Contents

Chapter 1

Package de.tum.in.net.WSNDataFramework

Package Contents *Page*

Classes

1.1 Classes

1.1.1 Class **FileStorage**

Declaration

public class FileStorage **extends** java.lang.Object

Constructors

- *FileStorage*
  `public` **FileStorage**`( java.io.File` **file** `)`

Methods

- *get*
  `public Serializable` **get**`( java.lang.String` **key**`, java.lang.Class` **type** `)`
- *reload*
  `public FileStorage` **reload**`( )`
- *save*
  `public FileStorage` **save**`( java.io.Serializable` **obj**`, java.lang.String` **key** `)`

1.1.2 Class **WSN**

WSN Manager

Declaration

public class WSN **extends** de.tum.in.net.WSNDataFramework.Event.EventProvider

Constructors

- *WSN*
  `public` **WSN**`( )`
  - **Usage**
    - uses the "'current working directory'/wsnconfig" as fileStorageDirectory. "'current working directory'/wsnconfig/tmp" as temporaryFilesDirectory.
  - **See Also**

  - `de.tum.in.net.WSNDataFramework.WSN.WSN(String fileStorageDirectory, String temporaryFilesDirectory)`

- *WSN*
  `public` **WSN**`( java.lang.String` **fileStorageDirectory**`, java.lang.String` **temporaryFilesDirectory** `)`
  - **Usage**
    - Creates a WSN instance using the given directories as file storage.
  - **Parameters**
    - `fileStorageDirectory` - directory where to store files created by this WSN and its modules.

Methods

- *_addModuleEvents*
  `protected void` **_addModuleEvents**`( de.tum.in.net.WSNDataFramework.WSNModule` **m** `)`
  - **Usage**
    - adds the events provided by module m to this WSN's provided events list
  - **Parameters**
    - `m` -

- *_eventRegistrationAllowed*
  `protected boolean` **_eventRegistrationAllowed**`( java.lang.Class` **eventClass** `)`
  - **Usage**
    - override EventProvider._eventRegistrationAllowed to allow subscriber to register for any event. Makes sure that Modules can register for events not known at the moment (to allow class A to register for an event fired by module B that will get loaded in future).
  - **Parameters**
    - `eventClass` -

- *_moduleEventOccurred*
  `protected void` **_moduleEventOccurred**`(`
  `de.tum.in.net.WSNDataFramework.Event.Event` **e** `)`
  - **Usage**
    - callback for all events thrown by any module. Rethrows any event for subscriber of this WSN
  - **Parameters**
    - `e` -

- *_updateDynamicProvidedEvents*
  `protected void` **_updateDynamicProvidedEvents**`( )`
  - **Usage**

    * updates this._dynamicProvidedEvents. may be called whenever a module was added or removed

---

- *addModule*
  `public synchronized WSN` **addModule**`( de.tum.in.net.WSNDataFramework.WSNModule` **m** `)`
  - **Usage**
    * Adds a new module. Has the same effect as calling m.setWSN(this).
  - **Parameters**
    * `m` - module
  - **Returns** - this for fluent interface

- *addNode*
  `public WSN` **addNode**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    * Adds a copy of the given Node to this WSN and fires a WSNNodeAddedEvent. If a node with the same ID is already present this method forwards to WSN.updateNode().
  - **Parameters**
    * `node` -
  - **Returns** - this for fluent interface
  - **Exceptions**
    * `java.lang.CloneNotSupportedException` - if node couldn't be cloned correctly!
  - **See Also**
    * `de.tum.in.net.WSNDataFramework.Events.WSNNodeAddedEvent`

- *createTemporaryFile*
  `public File` **createTemporaryFile**`( )`
  - **Usage**
    * Tries to create a unique temporary file (stored in getTemporaryFilesDirectory()). be sure to delete the file after it isn't needed anymore.
  - **Returns** - file handle

- *getFile*
  `public File` **getFile**`( java.lang.String` **filename** `)`
  - **Usage**
    * Gets a handler to the file called 'filename' within this.getFilesDirectory(). File gets created if it does not exist.
  - **Parameters**
    * `filename` -
  - **Returns** - file handle — NULL

- *getFilesDirectory*
  `public String` **getFilesDirectory**`( )`
  - **Usage**

  - ∗ Gets the path to the directory where files by this WSN and its modules are stored.

- *getIdentity*
  `public WSN.Identity` **getIdentity**`( )`
  - **Usage**
    - ∗ Gets the Identity of this WSN. May be used to compare two WSN identities to recognize a WSN and load WSN specific configurations etc.. Can perfectly be serialized!
  - **Returns** - WSN.Identity

- *getModule*
  `public WSNModule` **getModule**`( java.lang.Class` **moduleClass** `)`
  - **Usage**
    - ∗ Returns a specific loaded module.
  - **Parameters**
    - ∗ `moduleClass` -
  - **Returns** - module instance — NULL if not found

- *getModules*
  `public WSNModule` **getModules**`( )`
  - **Usage**
    - ∗ Returns a list of all loaded modules.

- *getNode*
  `public WSNNode` **getNode**`( java.lang.String` **nodeID** `)`
  - **Usage**
    - ∗ Gets a copy of the node with the given nodeID.
  - **Parameters**
    - ∗ `nodeID` -
  - **Returns** - WSNNode — NULL if nodeID not found

- *getNodes*
  `public Map` **getNodes**`( )`
  - **Usage**
    - ∗ Returns a list of all the nodes assigned to this WSN. Returns a copy of the actual list for thread safe access. To make adoptions to the actual nodes list use WSN.updateNode().
  - **Returns** - Map(nodeID =>Node)

- *getProvidedEvents*
  `public Class` **getProvidedEvents**`( )`
  - **Usage**
    - ∗ returns events provided by this WSN as well as by all registered modules

---

- *getTemporaryFilesDirectory*
  `public String` **getTemporaryFilesDirectory**`( )`
  - **Usage**
    - Gets path to the directory for temporary files.

- *getTopology*
  `public WSNTopology` **getTopology**`( )`
  - **Usage**
    - Gets a copy of the WSN's current topology.
Returns *null* if no topology is known!
  - **Returns** - current WSNTopology or *null*

- *isShutdown*
  `public boolean` **isShutdown**`( )`
  - **Usage**
    - checks if WSN is shut down
  - **Returns** - true if WSN is shut down

- *isShuttingDown*
  `public boolean` **isShuttingDown**`( )`
  - **Usage**
    - checks if WSN is currently shutting down
  - **Returns** - true if WSN is shutting down

- *removeModule*
  `public synchronized WSN` **removeModule**`(`
  `de.tum.in.net.WSNDataFramework.WSNModule` **m** `)`
  - **Usage**
    - Removes a module from the WSN. Has the same effect as calling m.shutdown()
  - **Parameters**
    - `m` - module
  - **Returns** - this for fluent interface

- *removeNode*
  `public WSN` **removeNode**`( java.lang.String` **nodeID** `)`
  - **Usage**
    - removes a Node from this WSN and fires a WSNNodeRemovedEvent.
  - **Parameters**
    - `nodeID` -
  - **Returns** - this for fluent interface
  - **See Also**
    - `de.tum.in.net.WSNDataFramework.Events.WSNNodeRemovedEvent`

- *setFilesDirectory*
  `public WSN` **setFilesDirectory**`( java.lang.String` **path** `)`

  - **Usage**
    - sets the path to the directory where files by this WSN and its modules are stored. Must exist and be writable.
  - **Parameters**
    - `path` -
  - **Returns** - this for fluent interface
  - **Exceptions**
    - `de.tum.in.net.WSNDataFramework.WSNException` - if directory not writable!

- *setTemporaryFilesDirectory*
  `public WSN` **setTemporaryFilesDirectory**`( java.lang.String` **path** `)`
  - **Usage**
    - Sets the path to the directory for temporary files.
  - **Parameters**
    - `path` -
  - **Returns** - this for fluent interface

- *shutdown*
  `public synchronized WSN` **shutdown**`( )`
  - **Usage**
    - shuts the WSN down a shut down WSN has terminated all its modules and threads and refuses all further interaction (a shut down WSN instance can not be reused) this method doesn't block. to wait for final shutdown use waitForShutdown()
  - **Returns** - this for fluent interface

- *subscribeEvent*
  `public boolean` **subscribeEvent**`( java.lang.Class` **eventClass**`,`
  `de.tum.in.net.WSNDataFramework.Event.EventBuffer` **eventBuffer** `)`
  - **Usage**
    - subscribes to an event that gets fired by this WSN or any of its modules. Allows to register for any event so you can register for an event currently not known but provided by a module loaded in future.
  - **See Also**
    - `de.tum.in.net.WSNDataFramework.Event.EventProvider.subscribeEvent`

- *updateNode*
  `public WSN` **updateNode**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    - Replaces a present node that has the same nodeID as the given one with it and fires a WSNNodeUpdatedEvent. Does nothing if no corresponding node was found.
  - **Parameters**
    - `node` -
  - **Returns** - this for fluent interface
  - **Exceptions**
    - `java.lang.CloneNotSupportedException` - if node couldn't be cloned correctly

- *updateTopology*
  `public WSN` **updateTopology**`( de.tum.in.net.WSNDataFramework.WSNTopology` **topology** `)`
  - **Usage**
    - Replaces WSN's current topology with a copy of the given one and fires WSNTopologyUpdatedEvent.
  - **Parameters**
    - `topology` -
  - **Exceptions**
    - `java.lang.CloneNotSupportedException` - if topology couldn't be cloned correctly.

- *waitForShutdown*
  `public WSN` **waitForShutdown**`( )`
  - **Usage**
    - blocks until WSN is shut down
  - **Returns** - this for fluent interface
  - **Exceptions**
    - `java.lang.InterruptedException` -

Methods inherited from class `de.tum.in.net.WSNDataFramework.Event.EventProvider`

- *_eventRegistrationAllowed*
  `protected boolean` **_eventRegistrationAllowed**`( java.lang.Class` **eventClass** `)`
  - **Usage**
    - checks if this EventProvider shall allow a registration for a specific event. Just maps to EventProvider.providesEvent() but may be overridden by subclass.
  - **Parameters**
    - `eventClass` -

- *fireEvent*
  `protected EventProvider` **fireEvent**`( de.tum.in.net.WSNDataFramework.Event.Event` **eve** `)`
  - **Usage**
    - fire specific event. Adds the fired event to all EventBuffers registered for it or any of its parent classes.
  - **Parameters**
    - `eve` - Event to fire
  - **Returns** - this for fluent interface

- *getProvidedEvents*
  `public Class` **getProvidedEvents**`( )`
  - **Usage**
    - get list of events offered by this provider. Uses reflection to determine events actually provided by this class.

- *providesEvent*
  `public boolean` **providesEvent**`( java.lang.Class` **event** `)`
  - **Usage**

  - ∗ checks if a specific event is provided. (considers inheritance, if provider provides (TestEvent extends Event) a check for Event would also return true)
  - **Parameters**
    - ∗ `event` -

- *subscribeEvent*
  `public boolean` **subscribeEvent**`( java.lang.Class` **eventClass**`, de.tum.in.net.WSNDataFramework.Event.EventBuffer` **eventBuffer** `)`
  - **Usage**
    - ∗ Subscribes to a specific event. Adds an instance of the fired event to eventBuffer each time it is fired.
  - **Parameters**
    - ∗ `eventClass` -
    - ∗ `eventBuffer` -

- *unsubscribeEvent*
  `public EventProvider` **unsubscribeEvent**`( java.lang.Class` **eventClass**`, de.tum.in.net.WSNDataFramework.Event.EventBuffer` **eventBuffer** `)`
  - **Usage**
    - ∗ Unsubscribes from a previously subscribed event.
  - **Parameters**
    - ∗ `eventClass` -
    - ∗ `eventBuffer` -

1.1.3 Class **WSN.Identity**

Representing the Identity of a WSN. Can be used to compare two WSNs regarding their structure. Can perfectly be serialized!

Compares the WSNs regarding their nodes to see if they are similar.. May be used to recognize an already seen WSN to load specific configurations etc..

Declaration

public static class WSN.Identity
extends java.lang.Object
implements java.io.Serializable

Constructors

- *WSN.Identity*
  `public` **WSN.Identity**`( de.tum.in.net.WSNDataFramework.WSN` **wsn** `)`
  - **Usage**
    - ∗ constructs the WSN.Identity of a given WSN.
  - **Parameters**
    - ∗ `wsn` -

Methods

- *compare*
  `public WSN.Identity.CompareResult` **compare(**
  `de.tum.in.net.WSNDataFramework.WSN.Identity` **identity)**
  - **Usage**
    - Compares the given identity to this Identity. Returns a Identity.CompareResult that specifies if theses identities may be considered the same as well as an degree how certain this decision was.
      Considers the nodes of the WSNs.
  - **Parameters**
    - `identity` -

1.1.4 Class WSN.Identity.CompareResult

representing the result of compare()

Declaration

public static class WSN.Identity.CompareResult
extends java.lang.Object

Fields

- public final boolean same
  –
- public final float certainty
  –

Constructors

- *WSN.Identity.CompareResult*
  `public` **WSN.Identity.CompareResult(** `boolean` **same,** `float` **certainty)**

1.1.5 Class WSNAggregationNode

represents a WSNNode that is capable of doing aggregation.

Declaration

public class WSNAggregationNode
extends de.tum.in.net.WSNDataFramework.WSNNode

Serializable Fields

- public List aggregatedNodes
  - a list of the Nodes this aggregation nodes aggregates.

Fields

- public List aggregatedNodes
  - a list of the Nodes this aggregation nodes aggregates.

Constructors

- *WSNAggregationNode*
  `public` **WSNAggregationNode**`( java.lang.String` **nodeID** `)`
  - **Usage**
    - constructor.
  - **Parameters**
    - `nodeID` -

- *WSNAggregationNode*
  `public` **WSNAggregationNode**`(`
  `de.tum.in.net.WSNDataFramework.WSNAggregationNode` **node** `)`
  - **Usage**
    - copy constructor. creates a deep copy (also copies its Fields + aggregatedNodes). Field.value is not cloned, so be careful using mutable types as Field.values.
  - **Parameters**
    - `node` -

- *WSNAggregationNode*
  `public` **WSNAggregationNode**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    - copy constructor. creates a deep copy (also copies its Fields). Field.value is not cloned, so be careful using mutable types as Field.values.
  - **Parameters**
    - `node` -

METHODS

- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`

- *same*
  `public boolean` **same**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    - Determines whether two nodes may be considered the same. Compares WSNNode.same() style and checks additionally if aggregatedNodes are the same.

- *updateAggregatedNode*
  `public WSNAggregationNode` **updateAggregatedNode**`(`
  `de.tum.in.net.WSNDataFramework.WSNNode` **aggregatedNode** `)`
  - **Usage**
    - Updates a single aggregated node. Adds it or replaces it if already present with the given node and updates WSNNode.updatedAt.
  - **Parameters**
    - `aggregatedNode` -
  - **Returns** - this for fluent interface

- *updateAggregatedNodes*
  `public WSNAggregationNode` **updateAggregatedNodes**`( java.util.List` **aggregatedNodes** `)`
  - **Usage**
    - replaces this node's aggregatedNodes by given list and updates WSNNode.updatedAt.
  - **Parameters**
    - `aggregatedNodes` -
  - **Returns** - this for fluent interface
  - **Exceptions**
    - `java.lang.CloneNotSupportedException` - if aggregatedNodes couldn't be cloned correctly

METHODS INHERITED FROM CLASS `de.tum.in.net.WSNDataFramework.WSNNode`

(in 1.1.10, page 22)

- *clone*
  `public WSNNode` **clone**`( )`
  - **Usage**
    - clones a WSNNode using the copy constructor of the actual class
  - **Exceptions**
    - `java.lang.CloneNotSupportedException` - if actual class doesn't offer a copy constructor!

- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`

  - **Usage**
    * Determines whether two nodes are exact the same. Checks all public fields for equality.

- *same*
  `public boolean` **same**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    * Determines whether two nodes may be considered the same. Compares ID + field entries (two field entries are the same if all their fields are equal except their value).

- *updateField*
  `public WSNNode` **updateField**`( de.tum.in.net.WSNDataFramework.WSNNode.Field` **Field** `)`
  - **Usage**
    * Updates a single Field. Adds it or replaces it if already present with the given Field and updates WSNNode.updatedAt.
  - **Parameters**
    * `Field` -
  - **Returns** - this for fluent interface

- *updateFields*
  `public WSNNode` **updateFields**`( java.util.List` **fields** `)`
  - **Usage**
    * updates this node's field list. replaces it by the given list and updates WSNNode.updatedAt.
  - **Parameters**
    * `fields` -
  - **Returns** - this for fluent interface

- *updateLocation*
  `public WSNNode` **updateLocation**`( de.tum.in.net.WSNDataFramework.WSNNode.Location` **location** `)`
  - **Usage**
    * Updates this node's location. replaces it by the given one and updates WSNNode.updatedAt.
  - **Parameters**
    * `location` -
  - **Returns** - this for fluent interface

1.1.6 Class **WSNException**

Represents an Exception thrown by the WSN. Wrapper around java.lang.Exception.

Declaration

public class WSNException
extends java.lang.Exception

CONSTRUCTORS

- *WSNException*
  `public` **WSNException**`( )`
- *WSNException*
  `public` **WSNException**`( java.lang.String` **message** `)`
- *WSNException*
  `public` **WSNException**`( java.lang.String` **message**`, java.lang.Throwable` **cause** `)`
- *WSNException*
  `public` **WSNException**`( java.lang.Throwable` **cause** `)`

METHODS INHERITED FROM CLASS `java.lang.Exception`

METHODS INHERITED FROM CLASS `java.lang.Throwable`

- *addSuppressed*
  `public final synchronized void` **addSuppressed**`( java.lang.Throwable` **arg0** `)`
- *fillInStackTrace*
  `public synchronized Throwable` **fillInStackTrace**`( )`
- *getCause*
  `public synchronized Throwable` **getCause**`( )`
- *getLocalizedMessage*
  `public String` **getLocalizedMessage**`( )`
- *getMessage*
  `public String` **getMessage**`( )`
- *getStackTrace*
  `public StackTraceElement` **getStackTrace**`( )`
- *getSuppressed*
  `public final synchronized Throwable` **getSuppressed**`( )`
- *initCause*
  `public synchronized Throwable` **initCause**`( java.lang.Throwable` **arg0** `)`
- *printStackTrace*
  `public void` **printStackTrace**`( )`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintStream` **arg0** `)`
- *printStackTrace*
  `public void` **printStackTrace**`( java.io.PrintWriter` **arg0** `)`
- *setStackTrace*
  `public void` **setStackTrace**`( java.lang.StackTraceElement []` **arg0** `)`
- *toString*
  `public String` **toString**`( )`

1.1.7 CLASS **WSNModule**

abstract parent class for WSN-Modules

Declaration

public abstract class WSNModule
extends de.tum.in.net.WSNDataFramework.Event.EventProvider

Constructors

- *WSNModule*
  `public` **WSNModule**`( )`

Methods

- *_getConfig*
  `protected FileStorage` **_getConfig**`( )`
  - **Usage**
    - Returns a FileStorage handler for this WSNModule's configuration data. FileStorage links to a file within the WSN's file directory. Filename: "moduleconfig.moduleName.moduleClassHash".
  - **Returns** - FileStorage for configuration data — NULL if not connected to a WSN

- *_init*
  `protected void` **_init**`( )`
  - **Usage**
    - handler that is called everytime the module is connected to a new WSN

- *_moduleDependent*
  `protected WSNModule` **_moduleDependent**`( java.lang.Class` **moduleClass**`, java.lang.String` **addCallback** `)`
  - **Usage**
    - Execute module dependent code. addCallback will be called when the module is loaded, remCallback when it is removed. This method may only be called when connected to a WSN =>call in _init()!
      Callbacks are called with an instance of the given module. (but they may be declared without parameters). If more than one fitting method is found (e.g. a method taking the actual module as argument as well as a method taking a WSNModule as argument) all methods are called starting with the most specific one.
  - **Parameters**
    - `module` -
    - `addCb` -
  - **Returns** - this for fluent interface

- *_moduleDependent*
  `protected WSNModule` **_moduleDependent**`( java.lang.Class` **moduleClass**`, java.lang.String` **addCallback**`, java.lang.String` **remCallback** `)`

- **Usage**
  - Execute module dependent code. addCallback will be called when the module is loaded, remCallback when it is removed. This method may only be called when connected to a WSN =>call in _init()!
    Callbacks are called with an instance of the given module. (but they may be declared without parameters). If more than one fitting method is found (e.g. a method taking the actual module as argument as well as a method taking a WSNModule as argument) all methods are called starting with the most specific one.
- **Parameters**
  - `module` -
  - `addCb` -
  - `remCb` -
- **Returns** - this for fluent interface

- *_postShutdown*
`protected void` **_postShutdown**`( )`
  - **Usage**
    - handler that is called everytime the module was shutdown properly

---

- *_run*
`protected void` **_run**`( )`
  - **Usage**
    - worker method (WSNModule worker thread, remember it must be interruptible! Otherwise the Module won't shut down correctly.

- *_setError*
`protected WSNModule` **_setError**`( java.lang.String` **message** `)`
  - **Usage**
    - set this module's status to ERROR with the given status message.
  - **Parameters**
    - `message` -
  - **Returns** - this for fluent interface

- *_setIdling*
`protected WSNModule` **_setIdling**`( java.lang.String` **message** `)`
  - **Usage**
    - set this module's status to IDLING with the given status message.
  - **Parameters**
    - `message` -
  - **Returns** - this for fluent interface

- *_setRunning*
`protected WSNModule` **_setRunning**`( java.lang.String` **message** `)`
  - **Usage**
    - set this module's status to RUNNING with the given status message.

- **Parameters**
  - `message` -
- **Returns** - this for fluent interface

- *_setStatus*
  `protected WSNModule` **_setStatus(**
  `de.tum.in.net.WSNDataFramework.WSNModule.WSNModuleStatus.STATUS` **status)**
  - **Usage**
    - sets this module's status
  - **Parameters**
    - `status` -
  - **Returns** - this for fluent interface

- *_setStatus*
  `protected WSNModule` **_setStatus(**
  `de.tum.in.net.WSNDataFramework.WSNModule.WSNModuleStatus.STATUS` **status,**
  `java.lang.String` **message)**
  - **Usage**
    - sets this module's status
  - **Parameters**
    - `status` -
    - `message` -
  - **Returns** - this for fluent interface

- *_shutdown*
  `protected void` **_shutdown(** **)**
  - **Usage**
    - handler that is called everytime the module gets shutdown

---

- *_subscribeTo*
  `protected boolean` **_subscribeTo(** `java.lang.Class` **event,** `java.lang.String` **cbName)**
  - **Usage**
    - subscribe to specific event. (listens to WSN and WSNModule events). Events are handled in an own thread so they won't block the WSN.
  - **Parameters**
    - `event` -
    - `cbName` -

- *getName*
  `public String` **getName(** **)**
  - **Usage**
    - Gets the name of this module.

- *getStatus*
  `public WSNModule.WSNModuleStatus` **getStatus(** **)**

  - **Usage**
    - Gets current Status.

- *getWSN*
  `public WSN` **getWSN**`( )`
  - **Usage**
    - Gets the currently attached WSN
  - **Returns** - currently attached WSN

- *isShutdown*
  `public boolean` **isShutdown**`( )`
  - **Usage**
    - checks if WSNModule is shut down
  - **Returns** - true if WSNModule is shut down

- *isShuttingDown*
  `public boolean` **isShuttingDown**`( )`
  - **Usage**
    - checks if WSNModule is currently shutting down
  - **Returns** - true if WSNModule is shutting down

- *setWSN*
  `public final synchronized WSNModule` **setWSN**`(`
  `de.tum.in.net.WSNDataFramework.WSN` **wsn** `)`
  - **Usage**
    - Attaches this module to a specific WSN and runs it runner thread.. This method implicitly calls shutdown() if the module is already attached to a WSN. Has the same effect as calling wsn.addModule(this).
  - **Parameters**
    - `wsn` - WSN to attach to
  - **Returns** - this for fluent interface

- *shutdown*
  `public synchronized WSNModule` **shutdown**`( )`
  - **Usage**
    - Shuts module down. stops runner thread and detaches the module from its WSN. Has the same effect as calling WSN.removeModule(this).
  - **Returns** - this for fluent interface

- *waitForShutdown*
  `public WSNModule` **waitForShutdown**`( )`
  - **Usage**
    - blocks until module is shut down
  - **Returns** - this for fluent interface
  - **Exceptions**
    - `java.lang.InterruptedException` -

METHODS INHERITED FROM CLASS `de.tum.in.net.WSNDataFramework.Event.EventProvider`

- *_eventRegistrationAllowed*
  `protected boolean` **_eventRegistrationAllowed**`( java.lang.Class` **eventClass** `)`
  - **Usage**
    - checks if this EventProvider shall allow a registration for a specific event. Just maps to EventProvider.providesEvent() but may be overridden by subclass.
  - **Parameters**
    - `eventClass` -

- *fireEvent*
  `protected EventProvider` **fireEvent**`( de.tum.in.net.WSNDataFramework.Event.Event` **eve** `)`
  - **Usage**
    - fire specific event. Adds the fired event to all EventBuffers registered for it or any of its parent classes.
  - **Parameters**
    - `eve` - Event to fire
  - **Returns** - this for fluent interface

- *getProvidedEvents*
  `public Class` **getProvidedEvents**`( )`
  - **Usage**
    - get list of events offered by this provider. Uses reflection to determine events actually provided by this class.

- *providesEvent*
  `public boolean` **providesEvent**`( java.lang.Class` **event** `)`
  - **Usage**
    - checks if a specific event is provided. (considers inheritance, if provider provides (TestEvent extends Event) a check for Event would also return true)
  - **Parameters**
    - `event` -

- *subscribeEvent*
  `public boolean` **subscribeEvent**`( java.lang.Class` **eventClass**`,`
  `de.tum.in.net.WSNDataFramework.Event.EventBuffer` **eventBuffer** `)`
  - **Usage**
    - Subscribes to a specific event. Adds an instance of the fired event to eventBuffer each time it is fired.
  - **Parameters**
    - `eventClass` -
    - `eventBuffer` -

- *unsubscribeEvent*
  `public EventProvider` **unsubscribeEvent**`( java.lang.Class` **eventClass**`,`
  `de.tum.in.net.WSNDataFramework.Event.EventBuffer` **eventBuffer** `)`
  - **Usage**
    - Unsubscribes from a previously subscribed event.
  - **Parameters**
    - `eventClass` -
    - `eventBuffer` -

1.1.8 Class **WSNModule.WSNModuleStatus**

represents the status of a module

Declaration

public static class WSNModule.WSNModuleStatus
extends java.lang.Object

Constructors

- *WSNModule.WSNModuleStatus*
  `public` **WSNModule.WSNModuleStatus**`(`
  `de.tum.in.net.WSNDataFramework.WSNModule.WSNModuleStatus.STATUS` **status**`,`
  `java.lang.String` **message** `)`
  - **Usage**
    - constructor
  - **Parameters**
    - `status` -
    - `message` -

Methods

- *getMessage*
  `public String` **getMessage**`( )`
  - **Usage**
    - gets attached message.

- *getStatus*
  `public WSNModule.WSNModuleStatus.STATUS` **getStatus**`( )`
  - **Usage**
    - gets WSNModuleStatus.STATUS.

1.1.9 Class **WSNModule.WSNModuleStatus.STATUS**

Declaration

public static final class WSNModule.WSNModuleStatus.STATUS
extends java.lang.Enum

FIELDS

- public static final WSNModule.WSNModuleStatus.STATUS RUNNING

  –
- public static final WSNModule.WSNModuleStatus.STATUS IDLING

  –
- public static final WSNModule.WSNModuleStatus.STATUS ERROR

  –

METHODS

- *valueOf*
  `public static WSNModule.WSNModuleStatus.STATUS` **valueOf**`( java.lang.String` **name** `)`
- *values*
  `public static WSNModule.WSNModuleStatus.STATUS` **values**`( )`

METHODS INHERITED FROM CLASS `java.lang.Enum`

- *clone*
  `protected final Object` **clone**`( )`
- *compareTo*
  `public final int` **compareTo**`( java.lang.Enum` **arg0** `)`
- *equals*
  `public final boolean` **equals**`( java.lang.Object` **arg0** `)`
- *finalize*
  `protected final void` **finalize**`( )`
- *getDeclaringClass*
  `public final Class` **getDeclaringClass**`( )`
- *hashCode*
  `public final int` **hashCode**`( )`
- *name*
  `public final String` **name**`( )`
- *ordinal*
  `public final int` **ordinal**`( )`
- *toString*
  `public String` **toString**`( )`
- *valueOf*
  `public static Enum` **valueOf**`( java.lang.Class` **arg0**`, java.lang.String` **arg1** `)`

1.1.10 CLASS WSNNode

represents a WSN Node.

Declaration

public class WSNNode
extends java.lang.Object
implements java.lang.Cloneable, java.io.Serializable

Serializable Fields

- public final String ID
  - node ID
- public Map fields
  - mutable list of the node's fields. mapped for quicker access as FieldID=>Field
- public WSNNode.Location location
  - node's location
- public Date updatedAt
  - time this node was updated the last time

Fields

- public final String ID
  - node ID
- public Map fields
  - mutable list of the node's fields. mapped for quicker access as FieldID=>Field
- public WSNNode.Location location
  - node's location
- public Date updatedAt
  - time this node was updated the last time

Constructors

- *WSNNode*
  `public` **WSNNode**`( java.lang.String` **nodeID** `)`
  - **Usage**
    - constructor.
  - **Parameters**
    - `nodeID` -

- *WSNNode*
  `public` **WSNNode**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    - copy constructor. creates a deep copy (also copies its 'fields'). Field.value is not cloned, so be careful using mutable types as Field.values.
  - **Parameters**
    - `node` -

Methods

- *clone*
  `public WSNNode` **clone**`( )`
  - **Usage**
    - clones a WSNNode using the copy constructor of the actual class
  - **Exceptions**
    - `java.lang.CloneNotSupportedException` - if actual class doesn't offer a copy constructor!

- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`
  - **Usage**
    - Determines whether two nodes are exact the same. Checks all public fields for equality.

- *same*
  `public boolean` **same**`( de.tum.in.net.WSNDataFramework.WSNNode` **node** `)`
  - **Usage**
    - Determines whether two nodes may be considered the same. Compares ID + field entries (two field entries are the same if all their fields are equal except their value).

- *updateField*
  `public WSNNode` **updateField**`( de.tum.in.net.WSNDataFramework.WSNNode.Field` **Field** `)`
  - **Usage**
    - Updates a single Field. Adds it or replaces it if already present with the given Field and updates WSNNode.updatedAt.
  - **Parameters**
    - `Field` -
  - **Returns** - this for fluent interface

- *updateFields*
  `public WSNNode` **updateFields**`( java.util.List` **fields** `)`
  - **Usage**
    - updates this node's field list. replaces it by the given list and updates WSNNode.updatedAt.

  - **Parameters**
    - `fields` -
  - **Returns** - this for fluent interface

- *updateLocation*
  `public WSNNode` **updateLocation**`(`
  `de.tum.in.net.WSNDataFramework.WSNNode.Location` **location** `)`
  - **Usage**
    - Updates this node's location. replaces it by the given one and updates WSNNode.updatedAt.
  - **Parameters**
    - `location` -
  - **Returns** - this for fluent interface

1.1.11 Class **WSNNode.Field**

field entry class

Declaration

public static class WSNNode.Field
extends java.lang.Object
implements java.lang.Cloneable, java.io.Serializable

Serializable Fields

- public final String ID
  - ID that uniquely identifies this field
- public String name
  - specific name (may be null)
- public String type
  - Data type
- public Object value
  - Field value
- public String unit
  - Data unit

FIELDS

- public final String ID
  - ID that uniquely identifies this field
- public String name
  - specific name (may be null)
- public String type
  - Data type
- public Object value
  - Field value
- public String unit
  - Data unit

CONSTRUCTORS

- *WSNNode.Field*
  `public` **WSNNode.Field**`(` `java.lang.String` **fieldID**`,` `java.lang.String` **type**`,` `java.lang.Object` **value**`,` `java.lang.String` **unit** `)`
  - **Usage**
    - constructor, omits Field.name.
  - **Parameters**
    - `fieldID` -
    - `type` -
    - `value` -
    - `unit` -

- *WSNNode.Field*
  `public` **WSNNode.Field**`(` `java.lang.String` **fieldID**`,` `java.lang.String` **name**`,` `java.lang.String` **type**`,` `java.lang.Object` **value**`,` `java.lang.String` **unit** `)`
  - **Usage**
    - constructor
  - **Parameters**
    - `fieldID` -
    - `name` -
    - `type` -
    - `value` -
    - `unit` -

- *WSNNode.Field*
  `public` **WSNNode.Field**`(` `de.tum.in.net.WSNDataFramework.WSNNode.Field` **Field** `)`
  - **Usage**

  - copy constructor. Does not clone Field.value ! So be careful if using mutable types as Field.value.
- **Parameters**
  - `Field` -

Methods

- *clone*
  `public final WSNNode.Field clone( )`
  - **Usage**
    - clones a WSNNode.Field using the copy constructor of the actual class
  - **Exceptions**
    - `java.lang.CloneNotSupportedException` - if actual class doesn't offer a copy constructor!

- *equals*
  `public boolean equals( java.lang.Object obj )`
  - **Usage**
    - Determines whether two Field may be considered equal. return true if obj is a Field and all its fields equal the fields of this.

- *same*
  `public boolean same( de.tum.in.net.WSNDataFramework.WSNNode.Field Field )`
  - **Usage**
    - Determines whether two Field may represent the same field. returns true if all fields except 'value' are equal.

1.1.12 Class **WSNNode.Location**

WSNNode location

Declaration

public static class WSNNode.Location **extends** java.lang.Object **implements** java.lang.Cloneable, java.io.Serializable

Serializable Fields

- public Double latitude
  –
- public Double longitude

–

- public Double elevation
  –

FIELDS

- public Double latitude
  –
- public Double longitude
  –
- public Double elevation
  –

CONSTRUCTORS

- *WSNNode.Location*
  `public` **WSNNode.Location**`( java.lang.Double` **latitude**`, java.lang.Double` **longitude**`, java.lang.Double` **elevation** `)`
  - **Usage**
    - constructor
  - **Parameters**
    - `latitude` -
    - `longitude` -
    - `elevation` -

- *WSNNode.Location*
  `public` **WSNNode.Location**`( de.tum.in.net.WSNDataFramework.WSNNode.Location` **location** `)`
  - **Usage**
    - copy constructor.
  - **Parameters**
    - `location` -

METHODS

- *clone*
  `public final WSNNode.Location` **clone**`( )`
  - **Usage**
    - clones a WSNNode.Location using the copy constructor of the actual class.
  - **Exceptions**

        * `java.lang.CloneNotSupportedException` - if actual class doesn't offer a copy constructor!

- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`
  - **Usage**
    * overrides equals. checks if two Locations equal regarding their latitude, longitude and elevation.

1.1.13 Class **WSNTopology**

Declaration

public class WSNTopology **extends** java.lang.Object **implements** java.lang.Cloneable, java.io.Serializable

Constructors

- *WSNTopology*
  `public` **WSNTopology**`( )`
  - **Usage**
    * default constructor.

- *WSNTopology*
  `public` **WSNTopology**`( de.tum.in.net.WSNDataFramework.WSNTopology` **topology** `)`
  - **Usage**
    * copy constructor. Copies all links from topology and adds them to the new one.
  - **Parameters**
    * `topology` -

Methods

- *addLink*
  `public WSNTopology` **addLink**`( de.tum.in.net.WSNDataFramework.WSNTopology.Link` **link** `)`
  - **Usage**
    * adds a link to the topology.
  - **Parameters**
    * `sourceID` - nodeID of the source node
    * `targetID` - nodeID of the target node

- *clone*
  `public final WSNTopology` **clone**`( )`
  - **Usage**
    - clones a WSNTopology using the copy constructor of the actual class

---

- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`
  - **Usage**
    - override Object.equals. Returns true if obj is a WSNTopology and contains the same links as this WSNTopology.

- *getLinks*
  `public Set` **getLinks**`( )`
  - **Usage**
    - gets all the links of this topology.

- *getLinks*
  `public Set` **getLinks**`( java.lang.String` **nodeID** `)`
  - **Usage**
    - gets all the links that include the given nodeID.
  - **Parameters**
    - `nodeID` -

- *same*
  `public boolean` **same**`( de.tum.in.net.WSNDataFramework.WSNTopology` **topology** `)`
  - **Usage**
    - Determines whether two WSNTopology instances may represent the same topology. Currently returns this.equals(topology) but may be overridden!

1.1.14 Class **WSNTopology.Link**

representing a WSNTopology Link.
holds source and target nodeID.
null as source/target stands for the base node (the global data sink).

Declaration

public static class WSNTopology.Link **extends** java.lang.Object **implements** java.lang.Cloneable

Fields

- public String source
  - source node ID
- public String target
  - target node ID
- public Double weight
  - link weight

Constructors

- *WSNTopology.Link*
  `public` **WSNTopology.Link**`( java.lang.String` **source**`, java.lang.String` **target**`, java.lang.Double` **weight** `)`
  - **Usage**
    - constructor.
  - **Parameters**
    - `source` -
    - `target` -

- *WSNTopology.Link*
  `public` **WSNTopology.Link**`( de.tum.in.net.WSNDataFramework.WSNTopology.Link` **link** `)`
  - **Usage**
    - copy constructor.
  - **Parameters**
    - `link` -

Methods

- *clone*
  `public final WSNTopology.Link` **clone**`( )`
  - **Usage**
    - clones a WSNTopology.Link using the copy constructor of the actual class

---

- *equals*
  `public boolean` **equals**`( java.lang.Object` **obj** `)`
  - **Usage**
    - Determines whether two WSNTopology.Links are the same. returns true if ‘obj‘ is an instance of WSNTopology.Link and obj’s source,target and weight equal the values of this.

- *hashCode*
  `public int` **hashCode**`( )`
  - **Usage**
    - override Object.hashCode for easy use in a Hash based Collection.
creates a hash by forming a unique string from source,target and weight.

- *involves*
  `public boolean` **involves**`( java.lang.String` **nodeID** `)`
  - **Usage**
    - checks if this link involves a specific nodeID.
  - **Parameters**
    - `nodeID` -
  - **Returns** - true if this.source or this.target equals nodeID

- *same*
  `public boolean` **same**`( de.tum.in.net.WSNDataFramework.WSNTopology.Link` **link** `)`
  - **Usage**
    - Determines whether two WSNTopology.Links may represent the same link but with a different ‘weight‘.