IntelliJ IDEA

IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

This guide assumes you have installed Java and Gradle and have used Git to clone some IHMC projects or a repository-group workspace.

Contents

Installation

There are generally three ways to install IntelliJ:

- Manually from Download IntelliJ IDEA
- Via a package manager (snapcraft, AUR, etc.)
- The JetBrains Toolbox

On Linux, it is recommended to install to `/opt/idea` . See Linux#Manualsoftwareinstallation and below.

```
$ curl -sL https://download.jetbrains.com/idea/ideaIC-2022.1.3.tar.gz -o idea.tar.gz
$ tar -xzf idea.tar.gz
# mv idea-IC-221.5921.22/ /opt/.
# rm /opt/idea
# ln -s /opt/idea-IC-221.5921.22/ /opt/idea
# ln -s /opt/idea/bin/idea.sh /usr/local/bin/idea
```

IntelliJ is now runnable from the command line:

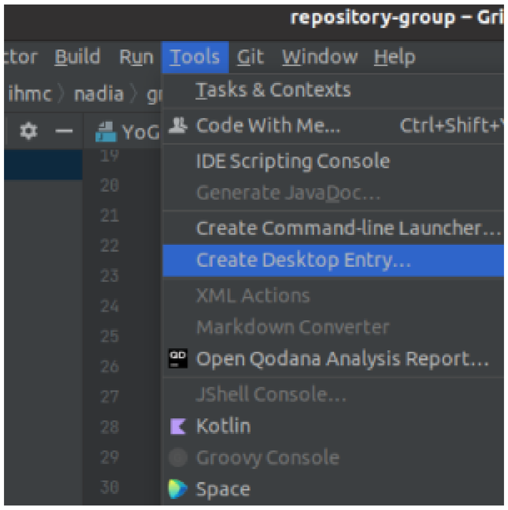
```
$ idea
```

Desktop Launcher

Create one manually or start IntelliJ manually once and it will offer to create a launcher for you.

You can use a different JDK to run IntelliJ by launching like so:
`env IDEA_JDK=/usr/lib/jvm/jdk-jetbrains /usr/bin/idea %f`

If the desktop launcher is not automatically created, you can open IntelliJ once and select Tools > Create Desktop Entry



Configuration

Searching for settings

Note: We are preferring to use text instead of screenshots in this section. Use the full settings path to the setting. You can find settings by using the search bar in the settings window. Doing things this way makes it easier to keep this documentation brief and up to date.

Quickstart

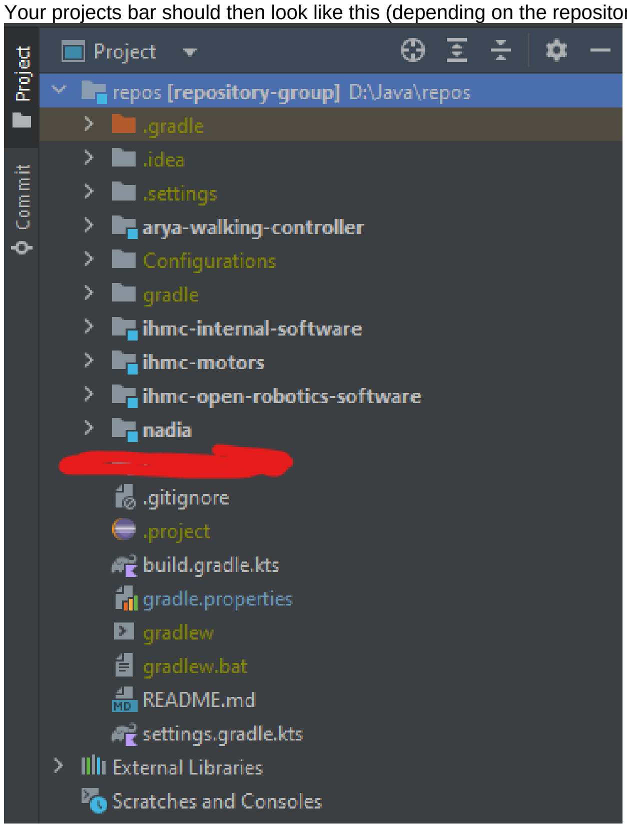
The following are the minimum settings needed to run IHMC software.

- Set the Java SDK that you downloaded.
    - Go to File > Project Structure > Project Settings > Project.
    - Set the <SDK> to Java 17
    - Set the <Language Level> to 8.

- Set the gradle version.
    - Go to File > Settings > Build, Exectution, Deployment > Build Tools > Gradle.
    - Set <Use Gradle From> and select Specified Location. Select the location where you installed gradle to.
        - On windows, this will be C:/Gradle/gradle-7.x
        - On Ubuntu, this will be /opt/gradle/gradle-7.x

        - Set <Gradle JVM> to Project SDK. This should point to the Java SDK you set in the previous step.
-

Your environment should be ready at this point. Import repository-group.

- File > Open... > select repository-group
- Gradle will take some time to build and index...
- Your projects bar should then look like this (depending on the repositories you cloned):


Project
Commit
Project
repos [repository-group] D:\Java\repos
.gradle
.idea
.settings
arya-walking-controller
Configurations
gradle
ihmc-internal-software
ihmc-motors
ihmc-open-robotics-software
nadia
[redacted]
.gitignore
.project
build.gradle.kts
gradle.properties
gradlew
gradlew.bat
README.md
settings.gradle.kts
External Libraries
Scratches and Consoles

- where all repositories have a blue square on the folder icon indicating they were recognized by gradle as modules.

Increase heap size

In order for IntelliJ to perform well with large projects such as IHMC's, it may require an increase from the default. In the Welcome launcher, click Configure > Edit Custom VM Options...

```
-Xmx2g
```

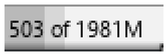
The values above are just an example. Scale them to fit your needs. Go here for instructions about Intellij settings https://intellij-support.jetbrains.com/hc/en-us/articles/206544869-Configuring-JVM-options-and-platform-properties

Potential configuration methods

- Option A: Use "Configure > Edit Custom VM Options..." in the "Welcome to IntelliJ IDEA" dialog
- Option B: Use the Jetbrains Toolbox. It has menus for JVM option configuration.

Official documentation: https://www.jetbrains.com/help/idea/increasing-memory-heap.html

Show memory indicator

Set "Show memory indicator" to confirm your settings are working. It looks like this in the bottom right corner: 

503 of 1981M

If you can't find the setting, right click on the status bar on the bottom of the screen.

Configure installed JDKs

Make sure IntelliJ detected your installed JDKs correctly. In "File>ProjectStructure>Platform Settings > SDKs" you should see JDKs. Check the Classpath and Sourcepath tabs for each one to confirm IntelliJ has picked them up properly.

Select your Project SDK (used by gradle) under Project Settings > Project > Project SDK

Fixing mixing JDK jars and missing JDK and JavaFX source attachments

In "Platform Settings > SDKs", with the problematic JDK selected, click the "-" sign to remove it. Then click add and navigate to the Java home for that specific JDK installation. IntelliJ should search and pick up everything.

Disable automatically optimizing imports

In Settings > Editor > General > Auto Import > Java, clear "Optimize imports on the fly".

Code Folding

In Settings > Editor > General > Code Folding

- Clear "One-line methods"

Code formatter

1. Locate the latest code style in `ihmc-open-robotics-software/ihmc-java-toolkit/CodeFormatTemplates/IHMCIntelliJFormatter.xml`
   If you don't have it cloned, download the IHMC IntelliJ Formatter XML file. Upon following that link, right click "Raw file" and "Save as..." to download it.
2. In "Settings > Editor > Code Style > Scheme: " drop down the gear icon and select "Import Scheme > IntelliJ code style XML".
3. (Optional) In "Code Style > Java" on the "Imports" tab, clear "Insert imports for inner classes"

Import Help

In Settings > Editor >Code Style > Java

- In the "Packages to Use Import with '*'" section, add an entry:
  - Set "Static"
  - Set "Package" to `org.junit.jupiter.api.Assertions`
  - Clear "With Subpackages"
- Add more entries for
  - (static) `org.hamcrest.Matchers`
  - (static) `org.hamcrest.MatcherAssert`

Disable spelling inspections

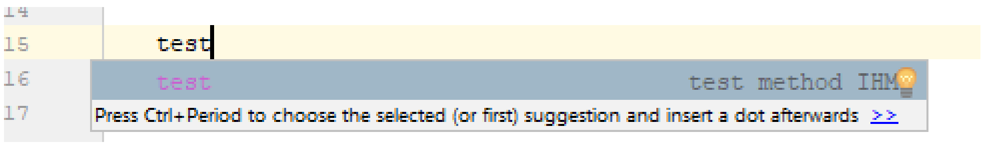
In Settings > Editor > Inspections

- Set "Profile" to "Default IDE"
- Clear "Proofreading> Typo"

Live templates

This enables auto completion for common IHMC structures in code, such as unit tests and log messages. This is not necessary for the casual developer.

Example of what this looks like:


```
test
test                                   test method IHM
```
Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>

In *Settings* > *Editor* > *Live Templates*:

Create new templates with the "+" icon. It will create a new "user" section.

Live template for JUnit 5 test

Abbreviation: test
Description: JUnit 5 test
Applicable in: *Java > declaration*

Template text

```
@org.junit.jupiter.api.Test
public void test$METHOD_NAME$()
{
   $END$
}
```

Live template for Log Tools

Tip: You can use copy and paste on the items to help create these.

Abbreviations: error, warn, info, debug, trace
Descriptions: (not necessary)
Applicable in: *Java* > *Statement*

Template text

```
us.ihmc.log.LogTools.error($END$);
us.ihmc.log.LogTools.warn($END$);
us.ihmc.log.LogTools.info($END$);
us.ihmc.log.LogTools.debug($END$);
us.ihmc.log.LogTools.trace($END$);
```

Disable Inlay Hints

In Settings > Editor > Inlay Hints

- Clear all the boxes

Java build and compiler settings

Build automatically

In "Settings > Build, Execution, Deployment > Compiler":

1. Set "Build project automatically" - This is the expected behavior when coming from Eclipse. As you change code, IntelliJ will build it. Otherwise you will not catch errors until you try to run.
2. (Experimental) Set "Compile independent modules in parallel"
3. Set "Rebuild module on dependency change"
4. (Experimental) Set "Build process heap size (Mbytes)" to 1500. This may speed up compiling if you've got RAM to spare

Making sure bytecode version is set to 8

"Settings > Build, Execution, Deployment > Compiler > Java Compiler":

1. Set "Project bytecode version" to 8.
   We are currently still using bytecode version 8 for now.

Make sure you're using the javac compiler

Eclipse compiler doesn't seem to work anymore with our setup. Use javac instead.

In "Settings > Build, Execution, Deployment > Compiler > Java Compiler":

1. Set "User compiler:" to javac.

Useful keyboard shortcuts

Manage keyboard shortcuts in "Settings > Keymap".

Using Alt+Left and Alt+Right to move tabs

IntelliJ has a some pretty unusual shortcuts for moving tabs around.

"Main menu > Window > Editor Tabs > Split and Move Right" - You can set this to `Alt+Right` to move the current tab right to (but also create, unfortunately) a new editor column.

"Other > Tabs > Move to Opposite Group" - You can set this to `Alt+Left` to move the current tab to the other side, whichever one that is.

Note: `Alt+Left` and `Alt+Right` are by default bound to the "Back" and "Forward" bindings, so you should remove them if you bind the above.

Tool window activation

It is suggested to map `Alt-#` shortcuts to your most commonly use tool windows. Settings > Keymap > Tool Windows. For us, this means adding shortcuts for "Gradle", "Build", and "Problems" and removing "TODO".

Swap Bindings for Rearranging methods (for those coming from Eclipse)

In Eclipse, we use Alt+Up and Alt+Down to move lines up and down, which is typically more useful than moving whole statements as IntelliJ does weird stuff with that.

So we unbind the following to allow the regular Eclipse keybinding to work. Make sure to remove their Alt+Up and Alt+Down bindings.

Keymap > Main menu > Code > Move Statement Up/Down:

- Ctrl+Shift+Up
- Ctrl+Shift+Down

Parameter Info

Parameter info shows the different options available to pass to a constructor or method.

Keymap > Main Menu > View > Parameter Info

`F3` is suggested as it's right next to `F2` which shows Javadoc.

Editor Tabs

It is useful to apply the following setting in "Settings > Editor > General > Editor Tabs":

- Clear "Show tabs in one row" - This prevents the usability issue (and sometimes a bug) where tabs can be open but hard to find.
- Clear "Show file extension" - Saves space in your tab area
- Set "Mark modified (*)" - Makes it clear when IntelliJ has not saved
- Set "Close button position" to "None". - Middle mouse click closes the tab. This setting prevent accidental closes and saves space.
- Set "Tab limit" to 20
- Set "When the current tab is closed, activate" to "Most recently opened tab"

Appearance & Behavior

In Appearance & Behavior > System Settings, it is useful to:

- Clear "Reopen last project on startup" (This reduces frustration of taking a long time to accidentally load a project you don't need.)

Run configuration

Defaults

You can set the defaults in "Run > Edit Configurations... > "Add new"/Templates/"+" > Application".

Click under "Modify options" if any below options do not appear.

"Working directory:" - It is hard to say what is the best setting for this. Mostly just be aware of it. Set to your build root or `$MODULE_DIR$`.

"Shorten command line:" - This setting is useful for Windows users who get "path too long" errors. To fix, set to **JAR manifest**. On Linux and Mac, leave set to **none**.

"Before launch:" - If using the Eclipse compiler, you can replace "Build" with "Build, no error check" in order to run with unimportant errors.

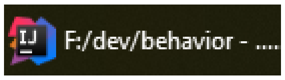
Increasing the default remembered Run Configurations past 5

In "Run > Edit Configurations... > Templates", set "Temporary configurations limit:" to a higher number like 20.

Recommended plugins

https://github.com/anthraxx/intellij-awesome-console - Fixes code links to be Eclipse shortened style (making it compatible with LogTools) creates links to all paths, even files and stuff in your console. One click to edit IHMCNetworkParameters.ini, for example.

Improve Window Title for Repository Groups

To make the text on the title bar more useful, there is a plugin that can change it with code. As in: 

1. Install https://plugins.jetbrains.com/plugin/9681-custom-title
2. Use the following code:

```
<% if (projectDefaultTitle) { %><%= projectPath %><% } %>
```

Saving Tool Window Layouts

The following plugins are useful for maintaining window layouts like Eclipse users are used to. You can use both of them at the same time as they compliment each other.

You can find these under "Marketplace" > search bar.

https://plugins.jetbrains.com/plugin/13005-window-layout-manager

https://plugins.jetbrains.com/plugin/10097-preserve-layout-plugin

Discussion on the issue here: https://youtrack.jetbrains.com/issue/IDEA-93140

Importing Gradle Projects

Importing Gradle projects into IntelliJ is very similar to running Gradle on the command line. When you import a Gradle project you are entering the Gradle Build Environment. When you import a Gradle project, IntelliJ is analyzing the Java build structure that Gradle comes up with and copying that configuration as if you had painstakingly configured IntelliJ's Project Structure window yourself. Learn more in JetBrains official documentation, although it seems relatively out of date.

This guide assumes you have read Gradle.

Steps

1. In IntelliJ, select "Open..."
   Note: Do not select "New Project" or attempt to set up the project manually.
   A Gradle project must be opened and it will set up the classpath and modules.
2. Select the directory containing your root `build.gradle` or the file itself.
3. Unfortunately, IntelliJ might not offer to allow you to set the Gradle settings before the first build, so it will try with the defaults and fail.
4. In "Settings > Build, Execution, and Deployment > Build Tools > Gradle", apply the setting described in the next section "Required Gradle Settings"
   a. Tip: You can also access the settings via the wrench icon in the Gradle view.
5. Use the "Gradle" view to manage and refresh your Gradle build over time

Required Gradle Settings

- Clear "Generate *.iml files for modules imported from Gradle" (we do not mix Gradle and IntelliJ IDEA modules)
- Clear "Automatically import this project on changes in build script files" (refresh takes too long and most of the time will be accidentally triggered, freezing up your IDE)
- Set "Build and run using" to "IntelliJ IDEA"
- Set "Run tests using" to "IntelliJ IDEA"
- Set "Use Gradle from" to "Specified location" and point it to your local Gradle installation. This is the directory containing the the Gradle `bin/` and `lib/` folders.
- Set "Gradle JVM" to "Use Project JDK". See Configure Installed JDKs. This is what is set in "File>ProjectStructure>Platform Settings > SDKs?"
- In Settings > Build, Execution, Deployment > Build Tools:
  - Clear "Reload project after changes in the build scripts"
    In projects where the build takes a long time, this can waste time with necessary refreshes.

For some reason after Gradle refresh Java compiler settings may change and you have to check that they are set correctly.
Refer to the Java Build and Compiler Settings section.

Troubleshooting

In the "Build" view, check for errors. See Gradle#Troubleshooting for more help.

A common issue is forgetting to create or maintain your user home gradle.properties file.

Troubleshooting

Restart and invalidate caches

Sometimes IntelliJ will show build or compile errors, general glitching, or weird state. In this case, it is a good idea to not only restart IntelliJ but invalidate its internal caches as well. This might be particularly important after upgrading to a new version of IntelliJ.

To do this:

1. File > Invalidate Caches / Restart...
2. "Invalidate and Restart"

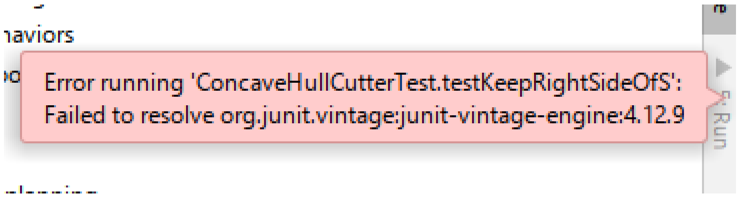
Aggressive reset

You could try deleting the .idea, .gradle, bin, and out folders throughout the repository group, or even better, just clone a new one.

You can delete recursively all files in a workspace with:

```
$ find . -type d -name DIRECTORY_NAME -prune -exec rm -rf {} \;
```

Problem running JUnit 5 tests



Error running 'ConcaveHullCutterTest.testKeepRightSideOfS':
Failed to resolve org.junit.vintage:junit-vintage-engine:4.12.9

Still waiting on solution: https://youtrack.jetbrains.com/issue/IDEA-233706

The solution was to get rid of JUnit 3 and 4 from the classpath. Their presence triggers a compatibility mode in IntelliJ.

You might also see that the test terminated without any error. This is caused by multiple versions of JUnit 5 being on the classpath at the same time.

Additional Templates

Getter template for StoredPropertySet

This template uses the velocity templating language.

```
#set( $doubleStr = "Double" )
#set( $booleanStr = "Boolean" )
#if($field.type.contains($doubleStr))
double ##
#elseif($field.type.contains($booleanStr))
boolean ##
#end
#set($name = $StringUtil.capitalizeWithJavaBeanConvention($StringUtil.sanitizeJavaIdentifier($helper.
getPropertyName($field, $project))))
  get##
${name}() {
  return get($field.name);
}
```

Setter template for Builder

TODO: This thing doesn't work well. Might need IntelliJ Ultimate. Refer here: https://www.jetbrains.com/help/idea/generating-code.html#customize-templates

```
#set( $paramName = $helper.getParamName($field, $project) )
#set( $fieldname = $field.type )
#set( $fieldnamesize = $fieldname.size() )
#set( $fieldSizeMinusOne = $fieldnamesize - 1 )
#set( $genericName = $fieldname.substring(6) )
void set$StringUtil.capitalizeWithJavaBeanConvention($StringUtil.sanitizeJavaIdentifier($helper.getPropertyName
($field, $project)))($genericName $paramName) {
   this.$field.getName()##
   .set($paramName);
}
```

See also