

Design document 0.5

SQUID

Helsinki 15th March 2005
Software Engineering Project
UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Mikko Jormalainen
Samuli Kaipiainen
Aki Korpua
Esko Luontola
Aki Sysmäläinen

Client

Lauri J. Pesonen
Fabio Donadini
Tomas Kohout

Project Masters

Juha Taina
Jenni Valorinta

Homepage

<http://www.cs.helsinki.fi/group/squid/>

Change Log

Version	Date	Modifications
0.1	4.3.2005	First version with nothing in it (Samuli Kaipiainen)
0.2	8.3.2005	Some class descriptions (Aki Korpua, Samuli Kaipiainen)
0.25	9.3.2005	Macros for class/field/method documentation (Esko Luontola) RunQueue (Esko Luontola)
0.3	11.3.2005	Conventions added, Class diagrams improved (Esko Luontola) Subsystem sections (Samuli Kaipiainen)
0.4	12.3.2005	Measurement controls (Samuli Kaipiainen) Calibration and Project Explorer (Samuli Kaipiainen) Main view, menu, settings, statusbar (Aki Korpua) Class diagram modified (Esko Luontola) Class descriptions for Project data (Esko Luontola)
0.45	13.3.2005	Squid-emu, Squid-interface (Aki Korpua) Fixes and refinements (Samuli Kaipiainen) Project class (Esko Luontola)
0.5	14.3.2005	Small fixes (Esko Luontola) Some fixes (Aki Korpua) Small and bigger fixes (Samuli Kaipiainen) Project class finished (Esko Luontola)

Contents

1	Introduction	1
1.1	Meaning and structure of the document	1
1.2	Glossary	1
2	Conventions	1
3	Overview of the system	2
4	Architecture description	2
5	Data classes and methods	5
5.1	Project data	5
5.1.1	Project	5
5.1.2	Project.Type	14
5.1.3	Project.State	14
5.1.4	MeasurementSequence	14
5.1.5	MeasurementStep	16
5.1.6	MeasurementStep.State	19
5.1.7	MeasurementResult	19
5.1.8	MeasurementResult.Type	21
5.1.9	MeasurementValue	21
5.1.10	ProjectEvent	23
5.1.11	ProjectEvent.Type	24
5.1.12	ProjectListener	24
5.1.13	MeasurementEvent	24
5.1.14	MeasurementEvent.Type	25
5.1.15	MeasurementListener	25
5.2	Squid interface	26
5.2.1	Squid	26
5.2.2	Degausser	27
5.2.3	Handler	29
5.2.4	Magnetometer	31
5.3	Squid emulator	33

5.3.1	SquidEmulator	33
5.4	Serial communication	35
5.5	Global settings	35
5.5.1	Settings	35
5.6	Utilities	37
5.6.1	RunQueue	37
5.6.2	RunQueue.RunQueueThread	39
5.6.3	RunQueue.RunDelayed	39
6	GUI classes and methods	40
6.1	Generic GUI components	40
6.1.1	ProjectComponent	40
6.2	Main window	41
6.2.1	MainViewPanel	41
6.3	Menu bar	43
6.3.1	MainMenuBar	43
6.4	Status bar	44
6.4.1	MainStatusBar	44
6.5	Configuration window	45
6.5.1	SettingsPanel	45
6.6	Project Explorer	47
6.6.1	ProjectExplorerPanel	47
6.6.2	NewProjectPanel	48
6.6.3	ProjectExplorerTable	49
6.6.4	ProjectExplorerPopupMenu	49
6.7	Calibration	50
6.7.1	CalibrationPanel	50
6.8	Project information	51
6.8.1	ProjectInformationPanel	51
6.9	Sequence and measurement data	53
6.9.1	MeasurementSequencePanel	53
6.9.2	MeasurementSequenceTableModel	54
6.9.3	MeasurementSequencePopupMenu	55
6.10	Measurement details	55

	iii
6.10.1 MeasurementDetailsPanel	55
6.11 Measurement controls	56
6.11.1 MeasurementControlsPanel	56
6.11.2 MagnetometerStatusPanel	57
6.11.3 ManualControlsPanel	57
6.12 Graphs	58
6.12.1 MeasurementGraphsPanel	58
7 Package structure	58
8 Bibliography	58

1 Introduction

1.1 Meaning and structure of the document

Document aids implementation phase? Allows anyone to continue from our plans (or should anyway)? Division between data and gui classes? Each data/gui part may have several classes?

1.2 Glossary

Probably dumped again...

2 Conventions

Everybody will follow the Code Conventions for the Java Programming Language set by Sun, with the following refinements.

- Line length will be set to 120 characters, because we prefer coding in high resolutions.
- If possible, set your IDE to use spaces instead of tabs (to avoid problems if somebody has set tab to 4 spaces, although it should be 8). Indentation is 4 spaces, as set by Sun.
- Every method and non-trivial field must have Javadoc comments. Every parameter, return value and exception of methods must be mentioned (except for trivial getters and setters).
- Every if, for and while loop must use braces `{ }`, even when there will be only one statement in the block, as set by Sun.
- The `@author` comment for every class should have the name of the person who wrote (and designed) the class. Then we will know who to ask, if there are some questions about the code.
- Every source file is subject to automatic code reformatting by a Java IDE, in which case the reformatter must follow these code conventions.
- TODO-comments should be set by the programmer, if there is some part that needs more work. The format is `" // TODO: comments "`

The Code Conventions are available at
<http://java.sun.com/docs/codeconv/>

This program will be written with Java 1.5. Every programmer should have a look at the new features that were introduced to the Java language. Especially noteworthy are Generics, Foreach-loop and Enums. The following article will explain them in a nutshell.

<http://java.sun.com/developer/technicalArticles/releases/j2se15/>

It is recommendable for everybody to have a quick glance at Design Patterns. Here are some useful links.

<http://sern.ucalgary.ca/courses/SENG/609.04/W98/notes/>

<http://www.dofactory.com/Patterns/Patterns.aspx>

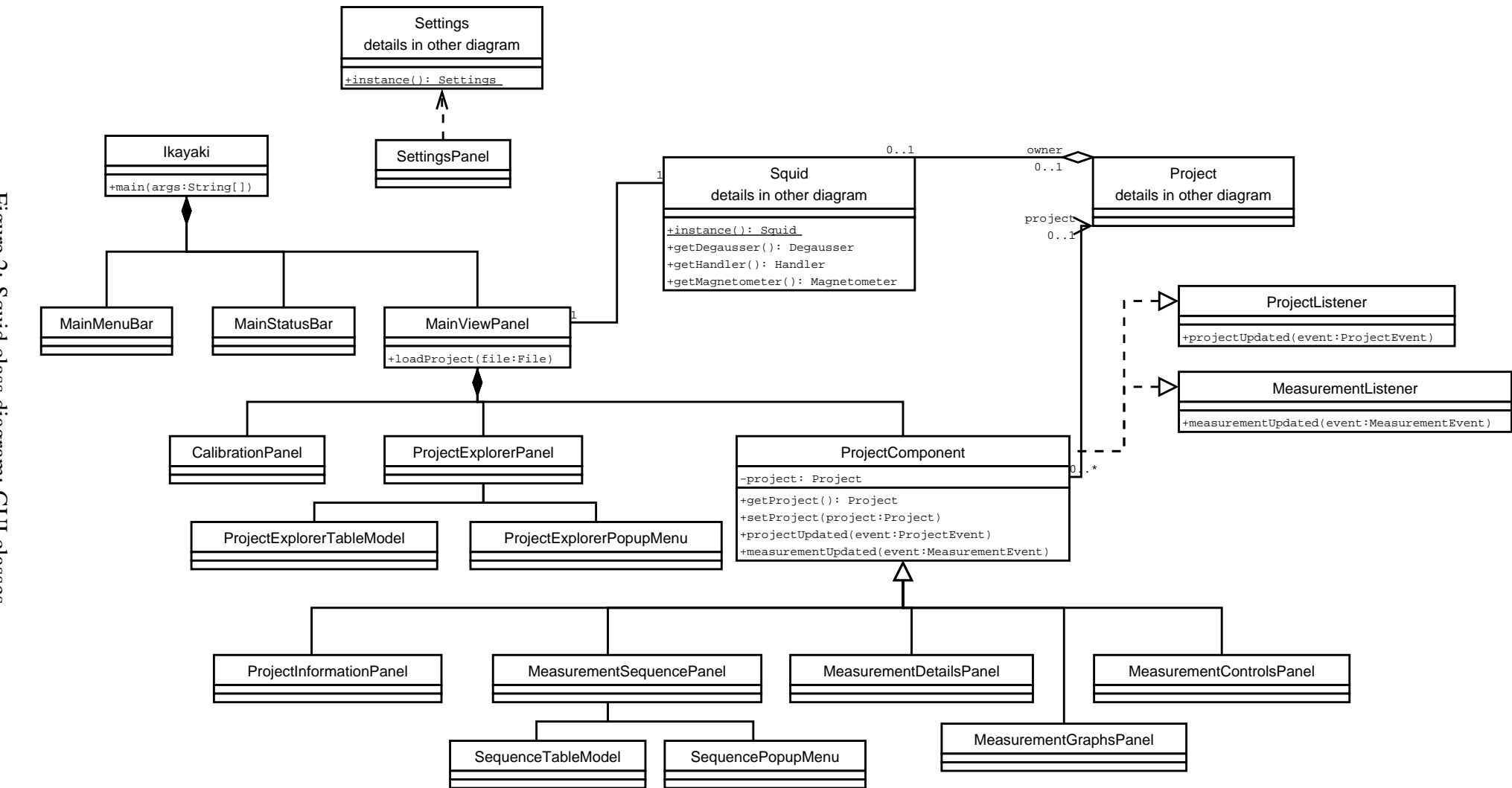
3 Overview of the system

See Figure 1 and Figure 2.

4 Architecture description

Is this needed? What would be here anyway? Perhaps some chainsaw-humour...

Figure 2: Squid class diagram: GUI classes



5 Data classes and methods

5.1 Project data

5.1.1 Project

Package	ikayaki
Declaration	public class Project
Created by	ProjectExplorerPanel (6.6.1)
Uses 1	MeasurementSequence (5.1.4)
Uses 2	MeasurementStep (5.1.5)
Uses 3	MeasurementResult (5.1.7)
Uses 4	MeasurementValue (5.1.9)
Uses 5	Squid (5.2.1)
Uses 6	RunQueue (5.6.1)
Uses 7	ProjectEvent (5.1.10)
Uses 8	MeasurementEvent (5.1.13)

Represents a measurement project file. Project is responsible for managing and storing the data that is recieved from the magnetometer measurements. Any changes made to the project will be written to file regularly (autosave).

Project is responsible for controlling the magnetometer through the SQUID API. Controlling the SQUID will be done in a private worker thread. Only one project at a time may access the SQUID.

All operations are thread-safe.

Design patterns Facade

Event A *On property change* - Autosaving will be invoked and the project written to file after a short delay.

Event B *On measurement started/ended/paused/aborted* - ProjectEvent will be fired to all project listeners.

Event C *On measurement subphase started/completed* - MeasurementEvent will be fired to all measurement listeners.

Event D *On declination/inclination/volume changed* - The updated transformation matrix will be applied to all measurements and a ProjectEvent will be fired to all project listeners.

```
private static Hashtable<File,Project> projectCache
```

Caches the created and loaded Project objects to make sure that no more than one object will be created for each physical file.

```
private File file
```

Location of the project file in the local file system. Autosaving will save the project to this file.

```
private Type type
```

Type of the measurement project. This will affect which features of the project are enabled and disabled.

private State state

Default value IDLE

Current state of the measurements. If no measurement is running, then state is IDLE. Only one measurement may be running at a time.

private Squid squid

Default value null

Pointer to the SQUID device interface, or null if this project is not its owner.

private Properties properties

Custom properties of this project stored in a map. The project is not interested in what properties are stored; it only saves them.

private MeasurementSequence sequence

Measurement sequence of this project. In the beginning are all completed measurement steps, and in the end are planned measurement steps. Completed measurements may NOT be deleted.

private double strike

Default value 0.0

Strike of the sample. Will be used to create the transform matrix.

private double dip

Default value 0.0

Dip of the sample. Will be used to create the transform matrix.

private SampleType sampleType

Default value CORE

Type of the sample. Will be used to create the transform matrix.

private Matrix3d transform

Default value new Matrix3d()

Matrix for correcting the sample's orientation. The matrix will be updated whenever the strike, dip or sampleType is changed. After that the updated matrix will be applied to all measurements.

private double mass

Default value -1.0

Mass of the sample, or a negative value if no mass is defined.

private double volume

Default value -1.0

Volume of the sample, or a negative value if no volume is defined.

private MeasurementStep currentStep

Default value null

Current measurement step, or null if no measurement is running.

```
private EventListenerList listenerList
```

Default value new EventListenerList()

Listeners for this project.

```
private RunQueue autosaveQueue
```

Default value new RunQueue(500, true)

Scheduler for automatically writing the modified project to file after a short delay.

```
public static Project createCalibrationProject(File
file)
```

Creates a calibration project file.

Parameter 1 *file* - path for the new project file.

Returns the created project, or null if file was not writable.

```
public static Project createAFProject(File file)
```

Creates an AF project file.

Parameter 1 *file* - path for the new project file.

Returns the created project, or null if file was not writable.

```
public static Project createThellierProject(File file)
```

Creates a thellier project file.

Parameter 1 *file* - path for the new project file.

Returns the created project, or null if file was not writable.

```
public static Project createThermalProject(File file)
```

Creates a thermal project file.

Parameter 1 *file* - path for the new project file.

Returns the created project, or null if file was not writable.

```
private static Project createProject(File file, Type
type)
```

Creates a project file of the specified type. Ensures that the project file has been written to disk. Adds the created Project object to projectCache.

Parameter 1 *file* - path for the new project file.

Parameter 2 *type* - type of the project.

Returns the created project, or null if file was not writable.

```
public static Project loadProject(File file)
```

Loads a saved project file. If the file has already been loaded, will return a reference to the existing Project object.

Parameter 1 *file* - project file to be loaded.

Returns the loaded project, or null if file is not a valid project file or it was not readable.

```
public static boolean closeProject(Project project)
```

Ensures that the project file is saved and frees the resources taken by the project. A project should not be used after it has been closed – any further use of the object is undefined (probably will create `NullPointerException`s). The closed project is removed from the `projectCache`. A project can not be closed if it has a measurement running.

Parameter 1 *project* - project to be closed.

Returns true if the project has been closed, false if a measurement is running and the project can not be closed.

Throws *NullPointerException* - if the project is null.

```
private Project(File file, Type type)
```

Creates a new project of the specified type. This constructor will not write to file, so the user of this method should call the `saveNow()` method after the project is initialized.

Parameter 1 *file* - path for this project file. The file should exist (may be empty) and be writable, but this constructor will not check it.

Parameter 2 *type* - type of the project.

Returns the created project.

```
private Project(File file, Document document)
```

Creates a new project from the specified document. This constructor will assume that the specified file is the same from which the document was read.

Parameter 1 *file* - path for this project file. The file should be the same from which document was read and be writable, but this constructor will not check it.

Parameter 2 *document* - the document from which this project will be created.

Returns the created project.

Throws *IllegalArgumentException* - if the import data was not in the right format.

```
public synchronized Document getDocument()
```

Exports this project to a DOM document.

```
public synchronized void save()
```

Invokes autosaving. This method will schedule a saving operation and return. After this method has not been called for a short while, the project will be written to file.

```
public void saveNow()
```

Writes this project to its project file and waits for the operation to complete. (NOTE: Synchronizing is done inside the method)

Throws *IOException* - if there was an error when writing to file.

```
public synchronized File getFile()
```

Returns the project file of this project.

```
public synchronized Type getType()
```

Returns the type of this project.

```
public synchronized State getState()
    Returns the current measurement state of this project.
```

```
public synchronized String getName()
    Returns the name of this project. The name is equal to the name of the project file
    without the file extension.
```

```
public synchronized Date getTimestamp()
    Returns the timestamp of the last completed measurement. This is usually less
    than the last modified date of the file, because this is not affected by changing the
    project's properties.
```

```
private synchronized Squid getSquid()
    Returns the Squid if this project is its owner, otherwise returns null.
```

```
public synchronized boolean setSquid(Squid squid)
    Sets this project the owner of the Squid. Uses the setOwner() method of the speci-
    fied Squid.
    Only one project may own the Squid at a time. The Squid must be first detached
    with "setSquid(null)" from its owner before it can be given to another project. De-
    taching the Squid is possible only when the project's state is IDLE.
Parameter 1    squid - pointer to the SQUID interface, or null to detach this
    project from it.
Returns        true if the operation was completed, false if the Squid has another
    owner or a measurement is running (in which case nothing was
    changed).
```

```
public synchronized String getProperty(String key)
    Returns a project information property.
Parameter 1    key - the key which is associated with the property.
Returns        the specified property, or an empty String if the property is not
    set.
```

```
public synchronized void setProperty(String key, String
value)
    Sets a project information property.
Parameter 1    key - the key which is associated with the property.
Parameter 2    value - new value for the property, or null to remove the property.
```

```
public synchronized double getStrike()
    Returns the strike of the sample.
```

```
public synchronized void setStrike(double strike)
    Sets the strike of the sample and calls updateTransforms().
```

```
public synchronized double getDip()
    Returns the dip of the sample.
```

```
public synchronized void setDip(double dip)
```

Sets the dip of the sample and calls updateTransforms().

```
public synchronized SampleType getSampleType()
```

Returns the type of the sample.

```
    public synchronized void setSampleType(SampleType
sampleType)
```

Sets the type of the sample and calls updateTransforms().

Throws *NullPointerException* - if sampleType is null.

```
synchronized Matrix3d getTransform()
```

Returns the current transformation matrix for the sample. For performance reasons, this method returns a reference to the internal data structure and not a copy of it.

WARNING!!! Absolutely NO modification of the data contained in this matrix should be made – if any such manipulation is necessary, it should be done on a copy of the matrix returned rather than the matrix itself.

Returns reference to the transformation matrix.

```
private synchronized void updateTransforms()
```

Recalculates the transformation matrix and updates all measurements. This method is called automatically by the setStrike(), setDip() and setSampleType() methods.

```
public synchronized double getMass()
```

Returns the mass of the sample.

Returns mass of the sample, or a negative number if no mass is specified.

```
public synchronized void setMass(double mass)
```

Sets the mass of the sample.

Parameter 1 *mass* - mass of the sample, or a negative number to clear it.

```
public synchronized double getVolume()
```

Returns the volume of the sample.

Returns volume of the sample, or a negative number if no volume is specified.

```
public synchronized void setVolume(double volume)
```

Sets the volume of the sample.

Parameter 1 *volume* - volume of the sample, or a negative number to clear it.

```
public synchronized void addProjectListener(ProjectListener
l)
```

Adds a ProjectListener to the project.

Parameter 1 *l* - the listener to be added.

```
public synchronized void removeProjectListener(ProjectListener
l)
```

Removes a ProjectListener from the project.

Parameter 1 *l* - the listener to be removed

```
private synchronized void fireProjectEvent(ProjectEvent.Type
type)
```

Notifies all listeners that have registered for ProjectEvents.

Parameter 1 *type* - type of the event.

```
public synchronized void addMeasurementListener(MeasurementListener
l)
```

Adds a MeasurementListener to the project.

Parameter 1 *l* - the listener to be added.

```
public synchronized void removeMeasurementListener(MeasurementListene
r l)
```

Removes a MeasurementListener from the project.

Parameter 1 *l* - the listener to be removed

```
private synchronized void fireMeasurementEvent(MeasurementStep
step, MeasurementEvent.Type type)
```

Notifies all listeners that have registered for MeasurementEvents.

Parameter 1 *step* - the measurement step that has generated the event.

Parameter 2 *type* - the type of the event.

```
public synchronized void addSequence(MeasurementSequence
sequence)
```

Appends a sequence to this project's sequence. Only the stepValues will be copied from the specified sequence and added as new steps to this project.

If isSequenceEditEnabled() is false, nothing will be done.

Parameter 1 *sequence* - the measurement sequence to be added.

Throws *NullPointerException* - if sequence is null.

```
public synchronized MeasurementSequence copySequence(int
start, int end)
```

Returns a copy of this project's sequence. Only the stepValues will be copied from this project's sequence. The returned sequence will have no name.

Parameter 1 *start* - index of the first step in the sequence.

Parameter 2 *end* - index of the last step in the sequence. If *end* < *start*, then an empty sequence will be returned.

Returns copy of the sequence with only stepValues and no results.

Throws *IndexOutOfBoundsException* - if the index is out of range (*start* < 0 || *end* >= *getSteps()*).

```
public synchronized void addStep(MeasurementStep step)
```

Appends a step to this project's sequence. Only the stepValue will be copied from the specified step and added as new steps to this project.

If isSequenceEditEnabled() is false, nothing will be done.

Parameter 1 *step* - the measurement step to be added.

Throws *NullPointerException* - if step is null.


```
public synchronized void addStep(int index,
MeasurementStep step)
```

Adds a step to the specified index of this project's sequence. Only the stepValue will be copied from the specified step and added as new steps to this project. The index must be such, that the indices of the completed measurements will not change.

If isSequenceEditEnabled() is false, nothing will be done.

Parameter 1 *index* - the index to which the step will be added.

Parameter 2 *step* - the measurement step to be added.

Throws *IndexOutOfBoundsException* - if the index is out of range ($\text{index} < \text{getCompletedSteps()} \parallel \text{index} > \text{getSteps()} \text{)}$.

Throws *NullPointerException* - if step is null.

```
public synchronized void removeStep(int index)
```

Removes a step from this project's sequence. Completed measurements can not be removed.

If isSequenceEditEnabled() is false, nothing will be done.

Parameter 1 *index* - the index of the step to be removed.

Throws *IndexOutOfBoundsException* - if the index is out of range ($\text{index} < \text{getCompletedSteps()} \parallel \text{index} >= \text{getSteps()} \text{)}$.

```
public synchronized void removeStep(int start, int end)
```

Removes a series of steps from this project's sequence. Completed measurements can not be removed.

If isSequenceEditEnabled() is false, nothing will be done.

Parameter 1 *start* - the first index to be removed.

Parameter 2 *end* - the last index to be removed. If $\text{end} < \text{start}$, no steps will be removed.

Throws *IndexOutOfBoundsException* - if the index is out of range ($\text{start} < \text{getCompletedSteps()} \parallel \text{end} >= \text{getSteps()} \text{)}$.

```
public synchronized int getSteps()
```

Returns the number of steps in this project.

```
public synchronized int getCompletedSteps()
```

Returns the number of completed steps in this project. Steps that are currently being measured, are included in this count. Completed steps are always first in the sequence.

```
public synchronized MeasurementStep getStep(int index)
```

Returns a step from the sequence.

Parameter 1 *index* - the index of the step.

Returns the specified step.

Throws *IndexOutOfBoundsException* - if the index is out of range ($\text{index} < 0 \parallel \text{index} >= \text{getSteps()} \text{)}$.

```
public synchronized MeasurementStep getCurrentStep()
```

Returns the step that is currently being measured.

Returns the currently measured step, or null if no measurement is active.

```
public synchronized <A> A getValue(int step,
MeasurementValue<A> algorithm)
```

Calculates and returns a value from a measurement step. The specified MeasurementValue's algorithm will be used and the results returned.

Parameter 1 *step* - the measurement step from which the value is calculated.

Parameter 2 *algorithm* - the algorithm for calculating the desired value.

Returns the value returned by the algorithm, or null if it was not possible to calculate it.

Throws *NullPointerException* - if algorithm is null.

```
public synchronized boolean isDegaussingEnabled()
```

Tells whether it is allowed to use the degausser in this project. The returned value depends on the type and state of this project.

```
public synchronized boolean isSequenceEditEnabled()
```

Tells whether it is allowed to edit the sequence. The returned value depends on the type and state of this project.

```
public synchronized boolean isManualControlEnabled()
```

Tells whether it is allowed to control the Squid manually. The returned value depends on the type and state of this project.

```
public synchronized boolean isAutoStepEnabled()
```

Tells whether it is allowed to do an auto step measurement. The returned value depends on the type and state of this project.

```
public synchronized boolean isSingleStepEnabled()
```

Tells whether it is allowed to do a single step measurement. The returned value depends on the type and state of this project.

```
public synchronized boolean isPauseEnabled()
```

Tells whether it is possible to pause the measurement. The returned value depends on the type and state of this project.

```
public synchronized boolean isAbortEnabled()
```

Tells whether it is possible to abort the measurement. The returned value depends on the type and state of this project.

```
public synchronized boolean doAutoStep()
```

Starts an auto step measurement. Will do nothing if isAutoStepEnabled() is false. The measurement will run in its own thread, and this method will not wait for it to finish.

Returns true if the measurement was started, otherwise false.

```
public synchronized boolean doSingleStep()
```

Starts a single step measurement. Will do nothing if `isSingleStepEnabled()` is false. The measurement will run in its own thread, and this method will not wait for it to finish.

Returns true if the measurement was started, otherwise false.

```
public synchronized boolean doPause()
```

Pauses the currently running measurement. A paused measurement will halt after it finishes the current measurement step. Will do nothing if `isPauseEnabled()` is false. This method will notify the measurement thread to pause, but will not wait for it to finish.

Returns true if the measurement will pause, otherwise false.

```
public synchronized boolean doAbort()
```

Aborts the currently running measurement. An aborted measurement will halt immediately, leave the handler where it was and enable manual control. Will do nothing if `isAbortEnabled()` is false.

This method will notify the measurement thread to abort, but will not wait for it to finish.

Returns true if the measurement will abort, otherwise false.

5.1.2 Project.Type

Package ikayaki

Declaration public enum Type

The type of the project. Options are CALIBRATION, AF, THELLIER and THERMAL.

5.1.3 Project.State

Package ikayaki

Declaration public enum State

The state of the project's measurements. Options are IDLE, MEASURING, PAUSED, ABORTED.

5.1.4 MeasurementSequence

Package ikayaki

Declaration public class MeasurementSequence

Created by Project (5.1.1)

Uses 1 MeasurementStep (5.1.5)

A list of measurement steps. Steps can be added or removed from the sequence.

All operations are thread-safe.

```
private String name
```

Default value

```
private List<MeasurementStep>
```

Default value `new ArrayList<MeasurementStep>()`

```
public MeasurementSequence()
```

Returns

Throws -

```
public MeasurementSequence(String name)
```

Parameter 1 *name* -

Returns

Throws -

```
public MeasurementSequence(Element import)
```

Parameter 1 *import* -

Returns

Throws -

```
    public MeasurementSequence(Element import, Project  
project)
```

Parameter 1 *import* -

Parameter 2 *project* -

Returns

Throws -

```
public synchronized Element getElement()
```

Returns

Throws -

```
public synchronized String getName()
```

Returns

Throws -

```
public synchronized void setName(String name)
```

Parameter 1 *name* -

Throws -

```
public synchronized int getSteps()
```

Returns

Throws -

```
public synchronized MeasurementStep getStep(int index)
```

Parameter 1 *index* -

Returns

Throws -

```
public synchronized void addStep(MeasurementStep step)
```

Parameter 1 *step* -

Throws -

```
public synchronized void addStep(int index,
MeasurementStep step)
```

Parameter 1 *index* -

Parameter 2 *step* -

Throws -

```
public synchronized void removeStep(int index)
```

Parameter 1 *index* -

Throws -

5.1.5 MeasurementStep

Package ikayaki

Declaration public class MeasurementStep

Created by Project, MeasurementSequencePanel (??)

Uses 1 Project (5.1.1)

Uses 2 MeasurementResult (5.1.7)

A single step in a measurement sequence. Each step can include multiple measurements for improved measurement precision. A step can have a different volume and mass than the related project, but by default the volume and mass of the project will be used. Only the project may change the state and results of a measurement step. All operations are thread-safe.

```
private Project project
```

```
private State state
```

Default value READY

```
private Date timestamp
```

```
private double stepValue
```

```
Default value    -1.0
```

```
private double mass
```

```
Default value    -1.0
```

```
private double volume
```

```
Default value    -1.0
```

```
private List<MeasurementResult> results
```

```
Default value    new ArrayList<MeasurementResult>()
```

```
public MeasurementStep()
```

```
Returns
```

```
Throws           -
```

```
public MeasurementStep(Project project)
```

```
Parameter 1      project -
```

```
Returns
```

```
Throws           -
```

```
public MeasurementStep(Element import)
```

```
Parameter 1      import -
```

```
Returns
```

```
Throws           -
```

```
public MeasurementStep(Element import, Project project)
```

```
Parameter 1      import -
```

```
Parameter 2      project -
```

```
Returns
```

```
Throws           -
```

```
public synchronized Element getElement()
```

```
Returns
```

```
Throws           -
```

```
public synchronized Project getProject()
```

Returns

Throws -

```
public synchronized State getState()
```

Returns

Throws -

```
void synchronized setState(State state)
```

Parameter 1 *state* -

Throws -

```
public synchronized Date getTimestamp()
```

Returns

Throws -

```
public synchronized double getStepValue()
```

Returns

Throws -

```
public synchronized void setStepValue(double stepValue)
```

Parameter 1 *stepValue* -

Throws -

```
public synchronized double getMass()
```

Returns

Throws -

```
public synchronized void setMass(double mass)
```

Parameter 1 *mass* -

Throws -

```
public synchronized double getVolume()
```

Returns

Throws -

```
public synchronized void setVolume(double volume)
```

Parameter 1 *volume* -

Throws -

```
synchronized void updateTransforms()
```

Throws -

```
public synchronized int getResults()
```

Returns

Throws -

```
public synchronized MeasurementResult getResult(int
index)
```

Parameter 1 *index* -

Returns

Throws -

```
public synchronized void addResult(MeasurementResult
result)
```

Parameter 1 *result* -

Throws -

5.1.6 MeasurementStep.State

Package ikayaki

Declaration public enum State

The state of a measurement step. Options are READY, MEASURING, DONE_RECENTLY and DONE.

5.1.7 MeasurementResult

Package ikayaki

Declaration public class MeasurementResult

Created by Magnetometer (5.2.4)

A set of X, Y and Z values measured by the magnetometer. The raw XYZ values will be rotated in 3D space by using a transformation matrix. The project will set and update the transformation whenever its parameters are changed.

```
private Type type
```

```
private Tuple3d rawTuple
```

Default value new Tuple3d()


```
private Tuple3d tuple
Default value    new Tuple3d()
```

```
public MeasurementResult(Type type, double x, double y,
double z
```

Parameter 1 *type* -

Parameter 2 *x* -

Parameter 3 *y* -

Parameter 4 *z* -

Returns

Throws -

```
public MeasurementResult(Element import)
```

Parameter 1 *import* -

Returns

Throws -

```
public Element getElement()
```

Returns

Throws -

```
void setTransform(Matrix3d transform)
```

Parameter 1 *transform* -

Throws -

```
public Type getType()
```

Returns

```
public double getX()
```

Returns

```
public double getY()
```

Returns

```
public double getZ()
```

Returns

```
public double getRawX()
```

Returns

```
public double getRawY()
```

Returns

```
public double getRawZ()
```

Returns

5.1.8 MeasurementResult.Type

Package ikayaki

Declaration public enum Type

The orientation of the sample when it was measured. Options are BG, DEG0, DEG90, DEG180 and DEG270.

```
public String getName()
```

Returns "BG", "0", "90", "180" or "270"

```
public Tuple3d rotate(Tuple3d t)
```

Rotates the raw XYZ values from the orientation of this object to that of DEG0. Rotating a BG or DEG0 will just copy the values directly.

Parameter 1 *t* - Old values that need to be rotated

Returns A new object with the rotated values.

```
public Tuple3d rotate(Tuple3d t, Tuple3d result)
```

Rotates the raw XYZ values from the orientation of this object to that of DEG0. Rotating a BG or DEG0 will just copy the values directly.

Parameter 1 *t* - Old values that need to be rotated

Parameter 2 *result* - Where the new values will be saved

Returns The same as the result parameter, or a new object if it was null.

5.1.9 MeasurementValue

Package ikayaki

Declaration public abstract class MeasurementValue<T>

Uses 1 MeasurementStep (5.1.5)

Algorithms for calculating values from the measurements. A MeasurementValue object will be passed to the getValue method of a project to retrieve the desired value.

Design patterns Strategy

```
public static final MeasurementValue<Double> X
```

```
public static final MeasurementValue<Double> Y
```

```

public static final MeasurementValue<Double> Z

public static final MeasurementValue<Double> DECLINATION

public static final MeasurementValue<Double> INCLINATION

public static final MeasurementValue<Double> MOMENT

public static final MeasurementValue<Double> REMANENCE

        public static final MeasurementValue<Double>
RELATIVE_REMANENCE

public static final MeasurementValue<Double> THETA63

private String caption

private String unit

private String description

        public MeasurementValue(String caption, String unit,
String description)

Parameter 1    caption -
Parameter 2    unit -
Parameter 3    description -
Returns
Throws        -

abstract T getValue(MeasurementStep step)

Parameter 1    step -
Returns
Throws        -

```

```
public String getCaption()
```

Returns

Throws -

```
public String getUnit()
```

Returns

Throws -

```
public String getDescription()
```

Returns

Throws -

5.1.10 ProjectEvent

Package ikayaki

Declaration public class ProjectEvent

Extends EventObject

Created by Project (5.1.1)

ProjectEvent is used to notify others about the state change of a project.

```
private Project project
```

```
private Type type
```

```
public ProjectEvent(Project project, Type type)
```

Parameter 1 *project* -

Parameter 2 *type* -

Returns

Throws -

```
public Project getProject()
```

Returns

Throws -

```
public Type getType()
```

Returns

Throws -

5.1.11 ProjectEvent.Type

Package ikayaki

Declaration public enum Type

The type of a project event. Options are STATE_CHANGED, DATA_CHANGED.

5.1.12 ProjectListener

Package ikayaki

Declaration public interface ProjectListener

Extends EventListener

Defines a listener for project events.

```
public void projectUpdated(ProjectEvent event)
```

Parameter 1 *event* -

5.1.13 MeasurementEvent

Package ikayaki

Declaration public class MeasurementEvent

Extends EventObject

Created by Project (5.1.1)

MeasurementEvent is used to notify listeners about the stages of an ongoing measurement.

```
private Project project
```

```
private MeasurementStep step
```

```
private Type type
```

```
public MeasurementEvent(Project project, MeasurementStep
step, Type type)
```

Parameter 1 *project* -

Parameter 2 *step* -

Parameter 3 *type* -

Returns

Throws -

```
public Project getProject()
```

Returns

Throws -

```
public MeasurementStep getStep()
```

Returns

Throws -

```
public Type getType()
```

Returns

Throws -

5.1.14 MeasurementEvent.Type

Package ikayaki

Declaration public enum Type

The type of a measurement event. Options are STEP_START, STEP_END, STEP_ABORTED, HANDLER_MOVE, HANDLER_ROTATE, HANDLER_STOP, DEMAGNETIZE_START, DEMAGNETIZE_END, VALUE_MEASURED.

5.1.15 MeasurementListener

Package ikayaki

Declaration public interface MeasurementListener

Extends EventListener

Defines a listener for measurement events.

```
public void measurementUpdated(MeasurementEvent event)
```

Parameter 1 *event* -

5.2 Squid interface

5.2.1 Squid

Package ikayaki.squid
Declaration public class Squid

Extends

Implements

Created by MainViewPanel (6.2.1)

Uses 1 Settings (5.5.1)

Subclass 1 Degausser (5.2.2)

Subclass 2 Handler (5.2.3)

Subclass 3 Magnetometer (5.2.4)

offers Squid interface for project-class that controls SQUID-system. It reads Settings-class for settings and creates classes degausser, handler and magnetometer and offers handlers for them.

Design patterns This class is singleton, there will be always only one instance of it and its created when class is needed first time.

```
private Project owner
    project currently using squid-interface

private Degausser degausser
    class for commanding degausser

private Handler handler
    class for commanding handler

private Magnetometer magnetometer
    class for commanding magnetometer

public Degausser getDegausser()
    Gives handler for Degausser.
Returns Degausser object if its not busy

public Handler getHandler()
    Gives handler for Degausser.
Returns Handler object.

public Magnetometer getMagnetometer()
    Gives handler for Magnetometer
Returns Magnetometer object.

public void updateSettings()
    When settings are saved, update all settings in subclasses.

private Squid()
    Creates instances of Degausser, Handler and Magnetometer.
```

```
public Settings instance()
    creates Squid, if not yet created, and return it.

public boolean isOK()
    checks all subclasses if they are set and online.
Returns          True if ok

public boolean setOwner(Project owner)
    Sets owner to new project, and only this project can call interface. Check if Squid
    is currently busy.
Parameter 1      owner - Project that we want to use Squid now.
Returns          True if success, false if Squid was busy.

public Project getOwner()
    Returns Project currently using Squid-interface.
Returns          Project, or null if none is using Squid.
```

5.2.2 Degausser

Package ikayaki.squid
Declaration public class Degausser
Implements SerialIOListener
Created by Squid (5.2.1)
Uses 1 Settings (5.5.1)
Uses 2 SerialIO (??)

Controls Degausser (demagnetizer). Sets it up and offers Interface to control it. Because the data link is implemented in the degausser by a single board computer running a small basic program, the response time of the degausser to commands is slow. Suitable wait loops will have to be used in the external computer code to prevent unreliable communications.

Event A *On New IO Message* - reads message and puts it in Buffer

```
private Stack messageBuffer
    buffer for incoming messages, readed when needed.

private String status
    Degaussers current status

private String comPort
    COM port for communication

private int degausserCoil
    (X, Y, Z) = (0,1,2) default axis Z

private int degausserAmplitude
    0->3000 default amp 0
```



```
private int degausserDelay
    1-9 seconds default delay 1 second

private int degausserRamp
    (3, 5, 7, 9) default 3

private char degausserRamp
    Z=Zero, T=Tracking, ?=Unknown

public void setCoil(char coil)
    Sets coil X,Y,Z.
    Parameter 1      coil - coil to set on.

public void setAmplitude(int amplitude)
    Sets amplitude to ramp, range 0 to 3000.
    Parameter 1      amplitude - amplitude to demag.

public void executeRampUp()
    Performs Ramp up.

public void executeRampDown()
    Brings Ramp down.

public void executeRampCycle()
    Performs Ramp up and down.

public boolean demagnetize(int amplitude)
    Performs full sequence to demagnetize with given amplitude.
    Parameter 1      amplitude - amplitude to demag.
    Returns          If process was sended succesfully, true.

public String getStatus()
    Gives configuration and ramp status.

public boolean isOK()
    checks if connection is ok.
    Returns          True if ok

public void updateSettings()
    Squid tells if settings are changed, update all settings.

public Degausser()
    Opens connection to Degausser COM port (if not open yet) and read settings to
    fields from Setting-class.
```

5.2.3 Handler

Package	ikayaki.squid
Declaration	public class Handler
Implements	SerialIOListener
Created by	Squid (5.2.1)
Uses 1	Settings (5.5.1)
Uses 2	SerialIO (??)
	Controls Sample Handler and sets it up, offering interface for it.
Event A	<i>On New IO Message</i> - reads message and puts it in Buffer

```
private Stack messageBuffer
    buffer for incoming messages, readed when needed.

private String status
    Handlers current status

private String comPort
    COM port for communication

private int acceleration
    value between 0 and 127 default 5. Settings in the 20-50 range are usually employed.

private int deceleration
    value between 0 and 127 default 10. Settings in the 20-50 range are usually employed.

private int velocity
    value between 50 and 12 000. The decimal number issued is 10 times the actual pulse rate to the motor. Since the motor requires 200 pulses (full step) or 400 pulses (half step) per revolution, a speed setting of M10000 sets the motor to revolve at 5 revolutions per second in full step or 2.5 revolutions in half step. This rate is one-half the sample rate rotation due to the pulley ratios. The sample handler is set up at the factory for half stepping.

private int measurementVelocity
    speed in measurement, should be small

private String handlerStatus
    5 end of move, previous G command complete, 7 hard limit stop, G motor is currently indexing

private int currentPosition
    value between 1 and 16,777,215

private int homePosition
    value between 1 and 16,777,215
```

```
private int transverseYAFPosition
    AF demag position for transverse

private int axialAFPosition
    axial AF demag position in steps, must be divisible by 10. Relative to Home.

private int backgroundPosition
    Position in steps, must be divisible by 10. Relative to Home.

private int measurementPosition
    Position in steps, must be divisible by 10. Relative to Home.

private int currentRotation
    angles are between 0 (0) and 2000 (360)

public void updateSettings()
    Squid tells if settings are changed, update all settings.

public String getStatus()
    Returns current status on Sample Handler.

public boolean isOK()
    checks if connection is ok.
Returns          True if ok

public void moveToHome()
    Send command to sample holder to move home

public void moveToHome()
    Send handler to Degaus position

public void moveToMeasurement()
    Send handler to Measure position

public void moveToBackground()
    Send handler to Background position

public boolean moveToPos(int pos)
    Value must be between 1 and 16,777,215. return true if good pos-value and moves
    handler there.
Parameter 1    pos - Position where handler are sent
Returns          if given position was ok, true.

public void stop()
    Tells handler to stop its curren job.

public void rotateTo(int angle)
    Value is in degrees, remainder of divided by 360. Rotates handler that much.
Parameter 1    angle - Angle in degrees to rotate handler.
```

```
public Handler()
```

Opens connection to Handler COM port and read settings to fields from Setting-class.

5.2.4 Magnetometer

Package ikayaki.squid

Declaration public class Magnetometer

Implements SerialIOListener

Created by Squid (5.2.1)

Uses 1 Settings (5.5.1)

Uses 2 SerialIO (??)

Controls Magnetometer and sets it up, offering interface for it. Commands are at most five characters in length including a carriage return <CR>. The syntax is as follows: "<device><command><subcommand><data><CR>"

Event A *On New IO Message* - reads message and puts it in Buffer

```
private Stack messageBuffer
```

buffer for incoming messages, readed when needed.

```
private String status
```

Magnetometers current status

```
private String comPort
```

COM port for communication

```
public void updateSettings()
```

Squid tells if settings are changed, update all settings.

```
public String reset(char axis)
```

Reset settings on axis

Parameter 1 *axis* - x,y,x or a (all)

```
public String resetCounter(char axis)
```

Reset counter for axis.

Parameter 1 *axis* - x,y,x or a (all)

```
public String configure(char axis, char subcommand, char
option)
```

Parameter 1 *axis* - x,y,x or a (all)

Parameter 2 *subcommand* - The CONFIGURE subcommands follow: "F" Set filter configuration. The data subfield sets the filter to the indicated range. The four possible data values are: "1" One Hertz Filter; 1 Hz "T" Ten Hertz Filter; 10 Hz "H" One hundred Hertz Filter; 100 Hz "W" Wide band filter; WB "R" Set DC SQUID electronic range. The data subfield selects the range desired. The four possible data values are: "1" One time range; 1x "T" Ten times range; 10x "H" One hundred times range; 100x "E" Extended range; 1000x "S" Set/Reset the fast-slew option. Two data values are possible: "E" Enable the fast-slew; turn it on. "D" Disable the fast-slew; turn it off. "L" This subcommand opens or closes the SQUID feedback loop or resets the analog signal to +/- 1/2 flux quantum about zero. The three possible data values are: "O" Open the feedback loop. (This command also zeros the flux counter) "C" Close the feedback loop. "P" Pulse-reset (open then close) the feedback loop. (This command also zeros the flux counter)

Parameter 3 *option* - see data values from subcommands.

```
public void latchAnalog(char axis)
```

axis is x,y,x or a (all).

Parameter 1 *axis* - x,y,x or a (all)

```
public void latchCounter(char axis)
```

axis is x,y,x or a (all).

Parameter 1 *axis* - x,y,x or a (all)

```
public void measure(char axis)
```

Performs full sequence to measure axis.

Parameter 1 *axis* - x,y,x.

```
public String getData(char axis, char command, String
datavalues)
```

Generic send message sender, use with cautions and knowledge. Check if commands are good.

Parameter 1 *axis* - x,y,x

Parameter 2 *command* - "D" Send back the analog data last captured with the LATCH command. The <data> field is not required. "C" Send back the counter value last captured with the LATCH command. The <data> field is not required. "S" Send back status. Various pieces of status can be sent by the magnetometer electronics.

Parameter 3 *axis* - Datavalues one or more: "A" Send back all status. "F" Send back all filter status. "R" Send back all range status. "S" Send back slew status. "L" Send back SQUID feedback loop status. Return feedback, waiting time?

```
public String getStatus()
```

Returns current status on Sample Handler.

```
public boolean isOK()
```

checks if connection is ok.

Returns True if ok

```
public Magnetometer()
```

Opens connection to Magnetometer COM port (if its not open already) and read settings to fields from Setting-class.

5.3 Squid emulator

5.3.1 SquidEmulator

Package ikayaki.squid

Declaration public class SquidEmulation

Extends Thread

Implements SerialIOListener

Created by (??)

Uses 1 SerialIO (??)

This class tries to emulate behavior of real squid-system. It generates random data values as results and generates random error situations to see that program using real squid system does survive those. Uses 2-3 COM ports. Usage SquidEmulator x z.. where x is 0 or 1 and indicates if Magnetometer and Demagnetizer are on same COM port. z... values are COM ports.

Event A *On New IO Message* - reads message and puts it in Buffer

```
private Stack messageBuffer
```

buffer for incoming messages

private bool online
indicates if system have been started

private int acceleration
value between 0 and 127 default 5. Settings in the 20-50 range are usually employed.

private int deceleration
value between 0 and 127 default 10. Settings in the 20-50 range are usually employed.

private int velocity
value between 50 and 12 000. The decimal number issued is 10 times the actual pulse rate to the motor. Since the motor requires 200 pulses (full step) or 400 pulses (half step) per revolution, a speed setting of M10000 sets the motor to revolve at 5 revolutions per second in full step or 2.5 revolutions in half step. This rate is one-half the sample rate rotation due to the pulley ratios. The sample handler is set up at the factory for half stepping.

private String handlerStatus
5 end of move, previous G command complete, 7 hard limit stop, G motor is currently indexing

private int commandedDistance
value between 1 and 16,777,215

private int currentPosition
value between 1 and 16,777,215

private int homePosition
value between 1 and 16,777,215

private int commandedRotation
angles are between 0 (0) and 2000 (360)

private int currentRotation
angles are between 0 (0) and 2000 (360)

private int degausserCoil
(X, Y, Z) = (0,1,2) default axis Z

private int degausserAmplitude
0->3000 default amp 0

private int degausserDelay
1-9 seconds default delay 1 second

private int degausserRamp
(3, 5, 7, 9) default 3

```

private char degausserRamp
    Z=Zero, T=Tracking, ?=Unknown

private SerialIO[] messageReader
    starts Threads which reads messages from selected COM port

public void getSequences()
    reads message and commits it.

public void writeMessage(String message ,int port))
    send message to SerialIO to be sented.
Parameter 1    message - any message reply we are sending back
Parameter 2    port - port number to be sent

public void run()
    runs sequence where read data from buffer and run cheduled actions (move, rotate,
    demag, measure) and send feedback to COM ports.

```

5.4 Serial communication

5.5 Global settings

5.5.1 Settings

Package ikayaki
Declaration public class Settings
 Singleton class for holding all global settings. It also holds saved sequences. All changes are automatically saved to file after a short delay.
Design patterns Singleton

```

private Properties properties
Default value    new Properties()
    All properties in a map. Keys are: magnetometerPort(String), demagnetiz-
    erPort(String), PorthandlerPort(String), xAxisCalibration(double), yAxisCalibra-
    tion(double), zAxisCalibration(double), demagRamp(int), demagDelay(int), ac-
    celeration(int), deceleration(int), velocity(int), measurementVelocity(int), trans-
    verseYPosition(int), axialPosition(int), sampleLoadPosition(int), backgroundPosi-
    tion(int), measurementPosition(int), rotation(int), handlerRightLimit(boolean)

private File propertiesFile
    File where the properties will be saved in XML format

private boolean propertiesModified
    true if the properties have been modified, otherwise false

```



```

private List<MeasurementSequence> sequences
Default value    new ArrayList<MeasurementSequence>()
All saved sequences

private File sequencesFile
File where the sequences will be saved in XML format

private bool sequencesModified
true if the sequences have been modified, otherwise false

private RunQueue autoSaveQueue
Queue for scheduling save operations after properties/sequences have been changed

public static Settings instance()
Returns          Pointer to the global Settings object. If not yet created, will first
                    create a new Settings object.

private Settings()
Loads settings from configuration files.

public void save()
Saves the settings after a while when no changes have come.

public void saveNow()
Saves the settings and keeps waiting until its done.

private String getProperty(String key)
Returns the value that maps to the specified key.
Parameter 1      key - key whose associated value is to be returned.
Returns          Value associated with key, or an empty string if none exists.

private void setProperty(String key, String value)
Associates the specified value with the specified key. Will invoke autosaving.
Parameter 1      key - key with which the specified value is to be associated.
Parameter 2      value - value to be associated with the specified key.

public Type getPropertyXXX()
Generic accessor for all properties. Returns the value from Properties in appropriate
type.
Returns          Value associated with key

public boolean setProperty(Type value)
Generic accessor for all properties. Invoke autosave. Checks if value is ok and sets
it.
Returns          True if value was ok.

public MeasurementSequence[] getSequences()
Returns all saved Sequences.

```

```
public void addSequence(MeasurementSequence sequence)
    Adds a sequence to the sequence list.
```

```
public void removeSequence(MeasurementSequence sequence)
    Removes a sequence from the sequence list. If the specified sequence is not in the
    list, it will be ignored.
```

5.6 Utilities

5.6.1 RunQueue

Package ikayaki.util
Declaration public class RunQueue
Uses 1 RunQueue.RunQueueThread (5.6.2)
Uses 2 RunQueue.RunDelayed (5.6.3)

Executes Runnable objects in a private worker thread after a pre-defined delay. The worker thread will terminate automatically when there are no runnables to be executed. Optionally executes only the last inserted runnable. All operations are thread-safe.

This class can be used for example in connection with a "continuous search" invoked by a series of GUI events (such as a DocumentListener), but it is necessary to react to only the last event after a short period of user inactivity.

Design patterns Command

```
private int delayMillis
```

Default value 0

Defines how long is the delay in milliseconds, after which the events need to be run.

```
private boolean execOnlyLast
```

Default value false

Defines if only the last event should be executed. If false, then all of the events are executed in the order of appearance.

```
private DelayQueue<RunDelayed> queue
```

Default value new DelayQueue<RunDelayed>()

Prioritized FIFO queue for containing the RunDelayed items that have not expired. If execOnlyLast is true, then this queue should never contain more than one item.

```
private Thread workerThread
```

Default value null

The worker thread that will run the inserted runnables. If the thread has no more work to do, it will set workerThread to null and terminate itself.

```
public RunQueue()
```

Creates an empty RunQueue with a delay of 0 and execOnlyLast set to false.

```
public RunQueue(int delayMillis)
```

Creates an empty RunQueue with `execOnlyLast` set to false.

Parameter 1 *delayMillis* - the length of execution delay in milliseconds; if less than 0, then 0 will be used.

```
public RunQueue(boolean execOnlyLast)
```

Creates an empty RunQueue with a delay of 0.

Parameter 1 *execOnlyLast* - if true, only the last event will be executed after the delay; otherwise all are executed in order of appearance.

```
public RunQueue(int delayMillis, boolean execOnlyLast)
```

Creates an empty RunQueue.

Parameter 1 *delayMillis* - the length of execution delay in milliseconds; if less than 0, then 0 will be used.

Parameter 2 *execOnlyLast* - if true, only the last event will be executed after the delay; otherwise all are executed in order of appearance.

```
public synchronized boolean isExecOnlyLast()
```

Returns true if only the last event will be executed after the delay; otherwise false.

```
public synchronized void setExecOnlyLast(boolean
execOnlyLast)
```

Parameter 1 *execOnlyLast* - if true, only the last event will be executed after the delay; otherwise all are executed in order of appearance.

```
public synchronized int getDelayMillis()
```

Returns the delay in milliseconds

```
public synchronized void setDelayMillis(int delayMillis)
```

Parameter 1 *delayMillis* - delay in milliseconds; if less than 0, then the new value is ignored.

```
public synchronized boolean offer(Runnable runnable)
```

Inserts a Runnable object to the end of the queue. It will remain there until it is executed or another object replaces it. If `execOnlyLast` is set to true, the queue will be cleared before inserting this runnable to it. If there is no worker thread running, a new one will be spawned.

Parameter 1 *runnable* - the Runnable to be run after a pre-defined delay

Returns true

Throws *NullPointerException* - if runnable is null

```
public synchronized void join()
```

Waits for the queue to become empty.

Throws *InterruptedException* - if another thread has interrupted the current thread. The interrupted status of the current thread is cleared when this exception is thrown.

5.6.2 RunQueue.RunQueueThread

Package ikayaki.util
Declaration private class RunQueueThread
Extends Thread
Created by RunQueue (5.6.1)

Keeps on checking the RunQueue.queue to see if there are Runnable's to be executed. If there is one, execute it and proceed to the next one. If an uncaught Throwable is thrown during the execution, prints an error message and stack trace to stderr. If the queue is empty, this thread will set RunDelayed.workerThread to null and terminate itself.

```
public void run()
```

5.6.3 RunQueue.RunDelayed

Package ikayaki.util
Declaration private class RunDelayed
Implements Delayed
Created by RunQueue (5.6.1)

Wraps a Runnable object and sets the delay after which it should be executed by a worker thread.

```
private long expires
```

The point in time when this RunDelayed will expire.

```
private Runnable runnable
```

Contained Runnable object to be run after this RunDelayed has expired.

```
public RunDelayed(Runnable runnable, int delayMillis)
```

Creates a new RunDelayed item that contains runnable.

Parameter 1 *runnable* - the Runnable to be contained

Parameter 2 *delayMillis* - delay in milliseconds

```
public long getDelay(TimeUnit unit)
```

Returns the remaining delay associated with this object, always in milliseconds.

Parameter 1 *unit* - ignored; always assumed TimeUnit.MILLISECONDS

Returns the remaining delay; zero or negative values indicate that the delay has already elapsed

```
public Runnable getRunnable()
```

Returns the contained Runnable.

Returns the Runnable given as constructor parameter

```
public int compareTo(Delayed delayed)
```

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameter 1 *delayed* - the Delayed to be compared.

Returns a negative integer, zero, or a positive integer as this delay is less than, equal to, or greater than the specified delay.

6 GUI classes and methods

6.1 Generic GUI components

6.1.1 ProjectComponent

Package ikayaki.gui

Declaration public class ProjectComponent

Extends JPanel

Created by MainViewPanel (6.2.1)

Uses 1 Project (5.1.1)

Subclass 1 ProjectInformationPanel (6.8.1)

Subclass 2 MeasurementSequencePanel (6.9.1)

Subclass 3 MeasurementDetailsPanel (6.10.1)

Subclass 4 MeasurementControlsPanel (6.11.1)

Subclass 5 MeasurementGraphsPanel (6.12.1)

Subclass 6 ProjectExplorerPanel (6.6.1)

Subclass 7 CalibrationPanel (6.7.1)

Generic gui component which uses Project and listens MeasurementEvents and ProjectEvents.

Event A *On ProjectEvent* - does nothing; subclasses may override this.

Event B *On MeasurementEvent* - does nothing; subclasses may override this.

```
private Project project
```

Active project.

```
public ProjectComponent()
```

Initializes this ProjectComponent with no Project (one probably arrives shortly with setProject(Project)).

```
public Project getProject()
```

Returns this.project.

```
public void setProject(Project project)
    Sets the project for this ProjectComponent. Unregisters MeasurementListener and
    ProjectListener from the old project, and registers them to the new project.
Parameter 1      project - new active project, or null to make no project active.
```

```
public projectUpdated(ProjectEvent event)
    Does nothing; subclasses override this if they want to listen ProjectEvents.
Parameter 1      event - ProjectEvent received.
```

```
public measurementUpdated(MeasurementEvent event)
    Does nothing; subclasses override this if they want to listen MeasurementEvents.
Parameter 1      event - MeasurementEvent received.
```

6.2 Main window

6.2.1 MainViewPanel

Package	ikayaki.gui
Declaration	public class MainViewPanel
Extends	JPanel
Created by	Ikayaki (??)
Uses 1	ProjectExplorerPanel (6.6.1)
Uses 2	CalibrationPanel (6.7.1)
Uses 3	Squid (5.2.1)
Uses 4	MainMenuBar (6.3.1)
Uses 5	MainStatusBar (6.4.1)
Uses 6	ProjectInformationPanel (6.8.1)
Uses 7	MeasurementSequencePanel (6.9.1)
Uses 8	MeasurementDetailsPanel (6.10.1)
Uses 9	MeasurementControlsPanel (6.11.1)
Uses 10	MeasurementGraphsPanel (6.12.1)

Creates the main view panels (split panels) and Squid and Project components. It also tells everybody if project is changed.

```
private ProjectExplorerPanel projectExplorer
```

```
private CalibrationPanel calibration
```

```
private Squid squid
```

```
private ProjectComponent project
    currently active project
```

```
private MainMenuBar menuBar
```

```
private MainStatusBar statusBar

private ProjectInformationPanel projectInformation

private MeasurementSequencePanel measurementSequence

private MeasurementControlsPanel measurementControls

private MeasurementDetailsPanel measurementDetails

private MeasurementGraphsPanel measurementGraphs

public MainViewPanel()
    Loads default view and creates all components and panels. Splitpanel between Cal-
    ibration,Explorer,Information and rest.

public boolean changeProject(Project project)
    Looks for file with filename, if not exist creates new other wise opens it. Then
    updates current project and tells Panels new project is opened.
```

6.3 Menu bar

6.3.1 MainMenuBar

Package	ikayaki.gui
Declaration	public class MainMenuBar
Extends	JMenuBar
Created by	MainViewPanel (6.2.1)
	Creates Menu items for Menubar and makes action listeners for them
Event A	<i>On newProject Clicked</i> - Opens File chooser and opens new file in selected folder
Event B	<i>On openProject Clicked</i> - Opens File chooser and opens selected file
Event C	<i>On exportToDAT Clicked</i> - Opens File chooser and tells Project to export in selected file
Event D	<i>On exportToDTDT Clicked</i> - Opens File chooser and tells Project to export in selected file
Event E	<i>On exportToSRM Clicked</i> - Opens File chooser and tells Project to export in selected file
Event F	<i>On configuration Clicked</i> - Opens SettingsPanel (frame)
Event G	<i>On helpItem Clicked</i> - Opens Help dialog (own frame?)
Event H	<i>On about Clicked</i> - Opens dialog with credits and version number
Event I	<i>On exit Clicked</i> - closes program

```
private JMenu file
```

```
private JMenu options
```

```
private JMenu help
```

```
private Action newProject
```

```
private Action openProject
```

```
private JMenu exportProject
```

```
private Action exportProjectToDAT
```

```
private Action exportProjectToDTD
```

```
private Action exportProjectToSRM
```



```
private Action exit

private Action configuration

private Action helpItem

private Action about

public MainMenuBar()
    Creates all components and makes menu and sets ActionListeners.
```

6.4 Status bar

6.4.1 MainStatusBar

Package ikayaki.gui
Declaration public class MainStatusBar
Extends ProjectComponent
Created by MainViewPanel (6.2.1)
 Creates its components and listens project events on status change and calculates estimated time for measurement
Event A *On Measurement Event* - recalculates progress and updates status for current measurement

```
private JLabel measurementStatus
    text comment of current status(moving,measurement,demagnetization)

private JProgressBar measurementProgress
    progress of sequence/measurement as per cent of whole process

private int[] currentSequence
    current projects sequence

private int projectType
    current projects type (we know if we are doing demagnetization or not)

public MainStatusBar()
    Creates all components with default settings and sets Listener for MeasurementEvent.

        private void calculateStatus(String phase, int
sequenceStep, int currentStep)
    Recalculates current progress and updates status.
```

```
private void setMeasurement(int projectType, int[]
sequence)
```

Formats status and creates new measurement status values.

6.5 Configuration window

6.5.1 SettingsPanel

Package ikayaki.gui
Declaration public class SettingsPanel
Extends JFrame
Created by MainStatusBar (6.4.1)
Uses 1 Settings (5.5.1)
Uses 2 Squid (5.2.1)

Creates its components and updates changes to Settings and saves them in Configuration file

Event A *On Save Clicked* - saves current configuration to Settings-singleton and closes window

Event B *On Cancel Clicked* - closes window (discarding changes)

```
private JComboBox magnetometerPort
COM port for magnetometer
```

```
private JComboBox demagnetizerPort
COM port for demagnetizer, can be sharing same port with magnetometer
```

```
private JComboBox handlerPort
COM port for sample handler
```

```
private JTextField xAxisCalibration
Calibration constants with polarization (factory set?)
```

```
private JTextField yAxisCalibration
Calibration constants with polarization (factory set?)
```

```
private JTextField zAxisCalibration
Calibration constants with polarization (factory set?)
```

```
private JComboBox demagRamp
how fast demagnetization goes
```

```
private JComboBox demagDelay
?
```

```
private JTextField acceleration
Handler acceleration
```

```
private JTextField deceleration
    Handler deceleration

private JTextField velocity
    Handler Max speed

private JTextField measurementVelocity
    speed in measurement, should be small

private JTextField transverseYAFPosition
    AF demag position for transverse

private JTextField axialAFPosition
    axial AF demag position in steps, must be divisible by 10. Relative to Home.

private JTextField sampleLoadPosition
    Position in steps, must be divisible by 10. Relative to Home. (same as Home?)

private JTextField backgroundPosition
    Position in steps, must be divisible by 10. Relative to Home.

private JTextField measurementPosition
    Position in steps, must be divisible by 10. Relative to Home.

private JTextField rotation
    steps to perform full rotation, must be clockwise, determined by sign

private JComboBox handlerRightLimit
    Refers to right limit switch on translation axis. And usually sample holder motion
    toward right limit is positive direction (default).

private JButton saveButton

private JButton cancelButton

public SettingsPanel()
    Creates all components and puts them in right places. Labels are created only here
    (no global fields). Creates ActionListeners for buttons.

public void closeWindow()
    Closes window, no changes saved.

public void saveSettings()
    Saves all settings to Settings-singleton and calls closeWindow().
```

6.6 Project Explorer

6.6.1 ProjectExplorerPanel

Package	ikayaki.gui
Declaration	public class ProjectExplorerPanel
Extends	ProjectComponent
Created by	MainViewPanel (6.2.1)
Uses 1	MainViewPanel (6.2.1)
Uses 2	ProjectExplorerTable (6.6.3)
Uses 3	ProjectExplorerPopupMenu (6.6.4)
Uses 4	NewProjectPanel (6.6.2)
Uses 5	RunQueue (5.6.1)

Creates a history/autocomplete field (`browserField`) for choosing the project directory, a listing of project files in that directory (`explorerTable`) and in that listing a line for creating new project, which has a textbox for project name, an AF/TH `ComboBox` and a "Create new" button (`createNewProjectButton`) for actuating the creation. Also has a right-click popup menu for exporting project files.

Event A	<i>On browserField change</i> - send autocomplete-results-finder with <code>browserField</code> 's text to <code>RunQueue</code> via <code>runQueue.offer(Runnable)</code> , which schedules disk access and displaying autocomplete results in <code>browserField</code> 's popup window.
Event B	<i>On browserField down-arrow-click</i> - show directory history in <code>browserField</code> 's popup window.
Event C	<i>On browseButton click</i> - open a <code>FileChooser</code> dialog for choosing new directory and tell <code>explorerTable</code> and <code>browserField</code> to change to that directory.
Event D	<i>On ProjectEvent</i> - highlight project whose measuring started, or un-highlight one whose measuring ended.

```
private JComboBox browserField
  Text field for writing directory to change to. Autocomplete results appear to Combo
  Box' popup window, scheduled by RunQueue. Directory history appears to the
  same popup window when the down-arrow right to text field is clicked.
```

```
private JButton browseButton
```

```
private ProjectExplorerTable explorerTable
```

```
private NewProjectPanel newProjectPanel
```

```
private RunQueue autocompleteQueue
```

Default value `new RunQueue(100, true)`

`RunQueue` for scheduling autocomplete results to separate thread (disk access and displaying).

```
private File directory
```

Default value null

Currently open directory.

```
private Vector<File> files
```

Default value new Vector<File>()

Project files in current directory.

```
public ProjectExplorerPanel(Project project)
```

Creates all components, sets directory as the last open directory, initializes files with files from that directory.

```
public void setProject(Project project)
```

Call super.setProject(project), highlight selected project, or unhighlight unselected project.

6.6.2 NewProjectPanel

Package ikayaki.gui

Declaration public class NewProjectPanel

Extends JPanel

Created by ProjectExplorerPanel (6.6.1)

Uses 1 MainViewPanel (6.2.1)

Panel with components for creating a new project. This Panel will be somewhere below the project file listing...

Event A *On createNewProjectButton click* - call Project.createXXXProject(File) with filename from newProjectField; if returns null, show error message and do nothing. Otherwise, update file listing, set new project active, tell explorerTable to reset newProjectField and newProjectType and call MainViewPanel.changeProject(Project) with returned Project.

```
private JTextField newProject
```

```
private JComboBox newProjectType
```

Default value AF/Thellier/Thermal

```
private JButton createNewProjectButton
```

6.6.3 ProjectExplorerTable

Package	ikayaki.gui
Declaration	public class ProjectExplorerTable
Extends	JTable
Created by	ProjectExplorerPanel (6.6.1)
Uses 1	MainViewPanel (6.2.1)

Creates a list of project files in directory. Handles loading selected projects and showing export popup menu (ProjectExplorerPopupMenu). Inner class of ProjectExplorerPanel.

Event A	<i>On explorerTable click</i> - call Project.loadProject(File) with clicked project file, call MainViewPanel.changeProject(Project) with returned Project unless null, on which case show error message and revert explorerTable selection to old project, if any.
Event B	<i>On explorerTable mouse right-click</i> - create a ProjectExplorerPopupMenu for right-clicked project file.

```
private TableModel projectExplorerTableModel
    TableModel which handles data from files (in upper-class ProjectExplorerPanel).
    Unnamed inner class.
```

6.6.4 ProjectExplorerPopupMenu

Package	ikayaki.gui
Declaration	public class ProjectExplorerPopupMenu
Extends	JPopupMenu
Created by	ProjectExplorerPanel (6.6.1)
	Shows popup menu with export choices: AF (.dat), Thellier (.tdt) and Thermal (.tdt), and executes selected.
Event A	<i>On selectItem mouseEvent</i> - call project.exportXXX()...

6.7 Calibration

6.7.1 CalibrationPanel

Package	ikayaki.gui
Declaration	public class CalibrationPanel
Extends	ProjectComponent
Created by	MainViewPanel (6.2.1)
Uses 1	MainViewPanel (6.2.1)
Uses 2	Project (5.1.1)

Holds predefined "Holder noise" and "Standard sample" projects for calibration; they are in a technically same table as Project explorer files. Also has a "Calibrate" button, which executes selected calibration project, similarly to clicking "Single step" in normal projects.

Event A	<i>On calibrateButton click</i> - call <code>project.doSingleStep()</code> ; show error message if false is returned.
Event B	<i>On calibrationProjectTable click</i> - call <code>Project.loadProject(File)</code> with clicked project file (calibrationProjectTable row); call <code>MainViewPanel.changeProject(Project)</code> with returned Project unless null, on which case show error message and revert calibrationProjectTable selection to old project, if any.
Event C	<i>On ProjectEvent</i> - highlight calibration project whose measuring started, or unhighlight one whose measuring ended; enable calibrateButton if measuring has ended, or disable if measuring has started.

```
private JButton calibrateButton
```

```
private JTable calibrationProjectTable
  Table for the two calibration projects; has "filename", "last modified" and "time"
  (time since last modification) columns.
```

```
private TableModel calibrationProjectTableModel
  TableModel which holds the data for calibration projects. Unnamed inner class.
```

```
public void setProject(Project project)
  Call super.setProject(project), highlight selected calibration project, or unhighlight un-
  elected calibration project.
```

6.8 Project information

6.8.1 ProjectInformationPanel

Package	ikayaki.gui
Declaration	public class ProjectInformationPanel
Extends	ProjectComponent
Created by	MainViewPanel (6.2.1)
	Allows inserting and editing project information.
Event A	<i>On change of contest in textfield</i> - Notify project about change in project information.
Event B	<i>On project event</i> - Update textfields to correspond new project information.

```

private JLabel operatorLabel

private JTextFields operatorTextField

private JLabel dateLabel

private JTextFields dateTextField

private ButtonGroup measurementType
  Groups autoMeasurement and manualMeasurement radiobuttons.

private JRadioButton autoMeasurement

private JRadioButton manualMeasurement

private JLabel rocktypeLabel

private JTextFields rocktypeTextField

private JLabel siteLabel

private JTextFields siteTextField

private JLabel commentLabel

private JTextFields commentTextField

```



```
private JLabel latitudeLabel

private JTextFields latitudeTextField

private JLabel longLabel

private JTextFields longTextField

private JLabel strikeLabel

private JTextFields strikeTextField

private JLabel dipLabel

private JTextFields dipTextField

private JLabel volumeLabel

private JTextFields volumeTextField

private JLabel massLabel

private JTextFields massTextField

private ButtonGroup sampleType
    Groups coreSample and handSample radiobuttons.

private JRadioButton coreSample

private JRadioButton handSample

public ProjectInformationPanel()
    Creates default ProjectInformationPanel.

public ProjectInformationPanel(Project project)
    Creates ProjectInformationPanel with information taken from project.

private void setProject(Project project)
    Calls super.setProject(project) and updates textfield with new projects data.
```

6.9 Sequence and measurement data

6.9.1 MeasurementSequencePanel

Package	ikayaki.gui
Declaration	public class MeasurementSequencePanel
Extends	ProjectComponent
Created by	MainViewPanel (6.2.1)
Uses 1	MeasurementSequenceTableModel (6.9.2)
Uses 2	MeasurementSequencePopupMenu (6.9.3)

Allows creating, editing and removing measurement sequences. Shows measurement data. Right-click brings popup menu for hiding columns, and saving sequence. Left-click selects a row. Multiple rows can be selected by painting or ctrl-clicking. Allows dragging rows to different order. Has three textfields for inserting new sequences, first field for start value, second for step and third for stop value. Clicking Add sequence-button appends sequence into table. Saved sequences can be loaded from dropdown menu.

Event A	<i>On SequenceTable mouse right-click</i> - Create a MeasurementSequencePopupMenu.
Event B	<i>On addSequence mouseclick</i> - Add measurement sequence to end of table and tell MeasurementSequenceTableModel to update itself.
Event C	<i>On sequenceSelector mouseclick</i> - Bring dropdown menu for selecting premade sequence.
Event D	<i>On selecting sequence from dropdown menu</i> - Add measurement sequence to table and tell MeasurementSequenceTableModel to update itself.
Event E	<i>On Project event</i> - Update contest of table to correspond projects state.
Event F	<i>On Measurement event</i> - If measurement step is finished, add measurement data to appropriate row and if row being measured was selected select next row unless measurement sequence ended.

```
private JButton addSequence

private JComboBox sequenceSelector

private JTextField sequenceStart

private JTextField sequenceStep

private JTextField sequenceStop
```

```
private JTable sequenceTable

private MeasurementSequenceTableModel tableModel

public MeasurementSequencePanel()
    Creates default MeasurementSequencePanel.

public MeasurementSequencePanel(Project project)
    Creates MeasurementSequencePanel and calculates shown data from project's measurement data. Highlights right row.

private void addSequence()
    Adds sequence determined by textfields to end of table.

private void setProject(Project project)
    Calls super.setProject(project), clears table and calculates shown data from project's measurement data. Highlights right row.
```

6.9.2 MeasurementSequenceTableModel

Package ikayaki.gui
Declaration public class MeasurementSequenceTableModel
Extends AbstractTableModel
Created by MeasurementSequencePanel (6.9.1)
 Handles data in table.

```
private Vector<TableColumns> shownColumns
    Currently shown columns.
```

```
private Vector<TableColumns> allColumns
    All possible columns.
```

```
public MeasurementSequenceTableModel()
    Creates SequenceTableModel
```

```
public void showColumn(String name)
    Shows named column.
```

Parameter 1 *name* - name of the column to be shown

```
public void hideColumn(String name)
    Hides named column.
```

Parameter 1 *name* - name of the column to be hidden

6.9.3 MeasurementSequencePopupMenu

Package ikayaki.gui
Declaration public class MeasurementSequencePopupMenu
Extends JPopupMenu
Created by MeasurementSequencePanel (6.9.1)
 Allows selection if volume is shown in table and saving sequence.

```
private JCheckBox volume

private JLabel nameLabel

private JCheckBox nameTextField

public MeasurementSequencePopupMenu()
  Creates SequencePopupMenu.

private void saveSequence()
  Saves current sequence into dropdown menu.
```

6.10 Measurement details

6.10.1 MeasurementDetailsPanel

Package ikayaki.gui
Declaration public class MeasurementDetailsPanel
Extends ProjectComponent
Created by MainViewPanel (6.2.1)
 Shows details of measurement selected in MeasurementSequencePanel.
Event A *On project event* - Update tables to correspond projects new state.
Event B *On change of selected row in MeasurementSequencePanel* -
 Change tables to correspond selected row.
Event C *On measurement event* - If row corresponding to ongoing measurement is selected in MeasurementSequencePanel update tables with new measurement data.

```
private JTable measurementDetails
  X, Y and Z components of BG1, 0, 90, 180, 270, BG2

private JTable errorDetails
  S/D, S/H and S/N of error

private DefaultTableModel tableMoled
```

```

public MeasurementDetails()
    Creates default MeasurementDetailsPanel.

public MeasurementDetails(Project project)
    Creates MeasurementDetailsPanel with measurement details taken from project.

private void setProject(Project project)
    Calls super.setProject(project), clears tables and shows new projects measurement
    details.

```

6.11 Measurement controls

6.11.1 MeasurementControlsPanel

Package	ikayaki.gui
Declaration	public class MeasurementControlsPanel
Extends	ProjectComponent
Created by	MainViewPanel (6.2.1)
Uses 1	Project (5.1.1)
Uses 2	MagnetometerStatusPanel (6.11.2)
Uses 3	ManualControlsPanel (6.11.3)

Has "Measure"/"Pause", "Single step" and "Stop now!" buttons for controlling measurements; "+z/-z" radiobuttons for changing sample orientation, help picture for inserting sample, picture of current magnetometer status, and, manual controls. Listens MeasurementEvents and ProjectEvents, and updates buttons and magnetometer status accordingly.

Event A	<i>On measureButton click</i> - call project.doAutoStep() or project.doPause(), depending on current button status. Show error message if false is returned.
Event B	<i>On singlestepButton click</i> - call project.doSingleStep(); show error message if false is returned.
Event C	<i>On stopButton click</i> - call project.doAbort(); show critical error message if false is returned.
Event D	<i>On ProjectEvent</i> - update buttons and manual controls according to project.isXXXEnabled().
Event E	<i>On MeasurementEvent</i> - call magnetometerStatusPanel.updateStatus(int, int) with the right values from MeasurementEvent.

```

private JButton measureButton
    Measure/pause -button; "Measure" when no measuring is being done, "Pause" when
    there is ongoing measuring sequence.

private JButton singlestepButton

```

```

private JButton stopButton

private JRadioButton zPlusRadioButton

private JRadioButton zMinusRadioButton

private JPanel sampleInsertPanel
  Draws a help image for sample inserting.

private MagnetometerStatusPanel magnetometerStatusPanel

private ManualControlsPanel manualControlsPanel

```

6.11.2 MagnetometerStatusPanel

Package ikayaki.gui
Declaration public class MagnetometerStatusPanel
Extends JPanel
Created by MeasurementControlsPanel (6.11.1)
 Picture of current magnetometer status, with sample holder position and rotation.

```

public MagnetometerStatusPanel()
  Sets magnetometer status to current position.

```

```

public updateStatus(int position, int rotation)
  Updates magnetometer status picture; called by MeasurementControlsPanel when
  it receives MeasurementEvent.

```

Parameter 1 *position* - sample holder position, from 1 to 16777215.
Parameter 2 *rotation* - sample holder rotation, from 0 (angle 0) to 2000 (angle 360).

6.11.3 ManualControlsPanel

Package ikayaki.gui
Declaration public class ManualControlsPanel
Extends JPanel
Created by MeasurementControlsPanel (6.11.1)
 Magnetometer manual control radiobuttons.
Uses 1 Project (5.1.1)
Event A *On xxxN click* - call project.xxxN(). Or something like that. I shouldn't care.

```
private JRadioButton demagX  
  
private JRadioButton demagY  
  
private JRadioButton demagZ  
  
private JRadioButton measureX  
  
private JRadioButton measureY  
  
private JRadioButton measureZ
```

6.12 Graphs

6.12.1 MeasurementGraphsPanel

Package	ikayaki.gui
Declaration	public class MeasurementGraphsPanel
Extends	ProjectComponent
Created by	MainViewPanel (6.2.1)
Uses 1	MainViewPanel (6.2.1)
Uses 2	Project (5.1.1)

7 Package structure

8 Bibliography