

# GangOfSix Documentation

## 1.0

Generated by Doxygen 1.9.5



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AirCraft Class Reference	7
4.1.1 Constructor & Destructor Documentation	7
4.1.1.1 AirCraft()	7
4.1.2 Member Function Documentation	7
4.1.2.1 clone()	7
4.1.2.2 doNotting()	8
4.2 AirSpace Class Reference	8
4.3 Alliance Class Reference	8
4.3.1 Constructor & Destructor Documentation	9
4.3.1.1 Alliance()	9
4.3.2 Member Function Documentation	9
4.3.2.1 addAlly()	9
4.3.2.2 getAlliance()	9
4.3.2.3 getHp()	9
4.3.2.4 removeAlly()	10
4.3.2.5 setAlliance()	10
4.4 Attack Class Reference	10
4.4.1 Member Function Documentation	11
4.4.1.1 handleChange()	11
4.5 AttackStrategy Class Reference	11
4.5.1 Member Function Documentation	11
4.5.1.1 LaunchAttack()	11
4.6 BattleState Class Reference	12
4.6.1 Constructor & Destructor Documentation	12
4.6.1.1 BattleState()	12
4.6.2 Member Function Documentation	12
4.6.2.1 Add()	12
4.6.2.2 handleChange()	13
4.6.3 Member Data Documentation	13
4.6.3.1 successor	13
4.7 Bomb Class Reference	13
4.7.1 Constructor & Destructor Documentation	13
4.7.1.1 Bomb()	13

4.7.2 Member Function Documentation	14
4.7.2.1 clone()	14
4.7.2.2 doNotting()	14
4.8 Context Class Reference	14
4.8.1 Constructor & Destructor Documentation	14
4.8.1.1 Context()	14
4.8.1.2 ~Context()	15
4.8.2 Member Function Documentation	15
4.8.2.1 implement()	15
4.8.2.2 SetState()	15
4.9 Country Class Reference	15
4.9.1 Constructor & Destructor Documentation	16
4.9.1.1 Country() [1/4]	17
4.9.1.2 Country() [2/4]	17
4.9.1.3 Country() [3/4]	17
4.9.1.4 Country() [4/4]	17
4.9.1.5 ~Country()	17
4.9.2 Member Function Documentation	17
4.9.2.1 add()	17
4.9.2.2 addArtillery()	18
4.9.2.3 addWarParticipant()	18
4.9.2.4 attackOpposingCountry()	18
4.9.2.5 createBackup()	18
4.9.2.6 createWarParticipantIterator()	19
4.9.2.7 createWarParticipants()	19
4.9.2.8 getArtillery()	19
4.9.2.9 getBattleState()	19
4.9.2.10 getCountryObservers()	19
4.9.2.11 getHp()	20
4.9.2.12 getName()	20
4.9.2.13 getOpposingC()	20
4.9.2.14 getWarTheatre()	20
4.9.2.15 InflictDamage()	20
4.9.2.16 notify()	21
4.9.2.17 pickOpposingCountry()	21
4.9.2.18 reinstateCountry()	21
4.9.2.19 remove()	21
4.9.2.20 selectWarTheatre()	22
4.9.2.21 setBattleState()	22
4.9.2.22 setHp()	22
4.9.2.23 withdraw()	22
4.9.3 Member Data Documentation	22

4.9.3.1 hp	22
4.10 CountryBackup Class Reference	23
4.10.1 Constructor & Destructor Documentation	23
4.10.1.1 CountryBackup()	23
4.10.1.2 ~CountryBackup()	23
4.10.2 Member Function Documentation	23
4.10.2.1 getBattleState()	23
4.10.2.2 getCountryObservers()	24
4.10.2.3 getWarParticipants()	24
4.10.2.4 getWarTheatre()	24
4.10.3 Friends And Related Function Documentation	24
4.10.3.1 Country	24
4.11 CountryMemory Class Reference	24
4.11.1 Constructor & Destructor Documentation	25
4.11.1.1 CountryMemory()	25
4.11.1.2 ~CountryMemory()	25
4.11.2 Member Function Documentation	25
4.11.2.1 retrieveBackup()	25
4.11.2.2 storeBackup()	25
4.12 CountryObserver Class Reference	26
4.12.1 Member Function Documentation	26
4.12.1.1 update()	26
4.13 CountryObserverIterator Class Reference	26
4.13.1 Constructor & Destructor Documentation	26
4.13.1.1 CountryObserverIterator()	27
4.13.2 Member Function Documentation	27
4.13.2.1 currentEI()	27
4.13.2.2 first()	27
4.13.2.3 isLastEI()	27
4.13.2.4 next()	27
4.13.3 Friends And Related Function Documentation	27
4.13.3.1 Country	27
4.14 Defend Class Reference	28
4.14.1 Member Function Documentation	28
4.14.1.1 handleChange()	28
4.15 DetonateExplosives Class Reference	28
4.15.1 Member Function Documentation	28
4.15.1.1 LaunchAttack()	29
4.16 ExplosiveFactory Class Reference	29
4.16.1 Member Function Documentation	29
4.16.1.1 createBomb()	29
4.16.1.2 createMissile()	30

4.17 FireArmFactory Class Reference	30
4.17.1 Member Function Documentation	30
4.17.1.1 createMachineGun()	30
4.17.1.2 createRifle()	31
4.18 FireMissile Class Reference	31
4.18.1 Member Function Documentation	31
4.18.1.1 LaunchAttack()	31
4.19 IndividualCountry Class Reference	32
4.19.1 Constructor & Destructor Documentation	32
4.19.1.1 IndividualCountry()	33
4.19.2 Member Function Documentation	33
4.19.2.1 add()	33
4.19.2.2 checkHp()	33
4.19.2.3 getAlliance()	33
4.19.2.4 getInitialHP()	33
4.19.2.5 getName()	34
4.19.2.6 getSize()	34
4.19.2.7 getSoldierHP()	34
4.19.2.8 getTransport()	34
4.19.2.9 getWarParticipants()	35
4.19.2.10 getWeaponHP()	35
4.19.2.11 notify()	35
4.19.2.12 remove()	35
4.19.2.13 setAlliance()	36
4.19.2.14 setInitial()	36
4.20 IteratorTool Class Reference	36
4.20.1 Constructor & Destructor Documentation	36
4.20.1.1 IteratorTool()	36
4.20.1.2 ~IteratorTool()	37
4.20.2 Member Function Documentation	37
4.20.2.1 first()	37
4.20.2.2 isLastEl()	37
4.20.2.3 next()	37
4.21 Land Class Reference	37
4.22 MachineGunner Class Reference	37
4.22.1 Constructor & Destructor Documentation	38
4.22.1.1 MachineGunner()	38
4.22.2 Member Function Documentation	38
4.22.2.1 clone()	38
4.22.2.2 doNotting()	38
4.23 Medic Class Reference	38
4.23.1 Constructor & Destructor Documentation	39

4.23.1.1 Medic()	39
4.23.2 Member Function Documentation	39
4.23.2.1 clone()	39
4.23.2.2 doNotting()	39
4.24 MedicFactory Class Reference	39
4.24.1 Member Function Documentation	40
4.24.1.1 createMedic()	40
4.25 Medics Class Reference	40
4.25.1 Constructor & Destructor Documentation	40
4.25.1.1 Medics()	41
4.25.2 Member Function Documentation	41
4.25.2.1 update()	41
4.26 Missile Class Reference	41
4.26.1 Constructor & Destructor Documentation	41
4.26.1.1 Missile()	41
4.26.2 Member Function Documentation	41
4.26.2.1 clone()	41
4.26.2.2 doNotting()	42
4.27 ObservingAllies Class Reference	42
4.27.1 Constructor & Destructor Documentation	42
4.27.1.1 ObservingAllies()	42
4.27.2 Member Function Documentation	42
4.27.2.1 update()	43
4.28 RequestAlliance Class Reference	43
4.28.1 Member Function Documentation	43
4.28.1.1 handleChange()	43
4.29 Rifleman Class Reference	44
4.29.1 Constructor & Destructor Documentation	44
4.29.1.1 Rifleman()	44
4.29.2 Member Function Documentation	44
4.29.2.1 clone()	44
4.29.2.2 doNotting()	44
4.30 Sea Class Reference	45
4.31 Shoot Class Reference	45
4.31.1 Member Function Documentation	45
4.31.1.1 LaunchAttack()	45
4.32 State Class Reference	45
4.32.1 Constructor & Destructor Documentation	46
4.32.1.1 State()	46
4.32.2 Member Function Documentation	46
4.32.2.1 getBattleState()	46
4.32.2.2 getCountryObservers()	46

4.32.2.3 getHP()	46
4.32.2.4 getOppCountry()	47
4.32.2.5 getW()	47
4.32.2.6 getWarParticipants()	47
4.32.2.7 getWarTheatre()	47
4.33 Surrender Class Reference	47
4.33.1 Member Function Documentation	47
4.33.1.1 handleChange()	47
4.34 WarParticipant Class Reference	48
4.34.1 Constructor & Destructor Documentation	48
4.34.1.1 WarParticipant()	48
4.34.2 Member Function Documentation	48
4.34.2.1 doNotting()	49
4.34.2.2 getDamage()	49
4.34.2.3 getHP()	49
4.34.2.4 getNumParticipants()	49
4.34.2.5 getType()	49
4.34.2.6 incrementParticipantNumber()	49
4.35 WarParticipantFactory Class Reference	50
4.36 WarParticipantIterator Class Reference	50
4.36.1 Constructor & Destructor Documentation	50
4.36.1.1 WarParticipantIterator()	50
4.36.2 Member Function Documentation	50
4.36.2.1 currentEI()	50
4.36.2.2 first()	51
4.36.2.3 isLastEI()	51
4.36.2.4 next()	51
4.36.3 Friends And Related Function Documentation	51
4.36.3.1 Country	51
4.37 WarShip Class Reference	52
4.37.1 Constructor & Destructor Documentation	52
4.37.1.1 WarShip()	52
4.37.2 Member Function Documentation	52
4.37.2.1 clone()	52
4.37.2.2 doNotting()	52
4.38 WarTheatre Class Reference	53
4.39 WarTransportFactory Class Reference	53
4.39.1 Member Function Documentation	53
4.39.1.1 createAirCraft()	53
4.39.1.2 createWarShip()	53

## 5 File Documentation

55



5.1 AirCraft.h File Reference . . . . .	55
5.2 AirCraft.h . . . . .	55
5.3 AirSpace.h File Reference . . . . .	55
5.4 AirSpace.h . . . . .	56
5.5 Alliance.h File Reference . . . . .	56
5.6 Alliance.h . . . . .	56
5.7 Attack.h File Reference . . . . .	56
5.8 Attack.h . . . . .	57
5.9 AttackStrategy.h File Reference . . . . .	57
5.10 AttackStrategy.h . . . . .	57
5.11 BattleState.h File Reference . . . . .	57
5.12 BattleState.h . . . . .	58
5.13 Bomb.h File Reference . . . . .	58
5.14 Bomb.h . . . . .	58
5.15 Context.h File Reference . . . . .	58
5.16 Context.h . . . . .	59
5.17 Country.h File Reference . . . . .	59
5.18 Country.h . . . . .	60
5.19 CountryBackup.h File Reference . . . . .	61
5.20 CountryBackup.h . . . . .	61
5.21 CountryMemory.h File Reference . . . . .	62
5.22 CountryMemory.h . . . . .	62
5.23 CountryObserver.h File Reference . . . . .	62
5.24 CountryObserver.h . . . . .	63
5.25 CountryObserverIterator.h File Reference . . . . .	63
5.26 CountryObserverIterator.h . . . . .	63
5.27 Defend.h File Reference . . . . .	63
5.28 Defend.h . . . . .	64
5.29 DetonateExplosives.h File Reference . . . . .	64
5.30 DetonateExplosives.h . . . . .	64
5.31 ExplosiveFactory.h File Reference . . . . .	64
5.32 ExplosiveFactory.h . . . . .	65
5.33 FireArmFactory.h File Reference . . . . .	65
5.34 FireArmFactory.h . . . . .	65
5.35 FireMissile.h File Reference . . . . .	65
5.36 FireMissile.h . . . . .	66
5.37 IndividualCountry.h File Reference . . . . .	66
5.38 IndividualCountry.h . . . . .	66
5.39 IteratorTool.h File Reference . . . . .	67
5.40 IteratorTool.h . . . . .	67
5.41 Land.h File Reference . . . . .	67
5.42 Land.h . . . . .	67

5.43 MachineGunner.h File Reference . . . . .	68
5.44 MachineGunner.h . . . . .	68
5.45 Medic.h File Reference . . . . .	68
5.46 Medic.h . . . . .	68
5.47 MedicFactory.h File Reference . . . . .	69
5.48 MedicFactory.h . . . . .	69
5.49 Medics.h File Reference . . . . .	69
5.50 Medics.h . . . . .	69
5.51 Missile.h File Reference . . . . .	70
5.52 Missile.h . . . . .	70
5.53 ObservingAllies.h File Reference . . . . .	70
5.54 ObservingAllies.h . . . . .	71
5.55 RequestAlliance.h File Reference . . . . .	71
5.56 RequestAlliance.h . . . . .	71
5.57 Rifleman.h File Reference . . . . .	71
5.58 Rifleman.h . . . . .	72
5.59 Sea.h File Reference . . . . .	72
5.60 Sea.h . . . . .	72
5.61 Shoot.h File Reference . . . . .	72
5.62 Shoot.h . . . . .	73
5.63 State.h File Reference . . . . .	73
5.64 State.h . . . . .	73
5.65 Surrender.h File Reference . . . . .	74
5.66 Surrender.h . . . . .	74
5.67 WarParticipant.h File Reference . . . . .	74
5.68 WarParticipant.h . . . . .	75
5.69 WarParticipantFactory.h File Reference . . . . .	75
5.70 WarParticipantFactory.h . . . . .	75
5.71 WarParticipantIterator.h File Reference . . . . .	75
5.72 WarParticipantIterator.h . . . . .	76
5.73 WarShip.h File Reference . . . . .	76
5.74 WarShip.h . . . . .	76
5.75 WarTheatre.h File Reference . . . . .	77
5.76 WarTheatre.h . . . . .	77
5.77 WarTransportFactory.h File Reference . . . . .	77
5.78 WarTransportFactory.h . . . . .	77

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

AttackStrategy . . . . .	11
DetonateExplosives . . . . .	28
FireMissile . . . . .	31
Shoot . . . . .	45
BattleState . . . . .	12
Attack . . . . .	10
Defend . . . . .	28
RequestAlliance . . . . .	43
Surrender . . . . .	47
Context . . . . .	14
Country . . . . .	15
Alliance . . . . .	8
IndividualCountry . . . . .	32
CountryBackup . . . . .	23
CountryMemory . . . . .	24
CountryObserver . . . . .	26
Medics . . . . .	40
ObservingAllies . . . . .	42
IteratorTool . . . . .	36
CountryObserverIterator . . . . .	26
WarParticipantIterator . . . . .	50
State . . . . .	45
WarParticipant . . . . .	48
AirCraft . . . . .	7
Bomb . . . . .	13
MachineGunner . . . . .	37
Medic . . . . .	38
Missile . . . . .	41
Rifleman . . . . .	44
WarShip . . . . .	52
WarParticipantFactory . . . . .	50
ExplosiveFactory . . . . .	29
FireArmFactory . . . . .	30

MedicFactory . . . . .	39
WarTransportFactory . . . . .	53
WarTheatre . . . . .	53
AirSpace . . . . .	8
Land . . . . .	37
Sea . . . . .	45

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AirCraft</a>	7
<a href="#">AirSpace</a>	8
<a href="#">Alliance</a>	8
<a href="#">Attack</a>	10
<a href="#">AttackStrategy</a>	11
<a href="#">BattleState</a>	12
<a href="#">Bomb</a>	13
<a href="#">Context</a>	14
<a href="#">Country</a>	15
<a href="#">CountryBackup</a>	23
<a href="#">CountryMemory</a>	24
<a href="#">CountryObserver</a>	26
<a href="#">CountryObserverIterator</a>	26
<a href="#">Defend</a>	28
<a href="#">DetonateExplosives</a>	28
<a href="#">ExplosiveFactory</a>	29
<a href="#">FireArmFactory</a>	30
<a href="#">FireMissile</a>	31
<a href="#">IndividualCountry</a>	32
<a href="#">IteratorTool</a>	36
<a href="#">Land</a>	37
<a href="#">MachineGunner</a>	37
<a href="#">Medic</a>	38
<a href="#">MedicFactory</a>	39
<a href="#">Medics</a>	40
<a href="#">Missile</a>	41
<a href="#">ObservingAllies</a>	42
<a href="#">RequestAlliance</a>	43
<a href="#">Rifleman</a>	44
<a href="#">Sea</a>	45
<a href="#">Shoot</a>	45
<a href="#">State</a>	45
<a href="#">Surrender</a>	47
<a href="#">WarParticipant</a>	48
<a href="#">WarParticipantFactory</a>	50

<a href="#">WarParticipantIterator</a> . . . . .	50
<a href="#">WarShip</a> . . . . .	52
<a href="#">WarTheatre</a> . . . . .	53
<a href="#">WarTransportFactory</a> . . . . .	53

## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

AirCraft.h	55
AirSpace.h	55
Alliance.h	56
Attack.h	56
AttackStrategy.h	57
BattleState.h	57
Bomb.h	58
Context.h	58
Country.h	59
CountryBackup.h	61
CountryMemory.h	62
CountryObserver.h	62
CountryObserverIterator.h	63
Defend.h	63
DetonateExplosives.h	64
ExplosiveFactory.h	64
FireArmFactory.h	65
FireMissile.h	65
IndividualCountry.h	66
IteratorTool.h	67
Land.h	67
MachineGunner.h	68
Medic.h	68
MedicFactory.h	69
Medics.h	69
Missile.h	70
ObservingAllies.h	70
RequestAlliance.h	71
Rifleman.h	71
Sea.h	72
Shoot.h	72
State.h	73
Surrender.h	74
WarParticipant.h	74
WarParticipantFactory.h	75

<a href="#">WarParticipantIterator.h</a>	75
<a href="#">WarShip.h</a>	76
<a href="#">WarTheatre.h</a>	77
<a href="#">WarTransportFactory.h</a>	77



## 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 [doNothing](#) ()

#### 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

<i>An</i>	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 <a href="#">Country</a> that is being added to an <a href="#">Alliance</a> 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

<i>a</i>	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

<i>ally</i>	The country to be removed from an <a href="#">Alliance</a> 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	<i>alliance</i>	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 <a href="#">Country</a> 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 <a href="#">Country</a> 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 <a href="#">BattleState</a> 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 [doNothing](#) ()

### 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	<i>c</i>	an observer to be added to the observer vector.
----	----------	---

## Return values

<i>returns</i>	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	<i>wp</i>	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	<i>c</i>	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

<i>returns</i>	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

<i>A</i>	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

<i>a</i>	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

<i>returns</i>	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

<i>A</i>	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 <a href="#">Country</a> 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	<i>b</i>	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

<i>returns</i>	a pointer to a <a href="#">BattleState</a> 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 <a href="#">CountryBackup</a> 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

<a href="#">C</a>	A pointer to the opposing <a href="#">Country</a> 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 <a href="#">WarParticipant</a> object.
---	---

#### 4.16.1.2 createMissile()

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

function used to create missiles in the war.

##### Parameters

<i>damage</i>	the damage the missile will inflict.
---------------	--------------------------------------

##### Return values

<i>a</i>	pointer to a <a href="#">WarParticipant</a> 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	<i>damage</i>	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	<i>damage</i>	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	<i>c</i>	an observer to be added to the observer vector.
----	----------	---

##### Return values

<i>returns</i>	TRUE if the observer was added successfully and FALSE if the observer 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

<i>AN</i>	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

<i>returns</i>	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

<i>A</i>	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

<i>An</i>	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

<i>An</i>	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

<i>in</i>	<i>c</i>	an observer to be removed from the country's vector of observers.
-----------	----------	---

## Return values

<i>returns</i>	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	<a href="#">MachineGunner</a> 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 <a href="#">WarParticipant</a> 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

<a href="#">in</a>	<a href="#">HP</a>	A health value of the medic.
--------------------	--------------------	------------------------------

##### Return values

<a href="#">A</a>	pointer to a <a href="#">Medic</a> 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 <a href="#">Missile</a> 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 <a href="#">Rifleman</a> 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 [doNothing](#) ()=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 participants 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 participants in the war.

Return values

<i>An</i>	integer value containing the number of participants.
-----------	--

#### 4.34.2.5 getType()

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

A function used to get the type of a weapon.

Return values

<i>A</i>	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

<i>Returns</i>	a pointer to the <a href="#">WarParticipant</a> 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

<i>Returns</i>	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

<i>Returns</i>	a pointer to the <a href="#">WarShip</a> 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

<i>in</i>	<i>HP</i>	The amount of AirCrafts created.
-----------	-----------	----------------------------------

#### Return values

<i>Returns</i>	a pointer to a <a href="#">WarParticipant</a> 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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_AIRCRAFT_H
2 #define PRACTICAL_ASSIGNMENT_3_AIRCRAFT_H
3 #include "WarParticipant.h"
4 #include "WarParticipantFactory.h"
5
6 class AirCraft: public WarParticipant {
7 public:
8     AirCraft(int d, int h);
13     AirCraft* clone();
14
15     void doNothing();
16
17 };
18
19
20 #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.](#)

```
1 //
2 // Created by JOHANES MATSEBA on 2022/10/27.
3 //
4
5 #ifndef PROJECTASSIGNMENT_AIRSPACE_H
6 #define PROJECTASSIGNMENT_AIRSPACE_H
7 #include "WarTheatre.h"
8
9 class AirSpace: public WarTheatre{
10
11 };
12
13
14 #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.](#)

```
1 #ifndef ALLIANCE_H
2 #define ALLIANCE_H
3 #include "Country.h"
4 #include <iostream>
5 #include <string>
6 #include <list>
7
8 class CountryObservers;
9 class Alliance: public Country{
10 public:
11     Alliance();
12     void setAlliance(vector<Country*> alliance);
13     vector<Country*> getAlliance();
14     void addAlly(Country* ally);
15     void removeAlly(Country* ally);
16     int getHp();
17 private:
18     int allianceHp;
19     vector<Country*> alliance;
20
21 };
22
23 #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.](#)

```
1 #ifndef Attack_h
2 #define Attack_h
3 #include "BattleState.h"
4 #include "DetonateExplosives.h"
5 #include "Shoot.h"
6 #include "FireMissile.h"
7 #include "Context.h"
8 #include "Country.h"
9 #include "WarParticipant.h"
10
11 class Attack : public BattleState
12 {
13 public:
14     void handleChange(Country *C);
15 };
16 #endif
```

## 5.9 AttackStrategy.h File Reference

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

## Classes

- class [AttackStrategy](#)

## 5.10 AttackStrategy.h

[Go to the documentation of this file.](#)

```
1 #ifndef AttackStrategy_h
2 #define AttackStrategy_h
3 #include "Country.h"
4
5 class AttackStrategy
6 {
7 public:
8     virtual void LaunchAttack(Country *C) = 0;
9 };
10 #endif
```

## 5.11 BattleState.h File Reference

```
#include <iostream>
```

## Classes

- class [BattleState](#)

## 5.12 BattleState.h

[Go to the documentation of this file.](#)

```
1 //
2 // Created by JOHANES MATSEBA on 2022/10/26.
3 //
4
5 #ifndef PROJECTASSIGNMENT_BATTLESTATE_H
6 #define PROJECTASSIGNMENT_BATTLESTATE_H
7 class Country;
8 #include <iostream>
9
10 using namespace std;
11
12 class BattleState {
13 public:
14     BattleState();
15     BattleState *successor;
16     void Add(BattleState *Succ);
17     virtual void handleChange(Country *C);
18 };
19
20 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_BOMB_H
2 #define PRACTICAL_ASSIGNMENT_3_BOMB_H
3 #include "WarParticipant.h"
4 #include "WarParticipantFactory.h"
5
6 class Bomb: public WarParticipant {
7 public:
8     Bomb(int d, int h);
9     Bomb* clone();
10     void doNothing();
11 };
12
13 #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.](#)

```
1 #ifndef Context_h
2 #define Context_h
3 #include "AttackStrategy.h"
4 #include "Country.h"
5
6 class Context
7 {
8 private:
9     AttackStrategy *state;
10
11 public:
12     Context(AttackStrategy *s);
13     ~Context();
14     void SetState(AttackStrategy *s);
15     void implement(Country *C);
16 };
17
18 #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.](#)

```

1 //
2 // Created by JOHANES MATSEBA on 2022/10/19.
3 //
4
5 #ifndef PROJECTASSIGNMENT_COUNTRY_H
6 #define PROJECTASSIGNMENT_COUNTRY_H
7 #include <list>
8 #include "WarParticipant.h"
9 #include "WarTheatre.h"
10 #include "BattleState.h"
11 #include "CountryObserver.h"
12 #include "Sea.h"
13 #include "AirSpace.h"
14 #include "Land.h"
15 #include <iostream>
16 #include <cstdlib>
17 #include "time.h"
18 #include "CountryBackup.h"
19 #include <vector>
20 #include <string>
21 #include "CountryObserverIterator.h"
22 #include "WarParticipantIterator.h"
23 using namespace std;
24 #include <string>
25
26 class BattleState;
27 class WarTheatre;
28 class CountryObserverIterator;
29 class WarParticipantIterator;
30 class CountryIterator;
31
32 class Country {
33
34
35 public:
36     Country(std::string, bool);
37     Country(BattleState* state);
38     Country(Country* c);
39     Country();
40     BattleState* getBattleState();
41     Country* pickOpposingCountry(vector<Country*> c);
42
43     virtual void withdraw(){}
44     void setBattleState(BattleState* b);
45     void selectWarTheatre();
46     void createWarParticipants();
47     void attackOpposingCountry(Country* c); // state.handleChange(this);
48     virtual int getHp()=0;
49     CountryBackup* createBackup(); //Memento
50     void reinstateCountry(CountryBackup* cb);
51
52     virtual WarParticipantIterator* createWarParticipantIterator(); //Tseko Iterator
53     virtual void addWarParticipant(WarParticipant* wp);
54     virtual std::vector<CountryObserver*> getCountryObservers();
55     virtual std::string getName();
56     WarTheatre* getWarTheatre();
57     virtual bool add(CountryObserver* c); //Country Observer
58
59     virtual bool remove(CountryObserver* c);
60     virtual void notify();
61
62     void InflictDamage(int dmg); //Country BattleState
63     Country* getOpposingC();
64     void setHp(int HP);
65     vector<WarParticipant*> getArtillery();
66     void addArtillery(WarParticipant* W);
67     virtual ~Country();
68
69 private:
70     State* state;
71     BattleState* battlestate;
72     WarTheatre* warTheatre;
73     std::string cName;
74     vector<CountryObserver*> countryObservers;
75     Country* OpposingCountry;
76     bool win=false;
77     bool ocean;
78
79 protected:
80     int hp=1000;
81 };
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141

```

```

142
143 #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.](#)

```

1 //
2 // Created by JOHANES MATSEBA on 2022/10/24.
3 //
4
5 #ifndef PROJECTASSIGNMENT_COUNTRYBACKUP_H
6 #define PROJECTASSIGNMENT_COUNTRYBACKUP_H
7 #include <list>
8 #include "WarParticipant.h"
9 #include "WarTheatre.h"
10 #include "BattleState.h"
11 #include "CountryObserver.h"
12 #include "Sea.h"
13 #include "AirSpace.h"
14 #include "Land.h"
15 #include <iostream>
16 #include <cstdlib>
17 #include "time.h"
18 // #include "Country.h"
19 #include "State.h"
20 #include <vector>
21
22 using namespace std;
23 class CountryBackup {
24 private:
25     friend class Country;
26     int hp;
27     State* state;
28     Country* opposingCountry;
29     BattleState* battleState;
30     WarTheatre* warTheatre;
31     std::vector<WarParticipant*> warParticipants;
32     std::vector<CountryObserver*> countryObservers;
33     bool win;
34
35 public:
36     CountryBackup(int, BattleState*,
37         WarTheatre*, vector<WarParticipant*>, vector<CountryObserver*>, Country*, bool);
41     BattleState* getBattleState();

```

```
42     WarTheatre* getWarTheatre();
43     WarParticipant* getWarParticipants();
44     CountryObserver* getCountryObservers();
45     virtual ~CountryBackup();
46 };
47
48
49 #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.](#)

```
1 //
2 // Created by JOHANES MATSEBA on 2022/10/24.
3 //
4
5 #ifndef PROJECTASSIGNMENT_COUNTRYMEMORY_H
6 #define PROJECTASSIGNMENT_COUNTRYMEMORY_H
7 #include "CountryBackup.h"
8
9 class CountryMemory {
10 private:
11     CountryBackup* backup;
12
13 public:
14     CountryMemory();
19     CountryBackup* retrieveBackup();
24     void storeBackup(CountryBackup* s);
25     ~CountryMemory();
26 };
27
28
29 #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.](#)

```
1 #ifndef COUNTRYOBSERVER_H
2 #define COUNTRYOBSERVER_H
3
4 #include <iostream>
5 #include <string>
6 #include <vector>
7 using namespace std;
8
9 class CountryObserver { //Observer
10     public:
11         virtual void update()=0;
12 };
13 #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.](#)

```
1 #ifndef COUNTRYOBSERVERITERATOR_H
2 #define COUNTRYOBSERVERITERATOR_H
3 #include "IteratorTool.h"
4 #include <list>
5 #include "CountryObserver.h"
6 #include "Country.h"
7
8 class Country;
9
10 class CountryObserverIterator : public IteratorTool
11 {
12     friend class Country;
13     public:
14         CountryObserverIterator(Country*);
15         void first();
16         void next();
17         bool isLastEl();
18         CountryObserver* currentEl();
19
20     private:
21         std::vector<CountryObserver*>::iterator it;
22         std::vector<CountryObserver*> storeList;
23         CountryObserver* curr;
24 };
25 #endif
```

## 5.27 Defend.h File Reference

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

## Classes

- class [Defend](#)

## 5.28 Defend.h

[Go to the documentation of this file.](#)

```
1 #ifndef Defend_h
2 #define Defend_h
3 #include "BattleState.h"
4
5 class Defend : public BattleState
6 {
7 public:
13     void handleChange(Country *C);
14 };
15 #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.](#)

```
1 #ifndef DetonateExplosives_h
2 #define DetonateExplosives_h
3 #include "AttackStrategy.h"
4 #include <list>
5 #include <iostream>
6
7 class DetonateExplosives : public AttackStrategy
8 {
9 public:
14     void LaunchAttack(Country *C);
15 };
16
17 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
2 #define PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
3 #include "WarParticipantFactory.h"
4 #include "WarParticipant.h"
5
6 class ExplosiveFactory: public WarParticipantFactory{
7 public:
13     WarParticipant* createBomb(int damage);
20     WarParticipant* createMissile(int damage);
21 };
22
23
24 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_FIREARMFACTORY_H
2 #define PRACTICAL_ASSIGNMENT_3_FIREARMFACTORY_H
3 #include "WarParticipantFactory.h"
4 #include "WarParticipant.h"
5 #include "MachineGun.h"
6 #include "Rifle.h"
7
8 class FireArmFactory: public WarParticipantFactory{
9 public:
15     Rifle* createRifle(int damage);
21     MachineGun* createMachineGun(int damage);
22 };
23
24
25 #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.](#)

```
1 #ifndef FireMissile_h
2 #define FireMissile_h
3 #include "AttackStrategy.h"
4 #include <list>
5 #include <iostream>
6
7 class FireMissile : public AttackStrategy
8 {
9 public:
14     void LaunchAttack(Country *C);
15 };
16
17 #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.](#)

```
1 #ifndef INDIVIDUALCOUNTRY_H
2 #define INDIVIDUALCOUNTRY_H
3 #include "Country.h"
4 #include "CountryObserver.h"
5 #include <iostream>
6 #include <string>
7 #include <vector>
8 using namespace std;
9
10 class CountryObservers;
11 class IndividualCountry: public Country{
12 public:
13     IndividualCountry(string n);
14     bool add(CountryObserver* c);
15     bool remove(CountryObserver* c);
19     void notify();
20     string getName();
25     int getWeaponHP();
30     int getSoldierHP();
35     int getTransport();
36     int getSize();
41     void checkHp();
46     int getInitialHP();
47     void setInitial();
48     void setAlliance(vector<IndividualCountry*> alliance);
```

```

49     vector<IndividualCountry*> getAlliance();
50     vector<WarParticipant*> getWarParticipants();
51 private:
52     vector<CountryObserver*> countryObservers;
53     vector<IndividualCountry*> alliance;
54     vector<WarParticipant*> wP;
55     int weaponHP;
56     int soldierHP;
57     int transport;
58     int currWeaponHP;
59     int initialHP;
60     string name;
61 };
62 #endif

```

## 5.39 IteratorTool.h File Reference

### Classes

- class [IteratorTool](#)

## 5.40 IteratorTool.h

[Go to the documentation of this file.](#)

```

1 #ifndef ITERATORTOOL_H
2 #define ITERATORTOOL_H
3
4 class IteratorTool
5 {
6 public:
7     IteratorTool();
8     virtual ~IteratorTool();
9     virtual void first() = 0;
10    virtual void next() = 0;
11    virtual bool isLastEl() = 0;
12 };
13 #endif

```

## 5.41 Land.h File Reference

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

### Classes

- class [Land](#)

## 5.42 Land.h

[Go to the documentation of this file.](#)

```

1 //
2 // Created by JOHANES MATSEBA on 2022/10/27.
3 //
4
5 #ifndef PROJECTASSIGNMENT_LAND_H
6 #define PROJECTASSIGNMENT_LAND_H
7 #include "WarTheatre.h"
8
9 class Land: public WarTheatre {
10
11 };
12
13
14 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_MACHINEGUNNER_H  
2 #define PRACTICAL_ASSIGNMENT_3_MACHINEGUNNER_H  
3 #include "WarParticipant.h"  
4 #include "WarParticipantFactory.h"  
5  
6 class MachineGunner: public WarParticipant {  
7 public:  
8     MachineGunner(int d, int h);  
13     MachineGunner* clone();  
14     void doNothing();  
15 };  
16  
17  
18 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_MEDIC_H  
2 #define PRACTICAL_ASSIGNMENT_3_MEDIC_H  
3 #include "WarParticipant.h"  
4 #include "WarParticipantFactory.h"  
5  
6 class Medic: public WarParticipant {  
7 private:  
8     int HP;  
9 public:  
10     Medic(int hp);  
15     WarParticipant* clone();  
16     void doNothing();  
17 };  
18  
19  
20 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_MEDICFACTORY_H
2 #define PRACTICAL_ASSIGNMENT_3_MEDICFACTORY_H
3 #include "WarParticipantFactory.h"
4 #include "Medic.h"
5
6 class MedicFactory: public WarParticipantFactory {
7 public:
13     Medic* createMedic(int HP);
14 };
15
16 #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.](#)

```
1 #ifndef MEDICS_H
2 #define MEDICS_H
3 #include "IndividualCountry.h"
4 #include "CountryObserver.h"
5 #include "Country.h"
6 #include <iostream>
7 #include <string>
8 #include <vector>
9
10
11 using namespace std;
12
13 class Medics : public CountryObserver { //Concrete Observer
```

```

14     public:
15         Medics(IndividualCountry* currentCountry);
19         void update();
20     private:
21         int observedHP;
22         int observedS;
23         int observedT;
24         int maxHeal=0;
25         int healingHp=90;
26         bool medicObserved=false; //set to false if true, when Allies send medics
27         IndividualCountry* currentCountry;
28         vector<WarParticipant*> warParticipants;
29 };
30 #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.](#)

```

1 #ifndef PRACTICAL_ASSIGNMENT_3_MISSILE_H
2 #define PRACTICAL_ASSIGNMENT_3_MISSILE_H
3 #include "WarParticipant.h"
4 #include "WarParticipantFactory.h"
5
6 class Missile: public WarParticipant {
7 public:
8     Missile(int d, int h);
13     Missile* clone();
14     void doNothing();
15 };
16
17
18 #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.](#)

```
1 #ifndef OBSERVINGALLIES_H
2 #define OBSERVINGALLIES_H
3 #include "BattleState.h"
4 #include "IndividualCountry.h"
5 #include <iostream>
6 #include <string>
7 #include <vector>
8
9
10 using namespace std;
11
12 class ObservingAllies : public CountryObserver{//Concrete Observers
13     public:
14         ObservingAllies(IndividualCountry* currentCountry);
15         void update();
16     private:
17         int observedH;
18         IndividualCountry* currentCountry;
19         vector<WarParticipant*> warParticipants;
20 };
21 #endif
```

## 5.55 RequestAlliance.h File Reference

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

### Classes

- class [RequestAlliance](#)

## 5.56 RequestAlliance.h

[Go to the documentation of this file.](#)

```
1 #ifndef RequestAlliance_h
2 #define RequestAlliance_h
3 #include "BattleState.h"
4
5 class RequestAlliance : public BattleState
6 {
7     public:
12         void handleChange(Country *C);
13 };
14 #endif
```

## 5.57 Rifleman.h File Reference

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

### Classes

- class [Rifleman](#)

## 5.58 Rifleman.h

[Go to the documentation of this file.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_RIFLEMAN_H
2 #define PRACTICAL_ASSIGNMENT_3_RIFLEMAN_H
3 #include "WarParticipant.h"
4
5 class Rifleman: public WarParticipant{
6 public:
7     Rifleman(int d, int h);
12     Rifleman* clone();
13     void doNothing();
14 };
15
16
17 #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.](#)

```
1 //
2 // Created by JOHANES MATSEBA on 2022/10/27.
3 //
4
5 #ifndef PROJECTASSIGNMENT_SEA_H
6 #define PROJECTASSIGNMENT_SEA_H
7
8
9 #include "WarTheatre.h"
10
11 class Sea: public WarTheatre {
12
13 };
14
15
16 #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.](#)

```
1 #ifndef Shoot_h
2 #define Shoot_h
3 #include "AttackStrategy.h"
4 #include <list>
5 #include <iostream>
6
7 class Shoot : public AttackStrategy
8 {
9 public:
14     void LaunchAttack(Country *C);
15 };
16
17 #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.](#)

```
1 //
2 // Created by JOHANES MATSEBA on 2022/10/29.
3 //
4
5 #ifndef PROJECTASSIGNMENT_STATE_H
6 #define PROJECTASSIGNMENT_STATE_H
7
8 #include <list>
9 #include "WarParticipant.h"
10 #include "WarTheatre.h"
11 #include "BattleState.h"
12 #include "CountryObserver.h"
13 #include "Sea.h"
14 #include "AirSpace.h"
15 #include "Land.h"
16 #include <iostream>
17 #include <cstdlib>
18 #include "time.h"
19
20 #include <vector>
21
22
23 using namespace std;
```

```

24 class Country;
25 class State {
26
27 public:
28     State(int, BattleState*, WarTheatre*, vector<WarParticipant*>, vector<CountryObserver*>, Country*, bool);
29     int getHP() {return hp;};
30     BattleState* getBattleState() {return battlestate;};
31     WarTheatre* getWarTheatre() {return warTheatre;};
32     vector<WarParticipant*> getWarParticipants() {return warParticipants;};
33     vector<CountryObserver*> getCountryObservers() {return countryObservers;};
34     Country* getOppCountry() {return opposingCountry;};
35     bool getW() {return win;};
36 private:
37     int hp;
38     BattleState* battlestate;
39     WarTheatre* warTheatre;
40     std::vector<WarParticipant*> warParticipants;
41     std::vector<CountryObserver*> countryObservers;
42     Country* opposingCountry;
43     bool win=false;
44 };
45
46
47 #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.](#)

```

1 #ifndef Surrender_h
2 #define Surrender_h
3 #include "BattleState.h"
4
5 class Surrender : public BattleState
6 {
7 public:
12     void handleChange(Country *C);
13 };
14 #endif

```

## 5.67 WarParticipant.h File Reference

```
#include <string>
```

### Classes

- class [WarParticipant](#)

## 5.68 WarParticipant.h

[Go to the documentation of this file.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_WARPARTICIPANT_H
2 #define PRACTICAL_ASSIGNMENT_3_WARPARTICIPANT_H
3
4 #include <string>
5 using namespace std;
6
7 class WarParticipant{
8
9 public:
10     WarParticipant(string manufacturer, string type, int d, int h);
11     virtual void doNothing()=0;
12     std::string getType();
13     void incrementParticipantNumber();
14     int getNumParticipants();
15     int getHP();
16     int getDamage();
17 private:
18     string manufacturer;
19     string type;
20     int damage;
21     int hp;
22     int numParticipants; //Iterator
23 };
24
25 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_WARPARTICIPANTFACTORY_H
2 #define PRACTICAL_ASSIGNMENT_3_WARPARTICIPANTFACTORY_H
3 #include "WarParticipantFactory.h"
4 #include "WarParticipant.h"
5
6 class WarParticipantFactory {
7 //public:
8     //WarParticipantFactory();
9     //~WarParticipantFactory();
10
11 };
12
13
14 #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.](#)

```

1 #ifndef WARPARTICIPANTITERATOR_H
2 #define WARPARTICIPANTITERATOR_H
3 #include "IteratorTool.h"
4 #include <list>
5 #include "WarParticipant.h"
6 #include "Country.h"
7
8 class Country;
9
10 class WarParticipantIterator : public IteratorTool
11 {
12     friend class Country;
13     public:
14         WarParticipantIterator(Country*);
18         void first();
22         void next();
27         bool isLastEl();
32         WarParticipant* currentEl();
33
34     private:
35         std::vector<WarParticipant*>::iterator it;
36         std::vector<WarParticipant*> storeList;
37         WarParticipant* curr;
38 };
39 #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.](#)

```

1 #ifndef PRACTICAL_ASSIGNMENT_3_WARSHIP_H
2 #define PRACTICAL_ASSIGNMENT_3_WARSHIP_H
3 #include "WarParticipant.h"
4 #include "WarParticipantFactory.h"
5
6 class WarShip: public WarParticipant {
7     public:
8         WarShip(int d, int h);
13         WarShip* clone();
14         void doNothing();
15 };
16
17
18 #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.](#)

```
1 //
2 // Created by JOHANES MATSEBA on 2022/10/26.
3 //
4
5 #ifndef PROJECTASSIGNMENT_WARTHEATRE_H
6 #define PROJECTASSIGNMENT_WARTHEATRE_H
7
8
9 class WarTheatre {
10
11 };
12
13
14 #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.](#)

```
1 #ifndef PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
2 #define PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
3 #include "WarParticipantFactory.h"
4 #include "WarParticipant.h"
5
6 class WarTransportFactory: public WarParticipantFactory{
7 public:
13     WarParticipant* createAirCraft(int HP);
14     WarParticipant* createWarShip(int HP);
15 };
16
17
18 #endif //PRACTICAL_ASSIGNMENT_3_EXPLOSIVEFACTORY_H
```

