Finken Java API Documentation

Vladimir Velinov

March 20, 2016

Contents

1 Abstract

The Finken Java API is a collection of Java projects. On top of these projects is build a small GUI application, that allows to connect the V-REP Simulation environment with the flying Finken quadrocopters.
This document provides information about how to install, use and extend the Finken Java API.

2 Definitions

In this document some abstract paths to directories are used. Below is a list of them:

- GITHUB_HOME represents the path to the GitHub directory where the master branch of the Simulation repository has been checked-out
- JAVA_HOME represents the path to the directory where the Java SE Development Kit has been installed
- PAPARAZZI_HOME represents the path to the directory where the Paparazzi software is installed.

3 Installation instructions

3.1 Java SE Development Kit installation

In order to compile the projects, the Java SE Development Kit should be installed. If no version of Java is installed, please visit Oracle website and download the appropriate version for your OS.

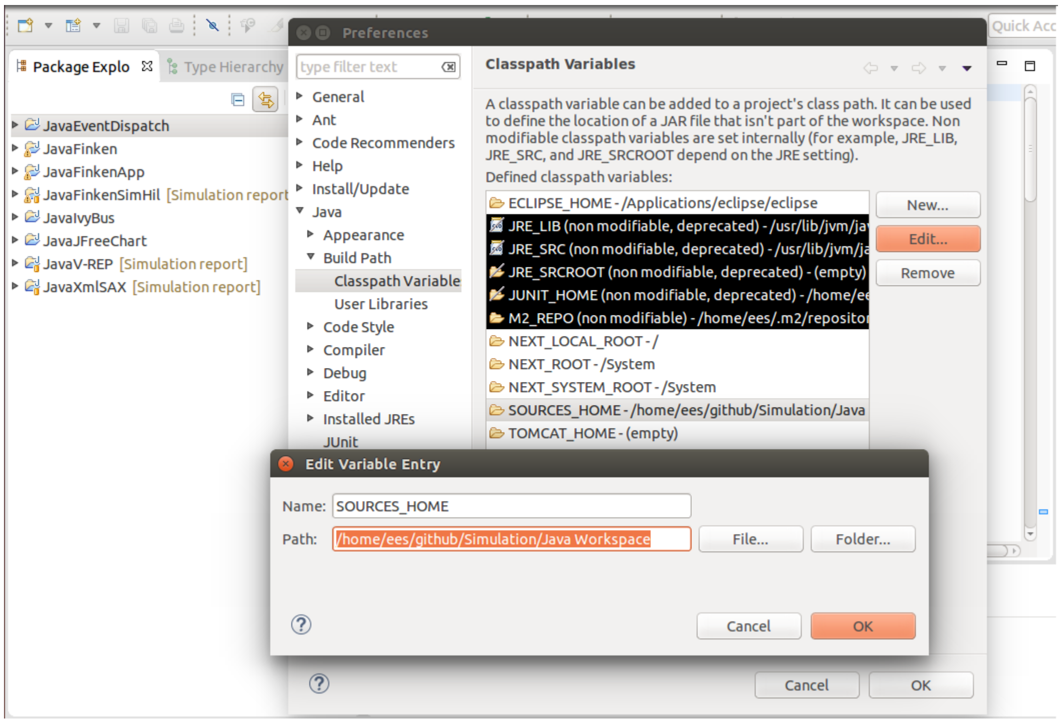
3.2 Eclipse installation

If no Eclipse version is installed, please go to Eclipse website and install the newest version of Eclipse IDE.

3.3 Importing and compiling the projects

3.3.1 Importing the projects in Eclipse

- Navigate to File →Import →Existing Projects in the Workspace
- Click on *"Select root Directory"* and press *"Browse"*
- Navigate to the GitHub folder where the projects are checked-out: "GITHUB_HOME/Simulation/Java Workspace/" and import all projects



Classpath Variables

A classpath variable can be added to a project's class path. It can be used to define the location of a JAR file that isn't part of the workspace. Non modifiable classpath variables are set internally (for example, JRE_LIB, JRE_SRC, and JRE_SRCROOT depend on the JRE setting).

Defined classpath variables:

```
ECLIPSE_HOME - /Applications/eclipse/eclipse
JRE_LIB (non modifiable, deprecated) - /usr/lib/jvm/ja
JRE_SRC (non modifiable, deprecated) - /usr/lib/jvm/ja
JRE_SRCROOT (non modifiable, deprecated) - (empty)
JUNIT_HOME (non modifiable, deprecated) - /home/ee
M2_REPO (non modifiable) - /home/ees/.m2/reposito
NEXT_LOCAL_ROOT - /
NEXT_ROOT - /System
NEXT_SYSTEM_ROOT - /System
SOURCES_HOME - /home/ees/github/Simulation/Java
TOMCAT_HOME - (empty)
```

Edit Variable Entry

Name: SOURCES_HOME

Path: /home/ees/github/Simulation/Java Workspace

Figure 1: Classpath Variable creation

3.3.2 Creating classpath variable

The projects require a classpath variable called *"SOURCES_HOME"* in order to locate the third-party libraries. Follow the steps below to create the classpath variable:

- Navigate through: Window →Preferences →Java →Build Path →Classpath Variable
- Click on *"New"* and create a Classpath Variable called "SOURCES_HOME", which points to the folder: /GITHUB_HOME/Simulation/Java Workspace.

On Figure 1 you can see how it should look like if the classpath variable is set-up correctly.

3.3.3 Add third party libraries to CLASSPATH

- Add the *remoteApiJava* Java V-REP Client to /JAVA_HOME/jre/lib/amd64/ or /JAVA_HOME/jre/lib/i386/ depending on your PC architecture. The *remoteApiJava* is located in V-REP's installation directory, under programming/remoteApiBindings/java

You need *remoteApiJava.dll* or *remoteApiJava.so* depending on your target platform.

- Add the Java Ivy-Bus library to /JAVA_HOME/jre/lib/ext/
  More detailed information can be found here.

3.3.4 Fixing hard-coded paths in code

Since the project relies on reading some Paparazzi XML configuration files, there are several hard-coded paths to files, which have to be corrected:

- Project: **JavaFinkenApp** class: **FinkenSimBridge** variable: **TELEMETRY_FILE**
- Project: **JavaFinken** class: **MessageXmlReader** variable: **DEFAULT_FILE_LOCATION**
- Project: **JavaFinken** class: **AircraftXmlReader** variable: **DEFAULT_FILE_LOCATION**

Go to the above listed classes and correct the variables, so that they point to the right directory on your PC. **Tipp**: use SHIFT + CTRL + T in Eclipse to locate the classes easily.
The package: *velinov.java.finken.app.filesaver* in project *JavaFinkenApp* can be used to implement a GUI based selection of the correct paths and to save them in a file.

3.3.5 Compiling and running the program

Now the workspace should contain no errors and you can start the program. Go to the project *JavaFinkenApp* and navigate to the class *FinkenSimBridge*, which contains the main method. Right click on it and choose "Run as Java Application". The project should compile and the application must start.
If you get *UnsatisfiedLinkError* this means that the project cannot find the libraries. Go and repeat step 3.3.3.
If a *FileNotFoundException* is thrown, it means that the project cannot locate a file. Go and repeat step 3.3.4.

4 Projects description

This section provides a very brief description of the Java projects.
To see a full description of all classes refer to the JavaDoc located in:
`GITHUB_HOME/Simulation/Java_Workspace/doc/index.html`

4.0.6 JavaV-REP

The *JavaV-REP* project is an extension of the V-REP Remote API. The interfaces *VrepClient* and *VrepServer* and their standard implementations make the connection with V-REP.

The class *AbsVrepObject* describes a V-REP object with its *VrepObjectName*, *VrepObjectType* and other properties.
The class *VrepScene* describes the V-REP simulation scene with all its objects.
To see a full description of all classes refer to the JavaDoc located in:
`GITHUB_HOME/Simulation/Java_Workspace/doc/index.html`

4.0.7 JavaIvyBus

The *JavaIvyBus* is an extension of the Ivy-Bus library. The most important class in this project is the *AbsIvyBusNode*, that each node, that wants to communicate on the Ivy-Bus has to extend.
The *AbsMessage* with its *MessageField*s describe the messages that the Ivy-Bus node can receive and subscribe to. It has the same structure as the messages defined in: `PAPARAZZI_HOME/conf/messages.xml`, since the *Message*s that the *AbsIvyBusNode* subscribes to are parsed from this XML file. See subsection 4.0.8.

4.0.8 JavaXmlSAX

This project defines the *AbsXmlSaxReader* class, which can be extended to create a custom XML reader. There are several custom XML readers defined in *JavaFinken* project, see 4.0.9, that read a specific Paparazzi XML file: *AircraftXmlReader*, *TelemetryXmlReader*, *MessageXmlReader* etc.

4.0.9 JavaFinken

The *JavaFinken* project uses the projects described in 4.0.6, 4.0.7 and 4.0.8 and describes the basic logic of the application.

4.0.10 JavaFinkenAPP

This projects defines the example GUI application build on top of 4.0.9. It provides a basic GUI to facilitate the connection with V-REP. The *FinkenSimBridge* is the class that contains the *main* method and starts the application. The *FinkenSimBridgeController* manages the connection between the View and the other classes. This class is a good starting point if one wants to make modifications on the API.
You can use and modify this application to meet your needs or create other mixed-reality scenarios.

5 Starting the Mixed-Reality Simulation

5.1 Create V-REP real quadrocopters

In order to create a real V-REP quadrocopter, a quadrocopter that is coupled to the real flying quadrocopter, the name of the V-REP object should match the name of the flying quadrocopter as configured in the Paparazzi XML file.

The `PAPARAZZI_HOME/conf/conf.xml` file contains a description of all quadrocopters with its names and other parameters. For example if you want to fly with the *Finken3_01* and have a V-REP real quadrocopter, that is coupled to it, simply name the V-REP quadrocopter with the same name as the real one.
If you want to fly with multiple quadrocopters and have multiple V-REP coupled quadrocopters, simply copy-paste the V-REP quadrocopters. Again the names must match with the names defined in the XML file.
If an object is copy-pasted (multiple instances of an object), then each instance of the object receives the following name, according to V-REP naming scheme: *base_name#index*. For example if we want to have three quadrocopters, their names in V-REP will be represented as follows: *Quad_Lia_ovgu_01*, *Quad_Lia_ovgu_02#0* and *Quad_Lia_ovgu_03#1*.
However you don't have to worry about the naming scheme, just copy-paste the quadrocopters, and make sure that the base-name matches the name of the corresponding quadrocopter in XML file. All other stuff is managed automatically by the class *VrepObjectName* in the project *JavaVrep*.

5.2 Create V-REP virtual quadrocopters

The virtual V-REP quadrocopters are not coupled with a real flying quadrocopters. They just fly in the scene, but are visible in the Paparazzi Ground Station.
In order to create a pure virtual quadrocopters in V-REP, the name should start with the prefix *Virtual*. The name again must match an aircraft in the `PAPARAZZI_HOME/conf/conf.xml` file. There could be created multiple instances of virtual quadrocopters by just copy-pasting. The V-REP naming scheme with the *base_name#index* also takes place.

5.3 Start the communication link

- Start the V-REP simulation
- Start the FinkenSimBridge
- Press the *"Connect"* button. If the connection is established successfully, the button label will be changed to *"Disconnect"*.

6 Where to start developing?

If you want to change or further develop the Java API it is recommended to start from the *FinkenSimBridgeController*. There you can trace all events, that take place in the application.
Your SwiffKnife by debugging the Ivy-Bus is the Ivy-Bus Probe. You can use the Ivy-Bus Probe to monitor all the messages on the Ivy-Bus and see if you get messages from the quadrocopters.

7 Further reading

- JavaDoc in `GITHUB_HOME/Simulation/Java_Workspace/doc/index.html`
- V-REP Remote API
- Java V-REP Client
- Java Ivy-Bus library