

COS214Project2021(SpaceXLaunchSimulator)

Generated by Doxygen 1.8.17

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 BuildCommand Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 BuildCommand()	7
4.1.3 Member Function Documentation	8
4.1.3.1 execute()	8
4.2 BuildSimulation Class Reference	8
4.2.1 Detailed Description	9
4.2.2 Member Function Documentation	9
4.2.2.1 buildCargo()	10
4.2.2.2 buildCrew()	10
4.2.2.3 buildMode()	10
4.2.2.4 buildSattelites()	10
4.2.2.5 buildTestMode()	11
4.2.2.6 exitProgram()	11
4.2.2.7 saveToFile()	11
4.2.2.8 startSim()	11
4.2.2.9 test9()	12
4.2.2.10 testCargo()	12
4.2.2.11 testCrew()	12
4.2.2.12 testCrewAndSatellite()	13
4.2.2.13 testHeavy()	13
4.2.2.14 testSatellites()	13
4.3 Cargo Class Reference	13
4.3.1 Detailed Description	14
4.3.2 Member Function Documentation	14
4.3.2.1 getType()	14
4.3.2.2 getWeight()	14
4.3.2.3 setType()	14
4.3.2.4 setWeight()	15
4.3.2.5 toString()	15
4.4 CargoDragon Class Reference	15
4.4.1 Detailed Description	16

4.4.2 Constructor & Destructor Documentation	16
4.4.2.1 CargoDragon()	16
4.4.3 Member Function Documentation	16
4.4.3.1 load()	16
4.4.3.2 unload()	17
4.5 CargoDragonCreator Class Reference	17
4.5.1 Member Function Documentation	17
4.5.1.1 createSpacecraft()	18
4.6 CargoIterator Class Reference	18
4.6.1 Detailed Description	18
4.6.2 Constructor & Destructor Documentation	18
4.6.2.1 CargoIterator()	18
4.6.3 Member Function Documentation	19
4.6.3.1 current()	19
4.6.3.2 first()	19
4.6.3.3 isDone()	19
4.6.3.4 next()	20
4.7 CarryType Class Reference	20
4.7.1 Detailed Description	20
4.7.2 Member Function Documentation	20
4.7.2.1 toString()	20
4.8 CheckEngineCommand Class Reference	21
4.8.1 Detailed Description	21
4.8.2 Constructor & Destructor Documentation	21
4.8.2.1 CheckEngineCommand()	21
4.8.3 Member Function Documentation	21
4.8.3.1 execute()	22
4.9 Cluster Class Reference	22
4.9.1 Detailed Description	22
4.9.2 Constructor & Destructor Documentation	22
4.9.2.1 Cluster()	22
4.9.3 Member Function Documentation	23
4.9.3.1 addSatellite()	23
4.9.3.2 checkCollisions()	23
4.9.3.3 generateSatellites()	23
4.9.3.4 getCraft()	24
4.9.3.5 getSize()	24
4.9.3.6 spreadOutSatellites()	24
4.10 Collections Class Reference	25
4.10.1 Detailed Description	25
4.11 Command Class Reference	25
4.11.1 Detailed Description	25

4.11.2 Member Function Documentation	25
4.11.2.1 add()	25
4.11.2.2 execute()	26
4.12 Crew Class Reference	26
4.12.1 Detailed Description	27
4.12.2 Constructor & Destructor Documentation	27
4.12.2.1 Crew()	27
4.12.3 Member Function Documentation	27
4.12.3.1 getJobTitle()	27
4.12.3.2 getName()	28
4.12.3.3 toString()	28
4.12.4 Member Data Documentation	28
4.12.4.1 next	28
4.13 CrewDragon Class Reference	28
4.13.1 Detailed Description	29
4.13.2 Constructor & Destructor Documentation	29
4.13.2.1 CrewDragon()	29
4.13.3 Member Function Documentation	29
4.13.3.1 load()	29
4.13.3.2 unload()	30
4.14 CrewDragonCreator Class Reference	30
4.14.1 Detailed Description	30
4.14.2 Member Function Documentation	31
4.14.2.1 createSpacecraft()	31
4.15 CrewIterator Class Reference	31
4.15.1 Detailed Description	31
4.15.2 Constructor & Destructor Documentation	32
4.15.2.1 CrewIterator()	32
4.15.3 Member Function Documentation	33
4.15.3.1 current()	33
4.15.3.2 first()	33
4.15.3.3 isDone()	33
4.15.3.4 next()	34
4.16 Dragon Class Reference	34
4.16.1 Detailed Description	34
4.16.2 Member Function Documentation	34
4.16.2.1 load()	35
4.16.2.2 unload()	35
4.17 Engine Class Reference	35
4.17.1 Constructor & Destructor Documentation	36
4.17.1.1 Engine()	36
4.17.2 Member Function Documentation	36

4.17.2.1 checkEngine()	36
4.17.2.2 checkOil()	36
4.17.2.3 clone()	37
4.17.2.4 getOn()	37
4.17.2.5 setOn()	37
4.17.2.6 startEngine()	37
4.17.2.7 turnOn()	38
4.18 Falcon Class Reference	38
4.18.1 Detailed Description	39
4.18.2 Constructor & Destructor Documentation	39
4.18.2.1 Falcon()	39
4.18.3 Member Function Documentation	39
4.18.3.1 change()	39
4.18.3.2 getCoreList()	39
4.18.3.3 getCurrentState()	40
4.18.3.4 getVacuumEngine()	40
4.18.3.5 setState()	40
4.19 Falcon9Creator Class Reference	40
4.19.1 Member Function Documentation	41
4.19.1.1 createSpacecraft()	41
4.20 FalconHeavyCreator Class Reference	41
4.20.1 Member Function Documentation	41
4.20.1.1 createSpacecraft()	42
4.21 FalconState Class Reference	42
4.21.1 Detailed Description	42
4.21.2 Member Function Documentation	42
4.21.2.1 getCurrentState()	42
4.21.2.2 handleChange()	43
4.22 Idle Class Reference	43
4.22.1 Detailed Description	44
4.22.2 Member Function Documentation	44
4.22.2.1 getCurrentState()	44
4.22.2.2 handleChange()	44
4.23 Iterator Class Reference	45
4.23.1 Detailed Description	45
4.23.2 Member Function Documentation	45
4.23.2.1 current()	45
4.23.2.2 first()	46
4.23.2.3 isDone()	46
4.23.2.4 next()	46
4.24 KeplerianCoords Class Reference	46
4.24.1 Detailed Description	47

4.24.2 Constructor & Destructor Documentation	47
4.24.2.1 KeplerianCoords()	47
4.24.3 Member Function Documentation	47
4.24.3.1 randomiseCoords()	47
4.24.3.2 toString()	48
4.25 Launched Class Reference	48
4.25.1 Detailed Description	48
4.25.2 Member Function Documentation	48
4.25.2.1 getCurrentState()	48
4.25.2.2 handleChange()	49
4.26 LinkedListOfCrew Class Reference	49
4.26.1 Detailed Description	50
4.26.2 Member Function Documentation	50
4.26.2.1 addCrewMember()	50
4.26.2.2 createIterator()	50
4.26.2.3 getHead()	51
4.26.2.4 getSize()	51
4.26.2.5 removeCrewMember()	51
4.27 Loader Class Reference	52
4.27.1 Detailed Description	52
4.27.2 Member Function Documentation	52
4.27.2.1 load()	52
4.27.2.2 unload()	52
4.28 Memento Class Reference	53
4.28.1 Detailed Description	53
4.28.2 Member Function Documentation	53
4.28.2.1 getState()	53
4.28.2.2 setState()	53
4.29 MerlinCore Class Reference	54
4.29.1 Detailed Description	54
4.29.2 Member Function Documentation	54
4.29.2.1 initiateEngineChecks()	54
4.29.2.2 off()	55
4.29.2.3 on()	56
4.30 MerlinEngine Class Reference	56
4.30.1 Detailed Description	57
4.30.2 Member Function Documentation	57
4.30.2.1 checkOil()	57
4.30.2.2 checkTemperature()	57
4.30.2.3 clone()	57
4.30.2.4 startEngine()	58
4.31 MerlinVacuumEngine Class Reference	58

4.31.1 Detailed Description	58
4.31.2 Member Function Documentation	58
4.31.2.1 checkOil()	59
4.31.2.2 checkTemperature()	59
4.31.2.3 clone()	59
4.31.2.4 startEngine()	59
4.32 MissionControl Class Reference	60
4.32.1 Detailed Description	60
4.32.2 Member Function Documentation	60
4.32.2.1 receiveRadioSignal()	60
4.32.2.2 sendRepositionRequest()	61
4.33 Returned Class Reference	61
4.33.1 Detailed Description	61
4.33.2 Member Function Documentation	61
4.33.2.1 getCurrentState()	61
4.33.2.2 handleChange()	62
4.34 Satellite Class Reference	62
4.34.1 Detailed Description	63
4.34.2 Constructor & Destructor Documentation	63
4.34.2.1 Satellite()	63
4.34.3 Member Function Documentation	63
4.34.3.1 clone()	64
4.34.3.2 getCoords()	64
4.34.3.3 positionSelf()	64
4.34.3.4 sendGroundSignal()	64
4.34.3.5 sendSatelliteSignal()	64
4.34.3.6 setMissionControl()	65
4.35 SatelliteCreator Class Reference	65
4.35.1 Member Function Documentation	65
4.35.1.1 createSpacecraft()	66
4.36 SelectCommand Class Reference	66
4.36.1 Detailed Description	66
4.36.2 Constructor & Destructor Documentation	66
4.36.2.1 SelectCommand()	66
4.36.3 Member Function Documentation	67
4.36.3.1 execute()	67
4.37 SelectSimulation Class Reference	67
4.37.1 Detailed Description	68
4.37.2 Member Function Documentation	68
4.37.2.1 exitProgram()	68
4.37.2.2 loadPrefabs()	68
4.37.2.3 simulateBatch()	68

4.37.2.4 simulateSingle()	69
4.37.2.5 startSim()	69
4.38 Separated Class Reference	70
4.38.1 Detailed Description	70
4.38.2 Member Function Documentation	70
4.38.2.1 getCurrentState()	70
4.38.2.2 handleChange()	71
4.39 Simulate Class Reference	71
4.39.1 Detailed Description	71
4.39.2 Member Function Documentation	72
4.39.2.1 build()	72
4.39.2.2 select()	72
4.39.2.3 setBuild()	72
4.39.2.4 setSelect()	72
4.40 Simulation Class Reference	73
4.40.1 Detailed Description	73
4.40.2 Member Function Documentation	74
4.40.2.1 createMemento()	74
4.40.2.2 getFilePath()	74
4.40.2.3 getState()	74
4.40.2.4 setMemento()	74
4.40.2.5 startSim()	75
4.41 Spacecraft Class Reference	75
4.41.1 Detailed Description	76
4.41.2 Constructor & Destructor Documentation	76
4.41.2.1 Spacecraft()	76
4.41.3 Member Function Documentation	76
4.41.3.1 getType()	76
4.42 SpacecraftCreator Class Reference	76
4.42.1 Detailed Description	77
4.42.2 Member Function Documentation	77
4.42.2.1 createSpacecraft()	77
4.43 spreadCommand Class Reference	77
4.43.1 Detailed Description	78
4.43.2 Constructor & Destructor Documentation	78
4.43.2.1 spreadCommand()	78
4.43.3 Member Function Documentation	78
4.43.3.1 execute()	78
4.44 State Class Reference	78
4.44.1 Detailed Description	79
4.44.2 Constructor & Destructor Documentation	79
4.44.2.1 State()	79

4.44.3 Member Function Documentation	80
4.44.3.1 addCommand()	80
4.44.3.2 getCluster()	80
4.44.3.3 getCommands()	80
4.44.3.4 getName()	81
4.44.3.5 getVessel()	81
4.44.3.6 remLastCommand()	81
4.44.3.7 runCommands()	81
4.44.3.8 setCluster()	81
4.44.3.9 setName()	82
4.44.3.10 setVessel()	82
4.45 StateChangeCommand Class Reference	83
4.45.1 Detailed Description	83
4.45.2 Constructor & Destructor Documentation	83
4.45.2.1 StateChangeCommand()	83
4.45.3 Member Function Documentation	83
4.45.3.1 execute()	83
4.46 Store Class Reference	84
4.46.1 Detailed Description	84
4.46.2 Member Function Documentation	84
4.46.2.1 returnMemento()	84
4.46.2.2 storeMemento()	84
4.47 UnloadCommand Class Reference	85
4.47.1 Detailed Description	85
4.47.2 Constructor & Destructor Documentation	85
4.47.2.1 UnloadCommand()	85
4.47.3 Member Function Documentation	86
4.47.3.1 execute()	86
4.48 VectorOfCargo Class Reference	86
4.48.1 Detailed Description	87
4.48.2 Member Function Documentation	87
4.48.2.1 addCargo()	87
4.48.2.2 removeCargo()	88
5 File Documentation	89
5.1 include/BuildCommand.h File Reference	89
5.2 include/CargoDragon.h File Reference	89
5.2.1 Detailed Description	89
5.3 include/CargoDragonCreator.h File Reference	90
5.3.1 Detailed Description	90
5.4 include/CargoIterator.h File Reference	90
5.4.1 Detailed Description	90

5.5 include/CarryType.h File Reference	90
5.5.1 Detailed Description	91
5.6 include/Cluster.h File Reference	91
5.6.1 Detailed Description	91
5.7 include/Command.h File Reference	91
5.7.1 Detailed Description	92
5.8 include/CrewDragonCreator.h File Reference	92
5.8.1 Detailed Description	92
5.9 include/CrewIterator.h File Reference	92
5.9.1 Detailed Description	92
5.10 include/Dragon.h File Reference	93
5.10.1 Detailed Description	93
5.11 include/Engine.h File Reference	93
5.11.1 Detailed Description	93
5.12 include/Falcon.h File Reference	93
5.12.1 Detailed Description	94
5.13 include/FalconHeavyCreator.h File Reference	94
5.13.1 Detailed Description	94
5.14 include/FalconState.h File Reference	94
5.14.1 Detailed Description	94
5.15 include/Idle.h File Reference	95
5.15.1 Detailed Description	95
5.16 include/Iterator.h File Reference	95
5.16.1 Detailed Description	95
5.17 include/KeplerianCoords.h File Reference	95
5.17.1 Detailed Description	96
5.18 include/Launched.h File Reference	96
5.18.1 Detailed Description	96
5.19 include/LinkedListOfCrew.h File Reference	96
5.19.1 Detailed Description	96
5.20 include/Loader.h File Reference	96
5.20.1 Detailed Description	97
5.21 include/Memento.h File Reference	97
5.21.1 Detailed Description	97
5.22 include/MerlinCore.h File Reference	97
5.22.1 Detailed Description	97
5.23 include/MissionControl.h File Reference	98
5.23.1 Detailed Description	98
5.24 include/Returned.h File Reference	98
5.24.1 Detailed Description	98
5.25 include/Satellite.h File Reference	98
5.25.1 Detailed Description	99

5.26 include/SatelliteCreator.h File Reference	99
5.26.1 Detailed Description	99
5.27 include/SelectCommand.h File Reference	99
5.27.1 Detailed Description	99
5.28 include/Seperated.h File Reference	100
5.28.1 Detailed Description	100
5.29 include/Simulate.h File Reference	100
5.29.1 Detailed Description	100
5.30 include/Spacecraft.h File Reference	100
5.30.1 Detailed Description	101
5.31 include/SpacecraftCreator.h File Reference	101
5.31.1 Detailed Description	101
5.32 include/State.h File Reference	101
5.32.1 Detailed Description	101
5.33 include/StateChangeCommand.h File Reference	102
5.33.1 Detailed Description	102
5.34 include/Store.h File Reference	102
5.34.1 Detailed Description	102
5.35 include/UnloadCommand.h File Reference	102
5.35.1 Detailed Description	103
5.36 include/VectorOfCargo.h File Reference	103
5.36.1 Detailed Description	103
Index	105

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

CarryType	20
Cargo	13
Crew	26
Cluster	22
Collections	25
LinkedListOfCrew	49
VectorOfCargo	86
Command	25
BuildCommand	7
CheckEngineCommand	21
SelectCommand	66
spreadCommand	77
StateChangeCommand	83
UnloadCommand	85
Engine	35
MerlinEngine	56
MerlinVacuumEngine	58
FalconState	42
Idle	43
Launched	48
Returned	61
Seperated	70
Iterator	45
Cargolterator	18
Crewlterator	31
KeplerianCoords	46
Loader	52
Memento	53
MerlinCore	54
MissionControl	60
Simulate	71
Simulation	73
BuildSimulation	8

SelectSimulation	67
Spacecraft	75
Dragon	34
CargoDragon	15
CrewDragon	28
Falcon	38
Satellite	62
SpacecraftCreator	76
CargoDragonCreator	17
CrewDragonCreator	30
Falcon9Creator	40
FalconHeavyCreator	41
SatelliteCreator	65
State	78
Store	84

Chapter 2

Class Index

2.1 Class List

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

BuildCommand	The Concrete Command for the Command Design Pattern and the Concrete Handler for the Chain of Responsibility	7
BuildSimulation	The implementation of the building process	8
Cargo	"A class for storing info about the Cargo that spacecraft will transport"	13
CargoDragon	"Dragon specialisation to carry cargo only"	15
CargoDragonCreator	17
CargoIterator	"The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"	18
CarryType	"A base class for Crew and Cargo"	20
CheckEngineCommand	Concrete Command for checking the engines	21
Cluster	"Represents a cluster of satellite"	22
Collections	"A base class for crew- and cargo-lists (Aggregate-Iterator)"	25
Command	The Command for the Command Design Pattern and the Handler for the Chain of Responsibility	25
Crew	"A class for storing info about the Crew members that will be boarding the spacecraft"	26
CrewDragon	"Specialisation for dragon to carry both cargo and crew"	28
CrewDragonCreator	"Creates a crew dragon object"	30
CrewIterator	"The base class of the cargo- and crew-list iterators (ConcreteIterator-Iterator)"	31
Dragon	"Interface for the dragon spacecraft"	34
Engine	35
Falcon	"Falcon Rocket"	38

Falcon9Creator	40
FalconHeavyCreator	41
FalconState	
"Interface for the current Falcon separation state"	42
Idle	
"Idle Falcon State"	43
Iterator	
"The base class of the cargo- and crew-list iterators (Iterator-Iterator)"	45
KeplerianCoords	
"Class for encoding Keplerian Coordinates for Satellites"	46
Launched	
"Launched Falcon State"	48
LinkedListOfCrew	
"A class to contain a linked list of Crew (ConcreteAggregate-Iterator)"	49
Loader	
"Context for loading dragon rocket"	52
Memento	
The Memento for the Memento Design Pattern	53
MerlinCore	
The Core of engines used on Falcon rockets	54
MerlinEngine	
""	56
MerlinVacuumEngine	
""	58
MissionControl	
"MissionControl for controlling rockets and other spacecraft"	60
Returned	
"Returned Falcon State"	61
Satellite	
"Starlink Satellite"	62
SatelliteCreator	65
SelectCommand	
The Concrete Command for the Command Design Pattern	66
SelectSimulation	
The implementation of the selection process	67
Seperated	
"Seperated falcon state"	70
Simulate	
The invoker for the Command Design Pattern	71
Simulation	
The interface for all Simulations, the Originator for the Memento Design Pattern and the Reciever for the Command Design Pattern	73
Spacecraft	
"Interface for all spacecraft"	75
SpacecraftCreator	
"Interface for the creation of spacecraft"	76
spreadCommand	
Command to spread out the satellites	77
State	
The abstraction of the State of the simulation	78
StateChangeCommand	
Concrete Command for the Command Design Pattern	83
Store	
The Caretaker for the Memento Design Pattern	84
UnloadCommand	
Concrete Command for the Command Design Pattern	85
VectorOfCargo	
"A class to contain a vector of Cargo (ConcreteAggregate-Iterator)"	86

Chapter 3

File Index

3.1 File List

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

include/ BuildCommand.h	
The header file for the BuildCommand class	89
include/ BuildSimulation.h	??
include/ Cargo.h	??
include/ CargoDragon.h	
The header file for the CargoDragon class	89
include/ CargoDragonCreator.h	
The header file for the CargoDragonCreator class	90
include/ CargoIterator.h	
The header file for the CargoIterator class	90
include/ CarryType.h	
"The header file for the base class CarryType"	90
include/ CheckEngineCommand.h	??
include/ Cluster.h	
The header file for the Cluster class	91
include/ Collections.h	??
include/ Command.h	
The header file for the Command class	91
include/ Crew.h	??
include/ CrewDragon.h	??
include/ CrewDragonCreator.h	
The header file for the CrewDragonCreator class	92
include/ CrewIterator.h	
The header file for the CrewIterator class	92
include/ Dragon.h	
The header file for the Dragon.h class	93
include/ Engine.h	
The header file for the Engine class	93
include/ Falcon.h	
The header file for the Falcon class	93
include/ Falcon9Creator.h	??
include/ FalconHeavyCreator.h	
The header file for the FalconHeavyCreator class	94
include/ FalconState.h	
The header file for the FalconState class	94

include/ Idle.h	
The header file for the Idle class	95
include/ Iterator.h	
The header file for the Iterator class	95
include/ KeplerianCoords.h	
The header file for the KeplerianCoords class	95
include/ Launched.h	
The header file for the Launched class	96
include/ LinkedListOfCrew.h	
The header file for the LinkedListOfCrew class	96
include/ Loader.h	
The header file for the Loader class	96
include/ Memento.h	
The header file for the Memento class	97
include/ MerlinCore.h	
The header file for the MerlinCore class	97
include/ MerlinEngine.h	??
include/ MerlinVacuumEngine.h	??
include/ MissionControl.h	
The header file for the MissionControl class	98
include/ Returned.h	
The header file for the Returned class	98
include/ Satellite.h	
The header file for the Satellite class	98
include/ SatelliteCreator.h	
The header file for the SatelliteCreator class	99
include/ SelectCommand.h	
The header file for the SelectCommand class	99
include/ SelectSimulation.h	??
include/ Seperated.h	
The header file for the Seperated class	100
include/ Simulate.h	
The header file for the Simulate class	100
include/ Simulation.h	??
include/ Spacecraft.h	
The header file for the Spacecraft class	100
include/ SpacecraftCreator.h	
The header file for the SpacecraftCreator class	101
include/ SpreadCommand.h	??
include/ State.h	
The header file for the State class	101
include/ StateChangeCommand.h	
The header file for the StateChangeCommand class	102
include/ Store.h	
The header file for the Store class	102
include/ UnloadCommand.h	
The header file for the UnloadCommand class	102
include/ VectorOfCargo.h	
The header file for the VectorOfCargo class	103

Chapter 4

Class Documentation

4.1 BuildCommand Class Reference

The Concrete [Command](#) for the [Command](#) Design Pattern and the Concrete 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 Concrete 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	<i>Falcon*</i>	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	<i>bool</i>	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 Cargolterator Class Reference

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

```
#include <CargoIterator.h>
```

Inheritance diagram for Cargolterator:

Collaboration diagram for Cargolterator:

Public Member Functions

- [~Cargolterator](#) ()
The destructor of the class.
- [Cargolterator](#) (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 (Concreteliterator-literator)"

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Cargolterator()

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

The accepted constructor of the class.

Parameters

<code>in</code>	<code>/list</code>	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	<i>v</i>	The vector of Merlin Cores to be used.
in	<i>mv</i>	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	<i>Falcon*</i>	The falcon used to to determine where the cluster is loaded.
----	----------------	--

4.9.3 Member Function Documentation

4.9.3.1 addSatellite()

```
Cluster::addSatellite (
    Satellite * )
```

Add a satellite to a cluster.

Parameters

in	<i>Satellite*</i>	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	<i>MissionControl*</i>	The mission control used to communicate with the satellites.
in	<i>int</i>	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 responsibility.
- 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	<i>c</i>	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 responsibility.

Parameters

in	<i>s</i>	String to determine which simulation to run based off the chain of responsibility.
in	<i>v</i>	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	<i>name</i>	The name of the crew member.
in	<i>jobTitle</i>	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	<i>Falcon*</i>	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	<i>bool</i>	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 (Concreteliterator-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 (Concreteliterator-Iterator)"

4.15.2 Constructor & Destructor Documentation

4.15.2.1 CrewIterator()

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

Constructor for this class.

Parameters

in	<i>LinkedListOfCrew*</i>	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	<i>MerlinCore*</i>	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	<i>bool</i>	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	<i>std::string</i>	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	<i>FalconState*</i>	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	<i>string</i>	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	<i>Falcon*</i>	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	<i>string</i>	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	<i>Falcon*</i>	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 [Cargoliterator](#), and [Crewliterator](#).

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 [Cargoliterator](#), and [Crewliterator](#).

4.23.2.4 next()

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

Gets the next element in the list/collection.

Returns

CarryType*

Reimplemented in [Cargoliterator](#), and [Crewliterator](#).

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	<i>int</i>	angle between equator and orbit plane
in	<i>semiMajorAxis</i>	
in	<i>trueAnomaly</i>	
in	<i>rightAscension</i>	
in	<i>argumentOfPerigree</i>	

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	<i>KeplerianCoords&</i>	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	<i>string</i>	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	<i>Falcon*</i>	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

<code>in</code>	<code>bool</code>	Indicate whether to print loading data
-----------------	-------------------	--

Returns

void

4.27.2.2 unload()

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


Performs unload action on member dragon.

Parameters

<code>in</code>	<code>bool</code>	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	<i>State*</i>	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	<i>Engine*</i>	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	<i>Engine*</i>	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	<i>int</i>	satellitID
in	<i>std::string</i>	Satellite Keplerian Coordinates

Returns

void

4.32.2.2 sendRepositionRequest()

```
MissionControl::sendRepositionRequest (
    Satellite * )
```

Send a reposition request to the referenced [Satelite](#).

Parameters

in	<i>Satellite*</i>	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	<i>string</i>	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	<i>Falcon*</i>	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 Satellite Class Reference

"Starlink Satellite"

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

Inheritance diagram for Satellite:

Collaboration diagram for Satellite:

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 Satellite"

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	<i>Satellite*</i>	A pointer to the satellite to check collision with.
----	-------------------	---

Returns

void

4.34.3.6 setMissionControl()

```
Satellite::setMissionControl (
    MissionControl * )
```

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

Parameters

in	<i>MissionControl*</i>	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	<i>sVector</i>	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	<i>sVector</i>	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	<i>sVector</i>	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	<i>string</i>	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

<code>in</code>	<code>m</code>	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

<code>in</code>	<code>sVector</code>	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

<code>in</code>	<code>c</code>	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	<i>s</i>	The State to copy.
in	<i>n</i>	The name of the simulation.
in	<i>s</i>	The spacecraft that will be used.
in	<i>n</i>	The name of the simulation.
in	<i>c</i>	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	<i>c</i>	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

<code>in</code>	<code>m</code>	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](#) *)
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](#) * [createliterator](#) ()
- [~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	<i>Cargo*</i>	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	<i>Cargo*</i>	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/Cargolterator.h File Reference

The header file for the [Cargolterator](#) class.

```
#include "CarryType.h"
```

```
#include "Iterator.h"
```

```
#include "Cargo.h"
```

```
#include <vector>
```

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

Classes

- class [Cargolterator](#)

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

5.4.1 Detailed Description

The header file for the [Cargolterator](#) 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

- add
 - Command, 25
- addCargo
 - VectorOfCargo, 87
- addCommand
 - State, 80
- addCrewMember
 - LinkedListOfCrew, 50
- addSatellite
 - Cluster, 23
- build
 - Simulate, 72
- buildCargo
 - BuildSimulation, 9
- BuildCommand, 7
 - BuildCommand, 7
 - execute, 8
- buildCrew
 - BuildSimulation, 10
- buildMode
 - BuildSimulation, 10
- buildSattelites
 - BuildSimulation, 10
- BuildSimulation, 8
 - buildCargo, 9
 - buildCrew, 10
 - buildMode, 10
 - buildSattelites, 10
 - buildTestMode, 10
 - exitProgram, 11
 - saveToFile, 11
 - startSim, 11
 - test9, 12
 - testCargo, 12
 - testCrew, 12
 - testCrewAndSatellite, 12
 - testHeavy, 13
 - testSatellites, 13
- buildTestMode
 - BuildSimulation, 10
- Cargo, 13
 - getType, 14
 - getWeight, 14
 - setType, 14
 - setWeight, 15
 - toString, 15
- CargoDragon, 15
 - CargoDragon, 16
 - load, 16
 - unload, 17
- CargoDragonCreator, 17
 - createSpacecraft, 17
- CargoIterator, 18
 - CargoIterator, 18
 - current, 19
 - first, 19
 - isDone, 19
 - next, 19
- CarryType, 20
 - toString, 20
- change
 - Falcon, 39
- checkCollisions
 - Cluster, 23
- checkEngine
 - Engine, 36
- CheckEngineCommand, 21
 - CheckEngineCommand, 21
 - execute, 21
- checkOil
 - Engine, 36
 - MerlinEngine, 57
 - MerlinVacuumEngine, 58
- checkTemperature
 - MerlinEngine, 57
 - MerlinVacuumEngine, 59
- clone
 - Engine, 36
 - MerlinEngine, 57
 - MerlinVacuumEngine, 59
 - Satelite, 63
- Cluster, 22
 - addSatellite, 23
 - checkCollisions, 23
 - Cluster, 22
 - generateSatellites, 23
 - getCraft, 24
 - getSize, 24
 - spreadOutSatellites, 24
- Collections, 25
- Command, 25
 - add, 25
 - execute, 26
- createIterator
 - LinkedListOfCrew, 50
- createMemento
 - Simulation, 74

- createSpacecraft
 - CargoDragonCreator, 17
 - CrewDragonCreator, 31
 - Falcon9Creator, 41
 - FalconHeavyCreator, 41
 - SatelliteCreator, 65
 - SpacecraftCreator, 77
- Crew, 26
 - Crew, 27
 - getJobTitle, 27
 - getName, 27
 - next, 28
 - toString, 28
- CrewDragon, 28
 - CrewDragon, 29
 - load, 29
 - unload, 30
- CrewDragonCreator, 30
 - createSpacecraft, 31
- CrewIterator, 31
 - CrewIterator, 32
 - current, 33
 - first, 33
 - isDone, 33
 - next, 33
- current
 - CargoIterator, 19
 - CrewIterator, 33
 - Iterator, 45
- Dragon, 34
 - load, 34
 - unload, 35
- Engine, 35
 - checkEngine, 36
 - checkOil, 36
 - clone, 36
 - Engine, 36
 - getOn, 37
 - setOn, 37
 - startEngine, 37
 - turnOn, 38
- execute
 - BuildCommand, 8
 - CheckEngineCommand, 21
 - Command, 26
 - SelectCommand, 67
 - spreadCommand, 78
 - StateChangeCommand, 83
 - UnloadCommand, 86
- exitProgram
 - BuildSimulation, 11
 - SelectSimulation, 68
- Falcon, 38
 - change, 39
 - Falcon, 39
 - getCoreList, 39
 - getCurrentState, 39
 - getVacuumEngine, 40
 - setState, 40
- Falcon9Creator, 40
 - createSpacecraft, 41
- FalconHeavyCreator, 41
 - createSpacecraft, 41
- FalconState, 42
 - getCurrentState, 42
 - handleChange, 43
- first
 - CargoIterator, 19
 - CrewIterator, 33
 - Iterator, 45
- generateSatellites
 - Cluster, 23
- getCluster
 - State, 80
- getCommands
 - State, 80
- getCoords
 - Satellite, 64
- getCoreList
 - Falcon, 39
- getCraft
 - Cluster, 24
- getCurrentState
 - Falcon, 39
 - FalconState, 42
 - Idle, 44
 - Launched, 48
 - Returned, 61
 - Seperated, 70
- getFilePath
 - Simulation, 74
- getHead
 - LinkedListOfCrew, 51
- getJobTitle
 - Crew, 27
- getName
 - Crew, 27
 - State, 80
- getOn
 - Engine, 37
- getSize
 - Cluster, 24
 - LinkedListOfCrew, 51
- getState
 - Memento, 53
 - Simulation, 74
- getType
 - Cargo, 14
 - Spacecraft, 76
- getVacuumEngine
 - Falcon, 40
- getVessel
 - State, 81
- getWeight

- Cargo, 14
- handleChange
 - FalconState, 43
 - Idle, 44
 - Launched, 49
 - Returned, 62
 - Seperated, 70
- Idle, 43
 - getCurrentState, 44
 - handleChange, 44
- include/BuildCommand.h, 89
- include/CargoDragon.h, 89
- include/CargoDragonCreator.h, 90
- include/CargoIterator.h, 90
- include/CarryType.h, 90
- include/Cluster.h, 91
- include/Command.h, 91
- include/CrewDragonCreator.h, 92
- include/CrewIterator.h, 92
- include/Dragon.h, 93
- include/Engine.h, 93
- include/Falcon.h, 93
- include/FalconHeavyCreator.h, 94
- include/FalconState.h, 94
- include/Idle.h, 95
- include/Iterator.h, 95
- include/KeplerianCoords.h, 95
- include/Launched.h, 96
- include/LinkedListOfCrew.h, 96
- include/Loader.h, 96
- include/Memento.h, 97
- include/MerlinCore.h, 97
- include/MissionControl.h, 98
- include/Returned.h, 98
- include/Satelite.h, 98
- include/SatelliteCreator.h, 99
- include/SelectCommand.h, 99
- include/Seperated.h, 100
- include/Simulate.h, 100
- include/Spacecraft.h, 100
- include/SpacecraftCreator.h, 101
- include/State.h, 101
- include/StateChangeCommand.h, 102
- include/Store.h, 102
- include/UnloadCommand.h, 102
- include/VectorOfCargo.h, 103
- initiateEngineChecks
 - MerlinCore, 54
- isDone
 - CargoIterator, 19
 - CrewIterator, 33
 - Iterator, 46
- Iterator, 45
 - current, 45
 - first, 45
 - isDone, 46
 - next, 46
- KeplerianCoords, 46
 - KeplerianCoords, 47
 - randomiseCoords, 47
 - toString, 48
- Launched, 48
 - getCurrentState, 48
 - handleChange, 49
- LinkedListOfCrew, 49
 - addCrewMember, 50
 - createIterator, 50
 - getHead, 51
 - getSize, 51
 - removeCrewMember, 51
- load
 - CargoDragon, 16
 - CrewDragon, 29
 - Dragon, 34
 - Loader, 52
- Loader, 52
 - load, 52
 - unload, 52
- loadPrefabs
 - SelectSimulation, 68
- Memento, 53
 - getState, 53
 - setState, 53
- MerlinCore, 54
 - initiateEngineChecks, 54
 - off, 54
 - on, 56
- MerlinEngine, 56
 - checkOil, 57
 - checkTemperature, 57
 - clone, 57
 - startEngine, 57
- MerlinVacuumEngine, 58
 - checkOil, 58
 - checkTemperature, 59
 - clone, 59
 - startEngine, 59
- MissionControl, 60
 - receiveRadioSignal, 60
 - sendRepositionRequest, 60
- next
 - CargoIterator, 19
 - Crew, 28
 - CrewIterator, 33
 - Iterator, 46
- off
 - MerlinCore, 54
- on
 - MerlinCore, 56
- positionSelf
 - Satelite, 64

- randomiseCoords
 - KeplerianCoords, 47
- receiveRadioSignal
 - MissionControl, 60
- remLastCommand
 - State, 81
- removeCargo
 - VectorOfCargo, 88
- removeCrewMember
 - LinkedListOfCrew, 51
- Returned, 61
 - getCurrentState, 61
 - handleChange, 62
- returnMemento
 - Store, 84
- runCommands
 - State, 81
- Satellite, 62
 - clone, 63
 - getCoords, 64
 - positionSelf, 64
 - Satellite, 63
 - sendGroundSignal, 64
 - sendSatelliteSignal, 64
 - setMissionControl, 65
- SatelliteCreator, 65
 - createSpacecraft, 65
- saveToFile
 - BuildSimulation, 11
- select
 - Simulate, 72
- SelectCommand, 66
 - execute, 67
 - SelectCommand, 66
- SelectSimulation, 67
 - exitProgram, 68
 - loadPrefabs, 68
 - simulateBatch, 68
 - simulateSingle, 69
 - startSim, 69
- sendGroundSignal
 - Satellite, 64
- sendRepositionRequest
 - MissionControl, 60
- sendSatelliteSignal
 - Satellite, 64
- Seperated, 70
 - getCurrentState, 70
 - handleChange, 70
- setBuild
 - Simulate, 72
- setCluster
 - State, 81
- setMemento
 - Simulation, 74
- setMissionControl
 - Satellite, 65
- setName
 - State, 82
- setOn
 - Engine, 37
- setSelect
 - Simulate, 72
- setState
 - Falcon, 40
 - Memento, 53
- setType
 - Cargo, 14
- setVessel
 - State, 82
- setWeight
 - Cargo, 15
- Simulate, 71
 - build, 72
 - select, 72
 - setBuild, 72
 - setSelect, 72
- simulateBatch
 - SelectSimulation, 68
- simulateSingle
 - SelectSimulation, 69
- Simulation, 73
 - createMemento, 74
 - getFilePath, 74
 - getState, 74
 - setMemento, 74
 - startSim, 75
- Spacecraft, 75
 - getType, 76
 - Spacecraft, 76
- SpacecraftCreator, 76
 - createSpacecraft, 77
- spreadCommand, 77
 - execute, 78
 - spreadCommand, 78
- spreadOutSatellites
 - Cluster, 24
- startEngine
 - Engine, 37
 - MerlinEngine, 57
 - MerlinVacuumEngine, 59
- startSim
 - BuildSimulation, 11
 - SelectSimulation, 69
 - Simulation, 75
- State, 78
 - addCommand, 80
 - getCluster, 80
 - getCommands, 80
 - getName, 80
 - getVessel, 81
 - remLastCommand, 81
 - runCommands, 81
 - setCluster, 81
 - setName, 82
 - setVessel, 82

- State, [79](#)
- StateChangeCommand, [83](#)
 - execute, [83](#)
 - StateChangeCommand, [83](#)
- Store, [84](#)
 - returnMemento, [84](#)
 - storeMemento, [84](#)
- storeMemento
 - Store, [84](#)
- test9
 - BuildSimulation, [12](#)
- testCargo
 - BuildSimulation, [12](#)
- testCrew
 - BuildSimulation, [12](#)
- testCrewAndSatellite
 - BuildSimulation, [12](#)
- testHeavy
 - BuildSimulation, [13](#)
- testSatellites
 - BuildSimulation, [13](#)
- toString
 - Cargo, [15](#)
 - CarryType, [20](#)
 - Crew, [28](#)
 - KeplerianCoords, [48](#)
- turnOn
 - Engine, [38](#)
- unload
 - CargoDragon, [17](#)
 - CrewDragon, [30](#)
 - Dragon, [35](#)
 - Loader, [52](#)
- UnloadCommand, [85](#)
 - execute, [86](#)
 - UnloadCommand, [85](#)
- VectorOfCargo, [86](#)
 - addCargo, [87](#)
 - removeCargo, [88](#)