COS214Project2021(SpaceXLaunchSimulator)

Generated by Doxygen 1.8.17

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 BuildCommand Class Reference

The Concrete Command for the Command Design Pattern and the Conrete Handler for the Chain of Responsibility.

`#include <BuildCommand.h>`

Inheritance diagram for BuildCommand:

Collaboration diagram for BuildCommand:

Public Member Functions

- BuildCommand (BuildSimulation *s)
  The constructor for the class.
- ~BuildCommand ()
  The destructor for the class.
- virtual void execute (std::string s, std::vector< State * > *v)
  The implementation of the virtual execute method.

4.1.1 Detailed Description

The Concrete Command for the Command Design Pattern and the Conrete Handler for the Chain of Responsibility.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BuildCommand()

```
BuildCommand::BuildCommand (
            BuildSimulation * s )
```

The constructor for the class.

Parameters

in	*s*	The simulation to be used in the command.

4.1.3 Member Function Documentation

4.1.3.1 execute()

```
void BuildCommand::execute (
            std::string s,
            std::vector< State * > * v )  [virtual]
```

The implementation of the virtual execute method.

Parameters

in	*s*	String to determine which simulation to run based off the chain of responsibility.
in	*v*	A pointer to the vector of states used to run the simulations.

Returns

void

Reimplemented from Command.

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

- include/BuildCommand.h

4.2 BuildSimulation Class Reference

The implementation of the building process.

```
#include <BuildSimulation.h>
```

Inheritance diagram for BuildSimulation:

Collaboration diagram for BuildSimulation:

Public Member Functions

- BuildSimulation ()

  The constructor for the class.
- ~BuildSimulation ()

  The destructor for the class.
- virtual void startSim (std::vector< State * > *v) override

  Starts the build simulation.
- void buildMode ()

  Build a simulation.
- void buildTestMode ()

  Build a simulation in test mode.
- void exitProgram ()

  Starts exit procedure.
- void saveToFile (State *s, int t)

  Used to save data to a file.
- void buildSattelites ()

  Build satellites for simulation.
- void buildCrew ()

  Build crew members for simulation.
- void buildCargo ()

  Build cargo for simulation.
- void test9 ()

  Perform test procedure for a falcon 9 rocket.
- void testHeavy ()

  Perform test procedure for a falcon heavy rocket.
- void testCargo ()

  Perform test procedure for cargo.
- void testCrewAndSatellite ()

  Perform test procedure for Crew members and Satellites.
- void testCrew ()

  Perform test procedure for crew members on dragon rocket.
- void testSatellites ()

  Perform test procedure for Satellites.

4.2.1 Detailed Description

The implementation of the building process.

4.2.2 Member Function Documentation

4.2.2.1 buildCargo()

```
BuildSimulation::buildCargo ( )
```

Build cargo for simulation.

Returns

void

4.2.2.2 buildCrew()

```
BuildSimulation::buildCrew ( )
```

Build crew members for simulation.

Returns

void

4.2.2.3 buildMode()

```
BuildSimulation::buildMode ( )
```

Build a simulation.

Returns

void

4.2.2.4 buildSattelites()

```
BuildSimulation::buildSattelites ( )
```

Build satellites for simulation.

Returns

void

4.2.2.5 buildTestMode()

```
BuildSimulation::buildTestMode ( )
```

Build a simulation in test mode.

Returns

void

4.2.2.6 exitProgram()

```
BuildSimulation::exitProgram ( )
```

Starts exit procedure.

Returns

void

4.2.2.7 saveToFile()

```
BuildSimulation::saveToFile (
            State * s,
            int t )
```

Used to save data to a file.

Parameters

in	*s*	The state to save to the file.
in	*t*	The id used to determine the type of payload.

Returns

void

4.2.2.8 startSim()

```
BuildSimulation::startSim (
            std::vector< State * > * v )  [override], [virtual]
```

Starts the build simulation.

Parameters

in	*v*	A pointer to the vector of states used to run the simulations.

Returns

void

Implements Simulation.

4.2.2.9 test9()

```
BuildSimulation::test9 ( )
```

Perform test procedure for a falcon 9 rocket.

Returns

void

4.2.2.10 testCargo()

```
BuildSimulation::testCargo ( )
```

Perform test procedure for cargo.

Returns

void

4.2.2.11 testCrew()

```
BuildSimulation::testCrew ( )
```

Perform test procedure for crew members on dragon rocket.

Returns

void

4.2.2.12 testCrewAndSatellite()

```
BuildSimulation::testCrewAndSatellite ( )
```

Perform test procedure for Crew members and Satellites.

Returns

void

4.2.2.13 testHeavy()

```
BuildSimulation::testHeavy ( )
```

Perform test procedure for a falcon heavy rocket.

Returns

void

4.2.2.14 testSatellites()

```
BuildSimulation::testSatellites ( )
```

Perform test procedure for Satellites.

Returns

void

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

- include/BuildSimulation.h

4.3 Cargo Class Reference

"A class for storing info about the Cargo that spacecraft will transport"

```
#include <Cargo.h>
```

Inheritance diagram for Cargo:

Collaboration diagram for Cargo:

Public Member Functions

- Cargo (string type, double weight)

  The constructor of the class.
- string getType ()

  Getter for the Cargo type attribute.
- double getWeight ()

  Getter for the Cargo weight attribute.
- void setWeight (double d)

  Setter for the Cargo weight attribute.
- void setType (string s)

  Setter for the Cargo type attribute.
- string toString () override

  Returns a summary of the Cargo attributes.

4.3.1 Detailed Description

"A class for storing info about the Cargo that spacecraft will transport"

4.3.2 Member Function Documentation

4.3.2.1 getType()

```
Cargo::getType ( )
```

Getter for the Cargo type attribute.

Returns

string

4.3.2.2 getWeight()

```
Cargo::getWeight ( )
```

Getter for the Cargo weight attribute.

Returns

double

4.3.2.3 setType()

```
Cargo::setType (
            string s )
```

Setter for the Cargo type attribute.

Parameters

in	*s*	The type of cargo.

Returns

void

4.3.2.4 setWeight()

```
Cargo::setWeight (
            double d )
```

Setter for the Cargo weight attribute.

Parameters

in	*d*	The wight of the cargo.

Returns

void

4.3.2.5 toString()

```
Cargo::toString ( )  [override], [virtual]
```

Returns a summary of the Cargo attributes.

Returns

string

Reimplemented from CarryType.

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

- include/Cargo.h

4.4 CargoDragon Class Reference

"Dragon specialisation to carry cargo only"

```
#include <CargoDragon.h>
```

Inheritance diagram for CargoDragon:

Collaboration diagram for CargoDragon:

Public Member Functions

- CargoDragon (Falcon *)

  The constructor for the class.
- ~CargoDragon ()

  The destructor for the class.
- void load (bool) override

  Template method for loading Dragon content.
- void unload (bool) override

  Method for unloading Dragon content.

4.4.1 Detailed Description

"Dragon specialisation to carry cargo only"

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CargoDragon()

```
CargoDragon::CargoDragon (
            Falcon *  )
```

The constructor for the class.

Parameters

in	*Falcon**	Falcon carrying that will carry this dragon spacecraft

4.4.3 Member Function Documentation

4.4.3.1 load()

```
CargoDragon::load (
            bool  )  [override], [virtual]
```

Template method for loading Dragon content.

Parameters

in	*bool*	Indicate whether to print loading data

Returns

void

Implements Dragon.

4.4.3.2 unload()

```
CargoDragon::unload (
            bool  )  [override], [virtual]
```

Method for unloading Dragon content.

Parameters

in	*bool*	Indicate whether to print loading data

Returns

void

Implements Dragon.

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

- include/CargoDragon.h

4.5 CargoDragonCreator Class Reference

Inheritance diagram for CargoDragonCreator:

Collaboration diagram for CargoDragonCreator:

Public Member Functions

- Spacecraft * createSpacecraft () override

  creates a new CargoDragonCreator

4.5.1 Member Function Documentation

4.5.1.1 createSpacecraft()

```
CargoDragonCreator::createSpacecraft ( )  [override], [virtual]
```

creates a new CargoDragonCreator

Returns

Spacecraft

Implements SpacecraftCreator.

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

- include/CargoDragonCreator.h

4.6 CargoIterator Class Reference

"The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"

`#include <CargoIterator.h>`

Inheritance diagram for CargoIterator:

Collaboration diagram for CargoIterator:

Public Member Functions

- ~CargoIterator ()
  The destructor of the class.
- CargoIterator (vector< Cargo * > &list)
  The accepted constructor of the class.
- CarryType * first () override
  Gets the first element in the list/collection.
- CarryType * next () override
  Gets the next element in the list/collection.
- CarryType * current () override
  Gets the current element in the list/collection.
- bool isDone () override
  Determines if there are more elements.

4.6.1 Detailed Description

"The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"

4.6.2 Constructor & Destructor Documentation

4.6.2.1 CargoIterator()

```
CargoIterator::CargoIterator (
            vector< Cargo * > & list )
```

The accepted constructor of the class.

Parameters

in	*list*	The vector of cargo that will be used.

4.6.3 Member Function Documentation

4.6.3.1 current()

```
CargoIterator::current ( )  [override], [virtual]
```

Gets the current element in the list/collection.

Returns

CarryType*

Reimplemented from Iterator.

4.6.3.2 first()

```
CargoIterator::first ( )  [override], [virtual]
```

Gets the first element in the list/collection.

Returns

CarryType*

Reimplemented from Iterator.

4.6.3.3 isDone()

```
CargoIterator::isDone ( )  [override], [virtual]
```

Determines if there are more elements.

Returns

Boolean: true if the end of the list/collection is reached

Reimplemented from Iterator.

4.6.3.4 next()

```
CargoIterator::next ( )  [override], [virtual]
```

Gets the next element in the list/collection.

Returns

CarryType*

Reimplemented from Iterator.

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

- include/CargoIterator.h

4.7 CarryType Class Reference

"A base class for Crew and Cargo"

```
#include <CarryType.h>
```

Inheritance diagram for CarryType:

Public Member Functions

- virtual ~CarryType ()
  Destructor for this class.
- virtual string toString ()
  Returns a summary of the Carry type object.

4.7.1 Detailed Description

"A base class for Crew and Cargo"

4.7.2 Member Function Documentation

4.7.2.1 toString()

```
CarryType::toString ( )  [virtual]
```

Returns a summary of the Carry type object.

Returns

string

Reimplemented in Cargo, and Crew.

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

- include/CarryType.h

4.8 CheckEngineCommand Class Reference

Concrete Command for checking the engines.

`#include <CheckEngineCommand.h>`

Inheritance diagram for CheckEngineCommand:

Collaboration diagram for CheckEngineCommand:

Public Member Functions

- CheckEngineCommand (std::vector< MerlinCore * > v, MerlinVacuumEngine *mv)
  The constructor for the class.
- ~CheckEngineCommand ()
  The destructor for the class.
- virtual void execute ()
  The implementation of the virtual execute method.

4.8.1 Detailed Description

Concrete Command for checking the engines.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 CheckEngineCommand()

```
CheckEngineCommand::CheckEngineCommand (
            std::vector< MerlinCore * > v,
            MerlinVacuumEngine * mv )
```

The constructor for the class.

Parameters

in	*v*	The vector of Merlin Cores to be used.
in	*mv*	The Merlin Vacuum Engine to be used.

4.8.3 Member Function Documentation

4.8.3.1 execute()

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

The implementation of the virtual execute method.

Returns

void

Reimplemented from Command.

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

- include/CheckEngineCommand.h

4.9 Cluster Class Reference

"Represents a cluster of satellite"

```
#include <Cluster.h>
```

Public Member Functions

- Cluster (Falcon *)
  The constructor for the class.
- ~Cluster ()
  The destructor for the class.
- void addSatellite (Satelite *)
  Add a satellite to a cluster.
- void generateSatellites (MissionControl *, int)
  Generate a variable amount of satellites and populate cluster.
- void spreadOutSatellites ()
  Spread out the satellites.
- void checkCollisions ()
  Check the whole cluster for any collisions.
- Falcon * getCraft ()
  Return the registered spacecraft.
- int getSize ()
  Return the size of cluster.

4.9.1 Detailed Description

"Represents a cluster of satellite"

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Cluster()

```
Cluster::Cluster (
            Falcon *  )
```

The constructor for the class.

Parameters

in	*Falcon*∗	The falcon used to to determine where the cluster is loaded.

4.9.3 Member Function Documentation

4.9.3.1 addSatellite()

```
Cluster::addSatellite (
            Satelite *  )
```

Add a satellite to a cluster.

Parameters

in	*Satellite*∗	Pointer to satellite that will be added to cluster

Returns

void

4.9.3.2 checkCollisions()

```
Cluster::checkCollisions ( )
```

Check the whole cluster for any collisions.

Returns

void

4.9.3.3 generateSatellites()

```
Cluster::generateSatellites (
            MissionControl * ,
            int  )
```

Generate a variable amount of satellites and populate cluster.

Parameters

in	*MissionControl*∗	The mission control used to communicate with the satellites.
in	*int*	The number of satellites to be generated.

Returns

void

4.9.3.4 getCraft()

`Cluster::getCraft ( )`

Return the registered spacecraft.

Returns

Falcon∗

4.9.3.5 getSize()

`Cluster::getSize ( )`

Return the size of cluster.

Returns

int

4.9.3.6 spreadOutSatellites()

`Cluster::spreadOutSatellites ( )`

Spread out the satellites.

Returns

void

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

- include/Cluster.h

4.10 Collections Class Reference

"A base class for crew- and cargo-lists (Aggregate-Iterator)"

`#include <Collections.h>`

Inheritance diagram for Collections:

4.10.1 Detailed Description

"A base class for crew- and cargo-lists (Aggregate-Iterator)"

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

- include/Collections.h

4.11 Command Class Reference

The Command for the Command Design Pattern and the Handler for the Chain of Responsibility.

`#include <Command.h>`

Inheritance diagram for Command:

Public Member Functions

- Command ()
  The constructor for the class.
- virtual ∼Command ()
  The virtual destructor for the class.
- virtual void execute ()
  The virtual execute method.
- virtual void execute (std::string s, std::vector< State * > *v)
  The overloaded virtual execute method used specifically for the chain of resposibility.
- void add (Command *c)
  The virtual execute method.

4.11.1 Detailed Description

The Command for the Command Design Pattern and the Handler for the Chain of Responsibility.

4.11.2 Member Function Documentation

4.11.2.1 add()

```
void Command::add (
            Command * c )
```

The virtual execute method.

Parameters

in	*c*	A pointer to the command to be added to the back of chain.

4.11.2.2 execute()

```
void Command::execute (
            std::string s,
            std::vector< State * > * v )  [virtual]
```

The overloaded virtual execute method used specifically for the chain of resposibility.

Parameters

in	*s*	String to determine which simulation to run based off the chain of responsibility.
in	*v*	A pointer to the vector of states used to run the simulations.

Reimplemented in BuildCommand, and SelectCommand.

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

- include/Command.h

4.12 Crew Class Reference

"A class for storing info about the Crew members that will be boarding the spacecraft"

`#include <Crew.h>`

Inheritance diagram for Crew:

Collaboration diagram for Crew:

Public Member Functions

- Crew (string name, string jobTitle)
  The constructor of the class.
- ~Crew ()
  The destructor of the class.
- string getName ()
  Getter for the Crew member's name.
- string getJobTitle ()
  Getter for the Crew member's job title.
- string toString () override
  Returns a summary of the Crew member's information.

Public Attributes

- Crew * next

4.12.1 Detailed Description

"A class for storing info about the Crew members that will be boarding the spacecraft"

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Crew()

```
Crew::Crew (
            string name,
            string jobTitle )
```

The constructor of the class.

Parameters

in	*name*	The name of the crew member.
in	*jobTitle*	The job of the crew member.

4.12.3 Member Function Documentation

4.12.3.1 getJobTitle()

```
Crew::getJobTitle ( )
```

Getter for the Crew member's job title.

Returns

string

4.12.3.2 getName()

```
Crew::getName ( )
```

Getter for the Crew member's name.

Returns

string

4.12.3.3 toString()

```
Crew::toString ( )  [override], [virtual]
```

Returns a summary of the Crew member's information.

Returns

string

Reimplemented from CarryType.

4.12.4 Member Data Documentation

4.12.4.1 next

```
Crew* Crew::next
```

The next crew member in the list.

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

- include/Crew.h

4.13 CrewDragon Class Reference

"Specialisation for dragon to carry both cargo and crew"

```
#include <CrewDragon.h>
```

Inheritance diagram for CrewDragon:

Collaboration diagram for CrewDragon:

Public Member Functions

- CrewDragon (Falcon *)

  The constructor for the class.
- ~CrewDragon ()

  The destructor for the class.
- void load (bool) override

  Method for loading Dragon content.
- void unload (bool) override

  Method for unloading Dragon content.

4.13.1 Detailed Description

"Specialisation for dragon to carry both cargo and crew"

4.13.2 Constructor & Destructor Documentation

4.13.2.1 CrewDragon()

```
CrewDragon::CrewDragon (
            Falcon *  )
```

The constructor for the class.

Parameters

in	*Faloon**	Reference to Falcon carrying this Dragon

4.13.3 Member Function Documentation

4.13.3.1 load()

```
CrewDragon::load (
            bool  )  [override], [virtual]
```

Method for loading Dragon content.

Parameters

in	*bool*	Indicate whether to print loading data

Returns

void

Implements Dragon.

4.13.3.2 unload()

```
CrewDragon::unload (
            bool  )  [override], [virtual]
```

Method for unloading Dragon content.

Parameters

in	*bool*	Indicate whether to print loading data

Returns

void

Implements Dragon.

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

- include/CrewDragon.h

4.14 CrewDragonCreator Class Reference

"Creates a crew dragon object"

`#include <CrewDragonCreator.h>`

Inheritance diagram for CrewDragonCreator:

Collaboration diagram for CrewDragonCreator:

Public Member Functions

- Spacecraft * createSpacecraft () override
  creates a new CargoDragonCreator

4.14.1 Detailed Description

"Creates a crew dragon object"

4.14.2 Member Function Documentation

4.14.2.1 createSpacecraft()

```
CrewDragonCreator::createSpacecraft ( )  [override], [virtual]
```

creates a new CargoDragonCreator

Returns

Spacecraft*

Implements SpacecraftCreator.

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

- include/CrewDragonCreator.h

4.15 CrewIterator Class Reference

"The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"

```
#include <CrewIterator.h>
```

Inheritance diagram for CrewIterator:

Collaboration diagram for CrewIterator:

Public Member Functions

- CrewIterator (LinkedListOfCrew *list)
  Constructor for this class.
- ~CrewIterator ()
  Destructor for this class.
- CarryType * first () override
  Gets the first element in the list/collection.
- CarryType * next () override
  Gets the next element in the list/collection.
- CarryType * current () override
  Gets the current element in the list/collection.
- bool isDone () override
  Determines if there are more elements.

4.15.1 Detailed Description

"The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"

4.15.2 Constructor & Destructor Documentation

4.15.2.1 CrewIterator()

```
CrewIterator::CrewIterator (
            LinkedListOfCrew * list )
```

Constructor for this class.

Parameters

in	*LinkedListOfCrew*∗	list of crew members

4.15.3 Member Function Documentation

4.15.3.1 current()

```
CrewIterator::current ( )  [override], [virtual]
```

Gets the current element in the list/collection.

Returns

CarryType∗ the current element in the list/collection

Reimplemented from Iterator.

4.15.3.2 first()

```
CrewIterator::first ( )  [override], [virtual]
```

Gets the first element in the list/collection.

Returns

CarryType∗ the first element in the list/collection

Reimplemented from Iterator.

4.15.3.3 isDone()

```
CrewIterator::isDone ( )  [override], [virtual]
```

Determines if there are more elements.

Returns

Boolean true if the end of the list/collection is reached

Reimplemented from Iterator.

4.15.3.4 next()

```
CrewIterator::next ( )  [override], [virtual]
```

Gets the next element in the list/collection.

Returns

CarryType* the next element in the list/collection

Reimplemented from Iterator.

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

- include/CrewIterator.h

4.16 Dragon Class Reference

"Interface for the dragon spacecraft"

```
#include <Dragon.h>
```

Inheritance diagram for Dragon:

Collaboration diagram for Dragon:

Public Member Functions

- Dragon (Falcon *)
  The destructor for the class.
- virtual ~Dragon ()
  The destructor for the class.
- virtual void load (bool)=0
  Method to load the content of the dragon spacecraft.
- virtual void unload (bool)=0
  Method to load the content of the dragon spacecraft.

4.16.1 Detailed Description

"Interface for the dragon spacecraft"

4.16.2 Member Function Documentation

4.16.2.1 load()

```
Dragon::load (
            bool  )  [pure virtual]
```

Method to load the content of the dragon spacecraft.

Returns

void

Implemented in CrewDragon, and CargoDragon.

4.16.2.2 unload()

```
Dragon::unload (
            bool  )  [pure virtual]
```

Method to load the content of the dragon spacecraft.

Returns

void

Implemented in CrewDragon, and CargoDragon.

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

- include/Dragon.h

4.17 Engine Class Reference

Inheritance diagram for Engine:

Public Member Functions

- Engine ()
  The constructor for the class.
- **Engine** (MerlinCore *m)
- virtual ~Engine ()
  The destructor for the class.
- void turnOn ()
  Turn engine on.
- void checkEngine ()
  Template method to check the engine.
- void setOn (bool)
  Set on state.
- virtual Engine * clone ()=0
  clone an engine.

Protected Member Functions

- virtual bool getOn () const

  Get on state.
- virtual void **checkTemperature** ()=0
- virtual void startEngine ()=0

  Completes the check up and finally start up the Engine.
- virtual void checkOil ()=0

  Check Oil level in Engine.

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Engine()

```
Engine::Engine ( )
```

The constructor for the class.

Parameters

in	*MerlinCore*∗	Desc

4.17.2 Member Function Documentation

4.17.2.1 checkEngine()

```
Engine::checkEngine ( )
```

Template method to check the engine.

Returns

void

4.17.2.2 checkOil()

```
Engine::checkOil ( )  [protected], [pure virtual]
```

Check Oil level in Engine.

Returns

void

Implemented in MerlinEngine, and MerlinVacuumEngine.

4.17.2.3 clone()

```
Engine::clone ( )  [pure virtual]
```

clone an engine.

Returns

Engine*

Implemented in MerlinEngine, and MerlinVacuumEngine.

4.17.2.4 getOn()

```
Engine::getOn ( ) const  [protected], [virtual]
```

Get on state.

Returns

void

4.17.2.5 setOn()

```
Engine::setOn (
            bool  )
```

Set on state.

Parameters

in	*bool*	The boolean to set the on state of the engine to.

Returns

void

4.17.2.6 startEngine()

```
Engine::startEngine ( )  [protected], [pure virtual]
```

Completes the check up and finally start up the Engine.

Returns

void

Implemented in MerlinEngine, and MerlinVacuumEngine.

4.17.2.7 turnOn()

```
Engine::turnOn ( )
```

Turn engine on.

Returns

void

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

- include/Engine.h

4.18 Falcon Class Reference

"Falcon Rocket"

```
#include <Falcon.h>
```

Inheritance diagram for Falcon:

Collaboration diagram for Falcon:

Public Member Functions

- Falcon (std::string)
  The constructor for the class.
- ~Falcon ()
  The destructor for the class.
- string getCurrentState () const
  Get current launch state.
- void change ()
  Change launch state.
- void setState (FalconState *)
  Set launch state.
- vector< MerlinCore * > getCoreList () const
  Set launch state.
- MerlinVacuumEngine * getVacuumEngine () const
  Ger Merlin Vacuum Engine.

4.18.1 Detailed Description

"Falcon Rocket"

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Falcon()

```
Falcon::Falcon (
            std::string  )
```

The constructor for the class.

Parameters

in	*std::string*	Type of falcon

4.18.3 Member Function Documentation

4.18.3.1 change()

```
Falcon::change ( )
```

Change launch state.

Returns

void

4.18.3.2 getCoreList()

```
Falcon::getCoreList ( ) const
```

Set launch state.

Returns

vector<MerlinCore*> list merlin cores associated with rocket

4.18.3.3 getCurrentState()

```
Falcon::getCurrentState ( ) const
```

Get current launch state.

Returns

string

4.18.3.4 getVacuumEngine()

```
Falcon::getVacuumEngine ( ) const
```

Ger Merlin Vacuum Engine.

Returns

MerlinVacuumEngine* reference to merlin vacuum engine

4.18.3.5 setState()

```
Falcon::setState (
            FalconState * )
```

Set launch state.

Parameters

in	*FalconState**	Falcon Launch State

Returns

void

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

- include/Falcon.h

4.19 Falcon9Creator Class Reference

Inheritance diagram for Falcon9Creator:

Collaboration diagram for Falcon9Creator:

Public Member Functions

- Spacecraft ∗ createSpacecraft () override
  creates a new create Falcon-9.

4.19.1 Member Function Documentation

4.19.1.1 createSpacecraft()

```
Falcon9Creator::createSpacecraft ( )  [override], [virtual]
```

creates a new create Falcon-9.

Returns

Spacecraft∗

Implements SpacecraftCreator.

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

- include/Falcon9Creator.h

4.20 FalconHeavyCreator Class Reference

Inheritance diagram for FalconHeavyCreator:

Collaboration diagram for FalconHeavyCreator:

Public Member Functions

- Spacecraft ∗ createSpacecraft () override
  creates a new Falcon-Heavy.

4.20.1 Member Function Documentation

4.20.1.1 createSpacecraft()

```
FalconHeavyCreator::createSpacecraft ( )  [override], [virtual]
```

creates a new Falcon-Heavy.

Returns

Spacecraft∗

Implements SpacecraftCreator.

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

- include/FalconHeavyCreator.h

4.21 FalconState Class Reference

"Interface for the current Falcon separation state"

```
#include <FalconState.h>
```

Inheritance diagram for FalconState:

Public Member Functions

- virtual ∼FalconState ()
  The destructor for the class.
- virtual void handleChange (Falcon ∗)=0
  Handles a change in separation state.
- virtual std::string getCurrentState ()=0
  Returns the current state name.

4.21.1 Detailed Description

"Interface for the current Falcon separation state"

4.21.2 Member Function Documentation

4.21.2.1 getCurrentState()

```
FalconState::getCurrentState ( )  [pure virtual]
```

Returns the current state name.

Parameters

out	*string*	String representing the name of the state

Returns

string

Implemented in Launched, Returned, Seperated, and Idle.

4.21.2.2 handleChange()

```
FalconState::handleChange (
            Falcon * ) [pure virtual]
```

Handles a change in separation state.

Parameters

in	*Falcon∗*	Context for the falcon changing state

Returns

void

Implemented in Launched, Returned, Seperated, and Idle.

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

- include/FalconState.h

4.22 Idle Class Reference

"Idle Falcon State"

`#include <Idle.h>`

Inheritance diagram for Idle:

Collaboration diagram for Idle:

Public Member Functions

- ∼Idle ()
  The destructor for the class.
- void handleChange (Falcon ∗) override
  Handles a change in separation state.
- std::string getCurrentState () override
  Returns the current state name.

4.22.1 Detailed Description

"Idle Falcon State"

4.22.2 Member Function Documentation

4.22.2.1 getCurrentState()

```
Idle::getCurrentState ( )  [override], [virtual]
```

Returns the current state name.

Parameters

out	*string*	String representing the name of the state

Returns

string

Implements FalconState.

4.22.2.2 handleChange()

```
Idle::handleChange (
            Falcon *  )  [override], [virtual]
```

Handles a change in separation state.

Parameters

in	*Falcon*∗	Context for the falcon changing state

Returns

void

Implements FalconState.

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

- include/Idle.h

4.23 Iterator Class Reference

"The base class of the cargo- and crew-list iterators (Iterator-Iterator)"

`#include <Iterator.h>`

Inheritance diagram for Iterator:

Public Member Functions

- virtual CarryType * first ()

  Gets the first element in the list/collection.
- virtual CarryType * next ()

  Gets the next element in the list/collection.
- virtual CarryType * current ()

  Gets the current element in the list/collection.
- virtual bool isDone ()

  Determines if there are more elements.

4.23.1 Detailed Description

"The base class of the cargo- and crew-list iterators (Iterator-Iterator)"

4.23.2 Member Function Documentation

4.23.2.1 current()

`Iterator::current ( ) [virtual]`

Gets the current element in the list/collection.

Returns

CarryType*

Reimplemented in CargoIterator, and CrewIterator.

4.23.2.2 first()

```
Iterator::first ( )  [virtual]
```

Gets the first element in the list/collection.

Returns

CarryType*

Reimplemented in CargoIterator, and CrewIterator.

4.23.2.3 isDone()

```
Iterator::isDone ( )  [virtual]
```

Determines if there are more elements.

Returns

Boolean: true if the end of the list/collection is reached

Reimplemented in CargoIterator, and CrewIterator.

4.23.2.4 next()

```
Iterator::next ( )  [virtual]
```

Gets the next element in the list/collection.

Returns

CarryType*

Reimplemented in CargoIterator, and CrewIterator.

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

- include/Iterator.h

4.24 KeplerianCoords Class Reference

"Class for encoding Keplerian Coordinates for Satellites"

```
#include <KeplerianCoords.h>
```

Public Member Functions

- KeplerianCoords ()

  The constructor for the class.
- **KeplerianCoords** (int, int, int, int, int)
- ~KeplerianCoords ()

  The destructor for the class.
- std::string toString () const

  Serialises the attributes of the class.
- void randomiseCoords ()

  Randomises the values of attributes.
- bool **operator**< (const KeplerianCoords &)

4.24.1 Detailed Description

"Class for encoding Keplerian Coordinates for Satellites"

4.24.2 Constructor & Destructor Documentation

4.24.2.1 KeplerianCoords()

```
KeplerianCoords::KeplerianCoords ( )
```

The constructor for the class.

Parameters

in	*int*	angle between equator and orbit plane
in	*semiMajorAxis*	
in	*trueAnomaly*	
in	*rightAscension*	
in	*argumentOfPerigree*	

4.24.3 Member Function Documentation

4.24.3.1 randomiseCoords()

```
KeplerianCoords::randomiseCoords ( )
```

Randomises the values of attributes.

Used in determining intersection of coords.

Returns

void

Parameters

in	*KeplerianCoords&*	Reference to KeplerianCoords object for comparison

4.24.3.2 toString()

`KeplerianCoords::toString ( ) const`

Serialises the attributes of the class.

Returns

string

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

- include/KeplerianCoords.h

4.25 Launched Class Reference

"Launched Falcon State"

`#include <Launched.h>`

Inheritance diagram for Launched:

Collaboration diagram for Launched:

Public Member Functions

- ~Launched ()
  The destructor for the class.
- void handleChange (Falcon *) override
  Handles a change in seperation state.
- std::string getCurrentState () override
  Returns the current state name.

4.25.1 Detailed Description

"Launched Falcon State"

4.25.2 Member Function Documentation

4.25.2.1 getCurrentState()

`Launched::getCurrentState ( ) [override], [virtual]`

Returns the current state name.

Parameters

out	*string*	String representing the name of the state

Returns

string

Implements FalconState.

4.25.2.2 handleChange()

```
Launched::handleChange (
            Falcon *  )  [override], [virtual]
```

Handles a change in seperation state.

Parameters

in	*Falcon*∗	Context for the falcon changing state

Returns

void

Implements FalconState.

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

- include/Launched.h

4.26 LinkedListOfCrew Class Reference

"A class to contain a linked list of Crew (ConcreteAggregate-Iterator)"

```
#include <LinkedListOfCrew.h>
```

Inheritance diagram for LinkedListOfCrew:

Collaboration diagram for LinkedListOfCrew:

Public Member Functions

- Iterator * createIterator ()

  creates a CrewIterator element
- LinkedListOfCrew ()

  The constructor of the class.
- ~LinkedListOfCrew ()

  The destructor of the class.
- void removeCrewMember (Crew *member)

  Removes the specified crew item from the list/collection/vector.
- void addCrewMember (Crew *member)

  Adds the specified crew item to the list/collection/vector.
- int getSize ()

  Returns the size of the linked list.
- Crew * getHead ()

  Returns the head of the linked list.

4.26.1 Detailed Description

"A class to contain a linked list of Crew (ConcreteAggregate-Iterator)"

4.26.2 Member Function Documentation

4.26.2.1 addCrewMember()

```
LinkedListOfCrew::addCrewMember (
            Crew * member )
```

Adds the specified crew item to the list/collection/vector.

Parameters

in	*Crew**	Crew Member to be added to the list

Returns

void

4.26.2.2 createIterator()

```
LinkedListOfCrew::createIterator ( )
```

creates a CrewIterator element

Returns

Iterator* a new instance of CrewIterator to iterate through CrewList

4.26.2.3 getHead()

```
LinkedListOfCrew::getHead ( )
```

Returns the head of the linked list.

Returns

Crew*

4.26.2.4 getSize()

```
LinkedListOfCrew::getSize ( )
```

Returns the size of the linked list.

Returns

int

4.26.2.5 removeCrewMember()

```
LinkedListOfCrew::removeCrewMember (
            Crew * member )
```

Removes the specified crew item from the list/collection/vector.

Parameters

in	*Crew**	Crew Member to be removed to the list

Returns

void

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

- include/LinkedListOfCrew.h

4.27 Loader Class Reference

"Context for loading dragon rocket"

`#include <Loader.h>`

Public Member Functions

- Loader (Dragon ∗)
  The constructor for the class.
- ∼Loader ()
  The destructor for the class.
- void load (bool)
  Performs load action on member dragon.
- void unload (bool)
  Performs unload action on member dragon.
- void **setDragon** (Dragon ∗)

4.27.1 Detailed Description

"Context for loading dragon rocket"

4.27.2 Member Function Documentation

4.27.2.1 load()

```
Loader::load (
            bool  )
```

Performs load action on member dragon.

Parameters

in	*bool*	Indicate whether to print loading data

Returns

void

4.27.2.2 unload()

```
Loader::unload (
            bool  )
```

Performs unload action on member dragon.

Parameters

in	*bool*	Indicate whether to print loading data

Returns

void

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

- include/Loader.h

4.28 Memento Class Reference

The Memento for the Memento Design Pattern.

`#include <Memento.h>`

Public Member Functions

- Memento ()
  The constructor for the class.
- ~Memento ()
  The destructor for the class.
- State * getState ()
  Get state attribute.
- void setState (State *s)
  The destructor for the class.

4.28.1 Detailed Description

The Memento for the Memento Design Pattern.

4.28.2 Member Function Documentation

4.28.2.1 getState()

```
Memento::getState ( )
```

Get state attribute.

Returns

State* state attribute

4.28.2.2 setState()

```
Memento::setState (
            State * s )
```

The destructor for the class.

Parameters

in	*State*∗	reference to state that will be set

Returns

void

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

- include/Memento.h

4.29 MerlinCore Class Reference

The Core of engines used on Falcon rockets.

`#include <MerlinCore.h>`

Public Member Functions

- MerlinCore ()
  The constructor for the class.
- void on (Engine ∗colleague)
  Sets the on to true for the engine parameter.
- void off (Engine ∗colleague)
  Sets the on to false for the engine parameter.
- ∼MerlinCore ()
  The destructor for the class.
- void initiateEngineChecks ()
  check every engine

4.29.1 Detailed Description

The Core of engines used on Falcon rockets.

4.29.2 Member Function Documentation

4.29.2.1 initiateEngineChecks()

`MerlinCore::initiateEngineChecks ( )`

check every engine

Returns

void

4.29.2.2 off()

```
MerlinCore::off (
            Engine * colleague )
```

Sets the on to false for the engine parameter.

Parameters

in	*Engine*∗	that will be turned off

Returns

void

4.29.2.3 on()

```
MerlinCore::on (
            Engine * colleague )
```

Sets the on to true for the engine parameter.

Parameters

in	*Engine*∗	that will be turned on

Returns

void

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

- include/MerlinCore.h

4.30 MerlinEngine Class Reference

""

```
#include <MerlinEngine.h>
```

Inheritance diagram for MerlinEngine:

Collaboration diagram for MerlinEngine:

Public Member Functions

- void checkTemperature () override
  Check Temperature of Engine.
- void startEngine () override
  Completes the check up and finally start up the Engine.
- void checkOil () override
  Check Oil level in Engine.
- Engine ∗ clone () override
  clone an engine.

Additional Inherited Members

4.30.1 Detailed Description

""

4.30.2 Member Function Documentation

4.30.2.1 checkOil()

```
MerlinEngine::checkOil ( )  [override], [virtual]
```

Check Oil level in Engine.

Returns

void

Implements Engine.

4.30.2.2 checkTemperature()

```
MerlinEngine::checkTemperature ( )  [override], [virtual]
```

Check Temperature of Engine.

Returns

void

Implements Engine.

4.30.2.3 clone()

```
MerlinEngine::clone ( )  [override], [virtual]
```

clone an engine.

Returns

Engine*

Implements Engine.

4.30.2.4 startEngine()

```
MerlinEngine::startEngine ( )  [override], [virtual]
```

Completes the check up and finally start up the Engine.

Returns

void

Implements Engine.

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

- include/MerlinEngine.h

4.31 MerlinVacuumEngine Class Reference

""

```
#include <MerlinVacuumEngine.h>
```

Inheritance diagram for MerlinVacuumEngine:

Collaboration diagram for MerlinVacuumEngine:

Public Member Functions

- void checkTemperature () override
  Check Temperature of Engine.
- void startEngine () override
  Completes the check up and finally start up the Engine.
- void checkOil () override
  Check Oil level in Engine.
- Engine * clone () override
  clone an engine.

Additional Inherited Members

4.31.1 Detailed Description

""

4.31.2 Member Function Documentation

4.31.2.1 checkOil()

```
MerlinVacuumEngine::checkOil ( )  [override], [virtual]
```

Check Oil level in Engine.

Returns

void

Implements Engine.

4.31.2.2 checkTemperature()

```
MerlinVacuumEngine::checkTemperature ( )  [override], [virtual]
```

Check Temperature of Engine.

Returns

void

Implements Engine.

4.31.2.3 clone()

```
MerlinVacuumEngine::clone ( )  [override], [virtual]
```

clone an engine.

Returns

Engine∗

Implements Engine.

4.31.2.4 startEngine()

```
MerlinVacuumEngine::startEngine ( )  [override], [virtual]
```

Completes the check up and finally start up the Engine.

Returns

void

Implements Engine.

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

- include/MerlinVacuumEngine.h

4.32 MissionControl Class Reference

"MissionControl for controlling rockets and other spacecraft"

`#include <MissionControl.h>`

Public Member Functions

- MissionControl ()

  The constructor for the class.
- ~MissionControl ()

  The destructor for the class.
- void receiveRadioSignal (int, std::string)

  Receive radio signal from a Satelite.
- void sendRepositionRequest (Satelite *)

  Send a reposition request to the referenced Satelite.

4.32.1 Detailed Description

"MissionControl for controlling rockets and other spacecraft"

4.32.2 Member Function Documentation

4.32.2.1 receiveRadioSignal()

```
MissionControl::receiveRadioSignal (
            int ,
            std::string  )
```

Receive radio signal from a Satelite.

Parameters

in	*int*	satelliteID
in	*std::string*	Satellite Keplerian Coordinates

Returns

void

4.32.2.2 sendRepositionRequest()

```
MissionControl::sendRepositionRequest (
            Satelite *  )
```

Send a reposition request to the referenced Satelite.

Parameters

in	*Satellite*∗	Reference to Satelite to be moved

Returns

void

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

- include/MissionControl.h

4.33 Returned Class Reference

"Returned Falcon State"

`#include <Returned.h>`

Inheritance diagram for Returned:

Collaboration diagram for Returned:

Public Member Functions

- ∼Returned ()
  The destructor for the class.
- void handleChange (Falcon ∗) override
  Handles a change in seperation state.
- std::string getCurrentState () override
  Returns the current state name.

4.33.1 Detailed Description

"Returned Falcon State"

4.33.2 Member Function Documentation

4.33.2.1 getCurrentState()

```
Returned::getCurrentState ( )  [override], [virtual]
```

Returns the current state name.

Parameters

out	*string*	String representing the name of the state

Returns

string

Implements FalconState.

4.33.2.2 handleChange()

```
Returned::handleChange (
            Falcon *  )  [override], [virtual]
```

Handles a change in seperation state.

Parameters

in	*Falcon*∗	Context for the falcon changing state

Returns

void

Implements FalconState.

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

- include/Returned.h

4.34 Satelite Class Reference

"Starlink Satelite"

```
#include <Satelite.h>
```

Inheritance diagram for Satelite:

Collaboration diagram for Satelite:

Public Member Functions

- Satelite ()
  The constructor for the class.
- Satelite (const Satelite &)
  The copy constructor for the class.
- ~Satelite ()
  The destructor for the class.
- Satelite * clone ()
  Clone the Current Satelite.
- void positionSelf ()
  Position self in the Satellite cluster.
- void sendGroundSignal ()
  Send signal to mission control.
- void sendSatelliteSignal (Satelite *)
  Send laser signal to fellow satellite.
- void setMissionControl (MissionControl *)
  Set reference to mission control to which radio signals are sent.
- KeplerianCoords * getCoords () const
  Return the Keplerian coordinates of the current satellite.

4.34.1 Detailed Description

"Starlink Satelite"

4.34.2 Constructor & Destructor Documentation

4.34.2.1 Satelite()

```
Satelite::Satelite (
            const Satelite &  )
```

The copy constructor for the class.

Parameters

in	*Satelite&*	The satellite to be copied.

4.34.3 Member Function Documentation

4.34.3.1 clone()

```
Satelite::clone ( )
```

Clone the Current Satelite.

Returns

Satelite* Cloned Satelite

4.34.3.2 getCoords()

```
Satelite::getCoords ( ) const
```

Return the Keplerian coordinates of the current satellite.

Returns

KeplerianCoords* pointer to the coordinates object

4.34.3.3 positionSelf()

```
Satelite::positionSelf ( )
```

Position self in the Satellite cluster.

Returns

void

4.34.3.4 sendGroundSignal()

```
Satelite::sendGroundSignal ( )
```

Send signal to mission control.

Returns

void

4.34.3.5 sendSatelliteSignal()

```
Satelite::sendSatelliteSignal (
            Satelite * )
```

Send laser signal to fellow satellite.

Parameters

in	*Satelite*∗	A pointer to the satellite to check collision with.

Returns

void

4.34.3.6 setMissionControl()

```
Satelite::setMissionControl (
            MissionControl *  )
```

Set reference to mission control to which radio signals are sent.

Parameters

in	*MissionControl*∗	The mission control to use.

Returns

void

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

- include/Satelite.h

4.35 SatelliteCreator Class Reference

Inheritance diagram for SatelliteCreator:

Collaboration diagram for SatelliteCreator:

Public Member Functions

- Spacecraft ∗ createSpacecraft () override
  creates a new CargoDragonCreator

4.35.1 Member Function Documentation

4.35.1.1 createSpacecraft()

```
SatelliteCreator::createSpacecraft ( )  [override], [virtual]
```

creates a new CargoDragonCreator

Returns

Spacecraft

Implements SpacecraftCreator.

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

- include/SatelliteCreator.h

4.36 SelectCommand Class Reference

The Concrete Command for the Command Design Pattern.

`#include <SelectCommand.h>`

Inheritance diagram for SelectCommand:

Collaboration diagram for SelectCommand:

Public Member Functions

- SelectCommand (SelectSimulation *s)
  The constructor for the class.
- ~SelectCommand ()
  The destructor for the class.
- virtual void execute (std::string s, std::vector< State * > *v)
  The implementation of the virtual execute method.

4.36.1 Detailed Description

The Concrete Command for the Command Design Pattern.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 SelectCommand()

```
SelectCommand::SelectCommand (
            SelectSimulation * s )
```

The constructor for the class.

Parameters

in	*s*	The simulation to be used in the command.

4.36.3 Member Function Documentation

4.36.3.1 execute()

```
void SelectCommand::execute (
            std::string s,
            std::vector< State * > * v )  [virtual]
```

The implementation of the virtual execute method.

Parameters

in	*s*	String to determine which simulation to run based off the chain of responsibility.
in	*v*	A pointer to the vector of states used to run the simulations.

Reimplemented from Command.

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

- include/SelectCommand.h

4.37 SelectSimulation Class Reference

The implementation of the selection process.

`#include <SelectSimulation.h>`

Inheritance diagram for SelectSimulation:

Collaboration diagram for SelectSimulation:

Public Member Functions

- SelectSimulation ()
  The constructor for the class.
- ~SelectSimulation ()
  The destructor for the class.
- virtual void startSim (std::vector< State * > *sVector) override
  Starts the select simulation.
- void simulateSingle (std::vector< State * > *sVector)

Simulate a single simulation.

- void simulateBatch (std::vector< State * > *sVector)

  The simulate multiple simulations.

- void exitProgram ()

  Starts exit procedure.

- void loadPrefabs ()

  Load saved simulations fro a file.

4.37.1 Detailed Description

The implementation of the selection process.

4.37.2 Member Function Documentation

4.37.2.1 exitProgram()

```
SelectSimulation::exitProgram ( )
```

Starts exit procedure.

Returns

void

4.37.2.2 loadPrefabs()

```
SelectSimulation::loadPrefabs ( )
```

Load saved simulations fro a file.

Returns

void.

4.37.2.3 simulateBatch()

```
SelectSimulation::simulateBatch (
            std::vector< State * > * sVector )
```

The simulate multiple simulations.

Parameters

in	*sVector*	A pointer to the vector of states used to run the simulations.

Returns

void

4.37.2.4 simulateSingle()

```
SelectSimulation::simulateSingle (
            std::vector< State * > * sVector )
```

Simulate a single simulation.

Parameters

in	*sVector*	A pointer to the vector of states used to run the simulations.

Returns

void

4.37.2.5 startSim()

```
SelectSimulation::startSim (
            std::vector< State * > * sVector )  [override], [virtual]
```

Starts the select simulation.

Parameters

in	*sVector*	A pointer to the vector of states used to run the simulations.

Returns

void

Implements Simulation.

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

- include/SelectSimulation.h

4.38 Seperated Class Reference

"Seperated falcon state"

`#include <Seperated.h>`

Inheritance diagram for Seperated:

Collaboration diagram for Seperated:

Public Member Functions

- ~Seperated ()
  The destructor for the class.
- void handleChange (Falcon *) override
  Handles a change in seperation state.
- std::string getCurrentState () override
  Returns the current state name.

4.38.1 Detailed Description

"Seperated falcon state"

4.38.2 Member Function Documentation

4.38.2.1 getCurrentState()

`Seperated::getCurrentState ( ) [override], [virtual]`

Returns the current state name.

Parameters

`out`	*string*	String representing the name of the state

Returns

string

Implements FalconState.

4.38.2.2 handleChange()

```
Seperated::handleChange (
            Falcon *  )  [override], [virtual]
```

Handles a change in seperation state.

Parameters

in	*Falcon*∗	Context for the falcon changing state

Returns

void

Implements FalconState.

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

- include/Seperated.h

4.39 Simulate Class Reference

The invoker for the Command Design Pattern.

```
#include <Simulate.h>
```

Public Member Functions

- Simulate ()
  The constructor for the class.
- ∼Simulate ()
  The destructor for the class.
- std::vector< State ∗ > select ()
  The method to call selectCommand execute().
- std::vector< State ∗ > build ()
  The method to call buildCommand execute().
- void setSelect (Command ∗c)
  The method to set the selectCommand.
- void setBuild (Command ∗c)
  The method to set the buildCommand.

4.39.1 Detailed Description

The invoker for the Command Design Pattern.

4.39.2 Member Function Documentation

4.39.2.1 build()

```
std::vector< State * > Simulate::build ( )
```

The method to call buildCommand execute().

Returns

std::vector<State*> vector of states

4.39.2.2 select()

```
std::vector< State * > Simulate::select ( )
```

The method to call selectCommand execute().

Returns

std::vector<State*> vector of states

4.39.2.3 setBuild()

```
void Simulate::setBuild (
            Command * c )
```

The method to set the buildCommand.

Parameters

in	*c*	The command to work with.

Returns

void

4.39.2.4 setSelect()

```
void Simulate::setSelect (
            Command * c )
```

The method to set the selectCommand.

Parameters

in	*c*	The command to work with.

Returns

void

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

- include/Simulate.h

4.40 Simulation Class Reference

The interface for all Simulations, the Originator for the Memento Design Pattern and the Reciever for the Command Design Pattern.

`#include <Simulation.h>`

Inheritance diagram for Simulation:

Public Member Functions

- Simulation ()
  The constructor for the class.
- virtual ~Simulation ()
  The destructor for the class.
- Memento * createMemento ()
  The function to create the memento.
- void setMemento (Memento *m)
  The function will set the state according to the memento.
- virtual void startSim (std::vector< State * > *sVector)=0
  The pure virtual function to start the current sim.
- std::string getFilePath ()
  The getter for the file path to the saved simulations.
- State * getState ()
  The getter for the abstraction of the current state.

4.40.1 Detailed Description

The interface for all Simulations, the Originator for the Memento Design Pattern and the Reciever for the Command Design Pattern.

4.40.2 Member Function Documentation

4.40.2.1 createMemento()

```
Memento * Simulation::createMemento ( )
```

The function to create the memento.

Returns

Memento*

4.40.2.2 getFilePath()

```
std::string Simulation::getFilePath ( )
```

The getter for the file path to the saved simulations.

Returns

std::string

4.40.2.3 getState()

```
State * Simulation::getState ( )
```

The getter for the abstraction of the current state.

Returns

State*

4.40.2.4 setMemento()

```
void Simulation::setMemento (
            Memento * m )
```

The function will set the state according to the memento.

Parameters

in	*m*	The passed in memento object that will be used to reinstate the State object.

Returns

void

4.40.2.5 startSim()

```
std::vector< State * > Simulation::startSim (
            std::vector< State * > * sVector )  [pure virtual]
```

The pure virtual function to start the current sim.

Parameters

in	*sVector*	A pointer to the vector of states used to run the simulations.

Returns

void

Implemented in BuildSimulation, and SelectSimulation.

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

- include/Simulation.h

4.41 Spacecraft Class Reference

"Interface for all spacecraft"

```
#include <Spacecraft.h>
```

Inheritance diagram for Spacecraft:

Public Member Functions

- Spacecraft (string)
  The constructor for the class.
- string getType () const
  The constructor for the class.

4.41.1 Detailed Description

"Interface for all spacecraft"

4.41.2 Constructor & Destructor Documentation

4.41.2.1 Spacecraft()

```
Spacecraft::Spacecraft (
            string  )
```

The constructor for the class.

Parameters

in	*string*	Spacecraft type

4.41.3 Member Function Documentation

4.41.3.1 getType()

```
Spacecraft::getType ( ) const
```

The constructor for the class.

Returns

string Spacecraft type

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

- include/Spacecraft.h

4.42 SpacecraftCreator Class Reference

"Interface for the creation of spacecraft"

```
#include <SpacecraftCreator.h>
```

Inheritance diagram for SpacecraftCreator:

Public Member Functions

- virtual Spacecraft * createSpacecraft ()=0
  interface method for creating a spacecraft

4.42.1 Detailed Description

"Interface for the creation of spacecraft"

4.42.2 Member Function Documentation

4.42.2.1 createSpacecraft()

```
SpacecraftCreator::createSpacecraft ( )  [pure virtual]
```

interface method for creating a spacecraft

Returns

newly created Spacecraft

Implemented in CrewDragonCreator, CargoDragonCreator, Falcon9Creator, SatelliteCreator, and FalconHeavyCreator.

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

- include/SpacecraftCreator.h

4.43 spreadCommand Class Reference

Command to spread out the satellites.

```
#include <SpreadCommand.h>
```

Inheritance diagram for spreadCommand:

Collaboration diagram for spreadCommand:

Public Member Functions

- spreadCommand (Cluster *c)
  The constructor for the class.
- ~spreadCommand ()
  The destructor for the class.
- void execute ()
  The implementation of the virtual execute method.

4.43.1 Detailed Description

Command to spread out the satellites.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 spreadCommand()

```
spreadCommand::spreadCommand (
            Cluster * c )
```

The constructor for the class.

Parameters

in	*c*	The cluster to be used.

4.43.3 Member Function Documentation

4.43.3.1 execute()

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

The implementation of the virtual execute method.

Returns

void

Reimplemented from Command.

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

- include/SpreadCommand.h

4.44 State Class Reference

The abstraction of the State of the simulation.

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

Public Member Functions

- State ()
  The constructor for the class.
- **State** (State ∗s)
- **State** (std::string n, Spacecraft ∗s)
- **State** (std::string n, Cluster ∗c)
- ∼State ()
  The destructor for the class.
- std::string getName ()
  The getter for the name of the simulation.
- void setName (std::string s)
  The setter for the name of the simulation.
- Spacecraft ∗ getVessel ()
  The getter for the spacecraft.
- void setVessel (Spacecraft ∗s)
  The setter for the name of the simulation.
- Cluster ∗ getCluster ()
  The getter for the cluster of satellites.
- void setCluster (Cluster ∗c)
  The setter for the name of the simulation.
- std::vector< Command ∗ > getCommands ()
  The getter for the commands to be executed.
- void addCommand (Command ∗c)
  The method to add a command to the list of commands.
- void remLastCommand ()
  The method to pop the last command.
- void runCommands ()
  The method to execute all the commands in the list.

4.44.1 Detailed Description

The abstraction of the State of the simulation.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 State()

```
State::State ( )
```

The constructor for the class.

The overloade constructor for the class to be used with a name and a cluster.

The overloaded constructor for the class to be used with a name and a spacecraft.

The copy constructor for the class.

Parameters

in	*s*	The State to copy.
in	*n*	The name of the simulation.
in	*s*	The spacecraft that will be used.
in	*n*	The name of the simulation.
in	*c*	The cluster to be used.

4.44.3 Member Function Documentation

4.44.3.1 addCommand()

```
State::addCommand (
            Command * c )
```

The method to add a command to the list of commands.

Parameters

in	*c*	The command to add.

Returns

void

4.44.3.2 getCluster()

```
State::getCluster ( )
```

The getter for the cluster of satellites.

Returns

Cluster*

4.44.3.3 getCommands()

```
State::getCommands ( )
```

The getter for the commands to be executed.

Returns

std::vector<Command*>

4.44.3.4 getName()

```
State::getName ( )
```

The getter for the name of the simulation.

Returns

std::string

4.44.3.5 getVessel()

```
State::getVessel ( )
```

The getter for the spacecraft.

Returns

Spacecraft*

4.44.3.6 remLastCommand()

```
State::remLastCommand ( )
```

The method to pop the last command.

Returns

void

4.44.3.7 runCommands()

```
State::runCommands ( )
```

The method to execute all the commands in the list.

Returns

void

4.44.3.8 setCluster()

```
State::setCluster (
            Cluster * c )
```

The setter for the name of the simulation.

Parameters

in	*c*	The cluster of satellites.

Returns

void

4.44.3.9 setName()

```
State::setName (
            std::string s )
```

The setter for the name of the simulation.

Parameters

in	*s*	The name of the simulation.

Returns

void

4.44.3.10 setVessel()

```
State::setVessel (
            Spacecraft * s )
```

The setter for the name of the simulation.

Parameters

in	*s*	The spacecraft.

Returns

void

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

- include/State.h

4.45 StateChangeCommand Class Reference

Concrete Command for the Command Design Pattern.

`#include <StateChangeCommand.h>`

Inheritance diagram for StateChangeCommand:

Collaboration diagram for StateChangeCommand:

Public Member Functions

- StateChangeCommand (Falcon *f)
  The constructor for the class.
- ~StateChangeCommand ()
  The destructor for the class.
- virtual void execute ()
  The implementation of the virtual execute method.

4.45.1 Detailed Description

Concrete Command for the Command Design Pattern.

4.45.2 Constructor & Destructor Documentation

4.45.2.1 StateChangeCommand()

```
StateChangeCommand::StateChangeCommand (
            Falcon * f )
```

The constructor for the class.

Parameters

in	*Falcon**	The falcon to change the state of.

4.45.3 Member Function Documentation

4.45.3.1 execute()

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

The implementation of the virtual execute method.

Returns

void

Reimplemented from Command.

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

- include/StateChangeCommand.h

4.46 Store Class Reference

The Caretaker for the Memento Design Pattern.

`#include <Store.h>`

Public Member Functions

- Store ()
  The constructor for the class.
- ~Store ()
  The destructor for the class.
- void storeMemento (Memento *m)
  The method to store a memento.
- Memento * returnMemento ()
  The getter for the stored memento.

4.46.1 Detailed Description

The Caretaker for the Memento Design Pattern.

4.46.2 Member Function Documentation

4.46.2.1 returnMemento()

```
Store::returnMemento ( )
```

The getter for the stored memento.

Returns

Memento*

4.46.2.2 storeMemento()

```
Store::storeMemento (
            Memento * m )
```

The method to store a memento.

Parameters

in	*m*	The memento to save.

Returns

void

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

- include/Store.h

4.47 UnloadCommand Class Reference

Concrete Command for the Command Design Pattern.

`#include <UnloadCommand.h>`

Inheritance diagram for UnloadCommand:

Collaboration diagram for UnloadCommand:

Public Member Functions

- UnloadCommand (Loader *l)
  The constructor for the class.
- ~UnloadCommand ()
  The destructor for the class.
- virtual void execute ()
  The implementation of the virtual execute method.

4.47.1 Detailed Description

Concrete Command for the Command Design Pattern.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 UnloadCommand()

```
UnloadCommand::UnloadCommand (
            Loader * l )
```

The constructor for the class.

Parameters

in	/	The loader to be used.

4.47.3 Member Function Documentation

4.47.3.1 execute()

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

The implementation of the virtual execute method.

Returns

void

Reimplemented from Command.

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

- include/UnloadCommand.h

4.48 VectorOfCargo Class Reference

"A class to contain a vector of Cargo (ConcreteAggregate-Iterator)"

```
#include <VectorOfCargo.h>
```

Inheritance diagram for VectorOfCargo:

Collaboration diagram for VectorOfCargo:

Public Member Functions

- Iterator * **createIterator** ()
- ~VectorOfCargo ()

  The destructor of the class.
- void removeCargo (Cargo *cargo)

  Removes the specified cargo item from the list/collection/vector.
- void addCargo (Cargo *cargo)

  Adds the specified cargo item to the list/collection/vector.

4.48.1 Detailed Description

"A class to contain a vector of Cargo (ConcreteAggregate-Iterator)"

4.48.2 Member Function Documentation

4.48.2.1 addCargo()

```
VectorOfCargo::addCargo (
            Cargo * cargo )
```

Adds the specified cargo item to the list/collection/vector.

Parameters

in	*Cargo*∗	The cargo to add to the vector.

Returns

void

4.48.2.2 removeCargo()

```
VectorOfCargo::removeCargo (
            Cargo * cargo )
```

Removes the specified cargo item from the list/collection/vector.

Parameters

in	*Cargo*∗	The specific cargo to be removed from the vector.

Returns

void

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

- include/VectorOfCargo.h

Chapter 5

File Documentation

5.1 include/BuildCommand.h File Reference

The header file for the BuildCommand class.

```
#include "Command.h"
#include "BuildSimulation.h"
```

Include dependency graph for BuildCommand.h:

5.2 include/CargoDragon.h File Reference

The header file for the CargoDragon class.

```
#include "Dragon.h"
#include "Falcon.h"
#include "Cargo.h"
#include "Engine.h"
#include <vector>
#include "VectorOfCargo.h"
```

Include dependency graph for CargoDragon.h: This graph shows which files directly or indirectly include this file:

Classes

- class CargoDragon

  "Dragon specialisation to carry cargo only"

5.2.1 Detailed Description

The header file for the CargoDragon class.

5.3 include/CargoDragonCreator.h File Reference

The header file for the CargoDragonCreator class.

```
#include "SpacecraftCreator.h"
#include "CargoDragon.h"
```
Include dependency graph for CargoDragonCreator.h:

Classes

- class CargoDragonCreator

5.3.1 Detailed Description

The header file for the CargoDragonCreator class.

5.4 include/CargoIterator.h File Reference

The header file for the CargoIterator class.

```
#include "CarryType.h"
#include "Iterator.h"
#include "Cargo.h"
#include <vector>
```
Include dependency graph for CargoIterator.h: This graph shows which files directly or indirectly include this file:

Classes

- class CargoIterator

  "The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"

5.4.1 Detailed Description

The header file for the CargoIterator class.

5.5 include/CarryType.h File Reference

"The header file for the base class CarryType"

```
#include <string>
```
Include dependency graph for CarryType.h: This graph shows which files directly or indirectly include this file:

Classes

- class CarryType

  "A base class for Crew and Cargo"

5.5.1 Detailed Description

"The header file for the base class CarryType"

5.6 include/Cluster.h File Reference

The header file for the Cluster class.

```
#include "Satelite.h"
#include "MissionControl.h"
#include "Falcon.h"
#include <vector>
```

Include dependency graph for Cluster.h: This graph shows which files directly or indirectly include this file:

Classes

- class Cluster

  "Represents a cluster of satellite"

5.6.1 Detailed Description

The header file for the Cluster class.

5.7 include/Command.h File Reference

The header file for the Command class.

```
#include <vector>
#include <iostream>
#include <string>
#include "State.h"
```

Include dependency graph for Command.h: This graph shows which files directly or indirectly include this file:

Classes

- class Command

  The Command for the Command Design Pattern and the Handler for the Chain of Responsibility.

5.7.1 Detailed Description

The header file for the Command class.

5.8 include/CrewDragonCreator.h File Reference

The header file for the CrewDragonCreator class.

```
#include "SpacecraftCreator.h"
#include "CrewDragon.h"
```

Include dependency graph for CrewDragonCreator.h:

Classes

- class CrewDragonCreator

  "Creates a crew dragon object"

5.8.1 Detailed Description

The header file for the CrewDragonCreator class.

5.9 include/CrewIterator.h File Reference

The header file for the CrewIterator class.

```
#include "CarryType.h"
#include "Iterator.h"
#include "Crew.h"
#include "LinkedListOfCrew.h"
#include <vector>
```

Include dependency graph for CrewIterator.h:

Classes

- class CrewIterator

  "The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"

5.9.1 Detailed Description

The header file for the CrewIterator class.

5.10 include/Dragon.h File Reference

The header file for the Dragon.h class.

```
#include "Spacecraft.h"
#include "Falcon.h"
#include <fstream>
```

Include dependency graph for Dragon.h: This graph shows which files directly or indirectly include this file:

Classes

- class Dragon

  "Interface for the dragon spacecraft"

5.10.1 Detailed Description

The header file for the Dragon.h class.

5.11 include/Engine.h File Reference

The header file for the Engine class.

```
#include "MerlinCore.h"
```

Include dependency graph for Engine.h: This graph shows which files directly or indirectly include this file:

Classes

- class Engine

5.11.1 Detailed Description

The header file for the Engine class.

5.12 include/Falcon.h File Reference

The header file for the Falcon class.

```
#include "Spacecraft.h"
#include "MerlinCore.h"
#include "MerlinVacuumEngine.h"
#include <vector>
#include "FalconState.h"
```

Include dependency graph for Falcon.h: This graph shows which files directly or indirectly include this file:

Classes

- class Falcon

  "Falcon Rocket"

5.12.1 Detailed Description

The header file for the Falcon class.

5.13 include/FalconHeavyCreator.h File Reference

The header file for the FalconHeavyCreator class.

```
#include "SpacecraftCreator.h"
#include "Falcon.h"
```

Include dependency graph for FalconHeavyCreator.h: This graph shows which files directly or indirectly include this file:

Classes

- class FalconHeavyCreator

5.13.1 Detailed Description

The header file for the FalconHeavyCreator class.

5.14 include/FalconState.h File Reference

The header file for the FalconState class.

```
#include <string>
#include "Falcon.h"
```

Include dependency graph for FalconState.h: This graph shows which files directly or indirectly include this file:

Classes

- class FalconState

  "Interface for the current Falcon separation state"

5.14.1 Detailed Description

The header file for the FalconState class.

5.15 include/Idle.h File Reference

The header file for the Idle class.

`#include "FalconState.h"`
Include dependency graph for Idle.h:

Classes

- class Idle
  "Idle Falcon State"

5.15.1 Detailed Description

The header file for the Idle class.

5.16 include/Iterator.h File Reference

The header file for the Iterator class.

`#include "CarryType.h"`
Include dependency graph for Iterator.h: This graph shows which files directly or indirectly include this file:

Classes

- class Iterator
  "The base class of the cargo- and crew-list iterators (Iterator-Iterator)"

5.16.1 Detailed Description

The header file for the Iterator class.

5.17 include/KeplerianCoords.h File Reference

The header file for the KeplerianCoords class.

`#include <string>`
Include dependency graph for KeplerianCoords.h: This graph shows which files directly or indirectly include this file:

Classes

- class KeplerianCoords
  "Class for encoding Keplerian Coordinates for Satellites"

5.17.1 Detailed Description

The header file for the KeplerianCoords class.

5.18 include/Launched.h File Reference

The header file for the Launched class.

```
#include "FalconState.h"
```
Include dependency graph for Launched.h:

Classes

- class Launched

  "Launched Falcon State"

5.18.1 Detailed Description

The header file for the Launched class.

5.19 include/LinkedListOfCrew.h File Reference

The header file for the LinkedListOfCrew class.

```
#include "../include/Collections.h"
#include "../include/Iterator.h"
#include "../include/Crew.h"
```
Include dependency graph for LinkedListOfCrew.h: This graph shows which files directly or indirectly include this file:

Classes

- class LinkedListOfCrew

  "A class to contain a linked list of Crew (ConcreteAggregate-Iterator)"

5.19.1 Detailed Description

The header file for the LinkedListOfCrew class.

5.20 include/Loader.h File Reference

The header file for the Loader class.

```
#include "Dragon.h"
```
Include dependency graph for Loader.h: This graph shows which files directly or indirectly include this file:

Classes

- class Loader

  "Context for loading dragon rocket"

5.20.1 Detailed Description

The header file for the Loader class.

5.21 include/Memento.h File Reference

The header file for the Memento class.

```
#include "State.h"
```

Include dependency graph for Memento.h: This graph shows which files directly or indirectly include this file:

Classes

- class Memento

  The Memento for the Memento Design Pattern.

5.21.1 Detailed Description

The header file for the Memento class.

5.22 include/MerlinCore.h File Reference

The header file for the MerlinCore class.

```
#include <vector>
#include "Engine.h"
```

Include dependency graph for MerlinCore.h: This graph shows which files directly or indirectly include this file:

Classes

- class MerlinCore

  The Core of engines used on Falcon rockets.

5.22.1 Detailed Description

The header file for the MerlinCore class.

5.23 include/MissionControl.h File Reference

The header file for the MissionControl class.

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

Include dependency graph for MissionControl.h: This graph shows which files directly or indirectly include this file:

Classes

- class MissionControl

  "MissionControl for controlling rockets and other spacecraft"

5.23.1 Detailed Description

The header file for the MissionControl class.

5.24 include/Returned.h File Reference

The header file for the Returned class.

```
#include "FalconState.h"
```

Include dependency graph for Returned.h:

Classes

- class Returned

  "Returned Falcon State"

5.24.1 Detailed Description

The header file for the Returned class.

5.25 include/Satelite.h File Reference

The header file for the Satelite class.

```
#include <string>
#include <thread>
#include <chrono>
#include <cmath>
#include <unistd.h>
#include <iostream>
#include <ctime>
#include "Spacecraft.h"
#include "KeplerianCoords.h"
#include "MissionControl.h"
```

Include dependency graph for Satelite.h: This graph shows which files directly or indirectly include this file:

Classes

- class Satelite

  "Starlink Satelite"

5.25.1 Detailed Description

The header file for the Satelite class.

5.26 include/SatelliteCreator.h File Reference

The header file for the SatelliteCreator class.

```
#include "SpacecraftCreator.h"
#include "Satelite.h"
```

Include dependency graph for SatelliteCreator.h:

Classes

- class SatelliteCreator

5.26.1 Detailed Description

The header file for the SatelliteCreator class.

5.27 include/SelectCommand.h File Reference

The header file for the SelectCommand class.

```
#include "Command.h"
#include "SelectSimulation.h"
```

Include dependency graph for SelectCommand.h:

Classes

- class SelectCommand

  The Concrete Command for the Command Design Pattern.

5.27.1 Detailed Description

The header file for the SelectCommand class.

5.28 include/Seperated.h File Reference

The header file for the Seperated class.

```
#include "FalconState.h"
```
Include dependency graph for Seperated.h:

Classes

- class Seperated

  "Seperated falcon state"

5.28.1 Detailed Description

The header file for the Seperated class.

5.29 include/Simulate.h File Reference

The header file for the Simulate class.

```
#include <iostream>
#include "Command.h"
#include "State.h"
```
Include dependency graph for Simulate.h:

Classes

- class Simulate

  The invoker for the Command Design Pattern.

5.29.1 Detailed Description

The header file for the Simulate class.

5.30 include/Spacecraft.h File Reference

The header file for the Spacecraft class.

```
#include <string>
```
Include dependency graph for Spacecraft.h: This graph shows which files directly or indirectly include this file:

Classes

- class Spacecraft

  "Interface for all spacecraft"

5.30.1 Detailed Description

The header file for the Spacecraft class.

5.31 include/SpacecraftCreator.h File Reference

The header file for the SpacecraftCreator class.

```
#include "Spacecraft.h"
```

Include dependency graph for SpacecraftCreator.h: This graph shows which files directly or indirectly include this file:

Classes

- class SpacecraftCreator

  "Interface for the creation of spacecraft"

5.31.1 Detailed Description

The header file for the SpacecraftCreator class.

5.32 include/State.h File Reference

The header file for the State class.

```
#include <string>
#include "Spacecraft.h"
#include "Cluster.h"
#include "CrewDragon.h"
#include "CargoDragon.h"
#include "Loader.h"
#include "Command.h"
```

Include dependency graph for State.h: This graph shows which files directly or indirectly include this file:

Classes

- class State

  The abstraction of the State of the simulation.

5.32.1 Detailed Description

The header file for the State class.

5.33 include/StateChangeCommand.h File Reference

The header file for the StateChangeCommand class.

```
#include "Command.h"
```
Include dependency graph for StateChangeCommand.h: This graph shows which files directly or indirectly include this file:

Classes

- class StateChangeCommand

  Concrete Command for the Command Design Pattern.

5.33.1 Detailed Description

The header file for the StateChangeCommand class.

5.34 include/Store.h File Reference

The header file for the Store class.

```
#include "Memento.h"
```
Include dependency graph for Store.h: This graph shows which files directly or indirectly include this file:

Classes

- class Store

  The Caretaker for the Memento Design Pattern.

5.34.1 Detailed Description

The header file for the Store class.

5.35 include/UnloadCommand.h File Reference

The header file for the UnloadCommand class.

```
#include "Command.h"
#include "Loader.h"
```
Include dependency graph for UnloadCommand.h: This graph shows which files directly or indirectly include this file:

Classes

- class UnloadCommand

  Concrete Command for the Command Design Pattern.

5.35.1 Detailed Description

The header file for the UnloadCommand class.

5.36 include/VectorOfCargo.h File Reference

The header file for the VectorOfCargo class.

```
#include "Collections.h"
#include "Iterator.h"
#include "Cargo.h"
#include <vector>
```

Include dependency graph for VectorOfCargo.h: This graph shows which files directly or indirectly include this file:

Classes

- class VectorOfCargo

  "A class to contain a vector of Cargo (ConcreteAggregate-Iterator)"

5.36.1 Detailed Description

The header file for the VectorOfCargo class.

Index