CommandPattern

Generated by Doxygen 1.9.4

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

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

Namespace Documentation

4.1 SadSapphicGames Namespace Reference

4.2 SadSapphicGames.CommandPattern Namespace Reference

Classes

- class AlreadyRunningException

  This exception indicates an AsyncCommand was executed but its task had yet to complete. Only one CommandTask per instance can be running at a time. Handled by CommandStream.
- class AsyncCommand

  The base class of all commands who's execute method involves asynchronous tasks
- class Command

  The base class of all commands
- class CommandStream

  This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.
- class CompositeCommand

  A Command that is composed of multiple child commands, all of which are executed together and leave one record in the CommandStream's history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark>
- interface IAsyncCommand

  This Interface of the AsyncCommand abstract class, It is strongly recommended you use the AsyncCommand class rather than implement this yourself unless you are very familiar with asynchronous programming
- interface ICommand

  This is the Interface of the Command abstract class, unless you are defining your own base type for commands you should probably inherit from Command over this
- interface IFailable

  Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the CommandStream's history
- class IrreversibleCompositeFailureException

  An exception that indicates a CompositeCommand is executed but one of its children failed and the composite cannot undo its executed commands
- interface IUndoable

  Indicates a command can be undone

- class NullCommand

  A Command that does nothing
- class NullCompositeCommand

  Like the NullCommand this is a composite command that does nothing, multiple times.
- class ReversibleCompositeFailureException

  An exception that indicates a CompositeCommand is executed but one of its children failed, however the composite was able to undo the commands it had executed
- class SimpleComposite

  A CompositeCommand created from a collection of Command's that cannot fail
- class SingletonCommandManager

  A singleton manager for a single-stream, out of the box implementation of the Command pattern. Once you understand how the package works it is highly recommended create your own CommandStream wrapper tailored to the needs of your project. Executes the next command in the CommandStream every frame.

Enumerations

- enum ExecuteCode {
  Success , Failure , QueueEmpty , CompositeFailure ,
  AwaitingCompletion , AlreadyRunning }

  The possible return values of CommandStream.TryExecuteNext()

4.2.1 Enumeration Type Documentation

4.2.1.1 ExecuteCode

`enum SadSapphicGames.CommandPattern.ExecuteCode`

The possible return values of CommandStream.TryExecuteNext()

Enumerator

Success	Top Command executed successfully
Failure	Top Command would fail
QueueEmpty	Command queue was empty
CompositeFailure	Top Command was a CompositeCommand that did not indicate it would fail but failed partway through execution - was reversible
AwaitingCompletion	Top command was an AsyncCommand that is awaiting completion
AlreadyRunning	Top command was an AsyncCommand but its task had already been started and hasn't been completed

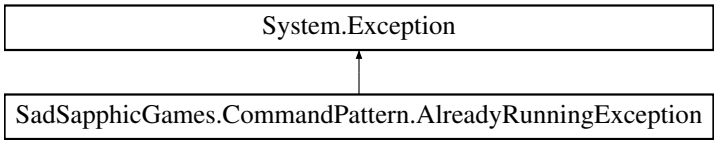
Chapter 5

Class Documentation

5.1 SadSapphicGames.CommandPattern.AlreadyRunningException Class Reference

This exception indicates an AsyncCommand was executed but its task had yet to complete. Only one Command↩ Task per instance can be running at a time. Handled by CommandStream.

Inheritance diagram for SadSapphicGames.CommandPattern.AlreadyRunningException:



Public Member Functions

- **AlreadyRunningException** (AsyncCommand command)
- **AlreadyRunningException** (string message)
- **AlreadyRunningException** (string message, System.Exception inner)

Protected Member Functions

- **AlreadyRunningException** (System.Runtime.Serialization.SerializationInfo info, System.Runtime.↩ Serialization.StreamingContext context)

5.1.1 Detailed Description

This exception indicates an AsyncCommand was executed but its task had yet to complete. Only one Command↩ Task per instance can be running at a time. Handled by CommandStream.

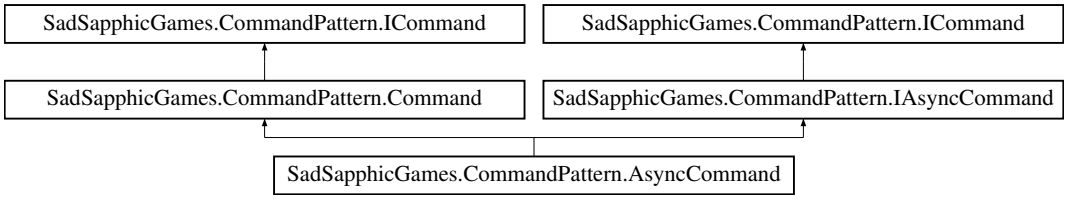
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/AsyncCommand.cs

5.2 SadSapphicGames.CommandPattern.AsyncCommand Class Reference

The base class of all commands who's execute method involves asynchronous tasks

Inheritance diagram for SadSapphicGames.CommandPattern.AsyncCommand:



Public Member Functions

- sealed override void Execute ()

  This method may not be overridden in asynchronous commands. Use ExecuteAsync for your command's logic instead.
- abstract Task ExecuteAsync ()

  Executes the command asynchronously. Remember to add the async keyword as that is not considered part of the method signature and cannot be added to abstract methods.
  IMPORTANT: if you want to be able to cancel the AsyncCommand's task you must invoke CancellationToken.Throw↩ IfCancellationRequested() within this method somewhere after the first await. If you do not do so attempting to cancel it will do nothing.

Protected Member Functions

- **AsyncCommand** ()

  Created the cancellation token of the async command and sets up its disposal once the task completes (wether that is from success, cancellation, or faulting)

Properties

- Task CommandTask `[get]`

  The task for the completion of the ExecuteAsync method after it reaches its first await and returns control back to the calling method (CommandStream.TryExecuteNext())
- CancellationToken CancellationToken `[get, set]`

  This can be used in ExecuteAsync to determine if the CommandTask has been canceled.

Events

- Action **OnTaskCompleted**

  This event is invoked when CommandTask is completed. I.E. - when we reach the end of ExecuteAsync(). Subscribe to it to preform an action at after the command has fully completed.
- Action **OnTaskCanceled**

  This event is invoked if the command task is canceled. Most bookkeeping that occurs in this situation is handled by the package but you can subscribe to this as well if needed.
- Action< Exception > **OnTaskFaulted**

  This event is invoked if the command task throws an exception.
- Action **OnAnyTaskEnd**

  This event is invoked when any of the OnTask[blank] events are to avoid needed to subscribe the same delegate to multiple events when logic needs to run regardless of how the task finished

5.2.1 Detailed Description

The base class of all commands who's execute method involves asynchronous tasks

5.2.2 Member Function Documentation

5.2.2.1 Execute()

```
sealed override void SadSapphicGames.CommandPattern.AsyncCommand.Execute ( )  [virtual]
```

This method may not be overridden in asynchronous commands. Use ExecuteAsync for your command's logic instead.

Implements SadSapphicGames.CommandPattern.Command.

5.2.2.2 ExecuteAsync()

```
abstract Task SadSapphicGames.CommandPattern.AsyncCommand.ExecuteAsync ( )  [pure virtual]
```

Executes the command asynchronously. Remember to add the async keyword as that is not considered part of the method signature and cannot be added to abstract methods.
IMPORTANT: if you want to be able to cancel the AsyncCommand's task you must invoke CancellationToken.↩ ThrowIfCancellationRequested() within this method somewhere after the first await. If you do not do so attempting to cancel it will do nothing.

Returns

The task for the completion of this method after it reaches its first await and returns control to the calling method.

Implements SadSapphicGames.CommandPattern.IAsyncCommand.

5.2.3 Property Documentation

5.2.3.1 CancellationToken

```
CancellationToken SadSapphicGames.CommandPattern.AsyncCommand.CancellationToken  [get], [set]
```

This can be used in ExecuteAsync to determine if the CommandTask has been canceled.

Implements SadSapphicGames.CommandPattern.IAsyncCommand.

5.2.3.2 CommandTask

```
Task SadSapphicGames.CommandPattern.AsyncCommand.CommandTask  [get]
```

The task for the completion of the ExecuteAsync method after it reaches its first await and returns control back to the calling method (CommandStream.TryExecuteNext())

Implements SadSapphicGames.CommandPattern.IAsyncCommand.

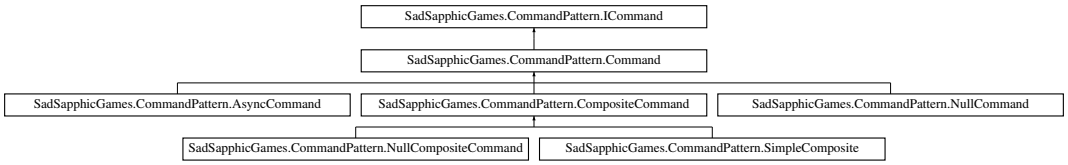
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/AsyncCommand.cs

5.3 SadSapphicGames.CommandPattern.Command Class Reference

The base class of all commands

Inheritance diagram for SadSapphicGames.CommandPattern.Command:



Public Member Functions

- abstract void Execute ()

  Executes the command, do not invoke directly, instead use a CommandStream.

5.3.1 Detailed Description

The base class of all commands

5.3.2 Member Function Documentation

5.3.2.1 Execute()

```
abstract void SadSapphicGames.CommandPattern.Command.Execute ( )  [pure virtual]
```

Executes the command, do not invoke directly, instead use a CommandStream.

Implements SadSapphicGames.CommandPattern.ICommand.

Implemented in SadSapphicGames.CommandPattern.AsyncCommand, SadSapphicGames.CommandPattern.CompositeCommand, and SadSapphicGames.CommandPattern.NullCommand.

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

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/Command.cs

5.4 SadSapphicGames.CommandPattern.CommandStream Class Reference

This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.

Public Member Functions

- ReadOnlyCollection< ICommand > GetCommandHistory ()

  Get the CommandStream's history of executed Commands, a linear time operation if HistoryCount has reached HistoryDepth

- ReadOnlyCollection< ICommand > GetCommandQueue ()

  Get the queue of commands to be executed by the command stream.

- ReadOnlyCollection< Task > GetRunningCommandTasks ()

  Get the Tasks being being run by AsyncCommands executed by this CommandStream

- CancellationTokenSource GetRunningTaskCTS (Task task)

  Gets the CancellationTokenSource of a running AsyncCommand's task

- CancellationTokenSource **GetRunningTaskCTS** (IAsyncCommand asyncCommand)
- void CancelRunningCommandTask (Task task)

  Cancels a task if that task is currently running

- void CancelRunningCommandTask (IAsyncCommand asyncCommand)

  Cancels the task of an IAsyncCommand if that task is currently running.

- ReadOnlyDictionary< Task, Exception > GetFaultedCommandTasks ()

  Get a dictionary of any faulted tasks and the exceptions that they threw

- CommandStream (float _historyDepth=Single.PositiveInfinity)

  Creates a new CommandStream

- void QueueCommand (ICommand command)

  This adds a Command to the command Queue

- void QueueCommands (IEnumerable< ICommand > commands)

  Adds multiple Commands to the queue

- List< ICommand > DropQueue ()

  This will remove all commands from the CommandStream's queue and replace it with a new empty queue.

- ReadOnlyCollection< ICommand > DropHistory ()

  This will remove all commands from the CommandStream's history and replace it with a new empty list.

- bool TryQueueUndoCommand (IUndoable undoable)

  Attempt to queue's the undo command of a Command object implementing IUndoable if that command exists in this CommandStream's history

- void ForceQueueUndoCommand (IUndoable undoable)

  Force the stream to queue's the undo command of a Command object implementing IUndoable regardless of whether the command is recorded in this CommandStream's history

- bool TryPeekNext (out ICommand nextCommand)

  Examine the next command in the commandQueue with out executing it

- void **RequeueNextCommand** ()

  Removes the next command in the CommandStream's queue, if it has one, and adds it back to the end of the queue.

- void **SkipNextCommand** ()

  Removes the next command from the queue without executing it

- ExecuteCode TryExecuteNext (out ICommand topCommand)

  Attempts to execute the next command in the queue, returns an enum indicating if it was able to or not or if the queue was empty or the command is async and awaiting completion.

- ExecuteCode TryExecuteImmediate (ICommand command)

  Bypass the command queue and immediately attempt to execute a command
- ExecuteCode TryUndoImmediate (IUndoable undoable)

  Bypass the command queue and immediately attempt to execute an IUndoable's undo command if the IUndoable is in the CommandStream's history
- ExecuteCode ForceTryUndoImmediate (IUndoable undoable)

  Bypass the command queue and immediately attempt to execute an IUndoable's undo command, regardless of wether the IUndoable is in the CommandStream's history
- ExecuteCode TryExecuteNext ()

  Attempts to execute the next command in the queue, returns false if it is empty or the command is IFailable and would fail.
- void **ExecuteFullQueue** ()

  Execute's Commands from the CommandStream queue until it is empty. Be warned this will not give any indication of commands failing.
- void ExecuteFullQueue (out List< ICommand > failedCommands)

  Executes Commands from the CommandStream's queue until it is empty. Returns the a list of any Commands that failed as an out parameter

Properties

- bool **QueueEmpty** `[get]`

  Gets commandQueue.Count == 0
- float **HistoryDepth** `[get]`

  this is the maximum number of commands that will be recorded in the CommandHistory
- int **HistoryCount** `[get]`

  Gets the number of Commands currently recorded in the CommandStream's history.
- int **QueueCount** `[get]`

  Gets the number of Commands currently waiting in the CommandStream's queue.

Events

- Action< Exception > **OnTaskFaulted**

  This event will be invoked if one of the tasks from an IAsyncCommand executed by this stream faults. Can be used to throw any exception's caused by tasks rather than storing them on the task object

5.4.1 Detailed Description

This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 CommandStream()

```
SadSapphicGames.CommandPattern.CommandStream.CommandStream (
            float _historyDepth = Single.PositiveInfinity )
```

Creates a new CommandStream

Parameters

_historyDepth	The depth to which previously executed commands will be recorded. Once this depth is reached the oldest commands will be forgotten first. <remark> To not record history, set to zero. To never forget executed commands, set to positive infinity. </remark>

5.4.3 Member Function Documentation

5.4.3.1 CancelRunningCommandTask() [1/2]

```
void SadSapphicGames.CommandPattern.CommandStream.CancelRunningCommandTask (
            IAsyncCommand asyncCommand )
```

Cancels the task of an IAsyncCommand if that task is currently running.

Parameters

asyncCommand	The IAsyncCommand to cancel the task off

5.4.3.2 CancelRunningCommandTask() [2/2]

```
void SadSapphicGames.CommandPattern.CommandStream.CancelRunningCommandTask (
            Task task )
```

Cancels a task if that task is currently running

Parameters

task	The task to cancel

5.4.3.3 DropHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.CommandStream.DropHistory ( )
```

This will remove all commands from the CommandStream's history and replace it with a new empty list.

<remark> This can be useful if you need the CommandStream to record all of its history but also need it to execute an extremely large number of commands without running out of memory. Even if every command is the same object a CommandStream will run out of memory at 2-3 hundred million commands in its queue or history. </remark>

Returns

The commands in the previous history, in case this information is needed

5.4.3.4 DropQueue()

```
List< ICommand > SadSapphicGames.CommandPattern.CommandStream.DropQueue ( )
```

This will remove all commands from the CommandStream's queue and replace it with a new empty queue.

<remark> This can be useful to rearrange the commands in a queue. Simple preform the needed changes on the returned list and re-queue it </remark>

Returns

The commands in the previous queue, in case this information is needed.

5.4.3.5 ExecuteFullQueue()

```
void SadSapphicGames.CommandPattern.CommandStream.ExecuteFullQueue (
            out List< ICommand > failedCommands )
```

Executes Commands from the CommandStream's queue until it is empty. Returns the a list of any Commands that failed as an out parameter

Parameters

failedCommands	A list of any Commands in the Queue that failed to execute

5.4.3.6 ForceQueueUndoCommand()

```
void SadSapphicGames.CommandPattern.CommandStream.ForceQueueUndoCommand (
            IUndoable undoable )
```

Force the stream to queue's the undo command of a Command object implementing IUndoable regardless of whether the command is recorded in this CommandStream's history

<remark>This is equivalent to passing the result of IUndoable.GetUndoCommand() into CommandStream.↩ QueueCommand(Command command) directly</remark>

Parameters

undoable	The IUndoable Command to queue the undo-Command of

5.4.3.7 ForceTryUndoImmediate()

```
ExecuteCode SadSapphicGames.CommandPattern.CommandStream.ForceTryUndoImmediate (
            IUndoable undoable )
```

Bypass the command queue and immediately attempt to execute an IUndoable's undo command, regardless of wether the IUndoable is in the CommandStream's history

Parameters

undoable	the IUndoable to execute the undo command of

Returns

The ExecuteCode of the attempt to execute the undo command of the IUndoable

5.4.3.8 GetCommandHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.CommandStream.GetCommandHistory
( )
```

Get the CommandStream's history of executed Commands, a linear time operation if HistoryCount has reached HistoryDepth

Returns

The history of executed commands, null if history is not recorded.

5.4.3.9 GetCommandQueue()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.CommandStream.GetCommandQueue (
)
```

Get the queue of commands to be executed by the command stream.

Returns

The queue of commands the commandStream will execute.

5.4.3.10 GetFaultedCommandTasks()

```
ReadOnlyDictionary< Task, Exception > SadSapphicGames.CommandPattern.CommandStream.GetFaulted↩
CommandTasks ( )
```

Get a dictionary of any faulted tasks and the exceptions that they threw

Returns

A read only copy of the dictionary of faulted tasks and their exceptions

5.4.3.11 GetRunningCommandTasks()

```
ReadOnlyCollection< Task > SadSapphicGames.CommandPattern.CommandStream.GetRunningCommandTasks
( )
```

Get the Tasks being being run by AsyncCommands executed by this CommandStream

Returns

The tasks that are currently being run

5.4.3.12 GetRunningTaskCTS()

```
CancellationTokenSource SadSapphicGames.CommandPattern.CommandStream.GetRunningTaskCTS (
            Task task )
```

Gets the CancellationTokenSource of a running AsyncCommand's task

Parameters

task	The task to get the CTS of

Returns

The CTS of task if it is running, null if it is not

5.4.3.13 QueueCommand()

```
void SadSapphicGames.CommandPattern.CommandStream.QueueCommand (
            ICommand command )
```

This adds a Command to the command Queue

Parameters

command	The Command to be Queued

5.4.3.14 QueueCommands()

```
void SadSapphicGames.CommandPattern.CommandStream.QueueCommands (
            IEnumerable< ICommand > commands )
```

Adds multiple Commands to the queue

Parameters

commands	The commands to be queued

5.4.3.15 TryExecuteImmediate()

```
ExecuteCode SadSapphicGames.CommandPattern.CommandStream.TryExecuteImmediate (
            ICommand command )
```

Bypass the command queue and immediately attempt to execute a command

Parameters

command	The command to immediately be executed

Returns

The ExecuteCode for the attempt to execute the command

5.4.3.16 TryExecuteNext() [1/2]

```
ExecuteCode SadSapphicGames.CommandPattern.CommandStream.TryExecuteNext ( )
```

Attempts to execute the next command in the queue, returns false if it is empty or the command is IFailable and would fail.

Returns

False if the command queue is empty, or the next command would fail. True otherwise.

5.4.3.17 TryExecuteNext() [2/2]

```
ExecuteCode SadSapphicGames.CommandPattern.CommandStream.TryExecuteNext (
            out ICommand topCommand )
```

Attempts to execute the next command in the queue, returns an enum indicating if it was able to or not or if the queue was empty or the command is async and awaiting completion.

Parameters

topCommand	The command that was next in the queue, null if the queue was empty

Returns

An ExecuteCode enum value indicating what happened when attempting to execute the next command

5.4.3.18 TryPeekNext()

```
bool SadSapphicGames.CommandPattern.CommandStream.TryPeekNext (
            out ICommand nextCommand )
```

Examine the next command in the commandQueue with out executing it

Parameters

nextCommand	The next command in the queue, null if empty

Returns

Wether there was a command in queue or not

5.4.3.19 TryQueueUndoCommand()

```
bool SadSapphicGames.CommandPattern.CommandStream.TryQueueUndoCommand (
            IUndoable undoable )
```

Attempt to queue's the undo command of a Command object implementing IUndoable if that command exists in this CommandStream's history

Parameters

undoable	The IUndoable Command to try and queue the undo-Command of

Returns

Wether the undo command was queued

5.4.3.20 TryUndoImmediate()

```
ExecuteCode SadSapphicGames.CommandPattern.CommandStream.TryUndoImmediate (
            IUndoable undoable )
```

Bypass the command queue and immediately attempt to execute an IUndoable's undo command if the IUndoable is in the CommandStream's history

Parameters

undoable	the IUndoable to execute the undo command of

Returns

The ExecuteCode of the attempt to execute the undo command of the IUndoable, Execute.Failure if the IUndoable was not in the CommandStream's history

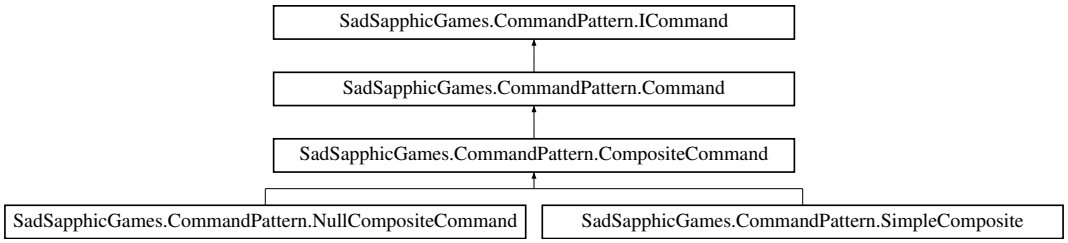
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/CommandStream.cs

5.5 SadSapphicGames.CommandPattern.CompositeCommand Class Reference

A Command that is composed of multiple child commands, all of which are executed together and leave one record in the CommandStream's history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark>

Inheritance diagram for SadSapphicGames.CommandPattern.CompositeCommand:



Public Member Functions

- override void Execute ()

  Queues all of the child commands into the internal CommandStream and attempts to invoke all of them. Will throw an exception if one of its children fails after attempting to revert all its executed commands.

Protected Member Functions

- virtual void AddChild (ICommand childCommand)

  Adds a Command to this objects children

Protected Attributes

- List< ICommand > **subCommands** = new List<ICommand>()

  The child Commands that will be executed upon executing this object
- CommandStream **internalStream** = new CommandStream()

  An internal CommandStream to provide more control of the execution of the subCommands of the Composite

Properties

- int **ChildCount** `[get]`

  Number of child Commands included in this object

5.5.1 Detailed Description

A Command that is composed of multiple child commands, all of which are executed together and leave one record in the CommandStream's history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark>

5.5.2 Member Function Documentation

5.5.2.1 AddChild()

```
virtual void SadSapphicGames.CommandPattern.CompositeCommand.AddChild (
            ICommand childCommand )  [protected], [virtual]
```

Adds a Command to this objects children

Parameters

childCommand	The Command to be added to the objects children

5.5.2.2 Execute()

```
override void SadSapphicGames.CommandPattern.CompositeCommand.Execute ( )  [virtual]
```

Queues all of the child commands into the internal CommandStream and attempts to invoke all of them. Will throw an exception if one of its children fails after attempting to revert all its executed commands.

Be aware that if you override this method you will bypass the implemented failsafe's for children of the CompositeCommand failing such as attempting to undo executed commands

Exceptions

IrreversibleCompositeFailureException	Indicates one of the children of the CompositeCommand failed and it executed one or more commands that cannot be undone. TryExecuteNext will catch this exception and throw it upwards
ReversibleCompositeFailureException	Indicates one of the children of the CompositeCommand failed but it was able to undo all of its executed commands. TryExecuteNext will catch this exception and handle it by returning false.

Implements SadSapphicGames.CommandPattern.Command.

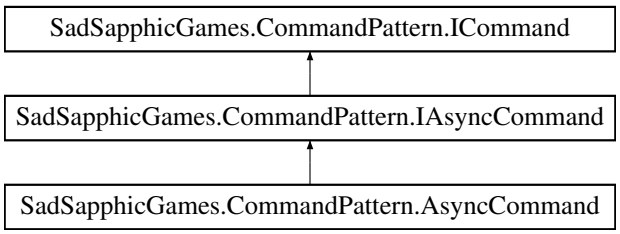
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/CompositeCommand.cs

5.6 SadSapphicGames.CommandPattern.IAsyncCommand Interface Reference

This Interface of the AsyncCommand abstract class, It is strongly recommended you use the AsyncCommand class rather than implement this yourself unless you are very familiar with asynchronous programming

Inheritance diagram for SadSapphicGames.CommandPattern.IAsyncCommand:



Public Member Functions

- abstract Task ExecuteAsync ()

  This is where the logic of executing the command should be placed for an AsyncCommand, Execute should just store the return in CommandTask and setup the OnTaskCompleted method. Remember to make this method async as that isn't considered part of its signature.

Properties

- Task CommandTask `[get]`

  This should get the asynchronous task returned by ExecuteAsync after it reaches its first await
- CancellationToken CancellationToken `[get, set]`

  This can be used to Cancel the task after it has been started.

Events

- Action **OnTaskCompleted**

  This event should be invoked when CommandTask is successfully completed.
- Action **OnTaskCanceled**

  This event should be invoked when CommandTask is cancelled.
- Action< Exception > **OnTaskFaulted**

  This event should be invoked when CommandTask throws an exception
- Action **OnAnyTaskEnd**

  This event should be invoked when any of the above three are

5.6.1 Detailed Description

This Interface of the AsyncCommand abstract class, It is strongly recommended you use the AsyncCommand class rather than implement this yourself unless you are very familiar with asynchronous programming

5.6.2 Member Function Documentation

5.6.2.1 ExecuteAsync()

```
abstract Task SadSapphicGames.CommandPattern.IAsyncCommand.ExecuteAsync ( )  [pure virtual]
```

This is where the logic of executing the command should be placed for an AsyncCommand, Execute should just store the return in CommandTask and setup the OnTaskCompleted method. Remember to make this method async as that isn't considered part of its signature.

Returns

The Task representing the completion of the method after it reaches its first await statement

Implemented in SadSapphicGames.CommandPattern.AsyncCommand.

5.6.3 Property Documentation

5.6.3.1 CancellationToken

```
CancellationToken SadSapphicGames.CommandPattern.IAsyncCommand.CancellationToken  [get], [set]
```

This can be used to Cancel the task after it has been started.

Implemented in SadSapphicGames.CommandPattern.AsyncCommand.

5.6.3.2 CommandTask

```
Task SadSapphicGames.CommandPattern.IAsyncCommand.CommandTask  [get]
```

This should get the asynchronous task returned by ExecuteAsync after it reaches its first await

Implemented in SadSapphicGames.CommandPattern.AsyncCommand.

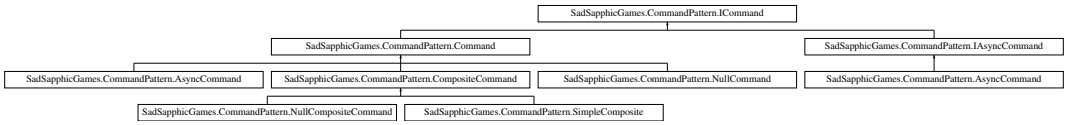
The documentation for this interface was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Interfaces/IAsyncCommand.cs

5.7 SadSapphicGames.CommandPattern.ICommand Interface Reference

This is the Interface of the Command abstract class, unless you are defining your own base type for commands you should probably inherit from Command over this

Inheritance diagram for SadSapphicGames.CommandPattern.ICommand:



Public Member Functions

- abstract void Execute ()

  Executes the function of the command

5.7.1 Detailed Description

This is the Interface of the Command abstract class, unless you are defining your own base type for commands you should probably inherit from Command over this

5.7.2 Member Function Documentation

5.7.2.1 Execute()

```
abstract void SadSapphicGames.CommandPattern.ICommand.Execute ( )  [pure virtual]
```

Executes the function of the command

Implemented in SadSapphicGames.CommandPattern.AsyncCommand, SadSapphicGames.CommandPattern.Command, SadSapphicGames.CommandPattern.CompositeCommand, and SadSapphicGames.CommandPattern.NullCommand.

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

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Interfaces/ICommand.cs

5.8 SadSapphicGames.CommandPattern.IFailable Interface Reference

Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the CommandStream's history

Public Member Functions

- bool WouldFail ()

  Determine if the implementing command would be able to be executed or if it would fail

5.8.1 Detailed Description

Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the CommandStream's history

5.8.2 Member Function Documentation

5.8.2.1 WouldFail()

```
bool SadSapphicGames.CommandPattern.IFailable.WouldFail ( )
```

Determine if the implementing command would be able to be executed or if it would fail

Returns

True if the implementing command would fail, false if it would execute successfully.

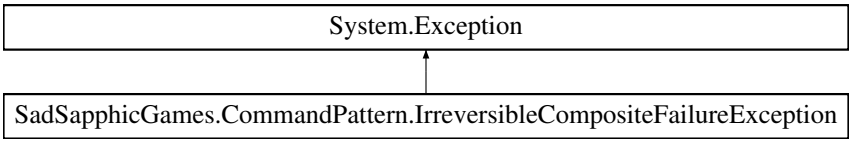
The documentation for this interface was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Interfaces/IFailable.cs

5.9 SadSapphicGames.CommandPattern.IrreversibleCompositeFailure↩ Exception Class Reference

An exception that indicates a CompositeCommand is executed but one of its children failed and the composite cannot undo its executed commands

Inheritance diagram for SadSapphicGames.CommandPattern.IrreversibleCompositeFailureException:



Public Member Functions

- **IrreversibleCompositeFailureException** (ICommand failedCommand, List< ICommand > irreversibleCommands)

Public Attributes

- readonly ICommand **failedCommand**

  The child command that failed
- readonly List< ICommand > **irreversibleCommands**

  The executed commands that could not be undone

5.9.1 Detailed Description

An exception that indicates a CompositeCommand is executed but one of its children failed and the composite cannot undo its executed commands

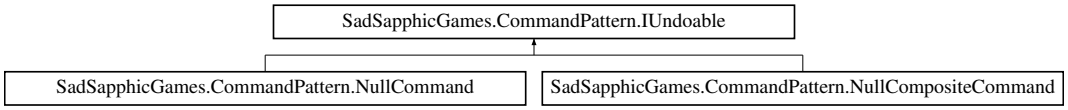
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/CompositeCommand.cs

5.10 SadSapphicGames.CommandPattern.IUndoable Interface Reference

Indicates a command can be undone

Inheritance diagram for SadSapphicGames.CommandPattern.IUndoable:



Public Member Functions

- ICommand GetUndoCommand ()

  Creates a command to revert the changes of the implementing command

5.10.1 Detailed Description

Indicates a command can be undone

5.10.2 Member Function Documentation

5.10.2.1 GetUndoCommand()

```
ICommand SadSapphicGames.CommandPattern.IUndoable.GetUndoCommand ( )
```

Creates a command to revert the changes of the implementing command

Returns

a command that reverts the implementing command

Implemented in SadSapphicGames.CommandPattern.NullCommand, and SadSapphicGames.CommandPattern.NullCompositeComm

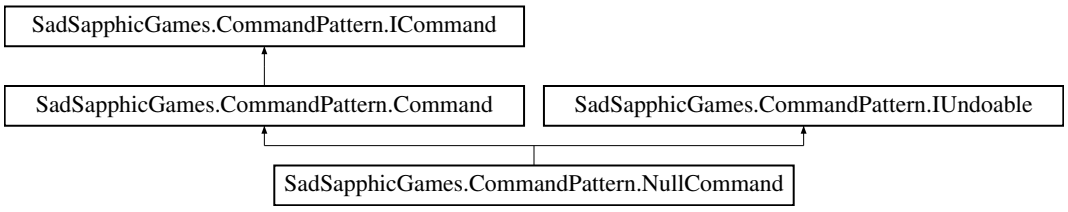
The documentation for this interface was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Interfaces/IUndoable.cs

5.11 SadSapphicGames.CommandPattern.NullCommand Class Reference

A Command that does nothing

Inheritance diagram for SadSapphicGames.CommandPattern.NullCommand:



Public Member Functions

- override void Execute ()

  Does Nothing
- ICommand GetUndoCommand ()

  Since the command doesn't do anything it returns itself

5.11.1 Detailed Description

A Command that does nothing

5.11.2 Member Function Documentation

5.11.2.1 Execute()

```
override void SadSapphicGames.CommandPattern.NullCommand.Execute ( )  [virtual]
```

Does Nothing

Implements SadSapphicGames.CommandPattern.Command.

5.11.2.2 GetUndoCommand()

```
ICommand SadSapphicGames.CommandPattern.NullCommand.GetUndoCommand ( )
```

Since the command doesn't do anything it returns itself

Returns

The same null command object

Implements SadSapphicGames.CommandPattern.IUndoable.

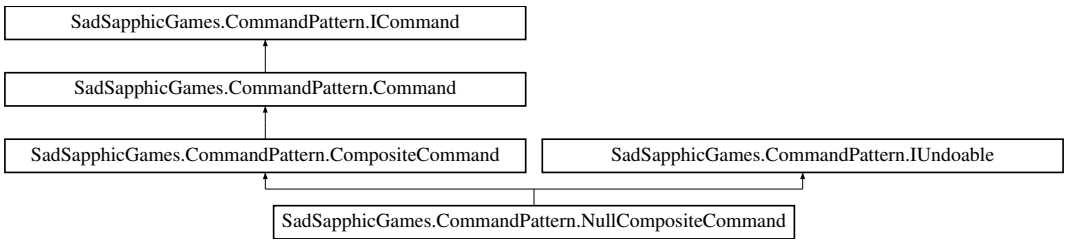
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/NullCommand.cs

5.12 SadSapphicGames.CommandPattern.NullCompositeCommand Class Reference

Like the NullCommand this is a composite command that does nothing, multiple times.

Inheritance diagram for SadSapphicGames.CommandPattern.NullCompositeCommand:



Public Member Functions

- NullCompositeCommand (int size)

  Creates a NullCompositeCommand composed of multiple NullCommands
- ICommand GetUndoCommand ()

  Like the NullCommand it is composed of, a NullCompositeCommand is its own undo-command

Additional Inherited Members

5.12.1 Detailed Description

Like the NullCommand this is a composite command that does nothing, multiple times.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 NullCompositeCommand()

```
SadSapphicGames.CommandPattern.NullCompositeCommand.NullCompositeCommand (
            int size )
```

Creates a NullCompositeCommand composed of multiple NullCommands

Parameters

size	How many NullCommands to include in the composite

5.12.3 Member Function Documentation

5.12.3.1 GetUndoCommand()

```
ICommand SadSapphicGames.CommandPattern.NullCompositeCommand.GetUndoCommand ( )
```

Like the NullCommand it is composed of, a NullCompositeCommand is its own undo-command

Returns

This object

Implements SadSapphicGames.CommandPattern.IUndoable.

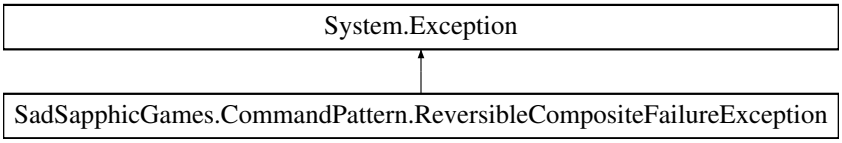
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/NullCompositeCommand.cs

5.13 SadSapphicGames.CommandPattern.ReversibleCompositeFailure↩ Exception Class Reference

An exception that indicates a CompositeCommand is executed but one of its children failed, however the composite was able to undo the commands it had executed

Inheritance diagram for SadSapphicGames.CommandPattern.ReversibleCompositeFailureException:



Public Member Functions

- **ReversibleCompositeFailureException** (ICommand failedCommand)

Public Attributes

- readonly ICommand **failedCommand**

  The child command that failed

5.13.1 Detailed Description

An exception that indicates a CompositeCommand is executed but one of its children failed, however the composite was able to undo the commands it had executed

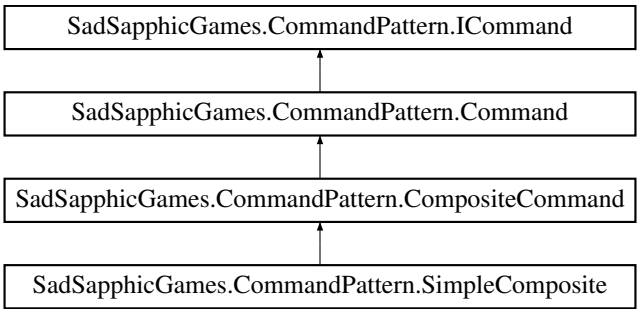
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/CompositeCommand.cs

5.14 SadSapphicGames.CommandPattern.SimpleComposite Class Reference

A CompositeCommand created from a collection of Command's that cannot fail

Inheritance diagram for SadSapphicGames.CommandPattern.SimpleComposite:



Public Member Functions

- SimpleComposite (IEnumerable< Command > subCommands)

  Creates a SimpleComposite from a collection of Command's that cannot fail

Additional Inherited Members

5.14.1 Detailed Description

A CompositeCommand created from a collection of Command's that cannot fail

5.14.2 Constructor & Destructor Documentation

5.14.2.1 SimpleComposite()

```
SadSapphicGames.CommandPattern.SimpleComposite.SimpleComposite (
            IEnumerable< Command > subCommands )
```

Creates a SimpleComposite from a collection of Command's that cannot fail

Parameters

subCommands	The collection of unfailable Commands to be included in the composite

Exceptions

System.ArgumentException	One or more of the Commands included in the argument implement IFailable

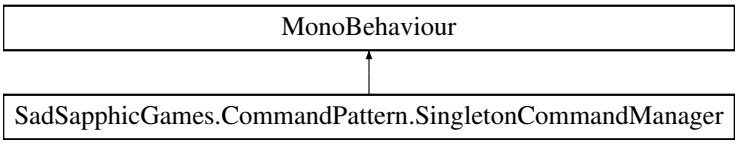
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Commands/SimpleComposite.cs

5.15 SadSapphicGames.CommandPattern.SingletonCommandManager Class Reference

A singleton manager for a single-stream, out of the box implementation of the Command pattern. Once you understand how the package works it is highly recommended create your own CommandStream wrapper tailored to the needs of your project. Executes the next command in the CommandStream every frame.

Inheritance diagram for SadSapphicGames.CommandPattern.SingletonCommandManager:



Public Member Functions

- void **ToggleCommandExecution** ()

  Turns command execution off if its on and on if its off
- void ToggleCommandExecution (bool onoff)

  Turns command execution on or off
- ReadOnlyCollection< ICommand > GetCommandHistory ()

  Get the underlying CommandStream's history
- ReadOnlyCollection< Task > GetRunningCommandTasks ()

  Get the currently uncompleted tasks from executed AsyncCommands
- void CancelRunningCommandTask (Task taskToCancel)

  Cancels an AsyncCommand's running task through a reference to the task
- void CancelRunningCommandTask (IAsyncCommand asyncCommand)

  Cancels an AsyncCommand's running task through a reference to the command
- ReadOnlyCollection< ICommand > DropCommandHistory ()

  Empties the history of the internal CommandStream and replaces it with an empty one.
- void QueueCommand (ICommand command)

  Queue's a Command into the CommandManager's CommandStream
- void QueueCommands (IEnumerable< ICommand > commands)

  Queue's multiple commands into the CommandManager's CommandStream
- bool TryQueueUndoCommand (IUndoable commandToUndo)

  Queue the undo-command of a Command implementing IUndoable into the CommandStream
- void ForceQueueUndoCommand (IUndoable commandToUndo)

  Forces the internal CommandStream to queue and IUndoable commands undo command

Public Attributes

- int **maximumHistoryDepth** = -1

  The value that will be used in the internal CommandStream's constructor, set to negative to record all history

Properties

- static SingletonCommandManager **Instance** `[get]`

  The singleton instance of the CommandManger.
- int **HistoryCount** `[get]`

  The number of Commands recorded by the CommandManager's CommandStream
- float **HistoryDepth** `[get]`

  The depth to which the CommandManager's CommandStream records its history
- int **QueueCount** `[get]`

  The Number of commands queued in the CommandManager's CommandStream
- bool **QueueEmpty** `[get]`

  Wether or not the CommandManger's CommandStream has an empty queue

5.15.1 Detailed Description

A singleton manager for a single-stream, out of the box implementation of the Command pattern. Once you understand how the package works it is highly recommended create your own CommandStream wrapper tailored to the needs of your project. Executes the next command in the CommandStream every frame.

5.15.2 Member Function Documentation

5.15.2.1 CancelRunningCommandTask() [1/2]

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.CancelRunningCommandTask (
            IAsyncCommand asyncCommand )
```

Cancels an AsyncCommand's running task through a reference to the command

Parameters

taskToCancel	the AsyncCommand who's task should be canceled

5.15.2.2 CancelRunningCommandTask() [2/2]

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.CancelRunningCommandTask (
            Task taskToCancel )
```

Cancels an AsyncCommand's running task through a reference to the task

Parameters

taskToCancel	the task of an AsyncCommand to cancel

5.15.2.3 DropCommandHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.SingletonCommandManager.Drop↩
CommandHistory ( )
```

Empties the history of the internal CommandStream and replaces it with an empty one.

Returns

The old history of the internal CommandStream

5.15.2.4 ForceQueueUndoCommand()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.ForceQueueUndoCommand (
            IUndoable commandToUndo )
```

Forces the internal CommandStream to queue and IUndoable commands undo command

Parameters

commandToUndo	The IUndoable command to undo

5.15.2.5 GetCommandHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.SingletonCommandManager.Get↩
CommandHistory ( )
```

Get the underlying CommandStream's history

Returns

A ReadOnlyCollection of all the commands executed by the CommandManager's CommandStream

5.15.2.6 GetRunningCommandTasks()

```
ReadOnlyCollection< Task > SadSapphicGames.CommandPattern.SingletonCommandManager.GetRunning↩
CommandTasks ( )
```

Get the currently uncompleted tasks from executed AsyncCommands

Returns

A ReadOnlyCollection of uncompleted tasks from executed AsyncCommands

5.15.2.7 QueueCommand()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.QueueCommand (
            ICommand command )
```

Queue's a Command into the CommandManager's CommandStream

Parameters

command	The Command to be Queued

5.15.2.8 QueueCommands()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.QueueCommands (
            IEnumerable< ICommand > commands )
```

Queue's multiple commands into the CommandManager's CommandStream

Parameters

commands	The collection of commands to be Queued

5.15.2.9 ToggleCommandExecution()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.ToggleCommandExecution (
            bool onoff )
```

Turns command execution on or off

Parameters

onoff	if false stops the execution of commands, if true enables it

5.15.2.10 TryQueueUndoCommand()

```
bool SadSapphicGames.CommandPattern.SingletonCommandManager.TryQueueUndoCommand (
            IUndoable commandToUndo )
```

Queue the undo-command of a Command implementing IUndoable into the CommandStream

Parameters

commandToUndo	The IUndoable Command to queue an undo-command for

Returns

Wether the undo command was allowed to be queued

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

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/Command↩ Pattern/Runtime/Monobehaviours/SingletonCommandManager.cs

Index