Project People

Generated by Doxygen 1.9.5

Chapter 1

COS214-Project

Project Description: COS214 end-of-year group project.

Group Members: Ethan - u19042699 Wian - u21433748 Matthew - u20514612 Franko - u20454661 Amicke - 20532009 Theodor - 04525087

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

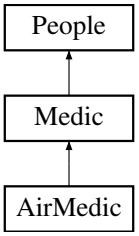
Chapter 5

Class Documentation

5.1 AirMedic Class Reference

`#include <AirMedic.h>`

Inheritance diagram for AirMedic:



Public Member Functions

- AirMedic ()

  Air Medics start alive with default damage multiplier of 1.
- ~AirMedic () override
- int act ()

Additional Inherited Members

5.1.1 Detailed Description

Author

Ethan

Date

17 October 2022

Specialisation of Medic - those stationed in the air

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AirMedic()

```
AirMedic::AirMedic ( )
```

Air Medics start alive with default damage multiplier of 1.

5.1.2.2 ~AirMedic()

```
AirMedic::~AirMedic ( )  [override]
```

5.1.3 Member Function Documentation

5.1.3.1 act()

```
int AirMedic::act ( )  [virtual]
```

AirMedic deals damage - multiplier times amount determined by alive, injured or dead

Returns

damage dealt

Reimplemented from Medic.

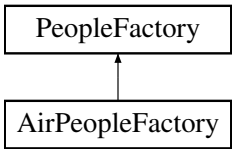
The documentation for this class was generated from the following files:

- AirMedic.h
- AirMedic.cpp

5.2 AirPeopleFactory Class Reference

```
#include <AirPeopleFactory.h>
```

Inheritance diagram for AirPeopleFactory:



Public Member Functions

- AirPeopleFactory ()
- ~AirPeopleFactory ()
- People * createSoldier ()
- People * createMedic ()
- People * createCitizen ()

5.2.1 Detailed Description

Author

Ethan

Date

17 October 2022

A specialisation of the PeopleFactory class, the AirPeopleFactory creates Pilot and AirMedic classes/objects

See also

AirMedic

Pilot

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AirPeopleFactory()

```
AirPeopleFactory::AirPeopleFactory ( )
```

5.2.2.2 ~AirPeopleFactory()

```
AirPeopleFactory::~AirPeopleFactory ( )
```

5.2.3 Member Function Documentation

5.2.3.1 createCitizen()

```
People * AirPeopleFactory::createCitizen ( )  [virtual]
```

Implements PeopleFactory.

5.2.3.2 createMedic()

```
People * AirPeopleFactory::createMedic ( )  [virtual]
```

Implements PeopleFactory.

5.2.3.3 createSoldier()

```
People * AirPeopleFactory::createSoldier ( )  [virtual]
```

Implements PeopleFactory.

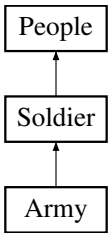
The documentation for this class was generated from the following files:

- AirPeopleFactory.h
- AirPeopleFactory.cpp

5.3 Army Class Reference

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

Inheritance diagram for Army:



Public Member Functions

- Army ()
- ~Army ()
- int act ()

Additional Inherited Members

5.3.1 Detailed Description

Author

Ethan

Date

17 October 2022

The Soldiers represented to be stationed on land

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Army()

```
Army::Army ( )
```

Soldiers start alive with a default damage of 3

5.3.2.2 ~Army()

```
Army::~Army ( )
```

5.3.3 Member Function Documentation

5.3.3.1 act()

```
int Army::act ( )  [virtual]
```

Soldier does damage - amount of which is determined by base damage times their capable damage based off of their state whether alive, injured or dead.

Returns

damage dealt

Reimplemented from Soldier.

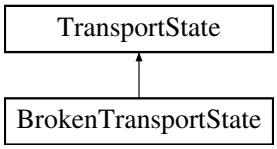
The documentation for this class was generated from the following files:

- Army.h
- Army.cpp

5.4 BrokenTransportState Class Reference

`#include <BrokenTransportState.h>`

Inheritance diagram for BrokenTransportState:



Public Member Functions

- float handle ()

  returns a value that can be used to determine the amount of damage that a country is able to do

5.4.1 Detailed Description

Author

Franko Swanepoel

Date

2022/10/24

A specialization of the TransportState class - used to indicate the broken state of the Transport class

5.4.2 Member Function Documentation

5.4.2.1 handle()

`float BrokenTransportState::handle ( )  [virtual]`

returns a value that can be used to determine the amount of damage that a country is able to do

Returns

int of 0.25

Implements TransportState.

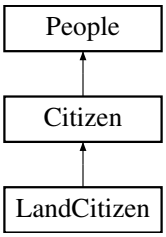
The documentation for this class was generated from the following files:

- BrokenTransportState.h
- BrokenTransportState.cpp

5.5 Citizen Class Reference

`#include <Citizen.h>`

Inheritance diagram for Citizen:



Public Member Functions

- virtual ~Citizen ()
- virtual int act ()

Additional Inherited Members

5.5.1 Detailed Description

Author

Ethan

Date

17 October 2022 A specialisation of the People class - Citizen represents a citizen of a country

5.5.2 Constructor & Destructor Documentation

5.5.2.1 ~Citizen()

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

5.5.3 Member Function Documentation

5.5.3.1 act()

```
int Citizen::act ( )  [virtual]
```

Citizen object carries out their action

Returns

-1 to indicate successful execution

Implements People.

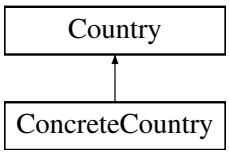
The documentation for this class was generated from the following files:

- Citizen.h
- Citizen.cpp

5.6 ConcreteCountry Class Reference

```
#include <ConcreteCountry.h>
```

Inheritance diagram for ConcreteCountry:



Public Member Functions

- ConcreteCountry ()
  base Constructor for concreteCountry
- ConcreteCountry (string name)
  Paramaterized Constructor for ConcreteCountry.
- virtual ~ConcreteCountry ()
- int attack () override
  attack() method sums the act() method of all people in this country's people[]
- void defend (int dmg) override
  defend() method uses the passed in integer to determine how much damage the country should take
- void heal () override
  heal() method allows the country to heal people based on the number of medics that are alive in the country
- Country * clone ()
  is able to return a clone of the country

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 ConcreteCountry() [1/2]

```
ConcreteCountry::ConcreteCountry ( )
```

base Constructor for concreteCountry

5.6.1.2 ConcreteCountry() [2/2]

```
ConcreteCountry::ConcreteCountry (
            string name )
```

Paramaterized Constructor for ConcreteCountry.

Parameters

name	Country Name as string

5.6.1.3 ~ConcreteCountry()

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

5.6.2 Member Function Documentation

5.6.2.1 attack()

```
int ConcreteCountry::attack ( )  [override], [virtual]
```

attack() method sums the act() method of all people in this country's people[]

Implements Country.

5.6.2.2 clone()

```
Country * ConcreteCountry::clone ( )  [virtual]
```

is able to return a clone of the country

Returns

Country* to the clone

Implements Country.

5.6.2.3 defend()

```
void ConcreteCountry::defend (
            int dmg )  [override], [virtual]
```

defend() method uses the passed in integer to determine how much damage the country should take

Parameters

dmg	the damage that the country will take

Implements Country.

5.6.2.4 heal()

```
void ConcreteCountry::heal ( )  [override], [virtual]
```

heal() method allows the country to heal people based on the number of medics that are alive in the country

Implements Country.

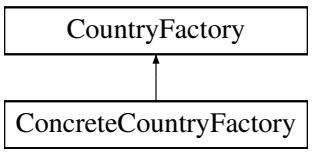
The documentation for this class was generated from the following files:

- ConcreteCountry.h
- ConcreteCountry.cpp

5.7 ConcreteCountryFactory Class Reference

```
#include <ConcreteCountryFactory.h>
```

Inheritance diagram for ConcreteCountryFactory:



Public Member Functions

- ConcreteCountryFactory ()

  Constructor.
- ~ConcreteCountryFactory ()
- Country * produceCountry (string name) override

  Creates and returns a new Country class object.

5.7.1 Constructor & Destructor Documentation

5.7.1.1 ConcreteCountryFactory()

```
ConcreteCountryFactory::ConcreteCountryFactory ( )
```

Constructor.

5.7.1.2 ~ConcreteCountryFactory()

```
ConcreteCountryFactory::~ConcreteCountryFactory ( )
```

5.7.2 Member Function Documentation

5.7.2.1 produceCountry()

```
Country * ConcreteCountryFactory::produceCountry (
            string name )  [override], [virtual]
```

Creates and returns a new Country class object.

Parameters

name	name of the country that will be instantiated

Returns

Country pointer to the new ConcreteCountry

Implements CountryFactory.

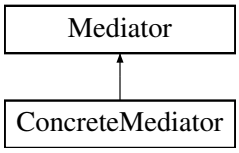
The documentation for this class was generated from the following files:

- ConcreteCountryFactory.h
- ConcreteCountryFactory.cpp

5.8 ConcreteMediator Class Reference

`#include <ConcreteMediator.h>`

Inheritance diagram for ConcreteMediator:



Public Member Functions

- ConcreteMediator ()
- ∼ConcreteMediator ()
- virtual void notify (ConcreteCountry ∗country)

  Notifies The specified Country.

5.8.1 Constructor & Destructor Documentation

5.8.1.1 ConcreteMediator()

```
ConcreteMediator::ConcreteMediator ( )
```

5.8.1.2 ∼ConcreteMediator()

```
ConcreteMediator::∼ConcreteMediator ( )
```

5.8.2 Member Function Documentation

5.8.2.1 notify()

```
void ConcreteMediator::notify (
            ConcreteCountry * country )  [virtual]
```

Notifies The specified Country.

Parameters

country	Country object to be notified

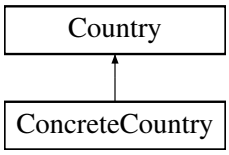
The documentation for this class was generated from the following files:

- ConcreteMediator.h
- ConcreteMediator.cpp

5.9 Country Class Reference

`#include <Country.h>`

Inheritance diagram for Country:



Public Member Functions

- Country ()
- Country (string name)

  Instantiates Country Object.
- virtual ∼Country ()
- virtual int attack ()=0

  Pure virtual attack() function.
- virtual void defend (int damage)=0

  Pure virtual defend() function.
- virtual void heal ()=0

  Pure virtual heal() function.
- void randomPeople ()

  Populates the people vector with between 20 - 30 people of random Jobs.
- string getName ()

  Getter for country name.
- int getNumPeople () const

  getter for numPeople(Population)
- int getNumAlive () const

  is able to get the total number of people that are alive within a country
- int getNumInjured () const

  is able to get the total number of people that are injured within a country
- void fixTransport ()

  method allowing the Country's transport state to be changed to the fixed state
- void breakTransport ()

  method allowing the Country's transport state to be changed to the broken state
- float requestTransport ()

  allows the country to request the transport
- bool isAlive () const

  checks whether the country has any people left alive in it's array
- virtual Country ∗ clone ()=0

  is able to return a clone of the country

Static Public Member Functions

- static int randomNumInRange (int min, int max)

  Generates a random number in the range.

Protected Attributes

- int numAlive
- Transport * transport
- vector< People * > citizens

5.9.1 Constructor & Destructor Documentation

5.9.1.1 Country() [1/2]

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

5.9.1.2 Country() [2/2]

```
Country::Country (
            string name )
```

Instantiates Country Object.

Parameters

name	Name of the country

5.9.1.3 ~Country()

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

5.9.2 Member Function Documentation

5.9.2.1 attack()

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

Pure virtual attack() function.

Returns

Damage number that the country will deal

Implemented in ConcreteCountry.

5.9.2.2 breakTransport()

```
void Country::breakTransport ( )
```

method allowing the Country's transport state to be changed to the broken state

See also

WorkingTransportState

5.9.2.3 clone()

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

is able to return a clone of the country

Returns

Country* to the clone

Implemented in ConcreteCountry.

5.9.2.4 defend()

```
virtual void Country::defend (
            int damage )  [pure virtual]
```

Pure virtual defend() function.

Parameters

damage	Damage number that the country will take

Implemented in ConcreteCountry.

5.9.2.5 fixTransport()

```
void Country::fixTransport ( )
```

method allowing the Country's transport state to be changed to the fixed state

See also

BrokenTransportState

5.9.2.6 getName()

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

Getter for country name.

Returns

string Country name

5.9.2.7 getNumAlive()

```
int Country::getNumAlive ( ) const
```

is able to get the total number of people that are alive within a country

Returns

int of the number of people alive in the country

5.9.2.8 getNumInjured()

```
int Country::getNumInjured ( ) const
```

is able to get the total number of people that are injured within a country

Returns

int of the number of people injured in the country

5.9.2.9 getNumPeople()

```
int Country::getNumPeople ( ) const
```

getter for numPeople(Population)

Returns

int numPeople

5.9.2.10 heal()

```
virtual void Country::heal ( )  [pure virtual]
```

Pure virtual heal() function.

Implemented in ConcreteCountry.

5.9.2.11 isAlive()

```
bool Country::isAlive ( ) const
```

checks whether the country has any people left alive in it's array

Returns

returns true if the country is still alive

5.9.2.12 randomNumInRange()

```
int Country::randomNumInRange (
            int min,
            int max )  [static]
```

Generates a random number in the range.

Parameters

min	Minimum int of range
max	Maximum int of range

Returns

int between min and max

5.9.2.13 randomPeople()

```
void Country::randomPeople ( )
```

Populates the people vector with between 20 - 30 people of random Jobs.

5.9.2.14 requestTransport()

```
float Country::requestTransport ( )
```

allows the country to request the transport

Returns

float of the value that the transport state returns

5.9.3 Member Data Documentation

5.9.3.1 citizens

```
vector<People*> Country::citizens  [protected]
```

5.9.3.2 numAlive

```
int Country::numAlive  [protected]
```

5.9.3.3 transport

```
Transport* Country::transport  [protected]
```

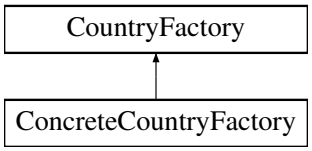
The documentation for this class was generated from the following files:

- Country.h
- Country.cpp

5.10 CountryFactory Class Reference

```
#include <CountryFactory.h>
```

Inheritance diagram for CountryFactory:



Public Member Functions

- CountryFactory ()
- virtual ~CountryFactory ()
- virtual Country * produceCountry (string name)=0

  Pure virtual function for Producing country.

5.10.1 Constructor & Destructor Documentation

5.10.1.1 CountryFactory()

```
CountryFactory::CountryFactory ( )
```

5.10.1.2 ~CountryFactory()

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

5.10.2 Member Function Documentation

5.10.2.1 produceCountry()

```
virtual Country * CountryFactory::produceCountry (
            string name )  [pure virtual]
```

Pure virtual function for Producing country.

Parameters

name	Country Name

Returns

new concreteCountry Object

Implemented in ConcreteCountryFactory.

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

- CountryFactory.h
- CountryFactory.cpp

5.11 Facade Class Reference

`#include <Facade.h>`

Public Member Functions

- Facade ()
- ~Facade ()
- void gameStart ()

  The WarEngine's loop() function is called thrice.

5.11.1 Detailed Description

Author

Ethan

Date

3 November 2022

The Façade cleans up management of the WarEngine

See also

WarEngine

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Facade()

```
Facade::Facade ( )
```

5.11.2.2 ~Facade()

```
Facade::~Facade ( )
```

5.11.3 Member Function Documentation

5.11.3.1 gameStart()

```
void Facade::gameStart ( )
```

The WarEngine's loop() function is called thrice.

See also

WarEngine

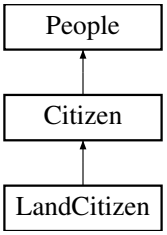
The documentation for this class was generated from the following files:

- Facade.h
- Facade.cpp

5.12 LandCitizen Class Reference

```
#include <LandCitizen.h>
```

Inheritance diagram for LandCitizen:



Public Member Functions

- LandCitizen ()

Additional Inherited Members

5.12.1 Detailed Description

Author

Ethan

Date

17 October 2022

Sole specialisation of Citizen class - only on land ∗

5.12.2 Constructor & Destructor Documentation

5.12.2.1 LandCitizen()

```
LandCitizen::LandCitizen ( )
```

LandCitizens start alive with a default damage multiplier of 2

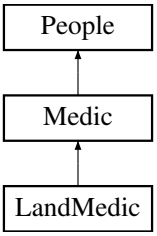
The documentation for this class was generated from the following files:

- LandCitizen.h
- LandCitizen.cpp

5.13 LandMedic Class Reference

```
#include <LandMedic.h>
```

Inheritance diagram for LandMedic:



Public Member Functions

- LandMedic ()
- ∼LandMedic ()
- int act ()

Additional Inherited Members

5.13.1 Detailed Description

Author

Ethan

Date

17 October 2022

Medics stationed on the ground

5.13.2 Constructor & Destructor Documentation

5.13.2.1 LandMedic()

```
LandMedic::LandMedic ( )
```

Medics start alive with a default damage multiplier of 1

5.13.2.2 ~LandMedic()

```
LandMedic::~LandMedic ( )
```

5.13.3 Member Function Documentation

5.13.3.1 act()

```
int LandMedic::act ( )  [virtual]
```

Medic deals damage (if needed) - multiplier times damage determined by alive, dead or injured.

Returns

damage dealt

Reimplemented from Medic.

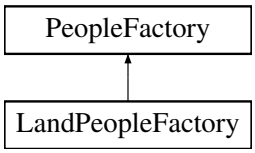
The documentation for this class was generated from the following files:

- LandMedic.h
- LandMedic.cpp

5.14 LandPeopleFactory Class Reference

`#include <LandPeopleFactory.h>`

Inheritance diagram for LandPeopleFactory:



Public Member Functions

- LandPeopleFactory ()
- ~LandPeopleFactory ()
- People * createSoldier ()
- People * createMedic ()
- People * createCitizen ()

5.14.1 Detailed Description

Author

Ethan

Date

17 October 2022

A specialisation of the PeopleFactory class - it creates Army, LandCitizen and LandMedic objects.

See also

LandCitizen

Army

LandMedic

5.14.2 Constructor & Destructor Documentation

5.14.2.1 LandPeopleFactory()

`LandPeopleFactory::LandPeopleFactory ( )`

5.14.2.2 ~LandPeopleFactory()

```
LandPeopleFactory::~LandPeopleFactory ( )
```

5.14.3 Member Function Documentation

5.14.3.1 createCitizen()

```
People * LandPeopleFactory::createCitizen ( )  [virtual]
```

Implements PeopleFactory.

5.14.3.2 createMedic()

```
People * LandPeopleFactory::createMedic ( )  [virtual]
```

Implements PeopleFactory.

5.14.3.3 createSoldier()

```
People * LandPeopleFactory::createSoldier ( )  [virtual]
```

Implements PeopleFactory.

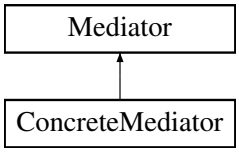
The documentation for this class was generated from the following files:

- LandPeopleFactory.h
- LandPeopleFactory.cpp

5.15 Mediator Class Reference

```
#include <Mediator.h>
```

Inheritance diagram for Mediator:



Public Member Functions

- Mediator ()
- virtual ~Mediator ()
- virtual void notify ()=0

  Notifies attached objects.

5.15.1 Constructor & Destructor Documentation

5.15.1.1 Mediator()

```
Mediator::Mediator ( )  [default]
```

5.15.1.2 ~Mediator()

```
Mediator::~Mediator ( )  [virtual], [default]
```

5.15.2 Member Function Documentation

5.15.2.1 notify()

```
virtual void Mediator::notify ( )  [pure virtual]
```

Notifies attached objects.

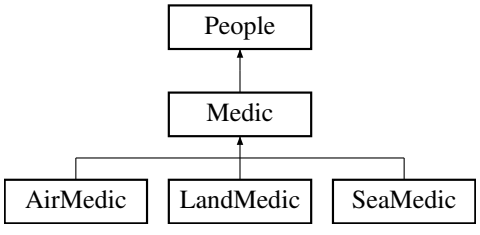
The documentation for this class was generated from the following files:

- Mediator.h
- Mediator.cpp

5.16 Medic Class Reference

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

Inheritance diagram for Medic:



Public Member Functions

- virtual ~Medic ()
- virtual int act ()

Additional Inherited Members

5.16.1 Detailed Description

Author

Ethan

Date

17 October 2022

A specialisation of the People class - Medic represents the role of a medic in the war scene

5.16.2 Constructor & Destructor Documentation

5.16.2.1 ~Medic()

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

5.16.3 Member Function Documentation

5.16.3.1 act()

```
int Medic::act ( )  [virtual]
```

The Medic object carries out their healing action

Returns

-1 to indicate execution

Implements People.

Reimplemented in AirMedic, LandMedic, and SeaMedic.

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

- Medic.h
- Medic.cpp

5.17 Memento Class Reference

`#include <Memento.h>`

Public Member Functions

- virtual ~Memento ()

  stateMem object will be deleted and state will be set to 0

Friends

- class WarEngine

5.17.1 Detailed Description

Author

Amicke

Date

2022/11/03

5.17.2 Constructor & Destructor Documentation

5.17.2.1 ~Memento()

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

stateMem object will be deleted and state will be set to 0

5.17.3 Friends And Related Function Documentation

5.17.3.1 WarEngine

`friend class WarEngine  [friend]`

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

- Memento.h
- Memento.cpp

5.18 mementostorage Class Reference

`#include <mementostorage.h>`

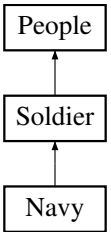
The documentation for this class was generated from the following files:

- mementostorage.h
- mementostorage.cpp

5.19 Navy Class Reference

`#include <Navy.h>`

Inheritance diagram for Navy:



Public Member Functions

- Navy ()
- ~Navy ()
- int act ()

Additional Inherited Members

5.19.1 Detailed Description

Author

Ethan

Date

17 October 2022

Soldiers stationed at sea

5.19.2 Constructor & Destructor Documentation

5.19.2.1 Navy()

```
Navy::Navy ( )
```

Soldiers in the navy start alive with default damage multiplier of 3

5.19.2.2 ~Navy()

```
Navy::~Navy ( )
```

5.19.3 Member Function Documentation

5.19.3.1 act()

```
int Navy::act ( )  [virtual]
```

Navy soldier deals damage - default times amount determined by alive, injured or dead

Returns

damage dealt

Reimplemented from Soldier.

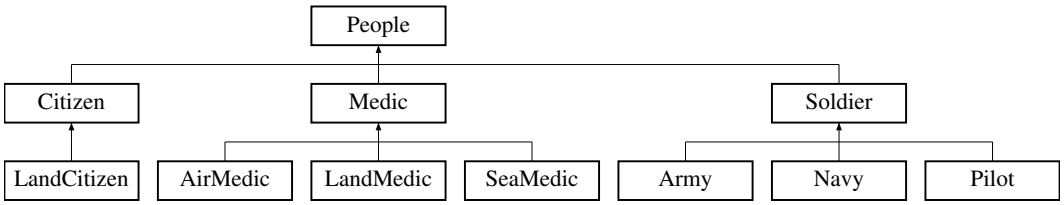
The documentation for this class was generated from the following files:

- Navy.h
- Navy.cpp

5.20 People Class Reference

```
#include <People.h>
```

Inheritance diagram for People:



Public Member Functions

- People ()
- virtual ~People ()
- virtual int act ()=0
- void changeStateDead ()
- void changeStateInjured ()
- void changeStateAlive ()

Public Attributes

- int dmg

  The damage a People object can do.
- PeopleStatus * state

5.20.1 Detailed Description

Author

Ethan

Date

17 October 2022

Represents a specific country's military capacity and population

5.20.2 Constructor & Destructor Documentation

5.20.2.1 People()

```
People::People ( )
```

5.20.2.2 ~People()

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

5.20.3 Member Function Documentation

5.20.3.1 act()

```
virtual int People::act ( )  [pure virtual]
```

Implemented in AirMedic, Army, Citizen, LandMedic, Medic, Navy, Pilot, SeaMedic, and Soldier.

5.20.3.2 changeStateAlive()

```
void People::changeStateAlive ( )
```

Changes the person's state to alive

See also

PeopleAliveState

5.20.3.3 changeStateDead()

```
void People::changeStateDead ( )
```

Changes the person's state to dead

See also

PeopleDeadState

5.20.3.4 changeStateInjured()

```
void People::changeStateInjured ( )
```

Changes the person's state to injured

See also

PeopleInjuredState

5.20.4 Member Data Documentation

5.20.4.1 dmg

```
int People::dmg
```

The damage a People object can do.

5.20.4.2 state

```
PeopleStatus* People::state
```

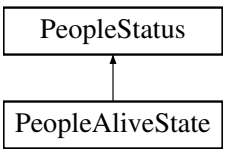
The documentation for this class was generated from the following files:

- People.h
- People.cpp

5.21 PeopleAliveState Class Reference

```
#include <PeopleAliveState.h>
```

Inheritance diagram for PeopleAliveState:



Public Member Functions

- PeopleAliveState ()
- int handle ()

5.21.1 Detailed Description

Author

Ethan

Date

17 October 2022

See also

People

Represents the state of a People object as being alive

5.21.2 Constructor & Destructor Documentation

5.21.2.1 PeopleAliveState()

```
PeopleAliveState::PeopleAliveState ( )
```

5.21.3 Member Function Documentation

5.21.3.1 handle()

```
int PeopleAliveState::handle ( )  [virtual]
```

The damage a People object can do in the Alive state

Returns

Damage the People object is capable of whilst "alive", in this case, 2

Implements PeopleStatus.

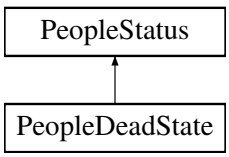
The documentation for this class was generated from the following files:

- PeopleAliveState.h
- PeopleAliveState.cpp

5.22 PeopleDeadState Class Reference

```
#include <PeopleDeadState.h>
```

Inheritance diagram for PeopleDeadState:



Public Member Functions

- PeopleDeadState ()
- int handle ()

5.22.1 Detailed Description

Author

Ethan

Date

17 October 2022

See also

People

Represents the state of being date of a People object

5.22.2 Constructor & Destructor Documentation

5.22.2.1 PeopleDeadState()

```
PeopleDeadState::PeopleDeadState ( )
```

5.22.3 Member Function Documentation

5.22.3.1 handle()

```
int PeopleDeadState::handle ( )  [virtual]
```

The damage a People object can do while dead

Returns

0 since logically no damage can be dealt while dead (for humans)

Implements PeopleStatus.

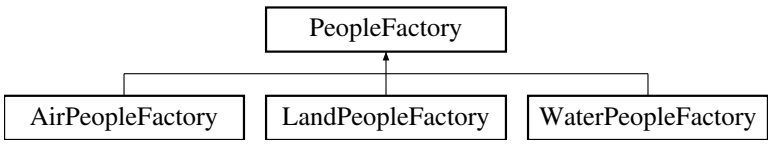
The documentation for this class was generated from the following files:

- PeopleDeadState.h
- PeopleDeadState.cpp

5.23 PeopleFactory Class Reference

`#include <PeopleFactory.h>`

Inheritance diagram for PeopleFactory:



Public Member Functions

- PeopleFactory ()
- virtual ~PeopleFactory ()
- virtual People * createSoldier ()=0
- virtual People * createMedic ()=0
- virtual People * createCitizen ()=0

5.23.1 Detailed Description

Author

Ethan

Date

17 October 2022

The abstract class for creations of People classes

5.23.2 Constructor & Destructor Documentation

5.23.2.1 PeopleFactory()

`PeopleFactory::PeopleFactory ( )`

5.23.2.2 ~PeopleFactory()

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

5.23.3 Member Function Documentation

5.23.3.1 createCitizen()

```
virtual People * PeopleFactory::createCitizen ( )  [pure virtual]
```

Implemented in AirPeopleFactory, LandPeopleFactory, and WaterPeopleFactory.

5.23.3.2 createMedic()

```
virtual People * PeopleFactory::createMedic ( )  [pure virtual]
```

Implemented in AirPeopleFactory, LandPeopleFactory, and WaterPeopleFactory.

5.23.3.3 createSoldier()

```
virtual People * PeopleFactory::createSoldier ( )  [pure virtual]
```

Implemented in AirPeopleFactory, LandPeopleFactory, and WaterPeopleFactory.

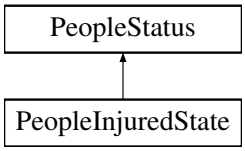
The documentation for this class was generated from the following files:

- PeopleFactory.h
- PeopleFactory.cpp

5.24 PeopleInjuredState Class Reference

```
#include <PeopleInjuredState.h>
```

Inheritance diagram for PeopleInjuredState:



Public Member Functions

- PeopleInjuredState ()
- int handle ()

5.24.1 Detailed Description

Author

Ethan

Date

17 October 2022

See also

People

Represents the state of being injured of a People object

5.24.2 Constructor & Destructor Documentation

5.24.2.1 PeopleInjuredState()

```
PeopleInjuredState::PeopleInjuredState ( )
```

5.24.3 Member Function Documentation

5.24.3.1 handle()

```
int PeopleInjuredState::handle ( )  [virtual]
```

The damage a People object can do while being injured

Returns

1

Implements PeopleStatus.

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

- PeopleInjuredState.h
- PeopleInjuredState.cpp

5.25 PeopleIterator Class Reference

`#include <PeopleIterator.h>`

Public Member Functions

- PeopleIterator (std::vector< People * > vec)
- People & get ()
- People & at (int x)

Public Attributes

- People * end

Protected Attributes

- People * current
- std::vector< People * > v

Friends

- class People

5.25.1 Constructor & Destructor Documentation

5.25.1.1 PeopleIterator()

```
PeopleIterator::PeopleIterator (
            std::vector< People * > vec )
```

Initialisation for the current end and c members, set to the argument.

Parameters

vec	The vector containing the People objects

5.25.2 Member Function Documentation

5.25.2.1 at()

```
People & PeopleIterator::at (
            int x )
```

Obtain a People object at a certain index in the vector

Parameters

x	Index of object to return

Returns

People object at index x in the vector

5.25.2.2 get()

```
People & PeopleIterator::get ( )
```

Obtain the current People object

Returns

People object which current is pointing to

5.25.3 Friends And Related Function Documentation

5.25.3.1 People

```
friend class People  [friend]
```

5.25.4 Member Data Documentation

5.25.4.1 current

```
People* PeopleIterator::current  [protected]
```

5.25.4.2 end

```
People* PeopleIterator::end
```

5.25.4.3 v

```
std::vector<People*> PeopleIterator::v  [protected]
```

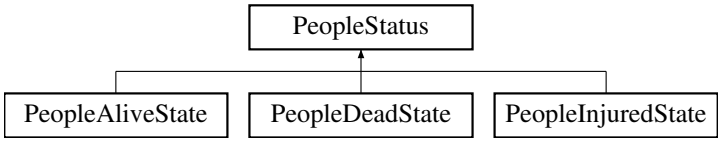
The documentation for this class was generated from the following files:

- PeopleIterator.h
- PeopleIterator.cpp

5.26 PeopleStatus Class Reference

```
#include <PeopleStatus.h>
```

Inheritance diagram for PeopleStatus:



Public Member Functions

- virtual ~PeopleStatus ()
- virtual int handle ()=0

5.26.1 Detailed Description

Author

Ethan

Date

17 October 2022

Represents the status of a People object - alive, dead or injured.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 ~PeopleStatus()

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

5.26.3 Member Function Documentation

5.26.3.1 handle()

```
virtual int PeopleStatus::handle ( )  [pure virtual]
```

Implemented in PeopleAliveState, PeopleDeadState, and PeopleInjuredState.

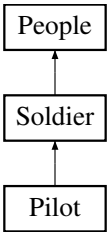
The documentation for this class was generated from the following files:

- PeopleStatus.h
- PeopleStatus.cpp

5.27 Pilot Class Reference

```
#include <Pilot.h>
```

Inheritance diagram for Pilot:



Public Member Functions

- Pilot ()
- ~Pilot ()
- int act ()

Additional Inherited Members

5.27.1 Detailed Description

Author

Ethan

Date

17 October 2022

Soldiers stationed for aerial battle

5.27.2 Constructor & Destructor Documentation

5.27.2.1 Pilot()

```
Pilot::Pilot ( )
```

Pilots start alive with default damage multiplier of 3

5.27.2.2 ~Pilot()

```
Pilot::~Pilot ( )
```

5.27.3 Member Function Documentation

5.27.3.1 act()

```
int Pilot::act ( )  [virtual]
```

Pilot deals damage - default multiplier times amount determined by alive, injured or dead

Returns

damage dealt

Reimplemented from Soldier.

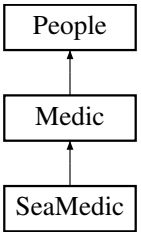
The documentation for this class was generated from the following files:

- Pilot.h
- Pilot.cpp

5.28 SeaMedic Class Reference

```
#include <SeaMedic.h>
```

Inheritance diagram for SeaMedic:



Public Member Functions

- SeaMedic ()

  SeaMedics start alive with default multiplier of 1.
- ~SeaMedic ()
- int act ()

Additional Inherited Members

5.28.1 Detailed Description

Author

Ethan

Date

17 October 2022

Specialisation of Medic - those stationed at sea

5.28.2 Constructor & Destructor Documentation

5.28.2.1 SeaMedic()

```
SeaMedic::SeaMedic ( )
```

SeaMedics start alive with default multiplier of 1.

5.28.2.2 ~SeaMedic()

```
SeaMedic::~SeaMedic ( )
```

5.28.3 Member Function Documentation

5.28.3.1 act()

```
int SeaMedic::act ( )  [virtual]
```

SeaMedic deals damage - multiplier times amount determined by alive, dead or injured

Returns

damage dealt

Reimplemented from Medic.

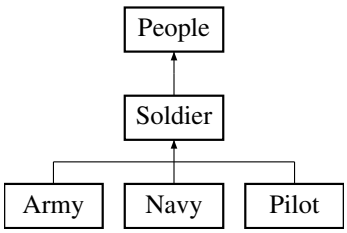
The documentation for this class was generated from the following files:

- SeaMedic.h
- SeaMedic.cpp

5.29 Soldier Class Reference

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

Inheritance diagram for Soldier:



Public Member Functions

- virtual ~Soldier ()
- virtual int act ()

Additional Inherited Members

5.29.1 Detailed Description

Author

Ethan

Date

17 October 2022

A specialisation of the People class - soldier represents soldier of a country in a war scene

5.29.2 Constructor & Destructor Documentation

5.29.2.1 ~Soldier()

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

5.29.3 Member Function Documentation

5.29.3.1 act()

```
int Soldier::act ( )  [virtual]
```

The Soldier object carries out their action - shoot/attack/defend etc.

Returns

-1 to indicate successful execution

Implements People.

Reimplemented in Army, Navy, and Pilot.

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

- Soldier.h
- Soldier.cpp

5.30 stateMem Class Reference

```
#include <stateMem.h>
```

Public Member Functions

- stateMem (TransportState *ts, vector< People * > parray)

  constructor of SateMem to initialize the TransportState object and vector with passed in parameters
- stateMem (const stateMem &s)

  constructor to initialize a new object that is passed through as parameter
- void showstate ()

  loops through the vector of People objects while outputting each person's state, the transportstate object of the country is then printed

5.30.1 Detailed Description

Author

Amicke

Date

2022/11/03

StateMem class will be used to clone the vector array of People objects then will be saved

5.30.2 Constructor & Destructor Documentation

5.30.2.1 stateMem() [1/2]

```
stateMem::stateMem (
            TransportState * ts,
            vector< People * > parray )  [inline]
```

constructor of SateMem to initialize the TransportState object and vector with passed in parameters

Parameters

ts	to initialize the TransportState object
parray	to initialize the vector array

5.30.2.2 stateMem() [2/2]

```
stateMem::stateMem (
            const stateMem & s )  [inline]
```

constructor to initialize a new object that is passed through as parameter

Parameters

s	to initialize the stateMem object with the trasnportstate and the vector of People objects

5.30.3 Member Function Documentation

5.30.3.1 showstate()

```
void stateMem::showstate ( )  [inline]
```

loops through the vector of People objects while outputting each person's state, the transportstate object of the country is then printed

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

- stateMem.h

5.31 Transport Class Reference

```
#include <Transport.h>
```

Public Member Functions

- virtual ~Transport ()
  deletion of the state object
- float request ()
  handles the request made of the state of transport to be broken or working
- void setStateWorking ()
  creates a new WorkingTransportState object
- void setStateBroken ()
  creates a new BrokenTransportState object

5.31.1 Detailed Description

Author

Franko Swanepoel

Date

2022/10/24

Transport.h represents the state of transport which can be broken or working

5.31.2 Constructor & Destructor Documentation

5.31.2.1 ~Transport()

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

deletion of the state object

5.31.3 Member Function Documentation

5.31.3.1 request()

```
float Transport::request ( )
```

handles the request made of the state of transport to be broken or working

Returns

Float used as a damage multiplier when calculating the amount of damage that a country can do

5.31.3.2 setStateBroken()

```
void Transport::setStateBroken ( )
```

creates a new BrokenTransportState object

See also

BrokenTransportState

5.31.3.3 setStateWorking()

```
void Transport::setStateWorking ( )
```

creates a new WorkingTransportState object

See also

WorkingTransportState

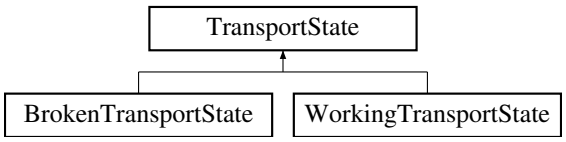
The documentation for this class was generated from the following files:

- Transport.h
- Transport.cpp

5.32 TransportState Class Reference

`#include <TransportState.h>`

Inheritance diagram for TransportState:



Public Member Functions

- virtual float handle ()=0

  pure virtual function for state of transport to be handled as either working or broken
- virtual ~TransportState ()=default

5.32.1 Detailed Description

Author

Franko Swanepoel

Date

2022/10/24

Abstract class to be used as base class for Transport.h's handle function

5.32.2 Constructor & Destructor Documentation

5.32.2.1 ~TransportState()

`virtual TransportState::~TransportState ( )  [virtual], [default]`

5.32.3 Member Function Documentation

5.32.3.1 handle()

```
virtual float TransportState::handle ( )  [pure virtual]
```

pure virtual function for state of transport to be handled as either working or broken

Returns

float of either 0.25 or 1

Implemented in BrokenTransportState, and WorkingTransportState.

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

- TransportState.h

5.33 WarEngine Class Reference

```
#include <WarEngine.h>
```

Public Member Functions

- void loop ()
  Loop method is used to run a phase of the WarEngine.
- bool alliesAlive ()
  Checks whether any allies are alive.
- bool enemiesAlive ()
  Checks whether any enemies are alive.

Static Public Member Functions

- static WarEngine & instance ()
  Method to return the Singleton instance of the WarEngine.

Protected Member Functions

- WarEngine ()
- ~WarEngine ()
- WarEngine (const WarEngine &)

Friends

- class WarPhaseEarly
- class WarPhaseMiddle
- class WarPhaseLate

5.33.1 Detailed Description

Author

Ethan

Date

31 October 2022

The WarEngine is the mastermind of the whole game. It is parameterised with a phase which then causes the engine to act accordingly in executing the game. The WarEngine has access to phases and countries.

5.33.2 Constructor & Destructor Documentation

5.33.2.1 WarEngine() [1/2]

```
WarEngine::WarEngine ( )  [protected]
```

War starts in the early phase

5.33.2.2 ~WarEngine()

```
WarEngine::~WarEngine ( )  [protected]
```

5.33.2.3 WarEngine() [2/2]

```
WarEngine::WarEngine (
            const WarEngine &  )  [protected]
```

5.33.3 Member Function Documentation

5.33.3.1 alliesAlive()

```
bool WarEngine::alliesAlive ( )
```

Checks whether any allies are alive.

Returns

Boolean - True if any allies are still alive

Checks whether each allied country's people are alive

Returns

alive - a true or false indicating whether all ally countries are alive

5.33.3.2 enemiesAlive()

```
bool WarEngine::enemiesAlive ( )
```

Checks whether any enemies are alive.

Returns

Boolean - True if any enemies are still alive

Checks whether each enemy country's people are alive

Returns

alive - true or false indicating whether all enemy countries are alive

5.33.3.3 instance()

```
WarEngine & WarEngine::instance ( )  [static]
```

Method to return the Singleton instance of the WarEngine.

Returns a reference to a static WarEngine object that acts as the Singleton

Returns

onlyInstance - a static reference to the WarEngine, ensuring that the Singleton is maintained

5.33.3.4 loop()

```
void WarEngine::loop ( )
```

Loop method is used to run a phase of the WarEngine.

The algorithm which acts as the execution of the war is called. Actions will depend on which phase the war is in.

5.33.4 Friends And Related Function Documentation

5.33.4.1 WarPhaseEarly

```
friend class WarPhaseEarly  [friend]
```

5.33.4.2 WarPhaseLate

```
friend class WarPhaseLate  [friend]
```

5.33.4.3 WarPhaseMiddle

```
friend class WarPhaseMiddle  [friend]
```

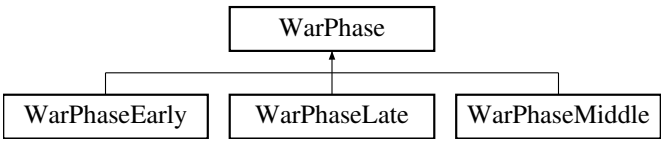
The documentation for this class was generated from the following files:

- WarEngine.h
- WarEngine.cpp

5.34 WarPhase Class Reference

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

Inheritance diagram for WarPhase:



Public Member Functions

- virtual ~WarPhase ()
- virtual void warAlgorithm (WarEngine &x)=0

Static Public Member Functions

- static int randomNum ()

  Returns a random number between 0 - 99.

5.34.1 Detailed Description

Author

Ethan

Date

31 October 2022

The interface/abstract class for the phase the war is in (early/middle/late)

See also

WarPhaseEarly

WarPhaseMiddle

WarPhaseLate

5.34.2 Constructor & Destructor Documentation

5.34.2.1 ~WarPhase()

```
WarPhase::~WarPhase ( )  [virtual], [default]
```

5.34.3 Member Function Documentation

5.34.3.1 randomNum()

```
int WarPhase::randomNum ( )  [static]
```

Returns a random number between 0 - 99.

Returns

Int

5.34.3.2 warAlgorithm()

```
void WarPhase::warAlgorithm (
            WarEngine & x )  [pure virtual]
```

Implemented in WarPhaseEarly, WarPhaseLate, and WarPhaseMiddle.

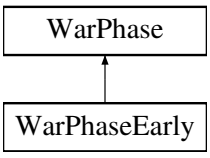
The documentation for this class was generated from the following files:

- WarPhase.h
- WarPhase.cpp

5.35 WarPhaseEarly Class Reference

```
#include <WarPhaseEarly.h>
```

Inheritance diagram for WarPhaseEarly:



Public Member Functions

- void warAlgorithm (WarEngine &x) override

Additional Inherited Members

5.35.1 Detailed Description

Author

Ethan

Date

31 October 2022

Represents the early state of the war - the setup before any actual battle

5.35.2 Member Function Documentation

5.35.2.1 warAlgorithm()

```
void WarPhaseEarly::warAlgorithm (
            WarEngine & x )  [override], [virtual]
```

Sets up the game by initialising all the member variables accordingly. In summary, this is done by asking for the war size and player country and then setting up the ally and enemy vectors (and countries), finally updating the phase of the WarEngine to the middle phase to progress the war game.

Parameters

x	The WarEngine object of the current game

< Vector from which participating countries can be chosen depending on war size

Implements WarPhase.

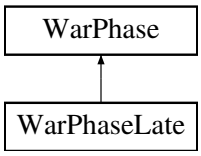
The documentation for this class was generated from the following files:

- WarPhaseEarly.h
- WarPhaseEarly.cpp

5.36 WarPhaseLate Class Reference

`#include <WarPhaseLate.h>`

Inheritance diagram for WarPhaseLate:



Public Member Functions

- void warAlgorithm (WarEngine &x) override

Additional Inherited Members

5.36.1 Detailed Description

Author

Ethan

Date

31 October 2022

Represents the late phase of the war - after all battles

5.36.2 Member Function Documentation

5.36.2.1 warAlgorithm()

```
void WarPhaseLate::warAlgorithm (
            WarEngine & x )  [override], [virtual]
```

Determines the victor of the war. This is done by checking whether allies or enemies are alive and providing output on the defeated.

Parameters

x	The current war game's WarEngine object

Implements WarPhase.

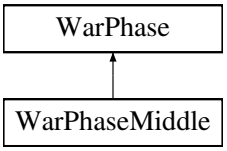
The documentation for this class was generated from the following files:

- WarPhaseLate.h
- WarPhaseLate.cpp

5.37 WarPhaseMiddle Class Reference

`#include <WarPhaseMiddle.h>`

Inheritance diagram for WarPhaseMiddle:



Public Member Functions

- void warAlgorithm (WarEngine &x) override

Static Public Member Functions

- static void tryRepair (Country &c)
- static void printStats (const vector< Country * > &v, const vector< Country * > &v2)

5.37.1 Detailed Description

Author

Ethan

Date

31 October 2022

Represents the middle phase of the war - the action and battle

5.37.2 Member Function Documentation

5.37.2.1 printStats()

```
void WarPhaseMiddle::printStats (
            const vector< Country * > & v,
            const vector< Country * > & v2 )  [static]
```

5.37.2.2 tryRepair()

```
void WarPhaseMiddle::tryRepair (
            Country & c )  [static]
```

Attempt to repair a country's transport lines - by a random 40% chance there will be a successful repair

Parameters

c	The country which transport lines are attempted to be repaired

5.37.2.3 warAlgorithm()

```
void WarPhaseMiddle::warAlgorithm (
            WarEngine & x )  [override], [virtual]
```

The actual battle during the war in the game. In summary, the player's enemies and allies are given. Next, the player is repeatedly given a choice in the move they wish to make - attack, repair, heal or undo.

Attack - the player chooses a country to attack out of the enemy countries. If a random chance succeeds, the player also breaks the country's transport lines.

Repair - attempt to repair the player's own transport lines

Heal - heal player's own troops

Undo - undo previous move

The AI or computer then also makes its move based on random chance, similar to the player's choices

As soon as anyone is defeated (people dead), the WarEngine is pushed to the late phase.

Parameters

x	the current game's WarEngine object

Implements WarPhase.

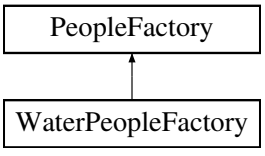
The documentation for this class was generated from the following files:

- WarPhaseMiddle.h
- WarPhaseMiddle.cpp

5.38 WaterPeopleFactory Class Reference

```
#include <WaterPeopleFactory.h>
```

Inheritance diagram for WaterPeopleFactory:



Public Member Functions

- WaterPeopleFactory ()
- ~WaterPeopleFactory ()
- People * createSoldier ()
- People * createMedic ()
- People * createCitizen ()

5.38.1 Detailed Description

Author

Ethan

Date

17 October 2022

A specialisation of the PeopleFactory class, the WaterPeopleFactory class creates Navy and SeaMedic objects

See also

SeaMedic

Navy

5.38.2 Constructor & Destructor Documentation

5.38.2.1 WaterPeopleFactory()

```
WaterPeopleFactory::WaterPeopleFactory ( )
```

5.38.2.2 ~WaterPeopleFactory()

```
WaterPeopleFactory::~WaterPeopleFactory ( )
```

5.38.3 Member Function Documentation

5.38.3.1 createCitizen()

```
People * WaterPeopleFactory::createCitizen ( )  [virtual]
```

Implements PeopleFactory.

5.38.3.2 createMedic()

```
People * WaterPeopleFactory::createMedic ( )  [virtual]
```

Implements PeopleFactory.

5.38.3.3 createSoldier()

```
People * WaterPeopleFactory::createSoldier ( )  [virtual]
```

Implements PeopleFactory.

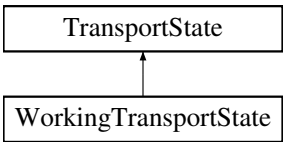
The documentation for this class was generated from the following files:

- WaterPeopleFactory.h
- WaterPeopleFactory.cpp

5.39 WorkingTransportState Class Reference

```
#include <WorkingTransportState.h>
```

Inheritance diagram for WorkingTransportState:



Public Member Functions

- float handle ()

  handles state of transport to be working when called

5.39.1 Detailed Description

Author

Franko Swanepoel

Date

2022/10/24

A specialization of the TransportState class - transport is in the working state

5.39.2 Member Function Documentation

5.39.2.1 handle()

```
float WorkingTransportState::handle ( )  [virtual]
```

handles state of transport to be working when called

Returns

int of 1

Implements TransportState.

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

- WorkingTransportState.h
- WorkingTransportState.cpp

Chapter 6

File Documentation

6.1 AirMedic.cpp File Reference

```
#include "AirMedic.h"
```

6.2 AirMedic.h File Reference

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

Classes

- class AirMedic

6.3 AirMedic.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef AIR_MEDIC_H
6 #define AIR_MEDIC_H
7 #include "Medic.h"
8
14 class AirMedic :  public Medic{
15 public:
16     AirMedic();
17     ~AirMedic() override;
18     int act();
19 };
20
21 #endif //AIR_MEDIC_H
```

6.4 AirPeopleFactory.cpp File Reference

```
#include "AirPeopleFactory.h"
```

6.5 AirPeopleFactory.h File Reference

```
#include "PeopleFactory.h"
#include "Pilot.h"
#include "AirMedic.h"
#include "LandCitizen.h"
```

Classes

- class AirPeopleFactory

6.6 AirPeopleFactory.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef AIR_PEOPLE_FACTORY_H
6 #define AIR_PEOPLE_FACTORY_H
7 #include "PeopleFactory.h"
8 #include "Pilot.h"
9 #include "AirMedic.h"
10 #include "LandCitizen.h"
11
21 class AirPeopleFactory :  public PeopleFactory{
22 public:
23     AirPeopleFactory();
24     ~AirPeopleFactory();
25
26     People* createSoldier();
27     People* createMedic();
28     People* createCitizen();
29 };
30
31 #endif //AIR_PEOPLE_FACTORY_H
```

6.7 Army.cpp File Reference

```
#include "Army.h"
```

6.8 Army.h File Reference

```
#include "Soldier.h"
```

Classes

- class Army

6.9 Army.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef ARMY_H
6 #define ARMY_H
7 #include "Soldier.h"
8
14 class Army :  public Soldier{
15 public:
16     Army();
17     ~Army();
18     int act();
19 };
20
21 #endif //ARMY_H
```

6.10 BrokenTransportState.cpp File Reference

```
#include "BrokenTransportState.h"
```

6.11 BrokenTransportState.h File Reference

```
#include "TransportState.h"
```

Classes

- class BrokenTransportState

6.12 BrokenTransportState.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/10/24.
3 //
4
5 #ifndef COS214_PROJECT_BROKENTRANSPORTSTATE_H
6 #define COS214_PROJECT_BROKENTRANSPORTSTATE_H
7
8 #include "TransportState.h"
9
16 class BrokenTransportState :  public TransportState{
17
18 public:
23     float handle();
24
25 };
26
27
28 #endif //COS214_PROJECT_BROKENTRANSPORTSTATE_H
```

6.13 Citizen.cpp File Reference

```
#include "Citizen.h"
```

6.14 Citizen.h File Reference

```
#include "People.h"
```

Classes

- class Citizen

6.15 Citizen.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef CITIZEN_H
6 #define CITIZEN_H
7 #include "People.h"
8
13 class Citizen :  public People{
14 public:
15     virtual ~Citizen();
16     virtual int act();
17 };
18
19 #endif //CITIZEN_H
```

6.16 ConcreteCountry.cpp File Reference

```
#include "ConcreteCountry.h"
```

6.17 ConcreteCountry.h File Reference

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

Classes

- class ConcreteCountry

6.18 ConcreteCountry.h

Go to the documentation of this file.

```
1 //
2 // Created by mattg on 2022/10/31.
3 //
4
5 #ifndef CONCRETECOUNTRY_H
6 #define CONCRETECOUNTRY_H
7
8 #include "Country.h"
9
10 class ConcreteCountry :  public Country {
11 public:
13     ConcreteCountry();
16     ConcreteCountry(string name);
17     virtual ~ConcreteCountry();
19     int attack() override;
22     void defend(int dmg) override;
24     void heal() override;
27     Country* clone();
28 };
29
30 #endif
```

6.19 ConcreteCountryFactory.cpp File Reference

```
#include "ConcreteCountryFactory.h"
```

6.20 ConcreteCountryFactory.h File Reference

```
#include "CountryFactory.h"
```

Classes

- class ConcreteCountryFactory

6.21 ConcreteCountryFactory.h

Go to the documentation of this file.

```
1 // Created by Matt on 2022/10/17.
2
3 #ifndef CONCRETECOUNTRYFACTORY_H
4 #define CONCRETECOUNTRYFACTORY_H
5
6 #include "CountryFactory.h"
7
8 class ConcreteCountryFactory :  public CountryFactory {
9 public:
11     ConcreteCountryFactory();
12     ~ConcreteCountryFactory();
16     Country* produceCountry(string name) override;
17 };
18
19 #endif
```

6.22 ConcreteMediator.cpp File Reference

```
#include "ConcreteMediator.h"
```

6.23 ConcreteMediator.h File Reference

```
#include "Country.h"
#include "ConcreteCountry.h"
#include "Mediator.h"
```

Classes

- class ConcreteMediator

6.24 ConcreteMediator.h

Go to the documentation of this file.

```
1 //
2 // Created by mattg on 2022/10/31.
3 //
4
5 #ifndef CONCRETEMEDIATOR_H
6 #define CONCRETEMEDIATOR_H
7
8 #include "Country.h"
9 #include "ConcreteCountry.h"
10 #include "Mediator.h"
11
12
13 class ConcreteMediator :  public Mediator {
14 public:
15     ConcreteMediator();
16     ~ConcreteMediator();
19     virtual void notify(ConcreteCountry* country);
20 private:
21     Country* countryList;
22 };
23
24
25 #endif //CONCRETEMEDIATOR_H
```

6.25 Country.cpp File Reference

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

6.26 Country.h File Reference

```
#include <string>
#include <iostream>
#include <vector>
#include <random>
#include "People.h"
#include "Mediator.h"
#include "AirPeopleFactory.h"
#include "LandPeopleFactory.h"
#include "WaterPeopleFactory.h"
#include "Transport.h"
#include "PeopleIterator.h"
```

Classes

- class Country

6.27 Country.h

Go to the documentation of this file.

```
1 // Created by Matt on 2022/10/17.
2
3 #ifndef COUNTRY_H
4 #define COUNTRY_H
5
6 #include <string>
7 #include <iostream>
8 #include <vector>
9 #include <random>
10
11 #include "People.h"
12 #include "Mediator.h"
13 #include "AirPeopleFactory.h"
14 #include "LandPeopleFactory.h"
15 #include "WaterPeopleFactory.h"
16 #include "Transport.h"
17 #include "PeopleIterator.h"
18
19 using namespace std;
20
21 class Country {
22 private:
23     string name;
24     int numPeople;
25
26 protected:
27     int numAlive;
28     Transport* transport;
29     vector<People*> citizens;
30 public:
31     Country();
34     Country(string name);
35     virtual ~Country();
38     virtual int attack() = 0;
41     virtual void defend(int damage) = 0;
43     virtual void heal() = 0;
45     void randomPeople();
50     static int randomNumInRange(int min, int max);
53     string getName();
56     int getNumPeople() const;
59     int getNumAlive() const;
62     int getNumInjured() const;
65     void fixTransport();
68     void breakTransport();
71     float requestTransport();
74     bool isAlive() const;
77     virtual Country* clone() = 0;
78
79 };
80
81 #endif
```

6.28 CountryFactory.cpp File Reference

```
#include "CountryFactory.h"
```

6.29 CountryFactory.h File Reference

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

Classes

- class CountryFactory

6.30 CountryFactory.h

Go to the documentation of this file.

```
1 // Created by Matt on 2022/10/17.
2
3 #ifndef COUNTRYFACTORY_H
4 #define COUNTRYFACTORY_H
5
6 #include "Country.h"
7 #include "ConcreteCountry.h"
8
9 class CountryFactory {
10 public:
11
12     CountryFactory();
13     virtual ~CountryFactory();
17     virtual Country* produceCountry(string name) = 0;
18 };
19
20 #endif
```

6.31 Facade.cpp File Reference

```
#include "Facade.h"
```

6.32 Facade.h File Reference

```
#include "WarEngine.h"
```

Classes

- class Facade

6.33 Facade.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/11/03.
3 //
4
5 #ifndef CODE_FACADE_H
6 #define CODE_FACADE_H
7 #include "WarEngine.h"
8
16 class Facade {
17 public:
18     Facade();
19     ~Facade();
20     void gameStart();
21 };
22
23
24 #endif //CODE_FACADE_H
```

6.34 LandCitizen.cpp File Reference

```
#include "LandCitizen.h"
```

6.35 LandCitizen.h File Reference

```
#include "Citizen.h"
```

Classes

- class LandCitizen

6.36 LandCitizen.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef LAND_CITIZEN_H
6 #define LAND_CITIZEN_H
7 #include "Citizen.h"
8
14 class LandCitizen :  public Citizen{
15     ~LandCitizen();
16     int act();
17
18 public:
19     LandCitizen();
20 };
21
22 #endif //LAND_CITIZEN_H
```

6.37 LandMedic.cpp File Reference

```
#include "LandMedic.h"
```

6.38 LandMedic.h File Reference

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

Classes

- class LandMedic

6.39 LandMedic.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef LAND_MEDIC_H
6 #define LAND_MEDIC_H
7 #include "Medic.h"
8
14 class LandMedic :  public Medic{
15 public:
16     LandMedic();
17     ~LandMedic();
18     int act();
19 };
20
21
22 #endif //LAND_MEDIC_H
```

6.40 LandPeopleFactory.cpp File Reference

```
#include "LandPeopleFactory.h"
```

6.41 LandPeopleFactory.h File Reference

```
#include "PeopleFactory.h"
#include "LandCitizen.h"
#include "Army.h"
#include "LandMedic.h"
```

Classes

- class LandPeopleFactory

6.42 LandPeopleFactory.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef LAND_PEOPLE_FACTORY_H
6 #define LAND_PEOPLE_FACTORY_H
7 #include "PeopleFactory.h"
8 #include "LandCitizen.h"
9 #include "Army.h"
10 #include "LandMedic.h"
11
22 class LandPeopleFactory :  public PeopleFactory{
23 public:
24     LandPeopleFactory();
25     ~LandPeopleFactory();
26
27     People* createSoldier();
28     People* createMedic();
29     People* createCitizen();
30 };
31
32 #endif //LAND_PEOPLE_FACTORY_H
```

6.43 Mediator.cpp File Reference

```
#include "Mediator.h"
```

6.44 Mediator.h File Reference

Classes

- class Mediator

6.45 Mediator.h

Go to the documentation of this file.

```
1 //
2 // Created by mattg on 2022/10/31.
3 //
4
5 #ifndef MEDIATOR_H
6 #define MEDIATOR_H
7
8 class Mediator {
9 public:
10     Mediator();
11     virtual ~Mediator();
13     virtual void notify() = 0;
14 };
15
16 #endif //MEDIATOR_H
```

6.46 Medic.cpp File Reference

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

6.47 Medic.h File Reference

```
#include "People.h"
```

Classes

- class Medic

6.48 Medic.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef MEDIC_H
6 #define MEDIC_H
7 #include "People.h"
8
14 class Medic :  public People{
15 public:
16     virtual ~Medic();
17     virtual int act();
18 };
19
20 #endif //MEDIC_H
```

6.49 Memento.cpp File Reference

```
#include "Memento.h"
```

6.50 Memento.h File Reference

```
#include "WarEngine.h"
#include "stateMem.h"
#include "Country.h"
```

Classes

- class Memento

6.51 Memento.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/11/03.
3 //
4
5 #ifndef COS214_PROJECT_MEMENTO_H
6 #define COS214_PROJECT_MEMENTO_H
7
8 #include "WarEngine.h"
9 #include "stateMem.h"
10 #include "Country.h"
11
17 class Memento {
18 private:
22     TransportState* ts;
26     vector<People*> peparray;
27      friend class WarEngine;
31     Memento();
35     stateMem* state;
36 public:
37     virtual ~Memento();
38 };
39
40
41 #endif //COS214_PROJECT_MEMENTO_H
```

6.52 mementostorage.cpp File Reference

```
#include "mementostorage.h"
```

6.53 mementostorage.h File Reference

```
#include "Memento.h"
```

Classes

- class mementostorage

6.54 mementostorage.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/11/03.
3 //
4
5 #ifndef COS214_PROJECT_MEMENTOSTORAGE_H
6 #define COS214_PROJECT_MEMENTOSTORAGE_H
7
8 #include "Memento.h"
9
10
11 class mementostorage {
12 private:
16     Memento* themem;
21    void storememento(Memento* mem);
26    Memento* getmem();
30    ~mementostorage();
31
32
33 };
34
35
36 #endif //COS214_PROJECT_MEMENTOSTORAGE_H
```

6.55 Navy.cpp File Reference

```
#include "Navy.h"
```

6.56 Navy.h File Reference

```
#include "Soldier.h"
```

Classes

- class Navy

6.57 Navy.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef NAVY_H
6 #define NAVY_H
7 #include "Soldier.h"
8
14 class Navy :  public Soldier{
15 public:
16     Navy();
17     ~Navy();
18     int act();
19 };
20
21 #endif //NAVY_H
```

6.58 People.cpp File Reference

```
#include "People.h"
```

6.59 People.h File Reference

```
#include "PeopleStatus.h"
#include "PeopleAliveState.h"
#include "PeopleDeadState.h"
#include "PeopleInjuredState.h"
```

Classes

- class People

6.60 People.h

Go to the documentation of this file.

```
1 //
2 // Created by Ethan on 2022/10/17.
3 //
4
5 #ifndef PEOPLE_H
6 #define PEOPLE_H
7 #include "PeopleStatus.h"
8 #include "PeopleAliveState.h"
9 #include "PeopleDeadState.h"
10 #include "PeopleInjuredState.h"
11
17 class People{
18 public:
19     int dmg;
20     PeopleStatus* state;
21
22     People();
23     virtual ~People();
24
25     virtual int act() = 0;
26     void changeStateDead();
27     void changeStateInjured();
28     void changeStateAlive();
29 };
30
31 #endif //PEOPLE_H
```

6.61 PeopleAliveState.cpp File Reference

```
#include "PeopleAliveState.h"
```

6.62 PeopleAliveState.h File Reference

```
#include "PeopleStatus.h"
```

Classes

- class PeopleAliveState

6.63 PeopleAliveState.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef PEOPLE_ALIVE_STATE_H
6 #define PEOPLE_ALIVE_STATE_H
7 #include "PeopleStatus.h"
8
15 class PeopleAliveState :  public PeopleStatus{
16 public:
17     PeopleAliveState();
18     int handle();
19 };
20
21 #endif //PEOPLE_ALIVE_STATE_H
```

6.64 PeopleDeadState.cpp File Reference

```
#include "PeopleDeadState.h"
```

6.65 PeopleDeadState.h File Reference

```
#include "PeopleStatus.h"
```

Classes

- class PeopleDeadState

6.66 PeopleDeadState.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef PEOPLE_DEAD_STATE_H
6 #define PEOPLE_DEAD_STATE_H
7 #include "PeopleStatus.h"
8
15 class PeopleDeadState :  public PeopleStatus{
16 public:
17     PeopleDeadState();
18     int handle();
19 };
20
21 #endif //PEOPLE_DEAD_STATE_H
```

6.67 PeopleFactory.cpp File Reference

```
#include "PeopleFactory.h"
```

6.68 PeopleFactory.h File Reference

```
#include "People.h"
```

Classes

- class PeopleFactory

6.69 PeopleFactory.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef PEOPLE_FACTORY_H
6 #define PEOPLE_FACTORY_H
7
8 #include "People.h"
9
15 class PeopleFactory{
16 public:
17     PeopleFactory();
18     virtual ~PeopleFactory();
19
20     virtual People* createSoldier() = 0;
21     virtual People* createMedic() = 0;
22     virtual People* createCitizen() = 0;
23 };
24
25 #endif //PEOPLE_FACTORY_H
```

6.70 PeopleInjuredState.cpp File Reference

```
#include "PeopleInjuredState.h"
```

6.71 PeopleInjuredState.h File Reference

```
#include "PeopleStatus.h"
```

Classes

- class PeopleInjuredState

6.72 PeopleInjuredState.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef PEOPLE_INJURED_STATE_H
6 #define PEOPLE_INJURED_STATE_H
7 #include "PeopleStatus.h"
8
15 class PeopleInjuredState :  public PeopleStatus{
16 public:
17     PeopleInjuredState();
18     int handle();
19 };
20
21 #endif //PEOPLE_INJURED_STATE_H
```

6.73 PeopleIterator.cpp File Reference

```
#include "PeopleIterator.h"
#include "People.h"
```

6.74 PeopleIterator.h File Reference

```
#include <vector>
```

Classes

- class PeopleIterator

6.75 PeopleIterator.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/27.
3 //
4
5 #ifndef CODE_PEOPLE_ITERATOR_H
6 #define CODE_PEOPLE_ITERATOR_H
7 #include <vector>
8
15 class People;
16
17 class PeopleIterator {
18     friend class People;
19 public:
20     PeopleIterator(std::vector<People*> vec);
21     People* end;
22
23     People& get();
24     People& at(int x);
25
26 protected:
27     People* current;
28     std::vector<People*> v;
29 };
30
31
32 #endif //CODE_PEOPLE_ITERATOR_H
```

6.76 PeopleStatus.cpp File Reference

```
#include "PeopleStatus.h"
```

6.77 PeopleStatus.h File Reference

Classes

- class PeopleStatus

6.78 PeopleStatus.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef PEOPLE_STATUS_H
6 #define PEOPLE_STATUS_H
7
13 class PeopleStatus{ //Alive,Dead,Injured states
14 public:
15     virtual ~PeopleStatus();
16     virtual int handle() = 0;
17 };
18
19 #endif //PEOPLE_STATUS_H
```

6.79 Pilot.cpp File Reference

```
#include "Pilot.h"
```

6.80 Pilot.h File Reference

```
#include "Soldier.h"
```

Classes

- class Pilot

6.81 Pilot.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef PILOT_H
6 #define PILOT_H
7 #include "Soldier.h"
8
14 class Pilot :  public Soldier{
15 public:
16     Pilot();
17     ~Pilot();
18     int act();
19 };
20
21 #endif //PILOT_H
```

6.82 README.md File Reference

6.83 SeaMedic.cpp File Reference

```
#include "SeaMedic.h"
```

6.84 SeaMedic.h File Reference

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

Classes

- class SeaMedic

6.85 SeaMedic.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef SEA_MEDIC_H
6 #define SEA_MEDIC_H
7 #include "Medic.h"
8
14 class SeaMedic :  public Medic{
15 public:
16     SeaMedic();
17     ~SeaMedic();
18     int act();
19 };
20
21 #endif //SEA_MEDIC_H
```

6.86 Soldier.cpp File Reference

```
#include "Soldier.h"
```

6.87 Soldier.h File Reference

```
#include "People.h"
```

Classes

- class Soldier

6.88 Soldier.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef SOLDIER_H
6 #define SOLDIER_H
7 #include "People.h"
8
14 class Soldier :  public People{
15 public:
16     virtual ~Soldier();
17     virtual int act();
18 };
19
20 #endif //SOLDIER_H
```

6.89 stateMem.h File Reference

Classes

- class stateMem

6.90 stateMem.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/11/03.
3 //
4
5 #ifndef COS214_PROJECT_STATEMEM_H
6 #define COS214_PROJECT_STATEMEM_H
7
15 class stateMem {
16 private:
20     TransportState *transportstate;
21   //  People *peoplearray;
22    // Citizen saveCit;
26     vector<People*> saveCit;
27
28
29 public:
30
36     stateMem(TransportState *ts, vector<People*> parray) {
37         transportstate = ts;
38         saveCit = parray;
39
40     }
41
46     stateMem(const stateMem &s) {
47         saveCit = s.saveCit;
48         transportstate = s.transportstate;
49     }
50
54     void showstate()
55     {
56         cout<<"States:  "<<endl;
57         for (int i = 0; i < saveCit.size(); ++i) {
58             cout<<saveCit[i]->state<<endl;
59         }
60         cout<<"Transportstate of country is :  "<<transportstate;
61
62     }
63 };
64
65
66
67 #endif //COS214_PROJECT_STATEMEM_H
```

6.91 Transport.cpp File Reference

```
#include "Transport.h"
```

6.92 Transport.h File Reference

```
#include "TransportState.h"
#include "WorkingTransportState.h"
#include "BrokenTransportState.h"
```

Classes

- class Transport

6.93 Transport.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/10/24.
3 //
4
5 #ifndef COS214_PROJECT_TRANSPORT_H
6 #define COS214_PROJECT_TRANSPORT_H
7
8 #include "TransportState.h"
9 #include "WorkingTransportState.h"
10 #include "BrokenTransportState.h"
11
18 class Transport {
19 private:
23     TransportState* state;
24 public:
28     virtual ~Transport();
33     float request();
38     void setStateWorking();
43     void setStateBroken();
44 };
45
46
47 #endif //COS214_PROJECT_TRANSPORT_H
```

6.94 TransportState.h File Reference

Classes

- class TransportState

6.95 TransportState.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/10/24.
3 //
4
5 #ifndef COS214_PROJECT_TRANSPORTSTATE_H
6 #define COS214_PROJECT_TRANSPORTSTATE_H
7
14 class TransportState {
15 public:
20     virtual float handle() =0;
21     virtual ~TransportState() = default;
22 };
23
24
25 #endif //COS214_PROJECT_TRANSPORTSTATE_H
```

6.96 WarEngine.cpp File Reference

```
#include "WarEngine.h"
#include "WarPhase.h"
#include "WarPhaseEarly.h"
#include "WarPhaseMiddle.h"
#include "WarPhaseLate.h"
```

6.97 WarEngine.h File Reference

```
#include "ConcreteCountry.h"
#include <vector>
#include "CountryFactory.h"
#include "ConcreteCountryFactory.h"
```

Classes

- class WarEngine

6.98 WarEngine.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/31.
3 //
4
5 #ifndef CODE_WAR_ENGINE_H
6 #define CODE_WAR_ENGINE_H
7 #include "ConcreteCountry.h"
8 //#include "WarPhase.h"
9 //#include "WarPhaseEarly.h"
10 #include <vector>
11 #include "CountryFactory.h"
12 #include "ConcreteCountryFactory.h"
13
14 class WarPhase;
15 class WarPhaseEarly;
16 class WarPhaseMiddle;
17 class WarPhaseLate;
18
25 class WarEngine {
26 private:
27     WarPhase* phase;
28     std::vector<Country*> countries;
29     std::vector<Country*> allies;
30     std::vector<Country*> enemies;
31     Country* player;
32     CountryFactory* factory;
33
34 protected:
35     WarEngine();
36     ~WarEngine();
37     WarEngine(const WarEngine&);
38
39 public:
41     void loop();
44     bool alliesAlive();
47     bool enemiesAlive();
49     static WarEngine& instance();
50
51     friend class WarPhaseEarly;
52     friend class WarPhaseMiddle;
53     friend class WarPhaseLate;
54 };
55
56 #endif //CODE_WAR_ENGINE_H
```

6.99 WarPhase.cpp File Reference

```
#include "WarPhase.h"
```

6.100 WarPhase.h File Reference

```
#include "WarEngine.h"
#include <random>
#include <iostream>
#include <chrono>
```

Classes

- class WarPhase

6.101 WarPhase.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/31.
3 //
4
5 #ifndef CODE_WAR_PHASE_H
6 #define CODE_WAR_PHASE_H
7 #include "WarEngine.h"
8 #include <random>
9 #include <iostream>
10 #include <chrono>
11
21 class WarPhase{
22 public:
23     virtual ~WarPhase();
24     virtual void warAlgorithm(WarEngine& x) = 0;
27     static int randomNum();
28 };
29
30 #endif //CODE_WAR_PHASE_H
```

6.102 WarPhaseEarly.cpp File Reference

```
#include "WarPhaseEarly.h"
#include <iostream>
#include <string>
```

6.103 WarPhaseEarly.h File Reference

```
#include "WarPhase.h"
#include "WarPhaseMiddle.h"
```

Classes

- class WarPhaseEarly

6.104 WarPhaseEarly.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/31.
3 //
4
5 #ifndef CODE_WAR_PHASE_EARLY_H
6 #define CODE_WAR_PHASE_EARLY_H
7 #include "WarPhase.h"
8 #include "WarPhaseMiddle.h"
9
10 class WarEngine;
11
17 class WarPhaseEarly :  public WarPhase {
18 public:
19     void warAlgorithm(WarEngine& x) override;
20 };
21
22
23 #endif //CODE_WAR_PHASE_EARLY_H
```

6.105 WarPhaseLate.cpp File Reference

```
#include "WarPhaseLate.h"
```

6.106 WarPhaseLate.h File Reference

```
#include "WarPhase.h"
```

Classes

- class WarPhaseLate

6.107 WarPhaseLate.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/31.
3 //
4
5 #ifndef CODE_WAR_PHASE_LATE_H
6 #define CODE_WAR_PHASE_LATE_H
7 #include "WarPhase.h"
8
14 class WarPhaseLate :  public WarPhase {
15 public:
16     void warAlgorithm(WarEngine& x) override;
17 };
18
19
20 #endif //CODE_WAR_PHASE_LATE_H
```

6.108 WarPhaseMiddle.cpp File Reference

```
#include "WarPhaseMiddle.h"
```

6.109 WarPhaseMiddle.h File Reference

```
#include "WarPhase.h"
#include "WarPhaseLate.h"
#include <cstring>
```

Classes

- class WarPhaseMiddle

6.110 WarPhaseMiddle.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/31.
3 //
4
5 #ifndef CODE_WAR_PHASE_MIDDLE_H
6 #define CODE_WAR_PHASE_MIDDLE_H
7 #include "WarPhase.h"
8 #include "WarPhaseLate.h"
9 #include <cstring>
10 class WarEngine;
11
17 class WarPhaseMiddle :  public WarPhase {
18 public:
19     void warAlgorithm(WarEngine& x) override;
20     static void tryRepair(Country& c);
21     static void printStats(const vector<Country*>& v, const vector<Country*>& v2);
22 };
23 #endif //CODE_WAR_PHASE_MIDDLE_H
```

6.111 WaterPeopleFactory.cpp File Reference

```
#include "WaterPeopleFactory.h"
```

6.112 WaterPeopleFactory.h File Reference

```
#include "PeopleFactory.h"
#include "Navy.h"
#include "SeaMedic.h"
#include "LandCitizen.h"
```

Classes

- class WaterPeopleFactory

6.113 WaterPeopleFactory.h

Go to the documentation of this file.

```
1 //
2 // Created by ethan on 2022/10/17.
3 //
4
5 #ifndef WATER_PEOPLE_FACTORY_H
6 #define WATER_PEOPLE_FACTORY_H
7 #include "PeopleFactory.h"
8 #include "Navy.h"
9 #include "SeaMedic.h"
10 #include "LandCitizen.h"
11
21 class WaterPeopleFactory :  public PeopleFactory{
22 public:
23     WaterPeopleFactory();
24     ~WaterPeopleFactory();
25
26     People* createSoldier();
27     People* createMedic();
28     People* createCitizen();
29 };
30
31 #endif //WATER_PEOPLE_FACTORY_H
```

6.114 WorkingTransportState.cpp File Reference

```
#include "WorkingTransportState.h"
```

6.115 WorkingTransportState.h File Reference

```
#include "TransportState.h"
```

Classes

- class WorkingTransportState

6.116 WorkingTransportState.h

Go to the documentation of this file.

```
1 //
2 // Created by Franko Swanepoel on 2022/10/24.
3 //
4
5 #ifndef COS214_PROJECT_CONCRETETRANSPORTSTATE_H
6 #define COS214_PROJECT_CONCRETETRANSPORTSTATE_H
7
8 #include "TransportState.h"
9
16 class WorkingTransportState :  public TransportState{
17
18 public:
23     float handle();
24 };
25
26
27 #endif //COS214_PROJECT_CONCRETETRANSPORTSTATE_H
```

Index