Development in Eclipse

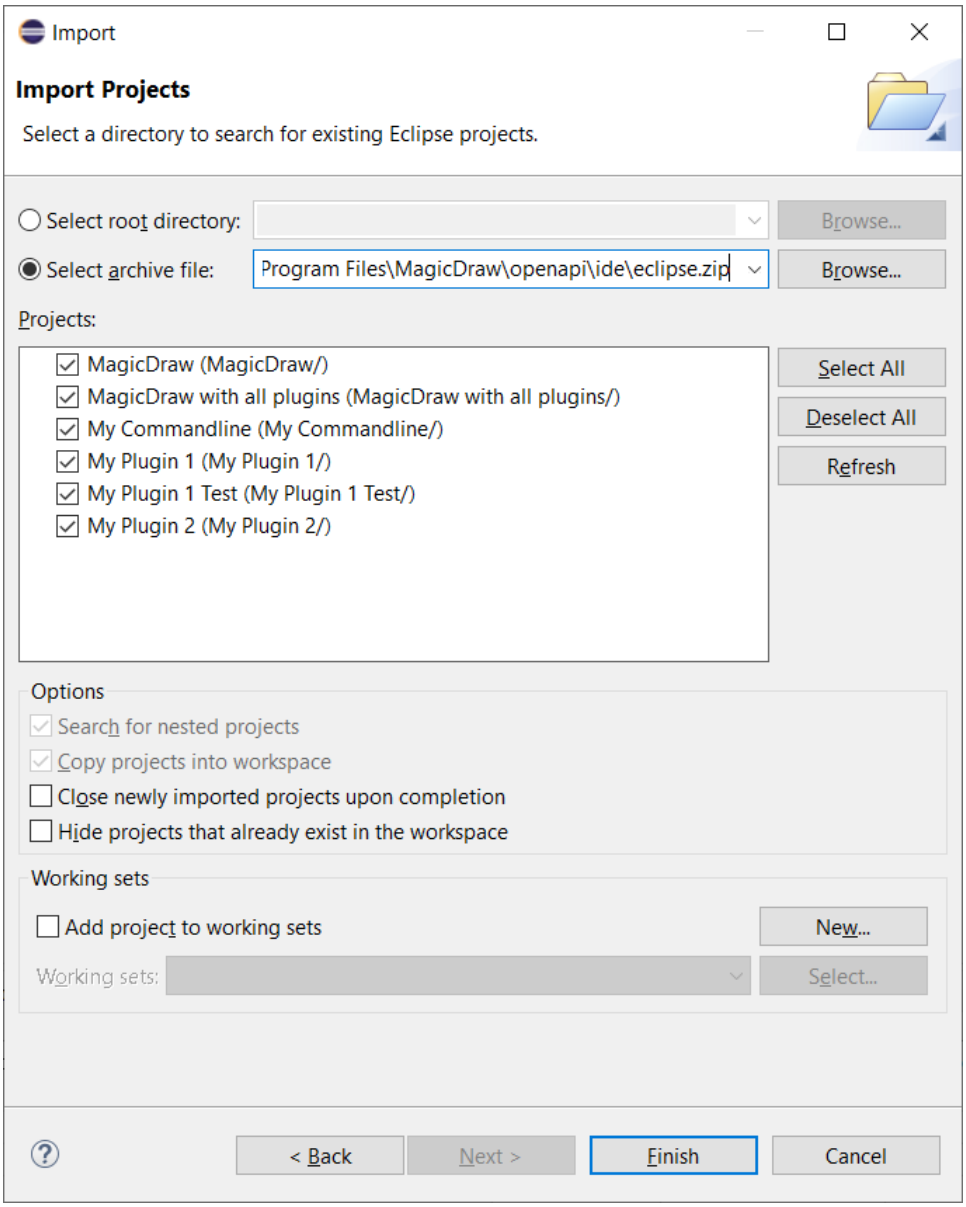
The preconfigured Eclipse projects for two sample plugins and a batch mode (command-line) tool are provided with a program installation. These projects can be found in *<modeling tool installation directory>/openapi/ide/eclipse.zip*.

Let's use MagicDraw as an example to describe the following procedures.

To setup the Eclipse environment for the modeling tool development

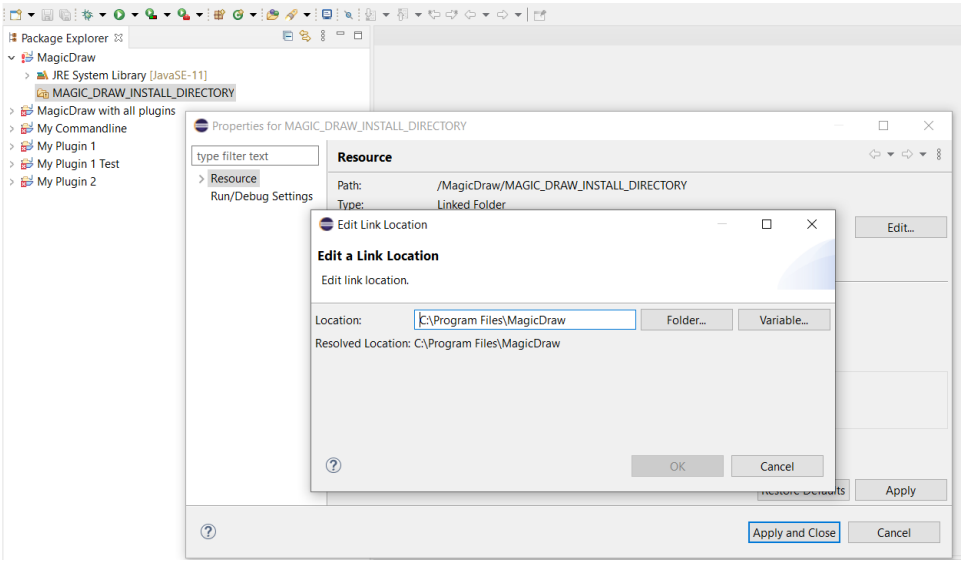
---

1. Import java projects from *<modeling tool installation directory>/openapi/ide/eclipse.zip* into your Eclipse Workspace:
   1.1. In the main menu, click **File** > **Import**. The **Import** dialog opens.
   1.2. In the **Select an import wizard** list, select **General** > **Existing Projects into Workspace** and click **Next**.
   1.3. Copy the java projects zip location to the **Select archive file** box or click **Browse** and browse to the zip file. The project list appears in the **Projects** box.
   1.4. Select projects you need and click **Finish.**



2. Open the **Package Explorer** view, expand the MagicDraw project and edit the MAGIC_DRAW_INSTALL_DIRECTORY link:
   2.1. Right-click the link and select **Properties**.
   2.2. On the left side of the opened dialog, select **Resource** and click the **Edit** button on the right side of the dialog. The **Edit Link Location** dialog opens.
   2.3. Select the location and click **OK**.

Be sure the link points to your MagicDraw (or other according to a modeling tool you are using) installation directory (see the following figure).



3. Make sure that precisely Java 11 version is used by the **MagicDraw** project in your Eclipse IDE:
   3.1. Right-click the **MagicDraw** project and select **Properties**.
   3.2. On the left side of the open dialog, select **Java Build Path** and then select **Libraries** tab on the right.
   3.3. Double-click **JRE System Library [JavaSE-11]**.
   3.4. Click **Installed JREs** and check if Java 11 is available here.

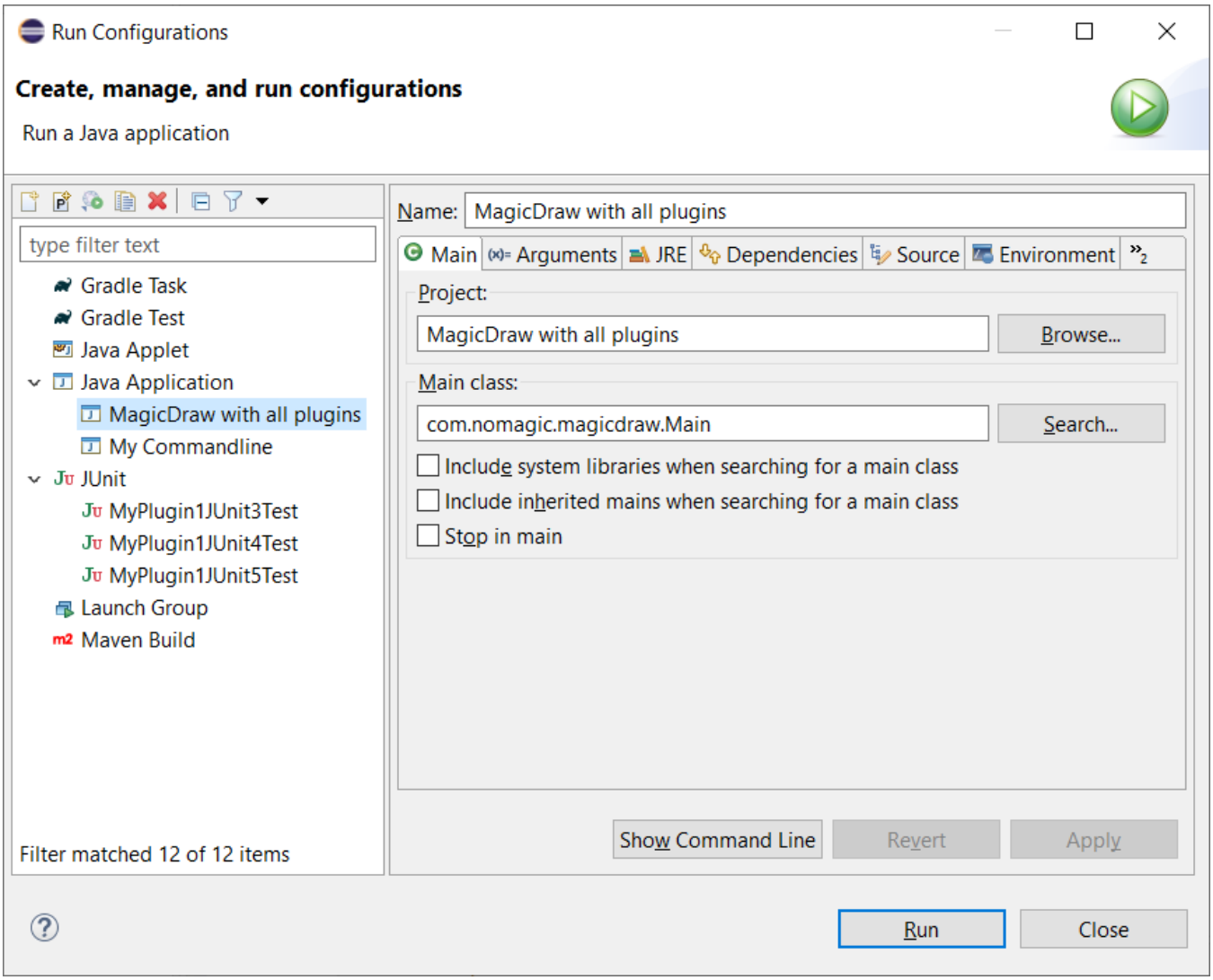
   If there is no Java 11, add it. *Java 11 version must be added, because later versions (e.g. 17) are not supported by our tool.*

   3.5. Click **Apply and Close**.
   3.6. Select **Execution environment** check box and in the combo box, select **JavaSE-11**.
   3.7. Click **Environments** button.
   3.8. In the **Execution Environments** list, select **JavaSE-11** and in the **Compatible JREs** list, select Java 11.
   3.9. Click **Apply and Close** and then **Finish** buttons.
   3.10. Select **Order and Export** tab, click **Select All** button to select all check boxes.
   3.11. Click **Apply and Close**.

Eclipse Workspace is ready for the source code development and running/debugging.

To use one of the prepared launch configurations

- On the main menu, click **Run** > **Run Configurations...** (or **Debug Configurations...**). The **Run Configurations** dialog opens.



The launch configuration is designed to load plugins from the MagicDraw (or other according to a modeling tool you are using) installation directory (see step #2) and two plugins from the Eclipse Workspace. Thus, if the *md.plugins.dir* java system property is not defined (see Plugins directories), developing plugins are not loaded.

- *MagicDraw with all plugins* launches MagicDraw (or other modeling tool) with plugins available in the IDE workspace namely *My Plugin 1* and *My Plugin 2*.
- *My Commandline* launches batch mode tool *My Commandline*.
- *MyPlugin1JUnit<n>Test* launches sample *JUnit<n>* test.

The libraries (jar files) of the plugin must be added to the development class path throughout the plugin dependency hierarchy if the developing code depends on that plugin.
For example, if the code depends on plugin A; plugin A depends on plugins B and C; plugin B depends on plugin D, the libraries of all plugins (A, B, C, and D) must be added to the class path.

When you launch your own plugin, you need to add all jar files that are required by your plugin from appropriate plugins. The MagicDraw jar files can be found in *MAGIC_DRAW_INSTALL_DIRECTORY/lib* and its sub directories, whereas plugins' jar files can be found in *MAGIC_DRAW_INSTALL_DIRECTORY/plugins* and its sub directories.

Even if the plugin descriptor file contains information about the runtime plugin *.jar* file, it is not necessary to build and deploy this *.jar* file to a plugin directory while the plugin is developed under Eclipse.