functions

- void handleChange (Country ∗C)

  The functions checks the opponent's hp and checks if they have enough hp to defend themselves.

Additional Inherited Members

4.14.1 Member Function Documentation

4.14.1.1 handleChange()

```
void Defend::handleChange (
            Country ∗ C )  [virtual]
```

The functions checks the opponent's hp and checks if they have enough hp to defend themselves.

Parameters

C	A pointer to the opposing Country conducting an attack.

Reimplemented from BattleState.

The documentation for this class was generated from the following file:

- Defend.h

4.15 DetonateExplosives Class Reference

```
#include <DetonateExplosives.h>
```

Public Member Functions

- void LaunchAttack (Country ∗C)

  The function is used to attack a country using explosives.

4.15.1 Member Function Documentation

4.15.1.1 LaunchAttack()

```
void DetonateExplosives::LaunchAttack (
            Country * C )  [virtual]
```

The function is used to attack a country using explosives.

Parameters

in	*C*	A country that is being attacked.

Implements AttackStrategy.

The documentation for this class was generated from the following file:

- DetonateExplosives.h

4.16 ExplosiveFactory Class Reference

```
#include <ExplosiveFactory.h>
```

Public Member Functions

- WarParticipant ∗ createBomb (int damage)
  function used to create bombs that will be used in war.
- WarParticipant ∗ createMissile (int damage)
  function used to create missiles in the war.

4.16.1 Member Function Documentation

4.16.1.1 createBomb()

```
WarParticipant * ExplosiveFactory::createBomb (
            int damage )
```

function used to create bombs that will be used in war.

Parameters

in	*damage*	A damage that the weapon will inflict.

Return values

a	pointer to a WarParticipant object.

4.16.1.2 createMissile()

```
WarParticipant * ExplosiveFactory::createMissile (
              int damage )
```

function used to create missiles in the war.

Parameters

damage	the damage the missile will inflict.

Return values

a	pointer to a WarParticipant object.

The documentation for this class was generated from the following file:

- ExplosiveFactory.h

4.17 FireArmFactory Class Reference

```
#include <FireArmFactory.h>
```

Public Member Functions

- Rifle * createRifle (int damage)

    A function used to create a rifle.
- MachineGun * createMachineGun (int damage)

    The function is used to create a machine gun.

4.17.1 Member Function Documentation

4.17.1.1 createMachineGun()

```
MachineGun * FireArmFactory::createMachineGun (
              int damage )
```

The function is used to create a machine gun.

Parameters

in	*damage*	A damage the Machine gun inflicts.

Return values

A	pointer to a MachineGun object is returned.

4.17.1.2 createRifle()

```
Rifle * FireArmFactory::createRifle (
             int damage )
```

A function used to create a rifle.

Parameters

in	*damage*	A damage the rifle inflicts.

Return values

A	pointer to a Rifle object returned.

The documentation for this class was generated from the following file:

- FireArmFactory.h

4.18 FireMissile Class Reference

```
#include <FireMissile.h>
```

Public Member Functions

- void LaunchAttack (Country *C)
  A function used to fire a missile.

4.18.1 Member Function Documentation

4.18.1.1 LaunchAttack()

```
void FireMissile::LaunchAttack (
             Country * C )  [virtual]
```

A function used to fire a missile.

Parameters

in	*C*	A country that the missile is directed to.

Implements AttackStrategy.

The documentation for this class was generated from the following file:

- FireMissile.h

4.19 IndividualCountry Class Reference

```
#include <IndividualCountry.h>
```

Public Member Functions

- IndividualCountry (string n)
- bool add (CountryObserver ∗c)

  A function used to add an observer to the vector of observes in a country.
- bool remove (CountryObserver ∗c)

  A function used to stop an observer from observing the country.
- void notify ()

  Used to notify all the observers about any changes that are affecting the country.
- string getName ()

  A function used to get the name of the country.
- int getWeaponHP ()

  Returns the amount of weapons the country still has left.
- int getSoldierHP ()

  Returns the amount of soldiers a country has left.
- int getTransport ()

  returns the amount of transpot units the country has.
- int getSize ()
- void checkHp ()

  Notifies observers about the country's health in the war.
- int getInitialHP ()

  Returns the initial hp the country begins the war with.
- void setInitial ()
- void setAlliance (vector< IndividualCountry ∗ > alliance)
- vector< IndividualCountry ∗ > getAlliance ()
- vector< WarParticipant ∗ > getWarParticipants ()

Additional Inherited Members

4.19.1 Constructor & Destructor Documentation

4.19.1.1 IndividualCountry()

```
IndividualCountry::IndividualCountry (
            string n )
```

4.19.2 Member Function Documentation

4.19.2.1 add()

```
bool IndividualCountry::add (
            CountryObserver * c )  [virtual]
```

A function used to add an observer to the vector of observes in a country.

Parameters

in	*c*	an observer to be added to the observer vector.

Return values

returns	TRUE if the observer was added successfully and FALSE if the obsever was not added.

Reimplemented from Country.

4.19.2.2 checkHp()

```
void IndividualCountry::checkHp ( )
```

Notifies observers about the country's health in the war.

4.19.2.3 getAlliance()

```
vector< IndividualCountry * > IndividualCountry::getAlliance ( )
```

4.19.2.4 getInitialHP()

```
int IndividualCountry::getInitialHP ( )
```

Returns the initial hp the country begins the war with.

Return values

AN	integer value indicating the health of a country at the beginning of the war.

4.19.2.5 getName()

```
string IndividualCountry::getName ( )  [virtual]
```

A function used to get the name of the country.

Return values

returns	a string containing the name of the country.

Reimplemented from Country.

4.19.2.6 getSize()

```
int IndividualCountry::getSize ( )
```

4.19.2.7 getSoldierHP()

```
int IndividualCountry::getSoldierHP ( )
```

Returns the amount of soldiers a country has left.

Return values

A	integer value indicating the amount of soldiers left.

4.19.2.8 getTransport()

```
int IndividualCountry::getTransport ( )
```

returns the amount of transpot units the country has.

Return values

An	integer value indicating a country's transport units.

4.19.2.9 getWarParticipants()

```
vector< WarParticipant * > IndividualCountry::getWarParticipants ( )
```

4.19.2.10 getWeaponHP()

```
int IndividualCountry::getWeaponHP ( )
```

Returns the amount of weapons the country still has left.

Return values

An	integer indicating the amount of weapons a country still has left.

4.19.2.11 notify()

```
void IndividualCountry::notify ( ) [virtual]
```

Used to notify all the observers about any changes that are affecting the country.

Reimplemented from Country.

4.19.2.12 remove()

```
bool IndividualCountry::remove (
            CountryObserver * c ) [virtual]
```

A function used to stop an observer from observing the country.

Parameters

in	*c*	an observer to be removed from the country's vector of observers.

Return values

returns	TRUE if the observer was removed successfully and FALSE otherwise.

Reimplemented from Country.

4.19.2.13 setAlliance()

```
void IndividualCountry::setAlliance (
            vector< IndividualCountry * > alliance )
```

4.19.2.14 setInitial()

```
void IndividualCountry::setInitial ( )
```

The documentation for this class was generated from the following file:

- IndividualCountry.h

4.20 IteratorTool Class Reference

```
#include <IteratorTool.h>
```

Public Member Functions

- IteratorTool ()
- virtual ∼IteratorTool ()
- virtual void first ()=0
- virtual void next ()=0
- virtual bool isLastEl ()=0

4.20.1 Constructor & Destructor Documentation

4.20.1.1 IteratorTool()

```
IteratorTool::IteratorTool ( )
```

4.20.1.2 ∼IteratorTool()

```
virtual IteratorTool::∼IteratorTool ( )  [virtual]
```

4.20.2 Member Function Documentation

4.20.2.1 first()

```
virtual void IteratorTool::first ( )  [pure virtual]
```

Implemented in CountryObserverIterator, and WarParticipantIterator.

4.20.2.2 isLastEl()

```
virtual bool IteratorTool::isLastEl ( )  [pure virtual]
```

Implemented in CountryObserverIterator, and WarParticipantIterator.

4.20.2.3 next()

```
virtual void IteratorTool::next ( )  [pure virtual]
```

Implemented in CountryObserverIterator, and WarParticipantIterator.

The documentation for this class was generated from the following file:

- IteratorTool.h

4.21 Land Class Reference

```
#include <Land.h>
```

The documentation for this class was generated from the following file:

- Land.h

4.22 MachineGunner Class Reference

```
#include <MachineGunner.h>
```

Public Member Functions

- MachineGunner (int d, int h)
- MachineGunner ∗ clone ()

    Used to create a new MachineGunner objects with similar properties.
- void doNotting ()

4.22.1 Constructor & Destructor Documentation

4.22.1.1 MachineGunner()

```
MachineGunner::MachineGunner (
            int d,
            int h )
```

4.22.2 Member Function Documentation

4.22.2.1 clone()

```
MachineGunner ∗ MachineGunner::clone ( )
```

Used to create a new MachineGunner objects with similar properties.

Return values

A	MachineGunner object.

4.22.2.2 doNotting()

```
void MachineGunner::doNotting ( ) [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- MachineGunner.h

4.23 Medic Class Reference

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

Public Member Functions

- Medic (int hp)
- WarParticipant ∗ clone ()

  The function is used to create a new Medic with characteristics similar to the existing one.
- void doNotting ()

4.23.1 Constructor & Destructor Documentation

4.23.1.1 Medic()

```
Medic::Medic (
               int hp )
```

4.23.2 Member Function Documentation

4.23.2.1 clone()

```
WarParticipant * Medic::clone ( )
```

The function is used to create a new Medic with characteristics similar to the existing one.

Return values

A	pointer to a WarParticipant object.

4.23.2.2 doNotting()

```
void Medic::doNotting ( )  [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- Medic.h

4.24 MedicFactory Class Reference

```
#include <MedicFactory.h>
```

Public Member Functions

- Medic ∗ createMedic (int HP)

  Used to create a new medic.

4.24.1 Member Function Documentation

4.24.1.1 createMedic()

```
Medic * MedicFactory::createMedic (
             int HP )
```

Used to create a new medic.

Parameters

in	*HP*	A health value of the medic.

Return values

A	pointer to a Medic object.

The documentation for this class was generated from the following file:

- MedicFactory.h

4.25 Medics Class Reference

```
#include <Medics.h>
```

Public Member Functions

- Medics (IndividualCountry ∗currentCountry)
- void update ()

  Used to heal soldiers that got injured during battle.

4.25.1 Constructor & Destructor Documentation

4.25.1.1 Medics()

```
Medics::Medics (
             IndividualCountry * currentCountry )
```

4.25.2 Member Function Documentation

4.25.2.1 update()

```
void Medics::update ( )  [virtual]
```

Used to heal soldiers that got injured during battle.

Implements CountryObserver.

The documentation for this class was generated from the following file:

- Medics.h

4.26 Missile Class Reference

```
#include <Missile.h>
```

Public Member Functions

- Missile (int d, int h)
- Missile ∗ clone ()

  used to create a new missile by copying the existing one.
- void doNotting ()

4.26.1 Constructor & Destructor Documentation

4.26.1.1 Missile()

```
Missile::Missile (
             int d,
             int h )
```

4.26.2 Member Function Documentation

4.26.2.1 clone()

```
Missile * Missile::clone ( )
```

used to create a new missile by copying the existing one.

Return values

A	pointer to a Missile object.

4.26.2.2 doNotting()

```
void Missile::doNotting ( )  [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- Missile.h

4.27 ObservingAllies Class Reference

```
#include <ObservingAllies.h>
```

Public Member Functions

- ObservingAllies (IndividualCountry ∗currentCountry)
- void update ()

  A function used to update all observers observing the country.

4.27.1 Constructor & Destructor Documentation

4.27.1.1 ObservingAllies()

```
ObservingAllies::ObservingAllies (
             IndividualCountry ∗ currentCountry )
```

4.27.2 Member Function Documentation

4.27.2.1 update()

```
void ObservingAllies::update ( )  [virtual]
```

A function used to update all observers observing the country.

Implements CountryObserver.

The documentation for this class was generated from the following file:

- ObservingAllies.h

4.28 RequestAlliance Class Reference

```
#include <RequestAlliance.h>
```

Public Member Functions

- void handleChange (Country ∗C)

  A function used to pick an available country that can assist the country that requires the help.

Additional Inherited Members

4.28.1 Member Function Documentation

4.28.1.1 handleChange()

```
void RequestAlliance::handleChange (
            Country ∗ C )  [virtual]
```

A function used to pick an available country that can assist the country that requires the help.

Parameters

in	*C*	A country requesting for help.

Reimplemented from BattleState.

The documentation for this class was generated from the following file:

- RequestAlliance.h

4.29 Rifleman Class Reference

```
#include <Rifleman.h>
```

Public Member Functions

- Rifleman (int d, int h)
- Rifleman ∗ clone ()

    Creates a new rifle man by copying the existing one.
- void doNotting ()

4.29.1 Constructor & Destructor Documentation

4.29.1.1 Rifleman()

```
Rifleman::Rifleman (
             int d,
             int h )
```

4.29.2 Member Function Documentation

4.29.2.1 clone()

```
Rifleman ∗ Rifleman::clone ( )
```

Creates a new rifle man by copying the existing one.

Return values

A	pointer to a Rifleman Object.

4.29.2.2 doNotting()

```
void Rifleman::doNotting ( )  [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- Rifleman.h

4.30 Sea Class Reference

```
#include <Sea.h>
```

The documentation for this class was generated from the following file:

- Sea.h

4.31 Shoot Class Reference

```
#include <Shoot.h>
```

Public Member Functions

- void LaunchAttack (Country ∗C)

  The function launches a shooting attack on the opposing country using machine guns and rifles.

4.31.1 Member Function Documentation

4.31.1.1 LaunchAttack()

```
void Shoot::LaunchAttack (
            Country ∗ C )  [virtual]
```

The function launches a shooting attack on the opposing country using machine guns and rifles.

Parameters

in	*C*	A country the attack is being directed to.

Implements AttackStrategy.

The documentation for this class was generated from the following file:

- Shoot.h

4.32 State Class Reference

```
#include <State.h>
```

Public Member Functions

- State (int, BattleState ∗, WarTheatre ∗, vector< WarParticipant ∗ >, vector< CountryObserver ∗ >, Country ∗, bool)
- int getHP ()
- BattleState ∗ getBattleState ()
- WarTheatre ∗ getWarTheatre ()
- vector< WarParticipant ∗ > getWarParticipants ()
- vector< CountryObserver ∗ > getCountryObservers ()
- Country ∗ getOppCountry ()
- bool getW ()

4.32.1 Constructor & Destructor Documentation

4.32.1.1 State()

```
State::State (
             int ,
             BattleState * ,
             WarTheatre * ,
             vector< WarParticipant * > ,
             vector< CountryObserver * > ,
             Country * ,
             bool  )
```

4.32.2 Member Function Documentation

4.32.2.1 getBattleState()

```
BattleState * State::getBattleState ( )  [inline]
```

4.32.2.2 getCountryObservers()

```
vector< CountryObserver * > State::getCountryObservers ( )  [inline]
```

4.32.2.3 getHP()

```
int State::getHP ( )  [inline]
```

4.32.2.4 getOppCountry()

```
Country * State::getOppCountry ( ) [inline]
```

4.32.2.5 getW()

```
bool State::getW ( ) [inline]
```

4.32.2.6 getWarParticipants()

```
vector< WarParticipant * > State::getWarParticipants ( ) [inline]
```

4.32.2.7 getWarTheatre()

```
WarTheatre * State::getWarTheatre ( ) [inline]
```

The documentation for this class was generated from the following file:

- State.h

4.33 Surrender Class Reference

```
#include <Surrender.h>
```

Public Member Functions

- void handleChange (Country *C)

  A function used to give up in the war.

Additional Inherited Members

4.33.1 Member Function Documentation

4.33.1.1 handleChange()

```
void Surrender::handleChange (
            Country * C ) [virtual]
```

A function used to give up in the war.

Parameters

in	*C*	a country that surrenders.

Reimplemented from BattleState.

The documentation for this class was generated from the following file:

- Surrender.h

4.34 WarParticipant Class Reference

```
#include <WarParticipant.h>
```

Public Member Functions

- WarParticipant (string manufacturer, string type, int d, int h)
- virtual void doNotting ()=0
- std::string getType ()

  A function used to get the type of a weapon.
- void incrementParticipantNumber ()

  used to increase the number of weapons by 1.
- int getNumParticipants ()

  Used to get the number of pariciipants in the war.
- int getHP ()
- int getDamage ()

4.34.1 Constructor & Destructor Documentation

4.34.1.1 WarParticipant()

```
WarParticipant::WarParticipant (
            string manufacturer,
            string type,
            int d,
            int h )
```

4.34.2 Member Function Documentation

4.34.2.1 doNotting()

```
virtual void WarParticipant::doNotting ( )  [pure virtual]
```

Implemented in AirCraft, Bomb, MachineGunner, Medic, Missile, Rifleman, and WarShip.

4.34.2.2 getDamage()

```
int WarParticipant::getDamage ( )
```

4.34.2.3 getHP()

```
int WarParticipant::getHP ( )
```

4.34.2.4 getNumParticipants()

```
int WarParticipant::getNumParticipants ( )
```

Used to get the number of pariciipants in the war.

Return values

An	integer value containing the number of paricipants.

4.34.2.5 getType()

```
std::string WarParticipant::getType ( )
```

A function used to get the type of a weapon.

Return values

A	string value containing the type of the weapon.

4.34.2.6 incrementParticipantNumber()

```
void WarParticipant::incrementParticipantNumber ( )
```

used to increase the number of weapons by 1.

The documentation for this class was generated from the following file:

- WarParticipant.h

4.35 WarParticipantFactory Class Reference

```
#include <WarParticipantFactory.h>
```

The documentation for this class was generated from the following file:

- WarParticipantFactory.h

4.36 WarParticipantIterator Class Reference

```
#include <WarParticipantIterator.h>
```

Public Member Functions

- WarParticipantIterator (Country ∗)
- void first ()

  Stores the first value of the WarParticipant vector in curr member variable.
- void next ()

  Used to move to the next element.
- bool isLastEl ()

  Used to if curr is the last element.
- WarParticipant ∗ currentEl ()

  Used to get the current element in the vector.

Friends

- class Country

4.36.1 Constructor & Destructor Documentation

4.36.1.1 WarParticipantIterator()

```
WarParticipantIterator::WarParticipantIterator (
            Country ∗ )
```

4.36.2 Member Function Documentation

4.36.2.1 currentEl()

```
WarParticipant ∗ WarParticipantIterator::currentEl ( )
```

Used to get the current element in the vector.

Return values

Returns	a pointer to the WarParticipant object.

4.36.2.2 first()

```
void WarParticipantIterator::first ( )  [virtual]
```

Stores the first value of the WarParticipant vector in curr member variable.

Implements IteratorTool.

4.36.2.3 isLastEl()

```
bool WarParticipantIterator::isLastEl ( )  [virtual]
```

Used to if curr is the last element.

Return values

Returns	TRUE if it is the last element and false otherwise.

Implements IteratorTool.

4.36.2.4 next()

```
void WarParticipantIterator::next ( )  [virtual]
```

Used to move to the next element.

Implements IteratorTool.

4.36.3 Friends And Related Function Documentation

4.36.3.1 Country

```
friend class Country  [friend]
```

The documentation for this class was generated from the following file:

- WarParticipantIterator.h

4.37 WarShip Class Reference

```
#include <WarShip.h>
```

Public Member Functions

- WarShip (int d, int h)
- WarShip ∗ clone ()

  Creates a clone of the current WarShip object.
- void doNotting ()

4.37.1 Constructor & Destructor Documentation

4.37.1.1 WarShip()

```
WarShip::WarShip (
             int d,
             int h )
```

4.37.2 Member Function Documentation

4.37.2.1 clone()

```
WarShip * WarShip::clone ( )
```

Creates a clone of the current WarShip object.

Return values

Returns	a pointer to the WarShip object.

4.37.2.2 doNotting()

```
void WarShip::doNotting ( )  [virtual]
```

Implements WarParticipant.

The documentation for this class was generated from the following file:

- WarShip.h

4.38 WarTheatre Class Reference

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

The documentation for this class was generated from the following file:

- WarTheatre.h

4.39 WarTransportFactory Class Reference

```
#include <WarTransportFactory.h>
```

Public Member Functions

- WarParticipant ∗ createAirCraft (int HP)

  Used to create an AirCraft object.
- WarParticipant ∗ createWarShip (int HP)

4.39.1 Member Function Documentation

4.39.1.1 createAirCraft()

```
WarParticipant ∗ WarTransportFactory::createAirCraft (
            int HP )
```

Used to create an AirCraft object.

Parameters

in	*HP*	The amount of AirCrafts created.

Return values

Returns	a pointer to a WarParticipant object.

4.39.1.2 createWarShip()

```
WarParticipant ∗ WarTransportFactory::createWarShip (
            int HP )
```

The documentation for this class was generated from the following file:

- WarTransportFactory.h

Chapter 5

File Documentation

5.1 AirCraft.h File Reference

```
#include "WarParticipant.h"
#include "WarParticipantFactory.h"
```

Classes

- class AirCraft

5.2 AirCraft.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_AIRCRAFT_H
#define PRACTICAL_ASSIGNMENT_3_AIRCRAFT_H
#include "WarParticipant.h"
#include "WarParticipantFactory.h"

class AirCraft:  public WarParticipant {
public:
    AirCraft(int d, int h);
     AirCraft* clone();

     void doNotting();

};


#endif //PRACTICAL_ASSIGNMENT_3_AIRCRAFT_H
```

5.3 AirSpace.h File Reference

```
#include "WarTheatre.h"
```

Classes

- class AirSpace

5.4 AirSpace.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/27.
//

#ifndef PROJECTASSIGNMENT_AIRSPACE_H
#define PROJECTASSIGNMENT_AIRSPACE_H
#include "WarTheatre.h"

class AirSpace:  public WarTheatre{

};


#endif //PROJECTASSIGNMENT_AIRSPACE_H
```

5.5 Alliance.h File Reference

```
#include "Country.h"
#include <iostream>
#include <string>
#include <list>
```

Classes

- class Alliance

5.6 Alliance.h

Go to the documentation of this file.

```
#ifndef ALLIANCE_H
#define ALLIANCE_H
#include "Country.h"
#include <iostream>
#include <string>
#include <list>

class CountryObservers;
class  Alliance:  public Country{
public:
    Alliance();
    void setAlliance(vector<Country*> alliance);
    vector<Country*> getAlliance();
    void addAlly(Country* ally);
    void removeAlly(Country* ally);
    int getHp();
private:
    int allianceHp;
    vector<Country*> alliance;

};
#endif
```

5.7 Attack.h File Reference

```
#include "BattleState.h"
#include "DetonateExplosives.h"
#include "Shoot.h"
#include "FireMissile.h"
#include "Context.h"
#include "Country.h"
#include "WarParticipant.h"
```

Classes

- class Attack

5.8 Attack.h

Go to the documentation of this file.
```
#ifndef Attack_h
#define Attack_h
#include "BattleState.h"
#include "DetonateExplosives.h"
#include "Shoot.h"
#include "FireMissile.h"
#include "Context.h"
#include "Country.h"
#include "WarParticipant.h"

class Attack :  public BattleState
{
public:
    void handleChange(Country *C);
};
#endif
```

5.9 AttackStrategy.h File Reference

```
#include "Country.h"
```

Classes

- class AttackStrategy

5.10 AttackStrategy.h

Go to the documentation of this file.
```
#ifndef AttackStrategy_h
#define AttackStrategy_h
#include "Country.h"

class AttackStrategy
{
public:
    virtual void LaunchAttack(Country *C) = 0;
};
#endif
```

5.11 BattleState.h File Reference

```
#include <iostream>
```

Classes

- class BattleState

5.12 BattleState.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/26.
//

#ifndef PROJECTASSIGNMENT_BATTLESTATE_H
#define PROJECTASSIGNMENT_BATTLESTATE_H
class Country;
#include <iostream>

using namespace std;

class BattleState {
public:
    BattleState();
    BattleState *successor;
    void Add(BattleState *Succ);
    virtual void handleChange(Country *C);
};


#endif //PROJECTASSIGNMENT_BATTLESTATE_H
```

5.13 Bomb.h File Reference

```
#include "WarParticipant.h"
#include "WarParticipantFactory.h"
```

Classes

- class Bomb

5.14 Bomb.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_BOMB_H
#define PRACTICAL_ASSIGNMENT_3_BOMB_H
#include "WarParticipant.h"
#include "WarParticipantFactory.h"

class Bomb:  public WarParticipant {
public:
    Bomb(int d, int h);
     Bomb* clone();
     void doNotting();
};


#endif //PRACTICAL_ASSIGNMENT_3_BOMB_H
```

5.15 Context.h File Reference

```
#include "AttackStrategy.h"
#include "Country.h"
```

Classes

- class Context

5.16 Context.h

Go to the documentation of this file.

```
#ifndef Context_h
#define Context_h
#include "AttackStrategy.h"
#include "Country.h"

class Context
{
private:
    AttackStrategy *state;

public:
    Context(AttackStrategy *s);
    ~Context();
    void SetState(AttackStrategy *s);
    void implement(Country *C);
};

#endif
```

5.17 Country.h File Reference

```
#include <list>
#include "WarParticipant.h"
#include "WarTheatre.h"
#include "BattleState.h"
#include "CountryObserver.h"
#include "Sea.h"
#include "AirSpace.h"
#include "Land.h"
#include <iostream>
#include <cstdlib>
#include "time.h"
#include "CountryBackup.h"
#include <vector>
#include <string>
#include "CountryObserverIterator.h"
#include "WarParticipantIterator.h"
```

Classes

- class Country

5.18 Country.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/19.
//

#ifndef PROJECTASSIGNMENT_COUNTRY_H
#define PROJECTASSIGNMENT_COUNTRY_H
#include <list>
#include "WarParticipant.h"
#include "WarTheatre.h"
#include "BattleState.h"
#include "CountryObserver.h"
#include "Sea.h"
#include "AirSpace.h"
#include "Land.h"
#include <iostream>
#include <cstdlib>
#include "time.h"
#include "CountryBackup.h"
#include <vector>
#include <string>
#include "CountryObserverIterator.h"
#include "WarParticipantIterator.h"
using namespace std;
#include <string>

class BattleState;
class WarTheatre;
class CountryObserverIterator;
class WarParticipantIterator;
class CountryIterator;

class Country {


public:
    Country(std::string, bool);
    Country(BattleState* state);
    Country(Country *C);
    Country();
    BattleState* getBattleState();
    Country* pickOpposingCountry(vector<Country*> c);

    virtual void withdraw(){}
    void setBattleState(BattleState* b);
    void selectWarTheatre();
    void createWarParticipants();
    void attackOpposingCountry(Country* c); // state.handleChange(this);
    virtual int getHp()=0;
    CountryBackup* createBackup();//Memento
    void reinstateCountry(CountryBackup* cb);

    virtual WarParticipantIterator* createWarParticipantIterator();//Tseko Iterator
    virtual void addWarParticipant(WarParticipant* wp);
    virtual std::vector<CountryObserver*> getCountryObservers();
     virtual std::string getName();
     WarTheatre* getWarTheatre();
     virtual bool add(CountryObserver* c);   //Country Observer

     virtual bool remove(CountryObserver* c);
     virtual void notify();

     void InflictDamage(int dmg);  //Country BattleState
     Country *getOpposingC();
     void setHp(int HP);
     vector<WarParticipant *>  getArtillery();
     void addArtillery(WarParticipant *W);
     virtual ~Country();

 private:
     State* state;
     BattleState* battlestate;
     WarTheatre* warTheatre;
     std::string cName;
     vector<CountryObserver *> countryObservers;
     Country* OpposingCountry;
     bool win=false;
     bool ocean;

 protected:
     int hp=1000;
 };

```

```

#endif //PROJECTASSIGNMENT_COUNTRY_H
```

5.19 CountryBackup.h File Reference

```
#include <list>
#include "WarParticipant.h"
#include "WarTheatre.h"
#include "BattleState.h"
#include "CountryObserver.h"
#include "Sea.h"
#include "AirSpace.h"
#include "Land.h"
#include <iostream>
#include <cstdlib>
#include "time.h"
#include "State.h"
#include <vector>
```

Classes

- class CountryBackup

5.20 CountryBackup.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/24.
//

#ifndef PROJECTASSIGNMENT_COUNTRYBACKUP_H
#define PROJECTASSIGNMENT_COUNTRYBACKUP_H
#include <list>
#include "WarParticipant.h"
#include "WarTheatre.h"
#include "BattleState.h"
#include "CountryObserver.h"
#include "Sea.h"
#include "AirSpace.h"
#include "Land.h"
#include <iostream>
#include <cstdlib>
#include "time.h"
//#include "Country.h"
#include "State.h"
#include <vector>

using namespace std;
class CountryBackup {
private:
    friend class Country;
    int hp;
    State* state;
    Country* opposingCountry;
    BattleState* battleState;
    WarTheatre* warTheatre;
    std::vector<WarParticipant*> warParticipants;
    std::vector<CountryObserver*> countryObservers;
    bool win;

public:
    CountryBackup(int,BattleState*,
   WarTheatre*,vector<WarParticipant*>,vector<CountryObserver*>,Country*,bool);
    BattleState* getBattleState();
    WarTheatre* getWarTheatre();
    WarParticipant* getWarParticipants();
    CountryObserver* getCountryObservers();
    virtual ~CountryBackup();
};


#endif //PROJECTASSIGNMENT_COUNTRYBACKUP_H
```

5.21 CountryMemory.h File Reference

```
#include "CountryBackup.h"
```

Classes

- class CountryMemory

5.22 CountryMemory.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/24.
//

#ifndef PROJECTASSIGNMENT_COUNTRYMEMORY_H
#define PROJECTASSIGNMENT_COUNTRYMEMORY_H
#include "CountryBackup.h"

class CountryMemory {
private:
    CountryBackup* backup;

public:
    CountryMemory();
    CountryBackup* retrieveBackup();
    void storeBackup(CountryBackup* s);
    ~CountryMemory();
};


#endif //PROJECTASSIGNMENT_COUNTRYMEMORY_H
```

5.23 CountryObserver.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class CountryObserver

5.24 CountryObserver.h

Go to the documentation of this file.

```
#ifndef COUNTRYOBSERVER_H
#define COUNTRYOBSERVER_H

#include <iostream>
#include <string>
#include <vector>
using namespace std;

class CountryObserver {//Observer
    public:
     virtual void update()=0;
};
#endif
```

5.25 CountryObserverIterator.h File Reference

```
#include "IteratorTool.h"
#include <list>
#include "CountryObserver.h"
#include "Country.h"
```

Classes

- class CountryObserverIterator

5.26 CountryObserverIterator.h

Go to the documentation of this file.

```
#ifndef COUNTRYOBSERVERITERATOR_H
#define COUNTRYOBSERVERITERATOR_H
#include "IteratorTool.h"
#include <list>
#include "CountryObserver.h"
#include "Country.h"

class Country;

class CountryObserverIterator :  public IteratorTool
{
    friend class Country;
    public:
        CountryObserverIterator(Country*);
        void first();
        void next();
        bool isLastEl();
        CountryObserver* currentEl();

    private:
        std::vector<CountryObserver*>::iterator it;
        std::vector<CountryObserver*> storeList;
        CountryObserver* curr;
};
#endif
```

5.27 Defend.h File Reference

```
#include "BattleState.h"
```

Classes

- class Defend

5.28 Defend.h

Go to the documentation of this file.

```
#ifndef Defend_h
#define Defend_h
#include "BattleState.h"

class Defend :  public BattleState
{
public:
    void handleChange(Country *C);
};
#endif
```

5.29 DetonateExplosives.h File Reference

```
#include "AttackStrategy.h"
#include <list>
#include <iostream>
```

Classes

- class DetonateExplosives

5.30 DetonateExplosives.h

Go to the documentation of this file.

```
#ifndef DetonateExplosives_h
#define DetonateExplosives_h
#include "AttackStrategy.h"
#include <list>
#include <iostream>

class DetonateExplosives :  public AttackStrategy
{
public:
    void LaunchAttack(Country *C);
};

#endif
```

5.31 ExplosiveFactory.h File Reference

```
#include "WarParticipantFactory.h"
#include "WarParticipant.h"
```

Classes

- class ExplosiveFactory

5.32 ExplosiveFactory.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
#define PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
#include "WarParticipantFactory.h"
#include "WarParticipant.h"

class ExplosiveFactory:  public WarParticipantFactory{
public:
    WarParticipant* createBomb(int damage);
    WarParticipant* createMissile(int damage);
};


#endif //PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
```

5.33 FireArmFactory.h File Reference

```
#include "WarParticipantFactory.h"
#include "WarParticipant.h"
#include "MachineGun.h"
#include "Rifle.h"
```

Classes

- class FireArmFactory

5.34 FireArmFactory.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_FIREARMFACTORY_H
#define PRACTICAL_ASSIGNMENT_3_FIREARMFACTORY_H
#include "WarParticipantFactory.h"
#include "WarParticipant.h"
#include "MachineGun.h"
#include "Rifle.h"

class FireArmFactory:  public WarParticipantFactory{
public:
    Rifle* createRifle(int damage);
    MachineGun* createMachineGun(int damage);
};


#endif //PRACTICAL_ASSIGNMENT_3_FIREARMFACTORY_H
```

5.35 FireMissile.h File Reference

```
#include "AttackStrategy.h"
#include <list>
#include <iostream>
```

Classes

- class FireMissile

5.36 FireMissile.h

Go to the documentation of this file.

```
#ifndef FireMissile_h
#define FireMissile_h
#include "AttackStrategy.h"
#include <list>
#include <iostream>

class FireMissile :  public AttackStrategy
{
public:
    void LaunchAttack(Country *C);
};

#endif
```

5.37 IndividualCountry.h File Reference

```
#include "Country.h"
#include "CountryObserver.h"
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class IndividualCountry

5.38 IndividualCountry.h

Go to the documentation of this file.

```
#ifndef INDIVIDUALCOUNTRY_H
#define  INDIVIDUALCOUNTRY_H
#include "Country.h"
#include "CountryObserver.h"
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class CountryObservers;
class  IndividualCountry:  public Country{
     public:
        IndividualCountry(string n);
        bool add(CountryObserver* c);
        bool remove(CountryObserver* c);
        void notify();
        string getName();
        int getWeaponHP();
        int getSoldierHP();
        int getTransport();
        int getSize();
        void checkHp();
        int getInitialHP();
        void setInitial();
        void setAlliance(vector<IndividualCountry*> alliance);
        vector<IndividualCountry*> getAlliance();
        vector<WarParticipant*> getWarParticipants();
    private:
        vector<CountryObserver*> countryObservers;
        vector<IndividualCountry*> alliance;
        vector<WarParticipant*> wP;
        int weaponHP;
        int soldierHP;
        int transport;
        int currWeaponHP;
        int initialHP;
        string name;
};
#endif
```

5.39 IteratorTool.h File Reference

Classes

- class IteratorTool

5.40 IteratorTool.h

Go to the documentation of this file.

```
#ifndef ITERATORTOOL_H
#define ITERATORTOOL_H

class IteratorTool
{
    public:
        IteratorTool();
        virtual ~IteratorTool();
        virtual void first() = 0;
        virtual void next() = 0;
        virtual bool isLastEl() = 0;
};
#endif
```

5.41 Land.h File Reference

```
#include "WarTheatre.h"
```

Classes

- class Land

5.42 Land.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/27.
//

#ifndef PROJECTASSIGNMENT_LAND_H
#define PROJECTASSIGNMENT_LAND_H
#include "WarTheatre.h"

class Land:  public WarTheatre {

};


#endif //PROJECTASSIGNMENT_LAND_H
```

5.43 MachineGunner.h File Reference

```
#include "WarParticipant.h"
#include "WarParticipantFactory.h"
```

Classes

- class MachineGunner

5.44 MachineGunner.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_MACHINEGUNNER_H
#define PRACTICAL_ASSIGNMENT_3_MACHINEGUNNER_H
#include "WarParticipant.h"
#include "WarParticipantFactory.h"

class MachineGunner:  public WarParticipant {
public:
    MachineGunner(int d, int h);
    MachineGunner* clone();
    void doNotting();
};


#endif //PRACTICAL_ASSIGNMENT_3_MACHINEGUNNER_H
```

5.45 Medic.h File Reference

```
#include "WarParticipant.h"
#include "WarParticipantFactory.h"
```

Classes

- class Medic

5.46 Medic.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_MEDIC_H
#define PRACTICAL_ASSIGNMENT_3_MEDIC_H
#include "WarParticipant.h"
#include "WarParticipantFactory.h"

class Medic:  public WarParticipant {
private:
    int HP;
public:
    Medic(int hp);
    WarParticipant* clone();
    void doNotting();
};


#endif //PRACTICAL_ASSIGNMENT_3_MEDIC_H
```

5.47 MedicFactory.h File Reference

```
#include "WarParticipantFactory.h"
#include "Medic.h"
```

Classes

- class MedicFactory

5.48 MedicFactory.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_MEDICFACTORY_H
#define PRACTICAL_ASSIGNMENT_3_MEDICFACTORY_H
#include "WarParticipantFactory.h"
#include "Medic.h"

class MedicFactory:  public WarParticipantFactory {
public:
    Medic* createMedic(int HP);
};

#endif //PRACTICAL_ASSIGNMENT_3_MEDICFACTORY_H
```

5.49 Medics.h File Reference

```
#include "IndividualCountry.h"
#include "CountryObserver.h"
#include "Country.h"
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class Medics

5.50 Medics.h

Go to the documentation of this file.

```
#ifndef MEDICS_H
#define MEDICS_H
#include "IndividualCountry.h"
#include "CountryObserver.h"
#include "Country.h"
#include <iostream>
#include <string>
#include <vector>


using namespace std;

class Medics :  public CountryObserver {//Concrete Observer
```

```
    public:
        Medics(IndividualCountry* currentCountry);
        void  update();
    private:
        int observedHP;
        int observedS;
        int observedT;
        int maxHeal=0;
        int healingHp=90;
        bool medicObserved=false;//set to false if true, when Allies send medics
        IndividualCountry* currentCountry;
        vector<WarParticipant*> warParticipants;
};
#endif
```

5.51 Missile.h File Reference

```
#include "WarParticipant.h"
#include "WarParticipantFactory.h"
```

Classes

- class Missile

5.52 Missile.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_MISSILE_H
#define PRACTICAL_ASSIGNMENT_3_MISSILE_H
#include "WarParticipant.h"
#include "WarParticipantFactory.h"

class Missile:  public WarParticipant {
public:
    Missile(int d, int h);
     Missile* clone();
     void doNotting();
};


#endif //PRACTICAL_ASSIGNMENT_3_MISSILE_H
```

5.53 ObservingAllies.h File Reference

```
#include "BattleState.h"
#include "IndividualCountry.h"
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class ObservingAllies

5.54 ObservingAllies.h

Go to the documentation of this file.

```
#ifndef OBSERVINGALLIES_H
#define OBSERVINGALLIES_H
#include "BattleState.h"
#include "IndividualCountry.h"
#include <iostream>
#include <string>
#include <vector>


using namespace std;

class ObservingAllies :  public CountryObserver{//Concrete Observers
    public:
        ObservingAllies(IndividualCountry* currentCountry);
        void  update();
    private:
        int observedH;
        IndividualCountry* currentCountry;
        vector<WarParticipant*> warParticipants;
};
#endif
```

5.55 RequestAlliance.h File Reference

```
#include "BattleState.h"
```

Classes

- class RequestAlliance

5.56 RequestAlliance.h

Go to the documentation of this file.

```
#ifndef RequestAlliance_h
#define RequestAlliance_h
#include "BattleState.h"

class RequestAlliance :  public BattleState
{
public:
    void handleChange(Country *C);
};
#endif
```

5.57 Rifleman.h File Reference

```
#include "WarParticipant.h"
```

Classes

- class Rifleman

5.58 Rifleman.h

Go to the documentation of this file.
```
#ifndef PRACTICAL_ASSIGNMENT_3_RIFLEMAN_H
#define PRACTICAL_ASSIGNMENT_3_RIFLEMAN_H
#include "WarParticipant.h"

class Rifleman:  public WarParticipant{
public:
    Rifleman(int d, int h);
     Rifleman* clone();
     void doNotting();
};


#endif //PRACTICAL_ASSIGNMENT_3_RIFLEMAN_H
```

5.59 Sea.h File Reference

```
#include "WarTheatre.h"
```

Classes

- class Sea

5.60 Sea.h

Go to the documentation of this file.
```
//
// Created by JOHANES MATSEBA on 2022/10/27.
//

#ifndef PROJECTASSIGNMENT_SEA_H
#define PROJECTASSIGNMENT_SEA_H


#include "WarTheatre.h"

class Sea:  public WarTheatre {

};


#endif //PROJECTASSIGNMENT_SEA_H
```

5.61 Shoot.h File Reference

```
#include "AttackStrategy.h"
#include <list>
#include <iostream>
```

Classes

- class Shoot

5.62 Shoot.h

Go to the documentation of this file.

```
#ifndef Shoot_h
#define Shoot_h
#include "AttackStrategy.h"
#include <list>
#include <iostream>

class Shoot :  public AttackStrategy
{
public:
    void LaunchAttack(Country *C);
};

#endif
```

5.63 State.h File Reference

```
#include <list>
#include "WarParticipant.h"
#include "WarTheatre.h"
#include "BattleState.h"
#include "CountryObserver.h"
#include "Sea.h"
#include "AirSpace.h"
#include "Land.h"
#include <iostream>
#include <cstdlib>
#include "time.h"
#include <vector>
```

Classes

- class State

5.64 State.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/29.
//

#ifndef PROJECTASSIGNMENT_STATE_H
#define PROJECTASSIGNMENT_STATE_H

#include <list>
#include "WarParticipant.h"
#include "WarTheatre.h"
#include "BattleState.h"
#include "CountryObserver.h"
#include "Sea.h"
#include "AirSpace.h"
#include "Land.h"
#include <iostream>
#include <cstdlib>
#include "time.h"

#include <vector>


using namespace std;
```

```
class Country;
class State {

public:
    State(int,BattleState*, WarTheatre*,vector<WarParticipant*>,vector<CountryObserver*>,Country*,bool);
    int getHP(){return hp;};
    BattleState* getBattleState(){ return battlestate;};
    WarTheatre* getWarTheatre(){return warTheatre;};
    vector<WarParticipant*> getWarParticipants(){return warParticipants;};
    vector<CountryObserver*> getCountryObservers(){return countryObservers;};
    Country* getOppCountry(){return opposingCountry;};
    bool getW(){return win;};
private:
    int hp;
    BattleState* battlestate;
    WarTheatre* warTheatre;
    std::vector<WarParticipant*> warParticipants;
    std::vector<CountryObserver*> countryObservers;
    Country* opposingCountry;
    bool win=false;
};


#endif //PROJECTASSIGNMENT_STATE_H
```

5.65 Surrender.h File Reference

```
#include "BattleState.h"
```

Classes

- class Surrender

5.66 Surrender.h

Go to the documentation of this file.

```
#ifndef Surrender_h
#define Surrender_h
#include "BattleState.h"

class Surrender :  public BattleState
{
public:
    void handleChange(Country *C);
};
#endif
```

5.67 WarParticipant.h File Reference

```
#include <string>
```

Classes

- class WarParticipant

5.68 WarParticipant.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_WARPARTICIPANT_H
#define PRACTICAL_ASSIGNMENT_3_WARPARTICIPANT_H

#include <string>
using namespace std;

class WarParticipant{

public:
    WarParticipant(string manufacturer, string type, int d, int h);
    virtual void doNotting()=0;
    std::string getType();
    void incrementParticipantNumber();
    int getNumParticipants();
    int getHP();
    int getDamage();
private:
    string manufacturer;
    string type;
    int damage;
    int hp;
    int numParticipants; //Iterator
};


#endif //PRACTICAL_ASSIGNMENT_3_WARPARTICIPANT_H
```

5.69 WarParticipantFactory.h File Reference

```
#include "WarParticipantFactory.h"
#include "WarParticipant.h"
```

Classes

- class WarParticipantFactory

5.70 WarParticipantFactory.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_WARPARTICIPANTFACTORY_H
#define PRACTICAL_ASSIGNMENT_3_WARPARTICIPANTFACTORY_H
#include "WarParticipantFactory.h"
#include "WarParticipant.h"

class WarParticipantFactory {
//public:
    //WarParticipantFactory();
    //~WarParticipantFactory();

};


#endif //PRACTICAL_ASSIGNMENT_3_WARPARTICIPANTFACTORY_H
```

5.71 WarParticipantIterator.h File Reference

```
#include "IteratorTool.h"
#include <list>
#include "WarParticipant.h"
#include "Country.h"
```

Classes

- class WarParticipantIterator

5.72 WarParticipantIterator.h

Go to the documentation of this file.

```
#ifndef WARPARTICIPANTITERATOR_H
#define WARPARTICIPANTITERATOR_H
#include "IteratorTool.h"
#include <list>
#include "WarParticipant.h"
#include "Country.h"

class Country;

class WarParticipantIterator :  public IteratorTool
{
    friend class Country;
    public:
        WarParticipantIterator(Country*);
        void first();
        void next();
        bool isLastEl();
        WarParticipant* currentEl();

    private:
        std::vector<WarParticipant*>::iterator it;
        std::vector<WarParticipant*> storeList;
        WarParticipant* curr;
};
#endif
```

5.73 WarShip.h File Reference

```
#include "WarParticipant.h"
#include "WarParticipantFactory.h"
```

Classes

- class WarShip

5.74 WarShip.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_WARSHIP_H
#define PRACTICAL_ASSIGNMENT_3_WARSHIP_H
#include "WarParticipant.h"
#include "WarParticipantFactory.h"

class WarShip:  public WarParticipant {
public:
    WarShip(int d, int h);
     WarShip* clone();
     void doNotting();
};


#endif //PRACTICAL_ASSIGNMENT_3_WARSHIP_H
```

5.75 WarTheatre.h File Reference

Classes

- class WarTheatre

5.76 WarTheatre.h

Go to the documentation of this file.

```
//
// Created by JOHANES MATSEBA on 2022/10/26.
//

#ifndef PROJECTASSIGNMENT_WARTHEATRE_H
#define PROJECTASSIGNMENT_WARTHEATRE_H


class WarTheatre {

};


#endif //PROJECTASSIGNMENT_WARTHEATRE_H
```

5.77 WarTransportFactory.h File Reference

```
#include "WarParticipantFactory.h"
#include "WarParticipant.h"
```

Classes

- class WarTransportFactory

5.78 WarTransportFactory.h

Go to the documentation of this file.

```
#ifndef PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
#define PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
#include "WarParticipantFactory.h"
#include "WarParticipant.h"

class WarTransportFactory:  public WarParticipantFactory{
public:
    WarParticipant* createAirCraft(int HP);
    WarParticipant* createWarShip(int HP);
};


#endif //PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
```