COS214 Project

Generated by Doxygen 1.9.2

Chapter 1

COS214 Project

Authors

The 6 Musketeers

1.1 COS214 Group Project

This is the documentation for the COS214 Project 2021.

1.2 Group Members

u20632429 - Chiara Goncalves u20444738 - Zoe Liebenberg u20438151 - Jade Peche u17030553 - Ben Pietersen u20498510 - Dylan Pietersen u20430516 - Steven Schormann

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

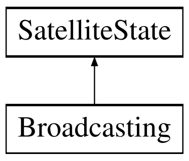
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Broadcasting Class Reference

Inheritance diagram for Broadcasting:



Public Member Functions

- **Broadcasting** ()

    Constructor for the Broadcasting object.
- string getType ()

    Returns the type of state the satellite is currently in (Broadcasting).
- SatelliteState ∗ handleChange ()

    Handles a change in state - sets the current state of the satellite to offline.

5.1.1 Member Function Documentation

5.1.1.1 getType()

```
string Broadcasting::getType ( )  [virtual]
```
Returns the type of state the satellite is currently in (Broadcasting).

Returns

string

Implements SatelliteState.

5.1.1.2 handleChange()

```
SatelliteState * Broadcasting::handleChange ( )  [virtual]
```
Handles a change in state - sets the current state of the satellite to offline.

Returns

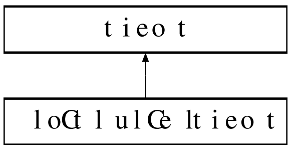
SatelliteState∗

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

- C:/Users/labuser2/Downloads/System/System/Broadcasting.h
- C:/Users/labuser2/Downloads/System/System/Broadcasting.cpp

5.2 Builder Class Reference

Inheritance diagram for Builder:



Public Member Functions

- virtual void buildFalcon9 ()=0

    Pure virtual function to be implemented in children classes. The function builds a Falcon9 Core for the rocket.

- virtual void buildFalconHeavy ()=0

    Pure virtual function to be implemented in children classes. The function builds a FalconHeavy Core for the rocket.
- virtual void constructCapsule (string c)=0

    Pure virtual function to be implemented in children classes. The function constructs a capsule for the rocket.
- virtual Component ∗ getSpacecraft ()=0

    Pure virtual function to be implemented in children classes. The function returns the current spacecraft.
- virtual Simulation ∗ createSimulation ()=0

    Pure virtual function to be implemented in children classes. The function creates a simulation for the rocket.

5.2.1 Member Function Documentation

5.2.1.1 buildFalcon9()

```
virtual void Builder::buildFalcon9 ( ) [pure virtual]
```

Pure virtual function to be implemented in children classes. The function builds a Falcon9 Core for the rocket.

Returns

void

Implemented in ConcreteRocketBuilder.

5.2.1.2 buildFalconHeavy()

```
virtual void Builder::buildFalconHeavy ( ) [pure virtual]
```

Pure virtual function to be implemented in children classes. The function builds a FalconHeavy Core for the rocket.

Returns

void

Implemented in ConcreteRocketBuilder.

5.2.1.3 constructCapsule()

```
virtual void Builder::constructCapsule (
             string c )  [pure virtual]
```
Pure virtual function to be implemented in children classes. The function constructs a capsule for the rocket.

Returns

void

Implemented in ConcreteRocketBuilder.

5.2.1.4 createSimulation()

```
virtual Simulation * Builder::createSimulation ( )  [pure virtual]
```
Pure virtual function to be implemented in children classes. The function creates a simulation for the rocket.

Returns

Simulation∗

Implemented in ConcreteRocketBuilder.

5.2.1.5 getSpacecraft()

```
virtual Component * Builder::getSpacecraft ( )  [pure virtual]
```
Pure virtual function to be implemented in children classes. The function returns the current spacecraft.

Returns

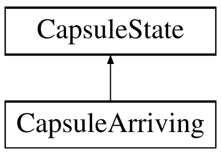
Component∗

Implemented in ConcreteRocketBuilder.
The documentation for this class was generated from the following file:

- C:/Users/labuser2/Downloads/System/System/Builder.h

5.3 CapsuleArriving Class Reference

Inheritance diagram for CapsuleArriving:



Public Member Functions

- **CapsuleArriving** ()

  Constructor for CapsuleArriving objects.
- string getState ()

  Returns the state that the capsule is currently in (Arriving).
- CapsuleState ∗ handleChange ()

  Handles a change in state - sets the state of the current capsule to 'docked'.

5.3.1 Member Function Documentation

5.3.1.1 getState()

```
string CapsuleArriving::getState ( )  [virtual]
```

Returns the state that the capsule is currently in (Arriving).

Returns

string

Implements CapsuleState.

5.3.1.2 handleChange()

```
CapsuleState * CapsuleArriving::handleChange ( )  [virtual]
```

Handles a change in state - sets the state of the current capsule to 'docked'.

Returns

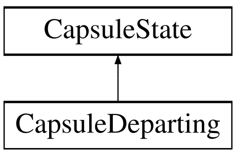
CapsuleState∗

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

- C:/Users/labuser2/Downloads/System/System/CapsuleArriving.h
- C:/Users/labuser2/Downloads/System/System/CapsuleArriving.cpp

5.4 CapsuleDeparting Class Reference

Inheritance diagram for CapsuleDeparting:



Public Member Functions

- **CapsuleDeparting** ()

    Constructor for CapsuleDeparting objects.
- string getState ()

    Returns the state that the capsule is currently in (Departing).
- CapsuleState ∗ handleChange ()

    Handles a change in state - sets the state of the current capsule to 'arriving'.

5.4.1 Member Function Documentation

5.4.1.1 getState()

```
string CapsuleDeparting::getState ( )  [virtual]
```

Returns the state that the capsule is currently in (Departing).

Returns

string

Implements CapsuleState.

5.4.1.2 handleChange()

```
CapsuleState * CapsuleDeparting::handleChange ( )  [virtual]
```
Handles a change in state - sets the state of the current capsule to 'arriving'.

Returns

CapsuleState∗

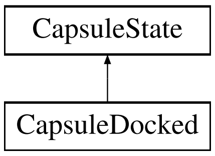
Implements CapsuleState.

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

- C:/Users/labuser2/Downloads/System/System/CapsuleDeparting.h
- C:/Users/labuser2/Downloads/System/System/CapsuleDeparting.cpp

5.5 CapsuleDocked Class Reference

Inheritance diagram for CapsuleDocked:



Public Member Functions

- **CapsuleDocked** ()

    Constructor for CapsuleDocked objects.
- string getState ()

    Returns the state that the capsule is currently in (Docked).
- CapsuleState ∗ handleChange ()

    Handles a change in state - sets the state of the current capsule to 'offline'.

5.5.1 Member Function Documentation

5.5.1.1 getState()

```
string CapsuleDocked::getState ( )  [virtual]
```
Returns the state that the capsule is currently in (Docked).

Returns

string

Implements CapsuleState.

5.5.1.2 handleChange()

```
CapsuleState * CapsuleDocked::handleChange ( )  [virtual]
```
Handles a change in state - sets the state of the current capsule to 'offline'.

Returns

CapsuleState∗

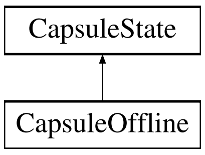
Implements CapsuleState.

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

- C:/Users/labuser2/Downloads/System/System/CapsuleDocked.h
- C:/Users/labuser2/Downloads/System/System/CapsuleDocked.cpp

5.6 CapsuleOffline Class Reference

Inheritance diagram for CapsuleOffline:



Public Member Functions

- **CapsuleOffline** ()

    Constructor for CapsuleOffline objects.
- string getState ()

    Returns the state that the capsule is currently in (Offline).
- CapsuleState ∗ handleChange ()

    Handles a change in state - sets the state of the current capsule to 'null'.

5.6.1 Member Function Documentation

5.6.1.1 getState()

```
string CapsuleOffline::getState ( )  [virtual]
```

Returns the state that the capsule is currently in (Offline).

Returns

string

Implements CapsuleState.

5.6.1.2 handleChange()

```
CapsuleState * CapsuleOffline::handleChange ( )  [virtual]
```

Handles a change in state - sets the state of the current capsule to 'null'.

Returns

void

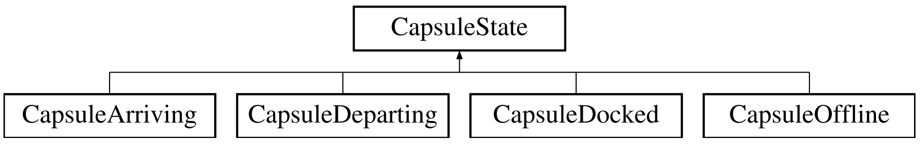
Implements CapsuleState.

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

- C:/Users/labuser2/Downloads/System/System/CapsuleOffline.h
- C:/Users/labuser2/Downloads/System/System/CapsuleOffline.cpp

5.7 CapsuleState Class Reference

Inheritance diagram for CapsuleState:



Public Member Functions

- virtual string getState ()=0

  Pure virtual function to be implemented in children classes. The function returns the current state of the capsule (Arriving/Departing/Docked/Offline).

- virtual CapsuleState ∗ handleChange ()=0

  Pure virtual function to be implemented in children classes. Handles the change in state by setting the state of the capsule to a new state.

5.7.1 Member Function Documentation

5.7.1.1 getState()

```
virtual string CapsuleState::getState ( )  [pure virtual]
```
Pure virtual function to be implemented in children classes. The function returns the current state of the capsule (Arriving/Departing/Docked/Offline).

Returns

string

Implemented in CapsuleArriving, CapsuleDeparting, CapsuleDocked, and CapsuleOffline.

5.7.1.2 handleChange()

```
virtual CapsuleState * CapsuleState::handleChange ( )  [pure virtual]
```
Pure virtual function to be implemented in children classes. Handles the change in state by setting the state of the capsule to a new state.

Returns

CapsuleState∗

Implemented in CapsuleArriving, CapsuleDeparting, CapsuleDocked, and CapsuleOffline.
The documentation for this class was generated from the following file:

- C:/Users/labuser2/Downloads/System/System/CapsuleState.h

5.8 Caretaker Class Reference

Public Member Functions

- **Caretaker** ()

  Constructor for Caretaker objects. Sets the store to NULL.
- ∼**Caretaker** ()

  Destructor for Caretaker objects. Deletes the store object and the memory allocated to it.
- void storeMemento (Memento ∗m)

  Stores the current state of the rocket.
- Memento ∗ retrieveMemento ()

  Returns the current state of the rocket .
- int getSize ()

  Returns the size of the store.

5.8.1 Member Function Documentation

5.8.1.1 getSize()

```
int Caretaker::getSize ( )
```

Returns the size of the store.

Returns

int

5.8.1.2 retrieveMemento()

```
Memento * Caretaker::retrieveMemento ( )
```

Returns the current state of the rocket .

Returns

Memento∗

5.8.1.3 storeMemento()

```
void Caretaker::storeMemento (
            Memento * m )
```

Stores the current state of the rocket.

Parameters

m	Memento∗ - the Memento storing the current state of the rocket.

Returns

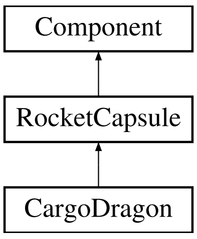
void

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

- C:/Users/labuser2/Downloads/System/System/Caretaker.h
- C:/Users/labuser2/Downloads/System/System/Caretaker.cpp

5.9 CargoDragon Class Reference

Inheritance diagram for CargoDragon:



Public Member Functions

- CargoDragon (Component ∗r)

Constructor for CargoDragon objects. Takes in a rocket as a parameter and uses the RocketCapsule(parent) constructor to initialize the rocket variable.

- void simulate ()

  Starts the simulation for CargoDragon objects.

- void test ()

  Tests if the CargoDragon meets all the requirements for a successful launch. The requirements:

Additional Inherited Members

5.9.1 Constructor & Destructor Documentation

5.9.1.1 CargoDragon()

```
CargoDragon::CargoDragon (
             Component * r )
```

Constructor for CargoDragon objects. Takes in a rocket as a parameter and uses the RocketCapsule(parent) constructor to initialize the rocket variable.

Parameters

r	Component*

5.9.2 Member Function Documentation

5.9.2.1 simulate()

```
void CargoDragon::simulate ( )  [virtual]
```

Starts the simulation for CargoDragon objects.

Returns

void

Implements RocketCapsule.

5.9.2.2 test()

```
void CargoDragon::test ( )  [virtual]
```

Tests if the CargoDragon meets all the requirements for a successful launch. The requirements:

- the cost must be >0
- it must have a capsuleType
- it must have a rocketType

  Returns

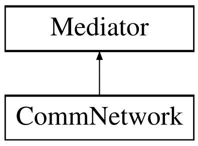
  void

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

- C:/Users/labuser2/Downloads/System/System/CargoDragon.h
- C:/Users/labuser2/Downloads/System/System/CargoDragon.cpp

5.10 CommNetwork Class Reference

Inheritance diagram for CommNetwork:



Public Member Functions

- **CommNetwork** (vector< Satellite ∗ >)
- void notify (int sender)

  Notifies all the satellites (colleagues) if the state of the one of the satellites have changed.
- void sendMessage (int sender, int receiver, string msg)

  Sends a string message to a particular satellite.

Public Attributes

- vector< Satellite ∗ > colleagueList

5.10.1 Member Function Documentation

5.10.1.1 notify()

```
void CommNetwork::notify (
             int sender )  [virtual]
```

Notifies all the satellites (colleagues) if the state of the one of the satellites have changed.

Parameters

colleague	Satellite∗ - the Satellite object that changed states.

Returns

void

Implements Mediator.

5.10.1.2 sendMessage()

```
void CommNetwork::sendMessage (
             int sender,
             int receiver,
             string msg )  [virtual]
```

Sends a string message to a particular satellite.

Parameters

sender	int - The ID of the sender.
receiver	int - The ID of the receiver.
msg	string - The message to send.

Returns

void

Implements Mediator.

5.10.2 Member Data Documentation

5.10.2.1 colleagueList

`vector<Satellite*> CommNetwork::colleagueList`

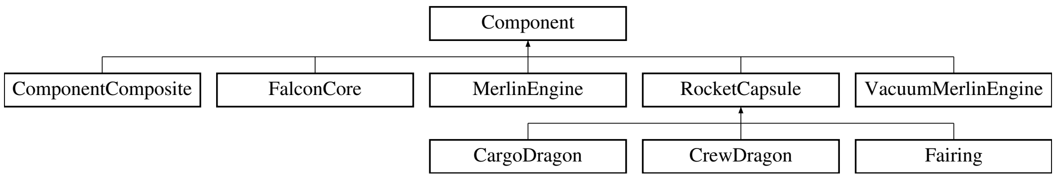
A vector of Satellite objects representing the colleagues of the Mediator pattern

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

- C:/Users/labuser2/Downloads/System/System/CommNetwork.h
- C:/Users/labuser2/Downloads/System/System/CommNetwork.cpp

5.11 Component Class Reference

Inheritance diagram for Component:



Public Member Functions

- Component (double c)

  Constructor for Component objects. Takes in the cost as a parameter and initializes the cost variable.
- virtual void simulate ()

  Virtual function that needs to be implemented in all the children classes. Starts the simulation for Component objects.
- virtual void test ()

  Virtual function that tests if the Component meets all the requirements for a successful launch. The requirements depend on the type of Component.
- virtual void add (Component ∗c)

  Adds a component to the rocket.
- virtual void remove (int pos)

  Virtual method that removes a component from the rocket based on its position.
- virtual Component ∗ getComponent (int pos)

  Virtual method that returns a component of the rocket based on its position.
- double getCost ()

  Returns the cost of the component.
- virtual void separate ()

  Virtual function that seperates the Component from the rocket.
- virtual void fireMerlin ()

  Virtual method to ignite a MerlinEngine object.
- virtual void land ()

  Virtual method called when a Component object lands.
- virtual int getSize ()

  Virtual method that returns the size of the Component. return @int.
- virtual void fireVacuumMerlin ()

  Virtual method to ignite a VacuumMerlin object.

Protected Attributes

- double cost

5.11.1 Constructor & Destructor Documentation

5.11.1.1 Component()

```
Component::Component (
             double c )
```

Constructor for Component objects. Takes in the cost as a parameter and initializes the cost variable.

Parameters

c	double - the cost of the compoenent.

5.11.2 Member Function Documentation

5.11.2.1 add()

```
void Component::add (
             Component * c )  [virtual]
```

Adds a component to the rocket.

Parameters

c	Component∗ - the Component to add to the rocket.

Returns

void

Reimplemented in ComponentComposite.

5.11.2.2 fireMerlin()

```
void Component::fireMerlin ( )  [virtual]
```

Virtual method to ignite a MerlinEngine object.

Returns

void

Reimplemented in ComponentComposite, and MerlinEngine.

5.11.2.3 fireVacuumMerlin()

```
void Component::fireVacuumMerlin ( )  [virtual]
```

Virtual method to ignite a VacuumMerlin object.

Returns

void

Reimplemented in VacuumMerlinEngine.

5.11.2.4 getComponent()

```
Component * Component::getComponent (
             int pos )  [virtual]
```

Virtual method that returns a component of the rocket based on its position.

Parameters

pos	int - the position of the Component in the vector of components.

Returns

Component∗

Reimplemented in ComponentComposite.

5.11.2.5 getCost()

```
double Component::getCost ( )
```

Returns the cost of the component.

Returns

double

5.11.2.6 getSize()

```
int Component::getSize ( )  [virtual]
```

Virtual method that returns the size of the Component. return @int.
Reimplemented in ComponentComposite.

5.11.2.7 land()

```
void Component::land ( )  [virtual]
```

Virtual method called when a Component object lands.

Returns

void

Reimplemented in ComponentComposite, and FalconCore.

5.11.2.8 remove()

```
void Component::remove (
             int pos )  [virtual]
```

Virtual method that removes a component from the rocket based on its position.

Parameters

pos	int - the position of the Component in the vector of components.

Returns

void

Reimplemented in ComponentComposite.

5.11.2.9 separate()

`void Component::separate ( ) [virtual]`

Virtual function that seperates the Component from the rocket.

Returns

void

Reimplemented in ComponentComposite, and FalconCore.

5.11.2.10 simulate()

`void Component::simulate ( ) [virtual]`

Virtual function that needs to be implemented in all the children classes. Starts the simulation for Component objects.

Returns

void

Reimplemented in CargoDragon, ComponentComposite, CrewDragon, Fairing, FalconCore, MerlinEngine, VacuumMerlinEngine, and RocketCapsule.

5.11.2.11 test()

`void Component::test ( ) [virtual]`

Virtual function that tests if the Component meets all the requirements for a successful launch. The requirements depend on the type of Component.

Returns

void

Reimplemented in CargoDragon, ComponentComposite, CrewDragon, Fairing, FalconCore, MerlinEngine, VacuumMerlinEngine, and RocketCapsule.

5.11.3 Member Data Documentation

5.11.3.1 cost

`double Component::cost [protected]`

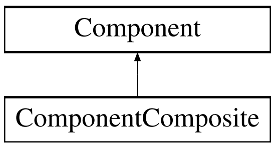
The cost of the component

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

- C:/Users/labuser2/Downloads/System/System/Component.h
- C:/Users/labuser2/Downloads/System/System/Component.cpp

5.12 ComponentComposite Class Reference

Inheritance diagram for ComponentComposite:



Public Member Functions

- **ComponentComposite** ()

  Constructor for ComponentComposite objects.
- void simulate ()

  Starts the simulation for ComponentComposite objects.
- void test ()

  Tests if the ComponentComposite meets all the requirements for a successful launch. The requirements depend on the type of Component.
- virtual void add (Component ∗c)

  Virtual function that adds a component to the rocket.
- virtual void remove (int pos)

  Virtual function the removes a component from the rocket based on its position.
- Component ∗ getComponent (int pos)

  Returns a component of the rocket based on its position.
- int getSize ()

  Returns the size of the ComponentComposite object.
- void separate ()

  Seperates the ComponentComposite from the rocket.
- void fireMerlin ()

  Ignites the MerlinEngine object.
- void land ()

  Lands the rocket.

Additional Inherited Members

5.12.1 Member Function Documentation

5.12.1.1 add()

```
void ComponentComposite::add (
            Component * c )  [virtual]
```

Virtual function that adds a component to the rocket.

Parameters

c	Component∗ - the Component to be added to the rocket.

Returns

void

Reimplemented from Component.

5.12.1.2 fireMerlin()

```
void ComponentComposite::fireMerlin ( )  [virtual]
```
Ignites the MerlinEngine object.

Returns

void

Reimplemented from Component.

5.12.1.3 getComponent()

```
Component * ComponentComposite::getComponent (
             int pos )  [virtual]
```
Returns a component of the rocket based on its position.

Parameters

pos	int - the position of the Componenet in the vector of components.

Returns

Component∗

Reimplemented from Component.

5.12.1.4 getSize()

```
int ComponentComposite::getSize ( )  [virtual]
```
Returns the size of the ComponentComposite object.

Returns

int

Reimplemented from Component.

5.12.1.5 land()

```
void ComponentComposite::land ( )  [virtual]
```
Lands the rocket.

Returns

void

Reimplemented from Component.

5.12.1.6 remove()

```
void ComponentComposite::remove (
             int pos )  [virtual]
```
Virtual function the removes a component from the rocket based on its position.

Parameters

pos	int - the position of the Componenet in the vector of components.

Returns

void

Reimplemented from Component.

5.12.1.7 separate()

```
void ComponentComposite::separate ( )  [virtual]
```

Seperates the ComponentComposite from the rocket.

Returns

void

Reimplemented from Component.

5.12.1.8 simulate()

```
void ComponentComposite::simulate ( )  [virtual]
```

Starts the simulation for ComponentComposite objects.

Returns

void

Reimplemented from Component.

5.12.1.9 test()

```
void ComponentComposite::test ( )  [virtual]
```

Tests if the ComponentComposite meets all the requirements for a successful launch. The requirements depend on the type of Component.

Returns

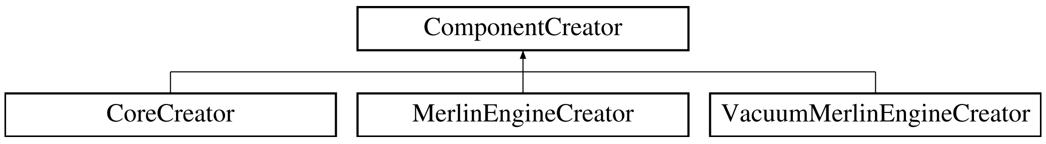
void

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

- C:/Users/labuser2/Downloads/System/System/ComponentComposite.h
- C:/Users/labuser2/Downloads/System/System/ComponentComposite.cpp

5.13 ComponentCreator Class Reference

Inheritance diagram for ComponentCreator:



Public Member Functions

- virtual Component ∗ factoryMethod ()=0

  Virtual functon to be implemented in all the children classes. Factory method to create different types of engines.

5.13.1 Member Function Documentation

5.13.1.1 factoryMethod()

```
virtual Component * ComponentCreator::factoryMethod ( )  [pure virtual]
```
Virtual functon to be implemented in all the children classes. Factory method to create different types of engines.

Returns

Component∗

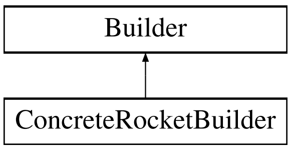
Implemented in CoreCreator, MerlinEngineCreator, and VacuumMerlinEngineCreator.

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

- C:/Users/labuser2/Downloads/System/System/ComponentCreator.h

5.14 ConcreteRocketBuilder Class Reference

Inheritance diagram for ConcreteRocketBuilder:



Public Member Functions

- **ConcreteRocketBuilder** ()

  Constructor for ConcreteRocketBuilder objects.
- Component ∗ getSpacecraft ()

  Returns the current spacecraft/rocket.
- RocketCapsule ∗ getCapsule ()

  Returns the current RocketCapsule object.
- void buildFalcon9 ()

  Builds the Falcon9 rocket. Adds the FalconCore, MerlinEngine and VacuumMerlinEngine components.
- void buildFalconHeavy ()

  Builds the FalconHeavy rocket. Adds the FalconCore, MerlinEngine and VacuumMerlinEngine components.
- void constructCapsule (string type)

  Creates a new capsule. The capsule can be either a CrewDragon, CargoDragon or Fairing. The capsule is constructed differently based on the capsule type.
- Simulation ∗ createSimulation ()

  Creates a new simulation based on the capsule, rocket and simulation state.

5.14.1 Member Function Documentation

5.14.1.1 buildFalcon9()

```
void ConcreteRocketBuilder::buildFalcon9 ( )  [virtual]
```
Builds the Falcon9 rocket. Adds the FalconCore, MerlinEngine and VacuumMerlinEngine components.

Returns

void

Implements Builder.

5.14.1.2 buildFalconHeavy()

```
void ConcreteRocketBuilder::buildFalconHeavy ( )  [virtual]
```

Builds the FalconHeavy rocket. Adds the FalconCore, MerlinEngine and VacuumMerlinEngine components.

Returns

void

Implements Builder.

5.14.1.3 constructCapsule()

```
void ConcreteRocketBuilder::constructCapsule (
             string type )  [virtual]
```

Creates a new capsule. The capsule can be either a CrewDragon, CargoDragon or Fairing. The capsule is constructed differently based on the capsule type.

Parameters

type	string - the type of capsule.

Returns

void

Implements Builder.

5.14.1.4 createSimulation()

```
Simulation * ConcreteRocketBuilder::createSimulation ( )  [virtual]
```

Creates a new simulation based on the capsule, rocket and simulation state.

Returns

Simulation∗

Implements Builder.

5.14.1.5 getCapsule()

```
RocketCapsule * ConcreteRocketBuilder::getCapsule ( )
```

Returns the current RocketCapsule object.

Returns

RocketCapsule∗

5.14.1.6 getSpacecraft()

```
Component * ConcreteRocketBuilder::getSpacecraft ( )  [virtual]
```

Returns the current spacecraft/rocket.

Returns

Component∗

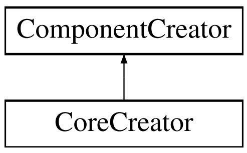
Implements Builder.

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

- C:/Users/labuser2/Downloads/System/System/ConcreteRocketBuilder.h
- C:/Users/labuser2/Downloads/System/System/ConcreteRocketBuilder.cpp

5.15 CoreCreator Class Reference

Inheritance diagram for CoreCreator:



Public Member Functions

- Component ∗ factoryMethod ()

    Factory method to create a new FalconCore object.

5.15.1 Member Function Documentation

5.15.1.1 factoryMethod()

```
Component * CoreCreator::factoryMethod ( )  [virtual]
```
Factory method to create a new FalconCore object.

Returns

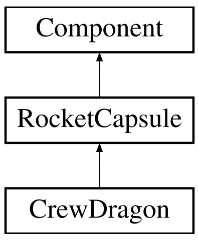
Component∗

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

- C:/Users/labuser2/Downloads/System/System/CoreCreator.h
- C:/Users/labuser2/Downloads/System/System/CoreCreator.cpp

5.16 CrewDragon Class Reference

Inheritance diagram for CrewDragon:



Public Member Functions

- CrewDragon (Component ∗r)

    Constructor for CrewDragon objects that takes in a rocket (Component) as a parameter and uses the RocketCapsule (parent) constructor to initialize the rocket object.
- void simulate ()

    Starts the simulation for CrewDragon objects.
- void test ()

    Tests if the CrewDragon meets all the requirements for a successful launch. The requirements:
- vector< string > getPassengers ()

Returns the vector of passsengers.
- void setPassengers (vector< string > p)

  Sets the vector of passengers to the vector passed in as a parameter.

Additional Inherited Members

5.16.1 Constructor & Destructor Documentation

5.16.1.1 CrewDragon()

```
CrewDragon::CrewDragon (
              Component * r )
```
Constructor for CrewDragon objects that takes in a rocket (Component) as a parameter and uses the RocketCapsule (parent) constructor to initialize the rocket object.

Parameters

r	Component∗ - the rocket to which the CrewDragon object needs to be added to.

5.16.2 Member Function Documentation

5.16.2.1 getPassengers()

```
vector< string > CrewDragon::getPassengers ( )
```
Returns the vector of passsengers.

Returns

vector<string>

5.16.2.2 setPassengers()

```
void CrewDragon::setPassengers (
              vector< string > p ) [virtual]
```
Sets the vector of passengers to the vector passed in as a parameter.

Parameters

p	vector<string> - the vector of passengers to set to.

Returns

void

Reimplemented from RocketCapsule.

5.16.2.3 simulate()

```
void CrewDragon::simulate ( ) [virtual]
```
Starts the simulation for CrewDragon objects.

Returns

void

Implements RocketCapsule.

5.16.2.4 test()

```
void CrewDragon::test ( ) [virtual]
```

Tests if the CrewDragon meets all the requirements for a successful launch. The requirements:

- the cost must be >0
- it must have a capsuleType
- it must have a rocketType

  Returns

  void

Implements RocketCapsule.

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

- C:/Users/labuser2/Downloads/System/System/CrewDragon.h
- C:/Users/labuser2/Downloads/System/System/CrewDragon.cpp

5.17 Director Class Reference

Public Member Functions

- Director (Builder ∗b)

  Constructor that takes in a builder object and initializes the builder variable.
- ∼**Director** ()

  Destructor for Director objects.
- Component ∗ construct ()

  Constructs the rocket. User will have the choice of creating a Falcon9 or FalconHeavy rocket. User will have the option of adding a capsule to the rocket. (Capsule can be a CrewDragon, CargoDragon or Fairing).
- void constructCapsule ()

  Creates a new capsule of types CrewDragon, CargoDragon or Fairing.
- Simulation ∗ createSimulation ()

  Creates a new simulation.

5.17.1 Constructor & Destructor Documentation

5.17.1.1 Director()

```
Director::Director (
                Builder * b )
```

Constructor that takes in a builder object and initializes the builder variable.

Parameters

b	Builder∗ - the builder associated with the Director object.

5.17.2 Member Function Documentation

5.17.2.1 construct()

`Component * Director::construct ( )`

Constructs the rocket. User will have the choice of creating a Falcon9 or FalconHeavy rocket. User will have the option of adding a capsule to the rocket. (Capsule can be a CrewDragon, CargoDragon or Fairing).

Returns

Component*

5.17.2.2 constructCapsule()

`void Director::constructCapsule ( )`

Creates a new capsule of types CrewDragon, CargoDragon or Fairing.

Returns

void

5.17.2.3 createSimulation()

`Simulation * Director::createSimulation ( )`

Creates a new simulation.

Returns

Simulation*

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

- C:/Users/labuser2/Downloads/System/System/Director.h
- C:/Users/labuser2/Downloads/System/System/Director.cpp

5.18 Facade Class Reference

Public Member Functions

- **Facade** ()

  Constructor for Facade objects.
- ∼**Facade** ()

  Destructor for Facade objects.
- void launch ()

  Tests and launches the rocket object.
- void test ()

  Tests if the rocket object is ready to launch. Calls the staticFireTest() on the simulation object.
- void build ()

  Builds a new rocket object. Creates a new Builder. Creates a new Director. Creates a new Simulation. Calls the construct() method on the Director object to actually create the rocket. Deletes the Director object.
- void storeSimulation ()

  Stores the simulation. Creates a Memento and stores it using storeMemento().
- void retrieveSimulation ()

  Sets the simulation variable to the Simulation stored in the Memento object.
- void useCommNetwork ()

Prompts the user with various Communication Network capabilities.

- void separateBoosters ()

  Seperates boosters from the rocket.
- Component ∗ getRocket ()

  Returns the current rocket.
- void fireMerlin ()

  Ignites the MerlinEngine objects.
- void **fireVacuumMerlin** ()

  Ignites the VacuumMerlinEngine objects.
- void runSimulation ()

  Starts the simulation.
- void deliverCrew ()

  Delivers crew once in low-earth orbit.
- void distributeSatellites ()

  Distributes satellites once in low-earth orbit.
- void staticFireTest ()

  Tests the merlin engines and then fires them if they work.
- void jettisonFairing ()

  Delivers Fairing capsule's payload (if Fairing is attached).
- void printSimulation ()

  Prints a visual representation of the Simulation method calls.
- bool editSimulation ()

  Makes adjustments to the store simulation.
- void retrieveAll ()

  Retrieves all the saved simulations.

5.18.1 Member Function Documentation

5.18.1.1 build()

```
void Facade::build ( )
```

Builds a new rocket object. Creates a new Builder. Creates a new Director. Creates a new Simulation. Calls the construct() method on the Director object to actually create the rocket. Deletes the Director object.

Returns

void

5.18.1.2 deliverCrew()

```
void Facade::deliverCrew ( )
```

Delivers crew once in low-earth orbit.

Returns

void

5.18.1.3 distributeSatellites()

`void Facade::distributeSatellites ( )`

Distributes satellites once in low-earth orbit.

Returns

void

5.18.1.4 editSimulation()

`bool Facade::editSimulation ( )`

Makes adjustments to the store simulation.

Returns

void

5.18.1.5 fireMerlin()

`void Facade::fireMerlin ( )`

Ignites the MerlinEngine objects.

Returns

void.

5.18.1.6 getRocket()

`Component * Facade::getRocket ( )`

Returns the current rocket.

Returns

Component*

5.18.1.7 jettisonFairing()

`void Facade::jettisonFairing ( )`

Delivers Fairing capsule's payload (if Fairing is attached).

Returns

void

5.18.1.8 launch()

`void Facade::launch ( )`

Tests and launches the rocket object.

Returns

void

5.18.1.9 printSimulation()

`void Facade::printSimulation ( )`

Prints a visual representation of the Simulation method calls.

Returns

void

5.18.1.10 retrieveAll()

`void Facade::retrieveAll ( )`

Retrieves all the saved simulations.

Returns

void

5.18.1.11 retrieveSimulation()

`void Facade::retrieveSimulation ( )`

Sets the simulation variable to the Simulation stored in the Memento object.

Returns

void

5.18.1.12 runSimulation()

`void Facade::runSimulation ( )`

Starts the simulation.

Returns

void

5.18.1.13 separateBoosters()

`void Facade::separateBoosters ( )`

Seperates boosters from the rocket.

Returns

void

5.18.1.14 staticFireTest()

`void Facade::staticFireTest ( )`

Tests the merlin engines and then fires them if they work.

Returns

void

5.18.1.15 storeSimulation()

```
void Facade::storeSimulation ( )
```

Stores the simulation. Creates a Memento and stores it using storeMemento().

Returns

void

5.18.1.16 test()

```
void Facade::test ( )
```

Tests if the rocket object is ready to launch. Calls the staticFireTest() on the simulation object.

Returns

void

5.18.1.17 useCommNetwork()

```
void Facade::useCommNetwork ( )
```

Prompts the user with various Communication Network capabilities.

Returns

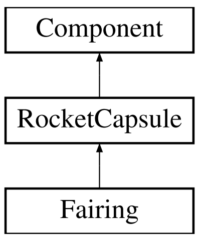
void

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

- C:/Users/labuser2/Downloads/System/System/Facade.h
- C:/Users/labuser2/Downloads/System/System/Facade.cpp

5.19 Fairing Class Reference

Inheritance diagram for Fairing:



Public Member Functions

- Fairing (Component ∗r)

  Constructor for Fairing objects that takes in a rocket (Component) as a parameter and uses the RocketCapsule (parent) constructor to initialize the rocket object.
- void simulate ()

  Starts the simulation for Fairing objects.
- void test ()

  Tests if the Fairing meets all the requirements for a successful launch. The requirements:
- vector< Satellite ∗ > getSatellites ()

  Returns the vector of satellites.
- void setSatellites (vector< Satellite ∗ > s)

  Sets the vector of satellites to the vector passed in as a parameter.
- Satellite ∗ getSatellite (int id)

  Returns the Satellite object with the id passed in as a parameter.

Additional Inherited Members

5.19.1 Constructor & Destructor Documentation

5.19.1.1 Fairing()

```
Fairing::Fairing (
            Component * r )
```

Constructor for Fairing objects that takes in a rocket (Component) as a parameter and uses the RocketCapsule (parent) constructor to initialize the rocket object.

Parameters

r	Component∗ - the rocket that the Fairing needs to be added to.

5.19.2 Member Function Documentation

5.19.2.1 getSatellite()

```
Satellite * Fairing::getSatellite (
            int id ) [virtual]
```

Returns the Satellite object with the id passed in as a parameter.

Parameters

id	int - the id of the Satellite to be returned.

Returns

Satellite∗

Reimplemented from RocketCapsule.

5.19.2.2 getSatellites()

```
vector< Satellite * > Fairing::getSatellites ( )
```

Returns the vector of satellites.

Returns

vector<Satellite∗>

5.19.2.3 setSatellites()

```
void Fairing::setSatellites (
            vector< Satellite * > s ) [virtual]
```

Sets the vector of satellites to the vector passed in as a parameter.

Parameters

s	vector<Satellite∗> - the vector of Satellite objects to be set to.

Reimplemented from RocketCapsule.

5.19.2.4 simulate()

`void Fairing::simulate ( )  [virtual]`

Starts the simulation for Fairing objects.

Returns

void

Implements RocketCapsule.

5.19.2.5 test()

`void Fairing::test ( )  [virtual]`

Tests if the Fairing meets all the requirements for a successful launch. The requirements:

- the cost must be >0
- it must have a capsuleType
- it must have a rocketType

  Returns

  void

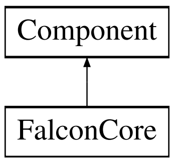
Implements RocketCapsule.

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

- C:/Users/labuser2/Downloads/System/System/Fairing.h
- C:/Users/labuser2/Downloads/System/System/Fairing.cpp

5.20 FalconCore Class Reference

Inheritance diagram for FalconCore:



Public Member Functions

- **FalconCore** ()

  Constructor for FalconCore objects.
- void simulate ()

  Starts the simulation for FalconCore objects.
- void test ()

  Tests if the FalconCore meets all the requirements for a successful launch. The requirements:
- void separate ()

  Outputs that the FalconCore has been seperated from the rocket.
- void land ()

  Outputs that the rocket has landed.

Additional Inherited Members

5.20.1 Member Function Documentation

5.20.1.1 land()

`void FalconCore::land ( ) [virtual]`

Outputs that the rocket has landed.

Returns

void

Reimplemented from Component.

5.20.1.2 separate()

`void FalconCore::separate ( ) [virtual]`

Outputs that the FalconCore has been seperated from the rocket.

Returns

void

Reimplemented from Component.

5.20.1.3 simulate()

`void FalconCore::simulate ( ) [virtual]`

Starts the simulation for FalconCore objects.

Returns

void

Reimplemented from Component.

5.20.1.4 test()

`void FalconCore::test ( ) [virtual]`

Tests if the FalconCore meets all the requirements for a successful launch. The requirements:

- the cost must be >0
- it must have a capsuleType
- it must have a rocketType

  Returns

  void

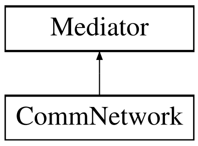
Reimplemented from Component.

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

- C:/Users/labuser2/Downloads/System/System/FalconCore.h
- C:/Users/labuser2/Downloads/System/System/FalconCore.cpp

5.21 Mediator Class Reference

Inheritance diagram for Mediator:



Public Member Functions

- virtual void notify (int sender)=0

  Pure virtual function to be implemented in all children. Notifies all the satellites (colleagues) if the state of the one of the satellites have changed.
- virtual void sendMessage (int sender, int receiver, string msg)=0

  Sends a string message to a particular satellite.

5.21.1 Member Function Documentation

5.21.1.1 notify()

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

Pure virtual function to be implemented in all children. Notifies all the satellites (colleagues) if the state of the one of the satellites have changed.

Parameters

colleague	Satellite∗ - the Satellite object that has changed states.

Returns

void

Pure virtual function to be implemented in all children. Notifies all the satellites (colleagues) if the state of the one of the satellites have changed.

Parameters

colleague	Satellite∗ - the Satellite object that has changed states.

Returns

void

Implemented in CommNetwork.

5.21.1.2 sendMessage()

```
virtual void Mediator::sendMessage (
             int sender,
             int receiver,
             string msg )  [pure virtual]
```

Sends a string message to a particular satellite.

Parameters

sender	The ID of the sender.
receiver	The ID of the receiver
msg	The message to send

Returns

void

Implemented in CommNetwork.
The documentation for this class was generated from the following file:

- C:/Users/labuser2/Downloads/System/System/Mediator.h

5.22 Memento Class Reference

Public Member Functions

- SimulationState ∗ getState ()

  Returns the current state of the simulation.
- void setState (SimulationState ∗c)

  Sets the simulationState the the object passed in as a paramter.
- ∼**Memento** ()

  Destructor for the Memento class.

5.22.1 Member Function Documentation

5.22.1.1 getState()

```
SimulationState * Memento::getState ( )
```

Returns the current state of the simulation.

Returns

SimulationState∗

5.22.1.2 setState()

```
void Memento::setState (
            SimulationState * c )
```

Sets the simulationState the the object passed in as a paramter.

Parameters

c	SimulationState∗ - the SimulationState to be set to.

Returns

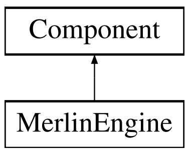
void

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

- C:/Users/labuser2/Downloads/System/System/Memento.h
- C:/Users/labuser2/Downloads/System/System/Memento.cpp

5.23 MerlinEngine Class Reference

Inheritance diagram for MerlinEngine:



Public Member Functions

- **MerlinEngine** ()

  Constructor for MerlinEngine objects. Uses the Component (parent) constructor to initilize the object.
- void simulate ()

  Starts the simulation for MerlinEngine objects.
- void test ()

  Tests if the MerlinEngine meets all the requirements for a successful launch. The requirements:
- void fireMerlin ()

  Outputs that a MerlinEngine has been ignited.

Additional Inherited Members

5.23.1 Member Function Documentation

5.23.1.1 fireMerlin()

```
void MerlinEngine::fireMerlin ( )  [virtual]
```

Outputs that a MerlinEngine has been ignited.

Returns

void

Reimplemented from Component.

5.23.1.2 simulate()

```
void MerlinEngine::simulate ( )  [virtual]
```

Starts the simulation for MerlinEngine objects.

Returns

void

Reimplemented from Component.

5.23.1.3 test()

```
void MerlinEngine::test ( )  [virtual]
```

Tests if the MerlinEngine meets all the requirements for a successful launch. The requirements:

- the cost must be $>$0
- it must have a capsuleType
- it must have a rocketType

Returns

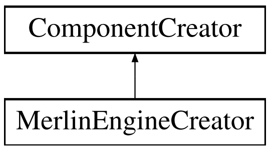
void

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

- C:/Users/labuser2/Downloads/System/System/MerlinEngine.h
- C:/Users/labuser2/Downloads/System/System/MerlinEngine.cpp

5.24 MerlinEngineCreator Class Reference

Inheritance diagram for MerlinEngineCreator:



Public Member Functions

- Component ∗ factoryMethod ()

  Factory method to create a MerlinEngine.

5.24.1 Member Function Documentation

5.24.1.1 factoryMethod()

```
Component ∗ MerlinEngineCreator::factoryMethod ( ) [virtual]
```
Factory method to create a MerlinEngine.

Returns

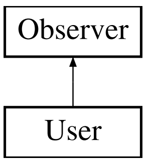
Component∗

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

- C:/Users/labuser2/Downloads/System/System/MerlinEngineCreator.h
- C:/Users/labuser2/Downloads/System/System/MerlinEngineCreator.cpp

5.25 Observer Class Reference

Inheritance diagram for Observer:



Public Member Functions

- virtual void update (int satelliteID, string status)=0

  Pure virtual function to be implemented in all the children classes. Updates the state of a satellite the class is currently observing.

5.25.1 Member Function Documentation

5.25.1.1 update()

```
virtual void Observer::update (
            int satelliteID,
            string status )  [pure virtual]
```

Pure virtual function to be implemented in all the children classes. Updates the state of a satellite the class is currently observing.

Returns

void

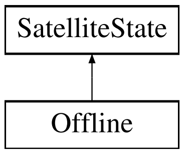
Implemented in User.

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

- C:/Users/labuser2/Downloads/System/System/Observer.h

5.26 Offline Class Reference

Inheritance diagram for Offline:



Public Member Functions

- **Offline** ()

    Constructor for Offline objects.
- string getType ()

    Returns the type of state the satellite is currently in (Offline).
- SatelliteState ∗ handleChange ()

    Handles a change in state - sets the current state of the satellite to null.

5.26.1 Member Function Documentation

5.26.1.1 getType()

```
string Offline::getType ( )  [virtual]
```

Returns the type of state the satellite is currently in (Offline).

Returns

string

Implements SatelliteState.

5.26.1.2 handleChange()

```
SatelliteState * Offline::handleChange ( )  [virtual]
```

Handles a change in state - sets the current state of the satellite to null.

Returns

SatelliteState*

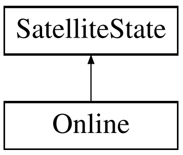
Implements SatelliteState.

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

- C:/Users/labuser2/Downloads/System/System/Offline.h
- C:/Users/labuser2/Downloads/System/System/Offline.cpp

5.27 Online Class Reference

Inheritance diagram for Online:



Public Member Functions

- **Online** ()

  Constructor for Online objects.
- string getType ()

  Returns the type of state the satellite is currently in (Online).
- SatelliteState * handleChange ()

  Handles a change in state - sets the current state of the satellite to 'Broadcasting'.

5.27.1 Member Function Documentation

5.27.1.1 getType()

```
string Online::getType ( )  [virtual]
```

Returns the type of state the satellite is currently in (Online).

Returns

string

Implements SatelliteState.

5.27.1.2 handleChange()

```
SatelliteState * Online::handleChange ( )  [virtual]
```

Handles a change in state - sets the current state of the satellite to 'Broadcasting'.

Returns

SatelliteState*

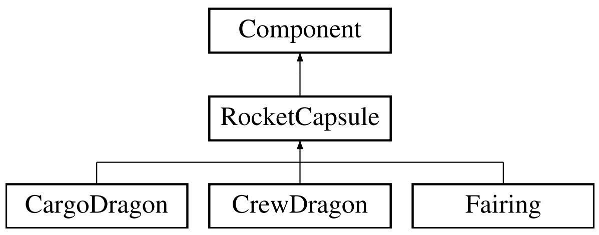
Implements SatelliteState.

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

- C:/Users/labuser2/Downloads/System/System/Online.h
- C:/Users/labuser2/Downloads/System/System/Online.cpp

5.28 RocketCapsule Class Reference

Inheritance diagram for RocketCapsule:



Public Member Functions

- **RocketCapsule** (Component ∗r)

  Constructor for RocketCapsule objects. Sets the rocket variable to the Component sent in as a parameter. Sets the state to CapsuleOffline.
- virtual void simulate ()=0

  Pure virtual function to be implemented in all the children classes. Starts the simulation for RocketCapsule objects.
- virtual void test ()=0

  Tests if the RocketCapsule meets all the requirements for a successful launch. The requirements:
- void addCapsule (Component ∗r)

  Adds a capsule to the rocket.
- void requestStateChange ()

  Requests a state change of the RocketCapsule. Calls the handleChange method on the CapsuleState object.
- void setState (CapsuleState ∗s)

  Sets the state of the RocketCapsule to the CapsuleState passed in as a parameter.
- double getPayloadWeight ()

  Getter function to return the payloadWeight of the RocketCapsule.
- void setPayloadWeight (double pw)

  Setter function to set the payloadWeight of the RocketCapsule to the value passed in as a parameter.
- virtual void setPassengers (vector< string > p)

  Virtual setter function to set the vector of passengers in a CrewDragon RocketCapsule. Only implemented in the CrewDragon class.
- virtual void setSatellites (vector< Satellite ∗ > s)

  Virtual setter function to set the vector of satellites in a Fairing RocketCapsule. Only implemented in the Fairing class.
- virtual Satellite ∗ getSatellite (int id)

  Get a specific Satellite object on board the Fairing.
- CapsuleState ∗ getState ()

  Get the CapsuleState object.

Protected Attributes

- string capsuleType
- CapsuleState ∗ state

5.28.1 Member Function Documentation

5.28.1.1 addCapsule()

```
void RocketCapsule::addCapsule (
            Component ∗ r )
```

Adds a capsule to the rocket.

Parameters

r	Component∗ - the rocket to be added to.

Returns

void

5.28.1.2 getPayloadWeight()

```
double RocketCapsule::getPayloadWeight ( )
```

Getter function to return the payloadWeight of the RocketCapsule.

Returns

double

5.28.1.3 getSatellite()

```
Satellite * RocketCapsule::getSatellite (
            int id ) [virtual]
```

Get a specific Satellite object on board the Fairing.

Returns

Satellite∗

Reimplemented in Fairing.

5.28.1.4 getState()

```
CapsuleState * RocketCapsule::getState ( )
```

Get the CapsuleState object.

Returns

CapsuleState∗

5.28.1.5 requestStateChange()

```
void RocketCapsule::requestStateChange ( )
```

Requests a state change of the RocketCapsule. Calls the handleChange method on the CapsuleState object.

Returns

void

5.28.1.6 setPassengers()

```
virtual void RocketCapsule::setPassengers (
            vector< string > p ) [inline], [virtual]
```

Virtual setter function to set the vector of passengers in a CrewDragon RocketCapsule. Only implemented in the CrewDragon class.

Parameters

p	vector<string> - the string vector containing the names of all the passengers.

Returns

void

Reimplemented in CrewDragon.

5.28.1.7 setPayloadWeight()

```
void RocketCapsule::setPayloadWeight (
            double pw )
```

Setter function to set the payloadWeight of the RocketCapsule to the value passed in as a parameter.

Parameters

pw	double - the value to be set to.

5.28.1.8 setSatellites()

```
virtual void RocketCapsule::setSatellites (
            vector< Satellite * > s ) [inline], [virtual]
```

Virtual setter function to set the vector of satellites in a Fairing RocketCapsule. Only implemented in the Fairing class.

Parameters

s	vector<Satellite*> - vector of all the Satellite objects in the Fairing.

Returns

void

Reimplemented in Fairing.

5.28.1.9 setState()

```
void RocketCapsule::setState (
            CapsuleState * s )
```

Sets the state of the RocketCapsule to the CapsuleState passed in as a parameter.

Parameters

s	CapsuleState* - the state to be set to.

Returns

void

5.28.1.10 simulate()

```
void RocketCapsule::simulate ( )  [pure virtual]
```

Pure virtual function to be implemented in all the children classes. Starts the simulation for RocketCapsule objects.

Returns

void

Reimplemented from Component.
Implemented in CargoDragon, CrewDragon, and Fairing.

5.28.1.11 test()

```
void RocketCapsule::test ( )  [pure virtual]
```

Tests if the RocketCapsule meets all the requirements for a successful launch. The requirements:

- the cost must be >0
- it must have a capsuleType
- it must have a rocketType

  Returns

  void

Reimplemented from Component.
Implemented in CargoDragon, CrewDragon, and Fairing.

5.28.2 Member Data Documentation

5.28.2.1 capsuleType

```
string RocketCapsule::capsuleType  [protected]
```

The type of capsule (CrewDragon, CargoDragon, Fairing).

5.28.2.2 state

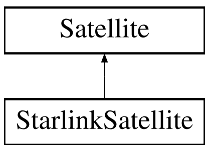
```
CapsuleState* RocketCapsule::state  [protected]
```

The state of the capsule (Docked, Arrriving, Departing, Offline).
The documentation for this class was generated from the following files:

- C:/Users/labuser2/Downloads/System/System/RocketCapsule.h
- C:/Users/labuser2/Downloads/System/System/RocketCapsule.cpp

5.29 Satellite Class Reference

Inheritance diagram for Satellite:



Public Member Functions

- Satellite (int ID)

  Constructor for Satellite objects. Sets the ID to the value sent in as a parameter.
- ∼**Satellite** ()

  Destructor for Satellite objects.
- void changed ()

  Notifies the other satellites that the state of the current satellite has changed by calling the notify() method on the mediator object.
- int getID ()

  Getter method that returns the ID of the Satellite object.
- void sendMessage (int id, string msg)

  Sends a string message to a particular satellite, based on ID.
- void receiveMessage (int id, string msg)

  Prints out the state and ID of the changed colleague.
- void setMediator (Mediator ∗m)

  Setter function to set the mediator to the Mediator object passed in as a parameter.
- Mediator ∗ **getMediator** ()
- void requestStateChange ()

  Changes the state of the satellite by calling setState(). Notifies the mediators by calling changed(). Notifies the observers by calling notify().
- void attach (Observer ∗o)

  Registers an observer with the Satellite. Adds the Observer sent in as a parameter to the observerList.
- void detach (Observer ∗o)

  Deregisters an observer from the Satellite. Removes the Observer sent in as a paramter from the observerList.
- void notify ()

  Notifies all the observers of the state change.
- virtual SatelliteState ∗ getState ()

  Virtual function that returns the state of the Satellite.
- void setState (SatelliteState ∗s)

  Sets the state of the Satellite to the SatelliteState sent in as a parameter.
- virtual Satellite ∗ clone (int id)

  Virtual function that clones a Satellite object. Returns NULL.

Protected Attributes

- SatelliteState ∗ satelliteState
- Mediator ∗ mediator
- vector< Observer ∗ > observerList
- int ID

5.29.1 Constructor & Destructor Documentation

5.29.1.1 Satellite()

```
Satellite::Satellite (
            int ID )
```

Constructor for Satellite objects. Sets the ID to the value sent in as a parameter.

Parameters

ID	int - the id of the satellite.

5.29.2 Member Function Documentation

5.29.2.1 attach()

```
void Satellite::attach (
             Observer * o )
```

Registers an observer with the Satellite. Adds the Observer sent in as a parameter to the observerList.

Parameters

o	Observer∗ - the Observer to be registered.

Returns

void

5.29.2.2 changed()

```
void Satellite::changed ( )
```

Notifies the other satellites that the state of the current satellite has changed by calling the notify() method on the mediator object.

Returns

void

5.29.2.3 clone()

```
virtual Satellite * Satellite::clone (
             int id ) [inline], [virtual]
```

Virtual function that clones a Satellite object. Returns NULL.

Returns

Satellite∗

Reimplemented in StarlinkSatellite.

5.29.2.4 detach()

```
void Satellite::detach (
             Observer * o )
```

Deregisters an observer from the Satellite. Removes the Observer sent in as a paramter from the observerList.

Parameters

o	Observer∗ - the Observer to be deregistered.

Returns

void

5.29.2.5 getID()

```
int Satellite::getID ( )
```

Getter method that returns the ID of the Satellite object.

Returns

int

5.29.2.6 getState()

```
SatelliteState * Satellite::getState ( )  [virtual]
```

Virtual function that returns the state of the Satellite.

Returns

SatelliteState*

Reimplemented in StarlinkSatellite.

5.29.2.7 notify()

```
void Satellite::notify ( )
```

Notifies all the observers of the state change.

Returns

void

5.29.2.8 receiveMessage()

```
void Satellite::receiveMessage (
             int id,
             string msg )
```

Prints out the state and ID of the changed colleague.

Parameters

msg	string - the message that needs to be printed.

Returns

void

5.29.2.9 requestStateChange()

```
void Satellite::requestStateChange ( )
```

Changes the state of the satellite by calling setState(). Notifies the mediators by calling changed(). Notifies the observers by calling notify().

Returns

void

5.29.2.10 sendMessage()

```
void Satellite::sendMessage (
            int id,
            string msg )
```

Sends a string message to a particular satellite, based on ID.

Parameters

id	int - The ID of the satellite.
msg	string - The message that needs to be printed.

Returns

void

5.29.2.11 setMediator()

```
void Satellite::setMediator (
            Mediator * m )
```

Setter function to set the mediator to the Mediator object passed in as a parameter.

Parameters

m	Mediator∗ - the Mediator object to be set to.

Returns

void

5.29.2.12 setState()

```
void Satellite::setState (
            SatelliteState * s )
```

Sets the state of the Satellite to the SatelliteState sent in as a parameter.

Parameters

s	SatelliteState∗ - the state to be set to.

Returns

void

5.29.3 Member Data Documentation

5.29.3.1 ID

`int Satellite::ID [protected]`

The integer used to uniquely identify the satellite.

5.29.3.2 mediator

`Mediator* Satellite::mediator [protected]`

The mediator object that controls the communication between satellites.

5.29.3.3 observerList

`vector<Observer*> Satellite::observerList [protected]`

The list of observers that are observing and monitoring the satellite.

5.29.3.4 satelliteState

`SatelliteState* Satellite::satelliteState [protected]`

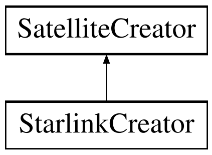
The state of the Satellite

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

- C:/Users/labuser2/Downloads/System/System/Satellite.h
- C:/Users/labuser2/Downloads/System/System/Satellite.cpp

5.30 SatelliteCreator Class Reference

Inheritance diagram for SatelliteCreator:



Public Member Functions

- **SatelliteCreator** ()

  Constructor for the SatelliteCreator objects. Sets count to 0.
- virtual Satellite ∗ factoryMethod ()=0

  Pure virtual function to be implemented in children classes. Factory method to create Satellite objects.
- virtual void setIDCount (int id)=0

  Pure virtual function to be implemented in children classes. Sets the count variable to the integer passed in as a parameter.

5.30.1 Member Function Documentation

5.30.1.1 factoryMethod()

`Satellite * SatelliteCreator::factoryMethod ( ) [pure virtual]`

Pure virtual function to be implemented in children classes. Factory method to create Satellite objects.

Returns

Satellite∗

Implemented in StarlinkCreator.

5.30.1.2 setIDCount()

```
virtual void SatelliteCreator::setIDCount (
             int id )  [pure virtual]
```

Pure virtual function to be implemented in children classes. Sets the count variable to the integer passed in as a parameter.

Parameters

id	int - the number to be set to.

Returns

void

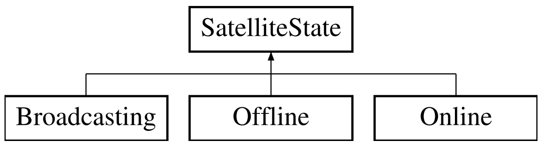
Implemented in StarlinkCreator.

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

- C:/Users/labuser2/Downloads/System/System/SatelliteCreator.h
- C:/Users/labuser2/Downloads/System/System/SatelliteCreator.cpp

5.31 SatelliteState Class Reference

Inheritance diagram for SatelliteState:



Public Member Functions

- virtual string getType ()=0

  Pure virtual function to be implemented in children classes. Returns the name of the current state of the Satellite.
- virtual SatelliteState ∗ handleChange ()=0

  Pure virtual function to be implemented in children classes. Handles a change in state.

5.31.1 Member Function Documentation

5.31.1.1 getType()

```
virtual string SatelliteState::getType ( )  [pure virtual]
```

Pure virtual function to be implemented in children classes. Returns the name of the current state of the Satellite.

Returns

string

Implemented in Broadcasting, Offline, and Online.

5.31.1.2 handleChange()

```
virtual SatelliteState * SatelliteState::handleChange ( )  [pure virtual]
```

Pure virtual function to be implemented in children classes. Handles a change in state.

Returns

SatelliteState∗

Implemented in Broadcasting, Offline, and Online.

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

- C:/Users/labuser2/Downloads/System/System/SatelliteState.h

5.32 Simulation Class Reference

Public Member Functions

- Simulation (RocketCapsule ∗c, Component ∗r, SimulationState ∗s)

  Constructor for Simulation objects. Sets the simulationState, rocket and capsule variables.
- RocketCapsule ∗ **getCapsule** ()
- Memento ∗ createMemento ()

  Creates a Memento object and sets the state of the memento to the current simulationState.
- void restoreMemento (Memento ∗m)

  Sets the simulationState to the state of the memento.
- void staticFireTest ()

  Adds a call to the simulationState and calls the test() method on the rocket.
- void launch ()

  Launches the rocket. IF FALCON9 ROCKET- – Stage 1: single falcon 9 core with 9 Merlin engines – Stage 2: single vacuum Merlin engine IF FALCONHEAVY ROCKET- – Stage 1: 3 Falcon Heavy cores with 27 Merlin engines – Stage 2: single Merlin engine.
- void printSimulation ()

  Pure virtual method to be implemented in all children classes. Tweaks the simulation on the rocket to represent a more realistic example of a real-world rocket simulation.
- void jettisonFairing ()

  Delivers Fairing capsule's payload (if Fairing is attached).
- void separateBoosters ()

  Seperates boosters from the rocket.
- void distributeSatellites ()

  Distributes satellites once in low-earth orbit.
- void deliverCrew ()

  Delivers crew once in low-earth orbit.
- void sendMessage (int sender, int reciever, string message)

  Sends a string message from the receiving satellite to a receiving satellite.
- void runSimulation ()

  Runs various launch methods from the simulationState consecutively in order to simulate launch event.
- SimulationState ∗ getState ()

  Returns the state of te Simulation.
- void fireMerlin ()

  Outputs that a MerlinEngine has been ignited.
- void landBoosters ()

  Outputs that the boosters have landed.
- void fireVacuumMerlin ()

  Outputs that a VacuumMerlinEngine has been ignited.
- void changeSatelliteState (int id, SatelliteState ∗state)

Changes the state of the Satellite with an id of the int passed in as a parameter to the state passed in as a parameter.
- void addCall (string c)

  Adds a call to vector of methodCalls.
- bool containsCall (string c)

  Checks if the methodCall vector contains the string passed in as a parameter. Returns true if it contains it. Returns false if it doesn't contain it.
- void updateSimulationState ()

  Updates the simulation state to the most recent method calls.
- void swapStage (int pos_1, int pos_2)

  Swaps between the indexes of the 2 method calls sent in as parameters.
- void removeStage (int pos)

  Removes the method call in the index of the number sent in as a parameter.
- int getSimulationSize ()

  Returns the size of the simulation - the number of method calls/ stages in the simulation.

5.32.1 Constructor & Destructor Documentation

5.32.1.1 Simulation()

```
Simulation::Simulation (
            RocketCapsule * c,
            Component * r,
            SimulationState * s )
```

Constructor for Simulation objects. Sets the simulationState, rocket and capsule variables.

Parameters

c	Component∗ - The capsule on the rocket on which the Simulation is performed.
r	Component∗ - The rocket on which the Simulation is performed.
s	SimulationState∗ - The state of the Simulation.

5.32.2 Member Function Documentation

5.32.2.1 addCall()

```
void Simulation::addCall (
            string c )
```

Adds a call to vector of methodCalls.

Parameters

c	string - the method call to add to the vector.

Returns

void

5.32.2.2 changeSatelliteState()

```
void Simulation::changeSatelliteState (
             int id,
             SatelliteState * state )
```

Changes the state of the Satellite with an id of the int passed in as a parameter to the state passed in as a parameter.

Parameters

id	int - the id of the Satellite of which we need to change the state.
state	SatelliteState∗ - the state to change to.

5.32.2.3 containsCall()

```
bool Simulation::containsCall (
             string c )
```

Checks if the methodCall vector contains the string passed in as a parameter. Returns true if it contains it. Returns false if it doesn't contain it.

Parameters

c	string - the method call to check for.

Returns

bool

5.32.2.4 createMemento()

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

Creates a Memento object and sets the state of the memento to the current simulationState.

Returns

Memento∗

5.32.2.5 deliverCrew()

```
void Simulation::deliverCrew ( )
```

Delivers crew once in low-earth orbit.

Returns

void

5.32.2.6 distributeSatellites()

```
void Simulation::distributeSatellites ( )
```

Distributes satellites once in low-earth orbit.

Returns

void

5.32.2.7 fireMerlin()

`void Simulation::fireMerlin ( )`

Outputs that a MerlinEngine has been ignited.

Returns

void

5.32.2.8 fireVacuumMerlin()

`void Simulation::fireVacuumMerlin ( )`

Outputs that a VacuumMerlinEngine has been ignited.

Returns

void

5.32.2.9 getSimulationSize()

`int Simulation::getSimulationSize ( )`

Returns the size of the simulation - the number of method calls/ stages in the simulation.

Returns

int

5.32.2.10 getState()

`SimulationState * Simulation::getState ( )`

Returns the state of te Simulation.

Returns

SimulationState*

5.32.2.11 jettisonFairing()

`void Simulation::jettisonFairing ( )`

Delivers Fairing capsule's payload (if Fairing is attached).

Returns

void

5.32.2.12 landBoosters()

`void Simulation::landBoosters ( )`

Outputs that the boosters have landed.

Returns

void

5.32.2.13 launch()

```
void Simulation::launch ( )
```

Launches the rocket. IF FALCON9 ROCKET- – Stage 1: single falcon 9 core with 9 Merlin engines – Stage 2: single vacuum Merlin engine IF FALCONHEAVY ROCKET- – Stage 1: 3 Falcon Heavy cores with 27 Merlin engines – Stage 2: single Merlin engine.

Returns

void

5.32.2.14 printSimulation()

```
void Simulation::printSimulation ( )
```

Pure virtual method to be implemented in all children classes. Tweaks the simulation on the rocket to represent a more realistic example of a real-world rocket simulation.

Returns

void

Prints a visual representation of the Simulation method calls.

Returns

void

5.32.2.15 removeStage()

```
void Simulation::removeStage (
            int pos )
```

Removes the method call in the index of the number sent in as a parameter.

Parameters

pos	int - the position of the method call to be removed.

Returns

void

5.32.2.16 restoreMemento()

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

Sets the simulationState to the state of the memento.

Parameters

m	Memento∗ - the Memento object storing the current simulation.

Returns

void

5.32.2.17 runSimulation()

```
void Simulation::runSimulation ( )
```

Runs various launch methods from the simulationState consecutively in order to simulate launch event.

Returns

void

5.32.2.18 sendMessage()

```
void Simulation::sendMessage (
             int sender,
             int reciever,
             string message )
```

Sends a string message from the receiving satellite to a receiving satellite.

Returns

void

5.32.2.19 separateBoosters()

```
void Simulation::separateBoosters ( )
```

Seperates boosters from the rocket.

Returns

void

5.32.2.20 staticFireTest()

```
void Simulation::staticFireTest ( )
```

Adds a call to the simulationState and calls the test() method on the rocket.

Returns

void

5.32.2.21 swapStage()

```
void Simulation::swapStage (
             int pos_1,
             int pos_2 )
```

Swaps between the indexes of the 2 method calls sent in as parameters.

Parameters

pos_1	int - current position.
pos2↩_2	int - current position.

Returns

void

5.32.2.22 updateSimulationState()

`void Simulation::updateSimulationState ( )`

Updates the simulation state to the most recent method calls.

Returns

void

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

- C:/Users/labuser2/Downloads/System/System/Simulation.h
- C:/Users/labuser2/Downloads/System/System/Simulation.cpp

5.33 SimulationState Class Reference

Public Member Functions

- **SimulationState** ()

  Constructor for SimulationState objects.
- string getCapsuleType ()

  Returns the capsule type of the rocket (CrewDragon, CargoDragon, Fairing).
- string getRocketType ()

  Returns the rocket type of the rocket (Falcon9, FalconHeavy).
- double getPayloadWeight ()

  Returns the payloadWeight of the capsule.
- vector< Satellite * > getSatellites ()

  Returns the vector of satellites on the rocket (if capsuleType==Fairing).
- vector< string > getPassengers ()

  Returns the vector of passengers on the rocket (if capsuleType==CrewDragon).
- vector< string > getMethodCalls ()

  Returns the vector of methodCalls on the rocket.
- void setCapsuleType (string s)

  Sets the capsuleType to the string passed in as a parameter.
- void setRocketType (string s)
- void setPayloadWeight (double d)

  Sets the payloadWeight to the string passed in as a parameter.
- void setSatellites (vector< Satellite * > s)

  Sets the vector of Satellites to the vector passed in as a parameter.
- void setPassengers (vector< string > p)

  Sets the vector of passengers (strings) to the vector passed in as a parameter.
- void setMethodCalls (vector< string > c)

  Sets the vector of method calls (strings) to the vector passed in as a parameter.

5.33.1 Member Function Documentation

5.33.1.1 getCapsuleType()

```
string SimulationState::getCapsuleType ( )
```

Returns the capsule type of the rocket (CrewDragon, CargoDragon, Fairing).

Returns

string

5.33.1.2 getMethodCalls()

```
vector< string > SimulationState::getMethodCalls ( )
```

Returns the vector of methodCalls on the rocket.

Returns

vector<string>

5.33.1.3 getPassengers()

```
vector< string > SimulationState::getPassengers ( )
```

Returns the vector of passengers on the rocket (if capsuleType==CrewDragon).

Returns

vector<string>

5.33.1.4 getPayloadWeight()

```
double SimulationState::getPayloadWeight ( )
```

Returns the payloadWeight of the capsule.

Returns

double

5.33.1.5 getRocketType()

```
string SimulationState::getRocketType ( )
```

Returns the rocket type of the rocket (Falcon9, FalconHeavy).

Returns

string

5.33.1.6 getSatellites()

```
vector< Satellite * > SimulationState::getSatellites ( )
```

Returns the vector of satellites on the rocket (if capsuleType==Fairing).

Returns

vector<Satellite*>

5.33.1.7 setCapsuleType()

```
void SimulationState::setCapsuleType (
             string s )
```

Sets the capsuleType to the string passed in as a parameter.

Parameters

s	string - the capsuleType to set to.

Returns

void

5.33.1.8 setMethodCalls()

```
void SimulationState::setMethodCalls (
            vector< string > c )
```

Sets the vector of method calls (strings) to the vector passed in as a parameter.

Parameters

c	vector<string> - the vector of string to set to.

Returns

void

5.33.1.9 setPassengers()

```
void SimulationState::setPassengers (
            vector< string > p )
```

Sets the vector of passengers (strings) to the vector passed in as a parameter.

Parameters

p	vector<string> - the vector of strings to set to.

Returns

void

5.33.1.10 setPayloadWeight()

```
void SimulationState::setPayloadWeight (
            double d )
```

Sets the payloadWeight to the string passed in as a parameter.

Parameters

d	double - the value to set to.

Returns

void e

5.33.1.11 setRocketType()

```
void SimulationState::setRocketType (
            string s )
```

Sets the rocketType tpo o the string passed in as a paramteeter.

Parameters

s	string - the rocketType to set to.

Returns

void

5.33.1.12 setSatellites()

```
void SimulationState::setSatellites (
            vector< Satellite * > s )
```

Sets the vector of Satellites to the vector passed in as a parameter.

Parameters

s	vector<Satellite*> - the vector of Satellites to set to.

Returns

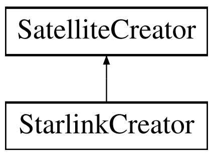
void

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

- C:/Users/labuser2/Downloads/System/System/SimulationState.h
- C:/Users/labuser2/Downloads/System/System/SimulationState.cpp

5.34 StarlinkCreator Class Reference

Inheritance diagram for StarlinkCreator:



Public Member Functions

- **StarlinkCreator** ()

  Constructor for StarlinkCreator objects. Sets IDcount to zero.
- Satellite ∗ factoryMethod ()

  Factory method to create StarlinkSatellite objects.

- void setIDCount (int count)

  Set the current StarlinkSatellite ID counter.

5.34.1 Member Function Documentation

5.34.1.1 factoryMethod()

```
Satellite * StarlinkCreator::factoryMethod ( )  [virtual]
```

Factory method to create StarlinkSatellite objects.

Returns

Satellite∗

Implements SatelliteCreator.

5.34.1.2 setIDCount()

```
void StarlinkCreator::setIDCount (
             int count )  [virtual]
```

Set the current StarlinkSatellite ID counter.

Parameters

count	int - the number to be set to.

Returns

void

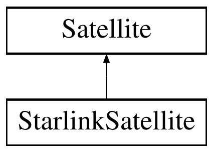
Implements SatelliteCreator.

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

- C:/Users/labuser2/Downloads/System/System/StarlinkCreator.h
- C:/Users/labuser2/Downloads/System/System/StarlinkCreator.cpp

5.35 StarlinkSatellite Class Reference

Inheritance diagram for StarlinkSatellite:



Public Member Functions

- StarlinkSatellite (int ID)

  Constructor method for StarlinkSatellite objects. Sets the ID to the integer sent in as a parameter.
- StarlinkSatellite (Satellite ∗s, int id)

  Copy constructor for StarlinkSatellite objects.
- SatelliteState ∗ getState ()

  Returns the state of the StarlinkSatellite.
- void setState (SatelliteState ∗s)

Sets the state of the Satellite to the SatelliteState sent in as a parameter.

- Satellite ∗ clone (int id)

  Calls the copy constructor in order to replicate current satellite.

Additional Inherited Members

5.35.1 Constructor & Destructor Documentation

5.35.1.1 StarlinkSatellite() [1/2]

```
StarlinkSatellite::StarlinkSatellite (
              int ID )
```

Constructor method for StarlinkSatellite objects. Sets the ID to the integer sent in as a parameter.

Parameters

ID	int

5.35.1.2 StarlinkSatellite() [2/2]

```
StarlinkSatellite::StarlinkSatellite (
              Satellite ∗ s,
              int id )
```

Copy constructor for StarlinkSatellite objects.

Parameters

s	Satellite∗ - the satellite to copy.
id	int - the id of the new satellite.

5.35.2 Member Function Documentation

5.35.2.1 clone()

```
Satellite ∗ StarlinkSatellite::clone (
              int id ) [virtual]
```

Calls the copy constructor in order to replicate current satellite.

Parameters

id	int - the id of the new satellite.

Returns

Satellite∗

Reimplemented from Satellite.

5.35.2.2 getState()

```
SatelliteState ∗ StarlinkSatellite::getState ( ) [virtual]
```

Returns the state of the StarlinkSatellite.

Returns

SatelliteState∗

Reimplemented from Satellite.

5.35.2.3 setState()

```
void StarlinkSatellite::setState (
             SatelliteState ∗ s )
```
Sets the state of the Satellite to the SatelliteState sent in as a parameter.

Parameters

s	SatelliteState

Returns

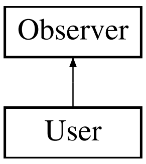
void

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

- C:/Users/labuser2/Downloads/System/System/StarlinkSatellite.h
- C:/Users/labuser2/Downloads/System/System/StarlinkSatellite.cpp

5.36 User Class Reference

Inheritance diagram for User:



Public Member Functions

- User (Satellite ∗s)

    Constructor for User objects. Sets the subject variable to the StarlinkSatellite passed in as a parameter.
- void update (int satelliteID, string status)

    Updates the satelliteState variable.

5.36.1 Constructor & Destructor Documentation

5.36.1.1 User()

```
User::User (
             Satellite ∗ s )
```
Constructor for User objects. Sets the subject variable to the StarlinkSatellite passed in as a parameter.

Parameters

s	StarlinkSatellite∗ - the StarlinkSatellite to set to.

5.36.2 Member Function Documentation

5.36.2.1 update()

```
void User::update (
             int satelliteID,
             string status )  [virtual]
```
Updates the satelliteState variable.

Returns

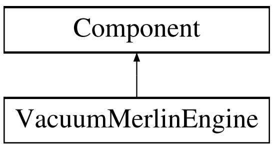
void

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

- C:/Users/labuser2/Downloads/System/System/User.h
- C:/Users/labuser2/Downloads/System/System/User.cpp

5.37 VacuumMerlinEngine Class Reference

Inheritance diagram for VacuumMerlinEngine:



Public Member Functions

- **VacuumMerlinEngine** ()

  Constructor for VacuumMerlinEngine objects. Calls the Component constructor the initialize the attributes.
- void simulate ()

  Starts the simulation for VacuumMerlinEngine objects.
- void test ()

  Tests if the MerlinEngine meets all the requirements for a successful launch. The requirements:
- void fireVacuumMerlin ()

  Outputs that a VacuumMerlin object has been ignited.

Additional Inherited Members

5.37.1 Member Function Documentation

5.37.1.1 fireVacuumMerlin()

```
void VacuumMerlinEngine::fireVacuumMerlin ( )  [virtual]
```
Outputs that a VacuumMerlin object has been ignited.

Returns

void

Reimplemented from Component.

5.37.1.2 simulate()

```
void VacuumMerlinEngine::simulate ( )  [virtual]
```

Starts the simulation for VacuumMerlinEngine objects.

Returns

void

Reimplemented from Component.

5.37.1.3 test()

```
void VacuumMerlinEngine::test ( )  [virtual]
```

Tests if the MerlinEngine meets all the requirements for a successful launch. The requirements:

- the cost must be $>$0
- it must have a capsuleType
- it must have a rocketType

  Returns

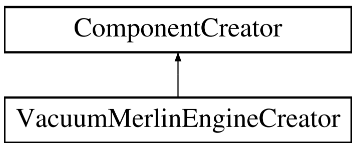
  void

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

- C:/Users/labuser2/Downloads/System/System/VacuumMerlinEngine.h
- C:/Users/labuser2/Downloads/System/System/VacuumMerlinEngine.cpp

5.38 VacuumMerlinEngineCreator Class Reference

Inheritance diagram for VacuumMerlinEngineCreator:



Public Member Functions

- Component ∗ factoryMethod ()

  Factory Method to create VacuumMerlinEngine objects.

5.38.1 Member Function Documentation

5.38.1.1 factoryMethod()

```
Component ∗ VacuumMerlinEngineCreator::factoryMethod ( )  [virtual]
```

Factory Method to create VacuumMerlinEngine objects.

Returns

Component∗

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

- C:/Users/labuser2/Downloads/System/System/VacuumMerlinEngineCreator.h
- C:/Users/labuser2/Downloads/System/System/VacuumMerlinEngineCreator.cpp

Chapter 6

File Documentation

6.1 C:/Users/labuser2/Downloads/System/System/Broadcasting.cpp File Reference

Implementation for Broadcasting.h.

```
#include "Broadcasting.h"
#include "Offline.h"
```

6.1.1 Detailed Description

Implementation for Broadcasting.h.

6.2 C:/Users/labuser2/Downloads/System/System/Broadcasting.h File Reference

Participant - Concrete State (State). Describes the properties and methods of a Satellite in the 'broadcasting' state.

```
#include "SatelliteState.h"
```

Classes

- class Broadcasting

6.2.1 Detailed Description

Participant - Concrete State (State). Describes the properties and methods of a Satellite in the 'broadcasting' state.

Author

The 6 Musketeers

6.3 Broadcasting.h

Go to the documentation of this file.

```

#ifndef BROADCASTING_H
#define BROADCASTING_H

#include "SatelliteState.h"

using namespace std;
class Satellite;
class Broadcasting : public SatelliteState
{
    public:
        Broadcasting();
        string getType();

        SatelliteState* handleChange();
};

#endif
```

6.4 C:/Users/labuser2/Downloads/System/System/Builder.h File Reference

Participant - Builder (Builder) Describes the methods to build the components of a rocket.

```
#include <iostream>
#include <string>
#include "Component.h"
#include "Simulation.h"
```

Classes

- class Builder

6.4.1 Detailed Description

Participant - Builder (Builder) Describes the methods to build the components of a rocket.

Author

The 6 Musketeers

6.5 Builder.h

Go to the documentation of this file.

```

#ifndef BUILDER_H
#define BUILDER_H

#include <iostream>
#include <string>
#include "Component.h"
#include "Simulation.h"

using namespace std;
class Builder
{
    public:
        virtual void buildFalcon9() = 0;

        virtual void buildFalconHeavy() = 0;

        virtual void constructCapsule(string c) = 0;


        virtual Component*  getSpacecraft() = 0;

        virtual Simulation* createSimulation() = 0;
};

#endif
```

6.6 C:/Users/labuser2/Downloads/System/System/CapsuleArriving.cpp File Reference

Implementation for CapsuleArriving.h.

```
#include "CapsuleArriving.h"
#include "CapsuleDocked.h"
```

6.6.1 Detailed Description

Implementation for CapsuleArriving.h.

6.7 C:/Users/labuser2/Downloads/System/System/CapsuleArriving.h File Reference

Participant - Concrete State (State) Describes the methods of a Capusle that is in an 'arriving' state.

```
#include "CapsuleState.h"
```

Classes

- class CapsuleArriving

6.7.1 Detailed Description

Participant - Concrete State (State) Describes the methods of a Capusle that is in an 'arriving' state.

Author

The 6 Musekteers

6.8 CapsuleArriving.h

Go to the documentation of this file.

```

#ifndef CAPSULEARRIVING_H
#define CAPSULEARRIVING_H

#include "CapsuleState.h"

using namespace std;

class CapsuleArriving : public CapsuleState
{
    public:
        CapsuleArriving();

        string getState();

        CapsuleState* handleChange();
};

#endif
```

6.9 C:/Users/labuser2/Downloads/System/System/CapsuleDeparting.cpp File Reference

Implementation for CapsuleDeparting.h.

```
#include "CapsuleDeparting.h"
#include "CapsuleArriving.h"
#include "RocketCapsule.h"
```

6.9.1 Detailed Description

Implementation for CapsuleDeparting.h.

6.10 C:/Users/labuser2/Downloads/System/System/CapsuleDeparting.h File Reference

Participant - Concrete State (State) Describes the methods of a Capusle that is in a 'departing' state.

```
#include "CapsuleState.h"
```

Classes

- class CapsuleDeparting

6.10.1 Detailed Description

Participant - Concrete State (State) Describes the methods of a Capusle that is in a 'departing' state.

Author

The 6 Musekteers

6.11 CapsuleDeparting.h

Go to the documentation of this file.

```
#ifndef CAPSULEDEPARTING_H
#define CAPSULEDEPARTING_H

#include "CapsuleState.h"

using namespace std;

class CapsuleDeparting : public CapsuleState
{
    public:

        CapsuleDeparting();

        string getState();

        CapsuleState* handleChange();
};

#endif
```

6.12 C:/Users/labuser2/Downloads/System/System/CapsuleDocked.cpp File Reference

Implementation for CapsuleDocked.h.

```
#include "CapsuleDocked.h"
#include "CapsuleOffline.h"
#include "CapsuleState.h"
#include "RocketCapsule.h"
```

6.12.1 Detailed Description

Implementation for CapsuleDocked.h.

6.13 C:/Users/labuser2/Downloads/System/System/CapsuleDocked.h File Reference

Participant - Concrete State (State) Describes the methods of a Capusle that is in a 'docked' state.

```
#include "CapsuleState.h"
```

Classes

- class CapsuleDocked

6.13.1 Detailed Description

Participant - Concrete State (State) Describes the methods of a Capusle that is in a 'docked' state.

Author

The 6 Musekteers

6.14 CapsuleDocked.h

Go to the documentation of this file.

```
#ifndef CAPSULEDOCKED_H
#define CAPSULEDOCKED_H

#include "CapsuleState.h"

using namespace std;

class CapsuleDocked : public CapsuleState
{
    public:

        CapsuleDocked();

        string getState();

        CapsuleState* handleChange();
};

#endif
```

6.15 C:/Users/labuser2/Downloads/System/System/CapsuleOffline.cpp File Reference

Implementation for CapsuleOffline.h.

```
#include "CapsuleOffline.h"
#include "CapsuleDeparting.h"
```

6.15.1 Detailed Description

Implementation for CapsuleOffline.h.

6.16 C:/Users/labuser2/Downloads/System/System/CapsuleOffline.h File Reference

Participant - Concrete State (State) Describes the methods of a Capusle that is in a 'docked' state.

```
#include "CapsuleState.h"
```

Classes

- class CapsuleOffline

6.16.1 Detailed Description

Participant - Concrete State (State) Describes the methods of a Capusle that is in a 'docked' state.

Author

The 6 Musekteers

6.17 CapsuleOffline.h

Go to the documentation of this file.

```
#ifndef CAPSULEOFFLINE_H
#define CAPSULEOFFLINE_H

#include "CapsuleState.h"

using namespace std;

class CapsuleOffline : public CapsuleState
{
    public:
        CapsuleOffline();

        string getState();

        CapsuleState* handleChange();
};

#endif
```

6.18 C:/Users/labuser2/Downloads/System/System/CapsuleState.h File Reference

Participant - State (State) Describes the interface for the different states of a capsule.

```
#include "RocketCapsule.h"
```

Classes

- class CapsuleState

6.18.1 Detailed Description

Participant - State (State) Describes the interface for the different states of a capsule.

Author

The 6 Musekteers

6.19 CapsuleState.h

Go to the documentation of this file.

```
```

```
#ifndef CAPSULESTATE_H
#define CAPSULESTATE_H

#include "RocketCapsule.h"

using namespace std;
class CapsuleState
{
    public:
        virtual string getState() = 0;

        virtual CapsuleState* handleChange() = 0;
};



#endif
```

6.20 C:/Users/labuser2/Downloads/System/System/Caretaker.cpp File Reference

Implementation for Caretaker.h.

`#include "Caretaker.h"`

6.20.1 Detailed Description

Implementation for Caretaker.h.

6.21 C:/Users/labuser2/Downloads/System/System/Caretaker.h File Reference

Participant - Caretaker (Memento) Describes the class responsible for the safekeeping of the Memento class' state.

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

Classes

- class Caretaker

6.21.1 Detailed Description

Participant - Caretaker (Memento) Describes the class responsible for the safekeeping of the Memento class' state.

Author

The 6 Musketeers

6.22 Caretaker.h

Go to the documentation of this file.

```

#ifndef CARETAKER_H
#define CARETAKER_H

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

```

```
#include "Memento.h"

using namespace std;

class Caretaker {

private:
    vector<Memento*> store;
public:

    Caretaker();

    ~Caretaker();

    void storeMemento(Memento* m);

    Memento* retrieveMemento();

    int getSize();
};

#endif
```

6.23 C:/Users/labuser2/Downloads/System/System/CargoDragon.cpp File Reference

Implementation for CargoDragon.h.

`#include "CargoDragon.h"`

6.23.1 Detailed Description

Implementation for CargoDragon.h.

6.24 C:/Users/labuser2/Downloads/System/System/CargoDragon.h File Reference

Participant - Concrete Decorator (Decorator) Defines the attributes and methods for a RocketCapsule that carries Cargo.

`#include "RocketCapsule.h"`

Classes

- class CargoDragon

6.24.1 Detailed Description

Participant - Concrete Decorator (Decorator) Defines the attributes and methods for a RocketCapsule that carries Cargo.

Author

The 6 Musketeers

6.25 CargoDragon.h

Go to the documentation of this file.

```

#ifndef CARGODRAGON_H
#define CARGODRAGON_H

#include "RocketCapsule.h"

class CargoDragon : public RocketCapsule
{
    public:
        CargoDragon(Component* r);

        void simulate();

        void test();
};

#endif
```

6.26 C:/Users/labuser2/Downloads/System/System/CommNetwork.cpp File Reference

Implementation for CommNetwork.h.

```
#include "CommNetwork.h"
#include <iterator>
```

6.26.1 Detailed Description

Implementation for CommNetwork.h.

6.27 C:/Users/labuser2/Downloads/System/System/CommNetwork.h File Reference

Participant - Concrete Mediator (Mediator) Defines the attributes and methods for the class used for communication between the satellites.

```
#include <string>
#include <vector>
#include "Mediator.h"
#include "Satellite.h"
```

Classes

- class CommNetwork

6.27.1 Detailed Description

Participant - Concrete Mediator (Mediator) Defines the attributes and methods for the class used for communication between the satellites.

Author

The 6 Musketeers

6.28 CommNetwork.h

Go to the documentation of this file.

```

#ifndef COMMNETWORK_H
#define COMMNETWORK_H

#include <string>
#include <vector>
#include "Mediator.h"
#include "Satellite.h"

class CommNetwork : public Mediator {
    public:
        vector<Satellite*> colleagueList;
    public:
        CommNetwork(vector<Satellite*>);
        void notify(int sender);
        void sendMessage(int sender, int receiver, string msg);
};

#endif
```

6.29 C:/Users/labuser2/Downloads/System/System/Component.cpp File Reference

Implementation for Component.h.

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

6.29.1 Detailed Description

Implementation for Component.h.

6.30 C:/Users/labuser2/Downloads/System/System/Component.h File Reference

Participant - Component (Decorator), Component (Composite), Client (Chain of Responsibility), Product (Builder), Product (Factory Method), Client (Prototype), Implementor (Brdige) Defines the attributes and methods for Component objects.

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

Classes

- class Component

6.30.1 Detailed Description

Participant - Component (Decorator), Component (Composite), Client (Chain of Responsibility), Product (Builder), Product (Factory Method), Client (Prototype), Implementor (Brdige) Defines the attributes and methods for Component objects.

Author

The 6 Musketeers

6.31 Component.h

Go to the documentation of this file.

```

#ifndef COMPONENT_H
#define COMPONENT_H

#include <string>
#include <iostream>

using namespace std;

class Component
{
    protected:
        double cost;
        //string rocketType;    /**< The type of the rocket */
        //string capsuleType;   /**< The type of the capsule */

    public:
        Component(double c);

        virtual void simulate();

        virtual void test();

        virtual void add(Component* c);

        virtual void remove(int pos);

        virtual Component* getComponent(int pos);



        //Component* clone();




        double getCost();

        virtual void separate();

        virtual void fireMerlin();

        virtual void land();

         virtual int getSize();

         virtual void fireVacuumMerlin();
};

static float rocketCost;
#endif
```

6.32 C:/Users/labuser2/Downloads/System/System/Component←Composite.cpp File Reference

Implementation for ComponentComposite.h.

```
#include "ComponentComposite.h"
```

6.32.1 Detailed Description

Implementation for ComponentComposite.h.

6.33 C:/Users/labuser2/Downloads/System/System/Component←Composite.h File Reference

Participant - Handler (Chain of Responsibility) Defines an interface to handle the requests and implementatins of the Falcon9 and FalconHeavy components.

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

Classes

- class ComponentComposite

6.33.1 Detailed Description

Participant - Handler (Chain of Responsibility) Defines an interface to handle the requests and implementatins of the Falcon9 and FalconHeavy components.

Author

The 6 Muskateers

6.34 ComponentComposite.h

Go to the documentation of this file.

```
#ifndef COMPONENTCOMPOSITE_H
#define COMPONENTCOMPOSITE_H

#include "Component.h"
#include <vector>

class ComponentComposite : public Component
{
    private:
        vector<Component*> components;
    public:
        ComponentComposite();

        void simulate();

        void test();

        virtual void add(Component* c);

        virtual void remove(int pos);

        Component* getComponent(int pos);

        int getSize();

        void separate();

        void fireMerlin();

        void land();
};

#endif
```

6.35 C:/Users/labuser2/Downloads/System/System/ComponentCreator.h File Reference

Participant - Creator (Factory Method), Prototype (Prototype). Defines the interface creating the Merlin, Vacuum Merlin and Core Engines.

`#include "Component.h"`

Classes

- class ComponentCreator

6.35.1 Detailed Description

Participant - Creator (Factory Method), Prototype (Prototype). Defines the interface creating the Merlin, Vacuum Merlin and Core Engines.

Author

The 6 Musketeers

6.36 ComponentCreator.h

Go to the documentation of this file.

```
#ifndef COMPONENTCREATOR_H
#define COMPONENTCREATOR_H

#include "Component.h"

class ComponentCreator
{
    private:
        Component* component;
    public:
        virtual Component* factoryMethod() = 0;



        //virtual Component* clone(Component* C) = 0;
};

#endif
```

6.37 C:/Users/labuser2/Downloads/System/System/ConcreteRocket↩ Builder.cpp File Reference

Implementation for ConcreteRocketBuilder.h.

```
#include "ConcreteRocketBuilder.h"
#include "CommNetwork.h"
```

6.37.1 Detailed Description

Implementation for ConcreteRocketBuilder.h.

6.38 C:/Users/labuser2/Downloads/System/System/ConcreteRocket↩ Builder.h File Reference

Participant - Concrete Builder (Builder) Defines the methods and attributes of the class that builds a rocket.

```
#include "Component.h"
#include "Builder.h"
#include "ComponentCreator.h"
#include "CoreCreator.h"
#include "VacuumMerlinEngineCreator.h"
#include "MerlinEngineCreator.h"
#include "ComponentComposite.h"
#include "RocketCapsule.h"
#include "CrewDragon.h"
#include "CargoDragon.h"
#include "Fairing.h"
#include "SimulationState.h"
#include "SatelliteCreator.h"
#include "StarlinkCreator.h"
#include "User.h"
#include "Simulation.h"
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
```

Classes

- class ConcreteRocketBuilder

6.38.1 Detailed Description

Participant - Concrete Builder (Builder) Defines the methods and attributes of the class that builds a rocket.

Author

The 6 Musketeers

6.39 ConcreteRocketBuilder.h

Go to the documentation of this file.

```
#ifndef CONCRETEROCKETBUILDER_H
#define CONCRETEROCKETBUILDER_H

#include "Component.h"
#include "Builder.h"
#include "ComponentCreator.h"
#include "CoreCreator.h"
#include "VacuumMerlinEngineCreator.h"
#include "MerlinEngineCreator.h"
#include "ComponentComposite.h"
#include "RocketCapsule.h"
#include "CrewDragon.h"
#include "CargoDragon.h"
#include "Fairing.h"
#include "SimulationState.h"
#include "SatelliteCreator.h"
#include "StarlinkCreator.h"
#include "User.h"

#include "Simulation.h"

#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>

class ConcreteRocketBuilder : public Builder
{
    private:
        ComponentCreator* merlinCreator;
        ComponentCreator* vacuumMerlinCreator;
        ComponentCreator* coreCreator;
        SatelliteCreator* starlinkCreator;
        Component* rocket;
        RocketCapsule* capsule;
        SimulationState* simulationState;
    public:
        ConcreteRocketBuilder();

        Component* getSpacecraft();

        RocketCapsule* getCapsule();

        void buildFalcon9();

        void buildFalconHeavy();

        void constructCapsule(string type);

        Simulation* createSimulation();
};

#endif
```

6.40 C:/Users/labuser2/Downloads/System/System/CoreCreator.cpp File Reference

Implementation for Cor.Creatorh.

```
#include "CoreCreator.h"
```

6.40.1 Detailed Description

Implementation for Cor.Creatorh.

6.41 C:/Users/labuser2/Downloads/System/System/CoreCreator.h File Reference

Participant - Concrete Creator (Factory Method), ConcretePrototype (Prototype). Defines the methods and attributes of the class that builds FalconCore engines.

```
#include "ComponentCreator.h"
#include "Component.h"
#include "FalconCore.h"
```

Classes

- class CoreCreator

6.41.1 Detailed Description

Participant - Concrete Creator (Factory Method), ConcretePrototype (Prototype). Defines the methods and attributes of the class that builds FalconCore engines.

Author

The 6 Musketeers

6.42 CoreCreator.h

Go to the documentation of this file.

```
#ifndef CORECREATOR_H
#define CORECREATOR_H

#include "ComponentCreator.h"
#include "Component.h"
#include "FalconCore.h"

//Factory Method Concrete Creator
class CoreCreator : public ComponentCreator
{
    public:
        Component* factoryMethod();

        //Component* clone(Component* C);
};

#endif
```

6.43 C:/Users/labuser2/Downloads/System/System/CrewDragon.cpp File Reference

Implementation for CrewDragon.h.

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

6.43.1 Detailed Description

Implementation for CrewDragon.h.

6.44 C:/Users/labuser2/Downloads/System/System/CrewDragon.h File Reference

Participant - Concrete Decorator (Decorator) Defines the attributes and methods for a Capsule carrying crew members.

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

Classes

- class CrewDragon

6.44.1 Detailed Description

Participant - Concrete Decorator (Decorator) Defines the attributes and methods for a Capsule carrying crew members.

Author

The 6 Musketeers

6.45 CrewDragon.h

Go to the documentation of this file.

```
#ifndef CREWDRAGON_H
#define CREWDRAGON_H

#include "RocketCapsule.h"

#include <vector>

class CrewDragon : public RocketCapsule
{
    private:
        vector<string> passengers;
    public:
        CrewDragon(Component* r);

        void simulate();

        void test();

        vector<string> getPassengers();

        void setPassengers(vector<string> p);
};

#endif
```

6.46 C:/Users/labuser2/Downloads/System/System/Director.cpp File Reference

Implementation for Director.h.

```
#include "Director.h"
```

6.46.1 Detailed Description

Implementation for Director.h.

6.47 C:/Users/labuser2/Downloads/System/System/Director.h File Reference

Participant - Director (Builder) Defines the attributes and methods for the class that constructs rockets using the Builder interface.

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

```
#include "Builder.h"
```

Classes

- class Director

6.47.1 Detailed Description

Participant - Director (Builder) Defines the attributes and methods for the class that constructs rockets using the Builder interface.

Author

The 6 Musketeers

6.48 Director.h

Go to the documentation of this file.

```
#ifndef DIRECTOR_H
#define DIRECTOR_H

#include <string>
#include <iostream>

#include "Builder.h"

using namespace std;

class Director
{
    private:
        string type;
        Builder* builder;
    public:
        Director (Builder* b);

        ~Director();

        Component* construct();

        void constructCapsule();

        Simulation* createSimulation();
};
#endif //DIRECTOR_H
```

6.49 C:/Users/labuser2/Downloads/System/System/Facade.cpp File Reference

Implementation for Facade.h.

```
#include "Facade.h"
#include "ConcreteRocketBuilder.h"
#include "Director.h"
#include "Caretaker.h"
```

6.49.1 Detailed Description

Implementation for Facade.h.

6.50 C:/Users/labuser2/Downloads/System/System/Facade.h File Reference

Participant - Facade (Facade) Delegates client requests to appropriate subsystem objects.

```
#include "Simulation.h"
#include "Online.h"
#include "Offline.h"
#include "Broadcasting.h"
#include "RocketCapsule.h"
```

Classes

- class Facade

6.50.1 Detailed Description

Participant - Facade (Facade) Delegates client requests to appropriate subsystem objects.

Author

The 6 Musketeers

6.51 Facade.h

Go to the documentation of this file.

```
#ifndef FACADE_H
#define FACADE_H

#include "Simulation.h"
#include "Online.h"
#include "Offline.h"
#include "Broadcasting.h"
#include "RocketCapsule.h"

class Facade
{
        private:
                Simulation* simulation;
                Component* rocket;
        public:
                Facade();

                ~Facade();

                //Bridge (Simulation) subsystem

                void launch();

                void test();

                //Builder subsystem
                void build();

                //Memento subsystem

                void storeSimulation();

                void retrieveSimulation();

                void useCommNetwork();

                void separateBoosters();

                Component* getRocket();

                void fireMerlin();

                void fireVacuumMerlin();

                void runSimulation();

                void deliverCrew();

                void distributeSatellites();

                void staticFireTest();

                void jettisonFairing();

                void printSimulation();

                bool editSimulation();

                void retrieveAll();
};

#endif
```

6.52 C:/Users/labuser2/Downloads/System/System/Fairing.cpp File Reference

Implementation for Fairing.h.
`#include "Fairing.h"`

6.52.1 Detailed Description

Implementation for Fairing.h.

6.53 C:/Users/labuser2/Downloads/System/System/Fairing.h File Reference

Participant - Concrete Decorator (Decorator) Defines the attributes and methods for a RocketCapsule of type Fairing.

```
#include "RocketCapsule.h"
#include "Satellite.h"
#include <vector>
```

Classes

- class Fairing

6.53.1 Detailed Description

Participant - Concrete Decorator (Decorator) Defines the attributes and methods for a RocketCapsule of type Fairing.

Author

The 6 Musketeers

6.54 Fairing.h

Go to the documentation of this file.

```

#ifndef FAIRING_H
#define FAIRING_H

#include "RocketCapsule.h"
#include "Satellite.h"

#include <vector>

class Fairing : public RocketCapsule
{
    private:
        vector<Satellite*> satellites;
    public:
        Fairing(Component* r);

        void simulate();
        void test();

        vector<Satellite*> getSatellites();

        void setSatellites(vector<Satellite*> s);
        Satellite* getSatellite(int id);
};

#endif
```

6.55 C:/Users/labuser2/Downloads/System/System/FalconCore.cpp File Reference

Implementation for FalconCore.h.

```
#include "FalconCore.h"
```

6.55.1 Detailed Description

Implementation for FalconCore.h.

6.56 C:/Users/labuser2/Downloads/System/System/FalconCore.h File Reference

Participant - ConcreteComponent (Decorator), Leaf (Composite), ConcreteProduct (Factory Method), Concrete Implementor (Bridge) Defines the methods of the class that defines a FalconCore engine.

```
#include "Component.h"
```

Classes

- class FalconCore

6.56.1 Detailed Description

Participant - ConcreteComponent (Decorator), Leaf (Composite), ConcreteProduct (Factory Method), Concrete Implementor (Bridge) Defines the methods of the class that defines a FalconCore engine.

Author

The 6 Musketeers

6.57 FalconCore.h

Go to the documentation of this file.

```
#ifndef FALCONCORE_H
#define FALCONCORE_H

#include "Component.h"

class FalconCore : public Component
{
    public:
        FalconCore();

        void simulate();

        void test();

        void separate();

        void land();
};

#endif
```

6.58 C:/Users/labuser2/Downloads/System/System/main.cpp File Reference

Runs the program.

```
#include "Facade.h"
#include <stdio.h>
#include <unistd.h>
```

Functions

- int **main** (int argc, char **argv)

6.58.1 Detailed Description

Runs the program.

6.59 mainpage.h

```
1
```

6.60 C:/Users/labuser2/Downloads/System/System/Mediator.h File Reference

Participant - Mediator (Mediator) Defines the methods of the interface that enables communication between the different satellites.

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

Classes

- class Mediator

6.60.1 Detailed Description

Participant - Mediator (Mediator) Defines the methods of the interface that enables communication between the different satellites.

Author

The 6 Musketeers

6.61 Mediator.h

Go to the documentation of this file.

```
#ifndef MEDIATOR_H
#define MEDIATOR_H

#include <string>
#include <iostream>

#include "Satellite.h"
class Satellite;
using namespace std;

class Mediator {

private:
```

```
public:
    virtual void notify(int sender) = 0;

    virtual void sendMessage(int sender, int receiver, string msg) = 0;
};

#endif
```

6.62 C:/Users/labuser2/Downloads/System/System/Memento.cpp File Reference

Implementation for Memento.h.

`#include "Memento.h"`

6.62.1 Detailed Description

Implementation for Memento.h.

6.63 C:/Users/labuser2/Downloads/System/System/Memento.h File Reference

Participant - Memento (Memento) Defines the methods of the class that stores the state of the simulation of a rocket.

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

Classes

- class Memento

6.63.1 Detailed Description

Participant - Memento (Memento) Defines the methods of the class that stores the state of the simulation of a rocket.

Author

The 6 Musketeers

6.64 Memento.h

Go to the documentation of this file.

```

#ifndef MEMENTO_H
#define MEMENTO_H

#include <iostream>
#include <string>

#include "SimulationState.h"

using namespace std;

class Memento
{
    private:
        SimulationState* state;
    public:
        SimulationState* getState();

        void setState(SimulationState* c);

        ~Memento();
};

#endif
```

6.65 C:/Users/labuser2/Downloads/System/System/MerlinEngine.cpp File Reference

Implementation for MerlinEngine.h.

```
#include "MerlinEngine.h"
```

6.65.1 Detailed Description

Implementation for MerlinEngine.h.

6.66 C:/Users/labuser2/Downloads/System/System/MerlinEngine.h File Reference

Participant - ConcreteProduct (Factory Method), Concrete Implementor (Bridge). Defines the methods of the class that defines a Merlin engine.

```
#include "Component.h"
```

Classes

- class MerlinEngine

6.66.1 Detailed Description

Participant - ConcreteProduct (Factory Method), Concrete Implementor (Bridge). Defines the methods of the class that defines a Merlin engine.

Author

The 6 Musketeers

6.67 MerlinEngine.h

Go to the documentation of this file.

```

#ifndef MERLINENGINE_H
#define MERLINENGINE_H

#include "Component.h"

class MerlinEngine : public Component
{
    public:
        MerlinEngine();

        void simulate();

        void test();

        void fireMerlin();
};

#endif
```

6.68 C:/Users/labuser2/Downloads/System/System/MerlinEngine↩ Creator.cpp File Reference

Implementation for MerlinEngineCreator.h.

```
#include "MerlinEngineCreator.h"
```

6.68.1 Detailed Description

Implementation for MerlinEngineCreator.h.

6.69 C:/Users/labuser2/Downloads/System/System/MerlinEngine↩ Creator.h File Reference

Participant - ConcreteCreator (Factory Method), ConcretePrototype (Prototype) Defines the methods of the class that creates MerlinEngine objects.

```
#include "ComponentCreator.h"
#include "Component.h"
#include "MerlinEngine.h"
```

Classes

- class MerlinEngineCreator

6.69.1 Detailed Description

Participant - ConcreteCreator (Factory Method), ConcretePrototype (Prototype) Defines the methods of the class that creates MerlinEngine objects.

Author

The 6 Musketeers

6.70 MerlinEngineCreator.h

Go to the documentation of this file.

```
#ifndef MERLINENGINECREATOR_H
#define MERLINENGINECREATOR_H

#include "ComponentCreator.h"
#include "Component.h"
#include "MerlinEngine.h"

class MerlinEngineCreator : public ComponentCreator
{
    public:
        Component* factoryMethod();


        //Component* clone(Component* C);
};

#endif
```

6.71 C:/Users/labuser2/Downloads/System/System/Observer.h File Reference

Participant - Observer (Observer) Defines the methods of the abstract class that observes the state of a satellite.

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

Classes

- class Observer

6.71.1 Detailed Description

Participant - Observer (Observer) Defines the methods of the abstract class that observes the state of a satellite.

Author

The 6 Musketeers

6.72 Observer.h

Go to the documentation of this file.

```
#ifndef OBSERVER_H
#define OBSERVER_H

#include <iostream>
#include <string>

using namespace std;

class Observer {

public:
    virtual void update(int satelliteID, string status) = 0;
};

#endif
```

6.73 C:/Users/labuser2/Downloads/System/System/Offline.cpp File Reference

Implementation for Offline.h.

```
#include "Offline.h"
#include "Online.h"
```

6.73.1 Detailed Description

Implementation for Offline.h.

6.74 C:/Users/labuser2/Downloads/System/System/Offline.h File Reference

Participant - Concrete State (State). Describes the properties and methods of a Satellite in the 'Offline' state.

```
#include "SatelliteState.h"
```

Classes

- class Offline

6.74.1 Detailed Description

Participant - Concrete State (State). Describes the properties and methods of a Satellite in the 'Offline' state.

Author

The 6 Musketeers

6.75 Offline.h

Go to the documentation of this file.

```
#ifndef OFFLINE_H
#define OFFLINE_H

#include "SatelliteState.h"

using namespace std;

class Offline: public SatelliteState
{
    public:
        Offline();

        string getType();


        SatelliteState* handleChange();
};

#endif
```

6.76 C:/Users/labuser2/Downloads/System/System/Online.cpp File Reference

Implementation for Online.h.

```
#include "Online.h"
#include "Broadcasting.h"
```

6.76.1 Detailed Description

Implementation for Online.h.

6.77 C:/Users/labuser2/Downloads/System/System/Online.h File Reference

Participant - Concrete State (State). Describes the properties and methods of a Satellite in the 'Online' state.

```
#include "SatelliteState.h"
```

Classes

- class Online

6.77.1 Detailed Description

Participant - Concrete State (State). Describes the properties and methods of a Satellite in the 'Online' state.

Author

The 6 Musketeers

6.78 Online.h

Go to the documentation of this file.

```
#ifndef ONLINE_H
#define ONLINE_H

#include "SatelliteState.h"

using namespace std;
```

```

class Online : public SatelliteState
{
    public:

        Online();

        string getType();

        SatelliteState* handleChange();
};

#endif
```

6.79 C:/Users/labuser2/Downloads/System/System/RocketCapsule.cpp File Reference

Implementation for RocketCapsule.h.

```
#include "RocketCapsule.h"
#include "CapsuleOffline.h"
```

6.79.1 Detailed Description

Implementation for RocketCapsule.h.

6.80 C:/Users/labuser2/Downloads/System/System/RocketCapsule.h File Reference

Participant - Decorator (Decorator), Context (State) Describes the properties and methods of a RocketCapule that can be added to a rocket Component.

```
#include <vector>
#include "Component.h"
#include "CapsuleState.h"
#include "Satellite.h"
```

Classes

- class RocketCapsule

6.80.1 Detailed Description

Participant - Decorator (Decorator), Context (State) Describes the properties and methods of a RocketCapule that can be added to a rocket Component.

Author

The 6 Musketeers

6.81 RocketCapsule.h

Go to the documentation of this file.

```

#ifndef ROCKETCAPSULE_H
#define ROCKETCAPSULE_H

#include <vector>

#include "Component.h"
#include "CapsuleState.h"
#include "Satellite.h"

class CapsuleState;

class RocketCapsule : public Component
{
    protected:
        string capsuleType;
        CapsuleState* state;
    private:
        Component* rocket;
        double payloadWeight;
    public:
        RocketCapsule(Component* r);

        virtual void simulate() = 0;

        virtual void test() = 0;

        void addCapsule(Component* r);

        void requestStateChange();

        void setState(CapsuleState* s);

        double getPayloadWeight();

        void setPayloadWeight(double pw);

        virtual void setPassengers(vector<string> p){};

         virtual void setSatellites(vector<Satellite*> s){};

         virtual Satellite* getSatellite(int id);

         CapsuleState* getState();
};

#endif
```

6.82 C:/Users/labuser2/Downloads/System/System/Satellite.cpp File Reference

Implementation for Satellite.h.

```
#include "Satellite.h"
```

6.82.1 Detailed Description

Implementation for Satellite.h.

6.83 C:/Users/labuser2/Downloads/System/System/Satellite.h File Reference

Participant - ConcreteSubject (Observer), Colleague (Mediator), Context (State), Product (Factory Method). Describes the properties and methods of a Satellite object.

```
#include <iostream>
#include <vector>
#include <string>
#include "Mediator.h"
#include "Observer.h"
#include "Offline.h"
```

Classes

- class Satellite

6.83.1 Detailed Description

Participant - ConcreteSubject (Observer), Colleague (Mediator), Context (State), Product (Factory Method). Describes the properties and methods of a Satellite object.

Author

The 6 Muskateers

6.84 Satellite.h

Go to the documentation of this file.

```

#ifndef SATELLITE_H
#define SATELLITE_H

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

#include "Mediator.h"
#include "Observer.h"
#include "Offline.h"

using namespace std;
class SatelliteState;
class Mediator;
class Satellite
{
    protected:
        SatelliteState* satelliteState;
        Mediator* mediator;
        vector<Observer*> observerList;
        int ID;
    public:
        Satellite(int ID);

        ~Satellite();

        void changed();



        //Mediator* getMediator();


        int getID();

        void sendMessage(int id, string msg);

        void receiveMessage(int id, string msg);

        void setMediator(Mediator* m);

        Mediator* getMediator();

        void requestStateChange();

         void attach(Observer* o);

         void detach(Observer* o);

         void notify();

         virtual SatelliteState* getState();

         void setState(SatelliteState* s);

         virtual Satellite* clone(int id){ return NULL;}
};

#endif
```

6.85 C:/Users/labuser2/Downloads/System/System/SatelliteCreator.cpp File Reference

Implementation for SatelliteCreator.h.

```
#include "SatelliteCreator.h"
```

6.85.1 Detailed Description

Implementation for SatelliteCreator.h.

6.86 C:/Users/labuser2/Downloads/System/System/SatelliteCreator.h File Reference

Participant - Creator (Factory Method). Describes the properties and methods the abstract class that creates a Satellite.

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

Classes

- class SatelliteCreator

6.86.1 Detailed Description

Participant - Creator (Factory Method). Describes the properties and methods the abstract class that creates a Satellite.

Author

The 6 Musketeers

6.87 SatelliteCreator.h

Go to the documentation of this file.

```
#ifndef SATELLITECREATOR_H
#define SATELLITECREATOR_H

#include <iostream>
#include <string>

#include "StarlinkSatellite.h"

using namespace std;

class SatelliteCreator
{
    private:
        int count;
    public:
        SatelliteCreator();

        virtual Satellite* factoryMethod()=0;

        virtual void setIDCount(int id) = 0;
};

#endif
```

6.88 C:/Users/labuser2/Downloads/System/System/SatelliteState.h File Reference

Participant - State (State) Describes the interface for the different states of a Satellite.

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

Classes

- class SatelliteState

6.88.1 Detailed Description

Participant - State (State) Describes the interface for the different states of a Satellite.

Author

The 6 Musekteers

6.89 SatelliteState.h

Go to the documentation of this file.

```
#ifndef SATELLITESTATE_H
#define SATELLITESTATE_H

#include <string>
#include <iostream>

using namespace std;

class SatelliteState
{
    public:
        virtual string getType() = 0;

        virtual SatelliteState* handleChange() = 0;
};

#endif
```

6.90 C:/Users/labuser2/Downloads/System/System/Simulation.cpp File Reference

Implementation for Simulation.h.

```
#include "Simulation.h"
#include "CapsuleDocked.h"
#include "User.h"
#include "Online.h"
#include "Offline.h"
#include "Broadcasting.h"
#include <stdio.h>
#include <unistd.h>
```

Functions

- void **animateRocket** ()

Variables

- const char rocket [ ]

6.90.1 Detailed Description

Implementation for Simulation.h.

6.90.2 Variable Documentation

6.90.2.1 rocket

```
const char rocket[]
```

Initial value:

```
=
"           _\n\
          /^\\\n\
          |-|\n\
          | |\n\
          |S|\n\
          |P|\n\
          |A|\n\
          |C|\n\
          |E|\n\
          | |\n\
          |X|\n\
         /| |\\\n\
        / | | \\\n\
       |  | |  |\n\
        `-\"\"\"-`\n\
"
```

6.91 C:/Users/labuser2/Downloads/System/System/Simulation.h File Reference

Participant - Abstraction (Bridge) Describes the abstract class for running a simulation of a rocket launch.

```
#include "SimulationState.h"
#include "Component.h"
#include "Fairing.h"
#include "Memento.h"
#include "Satellite.h"
#include "SatelliteState.h"
#include "CapsuleArriving.h"
#include "CapsuleDeparting.h"
#include <cstring>
#include <vector>
```

Classes

- class Simulation

6.91.1 Detailed Description

Participant - Abstraction (Bridge) Describes the abstract class for running a simulation of a rocket launch.

Author

The 6 Musketeers

6.92 Simulation.h

Go to the documentation of this file.

```
1
9 #ifndef SIMULATION_H
10 #define SIMULATION_H
11
12 #include "SimulationState.h"
13 #include "Component.h"
14 #include "Fairing.h"
```

```
#include "Memento.h"
#include "Satellite.h"
#include "SatelliteState.h"
#include "CapsuleArriving.h"
#include "CapsuleDeparting.h"
#include <cstring>
#include <vector>

class Simulation {

private:
    SimulationState* simulationState;
    Component* rocket;
    RocketCapsule* capsule;
    vector<string> methodCalls;
public:
    Simulation(RocketCapsule* c,Component* r, SimulationState* s);

    RocketCapsule* getCapsule();

    Memento* createMemento();

    void restoreMemento(Memento* m);

    void staticFireTest();

    void launch();

    //virtual void tweakSimulation() = 0;

    void printSimulation();

    void jettisonFairing();

    void separateBoosters();

     void distributeSatellites();

     void deliverCrew();

     void sendMessage(int sender,int reciever,string message);

     void runSimulation();

     SimulationState* getState();

     void fireMerlin();

     void landBoosters();

     void fireVacuumMerlin();

     void changeSatelliteState(int id, SatelliteState* state);

     void addCall(string c);

     bool containsCall(string c);

     void updateSimulationState();

     void swapStage(int pos_1, int pos_2);

     void removeStage(int pos);

     int getSimulationSize();
 };

 #endif
```

6.93 C:/Users/labuser2/Downloads/System/System/SimulationState.cpp File Reference

Implementation for SimulationState.h.

```
#include "SimulationState.h"
```

6.93.1 Detailed Description

Implementation for SimulationState.h.

6.94 C:/Users/labuser2/Downloads/System/System/SimulationState.h File Reference

Participant - State (Memento) Describes the attributes and methods of a SimulationState object.

```
#include "Component.h"
#include "Satellite.h"
#include <string>
#include <iostream>
#include <vector>
```

Classes

- class SimulationState

6.94.1 Detailed Description

Participant - State (Memento) Describes the attributes and methods of a SimulationState object.

Author

The 6 Musketeers

6.95 SimulationState.h

Go to the documentation of this file.

```

#ifndef SIMULATIONSTATE_H
#define SIMULATIONSTATE_H

#include "Component.h"
#include "Satellite.h"

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

using namespace std;

class SimulationState {

private:
    Component* rocket;
    string capsuleType;
    string rocketType;
    double payloadWeight;
    vector<Satellite*> satellites;
    vector<string> passengers;
    vector<string> methodCalls;
public:
    SimulationState();

    string getCapsuleType();

    string getRocketType();

    double getPayloadWeight();

    vector<Satellite*> getSatellites();

    vector<string> getPassengers();
    vector<string> getMethodCalls();

    void setCapsuleType(string s);

    void setRocketType(string s);

    void setPayloadWeight(double d);

    void setSatellites(vector<Satellite*> s);

    void setPassengers(vector<string> p);

    void setMethodCalls(vector<string> c);
};

#endif
```

6.96 C:/Users/labuser2/Downloads/System/System/StarlinkCreator.cpp File Reference

Implementation for StarlinkCreator.h.
`#include "StarlinkCreator.h"`

6.96.1 Detailed Description

Implementation for StarlinkCreator.h.

6.97 C:/Users/labuser2/Downloads/System/System/StarlinkCreator.h File Reference

Participant - Concrete Creator (Factory Method) Describes the attributes and methods of the class that creates StarlinkSatellite objects.
`#include "SatelliteCreator.h"`
`#include "StarlinkSatellite.h"`

Classes

- class StarlinkCreator

6.97.1 Detailed Description

Participant - Concrete Creator (Factory Method) Describes the attributes and methods of the class that creates StarlinkSatellite objects.

Author

The 6 Musketeers

6.98 StarlinkCreator.h

Go to the documentation of this file.

```

#ifndef STARLINKCREATOR_H
#define STARLINKCREATOR_H

#include "SatelliteCreator.h"
#include "StarlinkSatellite.h"
class StarlinkCreator : public SatelliteCreator
{
    private:
        int IDcount;
    public:
        StarlinkCreator();

        Satellite* factoryMethod();

        void setIDCount(int count);
};

#endif
```

6.99 C:/Users/labuser2/Downloads/System/System/StarlinkSatellite.cpp File Reference

Implementation for StarlinkSatellite.h.

```
#include "StarlinkSatellite.h"
```

6.99.1 Detailed Description

Implementation for StarlinkSatellite.h.

6.100 C:/Users/labuser2/Downloads/System/System/StarlinkSatellite.h File Reference

Participant - Concrete Product (Factory Method) Describes the attributes and methods of StarlinkSatellite objects.

```
#include "Satellite.h"
#include "SatelliteState.h"
```

Classes

- class StarlinkSatellite

6.100.1 Detailed Description

Participant - Concrete Product (Factory Method) Describes the attributes and methods of StarlinkSatellite objects.

Author

The 6 Musketeers

6.101 StarlinkSatellite.h

Go to the documentation of this file.

```
#ifndef STARLINKSATELLITE_H
#define STARLINKSATELLITE_H

#include "Satellite.h"
#include "SatelliteState.h"

class StarlinkSatellite : public Satellite
{
    public:
        StarlinkSatellite(int ID);

        StarlinkSatellite(Satellite* s, int id);

        SatelliteState* getState();

        void setState(SatelliteState* s);

        Satellite* clone(int id);
};

#endif
```

6.102 C:/Users/labuser2/Downloads/System/System/User.cpp File Reference

Implementation for User.h.

```
#include "User.h"
```

6.102.1 Detailed Description

Implementation for User.h.

6.103 C:/Users/labuser2/Downloads/System/System/User.h File Reference

Participant - Concrete Observer (Observer) Describes the attributes and methods of class that observes the state of Satellite objects.

```
#include "Observer.h"
#include "SatelliteState.h"
#include "StarlinkSatellite.h"
```

Classes

- class User

6.103.1 Detailed Description

Participant - Concrete Observer (Observer) Describes the attributes and methods of class that observes the state of Satellite objects.

Author

The 6 Musketeers

6.104 User.h

Go to the documentation of this file.

```

#ifndef USER_H
#define USER_H

#include "Observer.h"
#include "SatelliteState.h"
#include "StarlinkSatellite.h"

static int IDcounter = 0;

class User : public Observer
{
    private:
        SatelliteState* satelliteState;
        Satellite* subject;
        int ID;

    public:
        User(Satellite* s);

        void update(int satelliteID, string status);
};

#endif
```

6.105 C:/Users/labuser2/Downloads/System/System/VacuumMerlin↩ Engine.cpp File Reference

Implementation for VacuumMerlinEngine.h.

```
#include "VacuumMerlinEngine.h"
```

6.105.1 Detailed Description

Implementation for VacuumMerlinEngine.h.

6.106 C:/Users/labuser2/Downloads/System/System/VacuumMerlin↵ Engine.h File Reference

Participant - Concrete Product (Factory Method) Describes the attributes and methods of a VacuumMerlinEngine object.

```
#include "Component.h"
```

Classes

- class VacuumMerlinEngine

6.106.1 Detailed Description

Participant - Concrete Product (Factory Method) Describes the attributes and methods of a VacuumMerlinEngine object.

Author

The 6 Musketeers

6.107 VacuumMerlinEngine.h

Go to the documentation of this file.

```
#ifndef VACUUMMERLINENGINE_H
#define VACUUMMERLINENGINE_H

#include "Component.h"

class VacuumMerlinEngine : public Component
{
    public:
        VacuumMerlinEngine();

        void simulate();

        void test();

        void fireVacuumMerlin();
};

#endif
```

6.108 C:/Users/labuser2/Downloads/System/System/VacuumMerlin↵ EngineCreator.h File Reference

Participant - ConcretePrototype (Prototpe), Concrete Implementor (Bridge), Concrete Product (Factory Method). Describes the attributes and methods of the class to create VacuumMerlinEngine objects.

```
#include "ComponentCreator.h"
#include "Component.h"
#include "VacuumMerlinEngine.h"
```

Classes

- class VacuumMerlinEngineCreator

6.108.1 Detailed Description

Participant - ConcretePrototype (Prototpe), Concrete Implementor (Bridge), Concrete Product (Factory Method). Describes the attributes and methods of the class to create VacuumMerlinEngine objects.

Author

The 6 Musketeers

6.109 VacuumMerlinEngineCreator.h

Go to the documentation of this file.

```
#ifndef VACUUMMERLINENGINECREATOR_H
#define VACUUMMERLINENGINECREATOR_H

#include "ComponentCreator.h"
#include "Component.h"
#include "VacuumMerlinEngine.h"

class VacuumMerlinEngineCreator : public ComponentCreator
{
    public:
        Component* factoryMethod();


        //Component* clone(Component* C);
};

#endif
```

Index