

리눅스에서의 rosjava 설치방법과 사용법.

1. 준비

- ros 설치
- eclipse 설치

2. rosjava 설치

The screenshot shows the ROS Wiki page for rosjava installation. The browser address bar shows the URL: wiki.ros.org/rosjava/Tutorials/kinetic/Installation. The page title is "rosjava/ Tutorials/ kinetic/ Installation". A blue banner at the top says: "Please ask about problems and questions regarding this tutorial on answers.ros.org. Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags." The main heading is "Installation". Below it, the "Description" is "Installation instructions from debs, sources, or maven." The "Keywords" are "rosjava installation". The "Tutorial Level" is "BEGINNER". The "Next Tutorial" is "Creating Rosjava Packages". The section "1. Installation Methods" follows, with a paragraph explaining that there are three ways to install rosjava. A red box highlights the "Deb Installation" bullet point: "Deb Installation - install rosjava debs and overlay your rosjava/android source workspace on top of these." The other two methods are "Source Installation" and "No Ros Installation". The footer includes the Creative Commons Attribution 3.0 license and the Open Source Robotics Foundation logo.

rosjava/ Tutorials/ kinetic/ Installation

Please ask about problems and questions regarding this tutorial on answers.ros.org. Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Installation

Description: Installation instructions from debs, sources, or maven.

Keywords: rosjava installation

Tutorial Level: BEGINNER

Next Tutorial: [Creating Rosjava Packages](#)

1. Installation Methods

There are three different ways to install rosjava. All three give you access to a maven repository of rosjava artifacts (jars), but each differs by where and how those repositories are accessed. In addition, the ros source and deb installations give you the full ros package environment which allows you to utilise ros-specific files in your package (e.g. launchers).

- [Source Installation](#) - a mostly self-contained source installation of core components and messages for a working catkin-rosjava development environment.
- [Deb Installation](#) - install rosjava debs and overlay your rosjava/android source workspace on top of these.
- [No Ros Installation](#) - use the rosjava/message libraries in your own development environment via maven (no catkin-ros necessary!).

Except where otherwise noted, the ROS wiki is licensed under the [Creative Commons Attribution 3.0](#) | [Find us on Google+](#)

Wiki: rosjava/Tutorials/kinetic/Installation (last edited 2016-12-24 19:52:49 by jceruti)

Brought to you by: Open Source Robotics Foundation

-rosjava/./installation 페이지에서 deb installation을 선택한다.

rosjava/Tutorials/kinetic/Deb Installation - ROS Wiki - Mozilla Firefox

rosjava/Tutorials/kinetic/Deb Installation

Contents

1. Overview
2. Debs
3. Your Own Sources
4. Disabling GenJava in Non-Java Workspaces
5. Where to Now?

UPDATE IN PROGRESS: In case of trouble please refer to the indigo version.

1. Overview

The following instructions show how to build your own source workspace on top of a deb installation of rosjava.

2. Debs

All the rosjava debs are provided by a single rosjava metapackage:

```
> sudo apt-get install ros-kinetic-rosjava
```

3. Your Own Sources

You can now proceed to add your own source packages on top by creating an overlay on top of /opt/ros/kinetic (you will need to modify the URL for wstool appropriately first):

```
> mkdir -p ~/myjava/src
> cd ~/myjava/src
> wstool init -j4 https://raw.githubusercontent.com/<USERNAME>/rosinstalls/kinetic-devel/myjava.rosinstall
> source /opt/ros/kinetic/setup.bash
> cd ~/myjava
> rosdep update
> rosdep install --from-paths src -i -y
> catkin make
```

Wiki

- Distributions
- ROS/Installation
- ROS/Tutorials
- RecentChanges
- Deb Installation

Page

- Immutable Page
- Info
- Attachments
- More Actions: [v]

User

- Login

- 터미널에 `sudo apt-get install ros-<version name>-rosjava`를 입력.

```
tj@tj-dev: ~
tj@tj-dev:~$ sudo apt-get install ros-kinetic-rosjava

Selecting previously unselected package ros-kinetic-rosjava.
Preparing to unpack .../ros-kinetic-rosjava_0.3.0-0xenial-20170727-210812-0800_a
md64.deb ...
Unpacking ros-kinetic-rosjava (0.3.0-0xenial-20170727-210812-0800) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up default-jre-headless (2:1.8-56ubuntu2) ...
Setting up ant (1.9.6-1ubuntu1) ...
Setting up ant-optional (1.9.6-1ubuntu1) ...
Setting up default-jre (2:1.8-56ubuntu2) ...
Setting up default-jdk-headless (2:1.8-56ubuntu2) ...
Setting up default-jdk (2:1.8-56ubuntu2) ...
Setting up ros-kinetic-rosjava-build-tools (0.3.2-0xenial-20170217-225417-0800)
...
Setting up ros-kinetic-rosjava-bootstrap (0.3.2-0xenial-20170509-084359-0800) ..
.
Setting up ros-kinetic-genjava (0.3.3-0xenial-20170509-084757-0800) ...
Setting up ros-kinetic-rosjava-core (0.3.5-0xenial-20170727-184858-0800) ...
Setting up ros-kinetic-rosjava-extras (0.3.3-0xenial-20170727-210214-0800) ...
Setting up ros-kinetic-rosjava-messages (0.3.0-0xenial-20170727-184246-0800) ...
Setting up ros-kinetic-rosjava-test-msgs (0.3.0-0xenial-20170613-161643-0800) ..
.
Setting up ros-kinetic-zeroconf-jmdns-suite (0.3.0-0xenial-20170509-085616-0800)
...
Setting up ros-kinetic-rosjava (0.3.0-0xenial-20170727-210812-0800) ...
```

-설치가 끝났다면 위와 같이 출력이 된다.

3. rosjava 작업 환경 생성.

```
tj@tj-dev:~/rosjava$ cd
tj@tj-dev:~$ ls
바탕화면  Documents      examples.desktop  Public  Videos
catkin_ws Downloads      Music             skku    ws
Desktop  eclipse-workspace Pictures          Templates
tj@tj-dev:~$ cd ../../
tj@tj-dev:/$ source opt/ros/kinetic/setup.bash
tj@tj-dev:/$ cd
tj@tj-dev:~$ mkdir -p rosjava/src
tj@tj-dev:~$ ^C
tj@tj-dev:~$ cd rosjava/
tj@tj-dev:~/rosjava$ cd src/
tj@tj-dev:~/rosjava/src$ catkin_init_workspace
Creating symlink "/home/tj/rosjava/src/CMakeLists.txt" pointing to "/opt/ros/kinetic/share/catkin/cmake/toplevel.cmake"
tj@tj-dev:~/rosjava/src$ cd ..
tj@tj-dev:~/rosjava$ catkin_make

tj@tj-dev: ~/rosjava
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/tj/rosjava/build/test_results
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.6
-- BUILD_SHARED_LIBS is on
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tj/rosjava/build
####
#### Running command: "make -j4 -l4" in "/home/tj/rosjava/build"
####
tj@tj-dev:~/rosjava$
```

- ros의 명령어를 사용하기 위해서
source opt/ros/<version name>/setup.bash를 터미널에 입력을 한다.
- mkdir -p rosjava/src 입력 하여 디렉토리를 생성 한다.
- /src에서 catkin_init_workspace를 입력하면 workspace를 초기화하기 위한 파일들이 생성된다.
- /rosjava에서 catkin_make를 입력하면 위의 이미지처럼 출력된다.

4. rosjava workspace 생성.

[illegible]

- /rojava에서 `mkdir -p <workspace명>/src`를 입력하여 디렉토리를 생성한다.
- ../src에서 `catkin_create_rojava_pkg` 명령어를 사용하여 rojava package를 생성한다.

```

tj@tj-dev: ~/rosjava/rosjava_ws_skku
tj@tj-dev:~/rosjava/rosjava_ws_skku/src$ cd ..
tj@tj-dev:~/rosjava/rosjava_ws_skku$ ls
src
tj@tj-dev:~/rosjava/rosjava_ws_skku$ catkin_make

tj@tj-dev: ~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/tj/rosjava/rosjava_ws_skku/build/test_re
sults
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.6
-- BUILD_SHARED_LIBS is on
-- ~~~~
-- ~~~~ traversing 1 packages in topological order:
-- ~~~~ - rosjava_pkg_skku
-- ~~~~
-- +++ processing catkin package: 'rosjava_pkg_skku'
-- ==> add_subdirectory(rosjava_pkg_skku)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tj/rosjava/rosjava_ws_skku/build
####
#### Running command: "make -j4 -l4" in "/home/tj/rosjava/rosjava_ws_skku/build"
####
tj@tj-dev:~/rosjava/rosjava_ws_skku$ cd src/rosjava_pkg_skku/
tj@tj-dev:~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku$ ls
build.gradle  gradle  gradlew.bat  settings.gradle
CMakeLists.txt  gradlew  package.xml
tj@tj-dev:~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku$

```

- 다시 workspace 경로 돌아와서 catkin_make를 입력한다.
- 이 후에 build작업은 대부분 이 경로에서 catkin_make를 입력하여
진행을 한다. 위의 이미지처럼 디렉토리 내에 여러 파일들이 생성된다.

```

tj@tj-dev: ~/rosjava/rosjava_ws_skku
tj@tj-dev:~/rosjava/rosjava_ws_skku$ ls
build  devel  src
tj@tj-dev:~/rosjava/rosjava_ws_skku$ source devel/setup.bash
tj@tj-dev:~/rosjava/rosjava_ws_skku$ cd
tj@tj-dev:~$ roscd
tj@tj-dev:~/rosjava/rosjava_ws_skku/devel$ cd ..
tj@tj-dev:~/rosjava/rosjava_ws_skku$

```

- workspace의 경로로 쉽게 돌아오기 위해 setup.bash를 불러오면
roscd를 입력했을 때 쉽게 경로로 들어올수 있다.

5. 프로젝트 생성.

```
tj@tj-dev: ~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku
tj@tj-dev:~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku$ catkin_create_rosjava_
library_project jam

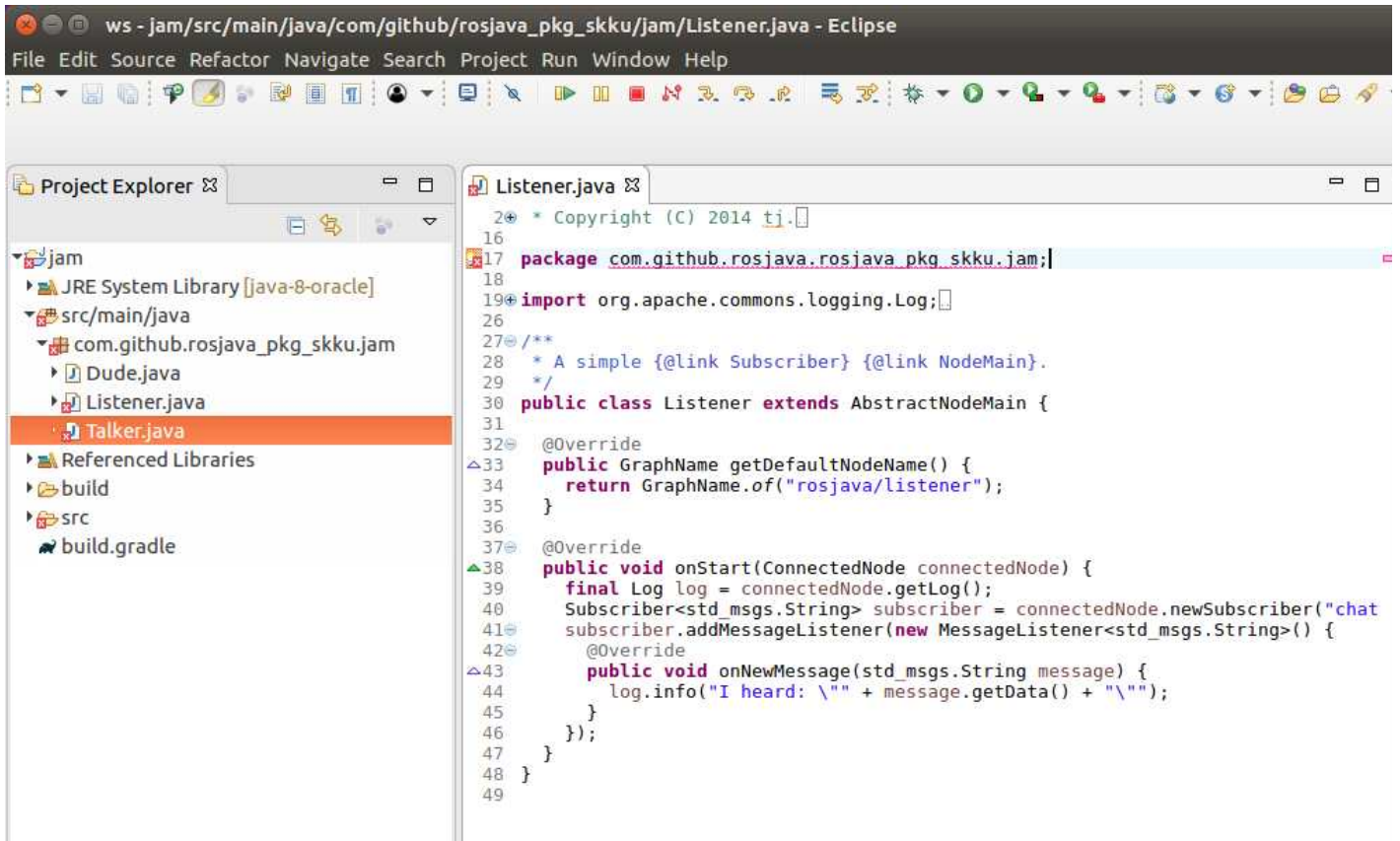
Creating rosjava project
Name      : jam
File      : build.gradle
File      : settings.gradle
File      : Dude.class
File      : CMakeLists.txt (gradle task update)
tj@tj-dev:~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku$ catkin_create_rosjava_
project jam

Creating rosjava project
Name      : jam
File      : build.gradle
File      : settings.gradle
File      : Talker.java
File      : Listener.java
File      : CMakeLists.txt (gradle task update)
tj@tj-dev:~/rosjava/rosjava_ws_skku/src/rosjava_pkg_skku$
```

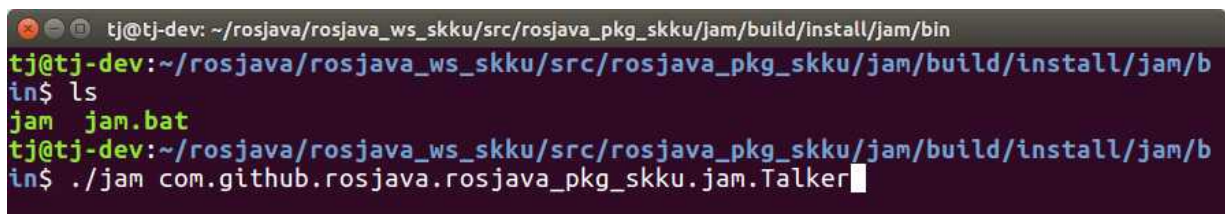
- 패키지 경로 내에서 catkin_create_rosjava_library_project 명령어와 catkin_create_rosjava_project 명령어 뒤에 프로젝트 이름과 함께 입력을 한다.
- library_project를 생성해야 다른 라이브러리를 import 할 수 있다.
- 빌드 후에는 Talker와 Listener, Dude를 지우고 새로운 Node클래스를 추가하여 작업을 할 수 있다.(이 후에 소개)

```
tj@tj-dev: ~/rosjava/rosjava_ws_skku
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/tj/rosjava/rosjava_ws_skku/build/test_re
sults
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.6
-- BUILD_SHARED_LIBS is on
-- 
-- traversing 1 packages in topological order:
--   - rosjava_pkg_skku
-- 
-- +++ processing catkin package: 'rosjava_pkg_skku'
-- ==> add_subdirectory(rosjava_pkg_skku)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/tj/rosjava/rosjava_ws_skku/build
####
#### Running command: "make -j4 -l4" in "/home/tj/rosjava/rosjava_ws_skku/build"
####
[100%] Gradling tasks for rosjava_pkg_skku
Could not find metadata com.github.rosjava.rosjava_pkg_skku:jam/maven-metadata.x
ml in remote (file:/home/tj/rosjava/rosjava_ws_skku/devel/share/maven/)
[100%] Built target gradle-rosjava_pkg_skku
tj@tj-dev:~/rosjava/rosjava_ws_skku$
```

- workspace로 돌아와 catkin_make를 입력하면 위처럼 빌드가 된다.



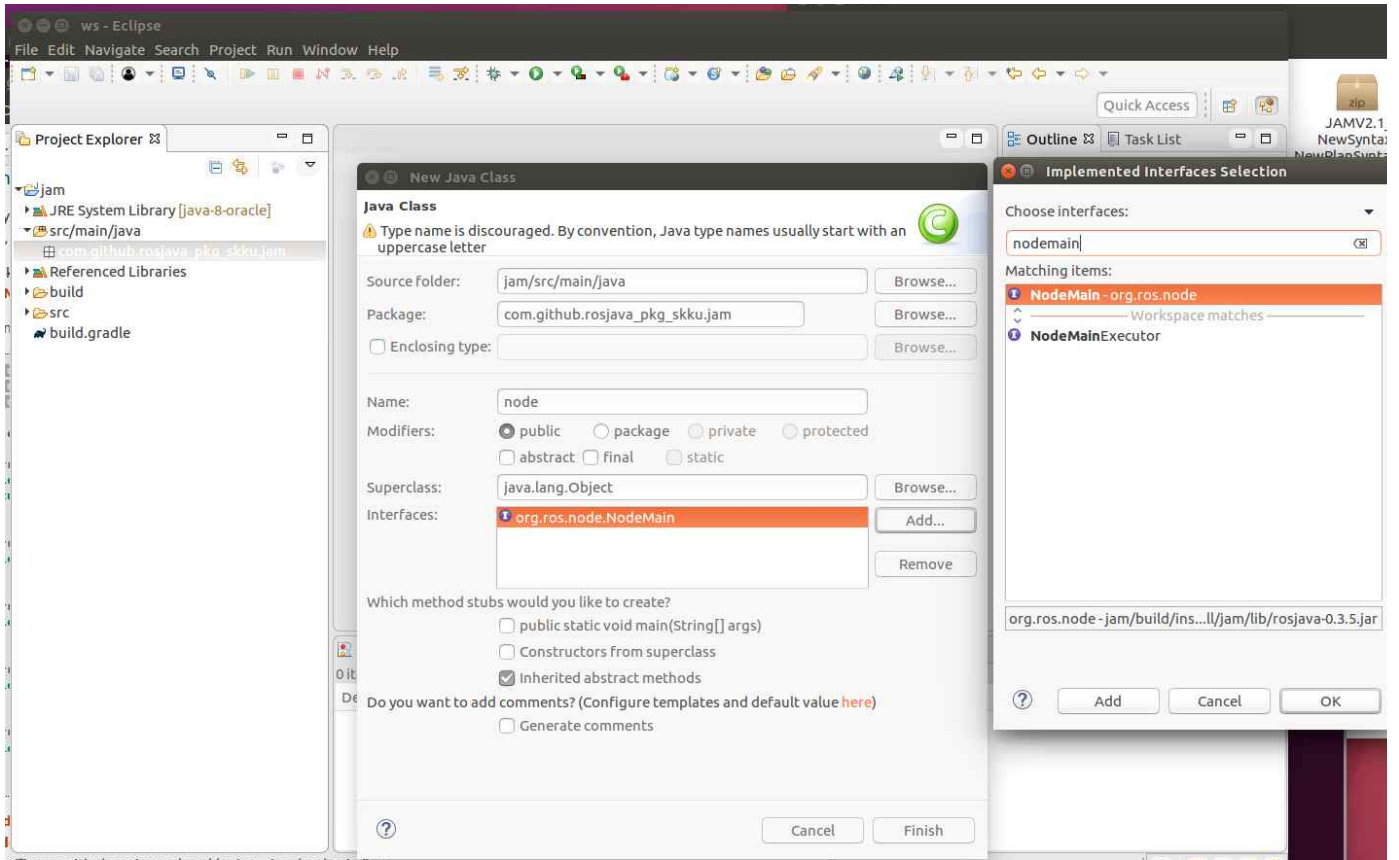
-이클립스에서 해당 프로젝트를 import 하면 위처럼 추가된다.



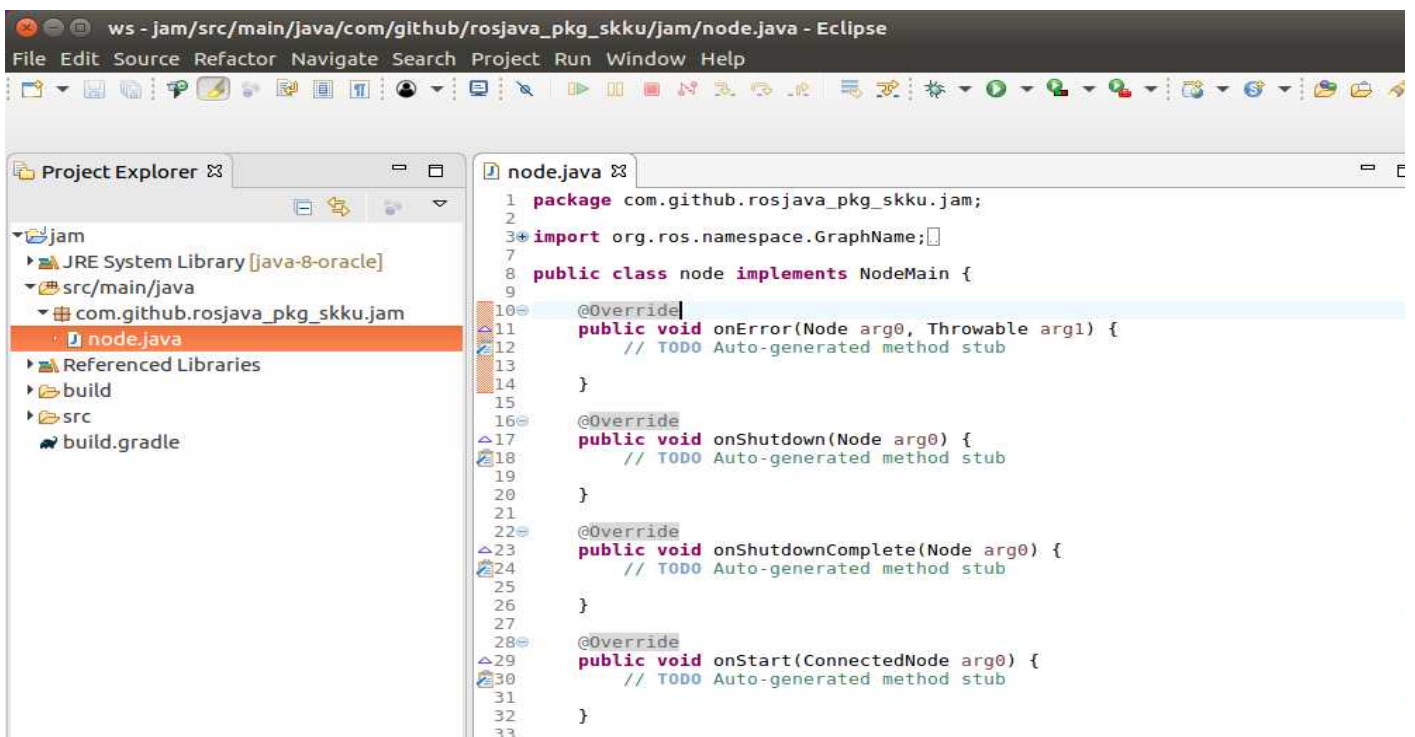
-빌드된 프로그램의 실행은 위처럼 입력을 하면 된다.

./<프로그램명> <패키지.클래스명>

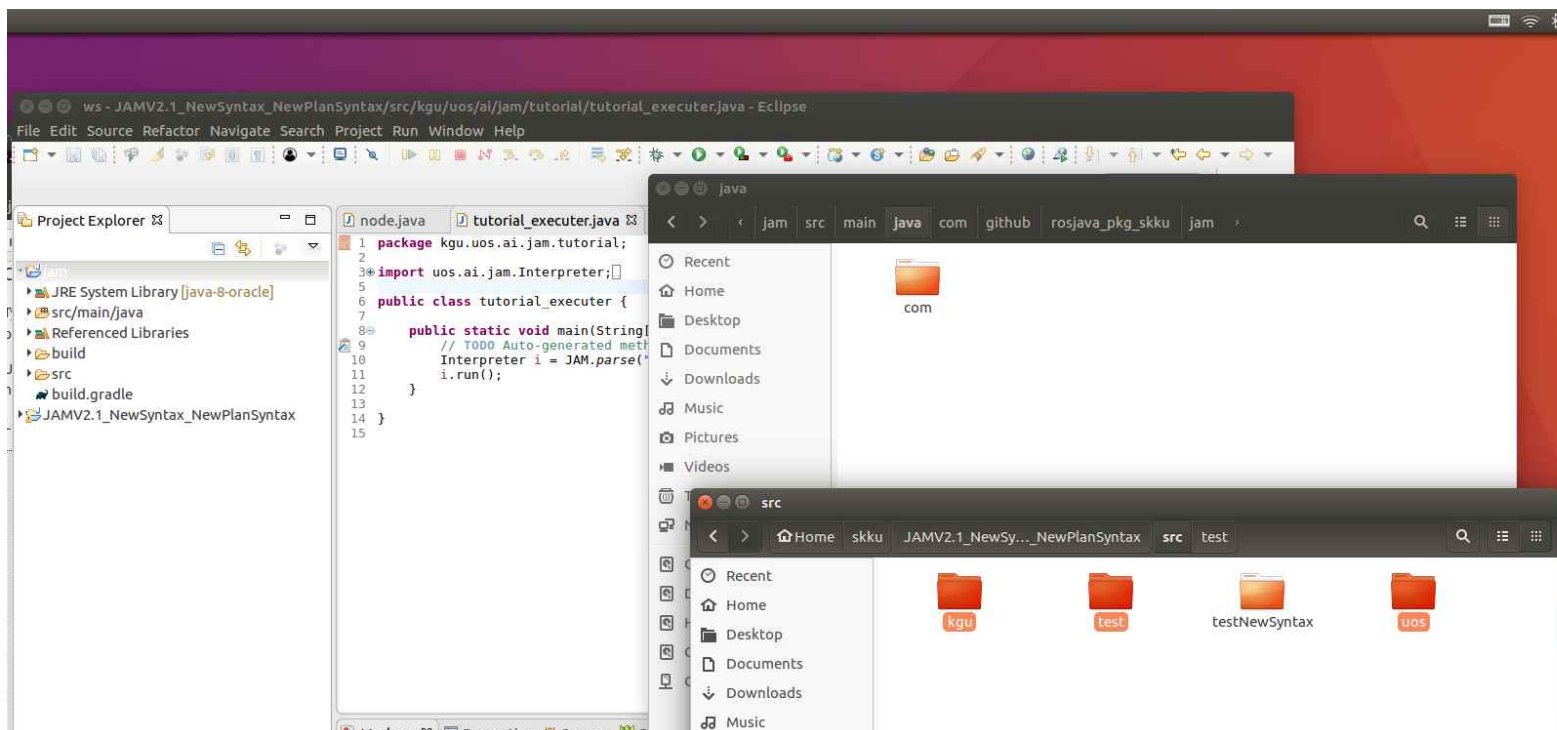
6. eclipse에서 작업.



-예제 클래스인 Dude, Talker, Listener를 삭제하고 클래스 추가를 실행하여 위의 이미지처럼 NodeMain의 interface를 상속받는다.



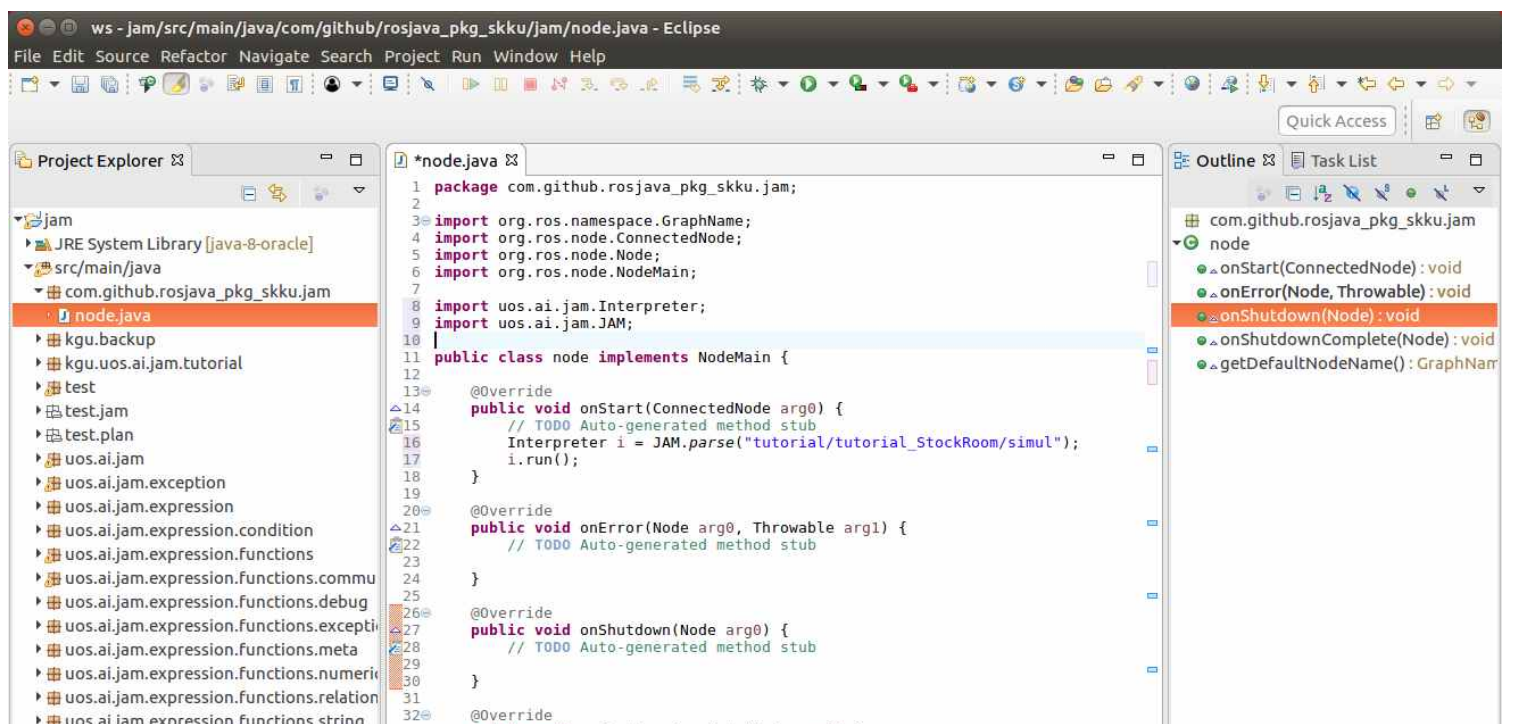
7. 프로젝트에 JAM 추가



- 프로젝트명/src/main/java/ 의 경로에 jam의 소스가 포함된 kgu,test,uos 폴더들을 복사한다.

- eclipse의 프로젝트를 새로고침하면 jam이 추가된 것을 볼 수 있고 아래의 이미지처럼 node클래스의 onStart 메소드에 jam에서 실행했던 interpreter를 추가하면 jam을 실행 시킬 수 있다.

(빌드 작업 후, 빌드 된 실행파일 경로에 tutorial 파일을 옮겨놔야 튜토리얼 프로그램을 실행 할 수 있다.)



- 추가로 eclipse가 사용자 권한보다 높은 권한에서 설치되었을 때, eclipse에서 작업을 하고 catkin_make로 빌드를 실행하면 빌드 에러가 발생 하는 경우가 있다. 이때 sudo catkin_make를 입력하면 해결이 된다.